2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Crazyswarm: A Large Nano-Quadcopter Swarm

James A. Preiss*, Wolfgang Hönig*, Gaurav S. Sukhatme, and Nora Ayanian

***Abstract*— We outline a system architecture for a large swarm of miniature quadcopters. We discuss onboard planning, control, state estimation, communications, and a method for motion-capture localization using identical marker arrangements. We validate the system with a 49-vehicle formation flight.**

I. INTRODUCTION

Quadcopters are a popular research platform due to their agility, simplicity, and wide applicability. Most research quadcopters are large enough to carry cameras and smartphone-grade computers, but they are also expensive and require a large space to operate safely. For research into very large swarms, smaller quadcopters are more attractive. The reduced size of these vehicles motivates different system design choices compared to a typical setup for larger vehicles in smaller numbers.

Here we outline the system architecture for a swarm of 49 very small quadcopters operating indoors. The vehicles use a motion-capture system for localization and communicate over three shared radios. Our system uses off-the-shelf hardware and performs most computation onboard. To our knowledge, the system described here is the largest indoor quadcopter swarm to date, and the largest number of quadcopters controlled per radio.

II. VEHICLE

The Crazyflie 2.0 quadcopter (Fig. 1, inset) measures 92 millimeters between diagonally opposed motor shafts and weighs 27 grams with a battery. It contains a 32-bit, 168-MHz ARM microcontroller with floating-point unit that is capable of significant onboard computation. Software and hardware are both open-source. The Crazyflie communicates with a PC over the Crazyradio PA, a 2.4 GHz USB radio that transmits up to two megabits per second in 32-byte packets.

The Crazyflie's small size makes it suitable for indoor flight in dense formations. It can survive high-speed crashes due to its low inertia and poses little risk to humans.

III. ARCHITECTURE OVERVIEW

Our system is outlined in Fig. 2. We track the vehicles with a Vicon motion capture system using passive spherical markers. While this creates a single point of failure, we chose it over alternatives due to its high performance: typical position errors are less than one millimeter [1]. In comparison, a state-of-the-art decentralized localization system using ultra-wideband radio triangulation [2] showed position errors of over 10 centimeters, too large for dense formations. While vision-based methods are both accurate and decentralized [3], the required cameras and computers necessitate a much larger vehicle.

In contrast to [1], [4], [5], we implement the majority of in-flight computation onboard. The base station sends complete trajectory descriptions to the vehicle in the form of polynomials, ellipses, etc. For position feedback, the base station broadcasts vehicle poses on a shared radio channel.

The main onboard loop runs at 500 Hz. In each loop cycle, the vehicle reads its inertial measurement unit (IMU) and runs the state estimator, trajectory evaluator, and position controller. Pose messages arrive asynchronously and are fused into the state estimate on the next cycle.

Since the full trajectory plan is stored onboard, the system is robust against significant radio packet loss. If a packet is dropped, the vehicle relies on its IMU to update the state estimate. We have successfully tested the system in a hover state with position broadcasts throttled to 5 Hz.

IV. COMPONENTS

A. *Object Tracking*

One major limitation of Vicon's standard *Tracker* software is that it requires a unique marker arrangement for each object. The Crazyflie's small size limits the number of locations to place a marker, making it impossible to form 49 unique arrangements that can be reliably distinguished. Therefore, we obtain only raw point clouds from the Vicon system, and implement our own object tracking algorithm that handles identical marker arrangements.

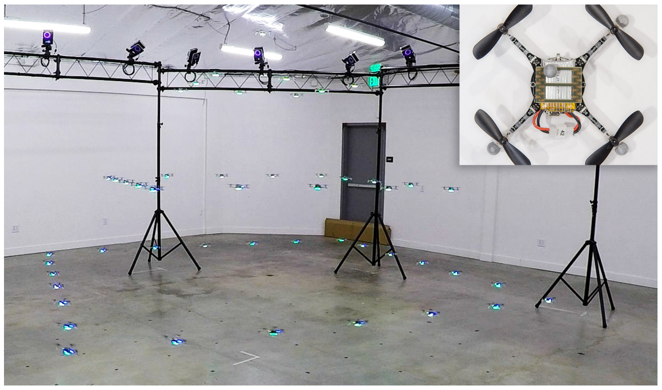

Fig. 1. Forty-nine Crazyflies flying in a four-layer rotating pyramid formation. The bottom layer is 3×3m with 0.5 m spacing between vehicles. A video of this flight is available at `https://youtu.be/ezTayb76x9U`. *Inset:* Crazyflie 2.0 quadcopter with motion capture markers.

*__Equal contributors.__

All authors are with the Department of Computer Science, University of Southern California, Los Angeles, CA, USA.
Email: {`japreiss, whoenig, gaurav, ayanian`}`@usc.edu`

This work was partially supported by the ONR grants N00014-16-1-2907 and N00014-14-1-0734, and the ARL grant W911NF-14-D-0005.

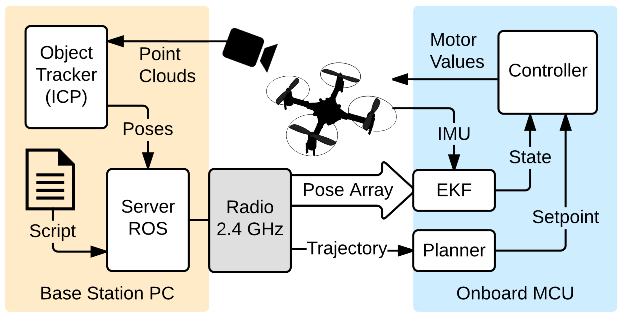

Fig. 2. Diagram of major system components. Note one-way data flow.

Our object tracker is based on the Iterative Closest Point (ICP) algorithm. We initialize the tracker with the vehicles on the floor in a known configuration. During flight, we track frame-by-frame motion with ICP. We estimate linear and angular velocity from the ICP alignments and reject physically implausible values as incorrect alignments.

B. State Estimation

An onboard Extended Kalman Filter (EKF) fuses Vicon and IMU measurements at 500 Hz. The filter state consists of position, velocity, and an attitude quaternion, following the indirect error-state formulation of [6].

Whereas some works estimate accelerometer and gyroscope bias in the EKF, we have found that these biases do not drift significantly during the Crazyflie's short battery life. We measure biases on a level floor at startup and subtract these measurements for the duration of the flight.

C. Planning

We have implemented several onboard planning methods:

- Large piecewise polynomials uploaded from the base station across multiple radio packets.
- Online planning of a single-piece polynomial starting at the current state and ending at a given state.
- Ellipses parameterized by the center, axes, and period.

These methods support a diversity of applications. For example, an aggressive maneuver might use a long piecewise polynomial optimized offline. A slow waypoint trajectory might use only the online single-piece planner. The ellipse planner is useful for demos, generating visually appealing behavior from few parameters.

In all cases, we make use of the quadcopter model's differential flatness to parameterize trajectories in a minimal representation of $\{x, y, z, \text{yaw}\}$. From these values and their derivatives, we compute the desired thrust, attitude, and angular velocities online at 500 Hz using the method of [5].

D. Communication

As illustrated in Fig. 2, all critical communication is one-way. For 49 vehicles we use three radios, with each vehicle permanently assigned to one radio. We transmit xyz positions as 16-bit fixed-point numbers and compress quaternions into 32 bits using the "smallest three" method. Compression allows us to fit two position updates in one 32-byte radio packet without degrading the measurements beyond their inherent noise level.

To support synchronized movement, we split trajectory commands into per-vehicle *upload* messages and a globally broadcast *start* message.

E. Control

We use the nonlinear position controller of [5], augmented with integral terms for position and yaw error. These are critical for coping with part variations, propeller damage, etc. Manual trimming can compensate for such issues on a single vehicle, but this is not feasible on a large fleet.

F. Software Tools

Many routine tasks become non-trivial when working with a fleet of 49 vehicles. We have developed command-line tools for mass rebooting, firmware updates, and battery voltage checks over the radio. A Python scripting layer supports development of complex multi-stage swarm flight plans.

Moving computation onboard complicates in-flight debugging due to constrained radio telemetry bandwidth and limited permanent storage for logs. To ease this problem, we structure major onboard procedures as platform-independent modules, allowing debugging in simulation on a PC.

V. CONCLUSIONS

We have outlined a system architecture for robust, synchronized, dynamic control of the largest indoor quadcopter swarm to date. Our system fully utilizes the vehicles' onboard computation, allowing for robustness against unreliable communication and a rich set of trajectory planning methods requiring little radio bandwidth. Fig. 1 shows the full 49-quadcopter swarm flying in formation.

Ongoing work focuses on reliability, including controller failure detection, downwash awareness, and object tracking failure recovery. The system described here serves as a foundation for a wide range of future work in multi-robot planning, coordination, and control. A forthcoming publication from our group will describe the components of this system, the platform constraints, tradeoffs, and the associated design decisions - in greater detail.

REFERENCES

[1] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The flying machine arena," *Mechatronics*, vol. 24, no. 1, pp. 41–54, 2014.

[2] A. Ledergerber, M. Hamer, and R. D'Andrea, "A robot self-localization system using one-way ultra-wideband communication," in *International Conference on Intelligent Robots and Systems*, 2015, pp. 3131–3137.

[3] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle," *Journal of Field Robotics*, vol. 1, 2015.

[4] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[5] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[6] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep. 2005-002, 2005.