Annual Reviews in Control 52 (2021) 390–427

Contents lists available at ScienceDirect

Annual Reviews in Control

journal homepage: www.elsevier.com/locate/arcontrol

Review article

A survey of guidance, navigation, and control systems for autonomous multi-rotor small unmanned aerial systems

Julius A. Marshall [a], Wei Sun [b], Andrea L'Afflitto [a,*]

[a] Department of Industrial and Systems Engineering, Virginia Tech, 1145 Perry St., Blacksburg, 24060, VA, USA
[b] School of Aerospace and Mechanical Engineering, University of Oklahoma, 865 Asp Ave., Norman, 73019, OK, USA

ARTICLE INFO

MSC:
00-02
93-02

Keywords:
UAS
UAV
Quadcopter
Guidance
Navigation
Control
Survey

ABSTRACT

This survey paper presents a holistic perspective on the state-of-the-art in the design of guidance, navigation, and control systems for autonomous multi-rotor small unmanned aerial systems (sUAS). By citing more than 300 publications, this work recalls fundamental results that enabled the design of these systems, describes some of the latest advances, and compares the performance of several techniques. This paper also lists some techniques that, although already employed by different classes of mobile robots, have not been employed yet on sUAS, but may lead to satisfactory results. Furthermore, this publication highlights some limitations in the theoretical and technological solutions underlying existing guidance, navigation, and control systems for sUAS and places special emphasis on some of the most relevant gaps that hinder the integration of these three systems. In light of the surveyed results, this paper provides recommendations for macro-research areas that would improve the overall quality of autopilots for autonomous sUAS and would facilitate the transition of existing results from sUAS to larger autonomous aircraft for payload delivery and commercial transportation.

1. Introduction

Class 1 multi-rotor unmanned aerial systems (UAS), also known as small UAS (sUAS) or micro unmanned aerial vehicles (MUAVs), that is, aircraft lighter than 20lbs, flying at an altitude of less than 1,200ft, and traveling at a speed of less than 185 km/h, have drawn exceptional attention from both the research and the industrial communities. Indeed, in 2021, in the US, 493,000 sUAS have been registered for recreational purposes and 372,000 sUAS have been registered for commercial purposes. Key factors for the extraordinary success of these vehicles, which 15 years ago were substantially considered as toys, are their simple architecture, their relatively low costs, their relative safety, and the fact that they do not require particularly sophisticated or expensive infrastructures to be operated.

Although the use of multi-rotor sUAS for aerial photography is considered as established, new frontiers of this technology involve diverse applications including autonomous payload delivery, surveillance and information gathering in extended and unknown areas, indoor and outdoor infrastructure inspection by means of both contact and non-contact sensors, and air, water, soil, and crop sampling. In the future, sUAS may serve as personal assistants to workers in industrial or office environments and people affected by limited mobility problems. However, this kind of application requires the sUAS to operate with high precision despite rapidly changing environments, uncertainties in their dynamical models, external disturbances, and the fact that state-of-the-art hardware and software for sUAS does not yet guarantee these high quality standards.

Similar to other mobile robots, the software architecture of multi-rotor sUAS is substantially divided into three parts, namely guidance, navigation, and control systems. The problem of designing components for guidance, navigation, and control systems has been tackled by researchers whose academic backgrounds range from aerospace to mechanical engineering, and electrical engineering to computer science. Thus, the state-of-the-art in aerial robotics has benefited from this vast diversity of expertise and advanced at a very fast pace. However, a careful analysis of the state-of-the-art shows increasing sectorization in the study of guidance, navigation, and control systems for autonomous multi-rotor sUAS. Furthermore, it is apparent how a larger number of contributions have been produced in the design of guidance systems over navigation and control systems.

The primary goal of this literature review is to provide a holistic perspective on the state-of-the-art in the design of guidance, navigation, and control systems for autonomous multi-rotor sUAS. In particular, this paper presents fundamental results in each of these areas, and describes some of the latest advances that enabled the use of sUAS for agricultural, industrial, and military applications, which were considered as unrealistic until a few years ago. Furthermore, this paper

* Corresponding author.
E-mail address: a.lafflitto@vt.edu (A. L'Afflitto).

https://doi.org/10.1016/j.arcontrol.2021.10.013
Received 16 July 2021; Accepted 23 October 2021
Available online 25 November 2021

lists some techniques that have not been implemented yet on quadcopters due to the limitations of existing single-board computers for sUAS, but have the potential to further improve the state-of-the-art. This paper also highlights some limitations of the existing theoretical and technological solutions in the guidance, navigation, and control of multi-rotor sUAS. Finally, this survey presents relevant gaps that hinder the integration of guidance, navigation, and control systems, and highlights ongoing efforts to reduce these gaps. To meet these goals, this publication surveys seminal or the most comprehensive papers for well-established techniques and some of the most meaningful works published in the past five years for illustrative examples and technological demonstrations, giving preference to journal and international conference papers. In the case of particularly novel and interesting results, pre-prints in public repositories have been cited as well.

The intended readers for this paper are researchers, graduate students, and professionals who are competent in the design of guidance, navigation, or control systems for sUAS and would like to explore other areas. To this goal, more popular results are merely cited, whereas the fundamental ideas underlying more recent or less common results are discussed at further length. Higher priority has been given to the breadth rather than the depth of results presented. Interested readers are referred to Atyabi, MahmoudZadeh, and Nefti-Meziani (2018) for a survey on mission planning for autonomous vehicles, including sUAS, Rubí, Pérez, and Morcego (2020) for a survey of recent path following control techniques, Aggarwal and Kumar (2020) for a survey of path planning and decision-making techniques with a perspective on data sharing through communication networks, Goerzen, Kong, and Mettler (2009) for an earlier, but comprehensive survey on path planning for sUAS, Indu and Singh (2020) for a survey on trajectory planning techniques for sUAS with a perspective on inter-vehicle communication, Cabreira, Brisolara, and Ferreira (2019) for a survey on coverage of extended areas by means of sensors transported by sUAS, Zhou, Yi, Liu, Huang, and Huang (2020) for a survey on mapping and imagery, Lu, Xue, Xia, and Zhang (2018) for a survey of vision-based navigation systems, and Kim, Gadsden, and Wilkerson (2020), L'Afflitto, Anderson, and Mohammadi (2018) for surveys on control techniques for multi-rotor sUAS. A tutorial survey on trajectory planning, estimation, and control for quadcopters is provided in Tang and Kumar (2018). Recent surveys on the guidance, navigation, and control of swarms of sUAS are presented in Abdelkader, Güler, Jaleel, and Shamma (2021) and Coppola, McGuire, De Wagter, and de Croon (2020). Finally, readers interested in detailed results are referred to the literature reviews presented in each of the papers cited herein.

Machine learning and artificial intelligence have proven their effectiveness in the design of advanced navigation systems, especially for problems in which object recognition and feature extraction are essential for sUAS to make informed decisions. Quite recently, data-driven techniques have been employed also in the design of guidance and control systems and soon will prove to be particularly valuable to complement and improve existing approaches. For these reasons, our survey includes a critical review of results produced within the machine learning and artificial intelligence contexts. The interested reader is referred to Carrio, Sampedro, Rodriguez-Ramos, and Campoy (2017) and Fraga-Lamas, Ramos, Mondéjar-Guerra, and Fernández-Caramés (2019) for surveys on deep learning methods applied to sUAS.

This paper also surveys topics that, in the authors' opinion, are rarely considered in other survey papers. For instance, we dedicate considerable space to guidance systems for autonomous multi-rotor sUAS employed in tactical missions, that is, sUAS operating at a very low altitude without being detected by opponents or threats, whose location is not necessarily known. Additionally, this paper surveys in detail recent guidance systems for sUAS that need to travel as fast as possible. The reason for considering these application niches is that law enforcement agencies, emergency teams, and ground troops are increasingly adopting sUAS to support their operations and reduce the potential harm to personnel. Furthermore, we give considerable attention to the problem of generating maps from data observed by camera and LiDAR (Light Detection And Ranging) systems, and discuss at length the few available approaches to extrapolate information from a map and, hence, enable path or trajectory planners. Finally, within the context of autonomous navigation, we discuss not only camera- and LiDAR-based systems, but also SONAR-based (SOund NAvigation and Ranging) systems.

This paper is organized as follows. Section 2 presents a brief overview of current and future mission objectives for multi-rotor sUAS and the hardware generally employed to meet these objectives. Section 3 presents the equations of motion of a multi-rotor quadcopter, and highlights some key features that are exploited by their guidance, navigation, and control systems or that pose challenges to their design. Section 4 surveys guidance systems for sUAS. This section is structured by considering path planning algorithms first and then trajectory planning algorithms. Path planners are surveyed by examining classical methods first and then by discussing bio-inspired and learning-based methods. Trajectory planners are surveyed by examining receding horizon or model predictive control (MPC) approaches, which still constitute a large portion of existing techniques, numerical methods based on alternative variational approaches, methods based on the use of reference governors, and learning-based methods. Finally, Section 4 discusses popular guidance systems for swarms of sUAS, some of the most recent approaches to the problem of integrating path and trajectory planners, and guidance systems for tactical sUAS. Section 5 primarily surveys simultaneous localization and mapping (SLAM) systems for single and multiple agents, while distinguishing between visual SLAM and LiDAR SLAM. Furthermore, Section 5 discusses some navigation techniques based on SONAR and the reflection of sounds in general. Finally, Section 5 gives special emphasis to the mapping problem and highlights some techniques used to model potential collisions in the environment. Section 6 surveys control systems for multi-rotor sUAS by considering linear control techniques, methods based on the feedback-linearization of the sUAS' equations of motion, nonlinear control laws, and data-driven methods. Finally, Section 7 provides concluding remarks by discussing some open research problems such as the current lack of integration or synchronization between guidance, navigation, and control systems for multi-rotor sUAS. Section 7 also examines some of the effects of current theoretical and technological gaps in the integration of guidance, navigation, and control systems for sUAS on larger autonomous aerial robots operating in close proximity to people and recommends some future research directions.

2. Mission objectives and hardware components for quadcopters

Commercial-off-the-shelf (COTS) multi-rotor sUAS for recreational purposes are usually designed to meet the popular demand of taking high-quality aerial images, reaching multiple waypoints specified by the user on a given map, or following a reference trajectory outlined by means of a remote controller or some graphical interface. Equipped with dedicated payloads, such as high-resolution cameras in the visible or infrared spectra, quadcopters are now being employed in agricultural and industrial applications such as monitoring crops and inspecting infrastructures, such as bridges, solar farms, and wind farms. Presently, armed forces and law enforcement agencies employ multi-rotor sUAS to collect detailed intelligence on areas where some mission will be performed within a relatively short time horizon.

In the near future, sUAS will be employed for payload delivery and advanced forms of infrastructure inspection that are not limited to visual inspection, but also require a physical interaction between one or multiple onboard sensors and some surfaces, such as temperature and humidity meters or hardness testers. Future applications for quadcopters also involve autonomous surveillance missions, wherein the aircraft is not only tasked to inspect a given perimeter and flag the presence of intruders, but also to follow potential intruders in a minimally invasive manner. Significant effort is being made to employ

Table 1
Some commonly used COTS autopilots for multi-rotor sUAS. Manuals for the popular autopilot firmware PX4 and ArduPilot can be found in Ardupilot Firmware (2021) and PX4 Firmware (2021), respectively.

Autopilot	Major Components	Weight	Power consumption
AscTec	1 IMU, barometer, GPS	20 g	300 mA/6V
ModalAI Flight Core	2 IMUs, barometer, PX4	6 g	330 mA/5V
Navio2	IMU, barometer, GNSS, ArduPilot, Raspberry Pi support	23 g	150 mA/5.3V
Piccolo Nano	Laser Altimeter, DGPS, Crista IMU, 14 GPIO pins	30 g	4 W
Piccolo SL	Laser Altimeter, DGPS, Crista IMU, 14 GPIO pins	233 g	4 W
Pixhawk	2 IMUs, barometer, PX4, ArduPilot	38 g	280 mA/5V
Pixracer	2 IMUs, barometer, PX4, ArduPilot	10 g	280 mA/5V

multi-rotor sUAS as mobile hotspots integrated in future 5G networks and strategically steered in areas where increased access points and bandwidth are needed. Finally, to be employed in close proximity with people and existing infrastructures, quadcopters will need to guarantee user-defined levels of precision and robustness to faults, failures, and external disturbances without relying on larger safety margins, but on a more intelligent use of existing hardware.

Most sUAS are sufficiently agile to perform aggressive maneuvers, which involve one or multiple complete rotations or sharp turns at a high speed. However, limitations due to safety and practicality, do not require to exploit these vehicles' agility, especially for civilian use. Quadcopters are primarily demanded for their ability to rapidly reach high altitudes and span long distances in short time while reducing both the costs and the risks implied by the direct involvement of human operators performing the same tasks.

The majority of quadcopters include an autopilot, which is responsible for position and pose estimation through an inertial measurement unit (IMU), actuator control, and communications with peripherals; some autopilots are able to generate simple paths or trajectories for waypoint navigation. Table 1 lists some of the most commonly used COTS autopilots for sUAS and presents some of their key characteristics. Fig. 1 provides a schematic representation of a common architecture of the guidance, navigation, and control system on COTS autopilots for sUAS.

Multiple antennas are used for telemetry transmissions and allow users to take manual control. Multiple sensors are usually responsible for determining the position and attitude of the aircraft relative to a given reference frame. In some cases, the quadcopter hosts plug-in modules for global navigation satellite system (GNSS) integration, such as GPS (Global Positioning System). If only onboard sensors can be used by the sUAS, then its position is determined by merging the autopilot's IMU data with plug-in optical flow sensors, which exploit the ground's texture and other visible features to determine the aircraft's translational velocity. Barometers provide coarse estimates on the sUAS' altitude, whereas COTS laser altimeters usually provide reliable estimates for sUAS flying at or below 50 m of altitude. In many cases, the sUAS position is determined by exploiting one or multiple plug-in cameras in the visible or infrared spectrum, which are responsible for detecting key features in the environment; one or multiple plug-in depth cameras, which use an infrared laser projector and two imagers to determine the depth of translucent and opaque surfaces in the environment; or one or multiple plug-in LiDARs, which use light in the form of a pulsed laser to measure the depth of translucent and opaque surfaces. Similarly, the sUAS attitude relative to a given reference frame is determined by merging data produced by the gyroscope embedded in the autopilot's IMU and plug-in depth cameras, tracking cameras, or LiDARs. Table 2 lists some of the most commonly used COTS plug-in sensors employed to determine position and orientation of sUAS and presents some of their key characteristics.

Table 2
Some commonly used COTS plug-in sensors employed to determine position and orientation of multi-rotor sUAS.

Common Sensors	Major Components & Characteristics	Type	Weight
ARK Flow	Distance sensor, IMU, PX4, indoor/outdoor	Optical Flow	5 g
Azure Kinect DK	90° × 59° FOV (Field Of View), 30FPS (Frames Per Second), IMU, Microphone	RGB-DCamera	440 g
Holybro M9N	Compass, PX4	GPS	32 g
Holybro PX4Flow	SONAR, Gyroscope, indoor/outdoor	Optical Flow	41 g
Realsense D435*i*	87° × 58° FOV, 90FPS, 9 m range	RGB-DCamera	72 g
Realsense L515	70° × 55° FOV, 30FPS, 9 m range, RGB camera, 3 W	LiDAR	100 g
Realsense T265	163.5° FOV, 30FPS, V-SLAM support with VPU, IMU	TrackingCamera	55 g
Ricoh Theta S	≈360° FOV, 30FPS, Microphone	MonocularCamera	125 g
VectorNav-200	72-channel GNSS, 0.03° accuracy, 0.05 $\frac{m}{s}$ accuracy	IMU+GNSS	4-16 g
VectorNav-300	2 × 72-channel GNSS, 0.03° accuracy, 0.05 $\frac{m}{s}$ accuracy	IMU+GNSS	5-30 g
Velodyne Puck LITE	360° × 30° FOV, 100 m range, GPS, 8 W	LiDAR	590 g

The large amount of data produced by the sUAS' sensors, some of which are particularly rich and complex, such as images produced by the onboard cameras, need to be processed and relevant information needs to be extrapolated to autonomously meet the mission objectives. Similarly, some guidance systems require high computational power to be executed in real time. For these reasons, sUAS involved in complex missions, such as navigating autonomously in unknown areas are equipped with one or multiple single-board computers. These computers, which can be as powerful as a modern smartphone or even a state-of-the-art laptop computer, are responsible for executing algorithms, whose cost exceeds the capabilities of common COTS autopilots and, in some cases, take the role of the autopilot. Table 3 lists some popular single-board computers used on sUAS and presents some of their key characteristics. Fig. 2 provides a schematic representation of a guidance, navigation, and control system architecture for sUAS. In this architecture, data produced by plug-in sensors, such as depth and tracking cameras, are employed to generate occupancy maps. Furthermore, data produced by altimeters and GNSS are merged with data produced by the autopilot's IMU to estimate the sUAS' state. Some depth and tracking cameras are able to produce reliable estimates of the sUAS' state as well. The occupancy maps and the estimates on the sUAS' state are exploited by the path planner to generate a sequence of collision-free waypoints, and by the trajectory planner to generate dynamically feasible reference trajectories that interpolate these waypoints. In this architecture, the autopilot is exploited for its

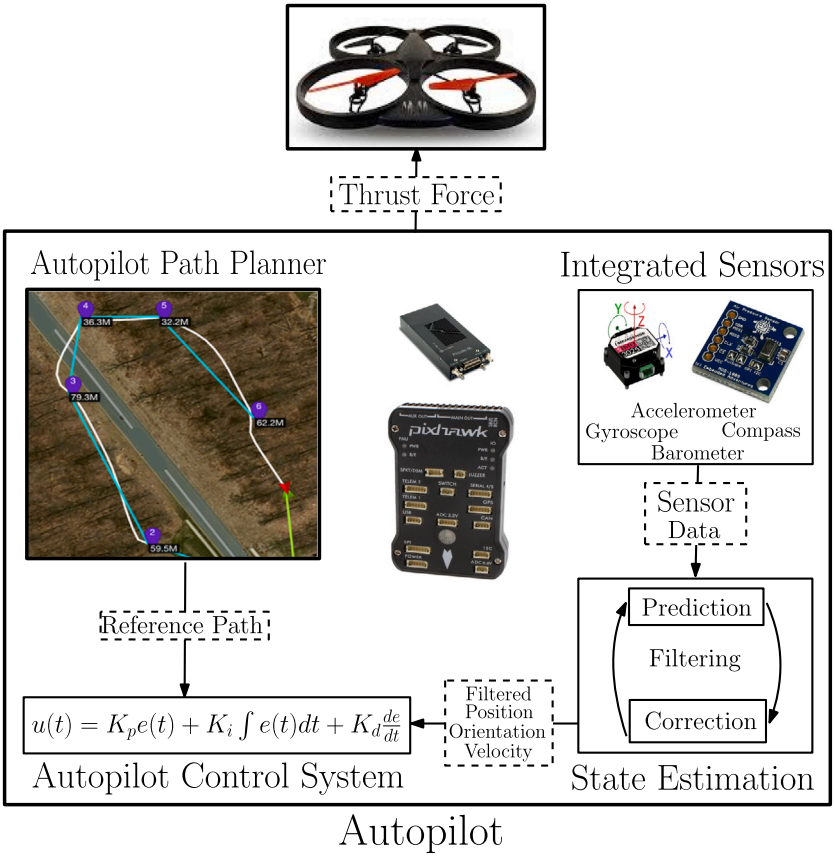

Fig. 1. An architecture of the guidance, navigation, and control system employed on COTS autopilots for sUAS. Data produced by the onboard sensors is elaborated by the navigation system to produce estimates of the sUAS' state. A path planner outlines sequences of waypoints to reach user-defined goal points. In some cases, the autopilot embeds a trajectory planner to interpolate the user-defined sequence of goal points or a subset thereof. Finally, a control law, typically linear, determines the thrust force each propeller must exert to track the reference path or trajectory.

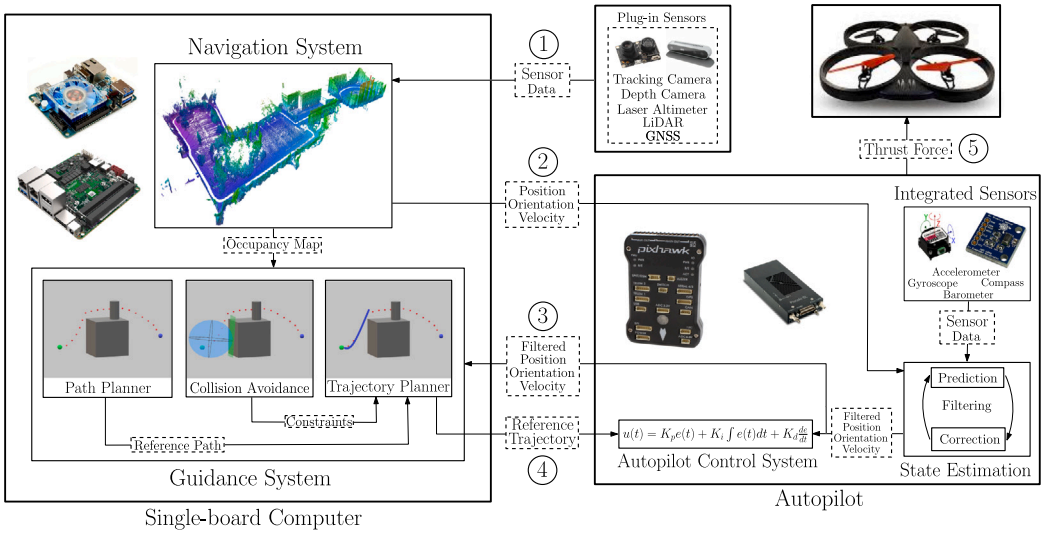

Fig. 2. Schematic representation of a guidance, navigation, and control system architecture for sUAS. Depth and tracking cameras or LiDARs are employed to generate occupancy maps. Data produced by altimeters, GNSS, and the autopilot's IMU are employed to estimate the sUAS' state. These data are exploited by the path planner and trajectory planner. The autopilot's control algorithm is used to determine the thrust force each propeller must exert. Circled numbers show the order in which information flows across sensors and algorithms. In some cases, trajectory planners, such as MPC algorithms, also produce a control input that overrides the control input computed by the autopilot.

sensors, its control algorithm, and its ability to coordinate the sUAS' motors. In some cases, the autopilot receives the reference trajectory for the sUAS' position as a sequence of points and hence, the autopilot's control law is tasked with closely following this sequence of points. In these cases, a trajectory planner is still necessary since, as discussed in Section 4 below, some path planners do not produce dynamically feasible paths. Usually, it is not possible to associate a time stamp to each sample point of a reference trajectory sent to the autopilot. To overcome this limitation, sampled points of the reference trajectory can be transmitted to the autopilot at a frequency computed so that,

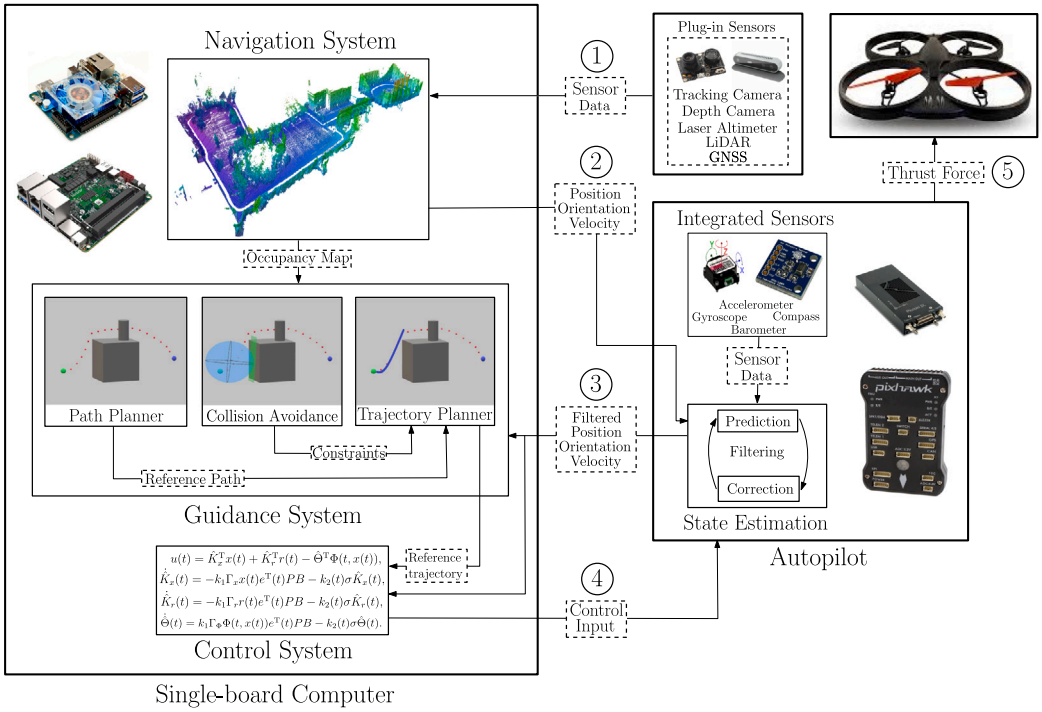

Fig. 3. Schematic representation of a guidance, navigation, and control system architecture for sUAS. The guidance, navigation, and control algorithms are executed on a single-board computer, and the autopilot is used only for its sensors for state estimation and for its ability to coordinate the sUAS' motors. Circled numbers show the information flow across sensors and algorithms. In some architectures, the single-board computer is interfaced with the motors directly.

Table 3
Some commonly used COTS single-board computers for multi-rotor sUAS.

Single-board computer	Major Characteristics	Weight	Power consumption
Intel NUC 7i7	2–32 GB RAM, 4.2 GHz, SSD storage	230 g	15 W
ODroid XU4	2 GB RAM, 2 GHz, GPIO (40 Pin header)	50 g	4 A/5 V
ODroid N2+	2–4 GB RAM, 2.4 GHz, GPIO (40 Pin header)	200 g	2A/12V
Raspberry Pi4	2–8 GB RAM, 1.5 GHz, GPIO (40 Pin header)	50 g	2.5 A/5 V
Nvidia Jetson TX2	8 GB RAM, 2 GHz, Camera support from 256 core GPU	300 g	7–15 W
Nvidia Jetson TX2 NX	4 GB RAM, 2 GHz, Camera support from 128 core GPU	90 g	5–10
VOXL Flight	4 GB RAM, 2.15 GHz, includes Flight Core autopilot	40 g	3–10 W

knowing the sUAS' reference velocity, the reference trajectory is followed at the desired speed. Alternatively, it is possible to send the sUAS' reference velocity, sampled over time, to the autopilot. In other cases, the autopilot receives the desired position in the sUAS' vertical direction and the desired attitude in the form of discrete-time sequences and hence, the autopilot's control input is designed to track the desired inner loop dynamics; for additional details, see Section 6.1 below. Finally, in some cases, trajectory planners, such as MPC algorithms, also produce a control input that is passed to the sUAS' motors through the autopilot, and the autopilot's control system is overridden.

In several mission scenarios, the sUAS must meet user-defined specifications on the trajectory tracking error despite poor knowledge of their payload, possible faults and failures, or strong disturbances. These requirements are usually met by employing control laws, whose complexity and computational costs exceed the capabilities of the processors installed in common COTS autopilots. Therefore, in these cases, the control laws are computed on a single-board computer. Fig. 3 shows a graphical representation of a guidance, navigation, and control system architecture for sUAS, whereby path planning, trajectory planning, localization and mapping, and control algorithms are executed on the single-board computer. In this case, the autopilot is exploited only for its sensors, whose data are merged with the data produced by additional plug-in sensors, and as an interface with the sUAS' motors. In similar architectures, the single board computer serves as autopilot, and COTS autopilots are not employed.

The hardware aboard sUAS is chosen according to the mission objectives and requirements. Quadcopters employed for recreational purposes are usually equipped with an autopilot, a relatively inexpensive GNSS unit, an IMU, cameras to capture aerial images, and antennas to transmit telemetry and images in real time. Quadcopters tasked with an autonomous indoor-only mission do not utilize GPS, since the satellite signal is usually unreliable or unavailable, and are usually equipped with optical flow sensors, depth and tracking cameras, or LiDARs to determine the vehicle's position and orientation relative to the environment, and a single-board computer to elaborate the sensors' data and enable autonomous operations. Quadcopters performing daytime missions outdoors may be equipped with LiDARs, since cameras may be blinded by strong light, and a GNSS unit. Quadcopters performing night-time missions or operating in subterranean environments, such as caves and mines, may be equipped with LiDARs and infrared cameras.

The required levels of precision and robustness to disturbances, faults, failures, and uncertainties varies with the mission objectives. For example, sUAS for recreational purposes need to guarantee submeter precision and moderate robustness to minor damages. Outdoor operations in complex environments, such as forests, may require subcentimeter precision, and hence are equipped with a differential GPS (DGPS), depth and tracking cameras, and high-resolution IMUs, to

determine the vehicle's position and attitude. Alternatively, if a quadcopter operates in an indoor industrial environment, in close proximity to untrained personnel and expensive infrastructures, sub-centimeter precision and high levels of robustness to uncertainties in the transported payload, environmental conditions, faults, and failures are usually demanded. The required level of precision influences both the quality of the hardware components and the expected ability of the guidance, navigation, and control algorithms to leverage these components and meet the desired levels of performance. This paper surveys guidance, navigation, and control techniques for multi-rotor sUAS considering both algorithms that meet lower levels of performance and more advanced algorithms that guarantee higher levels of precision and robustness.

3. Equations of motion of quadcopters

Quadcopter sUAS comprise four propellers, whose spin axes are parallel to one another, and which are connected by means of an H- or X-shaped frame made of carbon fiber, plastic, or other material with good stiffness and elasticity properties. The center of the frame hosts the vehicles' electronics, such as its single-board computer, a GNSS unit, an IMU, antennas to receive commands and transmit telemetry in real time, a battery, and one or multiple cameras to observe and collect data on the environment; Fig. 4 shows a typical quadcopter.

The optical axis of these cameras is usually aligned with the bisector of the angle between two arms of the sUAS' frame, and this axis is commonly chosen as the vehicle's *roll axis* $x_{\text{body}} : [t_0, \infty) \to \mathbb{R}^3$, where $\|x_{\text{body}}(t)\| = 1$, $t \geq t_0$. There is not a commonly accepted choice on the vehicle's *yaw axis* $z_{\text{body}} : [t_0, \infty) \to \mathbb{R}^3$, where $\|z_{\text{body}}(t)\| = 1$, $t \geq t_0$, and $z_{\text{body}}^{\text{T}} x_{\text{body}}(t) = 0$. Numerous authors, especially those with an aeronautical background, set the yaw axis so that it points down, whereas many other authors, especially those with an electrical engineering or computer science background, set, without loss of generality, the yaw axis so that it points up; in this paper, the aeronautical convention is followed. Having set the roll and yaw axes, the *pitch axis* $y_{\text{body}} : [t_0, \infty) \to \mathbb{R}^3$ is set so that the sUAS' reference frame $\mathbb{J}(\cdot) \triangleq \{A(\cdot); x_{\text{body}}(\cdot), y_{\text{body}}(\cdot), z_{\text{body}}(\cdot)\}$, which is centered at the reference point $A : [t_0, \infty) \to \mathbb{R}^3$, is right-handed and orthonormal, that is, $x_{\text{body}}^{\times}(t) y_{\text{body}}(t) = z_{\text{body}}(t)$, $t \geq t_0$, where $(\cdot)^{\times}$ denotes the cross-product operator. The reference point $A(\cdot)$ is usually chosen as the vehicle's center of mass. However, other choices, such as the payload's position in the case of autonomous sUAS transporting sling payloads (Feng, Rabbath, & Su, 2018; Schmuck & Chli, 2019a), are equally suitable. Fig. 4 shows an orthonormal reference frame set according to the aeronautical convention.

The sUAS' position and attitude are captured relative to an orthonormal reference frame $\mathbb{I} \triangleq \{O; X, Y, Z\}$ fixed with the Earth and considered as inertial. Following an aerospace convention, the axes of this reference frame can be set, for instance, so that X points North, Y points East, and Z points down, that is, aligned with the vehicle's weight captured by $F_{\text{g}}^{\mathbb{I}} = mgZ$, where $m > 0$ denotes the sUAS' *mass* and $g > 0$ denotes the *gravitational acceleration*. The sUAS' mass is usually considered as a constant; recently, the problem of modeling and controlling sUAS, whose mass varies as a function of time due to sudden payload dropping or a slow release of payload, has been addressed in L'Afflitto and Mohammadi (2017). In the following, if a vector $a \in \mathbb{R}^3$ is expressed in the reference frame $\mathbb{I}$, then this vector is denoted by $a^{\mathbb{I}}$. Alternatively, if a vector is expressed in the reference frame $\mathbb{J}(\cdot)$, then no superscript is used.

In the reference frame $\mathbb{I}$, the *translational kinematic equation* of a quadcopter is given by

$$\dot{r}_A^{\mathbb{I}}(t) = v_A^{\mathbb{I}}(t), \qquad r_A^{\mathbb{I}}(t_0) = r_{A,0}^{\mathbb{I}}, \qquad t \geq t_0, \tag{1}$$

and the *translational dynamic equation* is given by

$$F_{\text{g}}^{\mathbb{I}} - F_{\text{T}}^{\mathbb{I}}(t) + F^{\mathbb{I}}(t) = m\left[\dot{v}_A^{\mathbb{I}}(t) + \ddot{r}_C^{\mathbb{I}}(t)\right], \qquad v_A^{\mathbb{I}}(t_0) = v_{A,0}^{\mathbb{I}}, \tag{2}$$

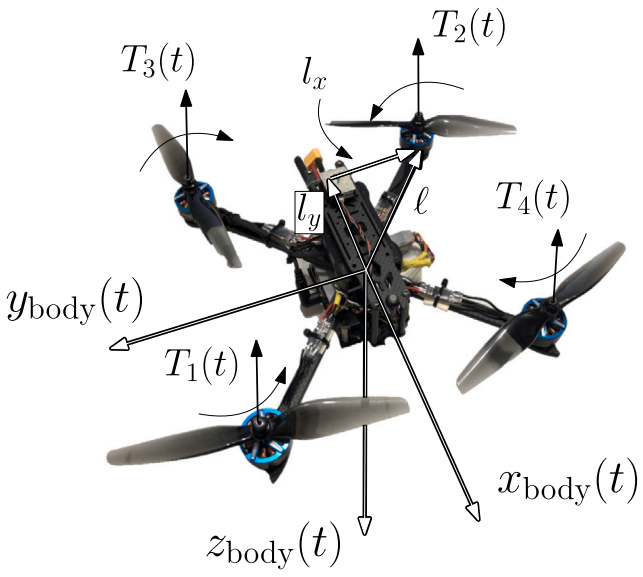

Fig. 4. Orthonormal reference frame centered at the sUAS' center of mass and oriented according to the aeronautical conventions, that is, with $z_{\text{body}}(\cdot)$ pointing down and $x_{\text{body}}(\cdot)$ aligned with the roll axis. Quadcopters are usually symmetric so that propellers are placed at a distance l_x from the pitch axis and l_y from the roll axis.

where $-F_{\text{T}}(t) \triangleq [0, 0, u_1(t)]^{\text{T}}$ denotes the *thrust force*, that is, the force produced by the propellers that allows a quadcopter to hover, $F : [t_0, \infty) \to \mathbb{R}^3$ denotes the aerodynamic forces acting on the sUAS, and $r_C : [t_o, \infty) \to \mathbb{R}^3$ denotes the position of the sUAS' center of mass relative to the user-defined reference point $A(\cdot)$. External disturbances can be accounted for in the left-hand side of (2) as unknown, bounded functions of time. Indeed, some authors embed both aerodynamic forces and external disturbances in $F^{\mathbb{I}}(\cdot)$.

The orientation of the body reference frame $\mathbb{J}(\cdot)$ relative to the inertial reference frame $\mathbb{I}$ is usually captured by means of a 3-2-1 rotation sequence of implicit Tait–Bryan angles so that

$$\left[x_{\text{body}}^{\mathbb{I}}(t), y_{\text{body}}^{\mathbb{I}}(t), z_{\text{body}}^{\mathbb{I}}(t)\right] = R(\phi(t), \theta(t), \psi(t)), \qquad t \geq t_0,$$

where

$$R(\phi, \theta, \psi) \triangleq \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix},$$
$$(\phi, \theta, \psi) \in [0, 2\pi) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [0, 2\pi),$$

$\phi : [t_0, \infty) \to [0, 2\pi)$ denotes the *roll angle*, $\theta : [t_0, \infty) \to (-\frac{\pi}{2}, \frac{\pi}{2})$ denotes the *pitch angle*, and $\psi : [t_0, \infty) \to [0, 2\pi)$ denotes the *yaw angle*. In this case, the *rotational kinematic equation* of a quadcopter is given by

$$\begin{bmatrix} \dot{\phi}(t) \\ \dot{\theta}(t) \\ \dot{\psi}(t) \end{bmatrix} = \Gamma(\phi(t), \theta(t))\omega(t), \qquad \begin{bmatrix} \phi(t_0) \\ \theta(t_0) \\ \psi(t_0) \end{bmatrix} = \begin{bmatrix} \phi_0 \\ \theta_0 \\ \psi_0 \end{bmatrix}, \tag{3}$$

where

$$\Gamma(\phi, \theta) \triangleq \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}, \qquad (\phi, \theta) \in [0, 2\pi) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right).$$

The Tait–Bryan angles, like any attitude representation system based on three independent parameters, are limited by the presence of a singular configuration, that is, a configuration wherein the attitude representation is not unique, and a finite angular velocity would imply an infinite

Sidebar 3.1. Although the use a 3-2-1 rotation sequence of implicit Tait-Bryan angles, which are often termed Euler angles, is universally accepted, several authors resort to Euler parameters, which are often confused with quaternions, whenever the sUAS' pitch angle $\theta(\cdot)$ approximates or exceeds the boundaries of $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$. The aerodynamic forces are $F(\cdot)$ are rarely considered in the literature on sUAS, except in those papers that present robust control laws for sUAS travelling at higher altitudes, where the effect of the wind on the vehicle's dynamics is not negligible.

time derivative of the angles capturing the vehicle's orientation. In the case of a 3-2-1 rotation sequence, this singular configuration is attained for $|\theta(t)| = \frac{\pi}{2}$, $t \geq t_0$. To overcome this problem, some authors employ attitude representation system based on four parameters, such as the Euler parameters, which are commonly referred to as quaternions. The *rotational dynamic equation* of a quadcopter, whose frame is modeled as a rigid body, whose payload is rigidly attached to the vehicle, and whose propellers are modeled as thin spinning discs, is given by

$$\begin{aligned} &M_{\mathrm{T}}(t) + M_{\mathrm{g}}(r_C(t), \phi(t), \theta(t)) + M(t) \\ &\quad = m r_C^{\times}(t) \left[\dot{v}_A(t) + \omega^{\times}(t) v_A(t)\right] + I\dot{\omega}(t) \\ &\qquad + \omega^{\times}(t) I \omega(t) + I_{\mathrm{P}} \sum_{i=1}^{4} \begin{bmatrix} 0 \\ 0 \\ \dot{\Omega}_{\mathrm{P},i}(t) \end{bmatrix} + \omega^{\times}(t) I_{\mathrm{P}} \sum_{i=1}^{4} \begin{bmatrix} 0 \\ 0 \\ \Omega_{\mathrm{P},i}(t) \end{bmatrix}, \\ &\omega(t_0) = \omega_0, \qquad t \geq t_0, \end{aligned} \tag{4}$$

where $M_{\mathrm{T}}(t) = [u_2(t), u_3(t), u_4(t)]^{\mathrm{T}}$ denotes the *moment of the forces induced by the propellers*, $M_{\mathrm{g}}(r_C, \phi, \theta) \triangleq r_C^{\times} F_{\mathrm{g}}^{\mathbb{I}}(\phi, \theta)$, $(r_C, \phi, \theta) \in \mathbb{R}^3 \times [0, 2\pi) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, denotes the moment of the quadcopter's weight with respect to A,

$$\begin{aligned} &F_{\mathrm{g}}^{\mathbb{I}}(\phi, \theta) = mg[-\sin(\theta), \cos(\theta)\sin(\phi), \cos(\theta)\cos(\phi)]^{\mathrm{T}}, \\ &\qquad (\phi, \theta) \in [0, 2\pi) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right), \end{aligned}$$

$M : [t_0, \infty) \to \mathbb{R}^3$ denotes the moment of the aerodynamic forces with respect to $A(\cdot)$, $I \in \mathbb{R}^{3\times3}$ denotes the symmetric positive-definite *matrix of inertia* of the sUAS relative to $A(\cdot)$, excluding the propellers, $I_{\mathrm{P}} \in \mathbb{R}^{3\times3}$ denotes the symmetric positive-definite *matrix of inertia* of each propeller relative to $A(\cdot)$, and $\Omega_{\mathrm{P},i} : [t_0, \infty) \to \mathbb{R}$, $i = 1, \ldots, 4$, denotes the angular velocity of the ith propeller. The terms $I_{\mathrm{P}} \sum_{i=1}^{4} [0, 0, \dot{\Omega}_{P_i}(t)]^{\mathrm{T}}$, $t \geq t_0$, and $\omega^{\times}(t) I_{\mathrm{P}} \sum_{i=1}^{4} [0, 0, \Omega_{P_i}(t)]^{\mathrm{T}}$ denote the *inertial counter-torque* and the *gyroscopic effect*, respectively. It is common practice to assume that the sUAS' axes are principal axes of inertia and, hence, the matrix I is usually considered as diagonal, and its entries are denoted by I_x, I_y, and $I_z > 0$. External disturbances can be accounted for in the left-hand side of (4) as unknown, bounded functions of time. Indeed, some authors embed both aerodynamic forces and external disturbances in $M(\cdot)$.

In the following, we denote the equations of motion (1)–(4) by

$$\dot{x}(t) = f(t, x(t)) + G(x(t))u(t), \qquad x(t_0) = x_0, \qquad t \geq t_0, \tag{5}$$

where $x(t) \triangleq \left[q^{\mathrm{T}}(t), q_{\mathrm{dot}}^{\mathrm{T}}(t)\right]^{\mathrm{T}}$, $q(t) = \left[\left(r_A^{\mathbb{I}}(t)\right)^{\mathrm{T}}, \phi(t), \theta(t), \psi(t)\right]^{\mathrm{T}}$ denotes the *vector of independent generalized coordinates*, and $q_{\mathrm{dot}}(t) \triangleq \left[\left(\dot{r}_A^{\mathbb{I}}(t)\right)^{\mathrm{T}}, \omega^{\mathrm{T}}(t)\right]^{\mathrm{T}}$ denotes the *vector of quasi-velocities*, $u(t) \triangleq [u_1(t), u_2(t), u_3(t), u_4(t)]^{\mathrm{T}}$ denotes the *control vector*; assuming, for brevity, that $r_C(t) \equiv 0$, $t \geq t_0$, and $I_{\mathrm{P}} = 0_{3\times3}$, it follows from (1)–(4) that

$$f(t, x) \triangleq \left[\left(v_A^{\mathbb{I}}\right)^{\mathrm{T}}, gZ + \left(F^{\mathbb{I}}(t)\right)^{\mathrm{T}}, \omega^{\mathrm{T}} \Gamma^{\mathrm{T}}(\phi, \theta), \left(M(t) - \omega^{\times} I\omega\right)^{\mathrm{T}} I^{-\mathrm{T}}\right]^{\mathrm{T}},$$

$$G(x) \triangleq \begin{bmatrix} 0_{5\times1} & 0_{5\times3} \\ m^{-1} & 0_{1\times3} \\ 0_{3\times1} & 0_{3\times3} \\ 0_{3\times1} & I^{-1} \end{bmatrix},$$

for all $(t, x) \in [t_0, \infty) \times \left(\mathbb{R}^3 \times [0, 2\pi) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [0, 2\pi) \times \mathbb{R}^6\right)$, where $0_{n\times m}$ denotes the zero matrix in $\mathbb{R}^{n\times m}$. The control vector $u(\cdot)$ comprises the third component of the thrust force $F_{\mathrm{T}}(\cdot)$ and the moment of the forces induced by the propellers $M_{\mathrm{T}}(\cdot)$. It is possible to verify that

$$\begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -l_y & l_y & -l_y & l_y \\ l_x & -l_x & -l_x & l_x \\ -c_{\mathrm{T}} l & -c_{\mathrm{T}} l & c_{\mathrm{T}} l & c_{\mathrm{T}} l \end{bmatrix} \begin{bmatrix} T_1(t) \\ T_2(t) \\ T_3(t) \\ T_4(t) \end{bmatrix}, \qquad t \geq t_0, \tag{6}$$

where $T_i : [t_0, \infty) \to \mathbb{R}$, $i = 1, \ldots, 4$, denotes the component of the force produced by the ith propeller along the $-z_{\mathrm{body}}(\cdot)$ axis of the reference frame $\mathbb{J}(\cdot)$, $l_x > 0$ denotes the distance of the sUAS' propellers from the $x_{\mathrm{body}}(\cdot)$ axis, $l_y > 0$ denotes the distance of the sUAS' propellers from the $y_{\mathrm{body}}(\cdot)$ axis, $l \triangleq \sqrt{l_x^2 + l_y^2}$, and $c_{\mathrm{T}} > 0$ denotes the propellers' drag coefficient. The thrust force generated by the ith propeller is related to the propeller's angular velocity by

$$T_i(t) = k\Omega_{\mathrm{P},i}^2(t), \qquad i = 1, \ldots, 4, \qquad t \geq t_0, \tag{7}$$

where $k > 0$ is usually determined experimentally. Some authors account also for the motors' dynamics while formulating the sUAS' equations of motion.

It follows from (5) that a quadcopter's equilibrium condition is attained by hovering at some constant altitude and with some constant yaw angle. It is common practice to design quadcopters so that their thrust-to-weight ratio is near 2-to-1. Thus, small rotations about the pitch or roll axes lead to rapid translations along the inertial horizontal plane. For these reasons, and since sUAS are seldom tasked with aggressive maneuvers, it is common practice to consider the linearized equations of motion of a quadcopter, while designing guidance, navigation, and control systems for sUAS. Linearizing (5) about a user-defined equilibrium condition, the sUAS' dynamics can be approximated by

$$\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t) + \tilde{B}\tilde{u}(t), \qquad \tilde{x}(t_0) = \tilde{x}_0, \qquad t \geq t_0, \tag{8}$$

where $\tilde{x}(t) \triangleq \left[q^{\mathrm{T}}(t), \dot{q}^{\mathrm{T}}(t)\right]^{\mathrm{T}}$ denotes the *linearized state vector*, $\tilde{u}(t) \triangleq [u_1(t) - mg, u_2(t), u_3(t), u_4(t)]^{\mathrm{T}}$, $\tilde{A} \triangleq \begin{bmatrix} 0_{6\times3} & 0_{6\times2} & 0_{6\times1} & \mathbf{1}_6 \\ 0_{2\times3} & \begin{bmatrix} 0 & -g \\ g & 0 \end{bmatrix} & 0_{2\times1} & 0_{2\times6} \\ 0_{4\times3} & 0_{4\times2} & 0_{4\times1} & 0_{4\times6} \end{bmatrix} \in \mathbb{R}^{12\times12}$, $\mathbf{1}_n$ denotes the *identity matrix* in $\mathbb{R}^{n\times n}$, and $\tilde{B} \triangleq \begin{bmatrix} 0_{8\times4} \\ \mathrm{diag}(m^{-1}, I_x^{-1}, I_y^{-1}, I_z^{-1}) \end{bmatrix} \in \mathbb{R}^{12\times4}$.

The guidance, navigation, and control problems for multi-rotor sUAS are usually solved numerically in discrete time. Thus, the continuous-time linearized equations of motion of multi-rotor sUAS can be recast in a discrete-time form by employing, for instance, the zero-order hold approach so that (8) is discretized as

$$\hat{x}(t_0 + (i+1)\Delta T) = A\hat{x}(t_0 + i\Delta T) + B\hat{u}(t_0 + i\Delta T),$$

Sidebar 3.2. A common assumption is to set the sUAS' reference point $A(\cdot)$ as its center of mass so that $r_C(t) \equiv 0$, $t \geq t_0$. This modeling choice allows to decouple the translational equations of motion (1) and (2) from the rotational equations of motion (3) and (4), and, hence, simplifies the design of guidance, navigation, and control systems for sUAS. Assuming to know the position of the sUAS' center of mass is usually realistic for Class 1 quadcopters since these vehicles are sufficiently small, their mass distribution is well-known, and the payload is fixed with the sUAS and well-known. However, assuming to know the position of the sUAS' center of mass becomes limiting whenever the sUAS' payload is heavy and sling and for large sUAS. Setting $r_C(t) \equiv 0$, $t \geq t_0$, is also advantageous since it decouples (4) from (2) and hence, it simplifies the problem of designing control laws.

$$\hat{x}(t_0) = \tilde{x}_0, \qquad i \in \mathbb{N} \cup \{0\}, \tag{9}$$

where $A = e^{\tilde{A}\Delta T}$, $B = \int_0^{\Delta T} e^{\tilde{A}\sigma} \mathrm{d}\sigma \tilde{B}$, and $\Delta T > 0$ denotes the time step. In some cases, it is more convenient to set the boundary conditions on the sUAS' position and translational velocity both at t_0 and at the time instant $t_0 + n_t \Delta T$, where $n_t \in \mathbb{N}$ is user-defined; in these cases, to accommodate 12 boundary conditions on the 12 states comprised in $\hat{x}(\cdot)$, the boundary conditions on the sUAS' attitude and angular velocity cannot be imposed arbitrarily.

4. Guidance systems for autonomous multi-rotor sUAS

Guidance can be described as the determination of the vehicle's reference path or trajectory that should be closely followed by a user-defined reference point on the vehicle. Path planning aims at outlining a collision-free sequence of waypoints from the vehicle's current position to the goal set. Trajectory planning aims at outlining a continuous collision-free curve, parameterized by time, compatibly with user-defined constraints such as the need to avoid saturation of the actuators and the need not to exceed the vehicles' flight envelope. Usually, trajectory planning systems for sUAS are also aimed at defining the sUAS' reference yaw angle so that the onboard sensors or the payload are steered in a specific direction. For instance, it is common practice to steer the sUAS' longitudinal axis so that the next waypoint is always in the field of view of the sUAS' cameras. Finally, in numerous cases, the trajectory planning system is tasked with defining the sUAS' reference angular position and angular velocity, and hence, with producing reference trajectories for the sUAS by steering the thrust force in some direction compatibly with the sUAS' rotational dynamics.

This section surveys path planning and trajectory planning techniques by recalling both classical approaches and recent learning-based methods. Thus far, the path planning and the trajectory planning problems have been largely addressed separately since some guidance systems embed either a path planner or a trajectory planner. However, addressing these two problems separately may produce several challenges. Indeed, numerous path planners do not account for the sUAS' dynamics, and if only a path planner is employed, then following dynamically unfeasible reference trajectories stresses the performance of the control system; for additional details, see Section 6.1 below. Numerous trajectory planners operate on sets characterized by strong properties, such as convexity, since convexity of the cost function over the constraint set is a sufficient condition on the existence of a unique solution for optimization-based planners; for additional details, see Section 5.5 below. However, operating on convex sets is a limiting factor in most problems of practical interest. Indeed, if only a trajectory planner is employed, and this planner operates on convex sets only, then, it may be impossible to find a reference trajectory leading the sUAS from its current location to the goal set. In many cases, path planners and trajectory planners are employed sequentially by exploiting the ability of most path planners to produce a sequence of waypoints from the sUAS' current location to the goal set in the presence of complex constraints and then, by employing trajectory planners to interpolate multiple waypoints produced by the path planner. Recently, some effort has been made to integrate path and trajectory planners, and this section also surveys approaches to merging path and trajectory planning systems. In this section, we also survey some of the most notable techniques employed to generate reference paths and trajectories for swarms of sUAS. Finally, we survey two relatively new areas, namely guidance systems for autonomous sUAS operating in a tactical manner to support operations of ground troops and law enforcement agencies and guidance systems for autonomous sUAS flying as fast as possible.

4.1. Path planners for sUAS

In the following, we present multiple path planning techniques grouped as optimization-based and non-optimization-based methods, local and global methods, roadmap methods, exact or approximate cell decomposition methods, reactive methods, methods based on motion primitives, bio-inspired approaches, and learning-based methods; these classifications are not mutually exclusive. Furthermore, we discuss a relatively under-explored class of path planners, that is, planners designed to guarantee stability of the vehicle. Other classifications of path planning methods for mobile robots, such as offline and online methods, are not discussed, since the totality of guidance systems for multi-rotor sUAS currently involve only online path planners. Key features of the most relevant path planning techniques surveyed in this paper are listed in Table 4. Presently, COTS autopilots, such as those listed in Table 1, implement relatively simple path planning algorithms; for instance, ArduPilot allows to implement the Dijkstra's algorithm. However, more complex algorithms, including the majority of those surveyed in the following, need to be executed on single-board computers such as those listed in Table 3.

Unless stated otherwise, in the following, the path planning problem is considered as three-dimensional. Exploiting the vertical direction allows to explore more solutions to the path planning problem than solvers designed for two-dimensional problems only. However, in general, exploring the three-dimensional space implies increased computational costs. Furthermore, increasing the sUAS' altitude usually depletes the batteries more rapidly.

4.1.1. Optimization and non-optimization based methods

Non-optimization-based path planning methods are merely concerned with the problem of finding a reference path from the sUAS' current location to the goal set, which is feasible according to the information on the environment available at the time the planning process is initiated. Among non-optimization-based path planning methods is the *rapidly exploring random tree* method (*RRT*) (Saravanakumar

Table 4
Some of the most common path planning methods for sUAS, some of their key features, and selected references applying these methods to multi-rotor sUAS. The majority of existing techniques, or variations thereof, produce reference paths that minimize some cost function. Relatively few methods are global, that is, search the entire space in which the sUAS operates; local planning techniques need to be applied on multiple partitions of the map. Few techniques are complete and hence, allow to find a reference path from any initial position to any goal set. In some cases, a complete solution may be attained by applying a path planning technique to multiple adjacent subsets of the free space, which meet some specific criteria such as convexity.

Algorithm	Optimal	Global	Complete	Cell decomposition	Ref.
A* & variations	✓	✓	✓	✓	Zhao, Zhang, and Zhao (2020)
D* & variations	✓	✓	✓	✓	Koenig and Likhachev (2002, 2005)
Disjunctive convex programming	✓		Through iterations		Blackmore, Ono, and Williams (2011)
Maneuver automata	✓				Frazzoli, Dahleh, and Feron (1999), Schouwenaars, Mettler, Feron, and How (2004)
Mixed integer linear program	✓		Through iterations		Babel (2019)
Mixed integer quadr. program	✓		Through iterations		Tordesillas, Lopez, and How (2019)
Motion primitives	✓				Yadav and Tanner (2020), Zhou, Gao, Wang, Liu, and Shen (2019a)
Potential field	✓		Through iterations		Woods and La (2019)
Probabilistic roadmap		✓	✓	✓	Xu, Deng, and Shimada (2021)
RRT & variations	RRT^*	✓	✓	✓	Karaman and Frazzoli (2011), Saravanakumar, Kaviyarasu, and Ashly Jasmine (2021) Meng, Pawar, Kay, and Li (2018)
Wavefront algorithm	✓	✓	✓	✓	Hebecker, Buchholz, and Ortmeier (2015)

et al., 2021), which is still considered as one of the fastest path planning methods, compatible with the quality of the underlying random number generator (Noreen, Khan, Ryu, Doh, & Habib, 2018), and the *probabilistic roadmap method* (Xu et al., 2021), which guarantees low computational costs.

It is rare that a satisfactory path is merely collision-free since multiple additional criteria such as minimum time or minimum distance, to name two of the most common metrics, must be met. Optimization-based methods are designed to meet both collision avoidance constraints, the requirement of maximizing some reward function or, equivalently, of minimizing some cost function. Among the optimization methods, A^* (Zhao et al., 2020), which is widely used for its fast convergence and low time complexity (Primatesta, Guglieri, & Rizzo, 2019), D^*, and D^*-lite (Koenig & Likhachev, 2002, 2005), which are suitable for highly dynamic environments, mixed integer quadratic programming (Tordesillas et al., 2019), RRT^* (Karaman & Frazzoli, 2011), *informed* RRT^* (Meng et al., 2018), which guarantees faster convergence than conventional RRT^*, mixed integer linear programming (Babel, 2019), and disjunctive convex programming (Blackmore et al., 2011).

Algorithm 1 provides an implementation of the A^* technique to reach the *goal node* n_g from the *initial node* n_0 through a weighted graph. In path planning applications for sUAS, this graph can be provided, for instance, by creating partitions of a map that are not occupied by obstacles, setting a point for each partition as a node of the graph, and setting the weights on the edges of the graph as the distance between nodes. Thus, the A^* algorithm provides a sequence of nodes $\mathcal{N}^*$ connecting n_0 to n_g that minimizes the *cost function*

$$f(n_0, n_g) \triangleq g(n_0, n_c) + h(n_c, n_g), \qquad (n_0, n_c, n_g) \in \mathcal{N} \times \mathcal{N} \times \mathcal{N}, \tag{10}$$

where the *cost-to-come function* $g : \mathcal{N} \times \mathcal{N} \to \mathbb{R}$ denotes the cost of reaching the current node n_c from n_0, the *heuristic function* $h : \mathcal{N} \times \mathcal{N} \to \mathbb{R}$ denotes an underestimate of the cost of reaching the node n_g from n_c, and $\mathcal{N}$ denotes the set of nodes of the graph. As it progresses through the graph, the A^* algorithm constructs $\mathcal{N}^*$ by investigating all the nodes adjacent to the current node n_c, selecting the node that provides the lowest $f(n_0, n_g)$, and finally tracing back $\mathcal{N}^*$ from n_g to n_0.

4.1.2. Global and local methods

Global methods provide a reference path by searching the entire space in which the sUAS operates, whereas local methods provide reference paths to subsets of the goal set or waypoints located in subsets of the search space that meet some specific criteria such as convexity. Many of the aforementioned techniques, such as RRT, A^*, or D^*-lite, are global methods. Among local path planning methods applied to sUAS, we recall the *wavefront algorithm* (Hebecker et al., 2015), and some versions of the *potential field method* (Woods & La, 2019) that require partitioning the search space into convex subsets. Restricting potential fields on convex sets allows to avoid multiple, undesired, local minima where the search algorithm may converge.

In general, global path planners are preferred over local path planners since a path planner's role is to coarsely outline a way for the sUAS to complete the given mission, whereas the trajectory planner locally refines the path and accounts for the vehicle's dynamic constraints. A notable exception to this consideration is given by the use of potential field methods to locally coordinate swarms of sUAS (Zhou & Schwager, 2016). However, the use of local repulsive fields to avoid collisions among formation agents and with obstacles alongside the use of attractive fields to draw the swarm to the goal set may lead to undesired conditions of local equilibria.

Algorithm 1: A^* algorithm for returning best path relative to the cost function $f(\cdot,\cdot)$ and the cost-to-come function $g(\cdot,\cdot)$.

Result: $\mathcal{N}^*$: nodes of the shortest path from n_0 to n_g.
Create empty priority queue $\mathcal{O}$;
n_0 = **Current node** ;
Insert $n_0 \to \mathcal{O}$;
while $\mathcal{O}$ *is not empty* **do**
  Set current node n **to first node in** $\mathcal{O}$;
  if $n = n_g$ **then**
    Break while and Reconstruct Path($\mathcal{N}^*$) ;
  end
  Remove n **from** $\mathcal{O}$;
  for *All* n' *adjacent to* n *in* $\mathcal{N}$ **do**
    if $n' \notin \mathcal{O}$ **then**
      Insert $n' \to \mathcal{O}$;
      In $\mathcal{O}$**, set pointer of** n' **towards** n ;
    end
    else if $g(\cdot, n') > g(n_0, n) + g(n, n')$ **then**
      Modify $\mathcal{O}$ **by setting pointer of** n' **towards** n ;
      Update $n' \to \mathcal{O}$;
    end
  end
end
Reconstruct Path($\mathcal{N}^*$): *Trace the pointers from* n_g *back to* n_0;

4.1.3. Roadmap methods

Roadmap approaches to the path planning problem for sUAS include the *visibility graph method* (Blasi, D'Amato, Mattei, & Notaro, 2020), which is usually implemented in static environments, *Voronoi diagrams* (Baek, Han, & Han, 2020), which are easy to implement, but are also inefficient in high dimensions because they require complex data structures and long pre-processing times, and *power diagrams* (Aurenhammer, 1987), which provide a generalization of Voronoi diagrams. To the authors' knowledge, alternative roadmap approaches, such as *freeway nets* (Latombe, 2012, Ch. 4) and *silhouettes* (LaValle, 2006, Ch. 6), which have been employed for other classes of mobile robots, such as autonomous ground vehicles operating indoors, have not been applied yet to the path planning problem for sUAS. Although current implementations of freeway nets concern two-dimensional problems, this technique provides a good solution to the path planning problem in structured environments, such as manufacturing plants or towns with grid street plans, where large safety margins are preferable and excursions in the vertical direction are discouraged. Thus, the use of freeway nets may be worthwhile being investigated. For the complexity of their construction, silhouettes may be particularly demanding for computers on current sUAS.

4.1.4. Decomposition methods

Decomposition methods consist in partitioning a map of the environment and outlining paths that join some points deemed as representatives of these partitions. Some path planning techniques, such as the visibility graph, Voronoi diagrams, A^*, and D^* are cell decomposition methods. An example of a path planning method that is not based on cell decomposition is the potential field method (Latombe, 2012, Ch. 7).

In general, decomposition methods are classified in exact and approximate methods. Exact cell decomposition methods, such as the *Schwartz and Sharir method* (Schwartz & Sharir, 1983) and the *Canny method* (Canny, 1988), operate over partitions of the free space, whose union is equal to the space perceived as free by the onboard navigation system. All cell decomposition methods are global path planning techniques and exact methods are more computationally onerous, but complete, that is, are guaranteed to provide a feasible path. Approximate cell decomposition methods, such as *grid-based methods* (Galceran & Carreras, 2013) and the *circle packing method* (Thurston, 1979), operate over partitions of the free space whose union is a subset of the free space. To the authors' knowledge, although these techniques were successfully implemented on relatively slow mobile ground robots operating indoors, neither the Schwartz and Sharir method nor the Canny method have been applied to sUAS since their computational cost may be overly large for current single-board computers.

4.1.5. Reactive path planning methods

The path planning methods presented thus far assume that a map of the environment is known at the time the planning algorithm is executed. Dynamic obstacles or newly detected obstacles are accounted for by executing the path planning algorithm at a frequency that is equal to, or, at least, similar to the frequency at which obstacle maps are produced. However, in densely occupied environments, creating consistent maps at a high speed may be challenging. If a map of the environment is incomplete, then some authors consider unknown areas as occupied by obstacles (Oleynikova et al., 2020), whereas other authors consider these areas as unoccupied (Han, Gao, Zhou, & Shen, 2019). The former approach is more cautious and requires to re-plan the reference path higher frequencies. The latter approach is less cautious and requires the mapping and path planning algorithms to be fast; for additional details on mapping algorithms for sUAS, see Section 5.4 below. Reactive methods provide an alternative approach to these path planners that require a global, consistent map of the environment.

Given a reference path leading the sUAS to the goal set, *reactive path planning* methods locally modify the reference path to avoid constraints that were unknown at the time the original reference path was outlined. Re-planning can be performed by executing any of the path planning techniques surveyed thus far in neighborhoods of the sUAS that are sufficiently large to comprise a point of the original reference path, known as *aiming point*, which is past those obstacles that are about to be intercepted. A key advantage in the use of reactive methods is that they do not need global, consistent maps of the environment, but only local obstacle maps. Therefore, these algorithms are useful for those applications, wherein the sUAS is required to reach a goal set, and a consistent map of the environment traversed is unnecessary. Examples of reactive path planning methods for sUAS are presented in Berger, Rudol, Wzorek, and Kleiner (2016), Choi, Kim, and Hwang (2011), Tripathi, Raja, and Padhi (2014) and Viquerat, Blackhall, Reid, Sukkarieh, and Brooker (2008).

4.1.6. Motion primitive libraries and motion automata

Some path planning techniques for sUAS consist of selecting in real time some pre-computed *motion primitives*, that is, libraries of maneuvers such as hover, ascent and descent at a given velocity, forward flight at a given pitch angle, and turns at a given roll angle (Zhou et al., 2019a); a schematic representation of path planning techniques based on the use of motion primitives is provided in Fig. 5. Some path planning techniques consist in devising behavior primitives, that is, collections of motion primitives to perform given tasks or subtasks (Engebraaten, Moen, Yakimenko, & Glette, 2020). Upon constructing motion primitives, if the sUAS is modeled as a point mass, then by employing motion primitives, the path planning problem is cast in six-dimensions. Alternatively, if both the translational and the rotational dynamics are considered, then the path planning problem is cast in twelve-dimensions.

Path planners based on motion primitives are advantageous because finding the reference path reduces to searching sparse graphs, whose nodes are given by the motion primitives, and hence, are usually fast. Additional advantages of path planners based on motion primitives involve their ability to account for the sUAS' dynamics, and hence, produce feasible paths, incorporate a variety of constraints, and estimate the time needed to follow the reference path. However, irrespective of the technique used to search the graph, these methods are not necessarily complete since there may not exist a combination of motion primitives leading the sUAS to the goal set. To overcome this limitation, the authors in Yadav and Tanner (2020) use motion primitives to generate local goals and employ a receding horizon control framework to generate feasible trajectories.

Similar to the notion of motion primitives is the one of *maneuver automata*. For instance, in Frazzoli et al. (1999), the set of trim conditions and maneuvers generate a cost-to-go map. The reference path is then chosen by a search algorithm over this map; maneuvers falling between the pre-computed values are deduced via interpolation. Similarly, the authors in Schouwenaars et al. (2004) employ the receding horizon control approach to search maneuver automata and hence, to search spaces of dynamically feasible paths over a predefined time horizon.

4.1.7. Bio-inspired methods

Bio-inspired path planning algorithms provide a recent trend in aerial robotics (Pehlivanoglu, 2012; Yang, Fang, & Li, 2016). Some of these methods, such as *evolutionary algorithms*, allow to solve non-deterministic polynomial-time-hard (NP-hard) multi-objective path planning problems, but suffer from high time complexity. Alternative bio-inspired path planning algorithms, such as those based on *neural networks*, require training with given data (de Souza, Marcato, de Aguiar, Juca, & Teixeira, 2019) and hence, may not be suitable for complicated missions. In general, the computational cost of bio-inspired path planning algorithms, such as evolutionary algorithms, still provide a challenge to real-time path planning problems in three-dimensional environments (Yang, Qi, et al., 2016).

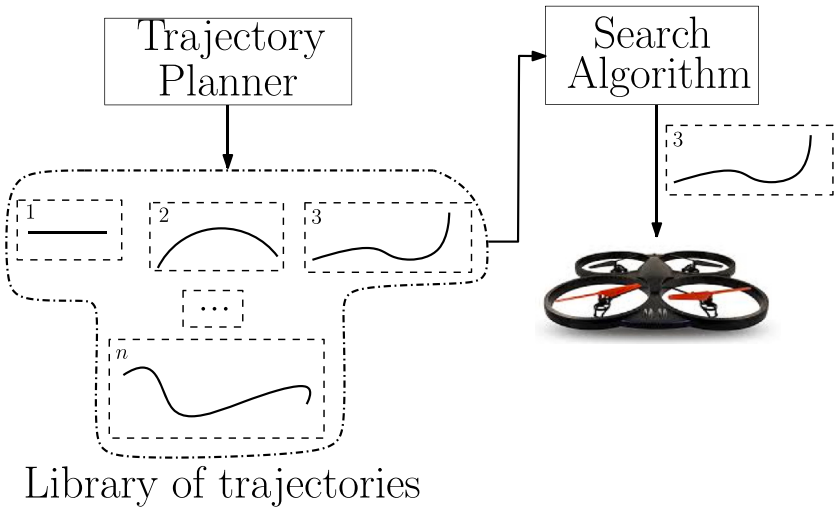

Fig. 5. An approach to path planning, which allows to account for the sUAS' dynamics, consists in generating libraries of n dynamically feasible trajectories off-line and storing these libraries on the sUAS' onboard computer. Each of these trajectories is associated with a cost index, such as the path length or the travel time. Casting each motion primitive as the arc of a weighted graph, search algorithms can be employed to determine sequences of dynamically feasible trajectories that lead the sUAS to its goal set. This approach was employed, for instance, in Han, Zhang, Pan, Xu, and Gao (2020) and Zhou, Gao, Wang, Liu, and Shen (2019b).

4.1.8. Learning-based methods

Machine learning methods such as *reinforcement learning* and *deep learning* have been utilized in recent years to improve the sampling-based path planning methods such as *RRT*. For instance, the sampling distribution in the sampling-based path planners can be learned through various machine learning methods such as a *conditional variational autoencoder* (Sohn, Lee, & Yan, 2015) or a *policy search-based method* (Zhang, Huh, & Lee, 2018). Alternatively, an incremental sampling-based motion planning algorithm based on *RRT* is presented in Li, Cui, Li, and Xu (2018), where the cost function is predicted by a neural network. Finally, an optimal path planning algorithm based on a *convolutional neural network* (CNN) and *RRT** is proposed in Wang, Chi, Li, Wang, and Meng (2020).

Some of these learning-based techniques have already been applied to path planning to sUAS. For instance, the conditional variational autoencoder has been recently employed in Xia et al. (2020). Only recently, *deep neural networks* and some deep reinforcement learning techniques, such as *Deep Q-network* (DQN) (Lv, Zhang, Ding, & Wang, 2019; Yan, Xiang, & Wang, 2020), have been employed to plan paths for sUAS. A DQN-based algorithm for optimal waypoint planning is presented in Eslamiat, Li, Wang, Sanyal, and Qiu (2019) and combined with optimal trajectory generation through waypoints and nonlinear tracking control for real-time operation of sUAS.

4.1.9. Stable path planners for sUAS

A relatively under-explored class of path planners comprises those techniques explicitly designed to account for the vehicle's stability properties. Indeed, the majority of path planning techniques are merely concerned with the problem of leading the vehicle to the goal set, while avoiding constraints. However, in the presence of dynamic constraints, sudden maneuvers may induce some instability.

Among the few methods for stable path planning known to the authors, it is worthwhile to recall (Kang et al., 2015), where the path planning problem is addressed by employing a genetic algorithm, and Norouzi, De Bruijn, and Miró (2012), which leverages the vehicle's dynamics. The use of motion primitive libraries appears a promising solution to the stable path planning problem. Indeed, motion primitives can be produced by applying some technique framed within the context of control theory, such as, for instance, one of the control laws surveyed in Section 6 below, to a suitable dynamical model of the sUAS, such as, for instance, (1) and (2), (5), (8), or (9). Since these techniques guarantee the stability of the controlled sUAS, this approach can provide a suitable framework for stable path planning, and the control input associated with the chosen motion primitive can be employed to directly regulate the sUAS. A potential limitation of this approach, however, is given by the fact that instabilities may be generated by switching across stable motion primitives and their associated control inputs (Liberzon, 2003). A solution to the problem of instabilities induced by switching mechanisms consists in finding some lower bound on the *dwell time*, that is, the minimum time between two consecutive switches across motion primitives.

4.2. Trajectory planners for sUAS

The large majority of mission planning problems involving sUAS are characterized by multiple constraints, such as saturation of the control input, which are intrinsically tied with the vehicle's dynamics. In these cases, analytical solutions to the trajectory planning problem are usually impossible to find, and numerical techniques must be employed to determine feasible reference trajectories. The trajectory planning problem becomes further challenging for sUAS operating in unknown, dynamic, or unstructured environments since low computational times and high precision are required, while these vehicles' communication capabilities are too limited to compute feasible trajectories in real time off-board. Presently, COTS autopilots, such as those listed in Table 1, could implement simple trajectory planners, such as linear MPC laws. However, in general, trajectory planning algorithms need to be executed on single-board computers such as those listed in Table 3. Additionally, despite the very latest advances in the design and production of single-board computers for small mobile robots, the computational capabilities of existing sUAS are too low to implement complex solvers. Therefore, the problem of computing realistic reference trajectories at a high speed and in complex environments is still considered as open.

Thus far, the trajectory planning problem has been substantially addressed from a control-theoretic perspective by considering a plant model, which is sufficiently representative of the sUAS' dynamics, and then finding a control input that allows the sUAS to reach the goal set, while meeting collision avoidance constraints (see Sections 5.4 and 5.5 below), constraints on the control input, and user-defined constraints on the flight envelope. Applying this approach, the state of the plant model associated with this control input is considered as the reference trajectory. Table 5 summarized some of the results on trajectory planning for sUAS surveyed in this section and highlights some of their key properties.

Sidebar 4.1. In some cases, trajectory planners are designed by assuming that there exist control forces that allow to translate the sUAS along any direction without considering how these forces can be produced. In these cases, the sUAS' rotational dynamics are ignored and only reference trajectories for the sUAS position are produced. This approach guarantees a higher computational speed. However, as discussed in Section 6.1, this approach also requires the control system to determine the orientation needed to steer the total thrust force and hence, track the sUAS' reference position. Furthermore, the sUAS' rotational dynamics may not allow to realize these forces at each time instant.

Table 5
Some of the most common trajectory planning methods for sUAS, some of their most remarkable properties, such as optimality and compatibility with COTS single-board computers for sUAS, and selected recent publications.

Algorithm	Optimal	Real-Time	Notes	Ref.
MPC	✓	✓	Very popular	Koo, Kim, and Suk (2015), Prodan et al. (2013) Kamel, Burri, and Siegwart (2017), Tang, Wang, Liu, and Wang (2021)
Dynamic window	✓	✓	Produces cont. diff. trajectories	Moon, Lee, and Tahk (2018)
Calculus of Variations	✓		Real-time solutions for simple models	Kaminer et al. (2012), L'Afflitto and Sultan (2010)
Reference Governor		✓	Add-on component	Hermand, Nguyen, Hosseinzadeh, and Garone (2018), Nicotra, Naldi, and Garone (2016)
Learning-based method		✓	To improve classical methods	Almeida, Moghe, and Akella (2019), Jardine, Givigi, and Yousefi (2017a)

In general, trajectory planners are designed by considering plant models that account for both the translational and the rotational dynamics of the sUAS. In rare cases, trajectory planners model the sUAS as point masses and hence, only account for their translational dynamics. These models, however, do not explicitly account for the fact that, to translate in the inertial horizontal plane, the quadcopter sUAS must pitch and roll and, hence, these reference trajectories may not be dynamically feasible.

It is worthwhile to remark that trajectory planners do not need to be *explicit* functions of the actual sUAS' state vector or measured output, but can be exclusive functions of the state of an internal model. In some cases, the control input produced as part of this process is merely discarded, while delegating the control system with the task of following the reference trajectory. In other cases, as discussed in Section 6.1 below, the control input computed by the trajectory planner is employed as a baseline controller for the control system. Finally, in some architectures, the control input computed by the trajectory planner is directly used to regulate the sUAS, and a control system is not employed. However, this approach may produce unsatisfactory results as this control input may be computed accounting for the state of a virtual model of the sUAS or the sUAS' state at the time the reference trajectory is computed and not for the sUAS' actual state at the time it is applied.

4.2.1. Receding horizon control and model predictive control

Until recently, optimal control has provided the only framework able to *explicitly* account for *hard constraints*, that is, equality and inequality constraints on both the state and the control vectors that must be verified. For this reason, the overwhelming majority of trajectory planners for sUAS are still based on some optimal control strategy.

Consider the user-defined cost function

$$J_{t_0,n_t\Delta T}[x_0,u(\cdot)] \triangleq h(t_0+n_t\Delta T, x_{\text{ref}}(t_0+nt\Delta T)) + \int_{t_0}^{t_0+n_t\Delta T} \mathcal{L}(t,x_{\text{ref}}(t),u(t))\mathrm{d}t \tag{11}$$

subject to (5) with $x(t)=x_{\text{ref}}(t)$, $t\in[t_0,t_0+n_t\Delta T]$, and

$$g(t,x_{\text{ref}}(t),u(t)) \leq\leq 0_l, \tag{12}$$

where $n_t \in \mathbb{N}$ and $\Delta T > 0$ are user-defined, $h : \mathbb{R}\times\mathbb{R}^{12} \to \mathbb{R}$ is continuous, $\mathcal{L} : [t_0,t_0+n_t\Delta T]\times\mathbb{R}^{12}\times\mathbb{R}^4 \to \mathbb{R}$ is integrable, $g : [t_0,t_0+n_t\Delta T]\times\mathbb{R}^{12}\times\mathbb{R}^4 \to \mathbb{R}^l$ is continuous, the partial order relation operator $\leq\leq$ captures component-wise inequalities, and 0_l denotes the zero vector in $\mathbb{R}^l$; the constraint inequalities (12) capture, for instance, collision avoidance constraints, saturation constraints, and constraints on the trajectory at $t_0+n_t\Delta T$. Within an optimal control framework, the trajectory planning problem can be formulated as finding both $u^*(\cdot)$ in the set of admissible control inputs $\mathcal{U}$ over $[t_0,t_0+n_t\Delta T]$ and the corresponding trajectory $x^*_{\text{ref}}(\cdot)$ such that (11) is minimized and (5) with $x(t)=x^*_{\text{ref}}(t)$, $t\in[t_0,t_0+n_t\Delta T]$, are verified; a typical example of set of admissible control inputs $\mathcal{U}$ is given by the set of continuous functions over a given time interval.

Due to the technological limitations still affecting single-board computers, some techniques to solve the optimal control problem given by (11) subject to (5) with $x(t)=x_{\text{ref}}(t)$, $t\in[t_0,t_0+n_t\Delta T]$, and (12), such as *direct* and *indirect multiple shooting methods* (Rao, 2014), *direct* and *indirect transcription methods* (Kelly, 2017), and *level set methods* (L'Afflitto, 2017) are still considered as difficult or impossible to execute in real time on sUAS. A feasible optimization-based approach to optimal trajectory planning for sUAS is provided by MPC (Koo et al., 2015), also known as *receding horizon control* (Prodan et al., 2013), whose general principles can be illustrated as follows. According to the *principle of optimality* (Bryson, 1975, Ch. 14), $u^*(\cdot)\in\mathcal{U}$ and the corresponding trajectory $x^*_{\text{ref}}(\cdot)$ are optimal over $[t_0,t_0+n_t\Delta T]$ with respect to (11) subject to (5) with $x(t)=x_{\text{ref}}(t)$ and (12) if and only if $u^*(\cdot)\in\mathcal{U}$ and $x^*_{\text{ref}}(\cdot)$ are optimal over $[t_1,t_0+n_t\Delta T]$ with respect to $J_{t_1,n_t\Delta T}[x_0,u(\cdot)]$ subject to (5) with $x(t)=x_{\text{ref}}(t)$ and (12), where $t_1\in[t_0,t_0+n_t\Delta T]$. The MPC framework exploits the principle of optimality by recomputing both $u^*(\cdot)$ and $x^*_{\text{ref}}(\cdot)$ at the time step $t_0+k\Delta T$ over the time interval $[t_0+k\Delta T, t_0+n_t\Delta T]$ for all $k\in\{0,1,\dots,n_t-1\}$; Fig. 6 provides a graphical representation of the MPC framework. This approach allows to compute reference trajectories over relatively short time horizons $n_t\Delta T$ and account for unpredictable changes in the environmental conditions.

Earlier implementations of MPC were only applicable to slow processes due to their high computational costs, and involved quadratic

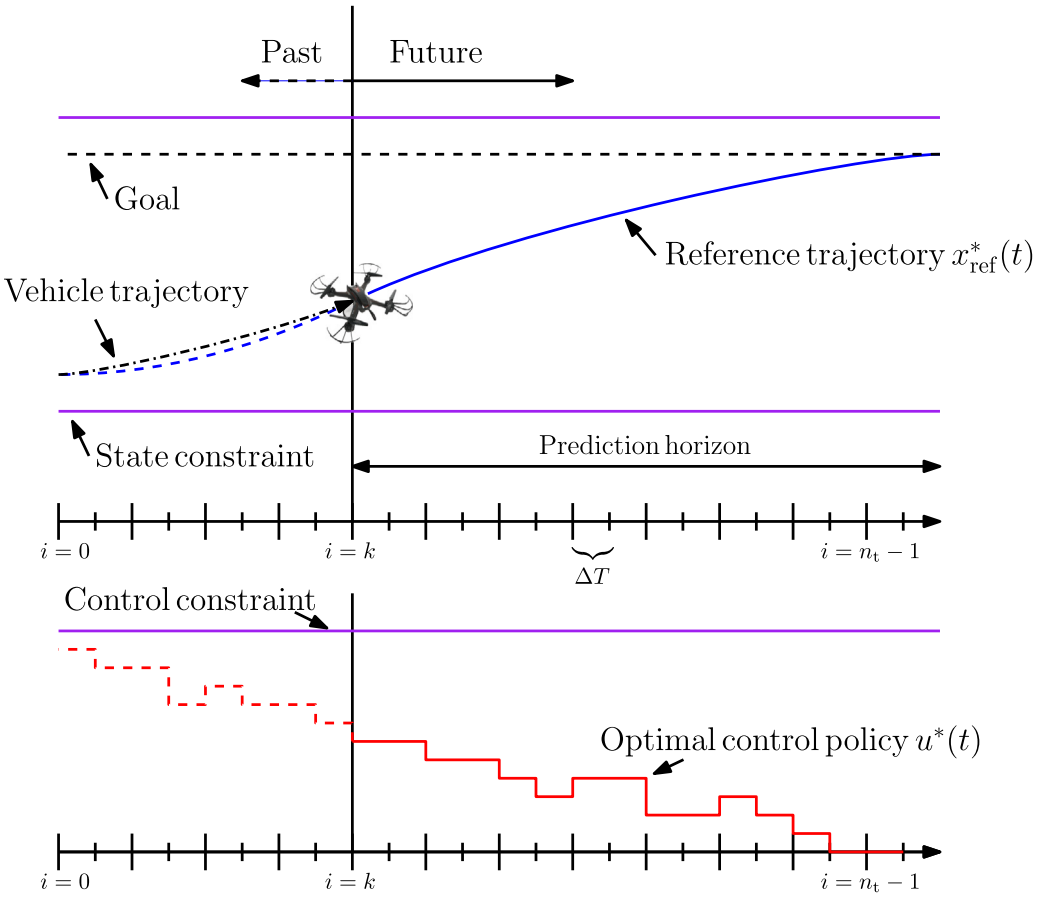

Fig. 6. Graphical representation of the MPC framework. At each time step $t_0+k\Delta T$, where $k \in \{0, 1, \dots, n_t-1\}$, the cost function is optimized over the time interval $[t_0+k\Delta T, t_0+n_t\Delta T]$ subject to constraints on the trajectory, the control input, and the sUAS' dynamics. The result is an optimal control policy $u^*(t)$, which drives the system to a neighborhood of the goal state by the terminal time $t_0 + n_t\Delta T$, and an optimal trajectory $x^*_{\text{ref}}(t)$.

Sidebar 4.2. The problem of identifying admissible control inputs for an optimal control problem can not be under-estimated. Indeed, if $\mathcal{U}$ is overly small, then there may not exist a solution to the Hamilton-Jacobi equation underlying the given optimal control problem. For instance, if $\mathcal{U}$ captures the set of continuous functions, it may be impossible to find $(x^*_{\text{ref}}(\cdot), u^*(\cdot))$ minimizing (11) subject to (5) and (12). A viable solution is to set $\mathcal{U}$ as the set of integrable functions. However, in this case, realizing $(x^*_{\text{ref}}(\cdot), u^*(\cdot))$ may be difficult or impossible.

cost functions subject to linear equality and inequality constraints, that is, addressed the discrete-time problem of minimizing

$$\begin{aligned}\hat{J}_{t_0,n_t\Delta T}[\tilde{x}_0, \hat{u}(\cdot)] \triangleq\ & \hat{x}^{\mathrm{T}}_{\text{ref}}(t_0 + n_t\Delta T) Q_f \hat{x}^{\mathrm{T}}_{\text{ref}}(t_0 + n_t\Delta T) \\ &+ \sum_{i=0}^{n_t-1} \Big[\hat{x}^{\mathrm{T}}_{\text{ref}}(t_0 + i\Delta T) Q \hat{x}_{\text{ref}}(t_0 + i\Delta T) \\ &\quad + 2\hat{x}^{\mathrm{T}}_{\text{ref}}(t_0 + i\Delta T) S \hat{u}(t_0 + i\Delta T) \\ &\quad + \hat{u}^{\mathrm{T}}(t_0 + i\Delta T) R \hat{u}(t_0 + i\Delta T)\Big] \end{aligned} \tag{13}$$

subject to (9) and

$$\hat{H} \begin{bmatrix} \hat{x}_{\text{ref}}(t_0 + i\Delta T) \\ \hat{u}(t_0 + i\Delta T) \end{bmatrix} \leq\leq \hat{h}, \tag{14}$$

where $Q \in \mathbb{R}^{12\times12}$ is symmetric, $S \in \mathbb{R}^{12\times4}$, $R \in \mathbb{R}^{4\times4}$ is symmetric and positive-definite, $Q - SR^{-1}S^{\mathrm{T}}$ is symmetric and positive-definite, $Q_f \in \mathbb{R}^{12\times12}$ is symmetric and positive-semidefinite, $\hat{H} \in \mathbb{R}^{l\times16}$, and $\hat{h} \in \mathbb{R}^l$. Recent approaches transform linear–quadratic optimal control problems from the MPC framework into the quadratic programming framework and exploit the structural properties of sparse or banded matrices to produce reference trajectories in real time at computational costs that are affordable by single-board computers compatible with fast sUAS (Wright, 2019).

The prediction horizon for a sUAS is typically 10s or less when planning a trajectory for the sUAS' translation, and less than 1s when planning a trajectory for the sUAS' orientation (Jardine et al., 2017a); this is due to the fact that sUAS' rotational dynamics is faster than their translational dynamics (L'Afflitto et al., 2018). Recently, a MPC architecture for sUAS has been implemented with a relatively long time horizon of 25s (Ladosz, Oh, & Chen, 2018). In general, the duration of the prediction horizon is chosen compatibly with the vehicle's dynamics and the computational resources available. Additionally, mission

Sidebar 4.3. The principle of optimality underlies numerous key results in optimal control theory. For instance, this principle is pivotal in the proofs of Pontryagin's principle and Bellman's principle. Moreover, the principle of optimality underlies numerous optimization-based techniques. For instance, as it appears from Algorithm 1, the A^* algorithm is the result of a direct implementation of the principle of optimality.

Sidebar 4.4. The principle of optimality can be interpreted as follows. To obtain the best result, the ideal course of actions needs to be taken at each time step. However, if some past action proved to be sub-optimal, then one needs to continue executing their best plan while accounting for the current situation. There is some common wisdom in this concept.

demands must be accounted for when choosing the length of the prediction horizon. For instance, in unknown dynamic environments, short prediction horizons are preferable, whereas in known static environments, long prediction horizons are recommended. The prediction horizon does not need to be considered as constant as it can be varied by modifying n_t and ΔT.

A limitation of linear MPC techniques rests in the fact that the constraint set given by (14) must be convex with affine boundaries. Although constraints on the control input are usually in the form of box constraints (Hehn & D'Andrea, 2015), obstacles rarely meet these assumptions and, hence, linear MPC algorithms can serve as local trajectory planners only. Furthermore, algorithms to produce convex search spaces in real time must be implemented to serve problems of practical interest; for additional details, see Section 5.5 below. Recently, dynamically feasible collision-free trajectories in the receding horizon control framework have been produced using B-splines and a nonuniform kinodynamic search algorithm (Tang et al., 2021).

The advent of fast nonlinear solvers has now enabled the implementation of *nonlinear MPC* algorithms on sUAS (Pereira, Leite, & Raffo, 2021), that is, algorithms to find a sequence $\{(x^*_{\text{ref}}(t_0+k\Delta T), u^*(t_0+k\Delta T))\}_{k=0}^{n_t} \subset \mathbb{R}^{12}\times\mathbb{R}^4$ that minimizes the cost function

$$I_{t_0,n_t\Delta T}[x_0,u(\cdot)] \triangleq h(t_0+n_t\Delta T, x_{\text{ref}}(t_0+nt\Delta T)) + \sum_{k=0}^{n_t}\mathcal{L}(t,x_{\text{ref}}(t_0+k\Delta T),u(t_0+k\Delta T)) \tag{15}$$

subject to

$$x_{\text{ref}}(t_0+(k+1)\Delta T) = x_{\text{ref}}(t_0+k\Delta T) + \hat{f}(t,x_{\text{ref}}(t_0+k\Delta T)) + \hat{G}(x_{\text{ref}}(t_0+k\Delta T))u(t_0+k\Delta T), \quad k\in\{0,\dots,n_t-1\}, \tag{16}$$

and

$$g(t,x_{\text{ref}}(t_0+k\Delta T),u(t_0+k\Delta T)) \leq\leq 0_l, \qquad k\in\{0,\dots,n_t\}, \tag{17}$$

where $\hat{f}(\cdot,\cdot)$ and $\hat{G}(\cdot)$ are obtained by discretizing $f(\cdot,\cdot)$ and $G(\cdot)$ in (5), respectively. The use of nonlinear MPC algorithms appears to be computationally justified in the case that high-precision aggressive maneuvers need to be performed (Kamel et al., 2017).

The asymptotic stability of a MPC scheme is typically guaranteed by proving the existence of a bound on the optimal value function at the end of the planning horizon, and proving boundedness of the running cost for all feasible trajectories; indeed, a tight connection between optimal value functions and some Lyapunov-like function certifying the stability properties of the controlled system have been proven in Bernstein (1993), Haddad and L'Afflitto (2016) and L'Afflitto, Haddad, and Bakolas (2016) for multiple classes of optimal control problems. Furthermore, recursive stability of a MPC scheme, that is, the existence of an admissible control at every step along the planning horizon, is an important property closely related to the stability of the multi-step feedback law. The authors in Grüne (2012) provide a survey on the analysis of nonlinear MPC without terminal constraints and proves exponential convergence of *economic* nonlinear MPC, that is, nonlinear MPC with zero weighting on the plant state in the cost function (15). A survey on the stability results of *distributed* MPC and economic MPC is provided in Müller and Allgöwer (2017). The asymptotic stability and recursive feasibility of a system controlled by a nonlinear MPC system without constraints or terminal costs is shown in Grüne (2009). Finally, the authors in Faulwasser and Findeisen (2015) present the convergence conditions for constrained path-following using nonlinear MPC with no velocity assignment.

Recently, automata have been applied to optimization-based trajectory planning systems. For instance, *learning automata* have been employed within a MPC framework to select the weighting parameters of the objective function (Jardine, Givigi, & Yousefi, 2017b).

As a concluding remark, it is worthwhile to note that the MPC framework is often presented within the context of control systems, and not trajectory planning systems, for sUAS. This is due to the fact that MPC algorithms produce both a reference trajectory and a control input, which is either employed as a baseline controller or is directly applied to regulate the sUAS' dynamics (Castillo-Lopez, Sajadi-Alamdari, Sanchez-Lopez, Olivares-Mendez, & Voos, 2018; Garimella, Sheckells, & Kobilarov, 2017; Hafez, Givigi, & Yousefi, 2018; Iskander, Elkassed, & El-Badawy, 2020; Koo et al., 2015; Lin, Chen, & Liu, 2016; Pereira et al., 2021; Shekhar, Kearney, & Shames, 2015); for additional details, see Sections Section 6.1, 6.4, and 6.5 below. This approach is advantageous since it allows to synchronize the control input and the reference trajectory. Conversely, relying on the input produced by the autopilot's control law or a dedicated algorithm executed on the sUAS' single-board computer poses the challenge of synchronizing the reference trajectory and the control input and is prone to delays in the overall architecture.

4.2.2. Alternative variational methods

Available approaches to the trajectory planning problem, which are not based on the MPC framework, but are still framed within the context of variational methods, involve finding numerical solutions of the Hamilton, D'Alembert, or Lagrange equations (Junge, Marsden, & Ober-Blöbaum, 2005; Marsden & West, 2001). Similarly to the MPC framework, these approaches produce both a reference trajectory and

Sidebar 4.5. The robustness of control inputs computed by applying the model predictive control algorithms derives from the fact that the control input is recomputed at each time step and the vehicle's state vector is measured at relatively high frequency, but is not formally guaranteed by Lyapunov or energy-like arguments.

an associated control input, which can be directly used to steer the sUAS or can serve as a baseline controller. Despite the MPC framework, these approaches do not need not be recomputed at each time step. However, if the underlying control input is directly applied to steer the sUAS, then these algorithms should be executed at a high frequency.

The *dynamic window approach* exploits an optimal control framework to produce feasible trajectories subject to acceleration constraints each time a new obstacle is detected (Moon et al., 2018). This approach is particularly appealing for its ability to produce continuously differentiable trajectories while controlling the trade-off between precision and computational time.

An energy-optimal trajectory planning technique is presented in (Nikhilraj, Simha, & Priyadarshan, 2019), where the Lie Group Variational Integrator is employed to discretize the sUAS' equations of motion on a manifold and the solution to a discrete-time optimal control problem is found using neighboring extremal method (Ghaemi, Sun, & Kolmanovsky, 2010) and the Newton–Armijo algorithm (Kelley, 1995, Ch 8). However, the computational complexity of this method makes it challenging to embed on sUAS for online trajectory planning.

Calculus of variations also provides a valuable tool for trajectory planning. For instance, the authors in Kaminer et al. (2012) and L'Afflitto and Sultan (2010) employ this classical approach to produce reference trajectories for formations of sUAS.

Leveraging the differential flatness property of quadcopters, which is discussed in detail in Section 6.2 below, the authors of Cowling, Yakimenko, Whidborne, and Cooke (2010) produce optimal reference trajectories by solving in real time a nonlinear constrained optimization problem. The key advantage of this approach is that the trajectory planning problem is solved in the output space in a virtual domain, not explicitly related to the time domain.

4.2.3. Reference governors for trajectory planning

An approach to design trajectory planners involves the application of a control law to a suitable model of the sUAS' dynamics, without accounting for constraints on the state and control vectors. Then, this control system is augmented by a *reference governor*. Reference governors are add-on systems designed so that if a constraint on the plant state or control input is about to be violated, then the signal fed to the control system is modified such that all constraints are verified. The state of the sUAS' dynamical model controlled by the original control law and the reference governor is employed as the reference trajectory; Fig. 7 provides a representation of this approach.

A substantial advantage of reference governors is that they maintain the stability properties provided by the control law designed for the unconstrained problem (Bemporad, 1998; Gilbert, Kolmanovsky, & Tan, 1995; Hermand et al., 2018; Nicotra et al., 2016). This consideration lead the authors of Arslan and Koditschek (2017) to devise a provably correct approach to extend the applicability of low-order feedback motion planners, that is, planners for which a robot's position and velocity are controlled directly, to high-order robot planners, that is, planners for which the robot's motion is controlled by forces and moments, while retaining stability and collision avoidance properties. A key result in Arslan and Koditschek (2017) consists of using reference governors to separate the problems of stability and constraint enforcement.

4.2.4. Learning-based methods

Recently, reinforcement learning and deep learning approaches started playing major roles in learning-based trajectory planning of sUAS (Hsu & Gau, 2020; Samir, Ebrahimi, Assi, Sharafeddine, & Ghrayeb, 2020). Some learning-based trajectory optimization methods are proposed in Choi, Jimenez, and Mavris (2017) to generate feasible trajectories of a single sUAS, while other learning-based trajectory planning approaches focus on sUAS networks with collision avoidance and communication constraints (Challita, Saad, & Bettstetter, 2019).

Machine learning methods have also been utilized to enhance the performance of some classical trajectory planning techniques surveyed so far. For instance, in Almeida et al. (2019), a supervised neural network is trained offline to accelerate online generation of minimum snap trajectories, that is, trajectories with minimum second derivative of the acceleration, for a quadcopter. A machine learning method is used in Jardine et al. (2017a) to tune the parameters in a MPC algorithm for collision-free trajectory planning of a quadcopter. Finally, merging nonlinear MPC and some learning mechanism has proven useful to account for poor modeling of the vehicle's dynamics, the effect of unknown payloads, or fault and failures (Hafez et al., 2018).

4.3. Integration of path and trajectory planning for autonomous sUAS

A relatively recent research trend involves integration of path and trajectory planning systems, beyond the use of motion primitive libraries and motion automata discussed in Section 4.1.6. For instance, in Han et al. (2020), a hybrid-state A^* algorithm is employed and smooth, time-optimal trajectories are generated by computing double descriptions of polynomials for tracking agile targets. In this work, the finite union of cubes is used to model the free space in the environment, and barrier functions are used to ensure that the time-optimal trajectories lie within the free space. Similarly, the authors of Zhou et al. (2019b) employ a hybrid-state A^* algorithm and adopt a receding horizon planning framework with a B-spline trajectory optimization method, which penalizes the planned trajectories' proximity to obstacles and uses a time adjustment algorithm to ensure constraints on the maximum acceleration and velocity are met. This path searching method, however, guarantees neither optimality nor completeness.

The authors of Richter, Bry, and Roy (2016) employed RRT^* to generate a basis of polynomials as the analytic solution of a constrained quadratic program together with a time allocation algorithm to generate feasible trajectories. Collisions in a trajectory segment are resolved by adding intermediate waypoints and re-optimizing. RRT^* is also employed in Iskander et al. (2020), where minimum-snap trajectories are determined using the nonlinear MPC framework.

4.4. Guidance systems for swarms of sUAS

The problem of planning paths and trajectories for swarms of sUAS has been addressed in multiple ways. The simplest approach consists of carefully designing a collision-free reference path or reference trajectory for each vehicle at a centralized level, and informing each agent of their reference trajectory. This approach is computationally expensive since a central unit is tasked with computing a path or a trajectory for all agents. Furthermore, this framework relies on the assumption that, if needed, these collision-free reference paths or trajectories can

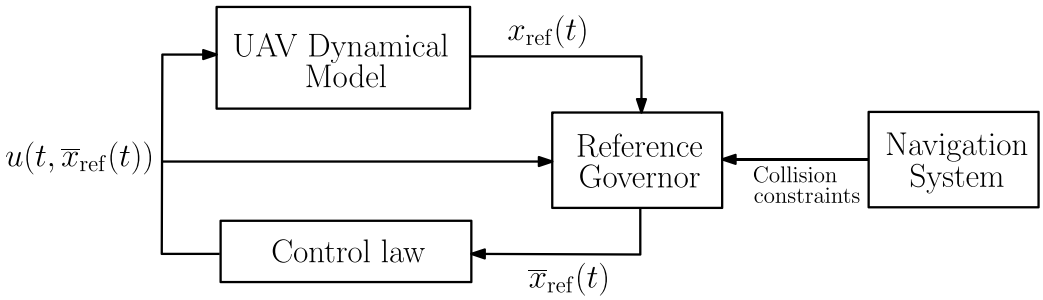

Fig. 7. Graphical representation of a trajectory planner designed by employing a reference governor. In this approach, any control law can be applied to a dynamical model of the sUAS to meet user-defined performance criteria such as reaching a given goal set and imposing asymptotic stability of the closed-loop system. This control law can be designed without accounting for constraints on the model's state and the control input. Then, a reference governor is introduced to modify the signal fed to the control law whenever a constraint is about to be violated. The state of the closed-loop dynamical model is the reference trajectory $x_{\text{ref}}(t)$, $t \geq t_0$. Assuming that the control law is a function of $x_{\text{ref}}(t)$, $t \geq t_0$, the signal $\overline{x}_{\text{ref}}(t)$ fed to the control law is modified by the reference governor to account for approaching collision avoidance constraints, which are deduced from the navigation system, and saturation constraints.

be readily recomputed and transmitted to all agents. Lastly, this central unit must rely on an updated map of the environment; for details of SLAM for swarms of sUAS, see Section 5.2.3 below. However, swarms of hundreds or thousands of sUAS may not have the computational and communication capabilities to meet these requirements.

The *leader-follower* approach provides a viable solution to path and trajectory planning for swarms of sUAS. In this framework, any of the techniques surveyed in Sections Section 4.1 or 4.2 could be employed to design a reference path or a reference trajectory for one of the agents, namely the leader, or for a virtual point that serves as the swarm leader. Then, all followers are tasked with maintaining a given a position relative to the leader. Collision avoidance with obstacles can be guaranteed by enforcing sufficiently large safety margins and, if needed, reshaping the formation. This approach, however, still requires to inform all agents of the leader's reference trajectory or to detect the leader's position at all time.

The *virtual structure* approach provides a popular alternative solution to the leader-follower technique. In this framework, the swarm is considered as a virtual structure, that is, a collection of sUAS that must maintain a given geometric relationship to each other and a reference frame Tan and Lewis (1996). Thus, a path or trajectory is outlined for the virtual structure, which is considered as a single body, by applying any of the techniques surveyed thus far. Finally, the reference path or trajectory for each element of the virtual structure is deduced from the trajectory of the virtual structure and the geometric relation between all agents (Zhou, Wang, & Schwager, 2018). Also, this approach relies on the assumption that all swarm members are informed in real time of the reference path or trajectory of the virtual structure and that a central unit is able to elaborate in real time each agent's information.

In their original formulations, both the leader-follower approach and the virtual structure approach require the swarm's shape to be rigid. These architectures can be modified by adapting the formation's shape to the assigned task or to avoid collisions with obstacles. However, in this case, also the shape of the swarm must be decided by a central unit. A more fluid approach to the guidance problem for swarms is given by the notion of *flocking* (Reynolds, 1987). While flocking, each agent must follow relatively simple rules that involve attraction and repulsion of agents, relative heading, and speed agreement. A recent example of outdoor flocking involving multi-rotor sUAS has been presented in Vásárhelyi et al. (2018).

4.5. Guidance system for autonomous tactical sUAS

A relatively new area of research concerns the design of guidance systems for autonomous tactical sUAS, that is, sUAS moving at low altitudes along trajectories of low visibility to observers. The interest in this area, however, is increasing steadily since sUAS are already employed by ground troops and law enforcement agencies to gather and relay data or serve logistic purposes. Indeed, in 2020, the US Federal Aviation Administration provided guidelines for waivers to regulations on the use of sUAS by first responders (*First responder tactical beyond visual line of sight (TBVLOS) 91.113 waiver guide*, 2021).

Recently, a guidance system for stealthy sUAS has been presented in Zhang, Wu, Dai, and He (2020), where, assuming that the opponent's location is known by means of netted radar systems, and the performance of several cell-decomposition based path planning algorithms are compared. The guidance system presented in Zhang et al. (2020) employs a classical MPC algorithm for trajectory planning. An alternative guidance system for sUAS attempting to reach a given location, while minimize their exposure to radar systems is presented in Chaudhry, Misovec, and D'Andrea (2004). This approach, however, assumes exact knowledge of the radar's position, and is unsuitable for real-time implementation due to the large number of constraints and variables needed for realistic mission scenarios in the mixed integer linear programming framework. Furthermore, this framework is sensitive to uncertainties in the underlying models (Radmanesh, Kumar, Guentert, & Sarim, 2018).

To induce a stealthy behavior in sUAS, the authors of Geraerts and Schager (2010) produced corridor maps by computing visibility polygons describing the field of view of the observers, and use an A^*-based path planning algorithm. Very recently, a method for covert tracking of a ground vehicle from a sUAS was developed in Huang, Savkin, and Ni (2020). Finally, the authors of Marzouqi and Jarvis (2003) introduced a two-dimensional guidance system for covert autonomous robots that measures the robot's visibility in obstacle-free partitions of a map, and generates a collision-free path that minimizes visibility. This method has yet to be employed on a multi-rotor sUAS. The results in Geraerts and Schager (2010), Huang et al. (2020) and Marzouqi and Jarvis (2003) rely on *a priori* information about the environment, and do not explicitly consider the cases wherein the vehicle is in an unexplored area.

A guidance system for sUAS operating in unknown dangerous areas is provided in Quan, Zhang, Xu, and Gao (2020). In this work, the guidance system employs an environmental safety awareness method to assess the danger of the local area and a MPC framework for its path planning and dynamic adjustment of speed for fast maneuvering in unknown environments.

4.6. Trajectory planners for fast sUAS

Moving at the highest speed possible is a technique to operate safely in hostile environments, and multiple guidance systems have been recently designed to allow sUAS to operate at high speed in unknown, cluttered environments (Deits & Tedrake, 2015b; Richter et al., 2016; Tal & Karaman, 2020; Tordesillas et al., 2019; Zhou et al., 2019b; Zhou, Pan, Gao, & Shen, 2021). Among these systems, it is worthwhile

giving special emphasis on FASTER (Tordesillas et al., 2019) and RAPTOR (Zhou, Boyu et al., 2021). The FASTER guidance system relies on jump point search for global planning, a convex decomposition toolbox to create convex regions of free space near the planned path (Liu et al., 2017), and mixed-integer quadratic programming for local planning (Tordesillas et al., 2019). The RAPTOR guidance system, whose path planner is based on Zhou et al. (2019b) and whose volumetric mapping is based on Han et al. (2019), was proposed to improve the robustness and safety of fast flight in unknown environments. In Zhou, Boyu et al. (2021), the authors focus on the re-planning strategy, and develop a path-guided trajectory optimization problem and trajectory refinement algorithm, which ensures the visibility of relevant unknown space from the planned trajectory and enforces a safe distance from obstacles.

It is worthwhile to briefly survey some key performance parameters of the trajectory planners presented in Deits and Tedrake (2015b), Richter et al. (2016), Tal and Karaman (2020), Tordesillas et al. (2019), Zhou et al. (2019b) and Zhou, Boyu et al. (2021), namely, the ability of these algorithms to be implemented in real time, their robustness to variations in the environmental conditions, and some statistical measure of their success rate, which may serve as a measure of their stability. The trajectory planning algorithms presented in Richter et al. (2016), Tal and Karaman (2020), Tordesillas et al. (2019), Zhou et al. (2019b) and Zhou, Boyu et al. (2021) produce solutions within time intervals comprised between 3 ms and 250 ms, on average, while executed on single-board computers such as those listed in Table 3. Therefore, these algorithms can be implemented in real time on sUAS. The algorithm presented in Deits and Tedrake (2015b) produces reference trajectories at a rate, in average, less than 1 Hz and, hence, should only be implemented in known or slowly changing environments. The authors of Tordesillas et al. (2019) report a 100% success rate of FASTER in 10 randomly generated simulation scenarios, and the authors of Zhou, Boyu et al. (2021) report a 100% success rate of RAPTOR in 10 simulated maps with 5 different obstacle densities. In Richter et al. (2016), the stability and robustness of the proposed trajectory planner was demonstrated by a 100% success rate over 20 simulations, where the locations of intermediate waypoints and segment times were randomly generated. The authors of Zhou et al. (2019b) demonstrated robustness and stability of their trajectory planning algorithm through two sets of real world experiments. The first set of experiments demonstrated a computational speed of 0.001s for the back-end trajectory optimizer of the proposed trajectory planner and, compared to the authors' previous work, faster convergence of the normalized objective function value to zero. Furthermore, the success rate of the authors' kinodynamic path searching algorithm was shown for varying voxel map resolutions. The success rate decreases and the control cost increases with decreasing voxel map resolution. The authors of Deits and Tedrake (2015b) demonstrated robustness through a simulation campaign in which the number of obstacles was increased with each test. Over three tests with increasing number of obstacles and 20 simulations per test, the trajectory planner proposed in Deits and Tedrake (2015b) demonstrated average computational time of less than 25s for long planning horizons with several waypoints and success rates greater than 50%. However, despite the aforementioned performances, which are highly remarkable for trajectory planners implemented on fast sUAS, solely relying on speed as a mechanism to enable a tactical behavior is not an effective strategy in the case the sUAS must be concealed from electromagnetic sensors such as cameras, radars, and LiDARs.

5. Navigation systems for autonomous multi-rotor sUAS

Navigation can be described in a broad sense as the determination of the vehicle's state, that is, its position, velocity, orientation, and angular velocity, at a given point in time. In most applications of practical interest, autonomous sUAS operate in dynamic, unknown environments. Therefore, autonomous sUAS must recognize objects or track landmarks, produce in real time maps of their surroundings, that is, discretized representations of the portions of the space occupied by obstacles, and finally localize themselves within this map to achieve a navigation solution.

Presently, both *Kalman filter*-based techniques and *SLAM* are the most prominent navigation techniques for autonomous sUAS. By employing a Kalman filter, or one of its numerous variations, it is possible to deduce the sUAS' state vector from data produced by the onboard sensors, such as the IMU or the GNSS module. Using SLAM, a map of the environment is created by adding the key features perceived by the sensors onboard the sUAS, and concurrently the aircraft localizes itself in the map by estimating its relative position and pose (Durrant-Whyte & Bailey, 2006). Hence, the main objectives of SLAM are to obtain a precise mapping of the unknown surrounding environment and localize the robot in the map in real time. This section surveys both Kalman filter-based navigation techniques and SLAM techniques for sUAS.

A less popular approach to indoor navigation for sUAS is given by SONAR- and other sound-based techniques. These techniques are particularly valuable whenever the light reflected by the obstacles surrounding the sUAS does not provide meaningful information to visual or LiDAR SLAM systems. This section surveys these techniques as well.

Mapping, that is, capturing and storing information about the environment surrounding a sUAS plays a key role in any navigation system, irrespectively of how information on nearby obstacles is acquired. Furthermore, these maps need to be exploited to deduce collision-free sets, where path and trajectory planning algorithms can search feasible solutions. In this section, we present some of the most popular mapping techniques for sUAS equipped with cameras or LiDARs, and survey some of the latest and most efficient collision avoidance algorithms for sUAS.

Presently, COTS autopilots, such as those listed in Table 1, employ a Kalman filter, or a variation thereof, to leverage data produced by a GNSS unit and their IMUs and estimate the sUAS' position and attitude. However, the SLAM and SONAR-based navigation techniques and the mapping techniques listed in this section must be executed on single-board computer such as those listed in Table 3.

5.1. Kalman filter-based techniques for state estimation

The Kalman filter (Grewal & Andrews, 2015) is one of the most common algorithms to estimate unknown state variables, based on measurements subject to uncertainty. The measurement uncertainty in GPS and IMU sensors onboard the sUAS is induced by measurement noise, atmospheric effects, and calibration errors. The Kalman filter and its variants have been extensively applied to estimate the state of vector of a sUAS using measured data despite these effects. Some noticeable variations of Kalman filter that can be applied to nonlinear sUAS' models include the extended Kalman filter (EKF) (Mao, Drake, & Anderson, 2007), which linearizes the system about the mean and covariance estimates of the state, the unscented Kalman filter (UKF) (De Marina, Pereda, Giron-Sierra, & Espinosa, 2011), which samples points of the state around the mean value and tracks the propagation of these points to estimate mean and covariance of the state, and the adaptive Kalman filter (Hajiyev & Soken, 2013), which adapts parameters to the system's behavior during data processing.

Besides state estimation, Kalman filters are also utilized to address the SLAM problem. For instance, the EKF-SLAM (Cheeseman, Smith, & Self, 1987) was introduced to estimate the location of a robot relative to landmarks and a covariance matrix is used to capture the uncertainty of the estimates. In this approach, the state vector and covariance matrix are updated when new measurements are made and new landmarks are observed. Another filter-based SLAM method is FastSLAM (Montemerlo, Thrun, Koller, Wegbreit, et al., 2002), which integrates particle filters with EKF on SLAM to improve the scalability of the algorithm. In the case of swarms of sUAS, estimates of each agent's state and location relative to landmarks can be improved by filtering shared data (Martinelli, 2011).

Table 6
Selected SLAM algorithms for multi-rotor sUAS. Vision-based SLAM methods are more popular in sUAS implementation mainly due to their low cost and low computational cost. Some LiDAR-based SLAM methods perform well in real time and, with the technological advance of LiDAR sensors, more LiDAR SLAM will soon be applied to sUAS systems.

SLAMAlgorithm	Class	Method	Real-time application on sUAS	Ref.
Cartographer	2D & 3D LiDAR	Direct		Hess, Kohler, Rapp, and Andor (2016)
ElasticFusion	Visual	Direct	✓	Whelan, Salas-Moreno, Glocker, Davison, and Leutenegger (2016)
GMapping	2D LiDAR	Filter		Grisetti, Stachniss and Burgard (2007)
KinectFusion	Visual	Direct		Newcombe et al. (2011)
LeGO-LOAM	3D LiDAR	Direct		Shan and Englot (2018)
LOAM	3D LiDAR	Direct	✓	Zhang and Singh (2014)
LSD SLAM	Visual	Direct	✓	Anandharaman, Sudhakaran, and Seyezhai (2016)
MonoSLAM	Visual	Feature	✓	Davison, Reid, Molton, and Stasse (2007)
ORB-SLAM	Visual	Feature	✓	Campos, Elvira, Rodríguez, M. Montiel, and D. Tardós (2021), Mur-Artal, Montiel, and Tardós (2015), Mur-Artal and Tardos (2017)
PTAM	Visual	Feature	✓	Klein and Murray (2007)

5.2. Simultaneous localization and mapping

Consider a sUAS whose sensors observe unknown landmarks as the vehicle moves through an unexplored environment. The SLAM problem aims to compute the probabilistic distribution $P(x_k, m|z_{0:k}, u_{0:k}, x_0)$, $k \in \{0, \dots, n_t\}$, where x_k denotes the state of the sUAS at time $t_k \in [t_0, t_0 + n_t \Delta T]$, $m \in \{m_1, \dots, m_q\}$ denotes the locations of the q landmarks, $u_{0:k} \triangleq \{u_0, \dots, u_k\}$ denotes the *history of control inputs*, $z_{0:k} \triangleq \{z_0, \dots, z_k\}$ denotes the *history landmark observations*, and x_0 denotes the state at time t_0 (Durrant-Whyte & Bailey, 2006). In particular, $P(x_k, m|z_{0:k}, u_{0:k}, x_0)$ is the *joint posterior density* for the state x_k and map m at time t_k based on the history of observations $z_{0:k}$ and control $u_{0:k}$. Given the *state transition model* denoted by $P(x_k|x_{k-1}, u_k)$, $k \in \{1, \dots, n_t\}$, the *observation model* denoted by $P(z_k|x_k, m)$, and an *estimate for the distribution* denoted by $P(x_{k-1}, m|z_{0:k-1}, u_{0:k-1}, x_0)$ at time $t_{k-1} \in [t_0, t_0 + (n_t - 1)\Delta T]$, SLAM algorithms can be updated in a two-step recursive form including a *prediction step*

$$P(x_k, m|z_{0:k-1}, u_{0:k}, x_0) = \int P(x_k|x_{k-1}, u_k) P(x_{k-1}, m|z_{0:k-1}, u_{0:k-1}, x_0) \mathrm{d}x_{k-1}, \tag{18}$$

and a *measurement update step*

$$P(x_k, m|z_{0:k}, u_{0:k}, x_0) = \frac{P(z_k|x_k, m) P(x_k, m|z_{0:k-1}, u_{0:k}, x_0)}{P(z_k|z_{0:k-1}, u_{0:k})}. \tag{19}$$

The SLAM algorithm comprises two essential functions, namely constructing a map and localizing the sUAS in this map. The map construction function can be formulated as computing the conditional density $P(x_k, m|x_{0:k}, z_{0:k}, u_{0:k})$, $k \in \{0, \dots, n_t\}$, given the state history $x_{0:k}$, and the localization function can be formulated as estimating the sUAS' location with respect to landmarks $P(x_k|z_{0:k}, u_{0:k}, m)$, assuming the landmark locations are known. It appears from (18) and (19) that a mutual relationship between the mapping and localization of the sUAS is established and maintained in SLAM as the vehicle requires mapping to be localized and the formation of a map requires the pose estimate of the vehicle.

The SLAM architecture generally consists of a *front-end*, where the robot movement is estimated, and a *back-end*, where the pose graph is optimized given constraints between poses. The performance of SLAM systems can be further improved by including a closed-loop detection step, wherein the robot evaluates whether its current position has been correctly described in the created map from the sensor information. This additional step mitigates cumulative error of an odometer, reduces pose drift, and helps with the convergence of optimization. A typical SLAM architecture is shown in Fig. 8, and in this section we survey several algorithms for the front-end, the back-end, and the loop closure.

Various SLAM algorithms have been developed based on different sensors, such as laser range sensors, rotary encoders, IMUs, GPS, and cameras. This section surveys current SLAM techniques for sUAS classified as visual, LiDAR, and multi-agent SLAM. A summary of the representative methods of the visual and LiDAR SLAM is presented in Table 6, along with their class, methods and whether they have been applied on sUAS in real time.

5.2.1. Visual SLAM

Vision-based navigation has recently gained popularity for sUAS. The cameras utilized in visual SLAM usually provide rich visual information, are affordable, and are lightweight; hence, are suitable for platforms of limited payload and low cost. Although some visual SLAM techniques are more suitable for cameras operating in the visible spectrum of light and hence, are imprecise in low-visibility environments, visual SLAM is advantageous for multiple reasons. For instance, it mostly exploits passive sensors, that is, sensors such as cameras that do not require to emit any signal. Therefore, passive sensors consume less energy than active sensors. Furthermore, passive sensors such as cameras reduce the sUAS' visibility in potentially contested environments.

In visual SLAM, the position of the robot is tracked by setting some key-points in consecutive frames of the environment taken by the camera. Then, the local map is created using these points and the robot localizes itself in the map by minimizing the difference between the observed points in the current key frame and projected points from the previous key frame. The sensors employed in visual SLAM can be divided into different types, the most typical of which include *monocular cameras*, *stereoscopic cameras*, which combines two monocular cameras, and *RGB-D cameras*, which are depth-sensing cameras working in association with conventional RGB cameras.

There are two main approaches in visual SLAM depending on how the camera motion is estimated. One is the *indirect* or *feature-based approach* (Klein & Murray, 2007; Mur-Artal et al., 2015; Mur-Artal & Tardos, 2017), which is based on key-point matching and builds a sparsely reconstructed map of the environment. The other one is the *direct approach*, which uses all the pixels from a camera frame to resolve the surrounding environment and produces a dense reconstruction map (Newcombe et al., 2011) or semi-dense map (Engel, Schops, & Cremers, 2014). Feature-based approaches are more flexible and efficient with joint optimization of all model parameters, while direct methods usually require more computational power, since they reconstruct all points instead of only key-points. However, direct-based approaches to visual SLAM produce maps that are easier to use and do not suffer from ambiguities.

Some of the popular feature-based SLAM systems include *MonoSLAM* (Davison et al., 2007), *PTAM* (Klein & Murray, 2007), *ORB-SLAM* (Mur-Artal et al., 2015), and *ORB-SLAM2* (Mur-Artal & Tardos, 2017). *MonoSLAM* is one of the first real-time monocular visual SLAM system and provides a real-time probabilistic three-dimensional map updated by the extended Kalman filter method. PTAM (Parallel Tracking And Mapping) is the first visual SLAM system to have separate but parallel tracking and mapping threads. ORB-SLAM (Campos et al., 2021; Mur-Artal et al., 2015; Mur-Artal & Tardos, 2017) integrates

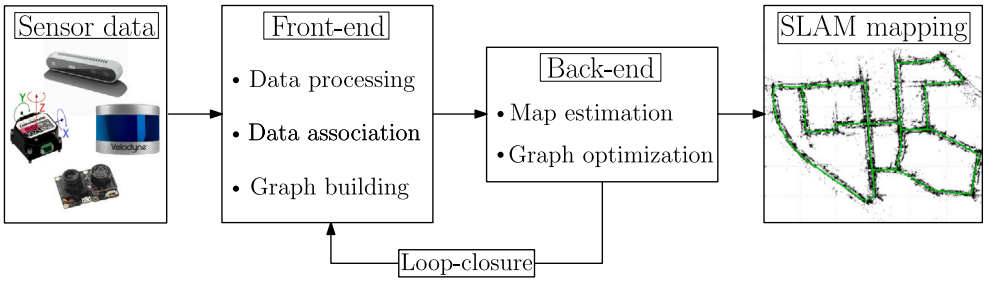

Fig. 8. Architecture of a typical SLAM system. This architecture contains two main components, namely the front-end and the back-end. The front-end estimates the sUAS' movement. In the back-end, the pose graph is optimized given constraints between poses. In some cases, the back-end provides feedback to the front-end for the loop closure process.

PTAM with an added loop closure thread, which takes the last keyframe processed by the local mapping thread to detect and close the loops. The ORB (Oriented FAST and Rotated BRIEF) feature is based on two types of descriptors, namely Features from Accelerated Segment Test (FAST) and Binary Robust Independent Elementary Features (BRIEF). After the ORB features are extracted, they are accumulated to initialize the map by performing bundle adjustments, which refine a visual reconstruction to produce jointly optimal three-dimensional structures and viewing parameter estimates, such as camera pose. On quadcopters, MonoSLAM has been applied in Huang, Tan, and Chen (2015), PTAM in Klein and Murray (2007) and Qiu et al. (2017), ORB-SLAM in Esrafilian and Taghirad (2016) and Lekkala and Mittal (2016) and ORB-SLAM2 in Bokovoy and Yakovlev (2018).

Among the SLAM systems based on the direct method, *Kinect-Fusion* (Newcombe et al., 2011) provides a dense three-dimensional reconstruction of complex room-sized scenes using the depth images obtained from the Kinect depth sensor. Employing this method, the scene model is maintained with a volumetric, truncated signed distance function (TSDF) representation using consecutive depth frames and the associated camera poses. *ElasticFusion* (Whelan et al., 2016) is a real-time dense SLAM technique that captures comprehensive dense globally consistent surfel-based maps of room-scale environments. Direct methods such as *large-scale direct* (LSD) *SLAM* (Engel et al., 2014) produce semi-dense global maps represented as pose graphs, consisting of keyframes as vertices and three-dimensional similarity transforms as edges. Finally, *Direct Sparse Odometry* (DSO) (Engel, Koltun, & Cremers, 2018) combines the benefits of direct methods with the flexibility of feature based approaches. Of these SLAM systems based on the direct approach, to the authors' knowledge, only ElasticFusion (Whelan et al., 2016) and LSD SLAM have been applied to small quadcopters (Anandharaman et al., 2016). However, the results in Anandharaman et al. (2016) have been obtained by shifting some computations on a ground station. Other direct SLAM methods have not yet been applied to sUAS for the high density of the maps produced, but are likely to produce satisfactory results when the computational power of single-board computers for sUAS will allow their implementation in real time.

Vision sensors can be integrated with inertial sensors, such as IMUs, for the development of visual-inertial navigation systems (Santoso, Garratt, & Anavatti, 2016). Many visual-inertial navigation algorithms have been introduced for *visual-inertial odometry* (VIO) (Delmerico & Scaramuzza, 2018) and *visual-inertial SLAM* (Jones & Soatto, 2011). VIO is an odometry method that uses the visual measurements to impose motion constraints on pose estimation. Visual-inertial SLAM extends VIO by including the features in the state and jointly estimating both the position of the environmental features and the poses of the camera and IMU. As a result, visual-inertial SLAM gains increased accuracy over classical VIO. A multi-state constraint Kalman filter is developed for vision-aided inertial navigation in Mourikis and Roumeliotis (2007) and applied to sUAS in Sun et al. (2018). Various geometric features such as points, lines and plane features have been utilized in visual-inertial navigation (Heo, Jung, & Park, 2018; Hsiao, Westman, & Kaess, 2018). A keyframe-based optimization approach has also been utilized for visual-inertial navigation (Leutenegger, Lynen, Bosse, Siegwart, & Furgale, 2015), and a keyframe-based visual-inertial SLAM is applied to sUAS in Fink, Franke, Lynch, Röbenack, and Godbolt (2017).

5.2.2. Lidar SLAM

SLAM techniques based on camera sensors can be severely affected by light and other environmental conditions, which results in limited performance especially outdoors. The LiDAR is a commonly used sensor in SLAM technology that overcomes some of these limitations. A LiDAR utilizes laser beams to determine ranges of surrounding objects and, hence, is unaffected by ambient light variations, performs well in low light conditions, and may produce acceptable results in environments filled with fog or smoke. Furthermore, LiDAR-based systems are characterized by detection ranges, which are longer than those of visual SLAM systems. Lastly, LiDARs are particularly appealing since they produce direct distance measurements and their signals can be easily converted into digital representation of the surrounding environment.

LiDAR can be classified in *two-dimensional* and *three-dimensional LiDARs* according to their ability to scan the environment within a plane or both in azimuth and elevation. A SLAM method called *GMapping* with two-dimensional LiDAR based on Rao-Blackwellisation Partial Filter is presented in Grisetti, Stachniss and Burgard (2007). In Kohlbrecher, Von Stryk, Meyer, and Klingauf (2011), a two-dimensional LiDAR SLAM is combined with an inertial sensing system for three-dimensional navigation. *Cartographer* (Hess et al., 2016) uses a branch-and-bound approach to achieve real-time mapping and loop closure in two-dimensional LiDAR SLAM, which can also be extended to three-dimensional LiDAR SLAM. *Lidar Odometry and Mapping* (LOAM) (Zhang & Singh, 2014) is a real-time state estimation and mapping method using a three-dimensional LiDAR, which has been applied on a sUAS system for power line safety distance detection (Qian, Mai, & Yuwen, 2018). One of its most recent extensions is the *lightweight and ground-optimized LiDAR odometry and mapping* (LeGO-LOAM) method that combines LiDAR with IMU for six degree-of-freedom pose estimation (Shan & Englot, 2018).

A commonly used registration algorithm at the front-end is the *iterative closest point* (ICP) algorithm (Besl & McKay, 1992), which associates two point clouds by minimizing the difference between them through nonlinear optimization methods. A *generalized ICP* (GICP) is presented in Segal, Haehnel, and Thrun (2009), where the original ICP and point-to-plane ICP are combined to model locally planar surface structure for improved performance. A multi-resolution scheme based on GICP is proposed in Droeschel, Stückler, and Behnke (2014) and applied to sUAS equipped with rotating three-dimensional laser scanners.

Widely used techniques of the back-end include the *Maximum A Posteriori* (MAP) estimation (Ramezani, Tinchev, Iuganov, & Fallon, 2020; Thrun & Montemerlo, 2006), where the posterior distribution is utilized for state estimate, and the *maximum likelihood estimation* (Grisetti,

Stachniss, et al., 2007), where the state is estimated by maximizing the likelihood function. Images produced by lasers may be deformed by the robots' relative motion at high speed. This problem can be addressed by modeling the sensor trajectory as a continuous function of time (Kaul, Zlot, & Bosse, 2016). A continuous-time SLAM for three-dimensional LiDAR-based Online Mapping with a hierarchical optimization back-end applied to sUAS is shown in Droeschel and Behnke (2018).

Within the context of SLAM, several solutions have been proposed for loop closure. A closed-loop detection method based on the branch and bound algorithm for two-dimensional LiDAR sensors is presented in Hess et al. (2016). A dense mapping approach called *Surfel-based Mapping* (SuMa) for three-dimensional LiDAR sensors is proposed in Behley and Stachniss (2018), where a virtual view of the map is composed to enable a robust closed-loop detection process. An incremental segmentation algorithm for closed-loop detection is discussed in Dubé et al. (2018). Finally, a LiDAR SLAM closed-loop detection method based on light calibration information from light fidelity (Li-Fi) is proposed in Wang, Zeng, Zou, and Li (2020). A LiDAR-based system with a closed-loop detection step has been applied to sUAS in Konolige et al. (2010). However, the application of closed-loop detection steps to LiDAR-based systems remains still largely under-explored in the field of aerial robotics. These techniques are likely to produce satisfactory results when the computational power of single-board computers for sUAS has considerably improved.

5.2.3. SLAM for swarms of sUAS

The main task of SLAM for swarms of sUAS is to merge the maps built by each robot and leverage the information shared among the agents to achieve improved efficiency, accuracy, and robustness. Although SLAM methods for multi-agent systems have gained considerable attention in recent years, this topic is still considered as new in aerial robotics.

A key problem concerning SLAM for swarms of sUAS is determining the relative orientation among the agents. Indeed, if each agent knows its position in the global reference frame using, for instance, a GNSS module, then extending single-robot SLAM techniques to the multi-agent problem is relatively straightforward Howard (2006). However, if each agent does not have reliable information on its initial position and correspondence, then the SLAM problem is considerably more challenging, and various methods have been proposed.

Among recent advances in SLAM for swarms of sUAS, we recall (Forster, Lynen, Kneip, & Scaramuzza, 2013), where one of the first multi-agent SLAM algorithms was implemented on multiple sUAS equipped with video camera sensors. Due to the low processing power of the onboard computer, a ground station is involved to receive data, create individual maps for each sUAS, and merge the maps when overlaps are detected. A similar approach utilizing a centralized architecture is presented in Schmuck and Chli (2017). A centralized collaborative monocular SLAM method is discussed in Schmuck and Chli (2019b), where the scalability and the robustness of the proposed algorithm to information loss and communication delays is addressed as well. A rendezvous-based map fusion approach is developed in Jang, Oh, Lee, and Kim (2021).

In the absence of a central base station, development of decentralized mapping methods becomes essential due to limitations in data communication of sUAS. A decentralized visual SLAM system is presented in Cieslewski, Choudhary, and Scaramuzza (2018), where decentralized place recognition, optimization, and visual feature association are integrated to reduce the amount of exchanged data. A distributed mapping algorithm is proposed in Choudhary et al. (2017), where object-based models are exchanged among the robots instead of raw sensor measurements to reduce the communication burden. A *Distributed, Online, and Outlier Resilient* (DOOR) *SLAM* for robotic teams has been recently proposed in Lajoie, Ramtoula, Chang, Carlone, and Beltrame (2020), where an outlier rejection mechanism and a inter-robot closed-loop detection module without exchanging raw sensor data are employed.

Some multi-agent SLAM methods combine visual sensors with other types of sensors to achieve more robust performance. A SLAM algorithm applied to a team of sUAS equipped with RGB-D cameras that provide images with both color and depth information is presented in Loianno, Thomas, and Kumar (2015), where RGB images and depth data are combined to generate dense maps. Visual sensors can also be integrated with IMUs in the visual-inertial SLAM framework and applied to multi-agent systems as shown in Houben, Quenzel, Krombach, and Behnke (2016), Karrer, Schmuck, and Chli (2018).

LiDAR-based SLAM has also been applied to swarms of mobile robots. In Dong, Nelson, Indelman, Michael, and Dellaert (2015), a distributed, online, and real-time cooperative SLAM scheme is implemented on a swarm of sUAS, where a two-dimensional laser scan correspondence method is proposed to form robust correspondences between laser scans shared among robots. Image and LiDAR data are combined in Tian et al. (2020) for search and rescue under the forest canopy using multiple sUAS with a collaborative SLAM.

Relative pose estimation among sUAS in a swarm is another important aspect for multi-agent localization and navigation. In vision-based pose estimation systems (Breitenmoser, Kneip, & Siegwart, 2011), markers, such as *AprilTags*, can be attached to the sUAS to aid other agents in detecting and tracking its pose (Hoogervorst, Stramigioli, Wopereis, & Fumagalli, 2015). A vision-based relative localization system with low computational cost based on a circular ring pattern is proposed in (Krajník et al., 2014) and applied to multi-agent sUAS cooperative surveillance in Saska et al. (2016). Another approach is to attach a set of infra-red or visible-light markers, which are detectable under a wide variety of illumination conditions, for relative positioning (Faessler, Mueggler, Schwabe, & Scaramuzza, 2014). Relative pose estimation for swarms of sUAS has also been studied in a leader-follower configuration, where the leader has infrared markers and the follower is equipped with a camera (Wilson, Göktoğan, & Sukkarieh, 2016). This approach has also been extended to the visual inertial fusion case in Teixeira, Maffra, Moos, and Chli (2018).

5.2.4. Stability and robustness of SLAM algorithms

Stability and robustness are essential aspects of navigation systems in order to achieve reliable autonomous operation of sUAS. To increase the stability and accuracy of their SLAM algorithm, the authors of Sujiwo, Ando, Takeuchi, Ninomiya, and Edahiro (2016) introduced a relocalization module, which recomputes the sensor pose with respect to the map when the tracking is lost due to fast sensor motion or other disturbances. Alternatively, the authors of Qin, Li, and Shen (2018a) exploited the global map optimization module, which leverages the pose-graph optimization method presented in Kummerle, Grisetti, Strasdat, Konolige, and Burgard (2011) and suppresses the accumulative estimation errors. A stability analysis of the state estimation error dynamics for a filter-based SLAM is performed in Vidal-Calleja, Andrade-Cetto, and Sanfeliu (2004b) and conditions to guarantee convergence of a linear-time implementation of SLAM for suboptimal filter stability is presented in Vidal-Calleja, Andrade-Cetto, and Sanfeliu (2004a).

Robust schemes in filtering and tracking are utilized in filter-based SLAM and feature-based SLAM to improve the robustness and efficiency of the algorithms (Liu, Zhang, & Bao, 2016; Sun et al., 2018). Sensor fusion, such as the combination of visual and IMU information (Qin, Li, & Shen, 2018b; Wang, Yang, Cao, Xu, & Lin, 2018), the fusion of camera and LiDAR sensor measurement (Shin, Park, & Kim, 2020), or the inclusion of semantic information (Atanasov, Bowman, Daniilidis, & Pappas, 2018), also enhances the robustness, stability, and reliability of the SLAM algorithms.

In the case of SLAM for swarms of sUAS, collaboration among agents improves the robustness and accuracy of the SLAM algorithms (Zou, Tan, & Yu, 2019). Stability of multi-agent SLAM is also critical for

reliable autonomous operations of multi-agent sUAS in dynamic environments. In Li, Wang, Jiang, and Xu (2019), a deep learning-based cloud robot framework is introduced to enhance feature extraction and place matching in order to improve the stability of the multi-agent SLAM and the accuracy of map fusion.

5.3. SONAR- and sound-based navigation systems

LiDAR-based navigation techniques may be unemployable in environments where thick layers of smoke and fog excessively scatter the laser beam. In these cases, a viable solution to the navigation problem for sUAS is provided by SONAR systems. For instance, the authors in Gustavsson (2015) propose a navigation system based on the fusion of data from an IMU, a SONAR, and satellite signals. Similarly, in Müller and Burgard (2013), the authors present a navigation system based on the fusion of data from an IMU, a SONAR, and air flow sensors. In Tardos, Neira, Newman, and Leonard (2002), the authors investigate the use of Polaroid SONAR data for indoor navigation of autonomous mobile robots. Finally, the authors in Calkins et al. (2020) exploit the sound produced by the sUAS' propellers to detect environmental features. As recognized in Tardos et al. (2002), SONAR-based systems are considerably more imprecise than navigation systems based on the use of optical devices. Furthermore, sUAS are unable to transport sufficiently powerful SONAR to obtain resolutions comparable to LiDAR-based systems. Thus, this topic is still largely under-explored within the context of lightweight multi-rotor sUAS. However, considering the high quality of flights performed by aerial mammals, such as bats, and cave-dwelling birds, it is expected that in the future, SONAR-based navigation systems can be further improved to enable precise navigation.

The sound produced by sUAS can also be leveraged to define the relative position of agents in swarms of sUAS (Basiri, 2015). However, in the case of identical or similar sUAS, this approach is highly imprecise. Solution to this problem involves the use of a 'chirp' (Basiri, Schill, Lima, & Floreano, 2016) or convolutional neural networks (Cabrera-Ponce, Martinez-Carranza, & Rascon, 2020).

5.4. Mapping

An essential factor in navigation is how precisely the robot maps its environment and, in general, mapping is a very challenging aspect in autonomous navigation. Some map representations employed on sUAS are described in the following.

5.4.1. Metric maps

Metric mapping is a symbolic structure that encodes the geometry of the environment. According to this approach, the environment is represented in terms of geometric relations between the objects and a fixed reference frame (Yousif, Bab-Hadiashar, & Hoseinnezhad, 2015). *Landmark-based maps* and *occupancy grid maps* are commonly used for metric mapping in two-dimensional cases. Landmark-based maps provide a sparse set of landmarks as the output map, whereas occupancy grid maps represent the map of the environment as a discrete grid where each cell in the grid is assigned with a value representing the probability of the occupancy of that cell. For three-dimensional cases, which are of greater interest for applications related to multi-rotor sUAS, there are numerous forms of mapping, including *feature-based maps*, *dense maps*, and *occupancy grid maps*.

Feature-based maps represent the environment in a form of sparse geometric shapes such as points and lines (Yousif et al., 2015). The features detected in the map consist of their location and geometrical shapes. ORB-SLAM and its variations (Campos et al., 2021; Mur-Artal et al., 2015; Mur-Artal & Tardos, 2017) detect the ORB features and map out the environment using these features in their correct location and shape in the form of points. A point cloud map is formulated through the accumulation of points with the help of the bundle adjustment technique.

In dense mapping systems, information collected from all the pixels from the image are applied. In CNN-SLAM (Tateno, Tombari, Laina, & Navab, 2017), depth measurements are also used to reconstruct the three-dimensional space depending on the selected keyframes and their pose. In ElasticFusion (Whelan et al., 2016), a dense surface element (surfel) map is created by fusing the surfel model with the initialized deformation graph, which mirrors the surfel model.

An occupancy grid map works by creating a discretized map of the environment. In the three-dimensional cases, the volume of the key features are discretized and reconstructed in the map. The OctoMap (Ait-Jellal & Zell, 2017; Hornung et al., 2013) is an efficient probabilistic three-dimensional mapping framework that is based on an Octree, a tree data structure. In this structure, each internal node has eight children, and explicitly represents not only occupied space, but also free and unknown areas; a graphical representation of the OctoMap structure is presented in Fig. 9. OctoMap has been shown to integrate 3D scans into the map in less than a second and it can cope with the demanding data of RGB-D-cameras that output up to 300,000 points at fast frame rates when evaluated on a single core of an Intel Core i7-2600, 3.4 GHz processor. In Ait-Jellal and Zell (2017), this mapping technique has been applied to support an $RRT^{\star}$ algorithm for efficient 3D path planning.

Voxel maps provide a viable alternative to OctoMap. These maps partition the environments in cubes of user-defined dimensions, whose probability of being occupied by obstacles are larger than a user-defined threshold (Pestana, Maurer, Muschick, Hofer, & Fraundorfer, 2019).

5.4.2. Semantic maps

Semantic SLAM (Bowman, Atanasov, Daniilidis, & Pappas, 2017) is an approach to SLAM, which adds semantic information about the objects in the environment. The creation of a semantic map is highly dependent on deep learning techniques (Goodfellow, Bengio, & Courville, 2016). Some of the semantic SLAM systems include CNN-SLAM (Tateno et al., 2017) and ElasticFusion (Whelan et al., 2016) to name those few currently employed on sUAS. In CNN-SLAM, reconstructed maps are generated using the semantic labels obtained from a convolutional neural network and fusing them with the depth map. Finally, ElasticFusion provides a long-term semantic dense correspondence.

5.5. Modeling collision avoidance constraints

An essential connection between guidance and navigation systems for autonomous mobile robots in general and autonomous sUAS in particular is provided by algorithms that produce search spaces for the path and trajectory planners, where to find collision-free solutions. Two approaches are usually pursued to design these algorithms, namely, either identifying subsets of the free space, where to plan reference trajectories for the sUAS, or bounding all known obstacles in some sets, and then planning reference trajectories anywhere on the map of the environment, except for these bounding sets. In the following, both approaches are surveyed and eventually compared.

5.5.1. Enforcing collision avoidance by bounding the sUAS

An approach to produce search spaces for path and trajectory planners consists of finding subsets of the free space that contain the sUAS and exclude any detected obstacle. Within this context, collision avoidance constraints are usually generated in the form of convex sets, whose boundaries are piecewise affine or quadratic; an example of affine constraints is provided by (14).

Few techniques have been applied on sUAS to compute convex representations of free space in cluttered environments. The *iterative regional inflation by semidefinite programming* (IRIS) technique has been used several times in the recent literature for computing large convex regions of obstacle-free space (Deits & Tedrake, 2015a). The key steps of the IRIS algorithm can be summarized as follows. Given an initial

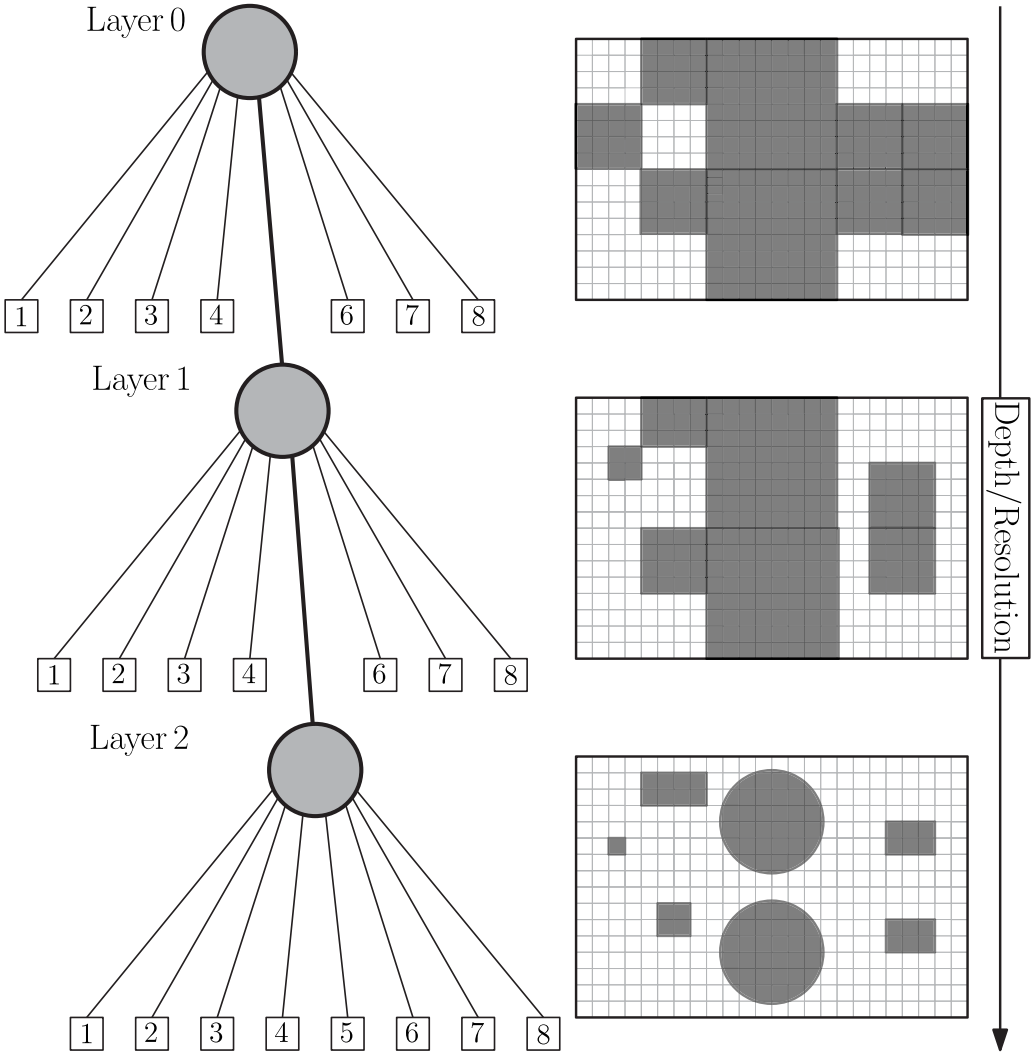

Fig. 9. Illustration of the Octree occupancy mapping underlying the OctoMap architecture. Point clouds from laser-range finders or stereoscopic cameras are organized into an Octree structure. The first layer of the Octree structure, whose eight nodes represent uniform or non-uniform subdivisions of an environment, encompasses all information about an environment, and provides the lowest resolution representation of occupancy. As the Octree's depth increases, the resolution increases, allowing the inquirer to see more detail. User-defined segmentation and hierarchy rules allow to create sub-maps, where certain features in an environment are expressed at higher or lower resolutions that others. The Octree occupancy mapping, user-defined segmentation, and hierarchy rules are encompassed in the OctoMap algorithm (Hornung, Wurm, Bennewitz, Stachniss, & Burgard, 2013).

seed point, IRIS solves a quadratic program to find an ellipsoid that separates the seed point from an obstacle. Then, IRIS creates a hyperplane that intersects the closest point to the seed point on that obstacle and separates the seed point. By repeating this procedure for each obstacle, IRIS produces multiple hyperplanes, each of which identifies a half-space. The intersection of these half-spaces creates a polytope. Thus, IRIS finds the maximum volume ellipsoid inscribed in this polytope by solving a semidefinite program, and hence, the ellipsoid computed previously is inflated; this ends the first step of the IRIS algorithm. As part of the next step, IRIS finds the point on each obstacle that is closest to the maximum volume inscribed ellipsoid. Then, as in the first step, IRIS computes a set of separating hyperplanes, the resulting polytope, and the maximum volume ellipsoid inscribed in this polytope. This second step is performed iteratively until the volume of the ellipsoid inscribed in the polytope at two consecutive iterations changes less than some user-defined threshold; Fig. 10 provides a schematic representation of the IRIS algorithm.

IRIS assumes that all obstacles are convex or encapsulated in some convex set. Furthermore, the choice of the seed point may affect the outcome of the IRIS algorithm in a significant manner. Recently, the authors of Deits and Tedrake (2015b) proposed to select the seed point in the IRIS algorithm as the point that is furthest from any obstacle point. This method has computational complexity that is linear in the number of obstacles, and is shown to compute convex sets of free-space quickly despite millions of obstacles. The authors of Alonso-Mora, Baker, and Rus (2017) presented a modified version of the IRIS algorithm. Two key features distinguish this algorithm from the original formulation of IRIS. Firstly, the initial separating ellipsoid is required to be a minimal ellipsoid containing both a point of interest, such as the goal point from a planned path, and a set of points $I_{\text{sUAS}} \subset \mathbb{R}^3$, which capture the edge of the sUAS. Secondly, this modified algorithm requires that if the set of points I_{sUAS} is not contained in the polytope after the ellipsoid inflation step, then the algorithm is terminated. This ensures that the trajectory planner finds a trajectory that proceeds towards the goal and separates the sUAS from the obstacles.

As an alternative to IRIS, the *safe flight corridor* (SFC) algorithm was developed for fast autonomous flight in unknown and cluttered environments (Liu et al., 2017). Given a planned path with straight path segments, the SFC algorithm first places a sphere centered at the midpoint of a path segment and containing a path segment. A bounding box is placed around the center point to save computational effort by neglecting the influence of distant obstacles. The SFC algorithm deforms the sphere until an ellipsoid is found that intersects the closest point on an obstacle, and contains the path segment. Then a hyperplane is

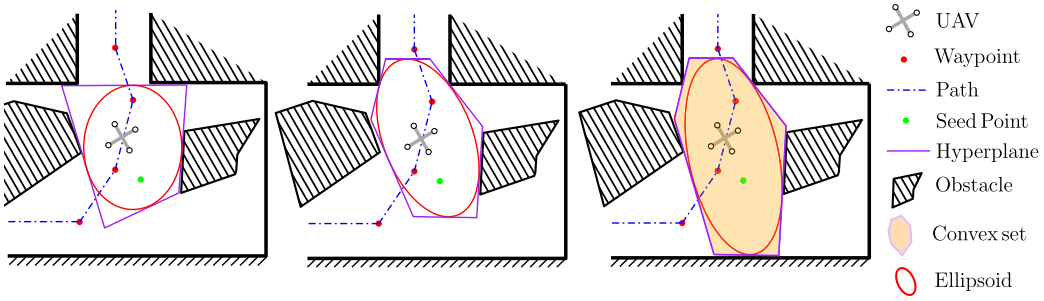

Fig. 10. Schematic representation of the IRIS algorithm (Deits & Tedrake, 2015a). Left: Given a seed point, IRIS computes an ellipsoid that separates the seed point from the obstacles, computes the hyperplanes tangent this ellipsoid and intersecting the nearby obstacles, and computes the polytope bounded by these hyperplanes. Middle: IRIS iteratively computes the maximum volume ellipsoid inscribed in the polytope produce at the previous step, and computes the hyperplanes tangent to this ellipsoid and intersecting the obstacles. Right: When the ellipsoid ceases to grow, the final polytope is the convex set of free space surrounding the sUAS.

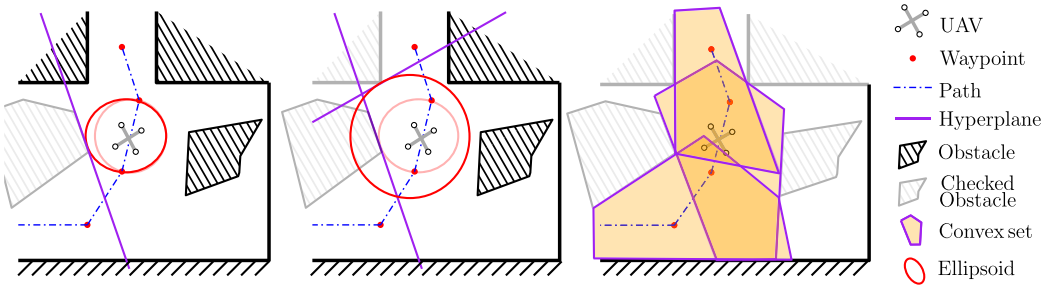

Fig. 11. Schematic representation of the SFC algorithm (Liu et al., 2017). Left: SFC computes an ellipsoid (pale red inner curve) that contains the sUAS and intersects adjacent waypoints, adjusts the size and shape of this ellipsoid (red outer curve) until it intersects the nearest obstacle point, and finally computes a hyperplane that is tangent to the ellipsoid and passes through the nearest obstacle point. Middle: SFC inflates the ellipsoid produced at the previous step until it intersects a different obstacle. Upon finding a new ellipsoid that intersects the current obstacle, the obstacle is designated as *checked* (light gray shaded polygon) and the next obstacle is queried. Right: The algorithm is terminated once every nearby obstacle has been checked. The space bounded by all hyperplanes is the convex set of free space surrounding the sUAS. This algorithm is applied to every pair of consecutive waypoints in the planned path, resulting in multiple, possibly overlapping, polyhedra.

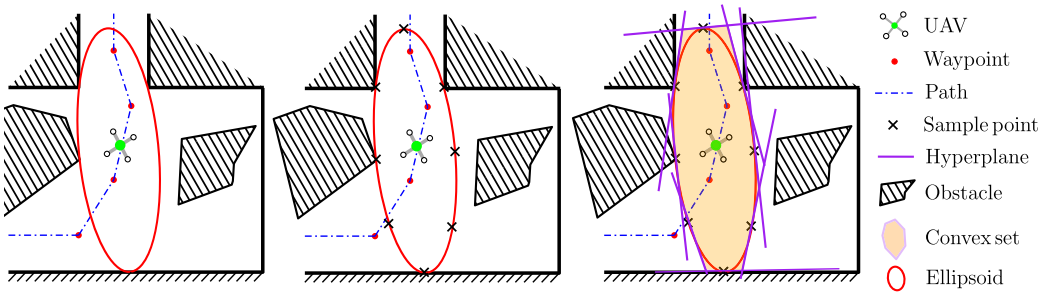

Fig. 12. Schematic representation of the algorithm presented in Marshall, Anderson, Chien, Johnson, and L'Afflitto (2021). Left: An ellipsoid containing the sUAS and tangent to the obstacle's set is computed. Middle: This ellipsoid is sampled at $N > 0$ points. Right: N hyperplanes, which are tangent to the ellipsoid and pass through the N sample points, are computed. The space bounded by all hyperplanes is the convex set of free-space surrounding the sUAS.

computed that is tangent to the ellipsoid and intersects the closest point on a particular obstacle. This ellipsoid continues to be deformed until it intersects the closest point on the next nearest obstacle and a new hyperplane is formed, and that obstacle is designated as *checked*. The process is terminated once the list of obstacles to check is exhausted. Fig. 11 provides a schematic representation of the SFC algorithm. This method is advantageous when a planned path is available and is unlikely to change as the sUAS traverses the environment. However, if the planned path is likely to change, then safe flight corridors found in the most recent executions of the SFC algorithm will no longer be safe and computational effort is wasted finding these regions. The authors of Mohta et al. (2018) introduce a fully featured guidance, navigation, and control system for fast autonomous flight in GPS-denied and cluttered environments and employ SFC to construct convex representations of free space.

In Marshall et al. (2021) a new guidance, navigation, and control system for tactical sUAS was presented, and a new algorithm was developed to quickly identify convex sets containing the sUAS and free of obstacles detected by the onboard cameras. This algorithm first solves a semidefinite programming problem to compute an ellipsoid that contains the sUAS and is tangent to the obstacle's set in at least one point. Successively, the ellipsoid's surface is sampled by N points. Lastly, N hyperplanes, which are tangent to the ellipsoid at the sample points, are generated. Numerical simulations show that this algorithm outperforms both IRIS and SFC in computational speed. Fig. 12 provides a schematic representation of this algorithm.

The problem of finding collision avoidance constraint sets clearly transcends aerial robotics. Algorithms to quickly find convex representations of free-space in an environment, are given in Liu, Jayakumar, Stein, and Ersal (2018) and Sarmientoy, Murrieta-Cidz, and Hutchinsony (2005). However, none of these algorithms have been applied to sUAS yet.

5.5.2. Enforcing collision avoidance by bounding the obstacles

An approach complementary to those presented in Section 5.5.1 to produce search spaces for path and trajectory planners consists of bounding all known obstacles in some sets, and then planning reference trajectories anywhere on the map of the environment, except for these bounding sets. Within this context, one of the most intuitive approaches involves repulsive or negative artificial potential fields (Koditschek, 1989; Wen, Zhao, Su, & Ma, 2015; Yan, Chenxiao, Juan, Yun, et al. Yi, 2020).

An approach similar to artificial potential fields consists in using *Euclidean-signed distance fields* to capture obstacles and hence, enable real-time collision avoidance on sUAS (Han et al., 2019; Oleynikova, Taylor, Siegwart, & Nieto, 2018; Zhou, Boyu et al., 2021). A Euclidean-signed distance field is a voxel map, where every free voxel is given a value representing the Euclidean distance to the nearest occupied voxel, and every occupied voxel is assigned a value representing the Euclidean distance to the nearest free voxel. An advantage of Euclidean-signed distance fields is that if the region being mapped has a smooth boundary, then the Euclidean-signed distance field is differentiable almost everywhere, and therefore, gradient information is obtained quickly to generate repulsive artificial potential fields.

The authors in Jiang and Liang (2018) address the collision avoidance problem by encapsulating obstacles, opponents, and other aircraft in ellipsoids. Then, reference trajectories are outlined by means of an optimal control framework, and collision avoidance is ensured by constraining the sUAS to lie outside of all ellipsoids. This is an intuitive approach to modeling obstacles in the environment. However, this method may imply an overly coarse approximation of the obstacles.

The problem of finding convex representations of free space for collision avoidance can be avoided in favor of embedding collision costs into the objective function of optimization-based path planners, trajectory planners, or both by means of adequate penalty functions. This approach is known as embedding collision avoidance as *soft constraints*, that is, constraints that should not be, but could be violated. The use of soft constraints, however, may undermine the ability to meet collision avoidance constraints. A remarkable example of penalty functions employed to embed constraints in the cost function is the Kreisselmeier–Steinhauser function (Kreisselmeier & Steinhauser, 1979): by proceeding as in Wang and Boyd (2009), quadratic optimal control problems subject to linear equality constraints can be reduced to quadratic programming problems characterized by block-tridiagonal matrices. Thus, the Kreisselmeier–Steinhauser function can be employed to embed inequality constraints in the cost function without altering the advantageous structure of the underlying matrices (Richards, 2015).

Recently, motion primitives (Spitzer et al., 2018) and optical flow-based methods (Keshavan, Gremillion, Alvarez-Escobar, & Humbert, 2015) have been employed in guidance systems of autonomous sUAS for collision avoidance. The method of motion primitives, which has been discussed in Section 4.1.6, can quickly provide dynamically feasible collision-free trajectories. However, the space of control actions must be discretized to avoid excessive computation, potentially excluding *a priori* more convenient maneuvers. Optical flow-based methods offer quick, reflexive navigation in unknown environments with low-cost sensors. However, to avoid collisions with obstacles in unknown areas, the sUAS should proceed in the direction of the onboard sensor, unless combined with other sensors and collision avoidance techniques.

5.5.3. Bounding the sUAS vs. Bounding the obstacles

The methods presented in Sections 5.5.1 and 5.5.2 have their peculiar advantages and disadvantages. For instance, bounding the sUAS within a convex set is useful to support optimization-based guidance systems. Indeed, a sufficient condition for the existence and uniqueness of a solution of optimization and optimal control problems, such as those underlying the optimal control-based trajectory planning algorithms surveyed in Section 4.2.1, is the convexity of the cost function over a convex set. Furthermore, representing constraint sets by means of affine or quadratic functions usually guarantee low computational costs. On the other hand, convex collision avoidance constraint sets must meet multiple requirements such as containing the sUAS at the time a result is produced and excluding all known obstacle points. Additionally, although collision avoidance constraint sets should also include a subset of the sUAS' goal set or, at least, the next waypoint produced by the path planner, this requirement may not always be met due to the complexity of the environment in sUAS may operate. To partly compensate for this problem, the collision avoidance constraint set should be as large as possible. However, the problem of deducing an analytical expression for the largest convex set, which excludes all obstacle points and whose boundaries are piecewise affine of quadratic, is particularly challenging, especially when obstacles are captured in discrete form by point clouds or voxel maps. For this reason, relatively few solutions to the problem of generating sets that contain the robot and exclude the obstacles have been found thus far.

Collision avoidance methods based on negative artificial potential fields, or variations thereof, are advantageous because the repulsive forces associated to these fields can be quickly constructed and updated online as new obstacles are discovered. On the other hand, these methods introduce an additional buffer around the obstacles' set, which may prevent the sUAS from traversing narrow passages.

The authors in Deits and Tedrake (2015b) compared the performance of the same trajectory planner integrated with two different approaches to collision avoidance. The first approach consists of a direct application of IRIS (Deits & Tedrake, 2015a), which was discussed in Section 5.5.1. The second approach consists of constraining the obstacles within some sets and searching reference trajectory anywhere in the sUAS' map, except for these sets. The result of this comparison, which involved four sets of simulations performed on a 2.7 GHz 12-Core Intel Xeon E5 processor, is that, statistically, IRIS is faster than the other technique. Furthermore, in average, the norm of the jerk of the reference trajectory computed using IRIS is smaller than the norm of the jerk of the reference trajectory computed using the alternative approach and, hence, employing IRIS, the sUAS' acceleration varies less. On the other hand, enforcing collision avoidance by excluding sets that contain obstacles guarantees a larger search space and, hence, produces feasible trajectories at a higher success rate than IRIS. This comes at the expense of a higher computational time and higher value of the norm of the jerk of the reference trajectory as the number obstacles grows. Indeed, as the number of obstacles increases, IRIS is less likely to find a reference trajectory.

6. Control systems for autonomous multi-rotor sUAS

The control problem can be described as the determination of the thrust force and the moment of the thrust force necessary for the sUAS to follow a given reference trajectory and meet user-defined constraints. This section presents common architectures of control systems for multi-rotor sUAS, and it surveys control systems for multi-rotor sUAS based on linear control techniques, feedback-linearizing control techniques, nonlinear control laws, and learning-based methods; some of these control techniques and some of their key features are summarized in Table 7. In this paper, those framework that allow to compute the control input as *explicit* functions of the sUAS' state vector or measured output at the time they are computed are considered as control techniques. To be effective in problems of practical interest, control laws must guarantee satisfactory results despite the fact that the dynamical model employed for their design may not capture the sUAS actual dynamics due to unforseen external disturbances, faults, failures, and modeling uncertainties.

Presently, all COTS autopilots, such as those listed in Table 1, employ some linear control technique to regulate quadcopters. The authors have also recently experimented with success an implementation of a classical model reference adaptive control (MRAC) law on a PixHawk autopilot. However, in general, nonlinear techniques are executed on single-board computers such as those listed in Table 3.

Table 7
Selected linear and nonlinear robust control techniques applied to multi-rotor sUAS. Linear control techniques are suitable whenever sUAS operate in relatively small neighborhoods of the equilibrium condition (hover), that is, when it ascends or descends at 0.2 m/s or less and pitches or rolls of 14 degrees or less. Nonlinear control techniques are recommended whenever the vehicle's dynamics must be exploited to perform complex maneuvers. Some control techniques, such as variable structure control laws, are inherently robust to modeling uncertainties and external disturbances. Other techniques, such as PID control, guarantee considerably smaller robustness margins.

Technique [Linear vs. Nonlinear]	Robustness	Ref.
PID [L]	To parametric uncertainties given by integral term. Not to external disturbances	L'Afflitto and Mohammadi (2018)
$\mathcal{H}_2$ control (e.g. LQG) [L]	To uncertainties in $\tilde{B}$ but not in $\tilde{A}$ in (8)	Martins, Cardeira, and Oliveira (2019)
$\mathcal{H}_\infty$ control [L]	To external disturbances	Noormohammadi-Asl, Esrafilian, Ahangar Arzati, and Taghirad (2020)
State- or output- feedback linearization [N]	None in original formulation. Requires robust formulation	Franco, Bourlès, and De Pieri (2007), Tal and Karaman (2020)
MPC [Both L and N]	Due to iterative application at a high frequency	Castillo-Lopez et al. (2018), Kamel et al. (2017), Koo et al. (2015), Pereira et al. (2021), Prodan et al. (2013)
Backstepping [N]	None in original formulation. Requires robust formulation	Loubar, Zammoum Boushaki, Aribi, and Abdellah (2019), Zhang, Gu, Deng, and Wen (2019)
Geometric control [N]	None in original formulation. Requires robust formulation	Garcia, Rojo-Rodriguez, Sanchez, Saucedo, and Munoz-Vazquez (2020), Lee (2018)
MRAC [N]	To parametric uncertainties in original formulation. Some variations (e.g. e- or σ-modification) are robust to external disturbances	Anderson, Marshall, and L'Afflitto (2021a), Anderson, Marshall, L'Afflitto, and Dotterweich (2020), Bakori and Moncayo (2021), Bakshi and Ramachandran (2016), Dydek, Annaswamy, and Lavretsky (2013), Ioannou and Fidan (2006), Niit and Smit (2017), Peterson and Narendra (1982), Xie, Yu, Cabecinhas, Cunha, and Silvestre (2021)
$\mathcal{L}_1$ adaptive control [N]	To uncertainties within the bandwidth of the control channel	Kotaru, Edmonson, and Sreenath (2019)
Variable structure (e.g. sliding mode) [N]	To parametric uncertainties & external disturbances	Jayakrishnan (2016), Le Nhu Ngoc Thanh and Hong (2018), Nguyen et al. (2021), Xiao (2020), Xiong and Zhang (2017)
Learning-based [Both L and N]	Due to periodic learning of system's dynamics	Bansal, Akametalu, Jiang, Laine, and Tomlin (2016), Becker-Ehmck, Karl, Peters, and van der Smagt (2020), Dierks and Jagannathan (2009)

6.1. Underactuation and control systems architectures for quadcopters

As discussed in L'Afflitto et al. (2018), it follows from (5) that quadcopters are underactuated since their propellers' axes are parallel to the vehicle's vertical axis, that is, the $z_{\text{body}}(\cdot)$ axis of the vehicle's reference frame $\mathbb{J}(\cdot)$. Therefore, the rotational dynamics of sUAS and their translational dynamics along the local vertical axis can be steered at will concurrently, while the translational dynamics in the vehicle's horizontal plane, that is, the plane spanned by $\{x_{\text{body}}(\cdot), y_{\text{body}}(\cdot)\}$ cannot be controlled directly. As discussed in the following this limitation can be overcome in multiple ways.

If the guidance system does not account for the aircraft's rotational dynamics, then control systems are designed by employing an outer loop and an inner loop. The *outer loop* is a dynamical system, whose state vector is given by the first two components of $r_A^{\mathbb{I}}(\cdot)$, that is, the sUAS' position in the inertial horizontal plane, and their first derivatives. The control inputs for the outer loop are given by the roll angle $\phi(\cdot)$ and the pitch angle $\theta(\cdot)$. Thus, applying a linearized dynamical model, the outer loop for a quadcopter can be captured by

$$\begin{bmatrix}\ddot{r}_{A,X}(t)\\ \ddot{r}_{A,Y}(t)\end{bmatrix} = \frac{u_1(t)}{m}\begin{bmatrix}\sin\psi(t) & -\cos\psi(t)\\ \cos\psi(t) & \sin\psi(t)\end{bmatrix}\begin{bmatrix}\phi(t)\\ \theta(t)\end{bmatrix},$$
$$\begin{bmatrix}r_{A,X}(t_0)\\ r_{A,Y}(t_0)\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix} r_{A,0}^{\mathbb{I}}, \quad \begin{bmatrix}\dot{r}_{A,X}(t_0)\\ \dot{r}_{A,Y}(t_0)\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix} v_{A,0}^{\mathbb{I}},$$
$$t \geq t_0, \tag{20}$$

where $r_{A,X}, r_{A,Y} : [t_0, \infty) \to \mathbb{R}$ denote the first two component of $r_A^{\mathbb{I}}(\cdot)$. Designing the outer loop reduces to finding control laws for $\phi(\cdot)$ and $\theta(\cdot)$ such that the aircraft follows the reference trajectory in the horizontal plane outlined by the onboard guidance system. The control laws for $\phi(\cdot)$ and $\theta(\cdot)$ are commonly referred to as the reference roll angle $\phi_{\text{ref}} : [t_0, \infty) \to [0, 2\pi)$ and reference pitch angle $\theta_{\text{ref}} : [t_0, \infty) \to \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, respectively. Alternative approaches to the outer-loop design process require to find $\phi(\cdot)$ and $\theta(\cdot)$ such that the position in vehicle's horizontal plane is regulated (Adıgüzel & Mumcu, 2019; Hamrah & Sanyal, 2020; Xiong & Zhang, 2016).

The *inner loop* is a dynamical system, whose state vector is given by the vehicle's vertical position in the inertial horizontal plane, that is, the third component of $r_A^{\mathbb{I}}(\cdot)$, the vehicle's vertical velocity, the angles $\phi(\cdot)$, $\theta(\cdot)$, and $\psi(\cdot)$, and the angular velocity. The control input for the inner loop is given by the propellers' thrust force and the moment of the thrust force. Thus, applying a linearized dynamical model, the outer loop for a quadcopter can be captured by

$$\dot{\tilde{x}}_{\text{OL}}(t) = \tilde{A}_{\text{OL}}\tilde{x}_{\text{OL}}(t) + \tilde{B}_{\text{OL}}\tilde{u}(t), \qquad \tilde{x}_{\text{OL}}(t_0) = \tilde{x}_{0,\text{OL}}, \qquad t \geq t_0, \tag{21}$$

where $\tilde{x}_{\text{OL}}(t) \triangleq C_{\text{OL}}\tilde{x}(t)$, $C_{\text{OL}} \triangleq \begin{bmatrix}0_{4\times2} & \mathbf{1}_4 & 0_{4\times2} & 0_{4\times4}\\ 0_{4\times2} & 0_{4\times4} & 0_{4\times2} & \mathbf{1}_4\end{bmatrix} \in \mathbb{R}^{8\times12}$. $\tilde{A}_{\text{OL}} \triangleq \begin{bmatrix}0_{4\times4} & \mathbf{1}_4\\ 0_{4\times4} & 0_{4\times4}\end{bmatrix} \in \mathbb{R}^{8\times8}$ and $\tilde{B}_{\text{OL}} \triangleq \begin{bmatrix}0_{4\times4}\\ \text{diag}(m^{-1}, I_x^{-1}, I_y^{-1}, I_z^{-1})\end{bmatrix} \in \mathbb{R}^{8\times4}$. Designing the inner loop reduces to finding $u(\cdot)$ such that $\phi(\cdot)$ tracks $\phi_{\text{ref}}(\cdot)$, $\theta(\cdot)$ tracks $\theta_{\text{ref}}(\cdot)$, the third components of $r_A^{\mathbb{I}}(\cdot)$ tracks the reference altitude outlined by the guidance system, and $\psi(\cdot)$ tracks the reference yaw angle $\psi_{\text{ref}}(\cdot)$. Alternative approaches to the inner-loop design process require to regulate the vehicle's position along the $z_{\text{body}}(\cdot)$ axis as well as its attitude. If the guidance system does not account for the sUAS' rotational dynamics, then the reference yaw angle can be computed, for instance, by imposing that the vehicle's roll axis is directed towards the next waypoint. Fig. 13 provides a graphical representation of a control system architecture comprising an inner loop and an outer loop (Ames, Notomista, Wardi, & Egerstedt, 2021; Anderson, Marshall, & L'Afflitto, 2021b; Arabi, Gruenwald, Yucelen, & Nguyen, 2017; Gilbert & Ong, 2011; Homer, Mahmood, & Mhaskar, 2020; Kalabić, Kolmanovsky, & Gilbert, 2014).

If the guidance system accounts for the vehicle's translational and rotational dynamics, and it produces a reference vector of independent generalized coordinates $q_{\text{ref}}(t)$, $t \geq t_0$, as it occurs, for instance, in the case motion primitive libraries are employed (see Section 4.1.6), then only the inner loop is needed, since the reference pitch and roll angles are provided by the guidance system. If the guidance system produces both a reference vector of independent generalized coordinates $q_{\text{ref}}(t)$, $t \geq t_0$, and a control input $u(t)$, as it occurs in the case the model

Sidebar 6.1. A substantial gap between the theoretical formulation of numerous control schemes, especially within the nonlinear system framework, and their implementation lays in the fact that control laws for sUAS are usually designed in the continuous-time domain, whereas their implementation occurs in the discrete-time domain. However, control laws that guarantee asymptotic stability of the closed-loop system in the continuous time domain may not be effective in the discrete-time domain, especially at low frequencies. The problem of designing nonlinear, discrete-time control law for sUAS has been addressed by relatively few researchers; some notable exceptions are provided by Adig'uzel and Mumcu (2019), Hamrah and Sanyal (2020) and Xiong and Zhang (2016).

Sidebar 6.2. Inner and outer loops can be considered as two interconnected dynamical systems. Indeed, the inner loop dynamics affects the outer loop dynamics, and hence, control laws for these two systems can not always be designed separately since the inner loop may lead the state vector of the outer loop outside its basin of attraction, that is, outside that region wherein asymptotic convergence of the state vector to the desired goal is guaranteed. In particular, if linearized equations of motion are employed to capture the UAV's dynamics and linear control laws are applied to regulate the dynamical models captured by inner and outer loops, then the separation between outer and inner loops does not provide any limitation since linear controllers are globally asymptotically stable.

If the controllers employed to design the inner or outer loops do not guarantee global asymptotic stability of the inner and outer loops, then these controllers must be designed concurrently and their effectiveness should be proven, for instance, by means of a single Lyapunov- like argument. Alternatively, the basin of attraction of the controlled inner and outer loops should be characterized, and each loop's state vector must be constrained not to depart from its basin of attraction. However, characterizing the domain of attraction of a nonlinear system is a rather complex task, which is usually addressed numerically and in a conservative manner (Homer et al. 2020). Few nonlinear techniques are able to constrain the closed-loop state vector within user-defined constraint sets, while bounding the control input as well. Within the MRAC framework, this result has been recently achieved in Anderson et al. (2021). Usually, constraints on both the state vector and the control input are imposed indirectly by assuming that the control input is produced as the result of a dynamical process (Arabi et al. 2017, Ames et al. 2021), or using add-on structures, such as command governors (Gilbert & Ong 2021, Kalabić et al. 2014).

predictive architecture is employed (see Section 4.2.1), then this control input $u(\cdot)$ may serve as a baseline controller. Baseline controllers generally provide satisfactory results whenever the sUAS' dynamics are closely approximated by the models underlying the guidance system or whenever the control input and the reference trajectory are recomputed at a high frequency. To account for unmodeled effects, such as external disturbances, faults, and failures, closed-loop control laws, such as those surveyed in the following, can be implemented to augment the baseline controller and meet user-defined levels of performance.

6.2. Linear control methods

A large portion of existing autopilots for commercially available sUAS relies on classical proportional–integral–derivative (PID) controllers to control a quadcopter's inner loop and, if needed, its outer loop. Indeed, although the dynamics of multi-rotor vehicles are nonlinear (L'Afflitto & Mohammadi, 2018), the linearized equations of motion (8) suffice to capture these vehicles' motion in conventional applications, such as surveillance and delivery of known payloads, which require the vehicle to operate in neighborhoods of the hover condition. The PID architecture can be interpreted as a proportional–derivative baseline controller augmented by an adaptive control law, that is, the integral term, and, hence, it guarantees robustness to parametric uncertainties in the linearized equations of motion. However, a considerable drawback of PID-based control laws for sUAS is that PID controllers are unable to guarantee satisfactory performance in scenarios involving motor failures, payload dropping, sling and heavy payloads, and faults and failures of the IMU (L'Afflitto et al., 2018).

In some cases, linear control laws for multi-rotor sUAS have been produced by applying classical techniques, such as $\mathcal{H}_2$ control techniques (e.g. the linear–quadratic regulator (Martins et al., 2019)) and $\mathcal{H}_\infty$ control (Noormohammadi-Asl et al., 2020). Applied to the linear dynamical model (8), the linear–quadratic regulator guarantees robustness to uncertainties in the $\tilde{B}$ matrix (Zhang & Fu, 1996). The $\mathcal{H}_\infty$ control framework guarantees large robustness margins for dynamical

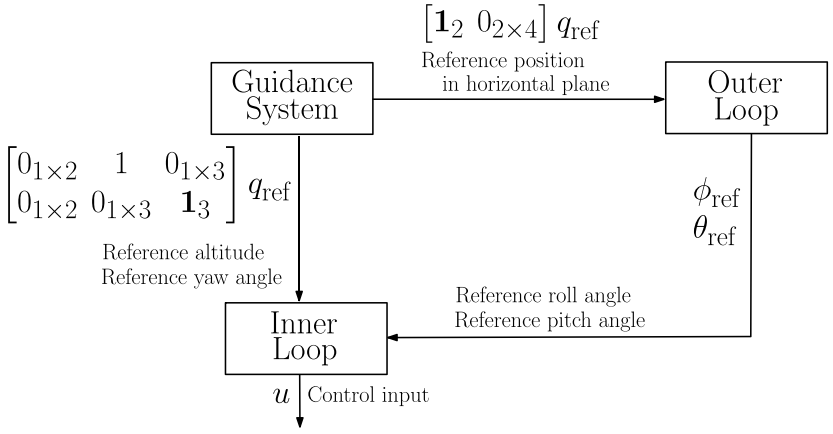

Fig. 13. Graphical representation of a control system architecture for quadcopter sUAS employing both an inner loop and an outer loop. The outer loop produces the reference pitch and roll angles needed to track the reference trajectory in the horizontal plane. The inner loop produces the control input, that is, the total thrust force and the moment of the thrust force, needed to track the reference altitude and the reference yaw angle, which are determined by the guidance system, and the reference pitch and roll angles.

systems of the form

$$\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t) + \tilde{B}\tilde{u}(t) + \xi(t), \qquad \tilde{x}(t_0) = \tilde{x}_0, \qquad t \geq t_0, \tag{22}$$

where $\xi : [t_0, \infty) \to \mathbb{R}^n$ captures unknown external disturbances or unmodeled higher-order terms in the sUAS' dynamics, and is such that $\|\xi(t)\| \leq \bar{\xi}$, where $\bar{\xi} > 0$ is known. However, both the linear–quadratic regulator and $\mathcal{H}_\infty$ control are reliable only in sufficiently small neighborhoods of the hover condition.

6.3. Feedback-linearizing control methods

Recently, the nonlinear dynamics of multi-rotor sUAS has been exploited by employing a differential flatness property of multi-rotor sUAS, that is, the existence of a nonlinear feedback control law that produces a linear map between a virtual control input and the output signal given by the vehicle's position and yaw angle (Tal & Karaman, 2020). In its original formulation, output-feedback linearization can be presented as follows. Assume that the quadcopter's dynamics is captured by (5), assume that the aircraft's total thrust is such that

$$\ddot{u}_1(t) = \eta(t), \qquad \begin{bmatrix}u_1(t_0)\\ \dot{u}_1(t_0)\end{bmatrix} = \begin{bmatrix}u_{1,0}\\ 0\end{bmatrix}, \qquad t \geq t_0, \tag{23}$$

and consider $\mu(t) \triangleq [\eta(t), u_2(t), u_3(t), u_4(t)]^{\mathrm{T}}$ as the control input for the dynamical system given by (5) and (23). Setting $[r_A^{\mathrm{T}}(t), \psi(t)]^{\mathrm{T}}$, $t \in [t_0, \infty)$, as the *measured output*, and applying Proposition 5.1.2 of Isidori (1995), we verify that

$$\begin{bmatrix}r^{(4)}(t)\\ \ddot{\psi}(t)\end{bmatrix} = f_{\text{fl}}(q(t), u_1(t)) + G_{\text{fl}}(q(t), u_1(t))\mu(t),$$
$$\begin{bmatrix}r_A(t_0)\\ v_A(t_0)\\ \ddot{r}_A(t_0)\\ r_A^{(3)}(t_0)\end{bmatrix} = \begin{bmatrix}r_{A,0}\\ v_{A,0}\\ 0_6\end{bmatrix}, \qquad \begin{bmatrix}\psi(t_0)\\ \dot{\psi}(t_0)\end{bmatrix} = \begin{bmatrix}\psi_0\\ 0\end{bmatrix}, \qquad t \geq t_0, \tag{24}$$

where $r^{(n)}(t)$ denotes the nth derivative of $r(\cdot)$, $n \in \{1, \ldots, 4\}$,

$$G_{\text{fl}}(q, u_1) \triangleq m\begin{bmatrix}R_{\text{fl}}(q) & 0_{3\times1}\\ \left[-\frac{I_z c\psi c\theta s\phi}{u_1 c\phi}, -\frac{I_z c\theta s\psi s\phi}{u_1 c\phi}, \frac{I_z s\theta s\phi}{u_1 c\phi}\right] & \frac{I_z c\theta}{mc\phi}\end{bmatrix}, \quad (q, u_1) \in \mathbb{R}^6 \times \mathbb{R}, \tag{25}$$

$$R_{\text{fl}}(q) \triangleq \begin{bmatrix}s\phi s\psi + c\phi c\psi s\theta & c\phi s\psi s\theta - c\psi s\phi & c\phi c\theta\\ \frac{I_x(c\phi s\psi - c\psi s\phi s\theta)}{u_1} & -\frac{I_x(c\phi c\psi + s\psi s\phi s\theta)}{u_1} & -\frac{I_x c\theta s\phi}{u_1}\\ \frac{I_y c\psi c\theta}{u_1} & \frac{I_y c\theta s\psi}{u_1} & -\frac{I_y s\theta}{u_1}\end{bmatrix}, \tag{26}$$

$c\alpha = \cos\alpha$, $\alpha \in \mathbb{R}$, $s\alpha = \sin\alpha$, $f_{\text{fl}} : \mathbb{R}^3 \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [0, 2\pi) \times \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^4$; an expression for $f_{\text{fl}}(\cdot, \cdot)$ is omitted for brevity. Thus, let

$$\zeta(q, u_1, \lambda) \triangleq G_{\text{fl}}^{-1}(q, u_1)\Bigg(-f_{\text{fl}}(q, u_1) + \begin{bmatrix}A_{r,0}r_A(t) + A_{r,1}\dot{r}_A(t) + A_{r,2}\ddot{r}_A(t) + A_{r,3}r_A^{(3)}(t)\\ A_{\psi,0}\psi(t) + A_{\psi,1}\dot{\psi}(t)\end{bmatrix} + \begin{bmatrix}B_r\\ B_\psi\end{bmatrix}\lambda\Bigg),$$
$$(q, u_1, \lambda) \in \mathbb{R}^3 \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [0, 2\pi) \times \mathbb{R} \times \mathbb{R}^4, \tag{27}$$

where

$$\tilde{A}_r \triangleq \begin{bmatrix}0_{3\times3} & \mathbf{1}_3 & 0_{3\times3} & 0_{3\times3}\\ 0_{3\times3} & 0_{3\times3} & \mathbf{1}_3 & 0_{3\times3}\\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \mathbf{1}_3\\ A_{r,0} & A_{r,1} & A_{r,2} & A_{r,3}\end{bmatrix} \in \mathbb{R}^{12\times12}, \quad \tilde{A}_\psi \triangleq \begin{bmatrix}0 & 1\\ A_{\psi,0} & A_{\psi,1}\end{bmatrix} \in \mathbb{R}^{2\times2},$$

are Hurwitz,

$$\tilde{B}_r \triangleq \begin{bmatrix}0_{9\times4}\\ B_r\end{bmatrix} \in \mathbb{R}^{12\times4}, \qquad \tilde{B}_\psi \triangleq \begin{bmatrix}0_{1\times4}\\ B_\psi\end{bmatrix} \in \mathbb{R}^{2\times4}$$

and the pairs $(\tilde{A}_r, \tilde{B}_r)$ and $(\tilde{A}_\psi, \tilde{B}_\psi)$ are controllable. If

$$[\eta_1(t), u_2(t), u_3(t), u_4(t)]^{\mathrm{T}} = \zeta(q(t), u_1(t), \lambda(t)), \qquad t \in [t_0, \infty), \tag{28}$$

then the dynamical system given by (5) and (23) is output-feedback linearized and

$$\dot{x}_{\text{fl}}(t) = A_{\text{fl}}x_{\text{fl}}(t) + B_{\text{fl}}\lambda(t), \qquad t \geq t_0, \tag{29}$$

where $x_{\text{fl}}(t) \triangleq \left[r_A^{\mathrm{T}}(t), \dot{r}_A^{\mathrm{T}}(t), \ddot{r}_A^{\mathrm{T}}(t), r_A^{(3)\mathrm{T}}(t), \psi(t), \dot{\psi}(t)\right]^{\mathrm{T}}$, $A_{\text{fl}} \triangleq \text{blockdiag}(\tilde{A}_r, \tilde{A}_\psi)$ and $B_{\text{fl}} \triangleq \left[\tilde{B}_r^{\mathrm{T}}, \tilde{B}_\psi^{\mathrm{T}}\right]^{\mathrm{T}}$. Thus, any linear control law, such as PID or $\mathcal{H}_\infty$ control, can be applied to design the virtual control input $\lambda(\cdot)$ in (29). Once $\lambda(\cdot)$ is defined, $u_2(\cdot)$, $u_3(\cdot)$, $u_4(\cdot)$ are deduced from (27), and $u_1(\cdot)$ is deduced from (23) (Naldi, Furci, Sanfelice, & Marconi, 2017).

Output-feedback linearization is a valuable technique because it requires the guidance system to produce a reference position and a reference yaw angle, but it does not require to produce reference pitch and roll angles. Additionally, this technique does not require to design the control system by employing inner and outer loops. Lastly, since it leverages the sUAS' nonlinear dynamics, output-feedback linearization allows to perform aggressive maneuvers. However, this technique is affected by multiple limitations. Firstly, in its original formulation, output-feedback linearization requires good knowledge of the sUAS' parameters, since it is not robust to parametric or modeling uncertainties and hence, robust variations need to be employed (Franco et al., 2007). Indeed, it follows from (27) that exact knowledge of the sUAS'

Sidebar 6.3. A challenging problem in the design of control laws for sUAS is given by the fact that if the sUAS' reference trajectory requires a free fall maneuver, that is, if $\ddot{r}^{\mathbb{I}}_{\text{ref}}(t) = -g\mathbf{e}_{3,3} - \ddot{r}^{\mathbb{I}}_{C}(t)$, $t \geq t_0$, then a control law may not be defined (Naldi, Furci, Sanfelice, & Marconi, 2017). However, this problem is usually ignored since sUAS are rarely tasked with this kind of unsafe maneuver.

Sidebar 6.4. A more challenging problem to consider when designing control laws for quadcopters is that the thrust force produced by each propeller, that is, $T_1(\cdot), \ldots, T_4(\cdot)$ must be nonnegative since sUAS almost always are equipped with push propellers. In practical implementations, motors for quadcopters are unable to deliver arbitrarily small angular velocities and hence, $T_1(\cdot), \ldots, T_4(\cdot)$ must be larger than some threshold. This problem requires imposing bounds on the control input, and relatively few frameworks allow to impose constraints on the control input *a priori*. In general, constraints on the control input can be imposed indirectly by tuning the control laws and estimating the control input in a conservative manner. Estimating the control input for a given maneuver may be challenging in numerous applications, and numerous authors implicitly rely on the fact that the reference trajectory does not require aggressive maneuvers and hence, does not require the sUAS to either turn off its motors or exceed the maximum allowed thrust.

inertial parameters and state is needed for (29) to be equivalent to (5) and (23). Secondly, this technique is usually applied assuming that the sUAS' inertial counter-torque and gyroscopic effect given by the presence of rotating propellers is negligible. Although this assumption is substantially valid for vehicles equipped with small propellers, it is not valid for larger vehicles. Thirdly, as it appears from (29), although the sUAS' position and yaw angle can be regulated directly, that is, it is possible to design $\lambda(\cdot)$ so that $r_A(\cdot)$ tracks any reference trajectory and $\psi(\cdot)$ tracks any reference yaw angle outlined by the guidance system, it is not possible to design $\lambda(\cdot)$ so that, concurrently, the roll angle $\phi(\cdot)$ and the pitch angle $\theta(\cdot)$ follow arbitrary reference values. Lastly, as it appears from $x_{\text{fl}}(\cdot)$, the classical output-feedback linearizing laws require knowledge of the first and second time derivatives of the sUAS' acceleration, which can be estimated by means of differentiators or estimators applied to the accelerometer's output, but are usually not measured directly. This problem has been solved in Franco et al. (2007) by implementing a linear observer on the feedback-linearized plant (29).

6.4. Nonlinear control methods

Among the nonlinear control techniques employed to design autopilots for multi-rotor sUAS, backstepping control exploits the cascaded kinematics and dynamics of mechanical systems, and geometric control exploits the vehicle's nonlinear configuration manifold, also in the presence of sling payloads, and allows to compute singularity-free control laws. From a theoretical standpoint, both classical backstepping control (Khalil, 2002, pp. 594–596) and classical geometric control (Bullo & Lewis, 2004, Ch. 5, 10) are not robust to external disturbances or parametric and modeling uncertainties since their original formulations require perfect knowledge of the vehicle's dynamic Eqs. (2) and (4). However, these classical structures have been successfully been tested experimentally (Bouabdallah & Siegwart, 2005; Lee, 2018; Loubar et al., 2019). To formally guarantee the robustness of control laws designed within the backstepping and geometric control frameworks, numerous results have been achieved and tested on multi-rotor sUAS. For instance, robust and adaptive versions of backstepping control has been applied to multi-rotor sUAS in Zhang et al. (2019) and geometric control has been robustified by a variable structure controller in Garcia et al. (2020).

Among the nonlinear control techniques that guarantee robustness to parametric and modeling uncertainties, it is worthwhile to recall direct MRAC (Niit & Smit, 2017). However, in its classical formulation, direct MRAC suffers from the so-called parameter drift, that is, in the presence of external disturbances, the adaptive gains may grow unbounded and so, too, the control input. To overcome this limitation, the so-called *dead-zone modification* (Peterson & Narendra, 1982), *σ-modification* (Ioannou & Fidan, 2006), or *e-modification* (Anderson et al., 2021a) of MRAC must be employed. However, to the authors' knowledge, these modifications of classical MRAC have been rarely employed. A viable alternative is given by the use of the *projection operator* (Xie et al., 2021), but tuning the admissible upper bounds on the adaptive gains may be difficult in practical applications. Recently, the direct MRAC framework has been extended to switched dynamical systems and employed to design a control algorithm for tilt-rotor quadcopters tasked with installing sensors by direct contact (Anderson et al., 2020).

A potential limitation of control techniques based on direct MRAC is that the adaptive gains are not designed to converge to their true values, but are merely designed to remain bounded. In case the vehicle's inertial properties must be estimated in real time for either parameter estimation or fault detection purposes, then alternative approaches are given by *indirect MRAC* (Bakori & Moncayo, 2021), which is designed to guarantee asymptotic convergence of the trajectory tracking error and of the adaptive gains under some persistence of excitation condition on the control input, *composite MRAC* (Dydek et al., 2013), which merges direct and indirect MRAC, $\mathcal{L}_1$ *adaptive control* (Kotaru et al., 2019),

Sidebar 6.5. In multiple applications, linear model predictive control algorithms have been employed to compute the virtual control input $\lambda(\cdot)$ in the feedback-linearized equations of motion (27). However, applying optimization-based algorithms to the feedback-linearized equations of motion guarantees optimality relative to a cost function involving the virtual control input, but not the actual control input $u(\cdot)$.

Sidebar 6.6. ***Classical MRAC and Some Modifications***
The MRAC framework applies to plants in the form

$$\dot{x}(t) = \tilde{A}x(t) + \tilde{B}\Lambda\left[u(t) + \Theta^{\mathrm{T}}\Phi(t, x(t))\right] + \xi(t), \quad x(t_0) = x_0, \quad t \geq t_0, \tag{30}$$

where $A \in \mathbb{R}^{n\times n}$ is unknown, $B \in \mathbb{R}^{n\times m}$ is known, $\Lambda \in \mathbb{R}^{m\times m}$ is diagonal, positive-definite, and unknown, $\Theta \in \mathbb{R}^{N\times m}$ is unknown, $\Phi : [t_0, \infty)\times \mathbb{R}^n \to \mathbb{R}^N$ is known, and $\xi : [t_0, \infty) \to \mathbb{R}^n$ is unknown. This plant model is a generalization of the linearized dynamical model given by (8) or the feedback-linearized dynamical model given (29). The term $\Theta^{\mathrm{T}}\Phi(t, x)$, $(t, x) \in [t_0, \infty) \times \mathbb{R}^n$, allows to account for higher-order nonlinear effects in the UAV dynamics that are neglected in the linearization process or external forces, whose functional dependency are captured by the *regressor vector* $\Phi(\cdot, \cdot)$. The unmateched uncertainty $\xi(\cdot)$ allows to account for any disturbance that is completely unmodeled or any term due to an incorrect output-feedback linearization.

concurrent learning adaptive control (Chowdhary, Wu, Cutler, Ure, & How, 2012), which requires to append recorded data of output and state estimate vectors to the baseline adaptive law and guarantees exponential convergence of the adaptive gains to their ideal values without requiring persistence of excitation of the input functions, and *composite learning adaptive control* (Jiang, Lin, & Chen, 2019), which guarantees exponential parameter convergence to the ideal values and practical exponential stability of the trajectory tracking error without the need of a singular value maximization technique or a fixed-point smoothing technique to compute the time derivatives of the state vector as required by concurrent learning adaptive control. In some cases, neural networks are used as plant emulators in the presence of uncertainties in the sUAS' parameters, and a MRAC law is used to regulate the vehicle using the identified model (Bakshi & Ramachandran, 2016).

A typical drawback of MRAC techniques is their inability to impose some user-defined transient behavior to the closed-loop system. This limitation can be overcome by employing some adaptive control law for prescribed performance (Anderson et al., 2021b; Wu, Ni, Qian, Bu, & Liu, 2021).

An alternative nonlinear robust control strategy is given by *variable structure control*, whereby the closed-loop trajectory is steered towards some user-defined manifold and the trajectory tracking error converges to zero. Among these techniques, we recall classical *sliding mode control* (Xiao, 2020), which guarantees asymptotic convergence of the trajectory tracking error to zero, and *terminal sliding mode control* (Xiong & Zhang, 2017), which guarantees finite-time convergence of the tracking error to zero. A typical drawback of classical variable structure control laws is chattering in neighborhoods of the sliding manifold (Khalil, 2002, pp. 554–556). This problem is usually resolved by employing *super-twisting control* (Jayakrishnan, 2016), *higher-order sliding mode control* (Le Nhu Ngoc Thanh & Hong, 2018), or *adaptive sliding mode control* (Nguyen et al., 2021). However, super-twisting control laws and higher-order sliding mode control laws may excite structural vibration modes in flexible structures of sUAS.

As commented in Sidebar 4.3, classical MPC is robust to uncertainties and external disturbances for being executed at a high frequency. However, the lack of formal guarantees of robustness makes classical MPC more suitable to plan reference trajectories than to control sUAS in the presence of large uncertainties. To overcome this limitation, robust and adaptive formulations of MPC have been presented recently in the literature. In Pereida, Brunke, and Schoellig (2021), a discrete-time $\mathcal{L}_1$ adaptive controller is proposed as an underlying controller to a robust MPC scheme in order to achieve accurate tracking and collision-free trajectories despite parametric uncertainties, and constraints on the input. The controller proposed in Garimella et al. (2017) provides an upper bound on the modeling error, allowing robust MPC to be employed. The uncertainty in the prediction of the state produced by the MPC law is taken into account for collision-free trajectory planning. Furthermore, a safety margin is imposed on the trajectory planner, based on the propagation of model and initial state uncertainties as high-confidence ellipsoids in the state space (Ladosz et al., 2018).

6.5. Data-driven control methods

The control techniques for sUAS surveyed thus far rely on the underlying dynamical models. These techniques, however, may not adapt well to disturbances, such as wind and rain, which directly affect the motion of the sUAS, and other exogenous effects, which affect the sUAS' dynamics indirectly. An example of these effects is given by fluctuations in the environment's temperature that affect the batteries' performance and, hence, the sUAS ability to deliver the required thrust force. Data-driven methods, including learning-based methods, produce control policies by analyzing measured data, adapt to disturbances and changes in the sUAS' dynamics, and, in some cases, generalize the learned policy to different tasks.

With the successful application of machine learning methods in computer vision (Sebe, Cohen, Garg, & Huang, 2005), the very same methods, or variations thereof, have been utilized in vision-based control of sUAS. In Cesetti, Frontoni, Mancini, Zingaretti, and Longhi

Sidebar 6.6 cont'd. According to the MRAC framework, $u(\cdot)$ is computed so that $x(\cdot)$ asymptotically tracks the trajectory of the reference model

$$\dot{x}_{\text{ref}}(t) = A_{\text{ref}}x_{\text{ref}}(t) + B_{\text{ref}}r(t), \qquad \tilde{x}_{\text{ref}} = x_{\text{ref},0}, \qquad t \geq t_0, \quad (31)$$

where $A_{\text{ref}} \in \mathbb{R}^{n\times n}$ is Hurwitz and user-defined and $r : [t_0, \infty) \to \mathbb{R}^m$ is user-defined. The matrices A_{ref} and B_{ref} must be chosen such that

$$A_{\text{ref}} = A + B\Lambda K_x^{\mathrm{T}}, \quad (32)$$
$$B_{\text{ref}} = B\Lambda K_r^{\mathrm{T}}, \quad (33)$$

for some $K_x \in \mathbb{R}^{n\times m}$ and $K_r \in \mathbb{R}^{m\times m}$. Apparently, the matrices K_x and K_r can not be computed since both A and Λ are unknown, but their existence can be postulated knowing the structure of A, which is usually the case for UAVs.

The matching conditions (32) and (33) should not be considered as a limitation to the MRAC framework, but rather summarize its essence. Indeed, MRAC laws confer to the closed-loop plant the same dynamical properties as the reference model. However, this result can only be attained if it is compatible with the plant's structural limits. For instance, if (30) captures the dynamics of a ground vehicle and (31) captures the dynamics of an aerial vehicle, then the matching conditions can not be verified since a ground vehicles can not track an aerial vehicle in three dimensions.

In the MRAC framework, the control input is given by

$$u(t) = \hat{K}_x^{\mathrm{T}}x(t) + \hat{K}_x^{\mathrm{T}}x(t) - \hat{\Theta}^{\mathrm{T}}\Phi(t, x(t)), \qquad t \geq t_0, \quad (34)$$

where the *adaptive gains* $\hat{K}_x : [t_0, \infty) \to \mathbb{R}^{n\times m}$, $\hat{K}_r : [t_0, \infty) \to \mathbb{R}^{m\times m}$, and $\hat{\Theta} : [t_0, \infty) \to \mathbb{R}^{N\times m}$ verify the *adaptive laws*

$$\dot{\hat{K}}_x = -k_1(t)\Gamma_x x(t)e^{\mathrm{T}}(t)PB - k_2(t)\sigma\hat{K}_x(t), \qquad \hat{K}_x(t_0) = K_{x_0}, \quad (35)$$

$$\dot{\hat{K}}_r = -k_1(t)\Gamma_r r(t)e^{\mathrm{T}}(t)PB - k_2(t)\sigma\hat{K}_r(t), \qquad \hat{K}_r(t_0) = K_{r_0}, \quad (36)$$

$$\dot{\hat{\Theta}} = k_1(t)\Gamma_\Phi\Phi(t, x(t))e^{\mathrm{T}}(t)PB - k_2(t)\sigma\hat{\Theta}(t), \qquad \hat{\Theta}(t_0) = \hat{\Theta}_0, \quad (37)$$

$e(t) \triangleq x(t) - x_{\text{ref}}(t)$ denotes the trajectory tracking error, the adaptive rates $\Gamma_x \in \mathbb{R}^{n\times n}$, $\Gamma_r \in \mathbb{R}^{m\times m}$, and $\Gamma_\Phi \in \mathbb{R}^{N\times N}$ are symmetric, positive-definite, and user-defined, $P \in \mathbb{R}^{n\times n}$ is the symmetric, positive-definite solution of the *algebraic Lyapunov equation*

$$0 = A^{\mathrm{T}}P + PA + Q, \quad (38)$$

$Q \in \mathbb{R}^{n\times n}$ is symmetric, positive-definite, and user-defined, $\sigma > 0$, and $k_1(t)$ and $k_2(t)$ are given in Table 8. If $\xi(t) \equiv 0$, $t \geq t_0$, then classical MRAC guarantees asymptotic convergence of the trajectory tracking error. If $\xi(t)$, $t \geq t_0$, is arbitrary, then the dead-zone modification, the σ-modification, and the e-modification of MRAC guarantee uniform ultimate boundedness of the trajectory tracking error.

Sidebar 6.7. The complexity of the dynamical models underlying nonlinear control laws for sUAS typically exceeds the ability of existing guidance systems to outline reference paths or trajectories, while employing the same dynamical models. Thus, autopilots for autonomous multi-rotor sUAS are usually tasked with following paths or trajectories that may not be dynamically feasible. Without sharing a common dynamical model and without exchanging information between the guidance and the control systems for a sUAS, this limitation may be overcome by using explicit reference governors (Hermand et al., 2018). However, to the authors' knowledge, this approach is still under-explored.

Table 8
Coefficients to be employed in the adaptive laws (35)–(37) to implement the classical MRAC law, the dead-zone modification, the σ-modification, and the e-modification of MRAC. Classical MRAC guarantees asymptotic convergence of the trajectory tracking error. However, this technique is not robust to external disturbances and unmatched uncertainties captured by $\xi(\cdot)$ in (30). To guarantee uniform ultimate boundedness of the trajectory tracking error despite external disturbances, the dead-zone modification requires to stop adaptive mechanism whenever the norm of the trajectory tracking error is smaller than a user-defined margin of error $e_0 > 0$. In both the σ-modification and the e-modification of MRAC, the second terms on the right-hand sides of (35)–(37) introduce a damping mechanism to guarantee uniform ultimate boundedness of the trajectory tracking error.

	$\xi(t)$	$k_1(t)$	$k_2(t)$
Classical MRAC	0	1	0
Dear-zone modification	Arbitrary	1 if $\|e(t)\| > e_0$,0 otherwise	0
σ-modification	Arbitrary	1	1
e-modification	Arbitrary	1	$\|e^{\mathrm{T}}(t)PB\|$

(2010), a controller fusion network is learned through a deep neural network for vision-based sUAS racing. A DQN-based control scheme is proposed in Xu, Liu, and Wang (2018) to achieve autonomous landing of a quadcopter equipped with a down-looking camera.

Besides vision-based controls, machine learning techniques have been utilized to produce flight control systems and address limitations of classical control architectures. For instance, in Koch, Mancuso, West, and Bestavros (2019), state-of-the-art reinforcement learning algorithms are incorporated in an intelligent flight control system to provide attitude control of a quadcopter. A learning-based adaptive control algorithm is proposed in Chowdhary, Wu, Cutler, and How (2013) to autonomously transfer control parameters from one aircraft to another. In Sheng, Min, Xiao, and Liu (2018), a deep reinforcement learning technique, called *neural episodic control*, is used to learn a control policy that prevents unauthorized sUAS from entering a target area in a dynamic game. A recurrent wavelet neural network-based control system is designed in Lin, Tai, and Chung (2014) for sUAS motion control in order to achieve desired trajectory tracking. Policy search-based methods focus on learning parameters for a policy parametrization, and they have been used to learn the control policies when applied to sUAS (Zhang, Kahn, Levine, & Abbeel, 2016).

Machine learning methods have also been utilized to learn the dynamics of sUAS and synthesize controls. For example, in Dierks and Jagannathan (2009), a nonlinear controller is proposed based on neural networks and output-feedback with learned dynamics. In Bansal et al. (2016), a deep learning technique learns a dynamical model of the sUAS, and then the learned model is deployed to synthesize controls to track trajectories. Finally, in Becker-Ehmck et al. (2020), a thrust-attitude controller for quadcopters is learned through deep model-based reinforcement learning.

A drawback of current learning-based methods is that they generally suffer from the lack of safety guarantees, which reduce the risk of system failure related to state and control constraints and disturbances. An emerging area of research aims at addressing safety issues in autonomous and intelligent systems through learning-based methods (Yu, Li, Murray, Ramesh, & Tomlin, 2018). In Geramifard, Redding, and How (2013), an intelligent cooperative control architecture is introduced by combining cooperative planners of multi-agent sUAS and safe reinforcement learning techniques to improve the control policy under safety conditions of the system. A learning-based robust control algorithm is presented in Berkenkamp and Schoellig (2015) by incorporating Gaussian process regression in a robust control framework and providing robust stability and performance guarantees during learning. A safety framework for learning-based control of robotic systems is proposed in Fisac et al. (2019) to guarantee constraint satisfaction, while minimally interfering with the learning process. In Shao, Chen, Kousik, and Vasudevan (2021), a reachability-based trajectory safeguard method is proposed by leveraging a reachability analysis and reinforcement learning to identify safe controls and adjust unsafe ones to ensure safety of the sUAS.

Some other data-driven control methods applied to sUAS assume no prior information on the sUAS model and apply a model-free control. Data-driven control approaches including virtual reference feedback tuning (Campi, Lecchini, & Savaresi, 2002) and correlation-based tuning (Van Heusden, Karimi, & Bonvin, 2011) are implemented on a quadcopter for attitude control in Invernizzi, Panizza, Riccardi, Formentin, and Lovera (2016) to test their robustness in the presence of disturbances. A hybrid control method that integrates PID control with data-driven sliding mode control is proposed in Liu et al. (2020) to deal with trajectory tracking control of sUAS with unknown dynamics and disturbances with robustness guarantees due to the design of the data-driven sliding surface. A data-driven, model-free, MPC approach is presented in Coulson, Lygeros, and Dörfler (2019) to control a quadcopter in simulation, and a robust version of data-driven MPC for linear time-invariant systems is introduced in Berberich, Köhler, Müller, and Allgöwer (2020).

7. Concluding remarks and future research directions

This survey on guidance, navigation, and control systems for multi-rotor sUAS highlighted how each of these three research areas have been extensively investigated in the past decade to produce truly autonomous vehicles. The production of commercial sUAS implementing some of the surveyed guidance, navigation, and control systems certifies how some of the existing solutions are sufficiently mature to support several current recreational, agricultural, industrial, and defense applications, such as those surveyed in Section 2. This survey also highlighted how some recent research results will soon enable some of the future applications examined in Section 2. However, the problem of researching guidance, navigation, and control systems for multi-rotor sUAS is still far from being considered as closed. For instance, the state-of-the-art in the design and production of sensors and single-board computers for sUAS needs to be further advanced before some of the most complex solutions, which are designed to guarantee high-quality results despite uncertainties and disturbances, are applied in real time. On the other hand, complex solutions could be further investigated and adapted to operate in real time compatibly with the available hardware. Additionally, data-driven approaches to guidance, navigation, and control systems design, which are cast in the discrete-time domain and loosely dependent on the acquisition frequency of onboard sensors, are needed to substantially advance the state-of-the-art. Lastly, the problem of creating integrated guidance, navigation, and control systems has been under-explored. Although the efficiency of non-coordinated or non-integrated guidance, navigation, and control systems is still to be assessed, it is realistic to assume that some form of coordination or integration may only improve the overall system's performance.

Thus far, some efforts to integrate some solutions to the guidance, navigation, and control problems have been made. For instance, maneuver automata and motion primitive libraries significantly help reducing the gap between path and trajectory planning subsystems. Furthermore, employing linear robust or nonlinear versions of the MPC architecture for trajectory planning and either directly applying the underlying control input or employing such control input as a baseline controller for some nonlinear robust control law significantly contributes to bridging the gap between guidance and control systems. Additional approaches, however, can still be attempted. For instance, the dynamical models underlying guidance, navigation, and control systems on a given quadcopter are not necessarily the same or even similar. Thus, a preliminary form of integration of the guidance, navigation, and control systems could require the use of the same or compatible dynamical models. However, to date, guidance systems executed onboard sUAS are still unable to produce in real time reference trajectories by employing dynamical models that are

as complex as those employed by the most advanced nonlinear robust control algorithms. This gap may be partly bridged by switching across dynamical models of different complexity to allow accordance among dynamical models in guidance, navigation, and control systems at least for simpler tasks or in nominal conditions. However, the complexity of applying extensions of the Barbalat's lemma or the LaSalle–Yoshizawa theorem within switched system frameworks, such as the Carathéodory or the Filippov frameworks, and the difficulty of finding either common Lyapunov functions or suitable state-dependent dwell-times and switching times still hinder the extension of many nonlinear robust control techniques to switched cases.

A more advanced attempt to integrate guidance, navigation, and control systems would consist in increasing the amount of relevant information exchanged among these systems. For instance, to the authors' knowledge, there is no guidance or navigation system explicitly designed to elaborate on information on the sUAS' actual dynamics deduced by the system identification and parameter estimation features of some control algorithms, such as some of the adaptive control laws surveyed in Section 6.4. The potential problems caused by such exchange of information, however, should not be underestimated. Indeed, the stability of the integrated guidance, navigation, and control systems should be proven; also in the case each of the guidance, navigation, and control systems are provably stable, the stability of the assembled system is still to be proven. Furthermore, some optimization-based trajectory planning methods explicitly rely on the assumption that the matrices or the nonlinear functions that capture the vehicle's dynamics are time-invariant and hence, independent from any effect caused by external systems. Similarly, most SLAM algorithms assume that the sUAS' dynamic properties are time-invariant and the world is substantially static, or evolve at a rate slower than the SLAM algorithm is standard. Only very recently, some advances have been made to extend SLAM methods in dynamic environments.

An additional problem that is still under-explored concerns the synchronization of the guidance, navigation, and control systems for sUAS. The most commonly used approach is to execute the algorithms underlying these systems as fast as possible and reduce bottlenecks. However, the fact that this approach produces better results in terms of computational burden and accuracy is to be proven. Furthermore, the frequency at which these algorithms are executed is either set irrespectively of the sUAS' translational velocity or is carefully tuned according to the expected forward velocity. Future research directions may involve studies on adaptive regulations of the frequency of execution of guidance, navigation, and control algorithms as functions of the sUAS' translational velocity and the complexity of the task being performed.

A complex and open question concerns the extent to which the guidance, navigation, and control systems should be integrated. For example, an autopilot architecture involving a guidance and a control system that underlie the same, but overly simplistic, dynamical model would not perform well in cluttered environments, in the presence of significant disturbances, and in the presence of faults and failures of the motors or other components. Conversely, an architecture involving a guidance and a control system that employ the same, but overly complex, dynamical model may be unnecessarily slow, considering the state-of-the-art of single-board computers for sUAS. Similarly, the inclusion of excessively complicated dynamics may lead to a decrease in efficiency of navigation systems and their ability to operate in real time, especially in multi-agent scenarios. Thus, a question concerns the trade-off between the extent to which guidance, navigation, and control systems should be integrated and should exchange relevant information.

Several DoD branches and forward-thinking companies are developing considerable interest in sUAS to be employed, for instance, for last-mile delivery of goods from local warehouses. Moreover, the use of autonomous aircraft, which are free of pilots and are able to transport from one to four passengers across cities, is considered by many stakeholders as a potential solution to the increasingly complex problem of urban mobility. To address these technological challenges, a common trend is to adapt the theoretical and technological solutions developed for sUAS. However, the gaps between existing guidance, navigation, and control solutions, which may be still considered as evanescent for small aircraft, are more likely to become substantial for complex, heavy, and powerful vehicles operating in close contact with humans and in confined environments. Addressing these research problems and bridging the gaps between guidance, navigation, and control problems will require significant effort from the aerial robotics community, or part thereof. However, addressing these problems at these relatively early stages in the history of aerial robotics will likely ease scaling some of the existing technological solutions and will propel this community towards new challenges.

Declaration of competing interest

This work was partly supported by the Office of Naval Research, the US Department of the Navy, and the National Science Foundation under Grants no. N00014N-19-1-24N22, N004N212110004N, and DUE-1700064N0 respectvely.

References

Abdelkader, M., Güler, S., Jaleel, H., & Shamma, J. S. (2021). Aerial swarms: Recent applications and challenges. *Current Robotics Reports*, *2*(3), 309–320. http://dx.doi.org/10.1007/s43154-021-00063-4.

Adıgüzel, F., & Mumcu, T. V. (2019). Discrete-time backstepping attitude control of a quadrotor UAV. In *International artificial intelligence and data processing symposium* (pp. 1–5). http://dx.doi.org/10.1109/IDAP.2019.8875891.

Aggarwal, S., & Kumar, N. (2020). Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications*, *149*, 270–299. http://dx.doi.org/10.1016/j.comcom.2019.10.014.

Ait-Jellal, R., & Zell, A. (2017). Outdoor obstacle avoidance based on hybrid visual stereo SLAM for an autonomous quadrotor MAV. In *European conference on mobile robots* (pp. 1–8). IEEE, http://dx.doi.org/10.1109/ECMR.2017.8098686.

Almeida, M. M. d., Moghe, R., & Akella, M. (2019). Real-time minimum snap trajectory generation for quadcopters: Algorithm speed-up through machine learning. In *International conference on robotics and automation* (pp. 683–689). IEEE, http://dx.doi.org/10.1109/ICRA.2019.8793569.

Alonso-Mora, J., Baker, S., & Rus, D. (2017). Multi-robot formation control and object transport in dynamic environments via constrained optimization. *International Journal of Robotics Research*, *36*(9), 1000–1021. http://dx.doi.org/10.1177/0278364917719333.

Ames, A. D., Notomista, G., Wardi, Y., & Egerstedt, M. (2021). Integral control barrier functions for dynamically defined control laws. *IEEE Control Systems Letters*, *5*(3), 887–892. http://dx.doi.org/10.1109/LCSYS.2020.3006764.

Anandharaman, S., Sudhakaran, M., & Seyezhai, R. (2016). A low-cost visual navigation and mapping system for unmanned aerial vehicle using LSD-SLAM algorithm. In *Online international conference on green engineering and technologies* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/GET.2016.7916802.

Anderson, R. B., Marshall, J. A., & L'Afflitto, A. (2021a). Constrained robust model reference adaptive control of a tilt-rotor quadcopter pulling an unmodeled cart. *Transactions on Aerospace and Electronic Systems*, *57*(1), 39–54. http://dx.doi.org/10.1109/TAES.2020.3008575.

Anderson, R. B., Marshall, J. A., & L'Afflitto, A. (2021b). Novel model reference adaptive control laws for improved transient dynamics and guaranteed saturation constraints. *Journal of the Franklin Institute*, http://dx.doi.org/10.1016/j.jfranklin.2021.06.020.

Anderson, R. B., Marshall, J. A., L'Afflitto, A., & Dotterweich, J. M. (2020). Model reference adaptive control of switched dynamical systems with applications to aerial robotics. *Journal of Intelligent & Robotic Systems*, *100*(3), 1265–1281. http://dx.doi.org/10.1007/s10846-020-01260-7.

Arabi, E., Gruenwald, B. C., Yucelen, T., & Nguyen, N. T. (2017). A set-theoretic model reference adaptive control architecture for disturbance rejection and uncertainty suppression with strict performance guarantees. *International Journal of Control*, 1–14. http://dx.doi.org/10.1080/00207179.2017.1312019, In press.

Ardupilot Firmware (2021). Retrieved from https://ardupilot.org/. (Accessed 20 September 2021).

Arslan, O., & Koditschek, D. E. (2017). Smooth extensions of feedback motion planners via reference governors. In *International conference on robotics and automation* (pp. 4414–4421). IEEE, http://dx.doi.org/10.1109/ICRA.2017.7989510.

Atanasov, N., Bowman, S. L., Daniilidis, K., & Pappas, G. J. (2018). A unifying view of geometry, semantics, and data association in SLAM. In *International joint conferences on artificial intelligence* (pp. 5204–5208). http://dx.doi.org/10.24963/ijcai.2018/722.

Atyabi, A., MahmoudZadeh, S., & Nefti-Meziani, S. (2018). Current advancements on autonomous mission planning and management systems: An AUV and UAV perspective. *Annual Reviews in Control*, *46*, 196–215. http://dx.doi.org/10.1016/j.arcontrol.2018.07.002.

Aurenhammer, F. (1987). Power diagrams: Properties, algorithms and applications. *SIAM Journal on Computing*, *16*(1), 78–96. http://dx.doi.org/10.1137/0216006.

Babel, L. (2019). Coordinated target assignment and UAV path planning with timing constraints. *Journal of Intelligent and Robotic Systems*, *94*(3–4), 857–869. http://dx.doi.org/10.1007/s10846-018-0910-9.

Baek, J., Han, S. I., & Han, Y. (2020). Energy-efficient UAV routing for wireless sensor networks. *Transactions on Vehicular Technology*, *69*(2), 1741–1750. http://dx.doi.org/10.1109/TVT.2019.2959808.

Bakori, M. S., & Moncayo, H. (2021). Comparison of discrete direct and indirect adaptive control laws for UAV fault-tolerance. In *AIAA scitech forum* (p. 1127). http://dx.doi.org/10.2514/6.2021-1127.

Bakshi, N. A., & Ramachandran, R. (2016). Indirect model reference adaptive control of quadrotor UAVs using neural networks. In *International conference on intelligent systems and control* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/ISCO.2016.7727123.

Bansal, S., Akametalu, A. K., Jiang, F. J., Laine, F., & Tomlin, C. J. (2016). Learning quadrotor dynamics using neural network for flight control. In *Conference on decision and control* (pp. 4653–4660). IEEE, http://dx.doi.org/10.1109/CDC.2016.7798978.

Basiri, M. (2015). *Audio-based positioning and target localization for swarms of micro aerial vehicles* (Ph.D. thesis), Lausanne, Switzerland: École polytechnique fédérale de Lausanne, http://dx.doi.org/10.5075/epfl-thesis-6508.

Basiri, M., Schill, F., Lima, P., & Floreano, D. (2016). On-board relative bearing estimation for teams of drones using sound. *Robotics and Automation Letters*, *1*(2), 820–827. http://dx.doi.org/10.1109/LRA.2016.2527833.

Becker-Ehmck, P., Karl, M., Peters, J., & van der Smagt, P. (2020). Learning to fly via deep model-based reinforcement learning. arXiv preprint arXiv:2003.08876.

Behley, J., & Stachniss, C. (2018). Efficient surfel-based SLAM using 3D laser range data in urban environments. In *Robotics: science and systems* (pp. 1–10). http://dx.doi.org/10.15607/RSS.2018.XIV.016.

Bemporad, A. (1998). Reference governor for constrained nonlinear systems. *Transactions on Automatic Control*, *43*(3), 415–419. http://dx.doi.org/10.1109/9.661611.

Berberich, J., Köhler, J., Müller, M. A., & Allgöwer, F. (2020). Data-driven model predictive control with stability and robustness guarantees. *IEEE Transactions on Automatic Control*, *66*(4), 1702–1717. http://dx.doi.org/10.1109/TAC.2020.3000182.

Berger, C., Rudol, P., Wzorek, M., & Kleiner, A. (2016). Evaluation of reactive obstacle avoidance algorithms for a quadcopter. In *International conference on control, automation, robotics and vision* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/ICARCV.2016.7838803.

Berkenkamp, F., & Schoellig, A. P. (2015). Safe and robust learning control with Gaussian processes. In *European control conference* (pp. 2496–2501). IEEE, http://dx.doi.org/10.1109/ECC.2015.7330913.

Bernstein, D. S. (1993). Nonquadratic cost and nonlinear feedback control. *International Journal of Robust and Nonlinear Control*, *3*(3), 211–229. http://dx.doi.org/10.1002/rnc.4590030303.

Besl, P. J., & McKay, N. D. (1992). Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures, Vol. 1611* (pp. 586–606). International Society for Optics and Photonics, http://dx.doi.org/10.1117/12.57955.

Blackmore, L., Ono, M., & Williams, B. C. (2011). Chance-constrained optimal path planning with obstacles. *Transactions on Robotics*, *27*(6), 1080–1094. http://dx.doi.org/10.1109/TRO.2011.2161160.

Blasi, L., D'Amato, E., Mattei, M., & Notaro, I. (2020). Path planning and real-time collision avoidance based on the essential visibility graph. *Applied Sciences*, *10*(16), http://dx.doi.org/10.3390/app10165613.

Bokovoy, A., & Yakovlev, K. (2018). Sparse 3D point-cloud map upsampling and noise removal as a vSLAM post-processing step: Experimental evaluation. In A. Ronzhin, G. Rigoll, & R. Meshcheryakov (Eds.), *Interactive collaborative robotics* (pp. 23–33). Cham: Springer, http://dx.doi.org/10.1007/978-3-319-99582-3_3.

Bouabdallah, S., & Siegwart, R. (2005). Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In *International conference on robotics and automation* (pp. 2247–2252). IEEE, http://dx.doi.org/10.1109/ROBOT.2005.1570447.

Bowman, S. L., Atanasov, N., Daniilidis, K., & Pappas, G. J. (2017). Probabilistic data association for semantic SLAM. In *International conference on robotics and automation* (pp. 1722–1729). IEEE, http://dx.doi.org/10.1109/ICRA.2017.7989203.

Breitenmoser, A., Kneip, L., & Siegwart, R. (2011). A monocular vision-based system for 6D relative robot localization. In *International conference on intelligent robots and systems* (pp. 79–85). IEEE, http://dx.doi.org/10.1109/IROS.2011.6094851.

Bryson, A. E. (1975). *Halsted Press book, Applied optimal control: optimization, estimation and control*. New York, NY: Taylor & Francis.

Bullo, F., & Lewis, A. (2004). *Texts in Applied Mathematics, Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*. New York, NY: Springer.

Cabreira, T. M., Brisolara, L. B., & Ferreira, P. R., Jr. (2019). Survey on coverage path planning with unmanned aerial vehicles. *Drones*, *3*(1), http://dx.doi.org/10.3390/drones3010004.

Cabrera-Ponce, A. A., Martinez-Carranza, J., & Rascon, C. (2020). Detection of nearby UAVs using a multi-microphone array on board a UAV. *International Journal of Micro Air Vehicles*, *12*, Article 1756829320925748.

Calkins, L., Lingevitch, J., McGuire, L., Geder, J., Kelly, M., Zavlanos, M. M., et al. (2020). Bio-inspired distance estimation using the self-induced acoustic signature of a motor-propeller system. In *International conference on robotics and automation* (pp. 5047–5053). IEEE, http://dx.doi.org/10.1109/ICRA40945.2020.9197143.

Campi, M. C., Lecchini, A., & Savaresi, S. M. (2002). Virtual reference feedback tuning: a direct method for the design of feedback controllers. *Automatica*, *38*(8), 1337–1346. http://dx.doi.org/10.1016/S0005-1098(02)00032-8.

Campos, C., Elvira, R., Rodríguez, J. J. G., M. Montiel, J. M., & D. Tardós, J. (2021). ORB-SLAM3: An accurate open-source library for visual, visual–Inertial, and multimap SLAM. *Transactions on Robotics*, 1–17. http://dx.doi.org/10.1109/TRO.2021.3075644.

Canny, J. (1988). *The complexity of robot motion planning* (Ph.D. thesis), Cambridge, MA: Massachusetts Institute of Technology.

Carrio, A., Sampedro, C., Rodriguez-Ramos, A., & Campoy, P. (2017). A review of deep learning methods and applications for unmanned aerial vehicles. *Journal of Sensors*, Article 3296874. http://dx.doi.org/10.1155/2017/3296874.

Castillo-Lopez, M., Sajadi-Alamdari, S. A., Sanchez-Lopez, J. L., Olivares-Mendez, M. A., & Voos, H. (2018). Model predictive control for aerial collision avoidance in dynamic environments. In *Mediterranean conference on control and automation* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/MED.2018.8442967.

Cesetti, A., Frontoni, E., Mancini, A., Zingaretti, P., & Longhi, S. (2010). A vision-based guidance system for UAV navigation and safe landing using natural landmarks. *Journal of Intelligent and Robotic Systems*, *57*(1), 233–257. http://dx.doi.org/10.1007/s10846-009-9373-3.

Challita, U., Saad, W., & Bettstetter, C. (2019). Interference management for cellular-connected UAVs: A deep reinforcement learning approach. *IEEE Transactions on Wireless Communication*, *18*(4), 2125–2140. http://dx.doi.org/10.1109/TWC.2019.2900035.

Chaudhry, A., Misovec, K., & D'Andrea, R. (2004). Low observability path planning for an unmanned air vehicle using mixed integer linear programming. In *IEEE conference on decision and control, Vol. 4* (pp. 3823–3829). http://dx.doi.org/10.1109/CDC.2004.1429334.

Cheeseman, P., Smith, R., & Self, M. (1987). A stochastic map for uncertain spatial relationships. In *International symposium on robotic research* (pp. 467–474). MIT Press Cambridge, http://dx.doi.org/10.5555/57425.57472.

Choi, Y., Jimenez, H., & Mavris, D. N. (2017). Two-layer obstacle collision avoidance with machine learning for more energy-efficient unmanned aircraft trajectories. *Robotics and Autonomous Systems*, *98*, 158–173. http://dx.doi.org/10.1016/j.robot.2017.09.004.

Choi, H., Kim, Y., & Hwang, I. (2011). Vision-based reactive collision avoidance algorithm for unmanned aerial vehicle. In *Guidance, navigation, and control conference* (pp. 1–18). Portland, OR: AIAA, http://dx.doi.org/10.2514/6.2011-6603.

Choudhary, S., Carlone, L., Nieto, C., Rogers, J., Christensen, H. I., & Dellaert, F. (2017). Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models. *International Journal of Robotics Research*, *36*(12), 1286–1311. http://dx.doi.org/10.1177/0278364917732640.

Chowdhary, G., Wu, T., Cutler, M., & How, J. P. (2013). Rapid transfer of controllers between UAVs using learning-based adaptive control. In *International conference on robotics and automation* (pp. 5409–5416). IEEE, http://dx.doi.org/10.1109/ICRA.2013.6631353.

Chowdhary, G., Wu, T., Cutler, M., Ure, N. K., & How, J. (2012). Experimental results of concurrent learning adaptive controllers. In *Guidance, navigation, and control conference* (p. 4551). Minneapolis, MN: AIAA, http://dx.doi.org/10.2514/6.2012-4551.

Cieslewski, T., Choudhary, S., & Scaramuzza, D. (2018). Data-efficient decentralized visual SLAM. In *International conference on robotics and automation* (pp. 2466–2473). IEEE, http://dx.doi.org/10.1109/ICRA.2018.8461155.

Coppola, M., McGuire, K. N., De Wagter, C., & de Croon, G. C. H. E. (2020). A survey on swarming with micro air vehicles: Fundamental challenges and constraints. *Frontiers in Robotics and AI*, *7*, 18. http://dx.doi.org/10.3389/frobt.2020.00018.

Coulson, J., Lygeros, J., & Dörfler, F. (2019). Data-enabled predictive control: In the shallows of the deePC. In *European control conference* (pp. 307–312). IEEE, http://dx.doi.org/10.23919/ECC.2019.8795639.

Cowling, I. D., Yakimenko, O. A., Whidborne, J. F., & Cooke, A. K. (2010). Direct method based control system for an autonomous quadrotor. *Journal of Intelligent & Robotic Systems*, *60*(2), 285–316. http://dx.doi.org/10.1007/s10846-010-9416-9.

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *Transactions on Pattern Analysis and Machine Intelligence*, *29*(6), 1052–1067. http://dx.doi.org/10.1109/TPAMI.2007.1049.

De Marina, H. G., Pereda, F. J., Giron-Sierra, J. M., & Espinosa, F. (2011). UAV attitude estimation using unscented Kalman filter and TRIAD. *IEEE Transactions on Industrial Electronics*, *59*(11), 4465–4474. http://dx.doi.org/10.1109/TIE.2011.2163913.

de Souza, J. P. C., Marcato, A. L. M., de Aguiar, E. P., Juca, M. A., & Teixeira, A. M. (2019). Autonomous landing of UAV based on artificial neural network supervised by fuzzy logic. *Journal of Control, Automation and Electrical Systems*, *40*(4), 522–531. http://dx.doi.org/10.1007/s40313-019-00465-y.

Deits, R., & Tedrake, R. (2015a). Computing large convex regions of obstacle-free space through semidefinite programming. In *Algorithmic foundations of robotics XI* (pp. 109–124). Springer.

Deits, R., & Tedrake, R. (2015b). Efficient mixed-integer planning for UAVs in cluttered environments. In *International conference on robotics and automation* (pp. 42–49). IEEE, http://dx.doi.org/10.1109/ICRA.2015.7138978.

Delmerico, J., & Scaramuzza, D. (2018). A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *International conference on robotics and automation* (pp. 2502–2509). IEEE, http://dx.doi.org/10.1109/ICRA.2018.8460664.

Dierks, T., & Jagannathan, S. (2009). Output feedback control of a quadrotor UAV using neural networks. *IEEE Transactions on Neural Networks*, *21*(1), 50–66. http://dx.doi.org/10.1109/TNN.2009.2034145.

Dong, J., Nelson, E., Indelman, V., Michael, N., & Dellaert, F. (2015). Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach. In *International conference on robotics and automation* (pp. 5807–5814). IEEE, http://dx.doi.org/10.1109/ICRA.2015.7140012.

Droeschel, D., & Behnke, S. (2018). Efficient continuous-time SLAM for 3D lidar-based online mapping. In *International conference on robotics and automation* (pp. 5000–5007). IEEE, http://dx.doi.org/10.1109/ICRA.2018.8461000.

Droeschel, D., Stückler, J., & Behnke, S. (2014). Local multi-resolution representation for 6D motion estimation and mapping with a continuously rotating 3D laser scanner. In *International conference on robotics and automation* (pp. 5221–5226). IEEE, http://dx.doi.org/10.1109/ICRA.2014.6907626.

Dubé, R., Gollub, M. G., Sommer, H., Gilitschenski, I., Siegwart, R., Cadena, C., et al. (2018). Incremental-segment-based localization in 3-D point clouds. *IEEE Robotics and Automation Letters*, *3*(3), 1832–1839. http://dx.doi.org/10.1109/LRA.2018.2803213.

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: Part I. *IEEE Robotics & Automation Magazine*, *13*(2), 99–110. http://dx.doi.org/10.1109/MRA.2006.1638022.

Dydek, Z. T., Annaswamy, A. M., & Lavretsky, E. (2013). Adaptive control of quadrotor UAVs: A design trade study with flight evaluations. *Transactions on Control Systems Technology*, *21*(4), 1400–1406. http://dx.doi.org/10.1109/TCST.2012.2200104.

Engebraaten, S. A., Moen, J., Yakimenko, O. A., & Glette, K. (2020). A framework for automatic behavior generation in multi-function swarms. *Frontiers in Robotics and AI*, *7*, 175. http://dx.doi.org/10.3389/frobt.2020.579403.

Engel, J., Koltun, V., & Cremers, D. (2018). Direct sparse odometry. *Transactions on Pattern Analysis and Machine Intelligence*, *40*(3), 611–625. http://dx.doi.org/10.1109/TPAMI.2017.2658577.

Engel, J., Schops, T., & Cremers, D. (2014). LSD-SLAM: Large scale direct monocular SLAM. In *European conference on computer vision* (pp. 834–849). Springer, http://dx.doi.org/10.1007/978-3-319-10605-2_54.

Eslamiat, H., Li, Y., Wang, N., Sanyal, A. K., & Qiu, Q. (2019). Autonomous waypoint planning, optimal trajectory generation and nonlinear tracking control for multi-rotor UAVs. In *European control conference* (pp. 2695–2700). IEEE, http://dx.doi.org/10.23919/ECC.2019.8795855.

Esrafilian, O., & Taghirad, H. D. (2016). Autonomous flight and obstacle avoidance of a quadrotor by monocular SLAM. In *International conference on robotics and mechatronics* (pp. 240–245). IEEE, http://dx.doi.org/10.1109/ICRoM.2016.7886853.

Faessler, M., Mueggler, E., Schwabe, K., & Scaramuzza, D. (2014). A monocular pose estimation system based on infrared LEDs. In *International conference on robotics and automation* (pp. 907–913). IEEE, http://dx.doi.org/10.1109/ICRA.2014.6906962.

Faulwasser, T., & Findeisen, R. (2015). Nonlinear model predictive control for constrained output path following. *Transactions on Automatic Control*, *61*(4), 1026–1039.

Feng, Y., Rabbath, C. A., & Su, C.-Y. (2018). Modeling of the dynamics of a micro UAV with a single slung load. In *Handbook of unmanned aerial vehicles* (pp. 1–19). Cham: Springer, http://dx.doi.org/10.1007/978-3-319-32193-6_108-2.

Fink, G., Franke, M., Lynch, A. F., Röbenack, K., & Godbolt, B. (2017). Visual inertial SLAM: Application to unmanned aerial vehicles. *IFAC-PapersOnLine*, *50*(1), 1965–1970. http://dx.doi.org/10.1016/j.ifacol.2017.08.162.

First responder tactical beyond visual line of sight (TBVLOS) 91.113 waiver guide. (2021). (Accessed 7 June 2021).

Fisac, J. F., Akametalu, A. K., Zeilinger, M. N., Kaynama, S., Gillula, J., & Tomlin, C. J. (2019). A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, *64*(7), 2737–2752. http://dx.doi.org/10.1109/TAC.2018.2876389.

Forster, C., Lynen, S., Kneip, L., & Scaramuzza, D. (2013). Collaborative monocular SLAM with multiple micro aerial vehicles. In *International conference on intelligent robots and systems* (pp. 3962–3970). IEEE, http://dx.doi.org/10.1109/IROS.2013.6696923.

Fraga-Lamas, P., Ramos, L., Mondéjar-Guerra, V., & Fernández-Caramés, T. M. (2019). A review on IoT deep learning UAV systems for autonomous obstacle detection and collision avoidance. *Remote Sensing*, *11*(18), http://dx.doi.org/10.3390/rs11182144.

Franco, A. L. D., Bourlès, H., & De Pieri, E. R. (2007). Robust feedback linearization without full state information. *IFAC Proceedings Volumes*, *40*(20), 70–75. http://dx.doi.org/10.3182/20071017-3-BR-2923.00012.

Frazzoli, E., Dahleh, M., & Feron, E. (1999). A hybrid control architecture for aggressive maneuvering of autonomous helicopters. In *Conference on decision and control, Vol. 3* (pp. 2471–2476). IEEE, http://dx.doi.org/10.1109/CDC.1999.831296.

Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, *61*(12), 1258–1276. http://dx.doi.org/10.1016/j.robot.2013.09.004.

Garcia, O., Rojo-Rodriguez, E. G., Sanchez, A., Saucedo, D., & Munoz-Vazquez, A. J. (2020). Robust geometric navigation of a quadrotor UAV on $SE(3)$. *Robotica*, *38*(6), 1019–1040. http://dx.doi.org/10.1017/S0263574719001231.

Garimella, G., Sheckells, M., & Kobilarov, M. (2017). Robust obstacle avoidance for aerial platforms using adaptive model predictive control. In *International conference on robotics and automation* (pp. 5876–5882). IEEE.

Geraerts, R., & Schager, E. (2010). Stealth-based path planning using corridor maps. In *Computer animation and social agents* (pp. 1–4).

Geramifard, A., Redding, J., & How, J. P. (2013). Intelligent cooperative control architecture: a framework for performance improvement using safe learning. *Journal of Intelligent and Robotic Systems*, *72*(1), 83–103. http://dx.doi.org/10.1007/s10846-013-9826-6.

Ghaemi, R., Sun, J., & Kolmanovsky, I. (2010). A neighboring extremal approach to nonlinear model predictive control. In *Symposium on nonlinear control systems, Vol. 43* (pp. 747–752). IFAC, http://dx.doi.org/10.3182/20100901-3-IT-2016.00111.

Gilbert, E. G., Kolmanovsky, I., & Tan, K. T. (1995). Discrete-time reference governors and the nonlinear control of systems with state and control constraints. *International Journal of Robust and Nonlinear Control*, *5*(5), 487–504. http://dx.doi.org/10.1002/rnc.4590050508.

Gilbert, E. G., & Ong, C.-J. (2011). Constrained linear systems with hard constraints and disturbances: An extended command governor with large domain of attraction. *Automatica*, *47*(2), 334–340. http://dx.doi.org/10.1016/j.automatica.2010.10.016.

Goerzen, C., Kong, Z., & Mettler, B. (2009). A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems*, *57*(1), 65. http://dx.doi.org/10.1007/s10846-009-9383-1.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Grewal, M., & Andrews, A. (2015). *Kalman filtering: theory and practice with MATLAB*. Hoboken, NJ: IEEE Press, Wiley.

Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, *23*(1), 34–46. http://dx.doi.org/10.1109/TRO.2006.889486.

Grisetti, G., Stachniss, C., Grzonka, S., & Burgard, W. (2007). A tree parameterization for efficiently computing maximum likelihood maps using gradient descent.. In *Robotics: science and systems, Vol. 3* (p. 9). http://dx.doi.org/10.15607/RSS.2007.III.009.

Grüne, L. (2009). Analysis and design of unconstrained nonlinear MPC schemes for finite and infinite dimensional systems. *SIAM Journal on Control and Optimization*, *48*(2), 1206–1228.

Grüne, L. (2012). NMPC without terminal constraints. *IFAC Proceedings Volumes*, *45*(17), 1–13.

Gustavsson, K. (2015). UAV pose estimation using sensor fusion of inertial, sonar and satellite signals. (p. 45). Engineering Physics.

Haddad, W. M., & L'Afflitto, A. (2016). Finite-time stabilization and optimal feedback control. *Transactions on Automatic Control*, *61*(4), 1069–1074. http://dx.doi.org/10.1109/TAC.2015.2454891.

Hafez, A. T., Givigi, S. N., & Yousefi, S. (2018). Unmanned aerial vehicles formation using learning based model predictive control. *Asian Journal of Control*, *20*(3), 1014–1026. http://dx.doi.org/10.1002/asjc.1774.

Hajiyev, C., & Soken, H. E. (2013). Robust adaptive Kalman filter for estimation of UAV dynamics in the presence of sensor/actuator faults. *Aerospace Science and Technology*, *28*(1), 376–383. http://dx.doi.org/10.1016/j.ast.2012.12.003.

Hamrah, R., & Sanyal, A. K. (2020). Finite-time stable tracking control for an underactuated system in $SE(3)$ in discrete time. *International Journal of Control*, 1–16. http://dx.doi.org/10.1080/00207179.2020.1841299.

Han, L., Gao, F., Zhou, B., & Shen, S. (2019). FIESTA: Fast incremental euclidean distance fields for online motion planning of aerial robots. In *International conference on intelligent robots and systems* (pp. 4423–4430). IEEE, http://dx.doi.org/10.1109/IROS40897.2019.8968199.

Han, Z., Zhang, R., Pan, N., Xu, C., & Gao, F. (2020). Fast-tracker: A robust aerial system for tracking agile target in cluttered environments. arXiv preprint arXiv:2011.03968.

Hebecker, T., Buchholz, R., & Ortmeier, F. (2015). Model-based local path planning for UAVs. *Journal of Intelligent & Robotic Systems*, *78*(1), 127–142. http://dx.doi.org/10.1007/s10846-014-0097-7.

Hehn, M., & D'Andrea, R. (2015). Real-time trajectory generation for quadrocopters. *Transactions on Robotics*, *31*(4), 877–892. http://dx.doi.org/10.1109/TRO.2015.2432611.

Heo, S., Jung, J. H., & Park, C. G. (2018). Consistent EKF-based visual-inertial navigation using points and lines. *IEEE Sensors Journal*, *18*(18), 7638–7649. http://dx.doi.org/10.1109/JSEN.2018.2858276.

Hermand, E., Nguyen, T. W., Hosseinzadeh, M., & Garone, E. (2018). Constrained control of UAVs in geofencing applications. In *Mediterranean conference on control and automation* (pp. 217–222). IEEE, http://dx.doi.org/10.1109/MED.2018.8443035.

Hess, W., Kohler, D., Rapp, H., & Andor, D. (2016). Real-time loop closure in 2D LIDAR SLAM. In *International conference on robotics and automation* (pp. 1271–1278). IEEE, http://dx.doi.org/10.1109/ICRA.2016.7487258.

Homer, T., Mahmood, M., & Mhaskar, P. (2020). A trajectory-based method for constructing null controllable regions. *International Journal of Robust and Nonlinear Control*, *30*(2), 776–786. http://dx.doi.org/10.1002/rnc.4805.

Hoogervorst, R., Stramigioli, S., Wopereis, H. W., & Fumagalli, M. (2015). Vision-IMU based collaborative control of a blind UAV. In *Workshop on research, education and development of unmanned aerial systems* (pp. 53–61). IEEE, http://dx.doi.org/10.1109/RED-UAS.2015.7440990.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, *34*(3), 189–206. http://dx.doi.org/10.1007/s10514-012-9321-0.

Houben, S., Quenzel, J., Krombach, N., & Behnke, S. (2016). Efficient multi-camera visual-inertial SLAM for micro aerial vehicles. In *International conference on intelligent robots and systems* (pp. 1616–1622). IEEE, http://dx.doi.org/10.1109/IROS.2016.7759261.

Howard, A. (2006). Multi-robot simultaneous localization and mapping using particle filters. *International Journal of Robotics Research*, *25*(12), 1243–1256. http://dx.doi.org/10.1177/0278364906072250.

Hsiao, M., Westman, E., & Kaess, M. (2018). Dense planar-inertial SLAM with structural constraints. In *International conference on robotics and automation* (pp. 6521–6528). IEEE.

Hsu, Y.-H., & Gau, R.-H. (2020). Reinforcement learning-based collision avoidance and optimal trajectory planning in UAV communication networks. *IEEE Transactions on Mobile Computing*, http://dx.doi.org/10.1109/TMC.2020.3003639.

Huang, H., Savkin, A. V., & Ni, W. (2020). A method for covert video surveillance of a car or a pedestrian by an autonomous aerial drone via trajectory planning. In *International conference on control, automation and robotics* (pp. 446–449). Singapore: IEEE.

Huang, R., Tan, P., & Chen, B. M. (2015). Monocular vision-based autonomous navigation system on a toy quadcopter in unknown environments. In *International conference on unmanned aircraft systems* (pp. 1260–1269). IEEE, http://dx.doi.org/10.1109/ICUAS.2015.7152419.

Indu, & Singh, R. (2020). Trajectory planning and optimization for UAV communication: A review. *Journal of Discrete Mathematical Sciences and Cryptography*, *23*(2), 475–483. http://dx.doi.org/10.1080/09720529.2020.1728901.

Invernizzi, D., Panizza, P., Riccardi, F., Formentin, S., & Lovera, M. (2016). Data-driven attitude control law of a variable-pitch quadrotor: a comparison study. *IFAC-PapersOnLine*, *49*(17), 236–241. http://dx.doi.org/10.1016/j.ifacol.2016.09.041.

Ioannou, P., & Fidan, B. (2006). *Adaptive control tutorial*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Isidori, A. (1995). *Nonlinear control systems*. New York, NY: Springer.

Iskander, A., Elkassed, O., & El-Badawy, A. (2020). Minimum snap trajectory tracking for a quadrotor UAV using nonlinear model predictive control. In *Novel intelligent and leading emerging sciences conference* (pp. 344–349). IEEE, http://dx.doi.org/10.1109/NILES50944.2020.9257897.

Jang, Y., Oh, C., Lee, Y., & Kim, H. J. (2021). Multirobot collaborative monocular SLAM utilizing rendezvous. *Transactions on Robotics*, http://dx.doi.org/10.1109/TRO.2021.3058502.

Jardine, P. T., Givigi, S. N., & Yousefi, S. (2017a). Experimental results for autonomous model-predictive trajectory planning tuned with machine learning. In *International systems conference* (pp. 1–7). IEEE, http://dx.doi.org/10.1109/SYSCON.2017.7934801.

Jardine, P. T., Givigi, S., & Yousefi, S. (2017b). Parameter tuning for prediction-based quadcopter trajectory planning using learning automata. In *World congress, Vol. 50* (pp. 2341–2346). IFAC, http://dx.doi.org/10.1016/j.ifacol.2017.08.420.

Jayakrishnan, H. (2016). Position and attitude control of a quadrotor UAV using super twisting sliding mode. In *Conference on advances in control and optimization of dynamical systems, Vol. 49* (pp. 284–289). IFAC, http://dx.doi.org/10.1016/j.ifacol.2016.03.067.

Jiang, H., & Liang, Y. (2018). Online path planning of autonomous UAVs for bearing-only standoff multi-target following in threat environment. *IEEE Access*, *6*, 531–b544. http://dx.doi.org/10.1109/ACCESS.2018.2824849.

Jiang, T., Lin, D., & Chen, H. (2019). Composite learning control for UAVs via prescribed performance. In *International conference on signal, information and data processing* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/ICSIDP47821.2019.9173151.

Jones, E. S., & Soatto, S. (2011). Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *International Journal of Robotics Research*, *30*(4), 407–430. http://dx.doi.org/10.1177/0278364910388963.

Junge, O., Marsden, J. E., & Ober-Blöbaum, S. (2005). Discrete mechanics and optimal control. In *IFAC world congress, Vol. 38* (pp. 538–543). http://dx.doi.org/10.3182/20050703-6-CZ-1902.00745.

Kalabić, U. V., Kolmanovsky, I. V., & Gilbert, E. G. (2014). Reduced order extended command governor. *Automatica*, *50*(5), 1466–1472. http://dx.doi.org/10.1016/j.automatica.2014.03.012.

Kamel, M., Burri, M., & Siegwart, R. (2017). Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles. *IFAC-PapersOnLine*, *50*(1), 3463–3469. http://dx.doi.org/10.1016/j.ifacol.2017.08.849, IFAC World Congress.

Kaminer, I., Yakimenko, O., Dobrokhodov, V., Pascoal, A., Hovakimyan, N., Patel, V., et al. (2012). Coordinated path following for time-critical missions of multiple UAVs via $\mathcal{L}_1$ adaptive output feedback controllers. In *Guidance, navigation and control conference and exhibit* (pp. 1–34). Hilton Head, SC: AIAA, http://dx.doi.org/10.2514/6.2007-6409.

Kang, W.-S., Yun, S., Kwon, H.-O., Choi, R.-h., Son, C.-S., & Lee, D.-H. (2015). Stable path planning algorithm for avoidance of dynamic obstacles. In *Systems conference* (pp. 578–581). IEEE, http://dx.doi.org/10.1109/SYSCON.2015.7116813.

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, *30*(7), 846–894. http://dx.doi.org/10.1177/0278364911406761.

Karrer, M., Schmuck, P., & Chli, M. (2018). CVI-SLAM—Collaborative visual-inertial SLAM. *IEEE Robotics and Automation Letters*, *3*(4), 2762–2769. http://dx.doi.org/10.1109/LRA.2018.2837226.

Kaul, L., Zlot, R., & Bosse, M. (2016). Continuous-time three-dimensional mapping for micro aerial vehicles with a passively actuated rotating laser scanner. *Journal of Field Robotics*, *33*(1), 103–132. http://dx.doi.org/10.1002/rob.21614.

Kelley, C. T. (1995). *Iterative methods for linear and nonlinear equations*. Philadelphia, PA: SIAM.

Kelly, M. (2017). An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, *59*(4), 849–904. http://dx.doi.org/10.1137/16M1062569.

Keshavan, J., Gremillion, G., Alvarez-Escobar, H., & Humbert, J. S. (2015). Autonomous vision-based navigation of a quadrotor in corridor-like environments. *International Journal of Micro Air Vehicles*, *7*(2), 111–123. http://dx.doi.org/10.1260/1756-8293.7.2.111.

Khalil, H. K. (2002). *Nonlinear systems*. Princeton, NJ: Prentice Hall.

Kim, J., Gadsden, S. A., & Wilkerson, S. A. (2020). A comprehensive survey of control strategies for autonomous quadrotors. *Canadian Journal of Electrical and Computer Engineering*, *43*(1), 3–16. http://dx.doi.org/10.1109/CJECE.2019.2920938.

Klein, G., & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *International symposium on mixed and augmented reality* (pp. 225–234). IEEE, http://dx.doi.org/10.1109/ISMAR.2007.4538852.

Koch, W., Mancuso, R., West, R., & Bestavros, A. (2019). Reinforcement learning for UAV attitude control. *ACM Transactions on Cyber-Physical Systems*, *3*(2), 1–21. http://dx.doi.org/10.1145/3301273.

Koditschek, D. E. (1989). Robot planning and control via potential functions. In *The robotics review, Vol. 1* (pp. 349–367). Cambridge, MA, USA: MIT Press.

Koenig, S., & Likhachev, M. (2002). *D** lite. In *National conference on artificial intelligence, Vol. 15* (pp. 476–483). Alberta, Canada: AAAI.

Koenig, S., & Likhachev, M. (2005). Fast replanning for navigation in unknown terrain. *Transactions on Robotics*, *21*(3), 354–363. http://dx.doi.org/10.1109/TRO.2004.838026.

Kohlbrecher, S., Von Stryk, O., Meyer, J., & Klingauf, U. (2011). A flexible and scalable SLAM system with full 3D motion estimation. In *IEEE international symposium on safety, security, and rescue robotics* (pp. 155–160). IEEE, http://dx.doi.org/10.1109/SSRR.2011.6106777.

Konolige, K., Grisetti, G., Kümmerle, R., Burgard, W., Limketkai, B., & Vincent, R. (2010). Efficient sparse pose adjustment for 2D mapping. In *International conference on intelligent robots and systems* (pp. 22–29). IEEE, http://dx.doi.org/10.1109/IROS.2010.5649043.

Koo, S., Kim, S., & Suk, J. (2015). Model predictive control for UAV automatic landing on moving carrier deck with heave motion. In *IFAC workshop on multivehicle systems, Vol. 48* (pp. 59–64). http://dx.doi.org/10.1016/j.ifacol.2015.06.464.

Kotaru, P., Edmonson, R., & Sreenath, K. (2019). Geometric $\mathcal{L}_1$ adaptive attitude control for a quadrotor unmanned aerial vehicle. *Journal of Dynamic Systems, Measurement, and Control*, *142*(3), http://dx.doi.org/10.1115/1.4045558, 031003.

Krajník, T., Nitsche, M., Faigl, J., Vaněk, P., Saska, M., Přeučil, L., et al. (2014). A practical multirobot localization system. *Journal of Intelligent and Robotic Systems*, *76*(3), 539–562. http://dx.doi.org/10.1007/s10846-014-0041-x.

Kreisselmeier, G., & Steinhauser, R. (1979). Systematic control design by optimizing a vector performance index. *IFAC Proceedings Volumes*, *12*(7), 113–117. http://dx.doi.org/10.1016/S1474-6670(17)65584-8, IFAC Symposium on computer Aided Design of Control Systems.

Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., & Burgard, W. (2011). G^{2o}: A General framework for graph optimization. In *International conference on robotics and automation* (pp. 3607–3613). IEEE, http://dx.doi.org/10.1109/ICRA.2011.5979949.

Ladosz, P., Oh, H., & Chen, W.-H. (2018). Trajectory planning for communication relay unmanned aerial vehicles in urban dynamic environments. *Journal of Intelligent and Robotic Systems*, *89*(1), 7–25.

L'Afflitto, A. (2017). Differential games, continuous Lyapunov functions, and stabilisation of non-linear dynamical systems. *IET Control Theory & Applications*, *11*, 2486–2496. http://dx.doi.org/10.1049/iet-cta.2017.0271.

L'Afflitto, A., Anderson, R. B., & Mohammadi, K. (2018). An introduction to nonlinear robust control for unmanned quadrotor aircraft. *IEEE Control Systems Magazine*, *38*(3), 102–121. http://dx.doi.org/10.1109/MCS.2018.2810559.

L'Afflitto, A., Haddad, W. M., & Bakolas, E. (2016). Partial-state stabilization and optimal feedback control. *International Journal of Robust and Nonlinear Control*, *26*(5), 1026–1050. http://dx.doi.org/10.1002/rnc.3349.

L'Afflitto, A., & Mohammadi, K. (2017). Robust observer-based control of nonlinear dynamical systems with state constraints. *Journal of the Franklin Institute*, *354*(16), 7385–7409. http://dx.doi.org/10.1016/j.jfranklin.2017.09.007.

L'Afflitto, A., & Mohammadi, K. (2018). Equations of motion of rotary-wing unmanned aerial system with time-varying inertial properties. *Journal of Guidance, Control, and Dynamics*, *41*(2), 559–564. http://dx.doi.org/10.2514/1.G003015.

L'Afflitto, A., & Sultan, C. (2010). On calculus of variations in aircraft and spacecraft formation flying path planning. In *Guidance, navigation, and control conference* (pp. 1–20). AIAA, http://dx.doi.org/10.2514/6.2010-8018.

Lajoie, P.-Y., Ramtoula, B., Chang, Y., Carlone, L., & Beltrame, G. (2020). DOOR-SLAM: Distributed, online, and outlier resilient SLAM for robotic teams. *IEEE Robotics and Automation Letters*, *5*(2), 1656–1663. http://dx.doi.org/10.1109/LRA.2020.2967681.

Latombe, J.-C. (2012). *Robot motion planning, Vol. 124*. Berlin, Germany: Springer.

LaValle, S. (2006). *Planning algorithms*. Cambridge, MA: Cambridge University Press.

Le Nhu Ngoc Thanh, H., & Hong, S. K. (2018). Quadcopter robust adaptive second order sliding mode control based on PID sliding surface. *IEEE Access*, *6*, 66850–66860. http://dx.doi.org/10.1109/ACCESS.2018.2877795.

Lee, T. (2018). Geometric control of quadrotor UAVs transporting a cable-suspended rigid body. *Transactions on Control Systems Technology*, *26*(1), 255–264. http://dx.doi.org/10.1109/TCST.2017.2656060.

Lekkala, K. K., & Mittal, V. K. (2016). Accurate and augmented navigation for quadcopter based on multi-sensor fusion. In *Annual india conference* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/INDICON.2016.7838890.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., & Furgale, P. (2015). Keyframe-based visual–inertial odometry using nonlinear optimization. *International Journal of Robotics Research*, *34*(3), 314–334. http://dx.doi.org/10.1177/0278364914554813.

Li, Y., Cui, R., Li, Z., & Xu, D. (2018). Neural network approximation based near-optimal motion planning with kinodynamic constraints using RRT. *IEEE Transactions on Industrial Electronics*, *65*(11), 8718–8729. http://dx.doi.org/10.1109/TIE.2018.2816000.

Li, Z., Wang, L., Jiang, L., & Xu, C.-Z. (2019). FC-SLAM: FEderated learning enhanced distributed visual-LiDAR SLAM in cloud robotic system. In *International conference on robotics and biomimetics* (pp. 1995–2000). IEEE, http://dx.doi.org/10.1109/ROBIO49542.2019.8961798.

Liberzon, D. (2003). *Switching in systems and control*. Boston, MA: Springer.

Lin, P., Chen, S., & Liu, C. (2016). Model predictive control-based trajectory planning for quadrotors with state and input constraints. In *International conference on control, automation and systems* (pp. 1618–1623). IEEE.

Lin, C.-M., Tai, C.-F., & Chung, C.-C. (2014). Intelligent control system design for UAV using a recurrent wavelet neural network. *Neural Computing and Applications*, *24*(2), 487–496. http://dx.doi.org/10.1007/s00521-012-1242-5.

Liu, J., Jayakumar, P., Stein, J. L., & Ersal, T. (2018). A nonlinear model predictive control formulation for obstacle avoidance in high-speed autonomous ground vehicles in unstructured environments. *Vehicle System Dynamics*, *56*(6), 853–882. http://dx.doi.org/10.1080/00423114.2017.1399209.

Liu, L., Nan, D., Li, J., Weng, Y., Lian, L., Li, S., et al. (2020). Hybrid robust PID control of unknown quadrotor UAVs. In *Data driven control and learning systems conference* (pp. 959–964). IEEE, http://dx.doi.org/10.1109/ICSSE50014.2020.9219290.

Liu, S., Watterson, M., Mohta, K., Sun, K., Bhattacharya, S., Taylor, C. J., et al. (2017). Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments. *IEEE Robotics and Automation Letters*, *2*(3), 1688–1695. http://dx.doi.org/10.1109/LRA.2017.2663526.

Liu, H., Zhang, G., & Bao, H. (2016). Robust keyframe-based monocular SLAM for augmented reality. In *International symposium on mixed and augmented reality* (pp. 1–10). IEEE, http://dx.doi.org/10.1109/ISMAR.2016.24.

Loianno, G., Thomas, J., & Kumar, V. (2015). Cooperative localization and mapping of MAVs using RGB-D sensors. In *International conference on robotics and automation* (pp. 4021–4028). IEEE, http://dx.doi.org/10.1109/ICRA.2015.7139761.

Loubar, H., Zammoum Boushaki, R., Aribi, Y., & Abdellah, K. (2019). Altitude back-stepping control of quadcopter. In *International conference on applied automation and industrial diagnostics, Vol. 1* (pp. 1–11). IEEE, http://dx.doi.org/10.1109/ICAAID.2019.8934965.

Lu, Y., Xue, Z., Xia, G.-S., & Zhang, L. (2018). A survey on vision-based UAV navigation. *Geo-Spatial Information Science*, *21*(1), 21–32. http://dx.doi.org/10.1080/10095020.2017.1420509.

Lv, L., Zhang, S., Ding, D., & Wang, Y. (2019). Path planning via an improved DQN-based learning policy. *IEEE Access*, *7*, 67319–67330. http://dx.doi.org/10.1109/ACCESS.2019.2918703.

Mao, G., Drake, S., & Anderson, B. D. (2007). Design of an extended Kalman filter for UAV localization. In *Information, decision and control* (pp. 224–229). IEEE, http://dx.doi.org/10.1109/IDC.2007.374554.

Marsden, J. E., & West, M. (2001). Discrete mechanics and variational integrators. *Acta Numerica*, *10*, 357–514. http://dx.doi.org/10.1017/S096249290100006X.

Marshall, J. A., Anderson, R. B., Chien, W.-Y., Johnson, E. N., & L'Afflitto, A. (2021). A guidance system for tactical autonomous unmanned aerial vehicles. *Journal of Intelligent and Robotic Systems*, In press.

Martinelli, A. (2011). Closed-form solution for attitude and speed determination by fusing monocular vision and inertial sensor measurements. In *International conference on robotics and automation* (pp. 4538–4545). Shanghai, China: IEEE, http://dx.doi.org/10.1109/ICRA.2011.5980081.

Martins, L., Cardeira, C., & Oliveira, P. (2019). Linear quadratic regulator for trajectory tracking of a quadrotor. *IFAC-PapersOnLine*, *52*(12), 176–181. http://dx.doi.org/10.1016/j.ifacol.2019.11.195, IFAC Symposium on Automatic Control.

Marzouqi, M., & Jarvis, R. A. (2003). Covert path planning for autonomous robot navigation in known environments. In *Australasian conference on robotics and automation* (pp. 1–10). Brisbane, Australia: Australian Robotics and Automation Association.

Meng, J., Pawar, V. M., Kay, S., & Li, A. (2018). UAV path planning system based on 3D informed RRT^* for dynamic obstacle avoidance. In *IEEE international conference on robotics and biomimetics* (pp. 1653–1658). http://dx.doi.org/10.1109/ROBIO.2018.8665162.

Mohta, K., Watterson, M., Mulgaonkar, Y., Liu, S., Qu, C., Makineni, A., et al. (2018). Fast, autonomous flight in GPS-denied and cluttered environments. *Journal of Field Robotics*, *35*(1), 101–120. http://dx.doi.org/10.1002/rob.21774.

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., et al. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Aaai/Iaai* (pp. 593–598). http://dx.doi.org/10.5555/777092.777184.

Moon, J., Lee, B.-Y., & Tahk, M.-J. (2018). A hybrid dynamic window approach for collision avoidance of VTOL UAVs. *International Journal of Aeronautical and Space Sciences*, *19*(4), 889–903. http://dx.doi.org/10.1007/s42405-018-0061-z.

Mourikis, A. I., & Roumeliotis, S. I. (2007). A multi-state constraint Kalman filter for vision-aided inertial navigation. In *International conference on robotics and automation* (pp. 3565–3572). IEEE, http://dx.doi.org/10.1109/ROBOT.2007.364024.

Müller, M. A., & Allgöwer, F. (2017). Economic and distributed model predictive control: Recent developments in optimization-based control. *SICE Journal of Control, Measurement, and System Integration*, *10*(2), 39–52.

Müller, J., & Burgard, W. (2013). Efficient probabilistic localization for autonomous indoor airships using sonar, air flow, and IMU sensors. *Advanced Robotics*, *27*(9), 711–724. http://dx.doi.org/10.1080/01691864.2013.779005.

Mur-Artal, R., Montiel, J. M. M., & Tardós, J. D. (2015). ORB-SLAM: A versatile and accurate monocular SLAM system. *Transactions on Robotics*, *31*(5), 1147–1163. http://dx.doi.org/10.1109/TRO.2015.2463671.

Mur-Artal, R., & Tardos, J. D. (2017). Orb-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-d cameras. *Transactions on Robotics*, *33*(5), 1255–1262. http://dx.doi.org/10.1109/TRO.2017.2705103.

Naldi, R., Furci, M., Sanfelice, R. G., & Marconi, L. (2017). Robust global trajectory tracking for underactuated VTOL aerial vehicles using inner-outer loop control paradigms. *Transactions on Automatic Control*, *62*(1), 97–112. http://dx.doi.org/10.1109/TAC.2016.2557967.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., et al. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *International symposium on mixed and augmented reality, Vol. 11* (pp. 127–136). http://dx.doi.org/10.1109/ISMAR.2011.6092378.

Nguyen, N. P., Mung, N. X., Thanh, H. L. N. N., Huynh, T. T., Lam, N. T., & Hong, S. K. (2021). Adaptive sliding mode control for attitude and altitude system of a quadcopter UAV via neural network. *IEEE Access*, *9*, 40076–40085. http://dx.doi.org/10.1109/ACCESS.2021.3064883.

Nicotra, M. M., Naldi, R., & Garone, E. (2016). A robust explicit reference governor for constrained control of unmanned aerial vehicles. In *American control conference* (pp. 6284–6289). IEEE, http://dx.doi.org/10.1109/ACC.2016.7526657.

Niit, E. A., & Smit, W. J. (2017). Integration of model reference adaptive control (MRAC) with PX4 firmware for quadcopters. In *International conference on mechatronics and machine vision in practice* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/M2VIP.2017.8211479.

Nikhilraj, A., Simha, H., & Priyadarshan, H. (2019). Optimal energy trajectory generation for a quadrotor UAV using geometrically exact computations on $SE(3)$. *Control Systems Letters*, *3*(1), 216–221. http://dx.doi.org/10.1109/LCSYS.2018.2874103.

Noormohammadi-Asl, A., Esrafilian, O., Ahangar Arzati, M., & Taghirad, H. D. (2020). System identification and H_∞-based control of quadrotor attitude. *Mechanical Systems and Signal Processing*, *135*, Article 106358. http://dx.doi.org/10.1016/j.ymssp.2019.106358.

Noreen, I., Khan, A., Ryu, H., Doh, N. L., & Habib, Z. (2018). Optimal path planning in cluttered environment using RRT*-AB. *Intelligent Service Robotics*, *11*(1), 41–52. http://dx.doi.org/10.1007/s11370-017-0236-7.

Norouzi, M., De Bruijn, F., & Miró, J. V. (2012). Planning stable paths for urban search and rescue robots. In T. Röfer, N. M. Mayer, J. Savage, & U. Saranlı (Eds.), *RoboCup 2011: Robot soccer world cup XV* (pp. 90–101). Berlin, Heidelberg: Springer Berlin Heidelberg.

Oleynikova, H., Lanegger, C., Taylor, Z., Pantic, M., Millane, A., Siegwart, R., et al. (2020). An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments. *Journal of Field Robotics*, *37*(4), 642–666.

Oleynikova, H., Taylor, Z., Siegwart, R., & Nieto, J. (2018). Safe local exploration for replanning in cluttered unknown environments for microaerial vehicles. *Robotics and Automation Letters*, *3*(3), 1474–1481. http://dx.doi.org/10.1109/LRA.2018.2800109.

Pehlivanoglu, Y. V. (2012). A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV. *Aerospace Science and Technology*, *16*(1), 47–55. http://dx.doi.org/10.1016/j.ast.2011.02.006.

Pereida, K., Brunke, L., & Schoellig, A. P. (2021). Robust adaptive model predictive control for guaranteed fast and accurate stabilization in the presence of model errors. *International Journal of Robust and Nonlinear Control*.

Pereira, J. C., Leite, V. J. S., & Raffo, G. V. (2021). Nonlinear model predictive control on SE(3) for quadrotor aggressive maneuvers. *Journal of Intelligent & Robotic Systems*, *101*(3), 62. http://dx.doi.org/10.1007/s10846-021-01310-8.

Pestana, J., Maurer, M., Muschick, D., Hofer, M., & Fraundorfer, F. (2019). Overview obstacle maps for obstacle-aware navigation of autonomous drones. *Journal of Field Robotics*, *36*(4), 734–762. http://dx.doi.org/10.1002/rob.21863.

Peterson, B., & Narendra, K. (1982). Bounded error adaptive control. *Transactions on Automatic Control*, *27*(6), 1161–1168. http://dx.doi.org/10.1109/TAC.1982.1103112.

Primatesta, S., Guglieri, G., & Rizzo, A. (2019). A risk-aware path planning strategy for UAVs in urban environments. *Journal of Intelligent and Robotic Systems*, *95*(2), 629–643. http://dx.doi.org/10.1007/s10846-018-0924-3.

Prodan, I., Olaru, S., Bencatel, R., Borges de Sousa, J., Stoica, C., & Niculescu, S.-I. (2013). Receding horizon flight control for trajectory tracking of autonomous aerial vehicles. *Control Engineering Practice*, *21*(10), 1334–1349. http://dx.doi.org/10.1016/j.conengprac.2013.05.010.

PX4 Firmware (2021). Retrieved from https://px4.io/. (Accessed 20 September 2021).

Qian, J., Mai, X., & Yuwen, X. (2018). Real-time power line safety distance detection system based on LOAM SLAM. In *Chinese automation congress* (pp. 3204–3208). IEEE, http://dx.doi.org/10.1109/CAC.2018.8623168.

Qin, T., Li, P., & Shen, S. (2018a). Relocalization, global optimization and map merging for monocular visual-inertial SLAM. In *International conference on robotics and automation* (pp. 1197–1204). IEEE, http://dx.doi.org/10.1109/ICRA.2018.8460780.

Qin, T., Li, P., & Shen, S. (2018b). VINS-MOno: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, *34*(4), 1004–1020. http://dx.doi.org/10.1109/TRO.2018.2853729.

Qiu, W., Bi, S., Zhong, C., Luo, Y., Li, J., & Sun, B. (2017). Obstacle avoidance of aerial vehicle based on monocular vision. In *Annual international conference on CYBER technology in automation, control, and intelligent systems* (pp. 657–662). IEEE, http://dx.doi.org/10.1109/CYBER.2017.8446065.

Quan, L., Zhang, Z., Xu, C., & Gao, F. (2020). EVA-planner: Environmental adaptive quadrotor planning. arXiv preprint arXiv:2011.04246.

Radmanesh, M., Kumar, M., Guentert, P. H., & Sarim, M. (2018). Overview of path-planning and obstacle avoidance algorithms for UAVs: A comparative study. *Unmanned Systems*, *6*(2), 95–118.

Ramezani, M., Tinchev, G., Iuganov, E., & Fallon, M. (2020). Online LiDAR-SLAM for legged robots with robust registration and deep-learned loop closure. In *International conference on robotics and automation* (pp. 4158–4164). IEEE, http://dx.doi.org/10.1109/ICRA40945.2020.9196769.

Rao, A. V. (2014). Trajectory optimization: A survey. In *Optimization and optimal control in automotive systems* (pp. 3–21). Springer.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, *21*(4), 25–34. http://dx.doi.org/10.1145/37402.37406.

Richards, A. (2015). Fast model predictive control with soft constraints. *European Journal of Control*, *25*, 51–59. http://dx.doi.org/10.1016/j.ejcon.2015.05.003.

Richter, C., Bry, A., & Roy, N. (2016). Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics research* (pp. 649–666). Springer.

Rubí, B., Pérez, R., & Morcego, B. (2020). A survey of path following control strategies for UAVs focused on quadrotors. *Journal of Intelligent & Robotic Systems*, *98*(2), 241–265. http://dx.doi.org/10.1007/s10846-019-01085-z.

Samir, M., Ebrahimi, D., Assi, C., Sharafeddine, S., & Ghrayeb, A. (2020). Trajectory planning of multiple dronecells in vehicular networks: A reinforcement learning approach. *IEEE Networking Letters*, *2*(1), 14–18. http://dx.doi.org/10.1109/ICC.2018.8422706.

Santoso, F., Garratt, M. A., & Anavatti, S. G. (2016). Visual–inertial navigation systems for aerial robotics: Sensor fusion and technology. *Transactions on Automation Science and Engineering*, *14*(1), 260–275. http://dx.doi.org/10.1109/TASE.2016.2582752.

Saravanakumar, A., Kaviyarasu, A., & Ashly Jasmine, R. (2021). Sampling based path planning algorithm for UAV collision avoidance. *Sādhanā*, *46*(3), 112. http://dx.doi.org/10.1007/s12046-021-01642-z.

Sarmientoy, A., Murrieta-Cidz, R., & Hutchinsony, S. (2005). A sample-based convex cover for rapidly finding an object in a 3-d environment. In *International conference on robotics and automation* (pp. 3486–3491). IEEE, http://dx.doi.org/10.1109/ROBOT.2005.1570649.

Saska, M., Vonásek, V., Chudoba, J., Thomas, J., Loianno, G., & Kumar, V. (2016). Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles. *Journal of Intelligent and Robotic Systems*, *84*(1), 469–492. http://dx.doi.org/10.1007/s10846-016-0338-z.

Schmuck, P., & Chli, M. (2017). Multi-UAV collaborative monocular SLAM. In *International conference on robotics and automation* (pp. 3863–3870). IEEE, http://dx.doi.org/10.1109/ICRA.2017.7989445.

Schmuck, P., & Chli, M. (2019a). CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams. *Journal of Field Robotics*, *36*(4), 763–781. http://dx.doi.org/10.1002/rob.21854.

Schmuck, P., & Chli, M. (2019b). CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams. *Journal of Field Robotics*, *36*(4), 763–781. http://dx.doi.org/10.1002/rob.21854.

Schouwenaars, T., Mettler, B., Feron, E., & How, J. (2004). Hybrid model for trajectory planning of agile autonomous vehicles. *Journal of Aerospace Computing, Information, and Communication*, *1*(12), 629–651. http://dx.doi.org/10.2514/1.12931.

Schwartz, J. T., & Sharir, M. (1983). On the "piano movers" problem. II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, *4*(3), 298–351. http://dx.doi.org/10.1016/0196-8858(83)90014-3.

Sebe, N., Cohen, I., Garg, A., & Huang, T. S. (2005). *Machine learning in computer vision, Vol. 29*. Springer Science & Business Media.

Segal, A., Haehnel, D., & Thrun, S. (2009). Generalized-ICP. In *Robotics: science and systems, Vol. 2* (pp. 435–442). http://dx.doi.org/10.15607/RSS.2009.V.021.

Shan, T., & Englot, B. (2018). Lego-loam: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain. In *International conference on intelligent robots and systems* (pp. 4758–4765). IEEE, http://dx.doi.org/10.1109/IROS.2018.8594299.

Shao, Y. S., Chen, C., Kousik, S., & Vasudevan, R. (2021). Reachability-based trajectory safeguard (RTS): A safe and fast reinforcement learning safety layer for continuous control. *IEEE Robotics and Automation Letters*, *6*(2), 3663–3670. http://dx.doi.org/10.1109/LRA.2021.3063989.

Shekhar, R. C., Kearney, M., & Shames, I. (2015). Robust model predictive control of unmanned aerial vehicles using waysets. *Journal of Guidance, Control, and Dynamics*, *38*(10), 1898–1907.

Sheng, G., Min, M., Xiao, L., & Liu, S. (2018). Reinforcement learning-based control for unmanned aerial vehicles. *Journal of Communications and Information Networks*, *3*(3), 39–48. http://dx.doi.org/10.1007/s41650-018-0029-y.

Shin, Y.-S., Park, Y. S., & Kim, A. (2020). Dvl-SLAM: sparse depth enhanced direct visual-LiDAR SLAM. *Autonomous Robots*, *44*(2), 115–130. http://dx.doi.org/10.1007/s10514-019-09881-0.

Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. *Advances in Neural Information Processing Systems*, *28*, 3483–3491.

Spitzer, A., Yang, X., Yao, J., Dhawale, A., Goel, K., Dabhi, M., et al. (2018). Fast and agile vision-based flight with teleoperation and collision avoidance on a multirotor. In *International symposium on experimental robotics* (pp. 524–535). Springer, http://dx.doi.org/10.1007/978-3-030-33950-0_45.

Sujiwo, A., Ando, T., Takeuchi, E., Ninomiya, Y., & Edahiro, M. (2016). Monocular vision-based localization using ORB-SLAM with LIDAR-aided mapping in real-world robot challenge. *Journal of Robotics and Mechatronics*, *28*(4), 479–490. http://dx.doi.org/10.20965/jrm.2016.p0479.

Sun, K., Mohta, K., Pfrommer, B., Watterson, M., Liu, S., Mulgaonkar, Y., et al. (2018). Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, *3*(2), 965–972. http://dx.doi.org/10.1109/LRA.2018.2793349.

Tal, E., & Karaman, S. (2020). Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness. *IEEE Transactions on Control Systems Technology*, 1–16. http://dx.doi.org/10.1109/TCST.2020.3001117.

Tan, K.-H., & Lewis, M. (1996). Virtual structures for high-precision cooperative mobile robotic control. In *International conference on intelligent robots and systems, Vol. 1* (pp. 132–139). IEEE, http://dx.doi.org/10.1109/IROS.1996.570643.

Tang, S., & Kumar, V. (2018). Autonomous flight. *Annual Review of Control, Robotics, and Autonomous Systems*, *1*(1), 29–52. http://dx.doi.org/10.1146/annurev-control-060117-105149.

Tang, L., Wang, H., Liu, Z., & Wang, Y. (2021). A real-time quadrotor trajectory planning framework based on B-spline and nonuniform kinodynamic search. *Journal of Field Robotics*, *38*(3), 452–475. http://dx.doi.org/10.1002/rob.21997.

Tardos, J. D., Neira, J., Newman, P. M., & Leonard, J. J. (2002). Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research*, *21*(4), 311–330. http://dx.doi.org/10.1177/027836402320556340.

Tateno, K., Tombari, F., Laina, I., & Navab, N. (2017). CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. In *Conference on computer vision and pattern recognition* (pp. 6565–6574). IEEE, http://dx.doi.org/10.1109/CVPR.2017.695.

Teixeira, L., Maffra, F., Moos, M., & Chli, M. (2018). VI-RPE: Visual-inertial relative pose estimation for aerial vehicles. *IEEE Robotics and Automation Letters*, *3*(4), 2770–2777. http://dx.doi.org/10.1109/LRA.2018.2837687.

Thrun, S., & Montemerlo, M. (2006). The graph SLAM algorithm with applications to large-scale mapping of urban structures. *International Journal of Robotics Research*, *25*(5–6), 403–429. http://dx.doi.org/10.1177/0278364906065387.

Thurston, W. (1979). *The geometry and topology of three-manifolds*. Princeton, NJ: Princeton University.

Tian, Y., Liu, K., Ok, K., Tran, L., Allen, D., Roy, N., et al. (2020). Search and rescue under the forest canopy using multiple UAVs. *International Journal of Robotics Research*, *39*(10–11), 1201–1221. http://dx.doi.org/10.1177/0278364920929398.

Tordesillas, J., Lopez, B. T., & How, J. P. (2019). FASTER: Fast and safe trajectory planner for flights in unknown environments. In *International conference on intelligent robots and systems* (pp. 1934–1940). http://dx.doi.org/10.1109/IROS40897.2019.8968021.

Tripathi, A. K., Raja, R. G., & Padhi, R. (2014). Reactive collision avoidance of UAVs with stereovision camera sensors using UKF. In *International conference on advances in control and optimization of dynamical systems, Vol. 47, No. 1* (pp. 1119–1125). IFAC, http://dx.doi.org/10.3182/20140313-3-IN-3024.00171.

Van Heusden, K., Karimi, A., & Bonvin, D. (2011). Data-driven model reference control with asymptotically guaranteed stability. *International Journal of Adaptive Control and Signal Processing*, *25*(4), 331–351. http://dx.doi.org/10.1002/acs.1212.

Vásárhelyi, G., Virágh, C., Somorjai, G., Nepusz, T., Eiben, A. E., & Vicsek, T. (2018). Optimized flocking of autonomous drones in confined environments. *Science Robotics*, *3*(20), http://dx.doi.org/10.1126/scirobotics.aat3536.

Vidal-Calleja, T., Andrade-Cetto, J., & Sanfeliu, A. (2004a). Conditions for suboptimal filter stability in SLAM. In *International conference on intelligent robots and systems, Vol. 1* (pp. 27–32). IEEE, http://dx.doi.org/10.1109/IROS.2004.1389324.

Vidal-Calleja, T., Andrade-Cetto, J., & Sanfeliu, A. (2004b). Estimator stability analysis in SLAM. *IFAC Proceedings Volumes*, *37*(8), 346–351. http://dx.doi.org/10.1016/S1474-6670(17)32000-1.

Viquerat, A., Blackhall, L., Reid, A., Sukkarieh, S., & Brooker, G. (2008). Reactive collision avoidance for unmanned aerial vehicles using Doppler radar. In C. Laugier, & R. Siegwart (Eds.), *Field and service robotics: results of the 6th international conference* (pp. 245–254). Berlin, Germany: Springer, http://dx.doi.org/10.1007/978-3-540-75404-6_23.

Wang, Y., & Boyd, S. (2009). Fast model predictive control using online optimization. *Transactions on Control Systems Technology*, *18*(2), 267–278. http://dx.doi.org/10.1109/TCST.2009.2017934.

Wang, J., Chi, W., Li, C., Wang, C., & Meng, M. Q.-H. (2020). Neural RRT*: Learning-based optimal path planning. *IEEE Transactions on Automation Science and Engineering*, *17*(4), 1748–1758. http://dx.doi.org/10.1109/TASE.2020.2976560.

Wang, P., Yang, R., Cao, B., Xu, W., & Lin, Y. (2018). DeLs-3D: DEep localization and segmentation with a 3D semantic map. In *Conference on computer vision and pattern recognition* (pp. 5860–5869). IEEE, http://dx.doi.org/10.1109/CVPR.2018.00614.

Wang, Q., Zeng, Y., Zou, Y.-K., & Li, Q.-Z. (2020). A closed loop detection method for Lidar simultaneous localization and mapping with light calibration information. *Sensors and Materials*, *32*(7), 2289–2301. http://dx.doi.org/10.18494/SAM.2020.2879.

Wen, N., Zhao, L., Su, X., & Ma, P. (2015). UAV online path planning algorithm in a low altitude dangerous environment. *Journal of Automatica Sinica*, *2*(2), 173–185. http://dx.doi.org/10.1109/JAS.2015.7081657.

Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J., & Leutenegger, S. (2016). ElasticFusion: Real-time dense SLAM and light source estimation. *International Journal of Robotics Research*, *35*(14), 1697–1716. http://dx.doi.org/10.1177/0278364916669237.

Wilson, D. B., Göktoğan, A. H., & Sukkarieh, S. (2016). Vision-aided guidance and navigation for close formation flight. *Journal of Field Robotics*, *33*(5), 661–686. http://dx.doi.org/10.1002/rob.21637.

Woods, A. C., & La, H. M. (2019). A novel potential field controller for use on aerial robots. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*(4), 665–676. http://dx.doi.org/10.1109/TSMC.2017.2702701.

Wright, S. J. (2019). Efficient convex optimization for linear MPC. In S. V. Rakovic, & W. S. Levine (Eds.), *Handbook of model predictive control* (pp. 287–303). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-77489-3_13.

Wu, Z., Ni, J., Qian, W., Bu, X., & Liu, B. (2021). Composite prescribed performance control of small unmanned aerial vehicles using modified nonlinear disturbance observer. *ISA Transactions*, http://dx.doi.org/10.1016/j.isatra.2021.01.032.

Xia, W., Rangan, S., Mezzavillla, M., Lozano, A., Geraci, G., Semkin, V., et al. (2020). Generative neural network channel modeling for millimeter-wave UAV communication. arXiv:2012.09133.

Xiao, J. (2020). Trajectory planning of quadrotor using sliding mode control with extended state observer. *Measurement and Control*, *53*(7–8), 1300–1308. http://dx.doi.org/10.1177/0020294020927419.

Xie, W., Yu, G., Cabecinhas, D., Cunha, R., & Silvestre, C. (2021). Global saturated tracking control of a quadcopter with experimental validation. *IEEE Control Systems Letters*, *5*(1), 169–174. http://dx.doi.org/10.1109/LCSYS.2020.3000561.

Xiong, J.-J., & Zhang, G. (2016). Discrete-time sliding mode control for a quadrotor UAV. *Optik*, *127*(8), 3718–3722. http://dx.doi.org/10.1016/j.ijleo.2016.01.010.

Xiong, J.-J., & Zhang, G.-B. (2017). Global fast dynamic terminal sliding mode control for a quadrotor UAV. *ISA Transactions*, *66*, 233–240. http://dx.doi.org/10.1016/j.isatra.2016.09.019.

Xu, Z., Deng, D., & Shimada, K. (2021). Autonomous UAV exploration of dynamic environments via incremental sampling and probabilistic roadmap. *IEEE Robotics and Automation Letters*, *6*(2), 2729–2736. http://dx.doi.org/10.1109/LRA.2021.3062008.

Xu, Y., Liu, Z., & Wang, X. (2018). Monocular vision based autonomous landing of quadrotor through deep reinforcement learning. In *Chinese control conference* (pp. 10014–10019). IEEE, http://dx.doi.org/10.23919/ChiCC.2018.8482830.

Yadav, I., & Tanner, H. G. (2020). Reactive receding horizon planning and control for quadrotors with limited on-board sensing. In *International conference on intelligent robots and systems* (pp. 7058–7063). IEEE, http://dx.doi.org/10.1109/IROS45743.2020.9341306.

Yan, Z., Chenxiao, C., Juan, Y., Yun, Z., & Yi, Y. (2020). Design of autonomous navigation system for quadrotor in subway tunnel. In *International conference on control & automation* (pp. 187–192). IEEE, http://dx.doi.org/10.1109/ICCA51439.2020.9264455.

Yan, C., Xiang, X., & Wang, C. (2020). Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments. *Journal of Intelligent & Robotic Systems*, *98*(2), 297–309. http://dx.doi.org/10.1007/s10846-019-01073-3.

Yang, Z., Fang, Z., & Li, P. (2016). Bio-inspired collision-free 4D trajectory generation for UAVs using tau strategy. *Journal of Bionic Engineering*, *13*(1), 84–97. http://dx.doi.org/10.1016/S1672-6529(14)60162-1.

Yang, L., Qi, J., Song, D., Xiao, J., Han, J., & Xia, Y. (2016). Survey of robot 3D path planning algorithms. *Journal of Control Science and Engineering*, http://dx.doi.org/10.1155/2016/7426913.

Yousif, K., Bab-Hadiashar, A., & Hoseinnezhad, R. (2015). An overview to visual odometry and visual SLAM: Applications to mobile robotics. *Intelligent Industrial Systems*, *1*(4), 289–311. http://dx.doi.org/10.1007/s40903-015-0032-7.

Yu, H., Li, X., Murray, R. M., Ramesh, S., & Tomlin, C. J. (2018). Safe, autonomous and intelligent vehicles. Springer.

Zhang, C., & Fu, M. (1996). A revisit to the gain and phase margins of linear quadratic regulators. *Transactions on Automatic Control*, *41*(10), 1527–1530. http://dx.doi.org/10.1109/9.539438.

Zhang, J., Gu, D., Deng, C., & Wen, B. (2019). Robust and adaptive backstepping control for hexacopter UAVs. *IEEE Access*, *7*, 163502–163514. http://dx.doi.org/10.1109/ACCESS.2019.2951282.

Zhang, C., Huh, J., & Lee, D. D. (2018). Learning implicit sampling distributions for motion planning. In *International conference on intelligent robots and systems* (pp. 3654–3661). IEEE, http://dx.doi.org/10.1109/IROS.2018.8594028.

Zhang, T., Kahn, G., Levine, S., & Abbeel, P. (2016). Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search. In *International conference on robotics and automation* (pp. 528–535). IEEE, http://dx.doi.org/10.1109/ICRA.2016.7487175.

Zhang, J., & Singh, S. (2014). LOAM: Lidar odometry and mapping in real-time. *Robotics: Science and Systems*, *2*(9), 1–9. http://dx.doi.org/10.15607/RSS.2014.X.007.

Zhang, Z., Wu, J., Dai, J., & He, C. (2020). A novel real-time penetration path planning algorithm for stealth UAV in 3D complex dynamic environment. *IEEE Access*, *8*, 122757–122771. http://dx.doi.org/10.1109/ACCESS.2020.3007496.

Zhao, X., Zhang, Y., & Zhao, B. (2020). Robust path planning for avoiding obstacles using time-environment dynamic map. *Measurement and Control*, *53*(1–2), 214–221. http://dx.doi.org/10.1177/0020294019847704.

Zhou, B., Gao, F., Wang, L., Liu, C., & Shen, S. (2019a). Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, *4*(4), 3529–3536. http://dx.doi.org/10.1109/LRA.2019.2927938.

Zhou, B., Gao, F., Wang, L., Liu, C., & Shen, S. (2019b). Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, *4*(4), 3529–3536.

Zhou, B., Pan, J., Gao, F., & Shen, S. (2021). RAPTOR: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Transactions on Robotics*, 1–18. http://dx.doi.org/10.1109/TRO.2021.3071527.

Zhou, D., & Schwager, M. (2016). Assistive collision avoidance for quadrotor swarm teleoperation. In *International conference on robotics and automation* (pp. 1249–1254). IEEE, http://dx.doi.org/10.1109/ICRA.2016.7487256.

Zhou, D., Wang, Z., & Schwager, M. (2018). Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures. *IEEE Transactions on Robotics*, *34*(4), 916–923. http://dx.doi.org/10.1109/TRO.2018.2857477.

Zhou, X., Yi, Z., Liu, Y., Huang, K., & Huang, H. (2020). Survey on path and view planning for UAVs. *Virtual Reality & Intelligent Hardware*, *2*(1), 56–69. http://dx.doi.org/10.1016/j.vrih.2019.12.004.

Zou, D., Tan, P., & Yu, W. (2019). Collaborative visual SLAM for multiple agents: A brief survey. *Virtual Reality & Intelligent Hardware*, *1*(5), 461–482. http://dx.doi.org/10.1016/j.vrih.2019.09.002.