Robust and Efficient Depth-Based Obstacle Avoidance for Autonomous Miniaturized UAVs

Hanna Müller, *Student Member, IEEE*, Vlad Niculescu, *Student Member, IEEE*, Tommaso Polonelli, *Member, IEEE*, Michele Magno, *Senior Member, IEEE*, and Luca Benini, *Fellow, IEEE*

***Abstract*—Nanosize drones hold enormous potential to explore unknown and complex environments. Their small size makes them agile and safe for operation close to humans and allows them to navigate through narrow spaces. However, their tiny size and payload restrict the possibilities for onboard computation and sensing, making fully autonomous flight extremely challenging. The first step toward full autonomy is reliable obstacle avoidance, which has proven to be challenging by itself in a generic indoor environment. Current approaches utilize vision-based or 1-D sensors to support nanodrone perception algorithms. This article presents a lightweight obstacle avoidance system based on a novel millimeter form factor 64 pixels multizone time-of-flight (ToF) sensor and a generalized model-free control policy. In-field tests are based on the Crazyflie 2.1, extended by a custom multizone ToF deck, featuring a total flight mass of 35 g. The algorithm only uses 0.3% of the onboard processing power (210 μs execution time) with a frame rate of 15 f/s. The presented autonomous nanosize drone reaches 100% reliability at 0.5 m/s in a generic and previously unexplored indoor environment.**

***Index Terms*—Autonomous navigation, nanodrones, obstacle avoidance, perception, time-of-flight (ToF) array, unmanned aerial vehicles (UAV).**

I. Introduction

UNMANNED aerial vehicles (UAVs) are nowadays used for monitoring, inspection, surveillance, transportation, logistics, and many other fields [1]. In several scenarios, mainly related to indoor applications, a small form factor brings advantages—smaller drones are more agile and can traverse complex environments ranging from cluttered offices to industrial facilities, allowing safe operation close to humans in locations otherwise inaccessible [2], [3]. For indoor use cases, standard size quadrotors weighting more than one kilograms and featuring large propellers can potentially generate hazardous situations while working nearby inexperienced users, hence they are not acceptable in crowded or indoor scenarios. Nano-UAVs [4] that weigh a few tens of grams mostly rely on off-board computation due to highly restricted onboard capabilities, typically milliwatt-power microcontrollers (MCU), strongly limited by power and size constraints [3]. MCUs are not powerful enough to run state-of-the-art solutions such as complex navigation models and simultaneous localization and mapping (SLAM) [5], [6], [7]. On the other hand, relying only on onboard sensing and computation brings many advantages—higher reliability when wireless links fail (or are jammed), lower latency in control actions, and reduced bandwidth requirements if external control is limited to high-level commands. Until now, autonomous exploration with the same agility and safety as an expert human pilot has been confined to low speed due to the lack of compact integrated low-power sensors, and resource-constrained navigation strategies [2], [5].

The major challenge for nano-UAVs is achieving autonomous navigation through a reliable and universal obstacle avoidance policy and trajectory planning in real-world applications [8]. To enable onboard decisions based on the nano-UAV surroundings, the processing should use only a minor fraction, i.e., 10%, of the overall energy envelope. For instance, in nano-UAV platforms like the Crazyflie 2.1, where the total power is around 10 W, including the motors, the maximum processing power needs to be in the orders of hundreds of megawatt to not substantially affect the flying time [9]. This power budget is compatible with a simple MCU, such as a general-purpose ARM Cortex-M4 core, commonly used on nano-UAVs, featuring a clock speed of just a few hundred megahertz and just $\sim$ 200 kB of RAM.

The motivation to enable local and lightweight navigation policies on highly constrained platforms pushes the research to explore alternative solutions that are not a direct downscale of their bigger and more powerful counterparts [10], reconsidering the whole perception pipeline, from the sensor to the navigation strategy [11], [12]. Moreover, the onboard pipeline has to be robust against disturbances, such as sensor noise, illumination conditions, and motion blur [6]. Obstacle avoidance is commonly vision-based, exploiting deep neural network (DNN) approaches to extract the navigation context from the scene [6], [7], [13], [14], but also solutions with laser range finders or even radars exist [15], [16], [17], [18]. On nano-UAVs, approaches with cameras, mono or stereo, have been investigated [13], [19]. However, they suffer from a fundamental drawback of a high number of input pixels, and therefore, a high computational load, which

Manuscript received 26 April 2023; accepted 17 August 2023. Date of publication 5 October 2023; date of current version 6 December 2023. This work was supported in part by Politecnico di Torino outgoing mobility program, in part by EDISU international mobility grant, and in part by *armasuisse Science & Technology*. This paper was recommended for publication by Associate Editor G. Huang and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. *(Corresponding author: Hanna Müller.)*

The authors are with the Department of Information Technology and Electrical Engineering, ETH Züurich, 8092 Zurich, Switzerland (e-mail: hanmuell@iis.ee.ethz.ch; vladn@iis.ee.ethz.ch; topolonelli@ethz.ch; michele.magno@pbl.ee.ethz.ch; lbenini@iis.ee.ethz.ch).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2023.3315710.

Digital Object Identifier 10.1109/TRO.2023.3315710

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

can absorb a significant fraction of the computational capabilities of an MCU, even a powerful multicore one [20]. Moreover, they are dataset-dependent, meaning that in general, the onboard network needs to be retrained if the operational environment changes [13], [21], [22]. This effect is further exacerbated by the limited amount of onboard memory, pushing researchers to optimize the model size at the cost of decreased generalization—the model can cover only a limited range of environments or conditions.

This work focuses on infrastructure less and autonomous exploration, where the nano-UAV can move through an unknown environment without a local/remote infrastructure for supporting positioning or distributed perception. Existing algorithms often rely on global map planning with known obstacles or are not suitable for position-blind contexts [23]. Moreover, they offer only a limited set of local decisions, like the *wall following* approach in the bug algorithm literature that generally exploits single point laser-beam sensors [10]. Nano-UAVs are up to now lacking the sensing and computational capabilities to build fine-grained internal maps that enable path-planning-based obstacle avoidance algorithms as potential fields, dynamic windows, or elastic bands. Our approach can be seen as a simplified version of the dynamic window approach, where we only have the current observation as information about our environment—the advantage is low memory requirements and decreased latency. Single-beam rangers have the drawback of only returning a 1-D measurement, making it impossible to acquire a depth map with a single sensor at a high frequency. Placing multiple sensors or using previously available multizone sensors is too power-hungry and heavy for operation on nanodrones. Engineering a more complex policy than wall following can be challenging, requiring tailored heuristics to find dataset-dependent architectures [6], [12], motivating the researchers to implement model-free methods [11], which have the potential to ease the engineering process, providing also a robust solution.

This work proposes a complete depth-based perception system optimized for lightweight and low-power flying robots and targeted to support obstacle avoidance and enable autonomous indoor navigation of nano-UAVs. It is tailored to cover a wide range of different application scenarios, mainly targeting indoor use cases, providing robust obstacle avoidance in unknown and previously unexplored areas. The system is additionally evaluated in outdoor environments to demonstrate the flexibility and robustness of the perception system. Our solution exploits a novel commercial multizone ranger to automatically extracts complex obstacle geometry from the background. The decision policy, together with the preprocessing, is a model-free solution; it replaces sophisticated frameworks targeted at high-end platforms with an algorithm that fits commercial MCU specs. We employed the VL53LC5CX, an 8x8 or 4x4 pixel time-of-flight (ToF) sensor from STMicroelectronics, which can generate a depth map and a pixel validity matrix at zero-computational cost and a frame rate up to 60 f/s. We empirically characterized the VL53LC5CX for its whole 45° field of view (FoV) in ranges between 20 cm and 4 m. The pixel validity indicator simplifies filtering outliers or out-of-range measurements, resulting in highly computationally efficient navigation for a low latency response.

As a reference platform, we selected the Crazyflie 2.1 from Bitcraze, on which a 2.49-g custom expansion board was designed to support two VL53LC5CX in opposite directions, front- and back-facing w.r.t the flying direction. The obstacle avoidance policy, developed on the reported dataset, consists of a decision tree fed by the depth 8×8 matrix. It runs in real time, with a frame processing time of just 210 μs, on the Crazyflie MCU, an ARM Cortex-M4f, using a mere 0.31% of its computational capacity. The depth information, prefiltered by removing invalid pixels, is categorized into four zones to control the 3-D spatial movements of the nano-UAV and the flying speed.

Although our perception solution requires only a fraction (9.4%) of the total power budget, results show best-in-class performance, with virtually zero crash rate flying at 0.5 m/s, the possibility to avoid moving obstacles at up to 2 m/s and to randomly explore complex unknown environments characterized by narrow corridors (65 cm) and thin reflective objects with standard ambient light and in complete darkness. Our lightweight model-free perception system successfully demonstrates its potential in real-world experiments characterized by outdoor and indoor environments other than controlled mazes.

On average, the speed of 1 m/s appears to be the best balance between crashing probability and flight distance, respectively, below 20% and 100 m in a variety of complex real case studies. The main scientific contributions of this work are listed as follows.

1) We designed a lightweight (2.49 g) perception board for nano-UAVs with up to two VL53LC5CX sensors. With the Crazyflie 2.1, it can be used as a plug and play expansion board. It can extract depth information without the support of standard vision-based frameworks and complex model-based approaches. Moreover, we empirically characterized the ToF sensor in nano-UAV flying conditions, demonstrating the possibility of using this solution to extract precise and reliable depth information from the scene.
2) We collected a dataset containing 43 records showing time-synchronized data from a gray-scale CMOS camera, one depth matrix extracted from the VL53LC5CX, the internal Crazyflie state, and an absolute 3-D position measured by a mocap system.
3) We developed a lightweight obstacle avoidance and random exploration algorithm that can be executed in real time on a resource-constrained microcontroller. It interactively reacts to complex obstacle geometries, commanding the escape maneuver to the internal state controller. Our algorithm is based on a model-free decision tree that groups objects at different distances and locations, which is extendable by adding path planning and mapping capabilities.
4) We carried out real-world assessment and performance evaluation in multiple operating conditions, such as mazes, indoor/outdoor environments, narrow passages, and moving obstacles. Evaluation metrics are

based on the maximum flight velocity, cumulative and individual success rate, flying time, distance, perception artifacts, and latency, and finally, the flight trajectory.

The whole project, together with the hardware, the dataset, and the obstacle avoidance policy, is released open-source.[1]

II. Related Work

UAVs are massively adopted in real-life scenarios, from civilian applications [24], such as surveillance, transportation, environmental and industrial monitoring, agriculture services, and first aid, to military services [25]. In particular, indoor navigation enables smart buildings and drone–machine or drone–human interaction, opening new research frontiers [26]. In this area, nanodrones have great potential [8], which is why nowadays there are numerous research projects aiming to address open challenges for enabling autonomous nano-UAV navigation, mapping, automation, distributed computing, and swarm formations [1], [3], [8], [24].

State-of-the-art exploration solutions that proved to work well on conventional drones, such as SLAM, are still too resource-demanding for nano-UAVs [13], [30]. While other works proved that mapping is also possible with nanodrones [27], their approach relies on off-board computing, which introduces the need of having a computer in the loop, limited range, and communication overhead.

Perceiving the 2-D/3-D structure of the environment is vital for the functionality of UAVs or robotic systems in general [6], [7], as it enables path planning and autonomous navigation through mapping and obstacle avoidance [31]. High-end UAV platforms extract the 3-D environmental information relying on complex specialized neural networks. By sensing the environment and interacting with other agents [31], they learn to generate a depth map estimating distances to objects using a variety of active sensors like monocular and stereo cameras [6], [32], asynchronous event cameras [33], structured light [34], lidar [35], and ToF sensors [11], sampling the scene at a fixed scan rate.

Loquercio et al. [6], [7] demonstrated the possibility to fly at high speed in complex environments, such as forests, exploiting a stereo camera and a neural network trained only on synthetic datasets. To remove the context bias from the simulator environment, and then, train the algorithms to work on a generalized scenarios, the authors do not directly process RGB images, but a depth map is extracted from the Intel RealSense 435 stereo pairs. Using the depth matrix not only for obstacle avoidance but also for mapping stages, they achieved a maximum fly speed of 10 m/s in real scenarios, in which the drones featured a success rate above 50–100% below 8 —in various and unknown environments. The authors present a state-of-the-art approach to enable autonomous navigation based on depth estimation in indoor and outdoor environments. However, their methodology cannot be applied on nano-UAV platforms as it demands high memory (i.e., gigabytes) and computational requirements. Furthermore, their approach also relies on high-resolution sensing, which is an uncrossable technological barrier for nano-UAVs [3], leaving *de facto* the nano-UAV autonomous exploration still an open challenge.

Due to recent technological advancements, miniature depth sensors are becoming a reality, being already incorporated in commercial devices such as smartphones or top-range quadrotors. The SONY DepthSense IMX556PLR back-illuminated ToF image sensor[2] features a resolution of 640 × 480 pixels with up to 8.3-m working distance, while the TeraRanger Evo 64px[3] proposed a compact 64 pixel and 12 g (and additionally a 3-g backboard) solution for robotic applications. Also, the commercial depth camera pmd flexx2[4] features a 640 × 480 pixel resolution for a total weight of 13 g. The aforementioned commercial sensors feature depth extraction and filtering, directly providing a pixel-by-pixel confidence flag on the sensor, moving relevant computation effort from the computing core. However, they are still not compatible with nano-UAV platforms due to their weight, e.g., every gram costs agility and the Crazyflie's total maximum payload is 15 g or incompatible digital interfaces with a commercial MCU, e.g., Camera Serial Interface targeted for high-end processors. Moreover, an input resolution of 640 × 480 pixels requires a minimum of 300 kB for a single frame, exceeding the memory available in most of the commercial MCUs, and at the same time, significantly increasing the processing time on resource-constrained platforms, like the STM32F405 employed in this article. Indeed, processing a 640 × 480 depth image requires $4.8 \cdot 10^3\times$ more computational effort than processing the 64-pixel matrix generated from the VL53LC5CX. The TeraRanger Evo 64px also features a low pixel count, however its high weight undermines the flight time and the agility. Moreover, its limited FoV of only 15° is not suitable for avoiding dynamic obstacles. Niculescu et al. [13] demonstrated the necessity to support at least 10 f/s to navigate at a forward speed of 1 m/s, thus pinpointing a maximum processing latency requirement that directly influences the sensor data size. Nevertheless, there is a clear research trend in this area, which aims to replace the traditional control framework by using an optimized solution able to sense and extract the depth map with a single system-on-chip (SoC) commercial component, simplifying the mapping and planning processing latency. Moreover, this increases the system's robustness against unexplored flying scenarios, in which depth-based approaches generalize better to new environments than learning-based algorithms only based on RGB cameras [13], [35].

Table I presents a comparison between the most recent SoA works on perception-based navigation with nanodrones. Niculescu et al. [13] present an automatic deployment flow of a convolutional neural network (CNN) that runs onboard a nanodrone and enables autonomous navigation and obstacle avoidance capabilities. The CNN used in [13] can reliably detect the presence of an obstacle and reduce the drone's forward velocity, or it can adjust the drone's heading when following a

[1] [Online]. Available: https://github.com/ETH-PBL/Matrix_ToF_Drones

[2] [Online]. Available: https://www.sony-depthsensing.com

[3] [Online]. Available: https://www.terabee.com

[4] [Online]. Available: https://3d.pmdtec.com/en/3d-cameras/flexx2

TABLE I
COMPARISON AGAINST SOA WORKS ON PERCEPTION-BASED NAVIGATION

Reference work	Model-free	Vision-based	Fully on-board	Moving obstacles	Field evaluation	Maximum speed [m/s]	Maximum flight time [s]	Dataset release	Code available	UAV weight
Our work	✓	×	✓	✓	✓	2	443	✓	✓	35 g
[13]	×	✓	✓	✓	✓	0.5	216	✓	✓	33 g
[11]	×	×	✓	×	✓	N/S	N/S	✓	✓	35.7 g
[28]	×	×	✓	×	✓	1.0	200	×	×	31.7 g
[22]	✓	✓	✓	×	✓	2.6*	N/S	×	×	72 g
[29]	✓	✓	✓	×	✓	0.6	540	×	×	20 g
[27]	✓	×	×	×	✓	0.4	N/S	×	×	N/S**

Results not reported explicitly with numerical analysis are flagged as not specified (N/S). Note *, in [22], the maximum speed is achieved in a controlled environment, where the average speed is 2 m/s. Note **, in [27], the total weight is not provided but it is estimated to be ≈35 g. It should be mentioned that the maximum flight time depends on multiple factors, including batteries, test setup, and type of vehicles used. This parameter is a coarse indication of the success of the methods but should not be used as a primary factor for comparison.

lane. However, despite its effectiveness with static and dynamic obstacles, the general performance seems poorer in unfamiliar environments, showing an almost certain probability of collision by flying with a velocity above 0.6 m/s in the presence of 90° turns [13], a condition not covered in the training dataset. Furthermore, it is often unable to steer around an unknown obstacle to avoid a collision, where the average flight time before crashing is around 1 min, especially in narrow corridors, where this time decreases down to 19 s [13]. In contrast to our approach, where all algorithms run in a single SoC (i.e., STM32), their system takes advantage of an additional multicore SoC, which is in charge of running the CNN and transmitting the inference result to the main MCU, increasing the total mass by 4.4 g.

McGuire et al. [11] introduce a swarm gradient bug algorithm (SGBA) for enabling autonomous exploration relying on an array of four ToF sensors. However, the drones can only follow walls and do not perform any localization or surrounding detection. The proposed scenario is to find "victims" in an office environment. Even so, the video data are stored on SD cards and has to be read off-board after the flying mission, which introduces a delay in obtaining the information about the environment.

Similarly, [28] relies on four ToF sensors and a light sensor and presents an approach based on deep reinforcement learning that enables a nanodrone to seek and find a light source while avoiding obstacles. However, they do not provide any results on dealing with dynamic obstacles, and the maximum speed they report during the testing phase is 1 m/s, which is significantly slower than our system.

Li et al. [22] propose a vision-based system for drone racing, whose goal is to detect "gates" and fly through them as fast as possible. While effective in passing through gates, their system is tuned to work with a particular type of obstacle and does not deal with general objects in the trajectory. Although their algorithms run entirely onboard, they use a power-hungry SoC (i.e., Cortex A7 plus a dual-core GPU), resulting in a drone that weighs twice as much as our solution.

In [29], stereo vision-based obstacle avoidance is proposed on a flapping wing UAV, reaching interesting results (15–30 Hz onboard) in the direction of common approaches applied on standard-size UAVs. However, the stereo camera board benefit of a higher resolution comes with the drawback of requiring an additional microcontroller (STM32F405) fully dedicated to image processing; instead, our approach occupies a mere 0.31% of the same processor computing budget, which we also use for the flight controller. In [29], the sensor board alone requires 484 mW. Consequently our solution does not only save energy while processing the data but also requires less weight for the additional hardware—2.49 g, compared to 4 g required in [29]. More important, even if the setup is tested in real environments, authors reported only successful flights, not providing any statistics on the mission success rate. Another concern discussed by the authors themselves is the limited robustness in nonideal conditions during the flight; for example, lighting, textureless, or small obstacles. Again, the effectiveness of current camera systems is also questioned, citing textually Tijmons et al. [29] "*The robustness of the method cannot be guaranteed when the camera system cannot produce reliable estimates. The limitations of the vision system might improve in the future as camera technology progresses.*".

Finally, [27] demonstrates the capabilities of creating a 2-D map of the environment, relying on a particle filter that fuses information from 12 ToF sensors. They create a custom deck to accommodate the 12 sensors, which they use in combination with a nanodrone, but the whole computation necessary to run their algorithms is carried off-board.

III. BACKGROUND AND HARDWARE PLATFORM

This work presents a complete system description of an obstacle avoidance system for nano-UAVs, from the hardware design to in-field evaluations. In this application scenario, design optimization and weight minimization are essential to enable longer flight times. We used the commercial Crazyflie 2.1 platform from Bitcraze, extending its functionality with a custom expansion board and new sensors, such as the VL53L5CX from STMicroelectronics. All used components are commercially available, and our design is released as open-source.

A. Crazyflie

The Crazyflie 2.1, henceforth Crazyflie, is an open software/hardware nano-UAV commonly used in research. It comes with a base board featuring an inertial measurement unit (IMU), a barometer, radio communication (using an nRF51822

from Nordic Semiconductors), and as the main processor, an STM32F405 (168 MHz, 196 kB RAM), responsible for sensor readout, state estimation, and real-time control. One important feature of the Crazyflie is its extension headers—there is a wide variety of commercially available decks to plug onto the base board to sense the environment and improve the state estimation or even plan where to fly. We use a downward-facing *Flow-deck v2*, featuring an optical flow sensor and a 1-D ToF sensor to improve the position estimation computed by the extended Kalman filter (eKF) . To collect the dataset, we also connected an *AI-deck*, featuring a QVGA grayscale camera and WiFi to stream the images to a local computer. It features a Himax HM01B0, an ultralow-power QVGA (320 × 240) grayscale camera with a 115° diagonal FoV and a NINA-W102 WiFi module from U-Blox. In our application, the 8+1 core RISC-V MCU does not directly communicate with the STM32F405, processing and compressing in a parallel task the acquired frame, which is then sent to a local gateway through a WiFi link, and then timestamped, at the arrival together with the Crazyflie state transmitted over Bluetooth (nRF51822). The base version of the Crazyflie weighs 27 g and can fly up to 7 min with its - battery; adding the *Flow-deck v2* adds 1.6 g and the *AI-deck* another 4.4 g. The maximum payload that still enables take-off is 15 g. However, the maneuverability and flight time are very poor when flying with the maximum payload [9]. To increase the flight time with multiple connected decks, we use a 350-mAh battery instead of the 250 mAh one that comes with the commercial drone, which features a 30-C current rate to support high motor current peaks but adds 1.1 g of extra-payload. In total, a maximum of 7.9 g are available for further decks.

B. ToF Multizone Sensor

The VL53L5CX[5] is designed for a wide range of ambient lighting conditions, and it is based on a vertical cavity surface emitting diode (VCSEL), a single-photon avalanche diode array, physical infrared filters, and diffractive optical elements. The novel feature of the VL53L5CX is the multizone capability; it can provide a matrix of either 8 × 8 or 4 x 4 pixels configurable by software. Each zone provides a distance measurement, and in case of ToF misscalculation or interference at 940-nm lightwave, an error flag is reported. This way, noise and errors can be filtered out through a validity matrix overlapped with the measurement matrix. From 2 cm to 2 m, the ranging accuracy is characterized by STMicroelectronics as an absolute value (±15 mm); above 2 m, the overall ranging accuracy degrades up to 11% of the absolute distance,[5] with a working range of up to 4 m.

The VL53L5CX can be configured in different ranging modes, with varying integration times, resolutions, ranging frequencies, and sharpener values. There are two ranging modes: continuous ranging and autonomous ranging. in continuous ranging, the VCSEL is always ON, and therefore, the integration time is maximized, while in the autonomous mode, the integration time can be configured, saving energy by turning OFF the VCSEL when not used. Two different resolutions are available, either 4x4 pixel or 8x8 pixels. The maximal ranging frequency is dependent on the resolution; for 4x4 pixels, 60 Hz can be reached; and for 8x8 pixels, the limit is 15 Hz.

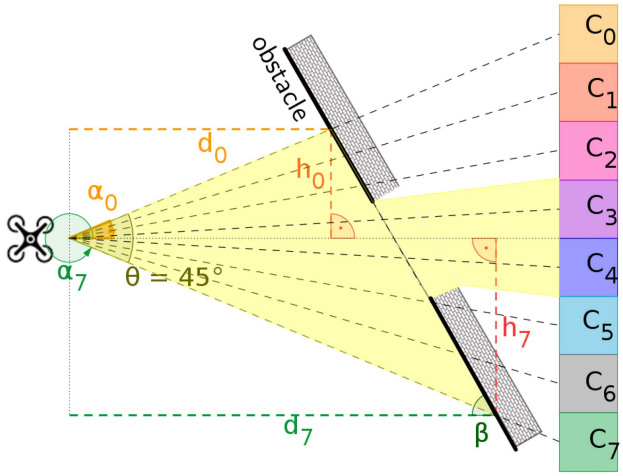


Fig. 1. Drone faces an obstacle with a gap (e.g., a door) with an angle β. C_x is the corresponding column associated with the 8x8 matrix, while d_x is the projects planar distance. The term h_x is calculated using the ToF sensor FoV and the measured d_x.

As the returned signal from a target does not have sharp edges, the sharpener value can be configured to remove some of the signal caused by veiling glare.[5] The FoV depends on the environment (target distance, reflectance, and ambient light level) and the sensor configuration (resolution, ranging mode, integration time, and sharpener). To ensure proper functionality, the cover window opening has to be at least as wide as the exclusion zone[6] (61° vertically and 55.5° horizontally). However, the detection volume[5] is narrower than the exclusion zone; it is reduced to around 45°. Fig. 1 visualizes the functionality, showing the drone facing an angled (β) wall with a gap (e.g., a door). The multizone ToF sensor measures d_x, but as angle α_x is known from the FoV, h_x can be computed. To interface the VL53L5CX with a commercial MCU, a standard 400-kHz I2C digital bus is required, along with two GPIOs. Both are available on the STM32F405. The power supply spans between 2.8 and 3.3 V, thus making it compatible with most of the MCUs and the open-source nano-UAV platforms available in the market.

C. Multizone Ranger Deck

To support in-field tests and complex flight paths, exploiting the full Crazyflie performances, we designed a custom deck specifically optimized for the VL53L5CX ToF sensor. Our new multizone ranger deck, shown in Fig. 2, can be used at the same time as the *AI-deck* and the *Flow-deck v2*. The multizone ranger deck features two mounting positions for a VL53L5CX sensor, one in the front and one in the back, enabling the possibility to detect obstacles in the front or the back. For this article, we investigate the flying performances using only the forward

[5][Online]. Available: https://www.st.com/en/imaging-and-photonics-solutions/vl53l5cx.html

[6]The exclusion zone is a region used for design integration defined by the manufacturer, and it is based on the sensor FoV tolerances.

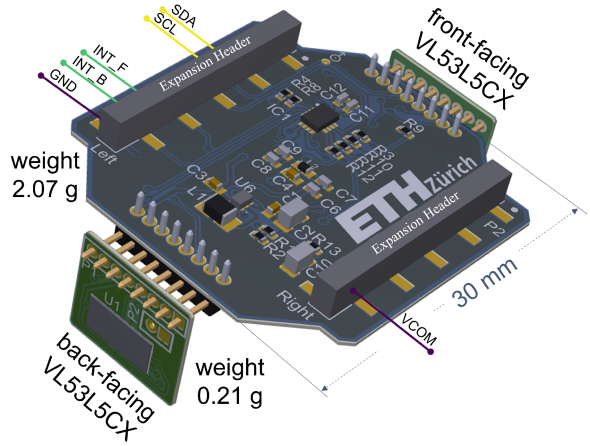


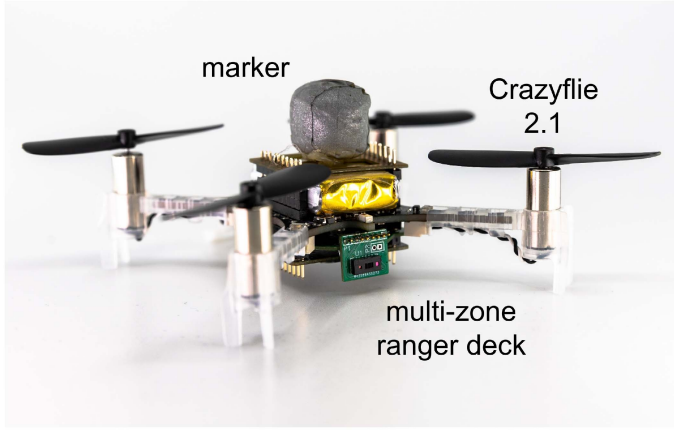
Fig. 2. Open source multizone ToF deck compatible with the Crazyflie 2.1. A forward and a backward facing VL53L5CX can be mounted vertically to a base board. The maximum weight is 2.49 g with a size of 9 cm^2.

VL53L5CX, as shown in Fig. 3(a). Each sensor requires 286 mW in continuous acquisition mode; thus, for providing a stable and low-noise - power source, we use the TPS62233 step-down switching voltage regulator with the battery voltage as input. The TCA6408 A - I2C GPIO expander manages the reset and power-down pins to decrease the amount of used line on the Crazyflie connector and to ensure compatibility with the *AI-deck* and the *Flow-deck v2*. Independent interrupt lines are used for each sensor to decrease the frame acquisition latency. The final design, in Fig. 2, has a total size of 29.4 mm × 30 mm × 9.5 mm, and in our configuration, a weight of only 2.07 g. Mounting the back-facing sensor board would add 0.21 g. As proposed in Fig. 3(a), the final flying setup used in this work uses a multizone ranger deck, a *Flow-deck v2*, and a battery holder with a reference marker for performance analysis. The payload is 4.8 g, whereas for the dataset collection, the *AI-deck* adds an extra 4.4 g. In general, we always mounted the multizone ranger deck below the Crazyflie frame [see Fig. 3(a)] and the *AI-deck* above the battery [see Fig. 3(b)].

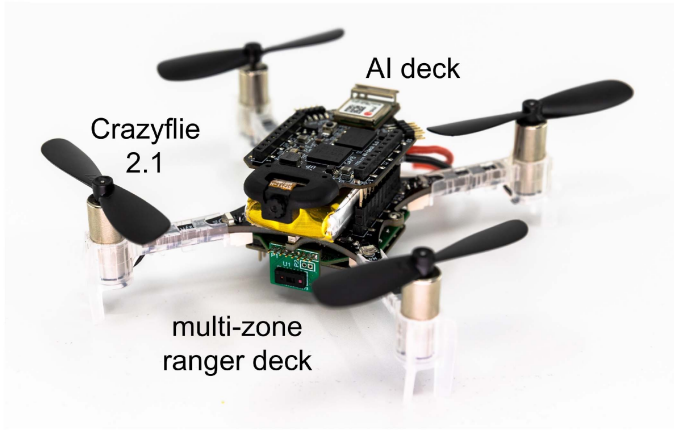
The hardware design, as well as the bill of materials, are released open-source on GitHub.[1]

IV. Characterization and Calibration

This section provides an empirical evaluation of the sensor to assess its effectiveness in measuring the distance to various objects in different lighting and flying conditions. Throughout this evaluation, the sensor is mounted at the height of 1 m from the ground on static support (i.e., a tripod). The whole setup is positioned such that the sensor is parallel to a wall, and the whole area covered in the sensor's FOV is flat. Using this setup, we sweep our sensor within the range of 0.2–3 m from the wall while maintaining its orientation. The movement is performed with a step of 0.2 m, and at each step, we acquire 1000 distance frames from the sensor using the 8x8 configuration. In addition to the distance matrix, we also store the measurement validity matrix provided by the sensor, which reports which entries in



(a)



(b)

Fig. 3. Our hardware setup for data collection and in-field testing. (a) Hardware setup featuring a flow deck and our custom multizone ToF deck on the bottom of the Crazyflie, combined with a battery holder and a Vicon marker on top. (b) For the dataset collection, the AI-deck was added instead of the battery holder. In this configuration, an extra 4.4-g payload is added.

the distance matrix are trustworthy. We repeat this acquisition procedure for the following four configurations:

1) white wall, ambient indoor light (i.e., ∼ 500 lx);
2) white wall, darkness (i.e., < 10 lx);
3) black wall, ambient indoor light;
4) black wall, darkness.

The data stored for these scenarios represent the foundation of our characterization since it covers most of the materials present in our office rooms. It is worth mentioning that large reflective or absorbent surfaces, e.g., large glass windows, limit sensor accuracy and obstacle observability. However, a complete sensor characterization would require a separate and specific study due to the large variety of materials available in everyday indoor scenarios. In this article, field experiments also include reflective and absorbent surfaces, showing the system's strengths/limitations in such adverse conditions.

First, we evaluate the error of the distance measurements in terms of mean and standard deviation. Fig. 4 shows these metrics for the case of a white background with normal ambient light

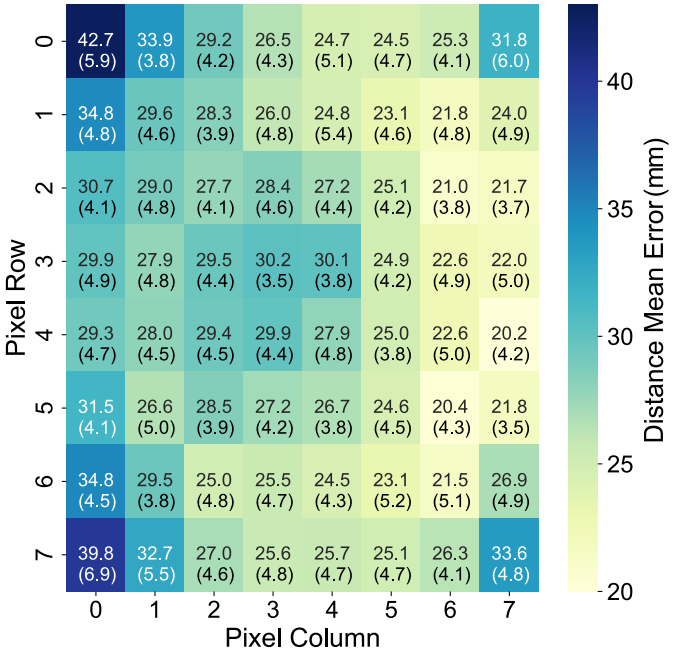


Fig. 4. VL53L5CX pixel-by-pixel characterization at 1m. Values are in millimeter. Each pixel includes the offset, on the top, and the variance, bottom, computed over 1000 successive samples in a fixed position.

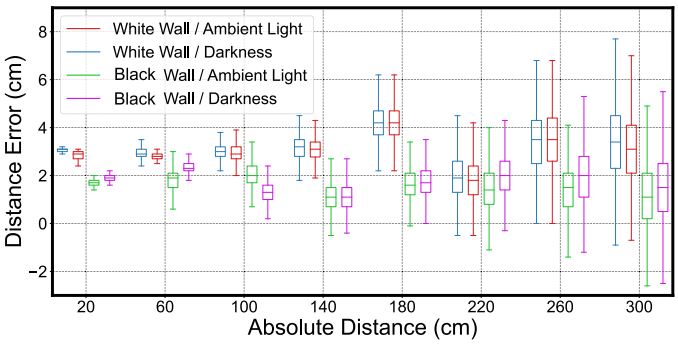


Fig. 5. Distance measurement error as a function of the absolute distance. The evaluation is performed for an absolute distance in the range 20–300 cm with a step of 40 cm for each of the four (two with an absorbing black wall and two with a more reflective white wall) considered scenarios.

when the sensor is positioned 1 m away from the wall. The error statistics are computed over 1000 samples for each individual pixel in the matrix. We note that the highest mean errors in the corners, being about 1–2 cm higher than errors associated with the rest of the pixels. The mean error takes values in the range of 19–42 mm. The standard deviation of the distance error takes values in the range of 3.4–7.3 mm, while the highest values are again encountered in the corners.

Second, we extend the previous investigation [36] to analyze the distance error and invalid pixel probability statistics in all four configurations introduced at the beginning of this section. Fig. 5 shows the distance error as a function of the distance to the wall for each scenario, considering only one pixel of the matrix (i.e., one of the four inner pixels). We highlight a pairwise similarity (i.e., median error < 0.5 cm) between the white wall

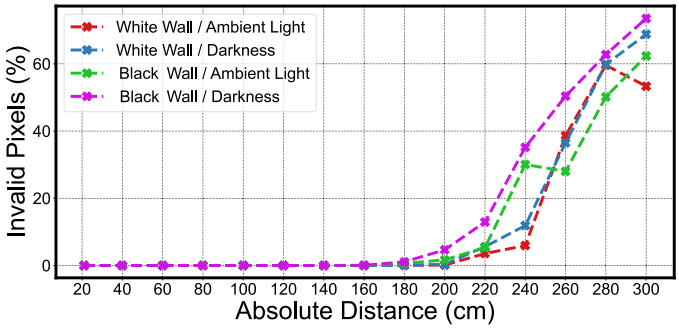


Fig. 6. Pixel validity as a function the absolute distance. The evaluation is performed for an absolute distance in the range 20–300 cm with a step of 20 cm. In all the four considered scenarios, the pixel validity decreases when the absolute distance increases.

scenarios. Furthermore, the same pattern applies to the brown wall scenarios, which in terms of median error, seem to lead to better results than the white wall case. However, this difference seems to be a constant offset, while the min–max range and interquartile difference are about the same for each scenario at a given distance. The min–max range spans up to 1 cm for an absolute distance of 20 cm and up to 8 cm for an absolute distance of 3 m.

Finally, we evaluate how the sensor measurement validity decreases with the absolute distance. The validity depends on the amount of reflected light but also external disturbances, such as ambient lighting. Fig. 6 shows the measurement validity curves for each of the four considered scenarios. We point out that the validity is higher than 95% in all scenarios, given an absolute distance to the wall smaller than 2 m and higher than 50% given an absolute distance of 2.6 m. Overall, the measurement validity is higher for the scenarios with a white wall due to a higher surface reflectivity.

To conclude the sensor characterization, we can claim that the sensor does not require any calibration phase, as its accuracy is very good within the operating range that we target (i.e., a few meters). Furthermore, we also observed that the reliability is high for absolute distances of up to 2 m, which is sufficient for enabling obstacle avoidance on nanodrones.

V. Dataset

After static tests and the VL53L5CX empirical assessment, a dynamic dataset was collected in different configurations while maneuvering in indoor environments. Tests were performed in controlled and open spaces, with the support of a motion capture system (mocap) *Vicon Vero 2.2*[7] at a rate of 50 Hz. A human pilot manually steered the Crazyflie. Initially, the dataset was used to develop and test the obstacle avoidance algorithm presented in Section VI. However, other researchers can also use it to improve our system by integrating the multizone ToF data with processed information from a CNN and the grayscale camera [13] or by applying a more general DNN algorithm to enhance the onboard intelligence [37]. For this reason, we release the acquired data

[7][Online]. Available: https://www.vicon.com/hardware/cameras/vero/

as open source,[1] We collected the following data in a time-series format with a millisecond accuracy:

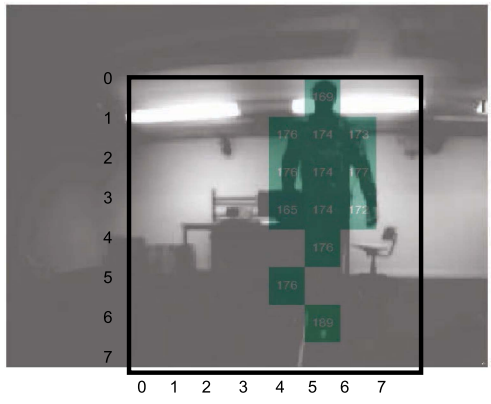
1) internal state estimation (attitude, velocity, and position) of the Crazyflie;
2) multizone ToF array in 8 × 8 pixel configuration;
3) camera images (QVGA grayscale);
4) Vicon data (attitude, position).

The dataset consists of three main groups: object approach moving the drone on a single axis, yaw rotations around the Z-axis, and a general-purpose set of flying tests approaching various obstacles and narrow holes. The first dataset group, named *linear movements*, consists of ten recordings of flights with 1)–4) data, approaching a wood panel at different speeds and stopping and flying back always on the same axis, rotations, and altitude variations are disabled. The total test time is 216 s with an average of 22 s per acquisition. The next group, *yaw rotations*, consists of three recordings with 1)–4) data, rotating on a single axis (yaw) at 1 m from an obstacle. Recorded data reach a total of 94 s. The third and final group, named *obstacle avoidance* is composed of 30 recordings with a mixed combination of 1)–4)—14 acquisitions and 1)–3)—16 acquisitions. In total, for the third group, 17 min of flight maneuvers are present in the GitHub[1] repository, with an average of 35 s per acquisition.

For each of the 43 released records, a pair of a.csv and.dat file format are present for 1), 2), and 4), whereas, for 3), a series of.jpg files are present, named with the acquisition frame time in milliseconds. To combine images and decode the time-series files, we also provide a Python script named "Flight_visualizer.py," which generates a 3-D visualization of the drone attitude and spatial position from the internal state estimator and the Vicon system. Moreover, images and the 8x8 ToF matrix are time-aligned and plotted together with the drone state. The script offers the possibility to test the control algorithm on the collected data. We provide an example in *object_detection* and *decision_making* functions that can be used as a reference point for future work. Figs. 7 and 8 are, respectively, two representative examples from *O16* and *O4* recordings,[1] reporting the grayscale image and the depth matrix. In Fig. 7, the drone is hovering in a fixed position, $V_x, V_y, V_z \approx 0$ and $(\text{yaw}, \text{pitch}, \text{roll}) \approx (0, 0, 0)$, at 1 m from the ground, while a person is walking perpendicularly to the VL53L5CX FoV. In Fig. 7(b), the 8x8 depth matrix shows the foreground distance from the nanodrone, which is reported within a centimeter precision.

Thanks to the sensor's ability to automatically detect invalid pixels, the background (out of range) is automatically subtracted, and the moving object, the foreground, is then extracted from the scene at zero-computational cost. Despite Fig. 7(a) supporting the reader in understanding the test setup and the 8 × 8 matrix, one can already notice that the HM01B0 is saturating due to the ambient lighting. This condition could decrease the integrity of algorithms fully based on vision-based sensing. Note that the legs of the person are right at the pixel border, leading to only one pixel for them in column 4, and the person is stepping forward, leading to the right knee (pixel 5/5) being out of range. On the other hand, Fig. 8 gives an example of a real flight controlled

(a)



(b)

Fig. 7. Still scene from the *O16* recording.[1] In this test, the UAV took off and hovered, and a person walked several times perpendicularly to the drone FoV. The two figures shows the identical Crazyflie status from different perspective, respectively, from the left to the right. (a) Grayscale camera from the *AI-Deck*. (b) Preprocessed 8 × 8 depth map from the multizone ToF sensor, scene equivalent to (a). Note that the Himax HM01B0 and the VL53L5CX have a different FoV and different mounting position on the Crazyflie 2.1 frame; see Fig. 3(b) for reference. The camera image is displayed as well for visualizing the FoV. (a) Grayscale QVGA image captured by the *AI-Deck*. A person is walking at approximately 1.7 m from the nanodrone. Note that the Himax HM01B0 camera is saturating due to the indoor lighting. (b) Preprocessed 8 × 8 depth map from the front-facing multizone ToF sensor, where invalid pixels are filtered and not shown. Each pixel features a distance expressed in centimeter and a colormap to help the reader in visualizing the depth of the front facing obstacle. The camera image is displayed as well for visualizing the FoV.

by a human pilot, in which the multizone ToF sensor does not correctly extract the object shape. The pilot took off, and then, swerved by 180° from the starting position and is approaching the chair at 0.35 m/s. Despite in Fig. 8(a), an office chair is correctly visible (note that pixels in row 7 belong to the ground), the depth map in Fig. 8(b) does not fully extract the foreground detail. Indeed, the chair sitting and backrest are identified and measured to be at 83 cm, but the metallic support between the two is completely invisible to the multizone ToF sensor, which then wrongly identifies a possible safe passage between rows 1 and 2. In this scenario, the chromed and thin metallic support reflects the majority of the 940-nm laser beam, being visible only from certain angles or at very short distances, i.e., below 50 cm. This peculiar behavior motivates the technical choice to

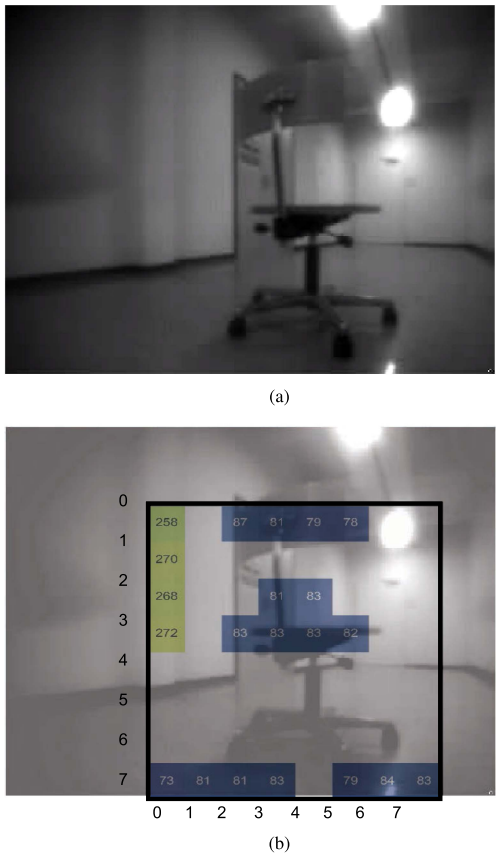


Fig. 8. Still scene from the *O4* recording.[1] In this test, the UAV took off starting to approach an office chair placed in a large room, an human operator supervised and controlled the whole maneuver. In this specifically example, we show that at this distance from the obstacle, the multizone ToF sensor correctly identifies the chair sitting and the backrest, but the metallic support between the two is not fully visible. The two figures show the identical Crazyflie status from different perspective, respectively from the left to the right. (a) Grayscale camera from the *AI-Deck*. (b) Preprocessed 8×8 depth map from the multizone ToF sensor, scene equivalent to (a). Note that the Himax HM01B0 and the VL53L5CX have a different FoV and different mounting position on the Crazyflie frame; see Fig. 3(b) for reference. The camera image is displayed as well for visualizing the FoV. (a) Grayscale QVGA image captured by the *AI-Deck*. A chair is placed at approximately 55 cm from the nano-drone. (b) Pre-processed 8×8 depth map from the front-facing multi-zone ToF sensor, where invalid pixels are filtered and not shown. Each pixel features a distance expressed in centimeter and a colormap to help the reader in visualizing the depth of the front facing obstacle. The camera image is displayed as well for visualizing the FoV.

use an image segmentation approach instead of a pixel-granular cost function to avoid obstacles.

VI. Low-Latency Lightweight Obstacle Avoidance

This section describes the whole pipeline used to implement the obstacle avoidance onboard the Crazyflie. In Fig. 9, we show how the proposed algorithm is integrated with the existing open-source Crazyflie firmware. The blocks in green belong to the base Crazyflie firmware and are used without modification, while the blocks in red represent our contribution and implement the obstacle avoidance algorithm. The base firmware performs state estimation relying on the information from the onboard IMU and the two sensors found on the *Flow-deck v2:* the downward facing

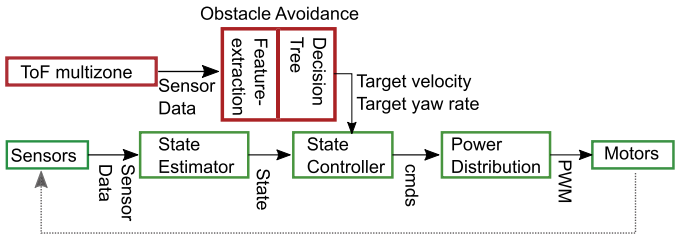


Fig. 9. Integration of the obstacle avoidance algorithm into the Crazyflie control flow. Additions are shown in red, and the default modules in green.

1-D ToF sensor used for height estimation and the optical flow sensor used for horizontal velocity estimation. The sensor data are fed into the eKF implemented in the base firmware, which produces the state estimate—position, velocity, and attitude. The "obstacle avoidance" block exploits the information from the multizone ToF sensor, producing a forward target velocity and a steering rate that enable the drone to avoid collision with the frontal obstacles. These commands are sent to the onboard controller implemented on the base firmware, which actuates the drone accordingly. In our obstacle avoidance pipeline, we first perform feature extraction to identify the objects in the ToF frame, and then, use a decision tree to determine the forward velocity and steering rate.

A. Feature Extraction

After a sensor frame is obtained, the system applies a preprocessing step before running the decision tree. First, we threshold the ToF frame, removing all pixels with an associated distance higher than 2 m—during the sensor characterization, we discovered that measurement validity decreases below 90% for higher distances. The outcome of the thresholding step is an occupancy frame (i.e., a binary frame), which indicates the presence of the obstacles for each pixel. In the following, the neighboring pixels are grouped in clusters that define the objects—with this approach, the overlapping objects within the FoV are treated as one object. We define this procedure as *grouping*, and the steps of the feature extraction algorithm are presented as pseudocode in Fig. 1. The algorithm adds all pixels that belong to the same object to *groups*. To mitigate the effect of noise/outliers, groups have a minimum number of 2 pixels, and single pixels (i.e., without any neighbor) are ignored.

Our algorithm first initializes all pixels as unvisited, then passes through them one by one to check if they belong to a group. The grouping starts with an unvisited pixel that belongs to an obstacle, then recursively adds all neighbors with a positive occupancy status. As we call the *GroupAddDFS* function at most once per pixel (resulting in a depth-first-search), the algorithm runs in $\mathcal{O}(n)$, where n is the number of pixels. The number of groups is in practice limited to 4. Several metrics characterize each group, which are as follows:

1) minimal and maximal *X/Y* coordinates (borders);
2) number of pixels;
3) position (averaged position of all pixels belonging to the group);
4) minimum distance to the object.

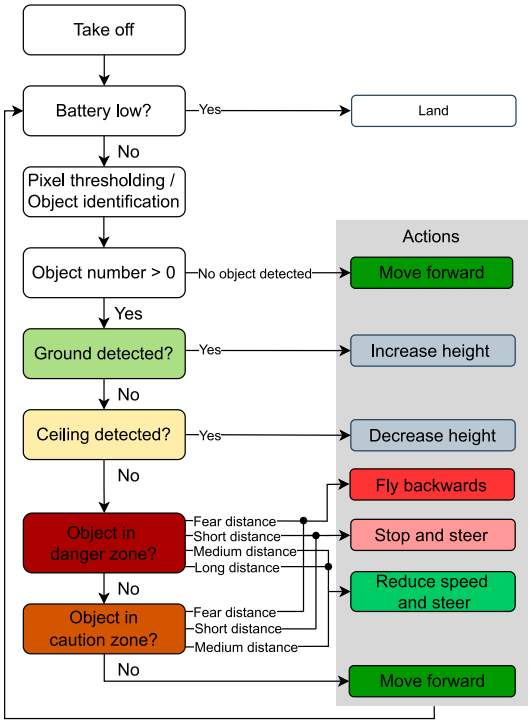


Fig. 10. Obstacle avoidance flowchart, illustrating the feature extraction blocks as well as the decision tree that provides the control commands for the drone.

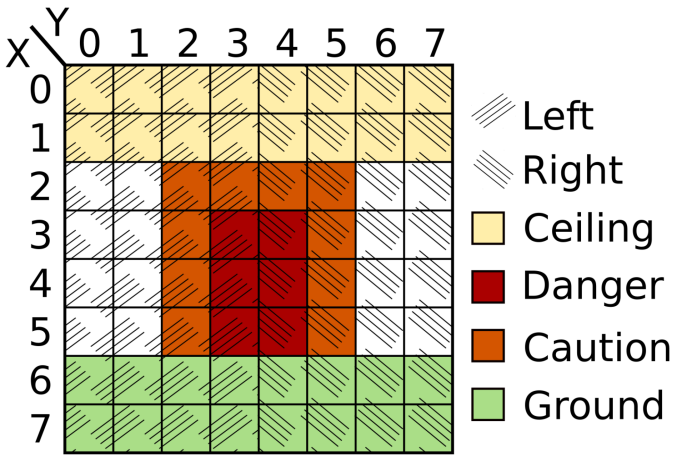


Fig. 11. Multizone ToF sensor is configured to measure 8x8 pixels, covering an FoV of around 64° (diagonally). We define four zones, a ceiling and ground zone to not fly too close to those, as well as a danger and caution zone in the center of the FoV.

B. Decision Tree

Fig. 10 illustrates the flow diagram that describes how our system interprets the ToF information and generates the flying commands. This flow runs in a continuous loop and is designed to have low latency and low complexity as it runs using only 520 800 cycles while providing accurate commands. To ensure safety, the system constantly checks the battery level, and if the battery is low, the drone lands. The "pixel thresholding / object identification" are related to the feature extraction presented in Section VI-A. Suppose the system identifies at least one object

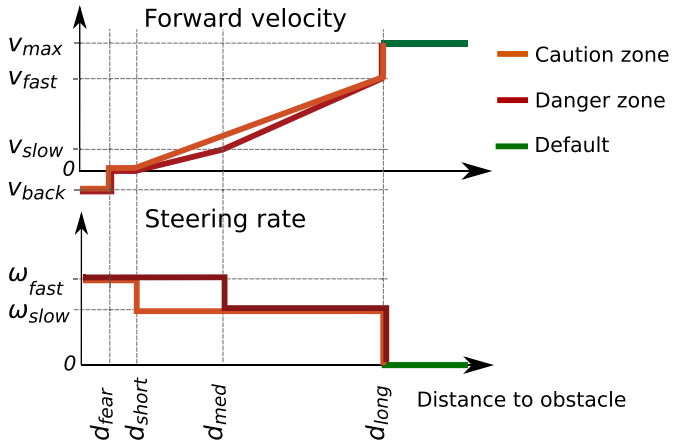


Fig. 12. Curves of the commanded forward velocity and steering rate for the caution, danger, and default zones.

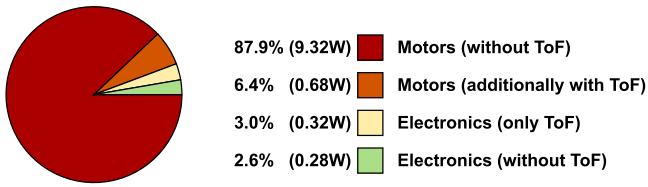


Fig. 13. Power breakdown to compare the power consumption with and without our ToF deck. The ToF deck contributes only 9.4% to the power consumption.

within the FoV during the feature extraction phase. In that case, the decision tree is employed as a collision avoidance algorithm to decide what flying command to apply. The decision tree provides the flying commands (i.e., steering rate and forward velocity) based on the distance to the object and the zone where the obstacle is found within the FoV.

The distance is split in five intervals determined by four thresholds: d_{fear}, d_{short}, d_{med}, and d_{long}, which take the values 0.15, 0.4, 0.7, and 1.4 m, respectively (determined empirically). The zones are defined by dividing the FoV into four zones (i.e., ground, ceiling, caution, and danger) and two sides (i.e., left and right), as shown in Fig. 11. Given that we target to mainly explore indoor environments (e.g., corridors and offices), we assume that the floor and the ceiling are mostly flat. Therefore, the drone is commanded to fly at a fixed height (i.e., 0.4 m) from the floor. While the cruising height is 0.4 m, the system continuously checks if the closest object is in the ceiling/ground zone, and if this is the case, it adjusts the height accordingly so that it keeps distance from the object.

If there is no obstacle in either of these two zones, the algorithm checks for the presence of the obstacle in the danger and caution zones and reacts according to the distance to the object. For instance, if the distance to the object is smaller than d_{fear}, the drone flies backward to avoid being very close to the object—we further motivate in Section VII that being close to a wall/object decreases the accuracy of the drone's state estimation. Moreover, if the distance to the object is in the interval $(d_{\text{fear}}, d_{\text{short}})$, the drone completely stops and steers until it determines that it is safe to fly in the forward direction regardless of the object's

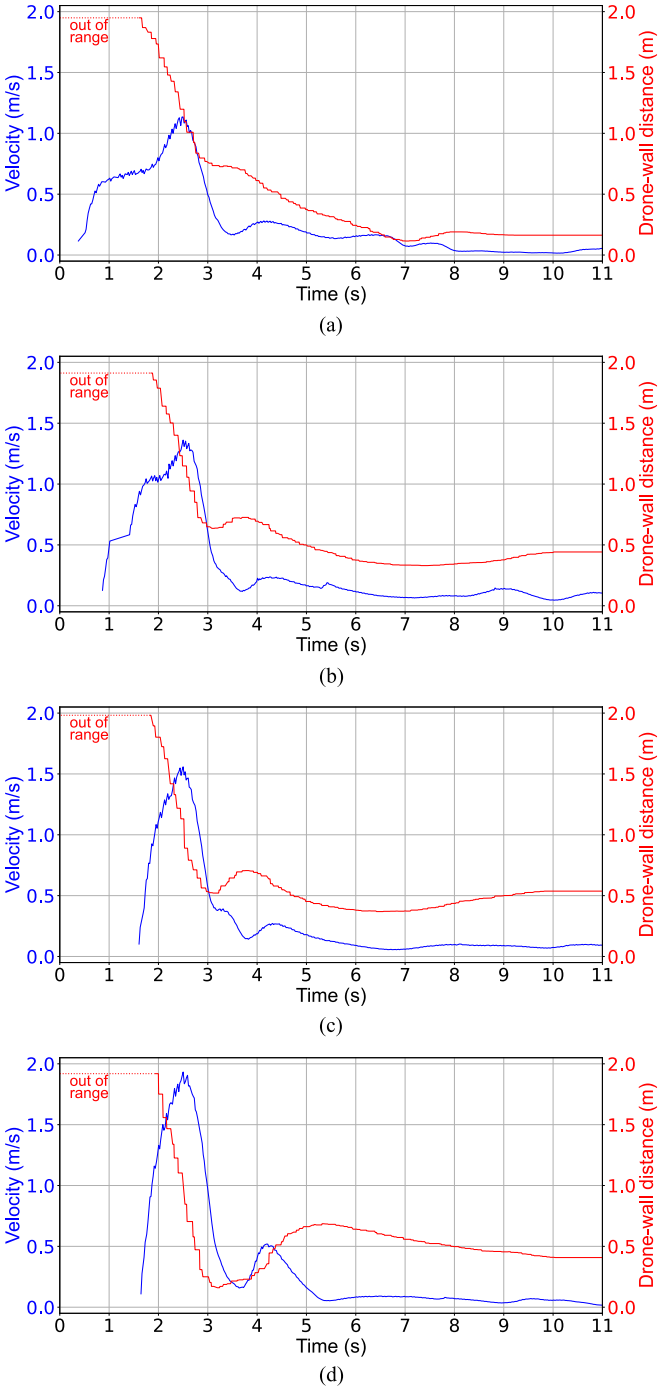


Fig. 14. Braking in front of a wall with different $v_{\max}$. In red, we show the distance from the wall, measured by the onboard ToF sensor. In blue, we show the velocity, recorded by the mocap system. (a) $v_{\max} = 1$ m/s. (b) $v_{\max} = 1.5$ m/s. (c) $v_{\max} = 2$ m/s. (d) $v_{\max} = 2.5$ m/s.

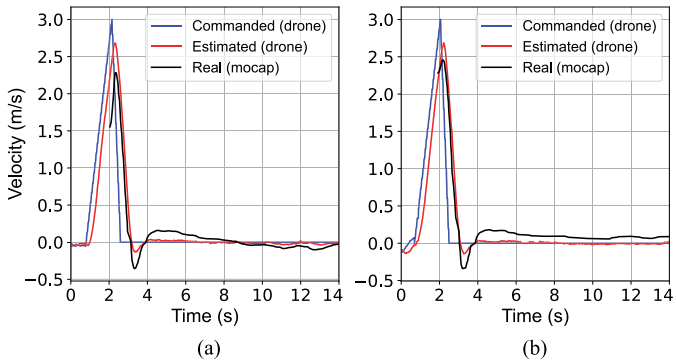


Fig. 15. Comparing real, commanded, and estimated forward velocities while braking in open space and in front of a wall. (a) Braking in open space. (b) Braking in front of a wall.

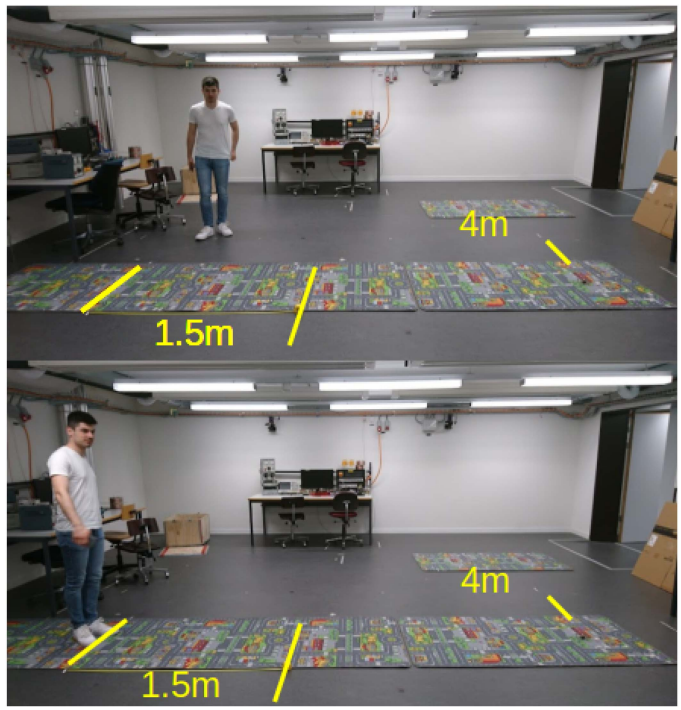


Fig. 16. Our test setup for testing dynamic braking. A person jumps in front of the drone once it sees it at 1.5-m distance.

shape. Finally, if the distance is higher than d_{med}, the drone does not have to stop completely, but it slows down and steers while flying.

However, the actions presented in Fig. 10 are rather simplified because the values of the velocity and steering rate depend on both zone and distance to the object. Fig. 12 presents the velocity and steering rate curves for the danger, caution, and default (i.e., no obstacle) zones. The v_{back}, v_{slow}, and v_{fast} take values of -0.2, 0.15, and 0.85 m/s, respectively. $v_{\max}$ represents the forward velocity when no obstacle is detected and is a configurable parameter. One can note that the forward velocity varies linearly with the distance to the object in the caution and danger zones when the distance takes values within (d_{short}, d_{long}). However, in the danger zone, the velocity slope is slightly different; taking two possible values—we determined empirically that this improves the system robustness. The steering rate can take the value of either $\omega_{\text{slow}} = 0.7\,\text{rad/s}$ or $\omega_{\text{slow}} = 1\,\text{rad/s}$ as shown in Fig. 12. In addition to what is shown in Fig. 10, our system also checks for dead ends. When the drone is stuck in front of a blocked path, it would typically start oscillating left and right, trying to find the way out. If the nano-UAV perceives close obstacles symmetrically (e.g., a flat surface straight ahead), it

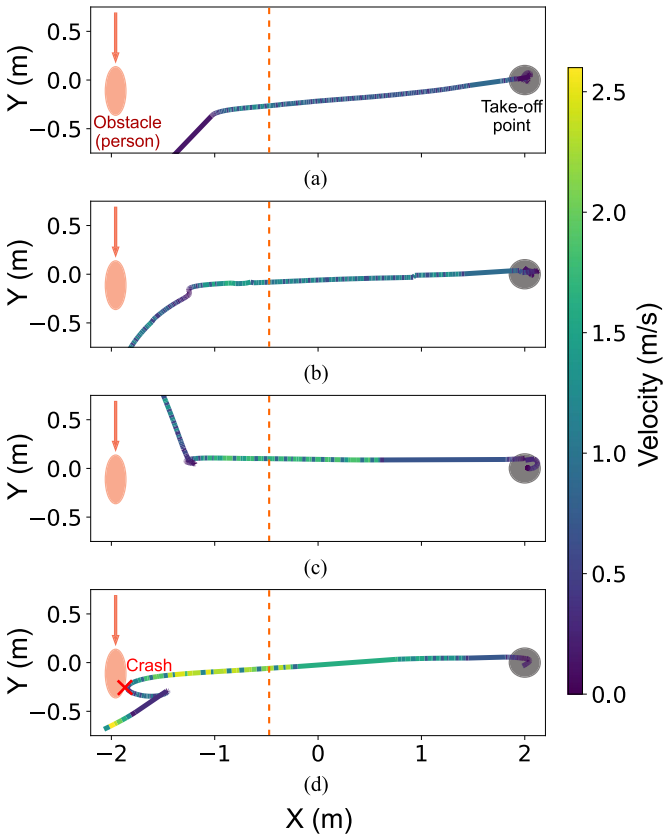


Fig. 17. Brake test in front of dynamic obstacles at different velocities. A person jumps in front of the drone once it sees it at 1.5-m distance. The red arrow shows the direction the person comes from and the red ellipse where it jumps to. The drone takes off in the gray circle and is headed in the negative X-direction. The drone altitude is fixed in this experiment. (a) $v_{\max} = 1$ m/s. (b) $v_{\max} = 1.5$ m/s. (c) $v_{\max} = 2$ m/s. (d) $v_{\max} = 2.5$ m/s.

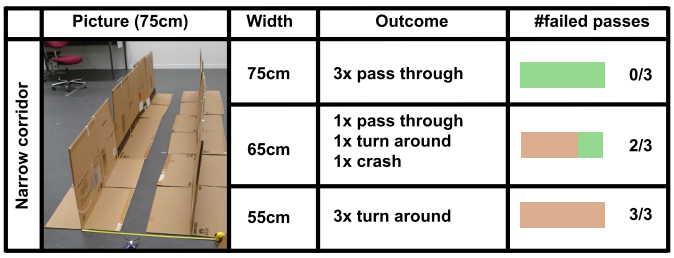


Fig. 18. Flying through a narrow corridor characterized by different widths.

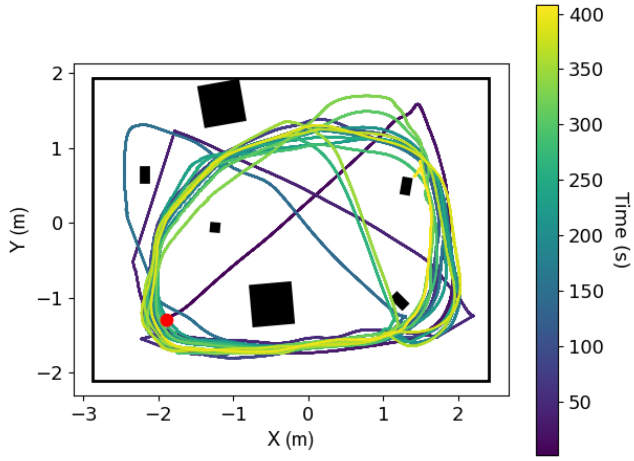


Fig. 19. Flying for 6 min and 45 s in an environment with static obstacles. The color map indicates the time course, while the data are logged with the Vicon system. The drone flies at a target height of 0.5 m, with a maximum acceleration of 1.5 $\mathrm{m/s^2}$. The flight path converges to the same loop because the algorithm is deterministic.

Listing 1: This algorithm clusters all the occupied pixels (i.e., distance < 2 m) into individual groups. *pixel.occupied* is a binary variable and indicates the occupancy status, while *pixel.visited* indicates if the algorithm already passed through the pixel. *group_index* refers to the number of a group, and after executing the *Grouping*, the value of *group_index* corresponds to the number of groups/clusters.

```
# for a given pixel, GroupAddDFS finds all
↪  connected, occupied, and unvisited pixels
def GroupAddDFS(pixel, group_index):
    for neighbor of pixel:
        if neighbor.occupied == true and
        ↪  neighbor.visited == false:
            neighbor.visited = true
            GroupAddDFS(neighbor, group_index)
            group[group_index].add(neighbor)
# finds all pixel groups/clusters
def Grouping(binary_frame):
    group_index=0
    for pixel in binary_frame:
        pixel.visited = false
    for pixel in binary_frame:
        if pixel.occupied == true:
            if pixel.visited == false:
                pixel.visited == true
                GroupAddDFS(pixel, group_index)
                if group[group_index] is not empty:
                    group[group_index].add(pixel)
                    group_index++
```

will turn right as the default action. Additionally, to mitigate the issue of getting stuck in a corner, a history mechanism is implemented to check for exceeding a threshold of repeated oscillations between left and right commands, which will steer the drone 180° to escape.

VII. Results

This section provides a power and computational requirements analysis of our approach. Furthermore, it presents an evaluation of our system, demonstrating the obstacle avoidance and exploration capabilities in real-world experiments. We evaluate the system's functionality with both static and dynamic obstacles in various environments.

A. Computational Load and Power Consumption

Our algorithm (displayed in red in Fig. 9) takes 35 k cycles to process one frame on average, at the maximum rate of the ToF multizone sensor (15 Hz). This means we add a mere 0.31% load to the STM32F405 on the Crazyflie. The latency from the acquired ToF image data to the flight command is 210 μs on average. The Crazyflie control flow (displayed in green in

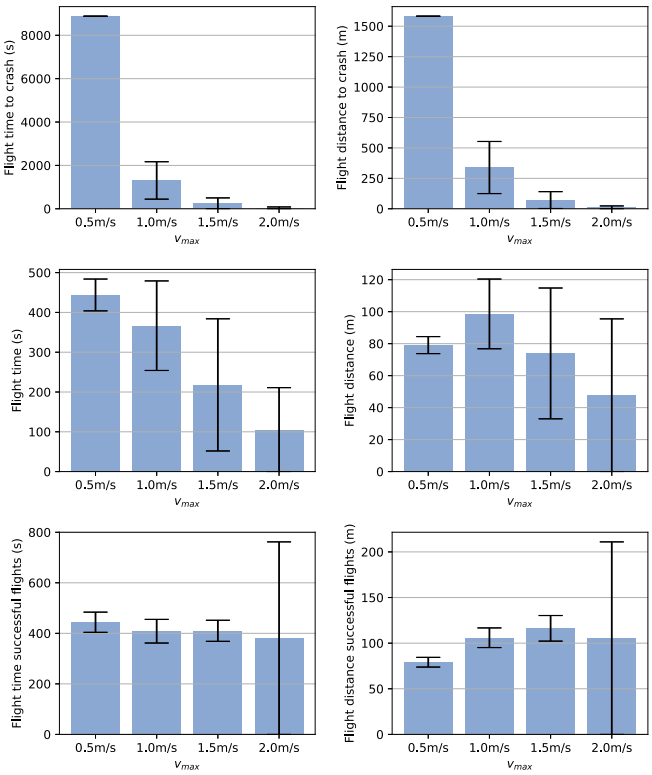


Fig. 20. In the top row, we show the average time and distance until we experience a crash - not counting battery changes besides the end of the test to display this metric also when we have 100% reliability. In the middle row, the average flight time and distance are shown. In the bottom row, we also display the average flight time and distance but only consider successful flights. The blue bar represents the average, and the black line the variance.

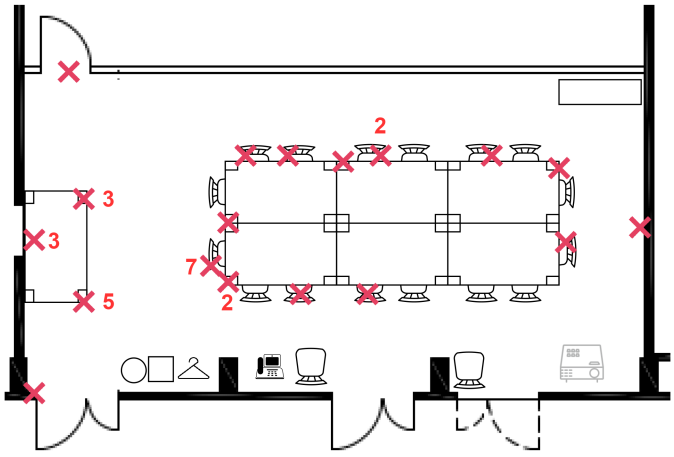


Fig. 21. Floorplan of the meeting room in which we ran the reliability test. Red crosses show the recorded crashes.

Fig. 9) with a flow-deck and configured to use an eKF for state estimation needs 35% of the computational capabilities of the STM32F405, meaning we only add a minimal additional load.

The increased power consumption from software load is, hence, negligible. However, the sensor typically consumes 286 mW—if we account for the voltage regulator's efficiency, it leads to a power consumption of 320 mW. Power consumption

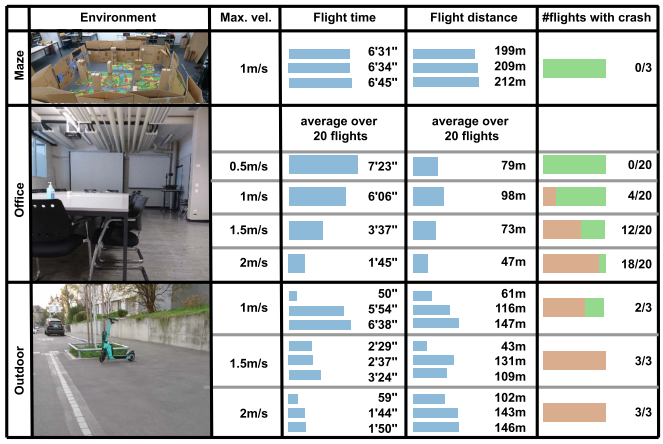


Fig. 22. Real-world assessment of the proposed perception system in a controlled environment *Maze*, in a general, office room *Office* and in *Outdoor*. Results are reported for each velocity, comparing the flight time, the traveled distance, and the reliability, expressed in the number of tests that include at least one crash.

also increases because of additional weight: we add 7.8 g (1.7-g flow deck, 2.3-g custom deck, 1.1-g heavier battery, 1.3-g battery holder incl. reflective marker, and 1.4-g long pin headers), leading to a 35-g heavy drone. The maximum payload is reached at 42 g, but already with our increased weight, we see degrading maneuverability and flight time.

To gain flight time and agility back, we chose a 350-mAh battery instead of the 250-mAh stock battery, at the cost of adding 1.1 g. However, to compare the additional power load brought by our ToF multizone deck, we tested how long the drone can hover with and without the ToF multizone deck. With it, the time until the low battery warning was triggered (battery voltage measurement below 3.2 V for 5 s) was on average 7'22", without it 7'56" . Assuming the full capacity of the battery is used and assuming 0.28 W for the Crazyflie electronics [11], we estimate that 680 mW are used for carrying the additional weight of the ToF multizone deck and 9.32 W for the remaining components. A power breakdown is shown in Fig. 13. We see the importance of lightweight sensors—we use 9.4% of the power for adding the multizone ToF sensor. However, the power needed for the sensor operation (3%) is less than half the additional power needed to carry the shield (6.4%).

B. Static Obstacle Avoidance Versus Speed

In the first experiment, we evaluate the braking capability of the drone when it flies toward a 1.2 m × 1.3 m static obstacle made out of cardboard. The drone flies in a straight line with a configurable maximum speed. As soon as an obstacle is detected, the drone decreases its velocity according to the algorithm presented in Section VI. We disable the steering throughout this experiment to evaluate the breaking capabilities in isolation. The drone takes off at a distance of 3.5 m away from the wall, giving it enough space to accelerate and reach the target speed. The evaluation is performed by sweeping the $v_{\max}$, which is a software parameter that indicates the target velocity the drone aims to reach in the absence of any obstacle within the FOV.

We perform the experiment for the following values of the parameter $v_{\max}$: 1, 1.5, 2, 2.5 m/s, and we present the curves for drone's position and the drone–wall distance in Fig. 14. The red curve represents the distance from the drone to the wall, and the ToF sensor provides it. The blue curve represents the velocity of the drone, and it is logged with the Vicon at a rate of 50 Hz. The subplots in Fig. 14 were aligned in time by velocity peaks, which are 1.11, 1.34, 1.53, and 1.92 m/s, for the cases (a)–(d), respectively. The drone successfully brakes in each of the four situations, and it stops about 0.2–0.5 m away from the obstacle. We also point out that the braking is not very smooth due to the oscillating behavior of the velocity curve right after braking, which is visible in all situations but especially in Fig. 14(d).

To better investigate this effect, we perform a separate experiment where the drone is commanded to accelerate up to 3 m/s, and then, suddenly brake, without using the avoidance algorithm. Furthermore, we perform this experiment for the following two cases: 1) the drone flies straight in an open space with no obstacles around and 2) the drone flies straight, but a cardboard panel is mounted 30 cm away from the stopping point of 1). Fig. 15 shows the commanded, estimated, and actual velocities in blue, red, and black, respectively. While the commanded and estimated velocities are acquired from the drone directly, the actual velocity (i.e., the ground truth) is observed and logged with the Vicon system. As its FOV is limited, the ground truth is not captured for the whole trajectory but only within the area of interest (i.e., where the drone brakes). Fig. 15(a) shows a velocity estimation error of about 0.13 m/s right after the drone brakes. Even if this error decays within about 4 s, it causes a forward drift as the drone believes it is stationary while it is actually moving forward. Therefore, during aggressive braking, the sensor readings' precision decreases, impacting the drone's state estimation accuracy. Furthermore, Fig. 15(b) shows that this effect is exacerbated by the presence of an obstacle in the proximity of the braking point, where the decay time of the velocity estimation error is significantly longer. This is due to wall effects that change the drone's dynamics and impact the accuracy of the down-pointing altitude sensor—since the detection area of the altitude sensor is instead a cone than a narrow beam, staying close to walls can lead to inaccurate altitude measurements. Therefore, poor state estimation after suddenly braking close to walls is a limitation of the drone controller itself and not of our algorithm, which explains the oscillating pattern from Fig. 15.

C. Dynamic Obstacle Avoidance Versus Speed

One of the key features of an indoor autonomous drone is the ability to avoid unpredictable dynamic obstacles, especially moving persons. Therefore, in the following experiment, we assess the avoidance capability when a person unexpectedly steps in front of the drone, leaving about 1.5 m for braking and collision avoidance as shown in Fig. 16. Similarly to the experiment in Section VII-B, we sweep the velocity in the range 1–2.5 m/s with a step of 0.5 m/s and report the results in Fig. 17. Each subplot shows the drone's trajectory, color-coded by its velocity. The red arrow indicates the moving direction of the person, while the orange dashed line indicates the drone's position when the person jumped in front of it. The drone's trajectory and velocity were acquired with the mocap, which observed the peak velocities of 1.36, 1.65, 1.93, and 2.66 m/s for the cases (a)–(d), respectively. The experiments in Fig. 17(a)–(c) show successful collision avoidance, and at low velocity [i.e., Fig. 17(a)], the avoidance appears to be smoother because the drone has more time to react. In the experiment depicted in Fig. 17(d), the drone does not manage to brake within 1.5 m, and it collides with the person. To ensure the reliability of the experiments, we performed several trials for each value of $v_{\max}$ and observed very similar behaviors to the ones presented in Fig. 17. However, for $v_{\max} = 2.5\,\mathrm{m/s}$, the drone does not always crash because of the collision but also because it gets unstable during the sudden braking.

D. Narrow Corridor

We built a 4-m long corridor of varying widths to test the drone's capability to explore narrow spaces. We started the drone at the beginning of the corridor, facing in the desired flying direction (heading straight through the corridor). We did three trials at each width: at 75 cm width, the drone always passed through without any issue; at 65 cm, only one of the trials was successful; and at 55 cm, the drone did not even enter the corridor once. Note that the drone can pass through much smaller gaps if they are not pipes but shorter obstacles, such as passing underneath a chair, as in Section VII-F. Following the geometric relations shown in Fig. 1, we can compute that the danger zone introduced in Section VI is 33 cm-wide at the reaction distance (1.4 m); however, the caution zone is 66 cm wide. As shown in Fig. 10, obstacles in the caution zone will already cause the drone to turn; however, slowly enough to successfully fly through the corridor. Fig. 18 shows the experimental setup and the results.

E. Flying in a Room With Static Obstacles

For this test, we built a closed environment in which we can track the drone with the Vicon system. We built a cardboard maze, as shown in the top row in Fig. 22. All obstacles are between 0.6 and 0.8 m high. To verify the reliability of our system, we started the drone at different random take-off points in the maze. We set the maximum target velocity to 1 m/s in all tests, as the environment is so cluttered that the drone always sees obstacles and is in the fixed velocity region anyway (see Section VI). In Fig. 19, we show one example of the flight path. We repeated the experiment three times without crash, with flight times of 6'31", 6'34", and 6'45". The drone flies at a target height of 0.5 m, with a maximum acceleration of 1.5 m/s^2 and a minimum acceleration of -20 m/s^2. As our algorithm is deterministic, the flight path converges to almost always the same loop. On average, we covered 206 m during the flights, resulting in an average velocity of 0.52 m/s.

F. Reliability Test

Reliability in a real-world scenario was assessed by 20 flights each at four different maximum target velocities in an office

environment, more precisely an 11 m× 6 m meeting room. The floorplan of the environment can be found in Fig. 21. The meeting room features seven tables, chairs, other utilities like a projector, phone, jacket rack, and several usually closed doors. Occasionally, people pass through the meeting room. We configured the drone to fly at 0.4 m over the ground and tested it with four different maximum velocities −0.5, 1, 1.5, and 2 m/s. At 0.5 m/s, we did not experience any crashes in 20 flights but always landed safely after a low battery warning. At higher velocities, the reliability dropped to 80% at 1 m/s and even 40% respectively 10% at and 2 m/s . Note that even when experiencing a crash, the drone often completed many successful obstacle avoidance scenarios beforehand, as one trial is not one obstacle avoidance scenario but several minutes of fully autonomous flight. We log the internal state estimation as an additional measure and compute the covered distance. We display our results in Fig. 20. We see that the maximum velocity only weakly influences the flight time for successful flights. The distance covered counting only successful flights is maximized at 1.5 m/s, but note that at 2 m/s, only 2 out of 20 flights were successful, leading to not enough data points for conclusions about the flight time and distance, however, we can conclude that at 2 m/s, our obstacle avoidance algorithm ceases to work. Looking at all flights, we observe the maximum of the covered distance at 1 m/s, even though the average flight time is higher at 0.5 m/s. We conclude that we can only fly at high speeds in easy environments (big and nonreflective obstacles). For office environments, 1 m/s covers most distance per flight, but 0.5 m/s is more reliable.

G. Different Environments at Different Speeds

We tested the limits of our system by performing fully autonomous flights in various challenging environments at different maximum target velocities. In Section VII-E, we already described our baseline test—flying in a cardboard maze. Those obstacles all have the same nonreflective surface, are all taller than how high the drone will fly, and the ground is flat. In this environment, we can fly autonomously for on average 6.5 min with our system until a landing procedure is automatically triggered because of a low battery. In this scenario, we always see obstacles, so we do not accelerate over 1 m/s even if we would allow it, and hence, only tested at this maximum target velocity. To challenge the drone in real-world environments, we also tested in an office environment, to be more specific, a large meeting room, described in Section VII-F, and outdoors. Those environments are much more challenging, as they feature static and dynamic objects of various forms and surfaces with unknown velocities and directions, requiring continuous adaption of the flight heading, altitude, and speed, even including the action "Fly backward" that is triggered as described in Fig. 10. In Fig. 22, we show the average flight times, covered distance, and crash reasons at different maximum target velocities.

In general, we observe that, as expected, a slower maximum target velocity leads to fewer crashes. Almost all crashes are due to highly reflective obstacles, such as metal chair legs or cars. While approaching reflective obstacles frontally at a rather low speed can work, as the sensor will measure the reflected light approaching them at steep angles leads to failure in recognizing them. This is due to almost no deflection, and hence, no light coming back being sensed by the sensor. Also, small obstacles can only be detected from shorter distances, as then more light is deflected. This leads to more crashes at higher velocities, as we need more time to brake. We also observe the highest covered distance in the maze—even though the drone almost always sees obstacles, and thus, rarely flies at high speeds, there are no narrow dead ends and no obstacles requiring height adjustments due to exclusively large obstacles. The office environment is far more complex, featuring tables and chairs between which it is challenging and slow to find an obstacle-free path.

We also tested the drone outdoor in a hilly environment, but as the distance is computed from the internal state estimation, we cannot take the climb into account. Over all different environments and speeds, we can say that for the highest reliability, the maximum target speed should be set to 0.5 m/s, unless in environments with large and nonreflective obstacles, 1 m/s is also possible. The flight distances are strongly influenced by the time the drone spends slowing down because of obstacles and getting out of dead ends. In general, high reliability is also beneficial for maximizing the flight distance. However, flight speeds up to 1 m/s can lead to longer covered distances, especially in environments with big obstacles, such as the maze.

VIII. Discussion and Future Work

The key element of our system is a lightweight and reliable obstacle avoidance algorithm that leaves enough resources (computationally and energy-related) for other tasks. Thus, we foresee our work as a base for many future applications since reliable obstacle avoidance is only the first task toward accomplishing autonomous flying. This section provides an open discussion about the future work that can extend our perception system: either by integrating additional hardware (i.e., sensors) or by using more advanced data processing techniques.

Integrating the proposed system with existing solutions, such as wall-following and bug algorithms optimized to be computationally lightweight, as proposed in [11], [27], and [28], would result in a more robust obstacle avoidance strategy. Indeed, these works are bounded by the ultralimited sensor resolution (only four measurement points are used for obstacle avoidance), a limitation easily surmountable with the introduction of multizone ToF sensors. Such extension would cover a large application area, wherever a nanorobot needs to navigate in unknown complex environments, eventually supporting mapping [38] and path planning [28].

The primary constraint in adding more sensors comes from the additional weight, so we aimed to design the custom ToF deck as light as possible, leaving enough weight budget for additional sensors. We showed that it is possible to fly with both the multizone ToF deck and the AI-deck during the dataset acquisition. Developing a sensor fusion algorithm that can use the camera and ToF sensor information could improve the obstacle avoidance robustness and enable new capabilities, such as reliable object recognition. Adding a rear and side-facing ToF sensors on our custom deck would extend the overall FoV and therefore the system awareness, enabling the system

to avoid rear/side approaching dynamic obstacles. One of the main limitations of our system is dealing with highly reflective materials from extreme angles. Even if such material also poses challenges for traditional cameras, novel miniature radars have the potential to mitigate these issues and complement the ToF sensors.

Incorporating more sensors would result in more information to be processed, and therefore, an increased need for computational resources. Our algorithm works with a relatively low-dimensional input (i.e., 64-pixel depth map) and requires about 0.31% load from the STM32F405 microcontroller onboard the Crazyflie. This not only leaves a large computational budget for developing more complex algorithms, but also enables the system to deal with larger dimensional inputs. The multizone ToF sensor released by STMicroelectronics is the first of its kind in terms of precision, form factor, and pixel number. However, further releases could come with improved performance, such as a higher measurement range, number of pixels, or data rate. Since higher dimensional outputs would not be as straightforward to process and interpret as in our case (i.e., 8×8), more complex algorithms such as CNNs could be a good candidate for dealing with a larger input, and therefore, enabling new functionalities, such as flying in uneven terrains (e.g., stairs) or detecting narrow passages. Even if CNNs usually require large amounts of memory and computational resource, novel parallel system-on-chips—such as the GAP8 (PULP-based) onboard the AI-deck—proved to be very effective in running such models [13] given the real-time constraints of autonomous navigation.

IX. Conclusion

The article presented an onboard obstacle avoidance perception system to enable autonomous navigation with nano-UAVs. It allowed nano-UAWs to autonomously explore office environments reliably, only using onboard computing. The authors used a Crazyflie 2.1 that already featured an IMU, extended by a flow deck and our multizone ToF deck, featuring a forward-facing 64 pixels ranger sensor. All the processing was done onboard, easily fitted on an STM32F405 microcontroller next to the flight controller, only using up 0.31% of the computational power and featuring a 210-μs latency. The power to lift the additional sensor with all accompanying electronics as well as the supply of it totals was less than 10% of the whole drone, making a flight time of around 7 min possible. The authors tested the system in various challenging environments, achieving autonomous flights with distances up to 212 m. The 100% reliability and high agility at a low speed in an office environment provided a base for many more complex future applications. The authors also provided a dataset with ToF, state estimation, and camera data to learn or simulate future applications.

Acknowledgment

The authors would like to thank STMicroelectronics for the support provided during the development of this work. The authors would also like to thank I. Ostovar for his work and Prof. E. Sanchez for his guidance and support.

References

[1] H. Shakhatreh et al., "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[2] N. Gyagenda, J. V. Hatilima, H. Roth, and V. Zhmud, "A review of GNSS-independent UAV navigation techniques," *Robot. Auton. Syst.*, vol. 152, 2022, Art. no. 104069.

[3] H. Müller, D. Palossi, S. Mach, F. Conti, and L. Benini, "Fünfiiber-drone: A modular open-platform 18-grams autonomous nano-drone," in *Proc. Des., Automat. Test Europe Conf. Exhib.*, 2021, pp. 1610–1615.

[4] R. PS and M. L. Jeyan, "Mini unmanned aerial systems (UAV)—A review of the parameters for classification of a mini UAV," *Int. J. Aviation, Aeronautics, Aerosp.*, vol. 7, no. 3, 2020, Art. no. 5.

[5] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 1205–1212.

[6] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Sci. Robot.*, vol. 6, no. 59, 2021, Art. no. eabg5810.

[7] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Trans. Robot.*, vol. 36, no. 1, pp. 1–14, Feb. 2020.

[8] S. Rezwan and W. Choi, "Artificial intelligence approaches for UAV navigation: Recent advances and future challenges," *IEEE Access*, vol. 10, pp. 26320–26339, 2022.

[9] N. Elkunchwar, S. Chandrasekaran, V. Iyer, and S. B. Fuller, "Toward battery-free flight: Duty cycled recharging of small drones," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5234–5241.

[10] K. N. McGuire, G. C. de Croon, and K. Tuyls, "A comparative study of bug algorithms for robot navigation," *Robot. Auton. Syst.*, vol. 121, 2019, Art. no. 103261.

[11] K. McGuire, C. De Wagter, K. Tuyls, H. Kappen, and G. C. De Croon, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Sci. Robot.*, vol. 4, no. 35, 2019, Art. no. aaw9710.

[12] M. Coppola, K. N. McGuire, C. De Wagter, and G. C. H. E. De Croon, "A survey on swarming with micro air vehicles: Fundamental challenges and constraints," *Front. Robot. AI*, vol. 7, 2020, Art. no. 18. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2020.00018

[13] V. Niculescu, L. Lamberti, F. Conti, L. Benini, and D. Palossi, "Improving autonomous nano-drones performance via automated end-to-end optimization and deployment of DNNs," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 4, pp. 548–562, Dec. 2021.

[14] D. Wang, W. Li, X. Liu, N. Li, and C. Zhang, "UAV environmental perception and autonomous obstacle avoidance: A deep learning and depth camera combined solution," *Comput. Electron. Agriculture*, vol. 175, 2020, Art. no. 105523.

[15] G. Schouten and J. Steckel, "A biomimetic radar system for autonomous navigation," *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 539–548, Jun. 2019.

[16] J. N. Yasin, S. A. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, "Unmanned aerial vehicles (UAVs): Collision avoidance systems and approaches," *IEEE Access*, vol. 8, pp. 105139–105155, 2020.

[17] P. Zhao, C. X. Lu, B. Wang, N. Trigoni, and A. Markham, "3D motion capture of an unmodified drone with single-chip millimeter wave radar," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 5186–5192.

[18] B. P. Duisterhof et al., "Learning to seek: Autonomous source seeking with deep reinforcement learning onboard a nano drone microcontroller," 2019, *arXiv:1909.11236*.

[19] S. Bahnam, S. Pfeiffer, and G. C. De Croon, "Stereo visual inertial odometry for robots with limited computational resources," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 9154–9159.

[20] A. Garofalo, M. Rusci, F. Conti, D. Rossi, and L. Benini, "PULP-NN: Accelerating quantized neural networks on parallel ultra-low-power RISC-V processors," *Philos. Trans. Roy. Soc. A*, vol. 378, no. 2164, 2020, Art. no. 20190155.

[21] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018.

[22] S. Li, E. Van der Horst, P. Duernay, C. De Wagter, and G. C. De Croon, "Visual model-predictive localization for computationally efficient autonomous racing of a 72-g drone," *J. Field Robot.*, vol. 37, no. 4, pp. 667–692, 2020.

[23] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2124–2136, Mar. 2019.

[24] N. Mohamed, J. Al-Jaroodi, I. Jawhar, A. Idries, and F. Mohammed, "Unmanned aerial vehicles applications in future smart cities," *Technological Forecasting Social Change*, vol. 153, 2020, Art. no. 119293.
[25] R. Shakeri et al., "Design challenges of multi-UAV systems in cyber-physical applications: A comprehensive survey and future directions," *IEEE Commun. Surveys Tut.*, vol. 21, no. 4, pp. 3340–3385, Fourthquarter 2019.
[26] T. Polonelli, Y. Qin, E. M. Yeatman, L. Benini, and D. Boyle, "A flexible, low-power platform for UAV-based data collection from remote sensors," *IEEE Access*, vol. 8, pp. 164775–164785, 2020.
[27] T. Chathuranga, M. Padmal, D. Bibile, P. Jayasekara, and N. Kottege, "Sensor deck development for sparse localization and mapping for micro UAVs to assist in disaster response," in *Proc. Australas. Conf. Robot. Automat.*, 2020.
[28] B. P. Duisterhof et al., "Tiny robot learning (tinyRL) for source seeking on a nano quadcopter," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 7242–7248.
[29] S. Tijmons, G. C. H. E. De Croon, B. D. W. Remes, C. De Wagter, and M. Mulder, "Obstacle avoidance strategy using onboard stereo vision on a flapping wing MAV," *IEEE Trans. Robot.*, vol. 33, no. 4, pp. 858–874, Aug. 2017.
[30] P. Foehn et al., "Alphapilot: Autonomous drone racing," *Auton. Robots*, vol. 46, no. 1, pp. 307–320, 2021.
[31] A. Rovira-Sugranes, A. Razi, F. Afghah, and J. Chakareski, "A review of AI-enabled routing protocols for UAV networks: Trends, challenges, and future outlook," *Ad Hoc Netw.*, vol. 130, 2022, Art. no. 102790.
[32] F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning vision-based flight in drone swarms by imitation," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4523–4530, Oct. 2019.
[33] D. Gehrig, M. Rüegg, M. Gehrig, J. Hidalgo-Carrió, and D. Scaramuzza, "Combining events and frames using recurrent asynchronous multimodal networks for monocular depth prediction," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2822–2829, Apr. 2021.
[34] M. Muglikar, D. P. Moeys, and D. Scaramuzza, "Event guided depth sensing," in *Proc. Int. Conf. 3D Vis.*, 2021, pp. 385–393.
[35] Y. D. Yasuda, L. E. G. Martins, and F. A. Cappabianco, "Autonomous visual navigation for mobile robots: A systematic literature review," *ACM Comput. Surveys*, vol. 53, no. 1, pp. 1–34, 2020.
[36] V. Niculescu, H. Müller, I. Ostovar, T. Polonelli, M. Magno, and L. Benini, "Towards a multi-pixel time-of-flight indoor navigation system for nano-drone applications," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf.*, 2022, pp. 1–6.
[37] C. Liu, E.-J. Van Kampen, and G. C. H. E. De Croon, "Adaptive risk tendency: Nano drone navigation in cluttered environments with distributional reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 7198–7204.
[38] S. Karam et al., "Micro and macro quadcopter drones for indoor mapping to support disaster management," *ISPRS Ann. Photogrammetry, Remote Sens. Spatial Inf. Sci.*, vol. 1, pp. 203–210, 2022.

Hanna Müller (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering and information technologies, in 2017 and 2020, respectively, from ETH Zürich, Zürich, Switzerland, where she is currently working toward the Ph.D. degree in electrical engineering with the Integrated Systems Laboratory.

Her research interests include low-power systems, wireless sensor networks, and onboard intelligence—especially for obstacle avoidance and localization of nanodrones.

Vlad Niculescu (Student Member, IEEE) received the master's degree in robotics, systems, and control, in 2019, from the ETH Zürich, Zürich, Switzerland, where he is currently working toward the Ph.D. degree in electrical engineering with the Integrated Systems Laboratory.

His research interests include developing localization and autonomous navigation algorithms that target ultralow-power platforms, which can operate onboard nanodrones.

Mr. Niculescu competed in more than ten international student competitions during the bachelor and master period, and he was the electrical lead of the student project Swissloop, which won second place and the innovation award in the SpaceX Hyperloop Pod Competition 2019.

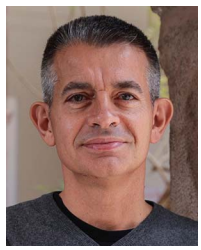
Tommaso Polonelli (Member, IEEE) received the M.Sc. and Ph.D. degrees in electronic engineering from the University of Bologna, Bologna, Italy, in 2017 and 2020, respectively.

He is currently a Postdoctoral Researcher with ETH Zürich, Zürich, Switzerland. He has collaborated with several universities and research centers, such as the University College Cork, Cork, Ireland, and the Imperial College London, London, U.K. He has authored more than 40 papers in international journals and conferences. His research interests include wireless sensor networks, Internet of Things, autonomous unmanned vehicles, power management techniques, structural health monitoring, and the design of ultralow power battery-supplied devices with onboard intelligence.

Michele Magno (Senior Member, IEEE) received the master's and Ph.D. degrees in electronic engineering from the University of Bologna, Bologna, Italy, in 2004 and 2010, respectively.

He has been working with ETH Zürich, Zürich, Switzerland, since 2013, and has become a Visiting Lecturer or Professor with several universities, namely the University of Nice Sophia, France, Enssat Lannion, France, University of Bologna, and Mid University Sweden, where he is currently is a full Visiting Professor with the Electrical Engineering Department. He is currently a Senior Scientist with the Department of Information Technology and Electrical Engineering (D-ITET), ETH, where since 2020, he has been leading the Center of Project-Based Learning. He has authored more than 220 papers in international journals and conferences. His current research interests include smart sensing, low-power machine learning, wireless sensor networks, wearable devices, energy harvesting, low-power management techniques, and extension of the lifetime of batteries-operating devices.

Dr. Magno is an ACM member. Some of his publications were awarded as best papers awards at IEEE conferences. He was also the recipient of awards for industrial projects or patents.



Luca Benini (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 1997.

He holds the chair of digital circuits and systems with ETH Zürich, Zürich, Switzerland, and is a Full Professor with the Università di Bologna, Bologna, Italy. His research interests include energy-efficient parallel computing systems, smart sensing microsystems, and machine learning hardware.

Dr. Benini is a Fellow of the ACM and a Member of the Academia Europaea. He was the recipient of the 2016 IEEE CAS Mac Van Valkenburg award, the 2020 EDAA achievement Award, and the 2020 ACM/IEEE A. Richard Newton Award.