



Target following on nano-scale Unmanned Aerial Vehicles

Conference Paper**Author(s):**

[Palossi, Daniele](#) ; Singh, Jaskirat; Magno, Michele; [Benini, Luca](#) 

Publication date:

2017

Permanent link:

<https://doi.org/10.3929/ethz-b-000231127>

Rights / license:

[In Copyright - Non-Commercial Use Permitted](#)

Originally published in:

<https://doi.org/10.1109/IWASI.2017.7974242>

Target Following on Nano-Scale Unmanned Aerial Vehicles

Daniele Palossi*, Jaskirat Singh*, Michele Magno*[†] and Luca Benini*[†]

*Department of Information Technology and Electrical Engineering - ETH Zürich, Switzerland

[†] Department of Electrical, Electronic and Information Engineering “Guglielmo Marconi” - University of Bologna, Italy
Email: name.surname@iis.ee.ethz.ch

Abstract—Unmanned Aerial Vehicles (UAVs) with high level autonomous navigation capabilities are a hot topic both in industry and academia due to their numerous applications. However, autonomous navigation algorithms are demanding from the computational standpoint, and it is very challenging to run them on-board of nano-scale UAVs (i.e., few centimeters of diameter) because of the limited capabilities of their MCU-based controllers. This work focuses on the object tracking capability, (i.e., target following capability) on such nano-UAVs. We present a lightweight hardware-software solution, bringing autonomous navigation on a commercial platform using only on-board computational resources. Furthermore, we evaluate a parallel ultra-low-power (PULP) platform that enables the execution of even more sophisticated algorithms. Experimental results demonstrate the benefits of our solution, achieving accurate target following using an ARM Cortex M4 microcontroller consuming $\approx 130\text{mW}$. Our evaluation on a PULP architecture shows the proposed solution running up-to 60 frame-per-second in a power envelope of $\approx 30\text{mW}$ leaving more than 70% of the computational resources free for further on-board processing of more complex algorithms.

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) with complex autonomous navigation capabilities are gaining interest in both industry and academia due to their use in several applications and in different sectors [1], [2]. UAVs are widely adopted for surveillance, inspection of hazardous and hostile environments, aerial photography, just to name a few [1], [3].

The major trends of the evolution of UAVs are miniaturization and autonomous navigation capabilities. In this context, commercial-off-the-shelf (COTS) quadrotors have already reached the nano-scale, featuring only few centimeters in diameter and few tens of grams in weight. Such devices, still lack of autonomous navigation capabilities that their larger counterparts boast. This is, mainly, due to their limited computational resources, heavily constrained by their small power envelopes, making them inadequate for the execution of sophisticated algorithms. In fact, state-of-the-art techniques for implementing autonomous navigation use computationally demanding techniques that process video streams produced by on-board cameras [4].

One of the basic autonomous flying functionalities is *hovering*: the capability of the vehicle to maintain a desired 3-dimensional position in space over time, autonomously correcting its flight asset [2], [5], [6]. State-of-the-art hovering on standard-size UAVs relies on visual algorithmic pipelines to precisely determine position and orientation by analyzing video streams from an on-board camera (i.e., Visual Odometry) [4]. Due to the already mentioned limitations in terms of compute power, such techniques are typically not affordable at the nano-scale; most existing COTS systems in this category

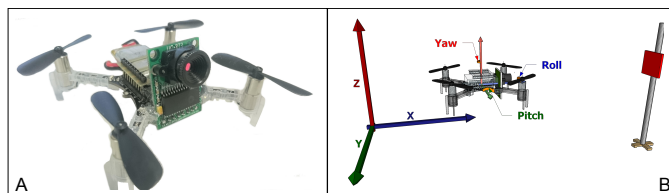


Fig. 1. A) Prototype of the system. B) Object tracking experimental setup.

(e.g., Bitcraze CrazyFlie 2.0) try to implement hovering by means of low-power and noisy sensors like barometers.

Thus, on nano-scale UAVs, the first challenge is represented by the need of a lightweight, precise hovering sub-system able to keep the vehicle at the target position (\pm some tolerance), minimizing the utilization of the limited computational resources. Precise hovering paves the way to the development of more complex autonomous navigation capability. Unfortunately, the visual hovering functionality alone is sufficient to saturate the computational capability of a state-of-the-art MCU, and the limited power budget of a nano-UAV prevents the use of more powerful embedded computers. Thus, an ultra-low-power parallel platform [2] is a promising solution to push autonomous navigation functions beyond the current limits.

In this work we extend the hardware and software of a 27-grams nano-size COTS quadrotor (Crazyflie 2.0) to achieve a object tracking capability. The Crazyflie 2.0 has been modified integrating a camera module on the vehicle from an hardware point of view. On the software side, we first increased the accuracy of the pre-existing inertial hovering control and then we implement a low-complexity visual tracker which enables the deployment of an object tracking control algorithm. The object tracking has been demonstrated to be able to achieve up to 60 frame-per-second (fps) on the quadrotor's microcontroller ARM Cortex M4. Moreover, we ported the target-following capability also on an ultra-low power parallel platform (PULP) to evaluate the benefits in terms of power consumption and computational resources used and available for further advanced algorithms.

The main contributions of this work are:

- an improved inertial hovering sub-system, capable of reducing the drift of $\approx 50\%$ w.r.t. the original system, has been implemented in a nano-scale UAV;
- a low-computation object tracking module, tested on the field, able to track movements of monochromatic red objects with an average error of $\approx 34\text{cm}$;
- an insightful evaluation on the accuracy and performance of the proposed solution on both the Cortex M4 and the PULP architecture;
- the development of a fully working cyber-physical prototype, shown in Fig. 1 (A);

- the evaluation of a novel PULP platform that can enable more sophisticated algorithms for autonomous navigation in nano-scale UAVs.

II. RELATED WORK

In the last decade, the trends of evolution for UAVs have mainly followed two directions. The first one focuses on the development of accurate algorithms (e.g., employing deep learning [7]), aimed to achieve autonomous navigation capability understanding the surrounding environment [1] (e.g. decision making, autonomous tracking, etc). For this first direction, the state-of-the-art is defined by high-level capabilities [7], [4] achieved on large (*standard-size*) robots operating in a power envelope of tens of *Watts* [8]. The literature demonstrates that UAVs autonomous navigation is a crucial feature [7], [8] and brining it in small form factor UAVs is still a challenging task [9].

In fact, the second direction focuses on vehicle miniaturization, which has today reached the *pico-scale* [10], featuring only few grams of mass and a total power consumption of $\approx 100mW$ [11]. Both the directions shown potential for the next generation UAVs that are targeting the nano-scale, coupled with autonomous capabilities. In this papers we tackle the challenge of autonomous navigation in nano-scale UAV combining video processing, inertial sensors and advanced control on a commercial MCU. Moreover, we demonstrate the benefit of the ultra-low power parallel paradigm exploring a novel ultra-low power platform that meets the strict power envelope constraints of nano-UAVs.

One of major challenge, in UAVs domain is the trade-off between power consumption and processing capability, as they are supplied by low weight batteries. Wood *et al.* [10] estimated in $5mW$ the power budget for on-board computation on *pico-size* UAVs, showing how this only accounts for 5% of the total, the rest being used by the propellers (86%) and the low-level control parts (9%). The taxonomy presented by Palossi *et al.* [2] defines for the nano-size UAV, that is the target of this work, a total power envelope of $\approx 5W$, bounding the computational budget to $250mW$.

Today, most of promising algorithms are using visual approaches to achieve the best performance in autonomous navigation [7]. Moreover, it has been shown that, in order to enable autonomous visual navigation, it is necessary to provide and process a real-time visual stream running at a minimum of $\approx 10fps$ [6], [12], defining a second bound on the computational requirement.

It is undeniable that energy-efficient architectures, highly optimized software and new classes of algorithms – conceived with low power consumption in mind – are compulsory, if state-of-the-art navigation capabilities (e.g. hovering, path-planning, object tracking, obstacle avoidance, etc.) are to be brought onto the most challenging *nano-size* and *pico-size* classes of UAVs. Palossi *et al.* [13] show how parallelism is a key asset to achieve energy-efficient path planning using only on-board computational resources for *standard-size* UAVs.

At the *micro-size* class there are several works able to map high level navigation skills in “relatively” low-power architectures, but they are still an order of magnitude away from the *nano-size* power-budget [4]. Forster *et al.* [4] propose a precise, robust and fast semi-direct monocular visual odometry algorithm, running on a embedded processing device *Odroid-*

U2 featuring a 4-core ARM Cortex A-9 computational unit running at $1.6GHz$.

In nano-size UAV, the state-of-the-art is represented by solutions that either perform very basic functionality like hovering [2], [6] or offload computation to some external base-station [5], [14]. A method to surpass the computational limitations of such MCU-based UAVs is presented by Briod *et al.* [6], who show an ego-motion estimation algorithm for hovering that does not rely on feature tracking. On one side their solution requires comparatively much simpler electronics and can rely on the available on-board MCU. On the other side, as discussed by the same authors, the method does not reach the accuracy of techniques based on feature tracking, with an average drift of $50cm$ after 2 minutes of flight.

Another option, presented by Dunkley *et al.* [5] is to offload all sensors’ information to a remote, power-unconstrained base-station through a wireless link, relying on a visual-inertial navigation system for a *nano-scale* quadrotor. Here all the computation is performed off-board, streaming video and inertial information to a ground-based laptop. Zhang *et al.* [14] proposed a visual-inertial odometry system for a 46-grams UAV, operating in a GPS-denied environments. Also in this case, both visual and inertial information are streamed to a laptop for the intense computations (i.e., pose estimation).

Considering our target application, i.e., *object tracking*, there are many examples of *standard-size*, power-hungry COTS UAVs able to provide such capability relaying on on-board computation (e.g., DJI Phantom 4). Pestana *et al.* [15], have shown a *standard-size* quadrotor (i.e., AR Drone 2.0) performing object-following offloading the intense computation to a base-station and receiving back attitude commands.

In this work we propose and demonstrate that *object tracking* on a nano-size UAV, relaying only on on-board sensors and computation is i) feasible through the proposed visual-inertial approach and ii) the autonomous navigation capability can be furthermore increased by the adoption of parallel MCUs-class-of-device paradigm.

III. ADVANCED NANO-SIZE UAV

In this section we introduce the UAV platform used in our work. We describe both the available computational resources and the existing sensing system. We also analyze the software architecture showing the computational and memory bounds. Then, we describe how we extended the on-board visual sensing capability through the camera integration. Lastly, we introduce a parallel ultra-low power (PULP) architecture as solution to improve the system, reducing the limitations introduced by computational intensive visual pipeline built on top of the camera.

A. Crazyflie Platform

Our target vehicle is *Bitcraze CrazyFlie 2.0* (CF), an open-source and open-hardware nano-size quadcopter. It weights $27g$ with a diameter of $92mm$ and maximum take-off weight of $42g$ (i.e., a payload of $15g$). The on-board main processing unit is the STM32F405 microcontroller (MCU), while a Nordic nRF51 module is responsible for the wireless communication. The STM32 is an ARM Cortex-M4F, operating at $168MHz$. The on-board sensing is performed by a 9-axis IMU MPU-9250 which contains a gyroscope, an accelerometer, and a magnetometer, and a ST LPS25H pressure sensor with a typical accuracy of ± 1 meter. The vehicle is powered by

a 240mAh Li-Po battery, that enables a flight time of ≈ 7 minutes. The CF has been chosen also for the available expansion interfaces, crucial for the integration of additional sensors, such as a camera. This interface offers both SPI and I2C buses directly accessible through an *expansion connector*.

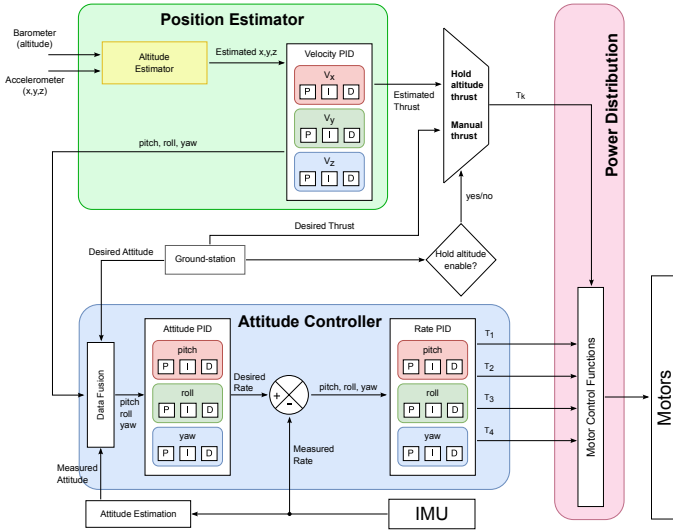


Fig. 2. Nano-size quadrotor control flow block diagram.

The main flight control flow is reported in Fig. 2. The overall control is built on top of several proportional integral derivative (PID) controllers in closed-loop configuration. Key software modules are the *attitude controller*, the *power distribution* and the *position estimator* block. The main flow, always active, is represented by the attitude controller that first tries to determine how to compensate the current attitude (based on gyroscope and accelerometer data) to match the desired attitude sent by the ground-station. Then, the angles (yaw, pitch, roll) are converted in angle rates and propagated to the power distribution that merges them with the target thrust specified by the user and computes the intensity for the motors.

The CF also offers the possibility to enter in *altitude hold* mode (i.e., it tries to keep the current altitude over time). This mode is mapped in the position estimator. It relies on the existing on-board sensors (i.e., accelerometer and barometer) to estimate the current vertical position (i.e., Z-axis) through a *complementary filter* [16] meaning both high-pass and low-pass filters simultaneously (*altitude estimator* in Fig. 2). Then, it tries to maintain the current altitude providing his attitude prediction to the attitude controller. Such prediction is then merged with the attitude controller internal estimation described in the main flow.

In order to evaluate the available computational and memory budget of the original architecture we performed a first evaluation on processing and communication modules and on memory usage running the CF’s stock firmware. As shown in Fig. 3, the firmware on the STM32 leaves free 56% of the total memory (i.e. $\approx 100KB$) and 25% of the computational budget. In Fig. 3 (on the right) we report the most computational intense tasks active during the flight: sensor data collection task (*SENS*), wireless communication task (*SYSL*), attitude control stabilizer task (*STAB*) and the task for the conversion of the transfer protocol (*CRTP*). In order to evaluate the free CPU load we introduced a computational intense “dummy” task

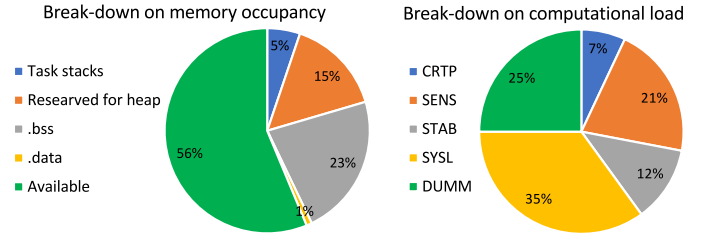


Fig. 3. Memory occupancy (left) and on computational load (right).

(*DUMMY*) in order to prevent the OS in over-scheduling other low-priority tasks due to the availability of CPU resources.

B. Visual Sensor Integration

To enable the object tracking capability we extended the CF’s sensing system with an image sensor. The ArduCAM Mini V2 camera shield was selected on the basis of its weight, power consumption, form factor, frame rate and software support. The camera module is based on the OmniVision OV2640 a 2Mpixel chip, with a maximum frame rate of 60fps and it is interfaced using an I2C bus and SPI bus. This lightweight camera, with a weight of only 5g and a form factor of $24 \times 34 \times 6$ mm, is based on a rolling shutter sensor that consumes on average 350mW (experimentally measured).

The I2C is directly connected to the OV2640 and is used for the registers configuration. Instead, the SPI channel is used for accessing the raw data through a FIFO buffer. We integrated the existing camera driver on the CF firmware achieving an image stream of 25fps. The prototype is shown in Fig. 1 (A).

C. PULP Architecture

To investigate on the possible technology enabling the new generation of autonomous nano-scale UAVs, we evaluate an energy efficient parallel ultra low power platform: *PULP*. *PULP* can boost up both energy efficiency and available computation resources to process complex algorithms. The platform is a scalable, clustered computing platform designed to operate on a large range of operating voltages, achieving in this way a high level of energy efficiency [17]. In particular, we focus on the fourth embodiment of the platform, called *Fulmine*. *Fulmine* features a single quad-core cluster integrated with 192kB of L2 SRAM memory and several IO peripherals.

The *PULP* cluster has 4 OR10N cores improved with extensions for higher throughput and energy efficiency in parallel signal processing workloads. The ISA extensions of the core include general-purpose enhancements, such as zero-overhead hardware loops and load and store operations embedding pointer arithmetic, and other DSP extensions that can be explicitly included by means of intrinsic calls.

To avoid the energy overhead of memory coherency, the cores do not have private data caches: they all share a L1 multi-banked tightly coupled data memory (*TCDM*) acting as a shared data scratchpad. Intra-cluster communication is based on a high bandwidth *low-latency interconnect* implementing word-level interleaving to reduce access contention to the *TCDM*. A lightweight multi-channel DMA enables fast communication with the L2 memory and external peripherals. The DMA features a direct connection to the *TCDM* to reduce power consumption.

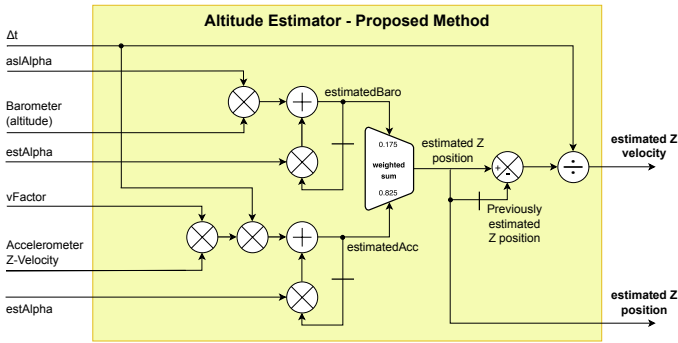


Fig. 4. Proposed altitude estimator control flow.

IV. AUTONOMOUS NAVIGATION CAPABILITY

In this section we describe the proposed techniques to extend the autonomous navigation capability of our target nano-UAV. We first illustrate our proposed solution to enhance the accuracy of the existing *altitude hold* control flow, only using the on-board inertial and pressure sensor. Then, thanks to the improved hovering capability and the on-board visual sensor, we present a simple visual algorithm for red-based object tracking capable of following a target object exploiting only on-board computational resources.

A. Altitude Hold Enhancement

As introduced in Sec. I, a key component of any autonomous rotorcraft is the capability of precise *hovering* [2], [6]. The CF flight mode relays on the inertial sensor (affected by constant drift [18]) and the pressure sensor characterized by an accuracy error of $\pm 1.69m$, that in our application scenario makes it unreliable as basic component to build more complex tasks on top of it.

To achieve the target-following, we first propose an enhanced *altitude hold* based on a more robust position estimator. The basic idea is to split the original complementary filter into two independent filters, one for the barometer and a second one for the accelerometer, both estimating the vehicle's position w.r.t the Z-axis (i.e., altitude), as shown in Fig. 4.

Each filter has its own feedback loop, where the resiliency to high frequency signal variation is determined by the *estAlpha* coefficient. Then, the two independent flows are merged with a weighted sum, where the accelerometer's data is weighted with a higher gain than the barometer's data. In this way, the estimation mostly rely on the accelerometer information, where the constant drift is partially compensated by the the barometer flow. Note that, for our target application constant smooth drift is favorable w.r.t. jerk movements (that would cause very noisy images).

B. Object Tracking Algorithm

Enabling a quasi-stationary vehicle's motion with smooth movement is a key feature to introduce into the system the object tracking capability with a simple, low-computation, visual algorithm. As proof-of-concept we propose a lightweight approach which localizes a structure that significantly differs from its neighbors in intensity and texture. This approach has been used to detect monochromatic red-based objects. Then, after the object coordinates are determined on the image plane, they are used to correct the desired attitude through the *control fusion* stage of the algorithm.

Object Detection: For each RGB image we first extract the red channel checking also the intensity of the others in order to distinguish between completely white and red regions. Then, the red image is divided, and labeled, into homogeneous segments on the basis of the pixel's intensity. The region of interest (ROI) is defined as the segment with the highest number of pixel and the coordinates of its center of mass (CoM) are forwarded to the next stage.

Control Fusion: In this stage the coordinates of the target's CoM (i.e., CoM of the ROI) are converted into a desired attitude for the existing control flow. In our approach we first split the image in 3×3 rectangular sectors (3 rows and 3 columns). Then, we adjust the attitude in order to keep, over time, the ROI both in the central sector and with a constant dimension (i.e., constant number of pixels). Moreover, we increase the robustness discarding ROIs that exceed a maximum displacement threshold between two consecutive images.

For example, if the ROI appears on the left-most column of the image, the nano-UAV needs to move to the left (Y-axis in Fig. 1 (B)) to bring back the ROI in center of the image. Similarly, if the ROI lies on the up-most row of the image, the drone needs to increase its altitude (Z-axis in Fig. 1 (B)) and if the ROI is bigger than a given threshold then the drone is too close to the target and it needs to fly back (X-axis in Fig. 1 (B)). These three dimensions of movements are evaluated simultaneously. Thus, if the ROI appears on the right-down corner, the vehicle will fly contemporary down and on the right.

The control flow parameters influenced by the visual pipeline are the vertical velocity (V_z), the *roll* angle and the *pitch* angle (in Fig. 2), respectively compensating for the Z, Y and X displacement in Fig. 1 (B).

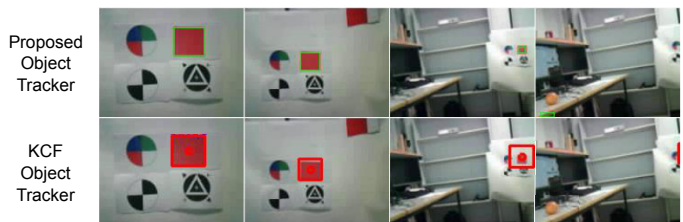


Fig. 5. Visual comparison of the proposed object tracker (green box) and the KCF algorithm (red-dot box), over four consecutive frames.

Fig. 5 presents a visual comparison of the proposed algorithm with a state-of-the-art object tracker: *Kernelized Correlation Filter* (KCF) [19]. From left to right four consecutive frames are evaluated, the target identified by the proposed algorithm is defined by a green box in contrast with the KCF detected target represented by a red box with a red dot on the center. In the first frame we can see the target object (i.e., red square) correctly tracked by both approaches. In the second frame we introduce some noise with a second potential red target (up-right corner of the image), but both approaches still hold on the correct one. Lastly, in the fourth frame the target is significantly occluded (exceeding the image boundary) and in this case only KCF is able to locate the target correctly.

V. RESULTS & DISCUSSION

In this section we will evaluate the accuracy, the power consumption and the performance of the proposed autonomous nano-UAV. In particular we will evaluate the accuracy of each

algorithm independently as well as a final evaluation of the complete system. Lastly, we will introduce an evaluation of the power consumption and computational load of both the on-board MCU and the proposed PULP platform. All the measurements addressing the motion analysis of the flying vehicle are based on external Vicon motion capture system, which is a camera-based positional system that provides sub-millimeters accuracy at $200Hz$.

A. Extended Inertial Hovering

The improvement in hovering accuracy is shown in Fig. 6 where, using the Vicon system, the original *altitude hold* and the proposed one are compared. The desired altitude (dotted yellow line) is set to 1 meter, the vertical variation (Z-axis) is reported in red for the original system and in blue for the one proposed. The experiment shows an evaluation over 37.5 seconds resulting in an improved average accuracy of the proposed inertial hovering w.r.t. the original system. In fact, the average error decrease from $33.6cm$ to $12.1cm$, with a reduction also of the maximum error, from $83.4cm$ to $41.6cm$. Note that, the reduction on the peak error implies smoother movements, resulting in the reduction of the noise in the image acquisition.

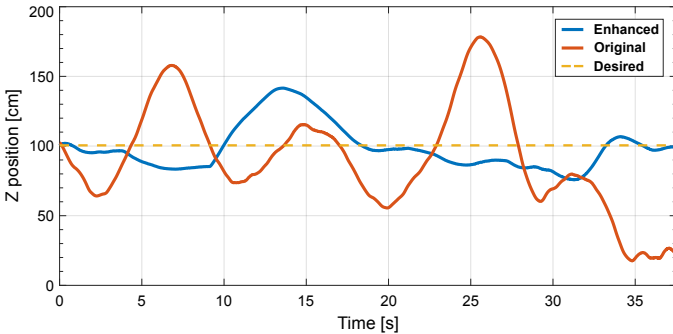


Fig. 6. Inertial hovering evaluation using the Vicon motion capture system.

B. Object Tracker

In order to evaluate the precision of the proposed re-based object tracker we compared it with the KCF algorithm [19]. In Fig. 7 we present an evaluation based on a video (708 frames) from the VOT dataset¹. In Fig. 7 (A) is presented a 4-frame sample of the video where the target is associated to a red moving object. The ground-truth, the KCF identified target and the target tracked by our algorithm are respectively represented by a green, blue and red box.

The precision has been calculated as the percentage of correctly tracked frames (w.r.t. the ground-truth) for a given tolerance threshold in pixel (X-axis). As expected KCF results more robust than the proposed algorithm with a precision of 0.9 with a tolerance error of 20 *pixel*. In fact, our object tracker needs to relax the tolerance error to 35 *pixel* to achieve similar precision.

C. Autonomous UAV Evaluation

As last test on the accuracy, we evaluate on the field the motion of the proposed autonomous nano-UAV during the object tracking activity. The setup, shown in Fig. 1 (B), is composed of a moving red marker and the vehicle free

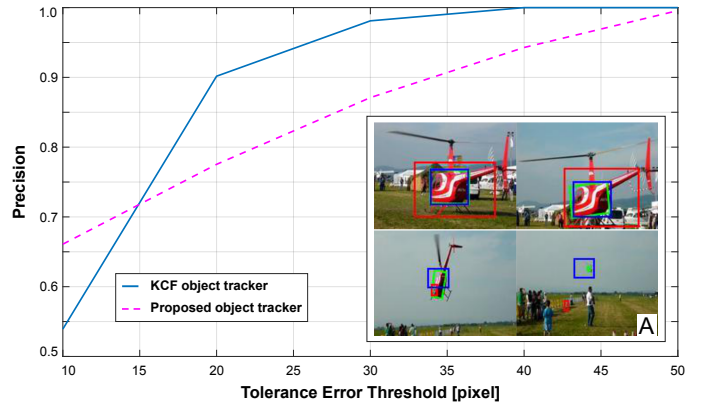


Fig. 7. Accuracy comparison between the proposed object tracker (red box in (A)) and KCF (blue box in (A)). In (A) 4 frames of the video stream evaluated with a green box for the ground-truth.

to fly autonomously, without any external interaction. The visual stream, processed on-board, provides $25fps$ due to the bottleneck introduced by the camera's driver. In Fig. 8 we evaluate each reference axis independently. The blue dashed line represents the ideal motion in order to follow each single movement of the target marker. The red line shows the actual movements of the drone. Respectively for the X, Y and Z-axis, the average error is of $34.7cm$, $28.6cm$ and $14.9cm$ and the peak error is respectively of $58.0cm$, $47.2cm$ and $29.4cm$, resulting in a smooth object following, as shown in Fig. 8.

D. Power Consumption vs. Performance

In this section we evaluate the power consumption and the computational load of the proposed solution for both the STM32 MCU and a PULP architecture [17]. For the parallel platform the results are gathered through the PULP simulator, capable of precisely modeling the behavior of the architecture in term of execution cycles, core stalls, memory conflicts, and cache misses.

Tab. I shows the power consumption and the free computational load for both architectures running the proposed object tracker algorithm with different image sizes streamed in at $60fps$, that represents the maximum frame-rate of our camera. In the experiments the STM32 has been considered with the same configuration presented in Sec. IV, characterized by an available computational budget of 25% for the object tracking. The quad-core PULP platform has been evaluated under few conservative assumptions. The implementation considered for the object tracking algorithm, is the same used on the STM32, thus non-optimized for the PULP platform. We verified that the single-core execution of this algorithm results in is very similar number of cycles on both architectures. For this reason one core of the PULP platform is dedicated to all the control tasks and the remaining three cores are used for the object tracking. A conservative speedup factor of $2.5x$ has been used scaling the single-core execution time of the algorithm to the three cores available. The results of this configuration is reported in Fig. 9.

As shown in Tab. I, with 80×60 image resolution both the STM32 and PULP are able to process the $60fps$ input stream, leaving 16.8% and $\approx 70\%$ of computational resources free, respectively. Instead, for the 160×120 pixel image only PULP is able to perform the required computation leaving more than half of the resources free for additional work. In

¹<http://www.votchallenge.net/vot2016/dataset.html>

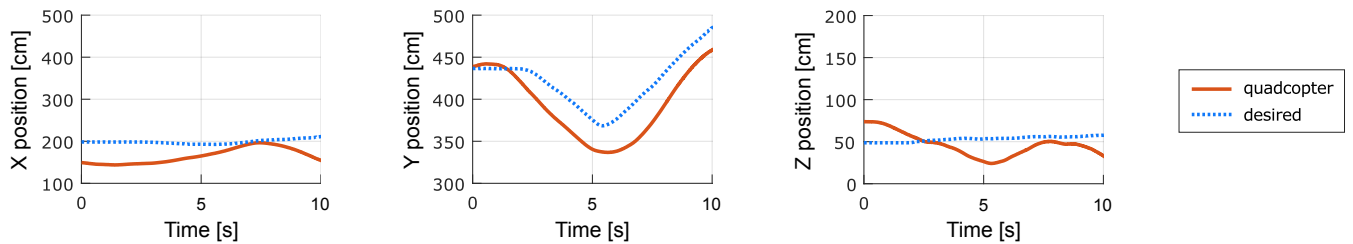


Fig. 8. Evaluation of the prototype during the target-following of a moving red marker. Measured with the Vicon motion capture system.

MCU@Frequency [MHz]	Power [mW]	Free Computation (80 × 60) [%]	Free Computation (160 × 120) [%]
STM32@168	≈ 130	16.8	-7.5
PULP@100	≈ 13	70.1	58.5
PULP@150	≈ 23	72.1	63.6
PULP@180	≈ 30	72.6	65.4
PULP@220	≈ 41	73.1	67.5

TABLE I

POWER CONSUMPTION AND FREE COMPUTATIONAL LOAD FOR THE PROPOSED OBJECT TRACKER (80 × 60 AND 160 × 120 PIXEL AT 60fps).

fact, we can see how the STM32 needs an additional 7.5% of the computational resources in order to process 60fps of 160 × 120 pixel. Tab. I shows also that both architectures (in all configurations) are able to respect the power envelope for the nano-size UAVs class of ≈ 250mW.

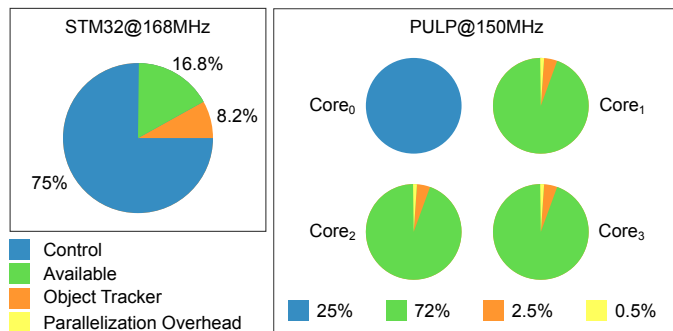


Fig. 9. Computational load breakdown for the STM32 and PULP architecture, running all the tasks with a frame resolution of 80 × 60 pixel at 60fps.

Lastly, considering the state-of-the-art KCF object tracking algorithm, we evaluated the STM32 (168MHz) capable of processing ≈ 2fps w.r.t. ≈ 11fps processed by PULP (200MHz). This last result shows how the ultra-low power parallel paradigm represents a good candidate for running high-accuracy and computational intense algorithms respecting the real-time requirement of ≈ 10fps and the nano-size UAVs power envelope.

VI. CONCLUSION

In this work we demonstrate the feasibility of precise hovering and target-following on a nano-size UAV, relying only on on-board sensors and computation. A commercial nano-quadrotor, has been equipped with a camera and its control flow has been extended i) enhancing the inertial hovering and ii) performing the proposed object tracking algorithm. Experimental results, in-field, demonstrated that simple autonomous navigation capabilities can be achieved with the existing hardware. Moreover, we shown how the adoption

of a PULP architecture can meet the required computation in ≈ 30mW power envelop, leaving enough computational resources (more than 70% of the total computational power) free for further on-board processing.

ACKNOWLEDGMENT

This work has been funded by projects EC H2020 HERCULES (688860) and Nano-Tera.ch YINS.

REFERENCES

- [1] E. Cano *et al.*, “Comparison of small unmanned aerial vehicles performance using image processing,” *Journal of Imaging*, 2017.
- [2] D. Palossi *et al.*, “Ultra low-power visual odometry for nano-scale unmanned aerial vehicles,” in *2017 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2017.
- [3] H. Menouar *et al.*, “Uav-enabled intelligent transportation systems for the smart city: Applications and challenges,” *IEEE Communications Magazine*, 2017.
- [4] C. Forster *et al.*, “Svo: Fast semi-direct monocular visual odometry,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014.
- [5] O. Dunkley *et al.*, “Visual-inertial navigation for a camera-equipped 25g nano-quadrotor,” in *IROS2014 Aerial Open Source Robotics Workshop*, 2014.
- [6] A. Briod *et al.*, “Optic-flow based control of a 46g quadrotor,” in *Workshop on Vision-based Closed-Loop Control and Navigation of Micro Helicopters in GPS-denied Environments, IROS 2013*, 2013.
- [7] D. Maravall *et al.*, “Vision-based anticipatory controller for the autonomous navigation of an uav using artificial neural networks,” *Neuro-computing*, vol. 151, pp. 101–107, 2015.
- [8] S. Ward *et al.*, “Autonomous uavs wildlife detection using thermal imaging, predictive navigation and computer vision,” in *Aerospace Conference, 2016 IEEE*. IEEE, 2016, pp. 1–8.
- [9] R. Brockers *et al.*, “Towards autonomous navigation of miniature uav,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 631–637.
- [10] R. Wood *et al.*, “Progress on ‘pico’ air vehicles,” *Int. J. Rob. Res.*, 2012.
- [11] S. B. Fuller *et al.*, “Controlling free flight of a robotic fly using an onboard vision sensor inspired by insect ocelli,” *Journal of The Royal Society Interface*, 2014.
- [12] F. Lin *et al.*, “A robust real-time embedded vision system on an unmanned rotorcraft for ground target following,” *IEEE Transactions on Industrial Electronics*, 2012.
- [13] D. Palossi *et al.*, “An energy-efficient parallel algorithm for real-time near-optimal uav path planning,” in *Proceedings of the ACM International Conference on Computing Frontiers*. ACM, 2016.
- [14] X. Zhang *et al.*, “Autonomous flight control of a nano quadrotor helicopter in a gps-denied environment using on-board vision,” *IEEE Transactions on Industrial Electronics*, 2015.
- [15] J. Pestana *et al.*, “Computer vision based general object following for gps-denied multirotor unmanned vehicles,” in *2014 American Control Conference*, 2014.
- [16] S. Colton and F. Mentor, “The balance filter,” *Presentation, Massachusetts Institute of Technology*, 2007.
- [17] F. Conti *et al.*, “An iot endpoint system-on-chip for secure and energy-efficient near-sensor analytics,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2017.
- [18] M. Rockwood *et al.*, “Adaptive drift calibration of accelerometers with direct velocity measurements,” in *Instrumentation and Measurement Technology Conference (I2MTC), 2015 IEEE International*. IEEE, 2015.
- [19] J. F. Henriques *et al.*, “High-speed tracking with kernelized correlation filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.