Poster: Testbed for Aerial Drone Applications

Mikhail Afanasov
Politecnico di Milano, Italy
mikhail.afanasov@polimi.it

Luca Mottola
Politecnico di Milano, Italy
SICS Swedish ICT, Sweden
luca.mottola@polimi.it

Kamin Whitehouse
University of Virginia, US
whitehouse@virginia.edu

Abstract

We present FLYZONE, a practical testbed architecture to experiment with airborne mobile applications. FLYZONE provides developers with a high-level interface designed for seamless transition from the testbed to real deployments. We design a low-cost localization system for the testbed, and a flexible architecture to aid scalability when operating with multiple drones. We deploy our testbed in different settings to study its performance. Preliminary results show only *7.8cm* localization error as compared to a motion capture-based systems that cost orders of magnitude more. Using modest hardware, FLYZONE maintains up to eight drones simultaneously without impacting the stability of control.

1 Introduction

Aerial robot platforms represent a new breed of computing platforms with extreme potential. The cost to build such platforms dropped significantly in recent years [5], which enables a range of sophisticated applications unfeasible with any other technology; for example, exploring nearly inaccessible areas [4] or 3D reconstruction [3, 9].

Experimenting with such application usually takes place in the target settings [12], which is, however, difficult, costly and time consuming. Bugs in software may, for example, result in the loss of drone, or damage to objects and people [13]. The continuously changing regulations on the use of civil drones further complicate matters [6], making it difficult to understand what can or cannot be tested in the wild.

To tackle these challenges, developers need to experiment in a *real-world* environment that can be practically deployed in *protected* areas and yet with the necessary functionality to *emulate real-world executions*. Such an environment is currently lacking in the state of the art.

FLYZONE aims to fill this gap by providing a number of key advantages: the testbed is not tied to a single drone platform, enabling *portability*; FLYZONE fosters *scalability* as it is capable to control multiple drones simultaneously without loss in control stability; it is assembled from *low-cost* off-the-shelf components.

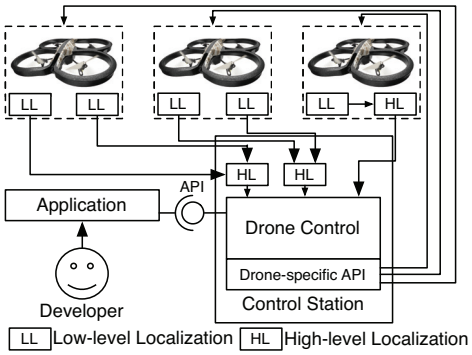


Figure 1. FLYZONE architecture.

2 Architechture

The design of FLYZONE is mainly intended for indoor operation, and its architecture aims to enable portability between the testbed and real deployment with little to no modifications in the written application. To this end, we provide a dedicated API which mimics actual drone programming systems, such as DroneKit [1] or APM Planner [2].

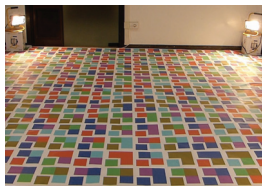
Fig. 1 shows the testbed architecture. The *drone control* component provides an API, based on which developers implement the application-specific functionality. Under the hood, this component implements a simple PID controller using drones' location as an input. The component exists in only one copy and is deployed on a dedicated machine we call *control station*. Through a drone-specific API, the component steers the deployed drones.

The *low-level localisation* component collects the raw data necessary to localize the drone. For example, it can simply read height information from the ultrasonic sensor, obtain frames from a bottom camera, or acquire signal strength for radio-based localization. There might be multiple such components deployed on a single drone. For example, one may combine visual-based localization with height detection, and thus install one such component to gather video feed, and another one to obtain height.

The low-level localization component must be installed aboard the drone. However, it does not necessarily require to be connected with the drone's software stack as our lo-

International Conference on Embedded Wireless Systems and Networks (EWSN) 2017
20–22 February, Uppsala, Sweden

ISBN: 978-0-9949886-1-4



(a) FLYZONE

(b) Deployed camera.

Figure 2. Deployment.

calization technique does not require such integration. The component can run on a separate hardware, which allows to move the functionality across different drone platforms as long as these can physically carry such separate hardware, as we describe in Sec. 3.

The *high-level localisation* component collects the data from the low-level localisation components and performs necessary processing needed to obtain the position of the drone. The number of these components in the system is proportional to the number of drones, as every such component calculates the position of a single drone.

The decoupling between the low- and high-level components further improves the portability, as the latter can run either on the control station or aboard the drone. The latter configuration help lower the processing load on the control station if needed, further improving potential scalability.

3 Deployment

At the moment of writing FLYZONE is installed in our laboratory in Politecnico di Milano, Italy and at University of Virginia, US. Both installations are actively used to experiment with aerial drone applications.

Localization in FLYZONE is based on our own visual tags, deployed on the ground at known positions, as shown in Fig. 2a. In our deployment, we use Parrot AR.Drone 2.0 equipped with a RaspberryPI V3 board and RaspiCam V2 for video capturing, as shown in Fig. 2b. The low-level localization component runs on the RaspberryPI that is powered by a separate battery. The added weight is only 105 grams.

The high-level localization component is fed with the video stream from the low-level component as input, and uses OpenCV [11] for the actual image processing. The control station takes the drone location and orientation coming from the high-level component and steers the drone using the ODC [10] Java SDK for the AR.Drone 2.0, according to the outputs of a standard PID controller. The control station we employ is equipped with an Intel Xeon E3 v1270 CPU, and a low-end NVIDIA GPU that can off-load some of the OpenCV processing from the main CPU. The current prototype operates 8 drones simultaneously. A concrete demonstration with 2 drones is visible at `vid.me/QFCg`.

The materials are entirely off-the-shelf, and their total cost is a function of how many drones need to be simultaneously supported. The total cost of our deployment that supports up to 8 drones is ≈ 2500 €, excluding the drones, and is essentially comparable to a high-end laptop.

4 Preliminary Evaluation

We compare the performance of FLYZONE against an industry-strength OptiTrack [8] motion capture system able to provide errors down to microns, at orders of magnitude higher costs. We do not investigate vertical accuracy; for

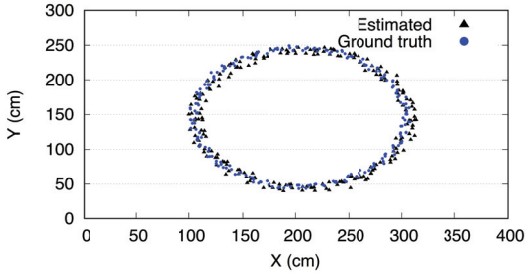


Figure 3. Localization performance of FLYZONE at 2 meters. *RMSE*=7.8*cm*

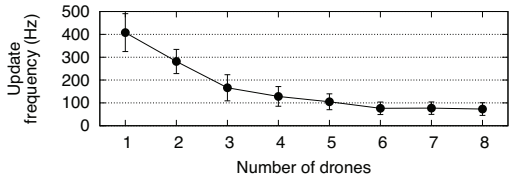


Figure 4. Computational performance of FLYZONE.

that, we use the ultrasound sensor exactly as the drones would do in a real deployment.

Fig. 3 visually depicts the accuracy of localization in FLYZONE against the OptiTrack system in a sample experiment, where the drone flies a circle so that the front camera always points to the center of the testbed. The measured RMSE is only *7.8cm*, which is better than the accuracy provided by available UWB technology in a similar setting [7].

Conversely, we measure how many drones the testbed can manage simultaneously whenever all high-level localization components are deployed on the control station, while maintaining a minimum requirement of 30 location updates per second. We observed experimentally that the latter is necessary for reliable PID control.

Fig. 4 demonstrates that the lower bound on location updates is maintained with up to eight drones with the hardware we use on the control station, described in Sec. 3. A high-end GPU expressly supported by OpenCV may, nevertheless, improve such a performance significantly [11].

5 References

[1] 3DRobotics. Dronekit. `goo.gl/IGX2t5`.
[2] ArduPilot. APM Planner. `goo.gl/3cUlFm`.
[3] BBC News. Disaster drones: How robot teams can help in a crisis. `goo.gl/6efliV`.
[4] Wolfram Burgard et al. Collaborative multi-robot exploration. In *Proceedings of ICRA*, 2000.
[5] C. Anderson. How I accidentally kickstarted the domestic drone boom. `goo.gl/SPOIR`.
[6] Federal Aviation Administration. Regulations for unmanned aircraft systems. `goo.gl/cahhbk`.
[7] Benjamin Kempke, Pat Pannuto, and Prabal Dutta. PolyPoint: Guiding indoor quadrotors with ultra-wideband localization. In *In Proceedings of HotWireless*, 2015.
[8] NaturalPoint. OptiTrack. `www.optitrack.com`.
[9] F. Nex and F. Remondino. UAV for 3D mapping applications: A review. *Applied Geomatics*, 2003.
[10] OpenDroneControl. `github.com/opendronecontrol/odc`.
[11] OpenCV: Open Source Computer Vision. `http://opencv.org`.
[12] A. Patelli et al. Model-based real-time testing of drone autopilots. In *Proceedings of DRONET*, 2016.
[13] Scientific American. 5 epic drone flying failures—and what the FAA is doing to prevent future mishaps. `goo.gl/tIXfHH`.