Towards Sensor Autonomy in Sub-Gram Flying Insect Robots: A Lightweight and Power-Efficient Avionics System

Yash P. Talwekar[1], Andrew Adie[1], Vikram Iyer[2], Sawyer B. Fuller[1,2]

***Abstract*— Flying insect robots weighing less than a gram (FIRs) have advantages over their larger counterparts due to their low materials cost, small size, and low weight, allowing for deployment in large numbers. Control autonomy in such aircraft introduces challenges arising from their small size such as high-speed dynamics, limited power and payload capacity. Previous work has produced and characterized sensors with compatible mass and power specifications, many of which are biologically-inspired. And controlled flight has been demonstrated using feedback from external motion capture cameras. But to date, no avionics system has been reported that is light enough and capable of providing the feedback necessary to perform controlled hovering flight using only components carried on-board. Here we present such a system. It consists a sensor package consisting of an inertial measurement unit, a laser rangefinder and an optical flow sensor, and an associated estimator based on the nonlinear Extended Kalman Filter (EKF). The sensor suite weighs 187 mg and consumes 21 mW. We implemented a low-latency wireless link to transmit this data at 1 kHz without cumbersome wires. The EKF estimates attitude, altitude and lateral velocities. We estimate that computation power usage is <400 µW using floating-point operations on a standard microcontroller. Our system's RMSE attitude and position error are less than 4°and 1 cm relative to motion capture estimates.**

I. Introduction

Because of their small size, flying insect-sized robots (FIRs) weighing less than a gram have the potential to outperform larger robots at tasks that include search and rescue operations, gas leak detection, and environment monitoring. Their advantages originate from a lower materials cost, allowing greater deployment numbers. Their small size also enables navigation in confined spaces, and around humans without impact hazard. Despite these advantages, such robots operating autonomously have not been realized because of the challenges of miniaturizing their actuators, mechanical systems, power system, and sensing and control systems.

We are concerned here with establishing the ability to hover in air without crashing. This requires "sensor autonomy," the first layer of autonomy in the hierarchy proposed in [1] and [2]. Stable hover is required before higher-level capabilities such as obstacle avoidance or navigation can be executed. The combination of small scale and the flapping-wing designs makes the governing dynamics inherently unstable for these small robots [3], [4]; therefore, to achieve a stable hovering flight, active stabilization of the attitude, altitude and lateral motion of the robot is necessary [5]. In nature, real insects rely heavily on optical flow perception for multiple tasks including navigation, speed regulation and collision avoidance [6], [7]. Various sensors of suitable mass and power usage have been explored for insect-sized robots. These include low-resolution cameras [8], [9], [10], [11], angular-rate-sensing ocelli [12] and gyroscopes [13], magnetometers [14], rangefinders [15], and wind sensors [16]. However, they have all been explored on an individual basis and have not been combined in such a way so that they could be used to estimate the state of a vehicle in flight.

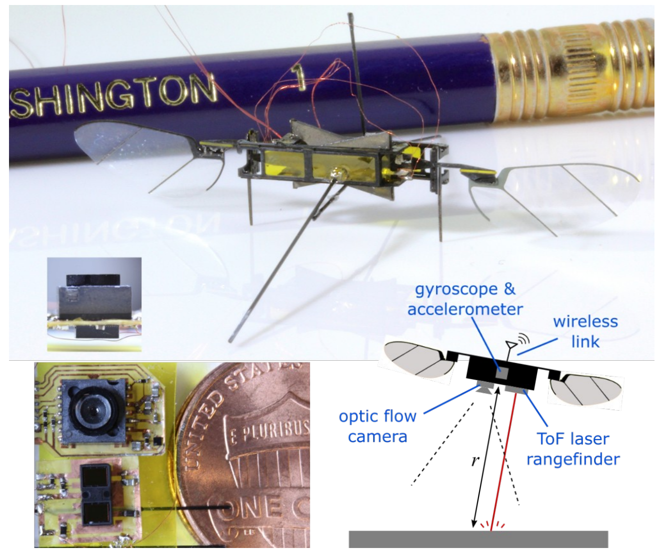

Fig. 1: The proposed miniaturized sensor suite, shown to scale with the University of Washington Robofly and a US penny. (bottom right) Diagram of sensor suite. A gyroscope measures angular velocity, an accelerometer senses inclination angle, a downward-facing rangefinder measures distance to ground r, and a downward-facing optic flow camera measures the rate of visual motion as the robot moves. An onboard radio transmits telemetry data to a remote receiver with low latency. The bottom left image shows the rangefinder and optic flow camera; a side view (inset) shows an IMU on the opposite side of the flex circuit board.

Previous work has proposed various mechanical designs and control architectures for FIRs that are precise enough that they are able to perform controlled flight maneuvers using feedback from external motion capture cameras [17], [18], [19], [20]. Larger drones such as quad-rotor drones can hover stably using only on-board sensor feedback such as from inertial measurement units (IMUs) and the global positioning system (GPS). In many applications of interest, however, such as indoors and in "urban canyons", the GPS signal is disrupted or denied entirely. We take inspiration from a sensor suite that is successful for small drones for

*This research was partially supported by the National Science Foundation (award number FRR-2054850).

[1]Department of Mechanical Engineering, University of Washington, Seattle, WA 98195, USA.

[2]Paul G. Allen School of Computer Science, University of Washington, Seattle WA 98195, USA

yasht16@uw.edu

GPS-denied environments that consists of a downward-facing camera, an IMU, and a downward-oriented ultrasonic or laser rangefinder [21], [22].

In this paper, we present, to our knowledge, the first avionics system to be suitable in terms of mass and power usage for a sub-gram FIR to perform sensor-autonomous hovering flight (Fig. 1). Our sensor package and estimator algorithm is able to estimate attitude, altitude, and lateral velocity of the vehicle. Our contributions address four objectives: 1) arrive at the most optimal state estimate given different update rates of the different sensors, 2) minimize the computation power consumed, 3) minimize the total weight of the sensor suite, 4) transmit this data for online estimation or offline analysis without cumbersome wires. In the remainder of the paper we describe the sensors, dynamics and measurement model, compare state estimation approaches in terms of computation power, and validate our results on data collected from our sensor suite that was transmitted wirelessly to a desktop computer for analysis.

II. Dynamics

The equations describing the dynamics of any aircraft in flight follow the Euler-Lagrange equations of motion for a rigid body:

$$\begin{aligned} m\dot{\boldsymbol{v}} &= \Sigma \boldsymbol{f} \\ \boldsymbol{J}\dot{\boldsymbol{\omega}} &= \Sigma \boldsymbol{\tau} - \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega} \end{aligned} \tag{1}$$

where $\boldsymbol{f}$ is the force and $\boldsymbol{\tau}$ is the torque acting on the robot, m and $\boldsymbol{J}$ are the mass and moment of inertia, $\boldsymbol{v}$ is the velocity vector, and $\boldsymbol{\omega}$ is the angular velocity vector. The first equation is typically expressed in world coordinates and the second in body-attached coordinates.

Let θ_x and θ_y be the angular rotation (Euler Angles) of the robot about body-fixed x and y axes. The 3D dynamics in Eq. (1) of the robot can be decoupled into two independent 2D dynamics in the x-z and y-z planes if the robot attitude is restricted to the neighborhood of the upright position ($\theta_x \approx 0, \theta_y \approx 0$) and ω_z, the angular velocity component in the body z direction is small [16].

We consider motion in x-z plane; the y-z plane is only slightly different. We define a minimal state vector that is observable with our proposed sensor suite that provides enough information to attain both stable hovering flight as well as the ability to follow trajectories:

$$\boldsymbol{q} = \begin{bmatrix} \theta & v_x & z & v_z \end{bmatrix}^T, \tag{2}$$

where θ is the robot's angular rotation w.r.t. the body-fixed y-axis, v_x is the velocity along the world x-axis, and z and v_z are the position and velocity along global z-axis.

Controlling flight entails varying wing amplitude and offset to produce forces and torques [17]. As in other domains of control, we assume that the model of the actuator is uncertain and rely on our sensors to provide robustness to this uncertainty. Instead of feeding motor inputs into the estimator, we use the more precise gyroscope measurement itself as an "input." In addition to allowing us to test the estimator without knowing inputs, this reduces the number of states, reducing computation requirements. Thus, in the 2-D plane in consideration, we can write the dynamics as

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \omega & 0 & v_z & 0 \end{bmatrix}^T, \tag{3}$$

where ω is the angular velocity along the body-fixed y-axis. As justified above, we define the control input vector as

$$\boldsymbol{u} = \begin{bmatrix} \omega_m \end{bmatrix}, \tag{4}$$

where ω_m is the angular velocity measurement from the gyroscope. This allows us to write the dynamics in Eq. (3) as

$$\dot{\boldsymbol{q}} = f_c(\boldsymbol{q}, \boldsymbol{u}). \tag{5}$$

III. Sensor Suite

For the estimator design, we consider a suite of sensors mounted on the robot consisting of a laser rangefinder, an optical flow sensor, and an IMU that houses a gyroscope and an accelerometer. Table I summarizes the relevant specifications of these sensors. We assume the noise in each of the sensors to be a zero-mean additive, uncorrelated Gaussian white noise.

component	**size**	**mass**	**data rate**	**power**
IMU	2.5×3×0.91	14	1000	3
rangefinder	4.9×2.5×1.56	16	50	6
optical flow	5×5×3.08	97	100	12
discretes	–	40	–	–
board+solder	–	20	–	–
total	–	187	–	21

TABLE I: Sensor specifications. Units are mm, mg, Hz, and mW, respectively.

A. Inertial Measurement Unit

We selected the ICM-20600 (TDK Invensense, USA) as the IMU for our system because it is small (2.5×3×0.91 mm) and light (14 mg). This single package contains both a 3-axis gyroscope and a 3-axis accelerometer. Briefly, the gyroscope operates by measuring angular velocity by sensing Coriolis forces in an electromechanical resonator; the accelerometer senses deflections in a proof mass. Both support data rates of over 1 kHz over the I_2C communication protocol and have a programmable full-scale range. They were configured for a range of $\pm 250°/\mathrm{s}$ and ±2g respectively. The sensors are mounted close to the robot body's center of gravity to avoid the effects of centripetal accelerations.

The sensor measurements for ω, and accelerations a_x about the world x-axis and world a_z about the z-axis can be expressed as

$$\begin{aligned} \omega_m &= \omega + \nu_g \\ a_{xm} &= a_x + \nu_{ax} \\ a_{zm} &= a_z + \nu_{az}, \end{aligned}$$

where ν_g, ν_{ax} and ν_{az} are the additive noise terms.

B. Rangefinder

A laser rangefinder (also known as a time-of-flight sensor) emits laser pulses towards a surface and estimates the distance to it based upon the time taken by the pulse to reach back to the sensor after reflecting from the surface. We used the VL53L1X (STMicroelectronics), which comes in a small package of 4.9×2.5×1.56 mm weighing 16 mg, and supports a data rate of up to 50 Hz over the I_2C protocol. We mounted this sensor below the robot, facing the ground, to get a measurement of the robot altitude (in the robot's rotated reference frame). This measurement can be expressed as [23]

$$r_m = \frac{z}{\cos(\theta)} + \nu_r$$

where ν_r is an additive noise term.

C. Optical Flow

An optical flow sensor is typically a camera module which computes the rate of relative visual motion by comparing consecutive frames. We used the PAW3902JF-TXQT (PixArt Imaging) which comes in a package of 5×5×3.08 mm and weighs 97 mg. The sensor provides an accumulated pixel count, which we then convert to rad/s with a scaling factor, at a frame rate of 126 fps over the SPI communication protocol. This high rate allows us to sample data at 100 Hz. We estimated its latency to be approximately 2 ms, negligible compared to its update rate. To do so we found the maximum cross-correlation between its output read in from an SPI-to-USB adaptor, and the time-derivative of the voltage from a linear potentiometer to which it was attached (measured by NI-6000 USB DAQ).

As with the rangefinder, we mounted this sensor at the bottom of the sensor package facing the directly down in the negative z-direction. In addition to translational motion, the rotation of the robot also contributes to the optical flow measured by the sensor, and therefore we place it exactly below the IMU in order to accurately compensate for the rotational effects in the measurement model. The measurement equation for the optical flow measured along the body x-axis can be written as

$$\Omega_m = \frac{\cos(\theta)}{z}\left(v_x \cos(\theta) + v_z \sin(\theta)\right) - \omega + \nu_o$$

where ν_o is the additive noise term.

D. Fabrication

We fabricated three separate circuit boards for the sensors using thin copper-clad flex circuit material (DuPont Pyralux AC121200E, 12.5 µm copper, 12.5 µm polyimide) to minimize the total board weight. We first coated the copper with an ink mask and patterned the circuit traces using a UV diode-pumped solid-state (DPSS) laser machining system. The remaining copper was etched using ferric chloride to produce the final circuit. Components and 43-gauge copper wires for power, I_2C and SPI connections were manually soldered onto the circuit. Figure 1 shows the final assembly and Table I gives the weight break-down of the assembly and estimated power requirements taken from datasheets.

E. Data Acquisition

Our sensor suite communicates over two different protocols, I_2C and SPI. Adding power and ground, this requires providing a total of 8 signals to the robot. We observe that even thin (>50 AWG) wires cause significant disturbance on a fly-sized flying robot. Our sensor suite instead incorporates an onboard wireless microcontroller to transfer the sensor data. We selected the nRF52832 (Nordic Semiconductors) because it offers a small 3.0x3.2 mm wafer-level package and, in addition to 2.4 Ghz Bluetooth low-energy wireless communication, it provides a high-speed protocol known as Enhanced ShockBurst (ESB). We wrote firmware for the microcontroller that uses hardware timers to query each sensor at its corresponding time intervals. As soon as the data is fetched for any sensor, it is transmitted over ESB to another nRF52832 chip acting as a receiver and can communicate over UART (RS-232) to a Windows PC.

In order to evaluate the state estimates from our estimator we used a four-camera motion capture arena (Prime13, OptiTrack Inc., Salem, OR) operating at 240 Hz to provide ground-truth measurements for our estimator. We attached reflective markers to the sensor suite. We recorded incoming sensor data and motion capture outputs simultaneously on the PC, along with timestamps from its internal clock, using a Python script running in Cygwin. Post-processing in Python was done to time-align sensor measurements with motion capture estimates.

IV. ESTIMATOR DESIGN

For designing a useful yet computationally efficient estimator, we start by introducing the full Extended Kalman Filter (EKF), before exploring simplifications aimed at power reduction.

We discretize the dynamics in Eq. (5) as

$$\boldsymbol{q}_{k+1} = f(\boldsymbol{q}_k, \boldsymbol{u}_k, \boldsymbol{w}_k) = \boldsymbol{q}_k + \Delta t f_c(\boldsymbol{q}_k, \boldsymbol{u}_k) + \boldsymbol{G}\boldsymbol{w}_k, \quad (6)$$

where Δt is the time interval between subsequent estimator updates (which varies depending on communication latency or dropouts), $k \in \{0\} \cup \mathbb{Z}_+$ is the time index, and $\boldsymbol{w}_k$ is the input noise vector propagated through linear dynamics $\boldsymbol{G}$. We denote the covariance of $\boldsymbol{w}_k$ by $\boldsymbol{Q} = E[\boldsymbol{w}_k \boldsymbol{w}_k^T]$.

The complete measurement model including rangefinder, optic flow camera, and accelerometer, is given by

$$\begin{aligned} \boldsymbol{y}_k &= h(\boldsymbol{q}_k, \boldsymbol{u}_k, \boldsymbol{\nu}_k) \\ &= \begin{bmatrix} \frac{z}{\cos(\theta)} \\ \frac{\cos(\theta)}{z}\left(v_x \cos(\theta) + v_z \sin(\theta)\right) - \omega \\ -g\sin(\theta) \\ g\cos(\theta) \end{bmatrix} + \boldsymbol{\nu}_k. \end{aligned} \quad (7)$$

We denote the covariance of $\boldsymbol{\nu}_k$ by $\boldsymbol{R} = E[\boldsymbol{\nu}_k \boldsymbol{\nu}_k^T]$. We further assume that the measurement and process noise are uncorrelated, i.e, $E[\boldsymbol{w}_k \boldsymbol{\nu}_k^T] = 0$. We further define the

Jacobians

$$\boldsymbol{F}_k = \frac{\partial f(\boldsymbol{q}, \boldsymbol{u}, 0)}{\partial \boldsymbol{q}}|_{(\boldsymbol{q}_k, \boldsymbol{u}_k)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \qquad (8)$$
$$\boldsymbol{H}_k = \frac{\partial h(\boldsymbol{q}, \boldsymbol{u}, 0)}{\partial \boldsymbol{q}}|_{(\boldsymbol{q}_k, \boldsymbol{u}_k)}$$

The simplicity of the dynamics model given by Eq. (3) results in a time-invariant dynamics Jacobian $\boldsymbol{F}$. We performed an observability analysis at anticipated hover equilibrium, $\boldsymbol{q} = [0, 0, z_{op}, 0]^T$, in which the Jacobian for the measurement model (Eq. (8)) is given by

$$\boldsymbol{H}_k = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & \frac{1}{z_{op}} & 0 & 0 \\ -g & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

We used the `obsv` command in python-control [24] to compute the observability matrix. The rank of the observability matrix is 4, which satisfies the observability criterion [25] and indicates that all states are observable in a neighborhood of this point.

The standard discrete-time EKF [26] (Supplement: Optimization-based control) is a widely used state estimation technique for nonlinear systems. At each time instant, the estimator starts with a knowledge of the present state and inputs and predicts the state at the next time instant based on the system dynamics $f(\hat{\boldsymbol{q}}_k, \boldsymbol{u}_k, 0)$. The estimator then computes the Kalman gain $\boldsymbol{K}$ which is multiplied to the difference in the observed and predicted measurements to produce a correction which is added to the earlier predicted state to provide an updated estimate.

In practice, different sensors produce readings at different rates, requiring an alternate formulation of the EKF. In our estimator, an update is performed every time a new IMU measurement arrives. The polling of the IMU, and consequently the estimator calls, runs at about 1 kHz. Given the different data rates of the sensors, at each update, we keep track of which sensors in the measurement model are available, based on which, we modify the update step of the EKF as described in more detail below. Occasionally, data arrives from one or two of the other sensors but not from the IMU. We implemented a workaround in which this data is stored and then used in combination with the subsequent IMU reading. This imposes an occasional, small latency penalty that is relatively insignificant compared to the intermittency of non-IMU measurements.

To incorporate the effect of disparate sensor update rates, we consider the general case in which there are n measurements possible, but at a given instant only m are available.

A. Sequential Update

In this approach we start with a measurement noise covariance matrix $\tilde{\boldsymbol{R}}$ such that the standard deviations of all the sensors is ∞, or equivalently, $\tilde{\boldsymbol{R}}^{-1} = 0$. Computationally we implement this as $\tilde{\boldsymbol{R}} = \xi \boldsymbol{I}_{n \times n}$, where ξ is a very large finite number. Then we loop through all the sensors, and for each j-th sensor that is available, we set $\tilde{\boldsymbol{R}}_{j,j} = \boldsymbol{R}_{j,j}$, and proceed with the usual EKF update step of computing the gain matrix and applying the correction. This update procedure is repeated until all the available sensors are accounted for, and then we consider the final update to be the state estimate. This approach is outlined in Algorithm 1.

Algorithm 1: Sequential Update

Data: $\hat{\boldsymbol{q}}_k, \boldsymbol{u}_k, \boldsymbol{y}_k, \boldsymbol{F}_k, \boldsymbol{H}_k, \boldsymbol{P}_k$
$\hat{\boldsymbol{q}}^-_{k+1} \leftarrow f(\hat{\boldsymbol{q}}_k, \boldsymbol{u}_k, 0)$
$\boldsymbol{P}^-_{k+1} \leftarrow \boldsymbol{F}_k \boldsymbol{P}_k \boldsymbol{F}_k^T + \boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^T$
for $j \in \{1, \cdots, n\}$ **do**
  $\tilde{\boldsymbol{R}} \leftarrow \xi \boldsymbol{I}_{n \times n}$
  if *j-th sensor is available* **then**
    $\tilde{\boldsymbol{R}}_{j,j} \leftarrow \boldsymbol{R}_{j,j}$
    $\boldsymbol{K} \leftarrow \boldsymbol{P}^-_{k+1} \boldsymbol{H}_k^T \left(\boldsymbol{H}_k \boldsymbol{P}^-_{k+1} \boldsymbol{H}_k^T + \tilde{\boldsymbol{R}} \right)^{-1}$
    $\hat{\boldsymbol{q}}_{k+1} \leftarrow \hat{\boldsymbol{q}}^-_{k+1} + \boldsymbol{K} \left(\boldsymbol{y}_k - h(\hat{\boldsymbol{q}}^-_{k+1}, \boldsymbol{u}_k, 0) \right)$
    $\boldsymbol{P}_{k+1} \leftarrow (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}_k) \boldsymbol{P}^-_{k+1}$
  end
end

B. Truncate Measurement Model

Let $\mathbb{S}$ be the set of m integers representing the indices of the available sensors. In this approach, we truncate the measurement model to include only the elements for the sensors that are available, i.e, define a vector $\tilde{\boldsymbol{z}} = h(\hat{\boldsymbol{q}}^-_{k+1}, \boldsymbol{u}_k, 0)[j] \in \mathbb{R}^{m \times 1}, j \in \mathbb{S}$ of m rows from $h(\boldsymbol{q}_k, \boldsymbol{u}_k, 0)$ which correspond to the available sensors. This vector has a noise covariance $\tilde{\boldsymbol{R}} = \text{diag}\{\boldsymbol{R}_{j,j} \,|\, j \in \mathbb{S}\} \in \mathbb{R}^{m \times m}$. We similarly truncate the Jacobian $\boldsymbol{H}_k$ to $\tilde{\boldsymbol{H}}_k = \boldsymbol{H}_k[j\,, :]$ and the measurement vector $\boldsymbol{y}_k$ to $\tilde{\boldsymbol{y}}_k = \boldsymbol{y}_k[j]$, $j \in \mathbb{S}$. We then proceed with the update step similar to that in the standard discrete-time EKF with these modifications. This approach is outlined in Algorithm 2.

Algorithm 2: Truncate Measurement Model

Data: $\hat{\boldsymbol{q}}_k, \boldsymbol{u}_k, \boldsymbol{y}_k, \boldsymbol{F}_k, \boldsymbol{H}_k, \boldsymbol{P}_k$
$\hat{\boldsymbol{q}}^-_{k+1} \leftarrow f(\hat{\boldsymbol{q}}_k, \boldsymbol{u}_k, 0)$
$\boldsymbol{P}^-_{k+1} \leftarrow \boldsymbol{F}_k \boldsymbol{P}_k \boldsymbol{F}_k^T + \boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^T$
$\tilde{\boldsymbol{R}} \leftarrow \text{diag}\{\boldsymbol{R}_{j,j} \,|\, j \in \mathbb{S}\}$
$\tilde{\boldsymbol{H}} \leftarrow \boldsymbol{H}_k[j, :]$, $j \in \mathbb{S}$
$\tilde{\boldsymbol{y}} \leftarrow \boldsymbol{y}_k[j]$, $j \in \mathbb{S}$
$\tilde{\boldsymbol{z}} \leftarrow h(\hat{\boldsymbol{q}}^-_{k+1}, \boldsymbol{u}_k, 0)[j]$, $j \in \mathbb{S}$
$\boldsymbol{K} \leftarrow \boldsymbol{P}^-_{k+1} \tilde{\boldsymbol{H}}^T \left(\tilde{\boldsymbol{H}} \boldsymbol{P}^-_{k+1} \tilde{\boldsymbol{H}}^T + \tilde{\boldsymbol{R}} \right)^{-1}$
$\hat{\boldsymbol{q}}_{k+1} \leftarrow \hat{\boldsymbol{q}}^-_{k+1} + \boldsymbol{K} (\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{z}})$
$\boldsymbol{P}_{k+1} \leftarrow \left(\boldsymbol{I} - \boldsymbol{K} \tilde{\boldsymbol{H}} \right) \boldsymbol{P}^-_{k+1}$

V. COMPUTATIONAL LOAD

Table II lists the number of cycles required for the algorithms to compute the state estimate based on the number of available sensors, and estimated energy consumption. Each update step consists of multiple single-cycle operations such as multiplications, additions and subtractions, and multi-cycle operations such as divisions, and sine and cosine computations. On an ARM Cortex-M4 based microcontroller like the STM32F4 (ST Microelectronics), divisions on floating-point numbers take 14 cycles [27], and fast-approximations to sine and cosine take around 20 cycles [28]. While the computation of the h and $\boldsymbol{H}_k$ involves many calls to sine and cosine functions, in actual implementation we can reduce these calls by calculating both values once and storing them in variables which we can re-use wherever required. Thus, in the calculations presented in table II, we consider only single calls to both sine and cosine functions, amounting to approximately 40 cycles in each estimator run.

To get an estimate of power usage of the algorithms, we start by calculating the number of cycles required by each algorithm based on the number of sensor measurements available. Since the accelerometer and gyroscope are polled simultaneously, and the estimator is called only when a gyroscope reading is available, we are guaranteed to have at least two available readings in the measurement vector. We then calculate the anticipated number of calls to the estimator in each case. Considering an ideal synchronization of the sensors, based on the data rates for each sensor, we can assume that all four measurements are available roughly every 20 ms (50 occurrences per second); optical flow is available in addition to the IMU every 10 ms thus, after removing the former case, we have three measurements available roughly every 20 ms (50 occurrences per second). The remaining 900 occurrences in a second only have the IMU readings available, which corresponds to only two sensors being available.

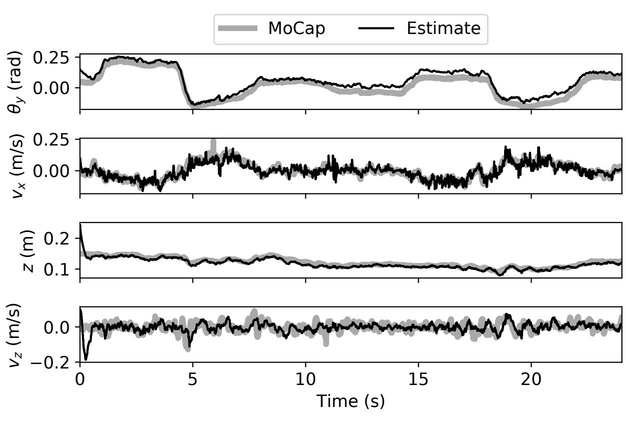

Fig. 2: Comparison of estimator performance against motion capture data

An STM32F4 operating at 2V and 89 µA current draws around 179 µW/MHz. Summing up the total cycles executed per second, we can compute the energy consumption for each algorithm. As shown in Table II, the Truncate Measurement Model approach (Algorithm 2) requires just over one-third of the power consumed by the Sequential Update approach (Algorithm 1).

VI. RESULTS

For analyzing the performance of the estimator, we mounted the sensor suite along with the microcontroller on a hand-held platform, and collected multiple sets of the sensors' data and the motion capture estimates for post-processing by manually moving the setup in the motion capture arena. Visual texture below the robot was a printed checkerboard pattern illuminated by LED light as well as the illumination from the motion capture cameras' infrared light sources. We estimated the sensor noise matrix to be

$$\boldsymbol{R} = \mathrm{diag}(0.007^2, 0.125^2, 0.5^2, 0.5^2).$$

We estimated the first two quantities by calculating the standard deviation of the error between ground-truth motion capture estimates and sensor readings for time-of-flight rangefinder and optic flow camera, respectively. For the camera, this was computed while translating the camera laterally at a constant, known height. The matrices $\boldsymbol{Q}$ and $\boldsymbol{G}$, which specify the size of disturbance noise and where it enters, respectively, are hard to measure. We took the perspective that these quantities should serve as tuning knobs to attain desirable performance. For $\boldsymbol{G}$, we assumed that the noise enters the system as white noise angular velocity, which affects θ, and white noise forces, which affect the translational velocities v_x and v_z, but not vertical position z. For the first input, the gyroscope's reading, we used a number that is higher than the datasheet ($0.004°\mathrm{s}^{-1}\mathrm{Hz}^{\frac{1}{2}}$, equivalent to a noise standard deviation of 0.007 at 1 kHz) for better performance.

$$\boldsymbol{Q} = \mathrm{diag}(0.15^2, 2^2, 2^2)$$
$$\boldsymbol{G} = \Delta t \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

We set the initial state of the estimator to be the state recorded by the motion capture system near that time instant, perturbed by +0.1 units in all four states to show the dynamics of estimator convergence. Figure 2 shows the comparison between the state estimates from the estimator using the Truncate Measurement Model approach against that of the motion capture system. Sequential Update provides a nearly identical estimate. It is evident from the comparison that the estimator is able to correct its tracking within 0.5 s. We further calculate the root-mean-squared error (RMSE) for each of the estimated states, tabulated in Table III. Since the comparison is drawn against the motion capture system, its accuracy also influences the RMSE values. We further observe that the estimator is able to maintain tracking performance well beyond 20 s, thus avoiding the gradual drift that affects estimates from dead-reckoning of IMU sensors. These results indicate the proposed sensor suite and the estimator design are capable of providing reliable state feedback for on-board control.

Algorithm	# Available sensors)	Occurrences per second	# Single-cycle operations per update	# Divisions per update (14 cycles)	# sin or cos operations per update (20 cycles)	Total cycles per update	Total cycles per second (MHz)	Power usage (µW)
Sequential update	2	900	1935	41	2	2549	5.394	965.38
	3	50	2695	57	2	3533		
	4	50	3455	73	2	4517		
Truncate measurement model	2	900	773	13	2	995	2.072	370.94
	3	50	956	18	2	1248		
	4	50	1175	25	2	1565		

TABLE II: Estimate of computational resources and power consumed by the algorithms on an STM32F4 microcontroller. For both the algorithms, we estimated the total number of cycles in each update by adding the number of single cycle operations, the number of divisions, and one call each to sine and cosine functions, and multiplying by the respective estimated number of cycles required. The last column shows estimated power usage per EKF (update in both x-z and y-z planes).

State	Exp 1	Exp 2	Exp 3	Exp 4
θ (rad)	0.027	0.042	0.068	0.031
v_x (m/s)	0.03	0.03	0.039	0.033
z (m)	0.007	0.007	0.01	0.01
v_z (m/s)	0.035	0.035	0.058	0.043

TABLE III: RMSE for the estimated states w.r.t to the motion capture estimates in four separate experiments

VII. CONCLUSIONS

In this work we presented a framework for on-board state estimation on sub-gram flying robots. We proposed a suite of sensors comprising of an optical flow sensor, a rangefinder, and a MEMS IMU. Using this sensor data we are able to formulate a model of the system that guarantees observability of attitude, altitude, and lateral and vertical velocities. The sensor-suite as a whole weighs less than 200 mg. Even with the addition of a microcontroller, it comes within the estimated 252 mg payload capacity of the 143 mg robot described in [18]. We also explored modifications to the standard EKF that are capable of handling varying sensor availability. We demonstrate satisfactory performance on data collected from physical sensors. The estimator is able to converge to the true state within 0.5 s, does not drift, and was able to maintain close tracking for over 20 s.

We also estimated the computational and power resources required by our sensing package. We estimate that sensor power usage is approximately 21 mW (Table I). Anticipated power usage by the microcontroller for EKF computation, using the more efficient Truncate Measurement Update, is negligible in comparison at 370 µW. We anticipate the power required to fly for an enlarged Robofly capable of carrying a 100 mg power system [29], 100 mg battery [30], 200 mg flapping mechanism, and our 200 mg sensor suite will be approximately 1.2 W after accounting for a 50% boost converter efficiency [31], [32]. A reasonable target for sensor suite efficiency is 10% of flight power: flight time is impacted as sensing power increases. Important examples of autonomous drones, e.g. the 1.5 kg system in [33] and the 30 g system in [34] hold to this. Our proposed avionics package power therefore falls well within this target.

It is worth remarking that our sensor suite does not provide estimates for x, y, or θ_z (the position and heading angle of the robot). In practice this means that these three quantities will slowly drift. For many applications, velocity control is sufficient. One important example is source seeking. In [35] our group showed that it is possible to use passive fins to steer into the wind during plume source seeking. The cast-and-surge algorithm is entirely specified in terms of velocities in the wind-aligned coordinate frame. If needed, drift could be mitigated by using the optic flow camera to intermittently take snapshot images. By computing the direction of deviations from an "initial condition" image, the robot can be brought into registration ("visual servoing" [36]). Because drift rate is low, this could be performed very intermittently, perhaps at 10 Hz, and still maintain reasonable performance without much more computational load.

The results presented in this paper are an important step toward on-board feedback control. By implementing a wireless connection using a tiny microcontroller, we have paved the way for future work in which state estimation is performed on-board the robot and then either transmitted wirelessly to an off-board computer for control. While the proposed system dynamics work well for a hand-held platform, to have a similarly robust estimation in free flight may require a slightly different dynamics model that accounts for the disturbance due to aerodynamic drag. An additional element is to include the effect that lateral velocities are influenced by gravity due a coupling with the attitude. Future work will validate our estimator on a freely-flying aerial platform by explicitly introducing these effects in the system model. We will also address any unexpected sensor non-idealities that occur in flight, such as distortion from vibration induced by flapping wings. Though initial results for the rangefinder [15] and gyroscope [13] suggest that such effects are likely to be minimal. Eventually, both estimation and control will happen on-board, with the wireless link used only for telemetry.

All of the components reported here have undergone mass reductions of 25–50% in the past few years due to miniaturization pressure from the consumer electronics industry, and we anticipate that this trend will continue. We used a 100 mg optic flow camera for simplicity, but much lighter cameras weighing 24 mg or less are possible [9], [10]. In the longer term, we foresee eventual mass production of robot flies in which avionics and power systems, including custom application-specific logic (ASIC) [37], are combined into just a few silicon parts. This will facilitate substantial further reductions in mass and power.

ACKNOWLEDGMENT

The authors wish to thank Yogesh Chukewad and Daksh Dhingra for insightful discussions.

REFERENCES

[1] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[2] R. St. Pierre and S. Bergbreiter, "Toward autonomy in sub-gram terrestrial robots," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 231–252, 2019.

[3] I. Paprotny and S. Bergbreiter, "Small-scale robotics: An introduction," in *workshop at the IEEE International Conference on Robotics and Automation*, pp. 1–15, Springer, 2013.

[4] J. J. Abbott, Z. Nagy, F. Beyeler, and B. J. Nelson, "Robotics in the small, part i: microbotics," *IEEE Robotics & Automation Magazine*, vol. 14, no. 2, pp. 92–103, 2007.

[5] G. K. Taylor and H. G. Krapp, "Sensory systems and flight stability: What do insects measure and why?," in *Insect Mechanics and Control* (J. Casas and S. Simpson, eds.), vol. 34 of *Advances in Insect Physiology*, pp. 231–316, New York: Academic Press, 2007.

[6] N. Franceschini, F. Ruffier, J. Serres, and S. Viollet, *Aerial Vehicles*, ch. Optic flow based visual guidance: from flying insects to miniature aerial vehicles, pp. 747–770. Vienna, Austria: In-tech Publishing, 2009.

[7] J. R. Serres and F. Ruffier, "Optic flow-based collision-free strategies: From insects to robots," *Arthropod structure & development*, vol. 46, no. 5, pp. 703–717, 2017.

[8] P.-E. Duhamel, N. Pérez-Arancibia, G. Barrows, and R. Wood, "Biologically inspired optical-flow sensing for altitude control of flapping-wing microrobots," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 2, pp. 556–568, 2013.

[9] S. Balasubramanian, Y. M. Chukewad, J. M. James, G. L. Barrows, and S. B. Fuller, "An insect-sized robot that uses a custom-built onboard camera and a neural network to classify and respond to visual input," in *IEEE Int. Conf. Biomedical Rob. and Biomechatronics (BIOROB)*, pp. 1297–1302, 2018.

[10] V. Iyer, A. Najafi, J. James, S. Fuller, and S. Gollakota, "Wireless steerable vision for live insects and insect-scale robots," *Science Robotics*, vol. 5, no. 44, 2020.

[11] S. Mange, E. F. Helbling, N. Gravish, and R. J. Wood, "An actuated gaze stabilization platform for a flapping-wing microrobot," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5409–5414, IEEE, 2017.

[12] S. B. Fuller, M. Karpelson, A. Censi, K. Y. Ma, and R. J. Wood, "Controlling free flight of a robotic fly using an onboard vision sensor inspired by insect ocelli," *J. Royal Society Interface*, vol. 11, August 2014.

[13] S. B. Fuller, E. F. Helbling, P. Chirarattananon, and R. J. Wood, "Using a MEMS gyroscope to stabilize the attitude of a fly-sized hovering robots," in *IMAV 2014: Int. Micro Air Vehicle Conf. and Competition 2014*, (Delft, The Netherlands), August 12-15 2014.

[14] E. F. Helbling, S. B. Fuller, and R. J. Wood, "Pitch and yaw control of a robotic insect using an onboard magnetometer," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5516–5522, IEEE, 2014.

[15] E. F. Helbling, S. B. Fuller, and R. J. Wood, "Altitude estimation and control of an insect-scale robot with an onboard proximity sensor," in *Robotics Research* (A. Bicchi and W. Burgard, eds.), pp. 57–69, Cham: Springer International Publishing, 2018.

[16] S. B. Fuller, A. Sands, A. Haggerty, M. Karpelson, and R. J. Wood, "Estimating attitude and wind velocity using biomimetic sensors on a microrobotic bee," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1374–1380, IEEE, 2013.

[17] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood, "Controlled flight of a biologically inspired, insect-scale robot," *Science*, vol. 340, no. 6132, pp. 603–607, 2013.

[18] S. B. Fuller, "Four wings: An insect-sized aerial robot with steering ability and payload capacity for autonomy," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 570–577, 2019.

[19] D. Dhingra, Y. M. Chukewad, and S. Fuller, "A device for rapid, automated trimming of insect-sized flying robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1373–1380, 2020.

[20] Y. M. Chukewad and S. B. Fuller, "Yaw control of a hovering flapping-wing aerial vehicle with a passive wing hinge," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1864–1871, 2021.

[21] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1736–1741, IEEE, 2013.

[22] M. W. Mueller, M. Hamer, and R. 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)*, pp. 1730–1736, May 2015.

[23] J. J. Koenderink and A. J. van Doorn, "Facts on optic flow," *Biological Cybernetics*, vol. 56, pp. 247–254, 1987.

[24] S. B. Fuller, B. Greiner, J. Moore, R. M. Murray, R. van Paasen, and R. Yorke, "The python control systems library (python-control)," in *IEEE Conf. Decision and Control (CDC)*.

[25] B. Friedland, *Control system design: an introduction to state-space methods*. Courier Corporation, 2012.

[26] K. J. Astrom and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. 41 William Street, Princeton, NJ 08540: Princeton University Press, 2008.

[27] ARM, "Cortex-M4 Technical Reference Manual." `https://developer.arm.com/documentation/ddi0439/b/BEHJADED`, 2010.

[28] ARM-Software, "CMSIS Version 5 github repository." `https://github.com/ARM-software/CMSIS_5/tree/develop/CMSIS/DSP/Source/FastMathFunctions`, 2021.

[29] J. M. James, V. Iyer, Y. M. Chukewad, S. Gollakota, and S. B. Fuller, "Liftoff of a 190 mg laser-powered aerial vehicle: The lightest wireless robot to fly," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3587–3594, IEEE, 2018.

[30] M. Duduta, S. de Rivaz, D. R. Clarke, and R. J. Wood, "Ultra-lightweight, high power density lithium-ion batteries," *Batteries & Supercaps*, vol. 1, no. 4, pp. 131–134, 2018.

[31] N. T. Jafferis, E. F. Helbling, M. Karpelson, and R. J. Wood, "Untethered flight of an insect-sized flapping-wing microscale aerial vehicle," *Nature*, vol. 570, pp. 491–495, 2019.

[32] J. M. James and S. B. Fuller, "A high-voltage power electronics unit for flying insect robots that can modulate wing thrust," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7212–7218, IEEE, 2021.

[33] H. D. Escobar-Alvarez, N. Johnson, T. Hebble, K. Klingebiel, S. A. P. Quintero, J. Regenstein, and N. A. Browning, "R-advance: Rapid adaptive prediction for vision-based autonomous navigation, control, and evasion," *Journal of Field Robotics*, vol. 35, no. 1, pp. 91–100, 2018.

[34] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64mw dnn-based visual navigation engine for autonomous nano-drones," *IEEE Internet of Things Journal*, 2019.

[35] M. J. Anderson, J. G. Sullivan, T. Horiuchi, S. B. Fuller, and T. L. Daniel, "A bio-hybrid odor-guided autonomous palm-sized air vehicle," *Bioinspiration and Biomimetics*, vol. 16, no. 2, p. 026002, 2020.

[36] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651–670, Oct 1996.

[37] X. Zhang, M. Lok, T. Tong, S. K. Lee, B. Reagen, S. Chaput, P. E. J. Duhamel, R. J. Wood, D. Brooks, and G. Y. Wei, "A fully integrated battery-powered system-on-chip in 40-nm cmos for closed-loop control of insect-scale pico-aerial vehicle," *IEEE Journal of Solid-State Circuits*, vol. PP, no. 99, pp. 1–14, 2017.