Gesture Controlled Drone

Ian Walter and Monette Khadr
University at Albany - State University of New York
Email: iwalter@albany.edu and mkhadr@albany.edu

***Abstract*—The aim of the project is to develop a system that uses hand gestures as a method to control the flight of a drone. In this system, the drone's absolute position is not being monitored or recorded. Instead, the drone is being told to move relative to its current position based on the detected motion of the user. In order to enable fully autonomous flight, an extended Kalman filter (EKF) based procedure is used to control and adjust all six DoF (degrees of freedom) of the drone. The EKF used the readings of the pre-mounted accelerometer and gyroscope sensors on the drone as well as a supplementary optical flow sensor and a time-of-flight (ToF) sensor. The estimator uses an extended aerodynamic model for the drone, where the sensor measurements are used to observe the full 3D airspeed. To detect the motion of the user, a near-field sensor is measuring the disturbance of an electric field due to conductive objects, like a finger. Finally, to combine these systems, code will be developed on a RaspberryPi to facilitate communication from the sensor to the drone and convert from the input X, Y, Z sensor values to the values compatible with the drone system.**

I. Introduction

Drones, also known as unmanned aerial vehicles (UAV), are being heavily deployed in a wide range of commercial and recreational applications [1]. Drones are be basically observed as special flying robots that perform multiple functionalities such as data capturing and sensing from its environment. There are two broad types of drones which are fixed-wing and multirotor. In this work, the open-source Crazyflie 2.0 is used, which is a quadcopter (*i.e.* drone with four rotors) [2]. Most of the available commercial drones come with designated controllers or software applications running on users' hand-held device. In both cases, commands with detailed movement information are sent through wireless channels, which can be via Wi-Fi or Bluetooth. The Crazyflie weighs only 27 grams and is 9.2cm in length and width. The communication with Crazyflie can either be via Bluetooth or using the Crazyradio which is a long range open USB radio dongle based on the nRF24LU1+ from Nordic Semiconductor.

This work presents an attempt to add new control dimensions by allowing more degrees of freedom (DoF) to control the drone. Instead of using predesignated buttons, users can move their fingers which are then translated by the sensor into digital commands. For autonomous operation, state estimation is a fundamental requirement for these vehicles. This work adopted a quadrocopter state estimation strategy, which uses these range measurements to localize the quadrocopter inspired by the work done in [3]. A closed loop control of a quadrocopter is developed using a 2-D positioning sensor, a time-of-flight (ToF) sensor measurements fused with a dynamic model of the quadrocopter and with measurements from on-board accelerometers and rate gyroscopes. The state estimator, controller, and trajectory generator all run on-board the Crazyflie's microcontroller. The goal is to fuse all sensors information arriving at variable rates, and for this purpose, an extended Kalman filter (EKF) is considered. An advantage of using a Kalman filter is allowing simple modification to support sensor fusion by including measurement equations in the update step [4].

The contribution of this work can be divided in two folds: (1) Demonstrating that closed loop control of the drone is necessary for agile and controllable drone flight maneuvers (2) Creating a framework that translates hand movements into drone trajectories. The remaineder of the paper is organized as follows: the system model is presented in Section II, with details on the utilized sensors. Section III discusses the drone's inner state estimation problem and the proposed EKF. System analysis is listed in Section IV, and finally the paper concludes in Section V.

II. System Model

The system model of the proposed gesture controlled drone system is depicted in Fig. 1. The system consists mainly of the hand movement sensor, the drone, and the RaspberryPi where all the processing and timing control is performed. The mathematical model of the fundamental components of the system, *i.e* the sensor and the drone, is discussed in the forthcoming subsections.

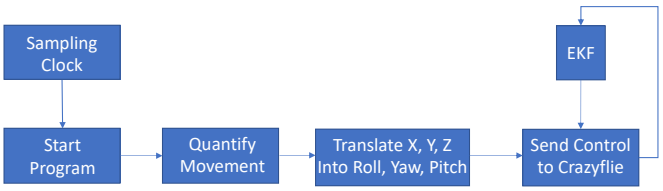


Fig. 1: System model detailing the main functional components of the sensor to drone system

A. Sensor

The Skywriter HAT near-field sensor operates by detecting fluctuations in a self-generated magnetic field by the introduction of conductive objects such as fingers, as can be seen in Fig. 2 [5]. From the Skywriter datasheet, the sensor has a sampling frequency of 200 Hz, a spatial resolution of 150 dpi, and a range of detection from 0 to 15 cm. However, from empirical testing, the range for detection seems to be consistently less than 3 cm in any direction from the center of

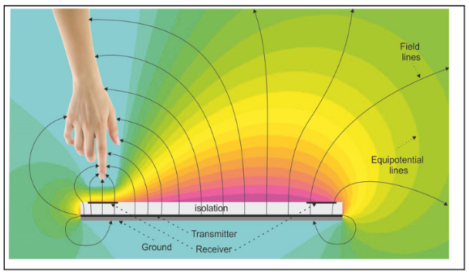


Fig. 2: Diagram describing how a conductive material interfaces with the Skywriter HAT

the device. Therefore, the dynamic model for the movement of the user's finger is based on the following assumptions:

- The range of detection is 2 cm from the center of the device in any direction
- The device operates consistently at 200 Hz
- The movement of the user's finger is less than

$$200 \text{ Hz} \times 4 \text{ cm} = 8 \ m/s$$

Due to the complexity of communicating with the Crazyflie drone, the rate at which we can send data is much lower than the rate at which the sensor samples. To deal with this and reduce effect of noise in the sensor, the Raspberry Pi will send processed data at a rate of approximately $10 \ Hz$. The sensor outputs information directly in X, Y, Z coordinates, so the instantaneous velocity of the users movement can modeled

$$v_x = \frac{\Delta d_x}{t} \quad v_y = \frac{\Delta d_y}{t} \quad v_z = \frac{\Delta d_z}{t}$$

where Δd in each direction is the most recent X, Y, Z measurement minus the value used for the previous transmission to the drone.

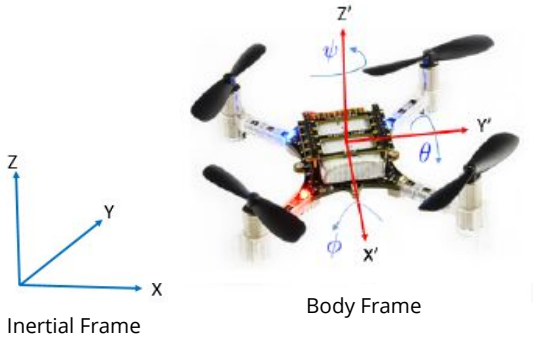


Fig. 3: Inertial frame and Body-fixed frame showing the Crazyflie.

B. The Drone

The dynamic equations of the quadcopter are made based on the following hypothesis [6]:

- The quadcopter is a rigid body that cannot be deformed, thus it is possible to use the well-known dynamic equations of a rigid body such as by using Euler-Newton approach.
- The quadcopter is perfectly symmetrical in its geometry, mass and propulsion system. Hence, the inertia matrix about this symmetry is diagonal.
- The quadcopter is time-invariant and the mass is constant.

According to Newton-Euler equations:

$$\begin{bmatrix} \mathbb{F}_b \\ \tau \end{bmatrix} = \begin{bmatrix} m & 0 \\ 0 & \mathbb{I} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \alpha \end{bmatrix} + \begin{bmatrix} \omega \times mv \\ \omega \times \mathbb{I}w \end{bmatrix} \tag{2}$$

where $\mathbb{F}_b$ is the body total force, τ is the total torque, m is mass, $\mathbb{I}$ moment of inertia, $\mathbf{a}$ is linear acceleration, α is the angular acceleration, w is the angular velocity, and v is the linear velocity. $\mathbb{F}_b$ is the summation of forces caused by the rotation of the rotors along the z-axis, such that $\mathbb{F}_b = [0 \ 0 \ f]^T$ with superscript ^{T} denoting the transpose operation. Thus, $f = \sum_{i=1}^{4} f_i$ given that $f_i = c_T w_i^2$ with c_T is the proportionality constant. Similarly, the torque can be expressed as $\tau_i = \pm c_Q w_i^2$ and a relationship between propeller speeds and generated thrusts and moments, due to body symmetry, can be defined as

$$\begin{bmatrix} f \\ \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ 0 & dc_T & 0 & -d_T \\ -dc_T & 0 & dc_T & 0 \\ -c_Q & c_Q & -c_Q & c_Q \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} \tag{3}$$

The idea is to translate Euler-Newton equations into the body frame by defining a rigid transformation matrix from the inertial frame to the body-fixed frame. In this case $\mathbb{F}_e = R\mathbb{F}_b - mg$, $\mathbb{F}_e$ is the inertial total force, R is the transformation matrix given in Eq. (1) and g is the gravity. Practically, due to air dynamics, the force generated by a propeller translating with respect to the free stream will typically be significantly different from the static thrust force f. This deviation is a function of the quadrocopter's relative airspeed.

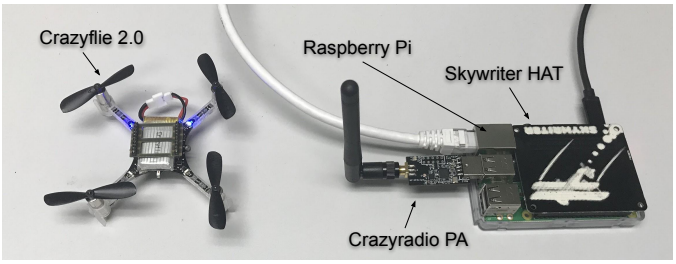


Fig. 4: Hardware setup with the Skywriter sensor connected directly to the GPIO pins of the raspberry pi.

C. Additional Sensors

In order to allow stable drone flight, we added an expansion board that contains two additional sensors, the VL53L0x time-of-flight (ToF) sensor and the optical flow sensor PMW3901. The ToF sensor is a laser ranging sensor that measures the distance of the drone from the ground. It contains a 940 nm invisible laser that can measure distances up to 4m at a maximum rate of 50 Hz. On the other hand, the optical flow sensor uses a low-resolution camera to measure movements in the x and y coordinates relative to the ground. The sensor

$$R = \begin{bmatrix} \cos\theta\cos\phi & \cos\theta\sin\phi & -\sin\theta \\ \sin\theta\sin\phi\cos\varphi - \cos\phi\sin\varphi & \sin\theta\sin\phi\sin\varphi + \cos\phi\cos\varphi & \sin\phi\cos\theta \\ \sin\theta\cos\phi\cos\varphi + \sin\phi\sin\varphi & \sin\theta\cos\phi\sin\varphi - \sin\phi\cos\varphi & \cos\theta\cos\phi \end{bmatrix} \quad (1)$$

requires a lens that enables the far-field tracking capability. It has a frame rate of 121 frames per second with a rate of 100 Hz. The term optical flow refers to a flow of two-dimensional images, in which certain features, such as patterns or pixel intensities, are tracked in time. The PMWB3901 requires SPI interface, while the VL53L0x requires I^2C connectivity, the PCB board handles the connectivity constraints and allows direct communication with the drone.

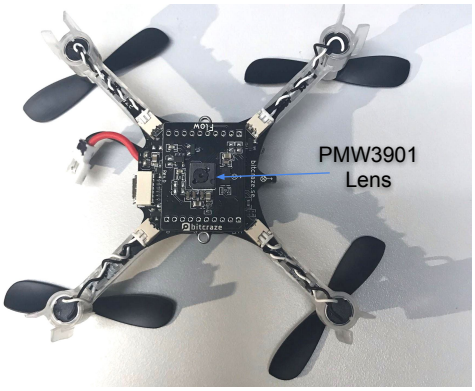


Fig. 5: The VL53L0x and PMW3901 sensors integrated within the same PCB mounted on the bottom of the Crazyflie.

III. Inner State Estimation

State estimation of the drone can be accomplished in many ways, however, a number of factors need to be taken into consideration when formulating realtime compliant and complexity constrained algorithms. The challenge lies in fusing the information arriving from different sensors at variable rates, and for this purpose, an EKF is considered. The Crazyflie has a pre-mounted inertial measurement unit (IMU) which includes a 3 axis gyro (MPU-9250), 3 axis accelerometer (MPU-9250), 3 axis magnetometer (MPU-9250), and a high precision pressure sensor (LPS25H). For this work, we used the readings from the gyroscope and accelerometer. The slowest rate at which the IMU sensor data is fetched is at $fs = 500$ Hz.

As previously mentioned, the component of the airspeed has an effect on the drone's flight. To account for the airspeed, a vector, f_a, is introduced in the Newton-Euler equations that captures the aerodynamic effects affecting the body-frame of the drone. Hence, the forces can be re-expressed as [3]

$$m\ddot{x} = R(f + f_a) + mg \quad (4)$$

where $\ddot{x}$ is the double derivative of the drone's position in an inertial reference frame. The measurement of the gyroscope can be modelled as

$$z_{gyro} = w + \eta_{gyro} \quad (5)$$

with η_{gyro} is assumed to be zero-mean additive white Gaussian noise (AWGN). The accelermometer measurements are also assumed to be corrupted by AWGN, η_{acc}, it can be expresssed as

$$z_{acc} = R^{-1}(\ddot{x} - g) + \eta_{acc} = \frac{1}{m}(f + f_a) + \eta_{acc} \quad (6)$$

The gyroscope measurements can be directly used as an estimate of the drone's angular velocity, while the accelerometer readings can be used to estimate the force of airspeed, f_a. However, as can be observed from Eq. (5) and (6), these readings are noisy. As a result, we incorporated the optical flow sensor and the ToF flight sensors, detailed in Section II.C, in order to improve the estimation reliability. Even though the drone dynamics are highly non-linear, insight can be gained from analysing the linearised system constituted by the error dynamics which makes using an EKF a viable solution.

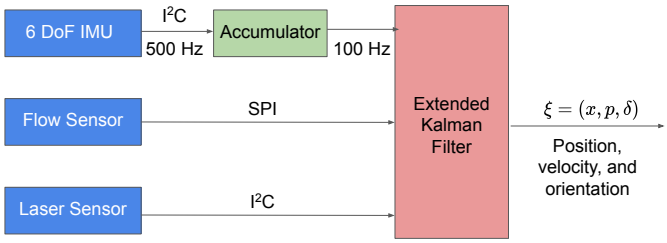


Fig. 6: The accumulator averages the last 5 IMU measurements, as the prediction loop is slower than the IMU loop. However, the IMU information is required externally at a higher rate for body rate control.

As the translational dynamics of the quadcopter is a triple integrator in essence, stability can only be guaranteed if the measurement equation contains information on the zeroth order states, that is translation and attitude. The IMU provides only second order derivative information which will cause the EKF to quickly diverge in a quadratic fashion for the positional states. Using the first order derivative information, obtained by the optical flow and ToF sensors, results in slower divergence as a linear drift in the positional estimates. In addition, the estimator contains a reference rotation matrix, $\hat{R}$, where all the orientations are expressed. The estimator aims to find the stochastic state ξ, such that $\xi = (x, p, \delta)$ denoting the drone's position, velocity, and orientation respectively. Figure 6 illustrates the utilized sensors and the EKF system block diagram. The IMU operates at the rate of 500 Hz, and the accumulator averages the samples such that the rate of the output is 100 Hz to match that of the other two sensors. For the flow sensor, a driver is used to written sample the accumulated pixel counts, rotate the accumulated pixel counts into the body frame, and run digital signal processing, including filtering, on

the measurements. Due to the compactness in size limitation, the rate at which the driver runs is limited to the previously mentioned 100 Hz.

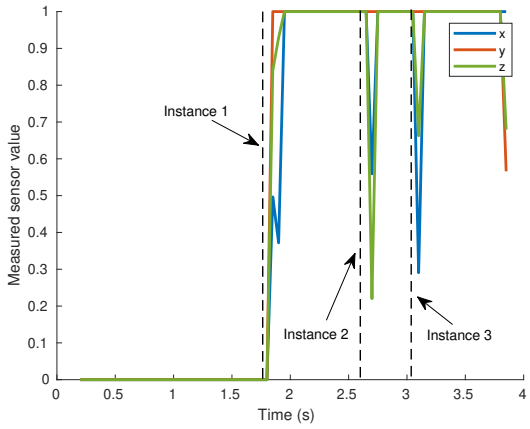
IV. Analysis

Exhaustive experiments are performed to verify the stability of the drone's flight and its responsiveness to precoded trajectories. With the aid of the closed loop control, the drone is tested for linear, ramp-like, and circular motions. A figure-8 trajectory is also performed and the drone landed repeatedly back to its point of origin. Because the drone works well and consistently when feedback through the optical sensor is introduced to the system, the focus for analysis is the behaviour of the sensor. There are three primary issues that are identified, each of these can be seen in 7, which displays the x, y and z coordinates of the user's hand as it moves over the Skywriter sensor in one of the three axis directions. The detection of motion is indicated by changes in the x, y and z lines, when the line is flat that entails that no change is detected and the coordinates are repeated. Each sub-figure displays three repetitions of the same motion in one primary direction. The first of the three issues is that in each instance there should have been only one axis with a major change, for illustration, in Fig.7(a) x should change from 0 to 1, in Fig.7(b) y should go from 0 to 1, and in Fig.7(c) z should go from 1 to 0. As can be seen in these graphs, rarely is there only one axis changing for each instance of motion and in some cases the wrong axis has much more movement than the axis that should be changing. Secondly, the sensor picks up radically different values even when the motion is as identical for humanly response. In some cases, no changes at all are detected such as in instance 1 in Fig.7(b). Lastly, the motion that is picked up by the sensor is very noisy, such as in instance 1 in Fig.7(a).

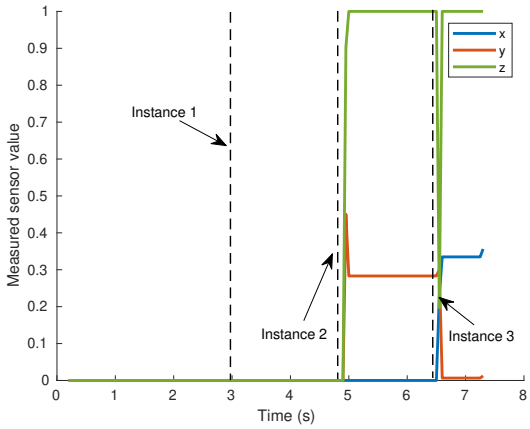
```
@skywriter.move()
def move(ax, ay, az):
    global x_sum, y_sum, z_sum, num
    # Sum up detected x, y, z values for averaging to reduce noise
    x_sum += ax
    y_sum += ay
    z_sum += az
    num += 1
    time.sleep(0.0001)
```

Fig. 9: Segment of code responsible for interfacing with Skywriter API and accumulating measured positions from user.

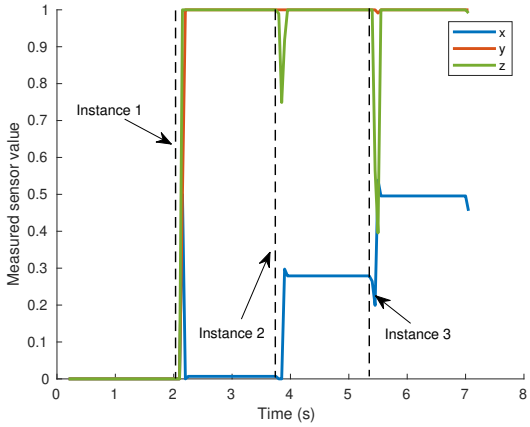
In order to overcome the impact of the noise, a modification in the code is made. This is done through averaging the samples collected between drone actuations, thus resulting in a minimal impact from noise but also a lower spatial resolution as can be seen throughout Fig.8. Measurements are accumulated at a maximum frequency of 10 kHz, as can be seen in Fig.9. These measurements are accumulated for at most 0.05 seconds before being taken in and averaged by the main code loop. Based on these values, there should



(a) Forward motion.

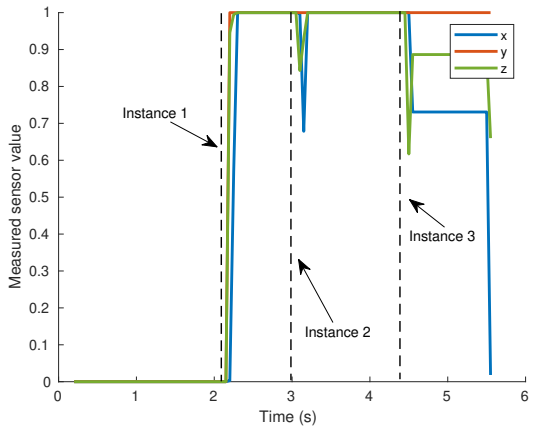


(b) Right motion.

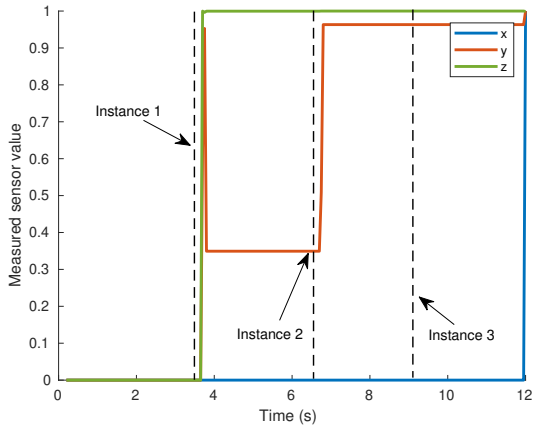


(c) Down motion.

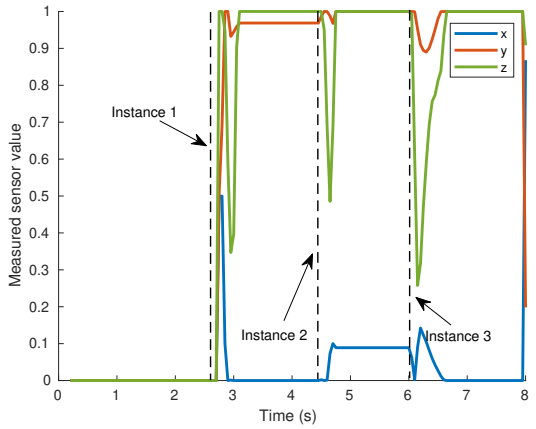
Fig. 7: Curves showing the non-averaged readings of the Skywriter sensor in the x, y, and z directions for three various actions, each was repeated three times marked as instances on the time axis.



(a) Forward motion.



(b) Right motion.



(c) Down motion.

Fig. 8: Curves showing the averaged readings of the Skywriter sensor in the x, y, and z directions for three various actions, each was repeated three times marked as instances on the time axis.

have been at most 500 samples between each loop. However, due to the Crazyflie's motion application program interface (API) implementation, there is additional time at which the program is stalled before the drone begins moving. Although this additional time varies based on the distance values fed into the function, the typical additional time is significantly less than 0.05 seconds, while the maximum theoretical time would be 0.5 seconds. Therefore, the typical number of samples accumulated each loop is well under 1000.

Because the sensor is relatively cheap, there is no calibration that can be done to adjust measurements for varying environments and because the user cannot access the raw electrode measurements there is little that can be done to improve its performance and sensitivity. By having no dynamic calibration, not only does changing the environment slightly have an impact on the sensor's output, the sensor also struggles with continued distortions of the surrounding electromagnetic emissions. This is exemplified when the user holds his/her hand in the exact same position near the sensor. After around a second, the measurements output by the device vary wildly without any motion from the user. This makes slow or precise movements almost impossible to track. All of these issues could have been minimized or avoided altogether by using another sensor such as an optical flow sensor, similar to the one used by the drone for stabilization. Crazyflie sells one of these sensors for $40 specifically for this kind of use. The relatively comparable price and potential increase in accuracy and performance make this kind of device a more suitable option for further studies.

V. Conclusion

In this work, a system that controls the motion of a drone based on users hand gestures was developed. The system consisted of a RaspberryPi, a near field sensor for hand motion detection, and a Crazyflie drone. In order to optimize the drone's flight and to provide a streamline operation, the drone's premounted sensors as well as additional sensors were used along with an EKF. Future work includes using a different hand motion sensor for a more seamless operation.

References

[1] Kathiravan Natarajan, Truong-Huy D. Nguyen, and Mutlu Mete. Hand Gesture Controlled Drones: An Open Source Library. *CoRR*, abs/1803.10344, 2018.

[2] Crazyflie 2.0. https://store.bitcraze.io/products/crazyflie-2-0.

[3] Mark W Mueller, Michael Hamer, and Raffaello D'Andrea. Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1730–1736, May 2015.

[4] Marcus Greiff. Modelling and control of the crazyflie quadrotor for aggressive and autonomous flight by optical flow driven state estimation, 2017. Master's Thesis.

[5] MGC3130 data sheet.

[6] Carlos Luis and Jerome Le Ny. Design of a Trajectory Tracking Controller for a Nanoquadcopter. *CoRR*, abs/1608.05786, 2016.