

©Copyright 2025

Zhitao Yu

Vision and Control for Insect-scale Robots

Zhitao Yu

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2025

Reading Committee:
Sawyer B. Fuller, Chair

Xu Chen

Ashis Banerjee

Program Authorized to Offer Degree:
Mechanical Engineering

University of Washington

Abstract

Vision and Control for Insect-scale Robots

Zhitao Yu

Chair of the Supervisory Committee:

Sawyer B. Fuller

Mechanical Engineering

This dissertation focuses on advancing vision and control for insect-scale robots, tackling two critical challenges: hovering and navigation. For hovering, the first study, presented in the TinySense paper (Best Student Paper Award at ICRA 2025) [1], develops a lightweight, energy-efficient avionics system, integrating a pressure sensor and a tiny global shutter camera for state estimation. Validated on the Crazyflie drone, this system demonstrates robust hovering state estimation performance comparable to the Crazyflie built-in state estimator and the ground truth captured by Mocap (Motion capture) system. The second study introduces a gyroscope-free, visual-inertial flight control system for 10 mg robots that combines an accelerometer and tiny optic flow camera for effective wind rejection, inspired by biological sensory fusion in fruit flies. This approach, detailed in our Science Robotics paper [2], significantly reduces weight and power consumption, enabling stable flight even under wind disturbances. For navigation, we first introduce a biologically inspired, bilinear optic flow approximation method, suitable for confined-space navigation in environments where traditional mapping techniques like SLAM are infeasible due to speed, size, weight, and power (SSWaP) constraints. This method leverages learned optic flow patterns for heading stability within a corridor, as outlined in our IROS paper [3], demonstrating confined-space navigation without reliance on a pre-mapped environment. Building on this foundation, we next leverage the emergence of embedded event cameras—whose characteristics align well

with the sensing constraints of insect-scale robots—to tackle the challenge of navigation in confined spaces. We work on developing a deep reinforcement learning framework that enables vision-based navigation using onboard event-based vision data alone. This approach has potential to achieve autonomous flight for insect-scale robots in confined spaces without external localization or mapping. Together, these contributions advance the potential for fully autonomous, insect-scale robotic platforms operating in real-world conditions.

TABLE OF CONTENTS

	Page
List of Figures	iii
Chapter 1: Introduction	1
1.1 Insect-scale Robots	1
1.2 Vision for Insect-scale Robots	2
1.3 Control for Insect-scale Robots	2
1.4 Tiny Machine Learning (TinyML)	3
1.5 Overview of This Thesis	4
Chapter 2: Tiny Sense: A Lighter Weight and More Power-efficient Avionics System for Flying Insect-scale Robots.	7
2.1 Introduction	7
2.2 Dynamics	9
2.3 Sensor Suite and Measurements	11
2.4 State Estimation	15
2.5 Experiments	17
2.6 Conclusion	21
Chapter 3: A gyroscope-free visual-inertial flight control and wind sensing system for 10 mg robots	23
3.1 Introduction	23
3.2 Method	26
3.3 Result	30
3.4 Discussion	48
Chapter 4: Visual Confined-Space Navigation Using an Efficient Learned Bilinear Optic Flow Approximation for Insect-scale Robots	58
4.1 Introduction	58

4.2	Methodology	62
4.3	Experiments	66
4.4	Conclusion	73
Chapter 5:	Towards Deep Reinforcement Learning for Onboard Confined-Space Navigation of Insect-Scale Robots Using Monocular Embedded Event-Based Vision	77
5.1	Introduction	77
5.2	Methodology	79
5.3	Hardware Platform	85
5.4	Experiments	91
5.5	Conclusion and Future work	94
Chapter 6:	Conclusion and Future work	96

LIST OF FIGURES

Figure Number		Page
2.1	The TinySense sensor suite shown next to a U. Washington Robofly and a standard pencil for scale. The width of the sensor suite board is approximately 12 mm and its mass is 78.4 mg.	8
2.2	Forces and state variables of a small hovering Flying Insect Robot (FIR).	10
2.3	Sensor suite (PAG7920LT optical sensor, ICM42688-P IMU, BMP388 pressure sensor) next to US 1 cent coin.	12
2.4	Sensor measurements from the onboard gyroscope, optic flow sensor, and pressure sensor of TinySense are compared with those from the Crazyflie for three different flight experiments (a), (b), and (c); trajectories are described in the main text and corresponding state estimates in Fig. 2.5. In the altitude plots, altitude is measured by the Crazyflie's laser rangefinder, while the TinySense uses a pressure sensor; mocap data is also included for comparison.	18
2.5	Comparison between state estimates from TinySense's Kalman Filter, Crazyflie, and mocap. We conducted three flight experiments (a), (b), and (c), with the Kalman filter estimates starting at time zero. Pressure sensor measurements were ignored during the first 1.8 s because pressure changes due to the ground effect at takeoff confound readings.	20
3.1	The sensor suite and size of the conceptual Nature-inspired Aerial Ten-milligram "NAT" robot compared to other small aircraft. (A) close-up view of the conceptual 10 mg robot and package sizes of its sensor suite made with off-the-shelf parts. Shown approximately to scale on the page are the (B) the NAT robot and its sensors along with the 1 mg fruit fly, (C) the 143 mg U. Washington Robofly [4, 5] with a pencil for scale, and (D) a palm-sized 30 g quad-rotor, the lightest vehicle yet to perform sensor-autonomous hover. Here it is shown equipped with the moth-based odor sensor from [6]. (E) Our proposed hovering sensors are much smaller and more power efficient than those used for autonomous hover in the palm drone (sensor package sizes also shown approximately to scale on the page).	24

3.2	Dynamics and flight control architecture overview. (A) Diagram of parameters that describe the dynamics of our model for a small hovering robot. (B) The proposed feedback control architecture is LQG (Linear Quadratic Gaussian), consisting of a Linear Quadratic Regulator for a controller and a Kalman Filter estimator. The sensor suite includes an accelerometer-based drag sensor that measures airspeed v_{am} and a camera that provides a measurement Ω_m of the rate of optic flow. To estimate the states x , y , and z , the estimates of v_x , v_y , and v_z , respectively, are numerically integrated.	32
3.3	Scaling laws in accelerometer-mediated wind sensing reduce the airspeed measurement noise as size reduces. (A) The ratio of damping to mass varies approximately with the inverse square of mass, with a least log-error exponential fit giving $\frac{b}{m} = 0.15m^{-1.9}$. (B) This results in reduced airspeed measurement noise σ_a as size diminishes.	41
3.4	Simulation results. (A) Flight path of the accelerometer-only sensor suite along the x and z dimensions. The aircraft performs a lateral maneuver starting at the lower right to a setpoint location (red cross mark) on the upper left. The true trajectory is shown in black; the state estimate (grey) is subject to an error in the initial estimated position, but is able to nonetheless maintain attitude stability. Lines show the path of the center of mass and boxes depict the vehicle's approximate size and attitude at 200 ms intervals. (B) Diagram of the camera model in the accelerometer-optic flow (AO) sensor suite: luminance readings $l(\gamma)$ are collected uniformly-spaced angles γ (red lines), which intersect a texture projected on the ground below at a position $\tilde{x} = x - z \tan(\gamma + \theta)$. (C) The correlator method produces similar output to Lucas-Kanade-based optic flow estimation, when following the same (arbitrary) trajectory. (D) The simulated AO system, when using autocorrelation-based optic flow estimation, is able to sense and react a wind disturbance. A step change in wind speed causes the aircraft to initially move in the direction of the wind. After about a second, the wind estimate has converged on its true value; the controller responds by commanding the robot to incline itself, resulting in a nearly perfect rejection of the disturbance.	42
3.5	Flight tests on a 30 g palm drone show that our accelerometer-vision system can estimate vehicle state and wind speed. (A) We used the accelerometer alone (no Kalman Filter) to measure airspeed v_a in still air ($v_w = 0$), providing an estimate of lateral velocity $v = -v_a$ in both the x and y directions during a rapid lateral maneuver. Estimates are very close to the helicopter's full GRO-based state estimate. (B) The full accelerometer-optic flow (AO) sensor suite and Kalman Filter estimator are able to estimate the wind velocity induced by a fan with reasonable performance, as well as the aircraft's attitude. . . .	45

3.6	Our simulated accelerometer-optic flow (AO) 10 mg aircraft with, when operated in closed-loop (blue), has a velocity response to an impulsive wind gust that closely resembles that of fruit flies. For this simulation, we set $\frac{b}{m}$ to equal to the 10.0 s^{-1} measured in 1 mg fruit flies. Compared to a passive response of a particle suspended in air (grey), both our system (blue) and the fly (red) have similar dynamics, exhibiting a larger impulse response followed a delayed compensatory overshoot beginning after about 0.2 s. The red line is the mean velocity response of fruit flies flying along length of a wind tunnel ($n = 92$, from [7])	47
3.7	Power scaling per pixel for a commercial imaging sensor family (Himax HM series, Tainan City, Taiwan).	55
3.8	States, measurements, and controller outputs (where a_z is z -axis acceleration $\tilde{f}_t$ and α_y is the y -axis angular acceleration $\tilde{\tau}_y$) during response to step input gust. (Figure 3.4D) Units for the state are rad, rad/s, m, m/s, m, m/s, and m/s, respectively.	56
3.9	States, measurements, and controller outputs (where a_z is z -axis acceleration $\tilde{f}_t$ and α_y is the y -axis angular acceleration $\tilde{\tau}_y$) during simulated response to impulsive gust (Figure 3.6). Units for the state are rad, rad/s, m, m/s, m, m/s, and m/s, respectively.	57
4.1	The 143 mg U. Washington Robofly (pencil in background for scale) weighs a bit more than a honeybee (101 mg). We aim to perform visual flight control with the very low compute power available on robots of this size and below.	59
4.2	A diagram of the robot in its environment (top view). The robot is assumed to move continuously in the positive x -direction ($\dot{x} > 0$). The red lines indicate directions of pixel readings taken by its omnidirectional camera. These are comparable to the ray casting algorithm used in computer graphics nomenclature. The solid dashed line is the heading angle ϕ of the robot. $\tilde{x}_1$ is the position of the intersection for the left wall, $\phi + \gamma < \pi$ and $\tilde{x}_2$ is the position of the intersection for the right wall, $2\pi > \phi + \gamma > \pi$	67
4.3	Validation of the camera model. We have $\text{RHS} = (\mathbf{s} \times \nabla_s \mathbf{l}) \cdot \boldsymbol{\omega} + \mu(\mathbf{s}, \mathbf{p}) \nabla_s \mathbf{l} \cdot \mathbf{v}$. And we have $\text{LHS} = \dot{l}$. We can see the RHS can follow LHS well.	70
4.4	The learned M_ϕ matrix. According to the derivation in [8], it should be a skew-symmetric matrix. We can see our simulation matches the derivation.	72
4.5	ϕ vs time. There are six trajectories with different initial conditions shown in the right bar.	73

4.6	ϕ and $\hat{\phi}$ for the trajectory with initial condition $\phi_0 = 5^\circ$. The left figure is plotting ϕ and raw $\hat{\phi}$. The right figure is plotting ϕ and Gaussian filtered $\hat{\phi}$ to remove spurious noise.	74
4.7	ϕ and $\hat{\phi}$ for the trajectory with initial condition $\phi_0 = 15^\circ$. The left figure is plotting ϕ and raw $\hat{\phi}$. The right figure is plotting ϕ and Gaussian filtered $\hat{\phi}$ to remove spurious noise.	74
4.8	An example trajectory of the simulated robot between the corridor walls shows that it is able to align itself with the long axis of the corridor.	75
5.1	Training in simulation and onboard inference on the robot platform. During training, BEMs are computed from consecutive grayscale frames and fed to the policy network. During deployment, BEMs are computed from the GenX320 event camera and fed to the same policy running on the OpenMV Cam RT1062 microcontroller.	80
5.2	Environment grid view in NVIDIA Isaac Sim	83
5.3	Crazyflie’s onboard camera view in the checkerboard corridor, simulated in NVIDIA Isaac Sim.	83
5.4	The yaw-rate step response from real Crazyflie.	84
5.5	The yaw-rate step response in simulation.	84
5.6	Integrated robot platform next to a pencil.	86
5.7	Architectural flow of the Crazyflie firmware.	87
5.8	Obstacle avoidance using the onboard GenX320 event-camera module.	92
5.9	Crazyflie trajectory (red dotted) in four corridor geometries. Left: straight (5 m). Right, top to bottom: 15° , 30° , and 60° bends. The bent corridors are composed of two 1.22m straight sections connected at the midpoint.	93
5.10	CPU load breakdown.	94

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Sawyer Fuller, for his invaluable mentorship and guidance throughout my PhD journey. His support not only shaped my technical growth but also influenced my attitude to research and my perspective on life and values.

I am also sincerely grateful to my supervisory committee members—Dr. Xu Chen, Dr. Ashis Banerjee, Dr. Blake Hannaford, and Dr. Dieter Fox—for their thoughtful insights, feedback, and support.

During my PhD, I pursued multiple concurrent degrees and served as reasearch/teaching assistant across several UW departments and devisions. I am grateful to the faculty, staff, classmates, and students from the Paul G. Allen School of Computer Science & Engineering, the Departments of Mechanical Engineering, Statistics, Applied Mathematics, Electrical & Computer Engineering and Bioengineering, the Foster School of Business, the Global Innovation Exchange, and the Engineering Teaching & Learning office in the College of Engineering for their support and inspiration.

My industrial internships significantly broadened my perspective. I am grateful to my colleagues at Amazon Robotics, Google, and Schlumberger for their collaboration and for the valuable experiences I gained during my time there.

A highlight of my PhD was our TinySense paper receiving the Best Student Paper Award at ICRA 2025. I thank all of my collaborators and the award committee for this recognition, which has been a tremendous motivation in my robotics research career. I also extend my appreciation to the UW Robotics community for enriching my experience and fostering a vibrant environment for research and innovation.

On a personal note, I deeply thank my family. I have not been able to return home for six years due to the Covid pandemic and other circumstances, and I miss them greatly. Despite the distance, their constant support has sustained me throughout this journey.

Finally, I am thankful to my friends, academic collaborators, and industrial colleagues for their encouragement, companionship, and support along the way.

Chapter 1

INTRODUCTION

1.1 *Insect-scale Robots*

Insect-scale robots could be fielded in the thousands or millions. Their small size enables operation in confined spaces and near people with minimal impact hazard, while energy harvesting supports persistent use. Potential applications include methane-leak localization in dense piping, early wildfire detection, and space exploration at substantially reduced launch cost. Representative platforms include Robofly [9], Robobee [10, 11], quad wing [12].

The principal advantage of small scale is reduced power consumption. Insect-scale robots could, in principle, operate indefinitely by harvesting ambient energy sources which are too small to power larger platforms. This approach could overcome the 30-minute endurance limit of battery-powered hovering drones. Indoors, Wi-Fi power is nearly ubiquitous and can be harvested without disrupting communications, supplying 0.02 mW [13]. By landing on floors or overhangs [14], a 75 mg robotic fly could recharge and fly intermittently. Assuming a flight power of 60 mW [15], this yields roughly 1 second of flight per hour. Solar photovoltaics likewise favor small scales: the high surface-area-to-volume ratio at the insect scale yields greater available power per unit mass. Other advantages of the insect scale include reduced cost, footprint, and mass. Lower material use decreases unit cost, enabling fleet-scale deployment for broad coverage. Small size enables traversal of tight, hard-to-reach areas and operate near people with minimal impact hazard. Low mass reduces launch costs, lowering the barrier to space deployment.

However, those advantages come with the challenges. The primary challenge with sensor-autonomous flight at insect scale are extreme speed, size, weight, and power (SSWaP) limitations. For a robot the size of a Robofly, the entire sensor suite must have a mass similar

to or less than a toothpick. The power used by a laden Robofly in flight is approximately 250 mW, suggesting a sensing and computation power budget drastically below this. Thus, we must employ lightweight sensors and efficient algorithms for insect robots operating in real-world environments.

1.2 Vision for Insect-scale Robots

Vision at the insect scale has largely focused on visual flight control and navigation tasks. From our knowledge, a fully integrated visual navigation system has not yet been demonstrated on a free-flying insect-scale robot; however, advances on small drones offer building blocks that have potential to solve problems at this scale.

Conroy et al. [16] employed Lucas–Kanade optical flow to navigate a 700 g quadrotor through a corridor, with power consuming about 300 mW. Escobar-Alvarez et al. [17] implemented an optic-flow-based method on a 10 W onboard computer to estimate time-to-contact (TTC) with obstacles. De Wagter et al. [18] presented a 20 g flapping-wing micro air vehicle (FWMAV) equipped with a 0.98 g autopilot and a 4.0 g onboard stereo-vision system that consumes about 300 mW. Zufferey et al. [19] presented a 10 g Microflyer that achieved vision-based indoor navigation using optical-flow measurements. Palossi et al. [20] demonstrated an onboard deep neural network that navigates a 30 g Crazyflie nano-quadrotor through hallways while avoiding obstacles. The network runs on an 8-core parallel processor at 6 fps, drawing about 64 mW—roughly 1% of the flight-motor power. Prior systems consume on the order of 100 mW or more for visual processing and carry gram-scale sensor suites, which is infeasible for insect-scale flight. For insect robots like Robofly, the sensor suite must weigh less than 150 mg and consume under 20 mW of power.

1.3 Control for Insect-scale Robots

For insect-scale robots, Chirarattananon et al. [21] used adaptive PID controllers to achieve state-of-the-art hover performance. Maneuvers beyond hover—such as perching [14] and somersaulting [22]—employ sliding-mode control with iterative learning of trajectory pa-

rameters. Recent work introduces modular optimal-control strategies, notably MPC architectures that pair a high-level MPC with a low-level torque controller [23]. Data-driven MPC, including Tube-MPC [24], shows potential in optimizing actuator constraints and trajectory tracking for complex maneuvers.

Recently, deep reinforcement learning (RL) has many breakthroughs in Robotics [25–28]. Deep RL is well suited to robotic systems with high-dimensional state or observation spaces and strongly nonlinear dynamics, enabling performance on tasks that conventional decision-making, planning, and control approaches (e.g., classical control, optimal control, sampling-based planning) do not handle well [29]. As far as we are aware, deep RL has not yet been demonstrated for insect-scale robots; nevertheless, recent results on small drones point to a promising route to deployment at this scale [30–32].

1.4 Tiny Machine Learning (TinyML)

Using deep RL in insect-scale robots is a very resource constrained problem, which requires TinyML techniques. Tiny Machine Learning (TinyML) refers to deploying machine-learning (and increasingly deep-learning) models on highly resource-constrained embedded devices—microcontrollers and small SoCs whose compute, memory, and power budgets are orders of magnitude below those of traditional computers. Conceptually, TinyML “migrates” classic ML/DL algorithms (e.g., pattern recognition, classification) onto tiny embedded platforms by aggressively optimizing models and runtimes so they execute within tight hardware limits. This has matured into a distinct research area at the intersection of ML and the AIoT (Artificial Intelligence of Things), focused on making learning and inference feasible under tight size, weight, and power constraints.

A common workflow trains models on a host (e.g., Keras/TensorFlow, PyTorch), converts to compact formats (TensorFlow Lite [33]), and deploys via MCU-friendly toolchains (Arduino IDE, RTOS options, or vendor SDKs). Empirical implementations demonstrate that even modest MCUs (e.g., Raspberry Pi Pico, ESP32, Arduino Nano 33 BLE) with >60 MHz clocks and >200 KB RAM can run TinyML pipelines for embedded vision when paired

with careful model compression and runtime integration [34].

While TinyML is increasingly used on microcontrollers, its application to insect-scale robots remains, to our knowledge, unexplored.

1.5 Overview of This Thesis

This dissertation investigates vision and control for insect-scale robots, addressing two fundamental challenges: hovering and navigation. These challenges are particularly difficult at such a small scale due to tighy limitations in speed, size, weight, and power (SSWaP). Drawing on bio-inspired approaches, each chapter presents a unique solution tailored to these constraints, advancing the autonomy of insect-scale flying robots.

For the hovering task, we first introduce advances in the sensor suite of an autonomous flying insect robot (FIR) weighing less than a gram. FIRs, because of their small weight and size, offer unparalleled advantages in terms of material cost and scalability. However, their size introduces considerable control challenges, notably high-speed dynamics, restricted power, and limited payload capacity. While there have been advancements in developing lightweight sensors, often drawing inspiration from biological systems, no sub-gram aircraft has been able to attain sustained hover without relying on feedback from external sensing such as a motion capture system. The lightest vehicle capable of sustained hovering—the first level of “sensor autonomy”—is the much larger 28 g Crazyflie. Previous work reported a reduction in size of that vehicle’s avionics suite to 187 mg and 21 mW. Here, we report a further reduction in mass and power to only 78.4 mg and 15 mW. We replaced the laser rangefinder with a lighter and more efficient pressure sensor, and built a smaller optic flow sensor around a global-shutter imaging chip. A Kalman Filter (KF) fuses these measurements to estimate the state variables that are needed to control hover: pitch angle, translational velocity, and altitude. Our system achieved performance comparable to that of the Crazyflie’s estimator while in flight, with root mean squared errors of 1.573 deg, 0.186 m/s, and 0.136 m, respectively, relative to motion capture. This work won the Best Student Paper Award at ICRA 2025.

Building on this, Chapter 3 is to push the limits further to NAT robots. Tiny “gnat robots,” weighing just a few milligrams, were first conjectured in the 1980’s. How to stabilize one if it were to hover like a small insect has not been answered. Challenges include the requirement that sensors be both low mass and high bandwidth, and that silicon-micromachined rate gyroscopes are too heavy. The smallest robot to perform controlled hovering uses a sensor suite weighing hundreds of milligrams. Here we demonstrate that an accelerometer represents perhaps the most direct way to stabilize flight while satisfying the extreme size, speed, weight, and power (SSWaP) constraints of a flying robot even as it scales down to a few milligrams. As aircraft scale reduces, scaling physics dictates that the ratio of aerodynamic drag to mass increases. This results in reduced noise in an accelerometer’s airspeed measurement. We show through simulation and experiment on a 30 gram robot that a 2 milligram off-the-shelf accelerometer is able in principle to stabilize a 10 milligram robot despite high noise in the sensor itself. Inspired by wind-vision sensory fusion in the flight controller of the fruit fly *Drosophila melanogaster*, we then added a tiny camera and efficient, fly-inspired autocorrelation-based visual processing to allow the robot to estimate and reject wind as well as control its attitude and flight velocity using a Kalman Filter. Our biology-inspired approach, validated on a small flying helicopter, has a wind gust response comparable to the fruit fly and is small and efficient enough for a 10 milligram flying vehicle (weighing less than a grain of rice). This work was published in Science Robotics.

For navigation, we introduce a biologically inspired method for confined-space navigation (Chapter 4), as detailed in our IROS paper [3]. This method leverages an efficient bilinear optic flow approximation, which mimics insect vision to navigate through narrow corridors. Traditional navigation techniques like Simultaneous Localization and Mapping (SLAM) are impractical at this scale due to SSWaP constraints, so we present an optic flow-based control system that learns spatial patterns through “motor babbling.” This enables the robot to maintain heading stability in a simulated corridor environment without relying on a pre-stored map, demonstrating the feasibility of low-power, real-time navigation in confined spaces.

In Chapter 5, we develop a deep-reinforcement-learning framework that enables vision-only confined space navigation using onboard event-based sensing. We trained the policy in NVIDIA Isaac Lab, then applied TinyML quantization for deployment on an OpenMV Cam RT1062 board paired with a GenX320 event-camera module, achieving fully onboard inference. We demonstrate that the resulting power consumption meets insect-scale budgets. To our knowledge, this is the smallest and lightest aerial robotic platform with an onboard event camera to date.

Chapters 2–5 are self-contained. Variables and notation are defined within each chapter, and figures and equations are numbered locally.

Chapter 2

TINY SENSE: A LIGHTER WEIGHT AND MORE POWER-EFFICIENT AVIONICS SYSTEM FOR FLYING INSECT-SCALE ROBOTS.

2.1 Introduction

Insect-scale robots have the potential to be deployed in the thousands or millions to perform “fast, cheap, and out of control” space missions, collective assembly tasks, or hazard detection, owing to their small size and low materials cost. Recent incarnations of sub-gram aircraft have demonstrated controlled flight [11, 35], as well as electromagnetically-mediated power delivery [36–38]. But sensor autonomy, that is, the ability to hover without external feedback, has not yet been demonstrated below 10 g, never mind 1 g. A central question is “What are the minimum sensor suite and computation resources needed for the task of flight control?” Of particular concern when designing such a control architecture is how the physics of scale affects sensing and control. The paper [2] serves as a step toward a missing but complementary analysis to previous work that has largely focused on scaling effects on actuation and mechanics in small robots [39–42].

Insect-scale robots have extreme size, speed, weight, and power (SSWaP) constraints. “Sensor autonomy” including stable hovering, represents the first level of autonomy in the hierarchy proposed in [43], and is the foundation upon which higher level capability is built. We impose a requirement that sensing and computation take no more than 10% of the power required to stay in the air; any more than this and flight time is severely impacted. This factor holds for important examples of autonomous drones, such as the 1.5 kg system in [44] and the 30 g system in [45]. This suggests a power budget of ~ 20 mW for a 150 mg Robofly (Figure 2.1) that consumes 100 mW to fly (about 200 mW after losses from a 50% efficient

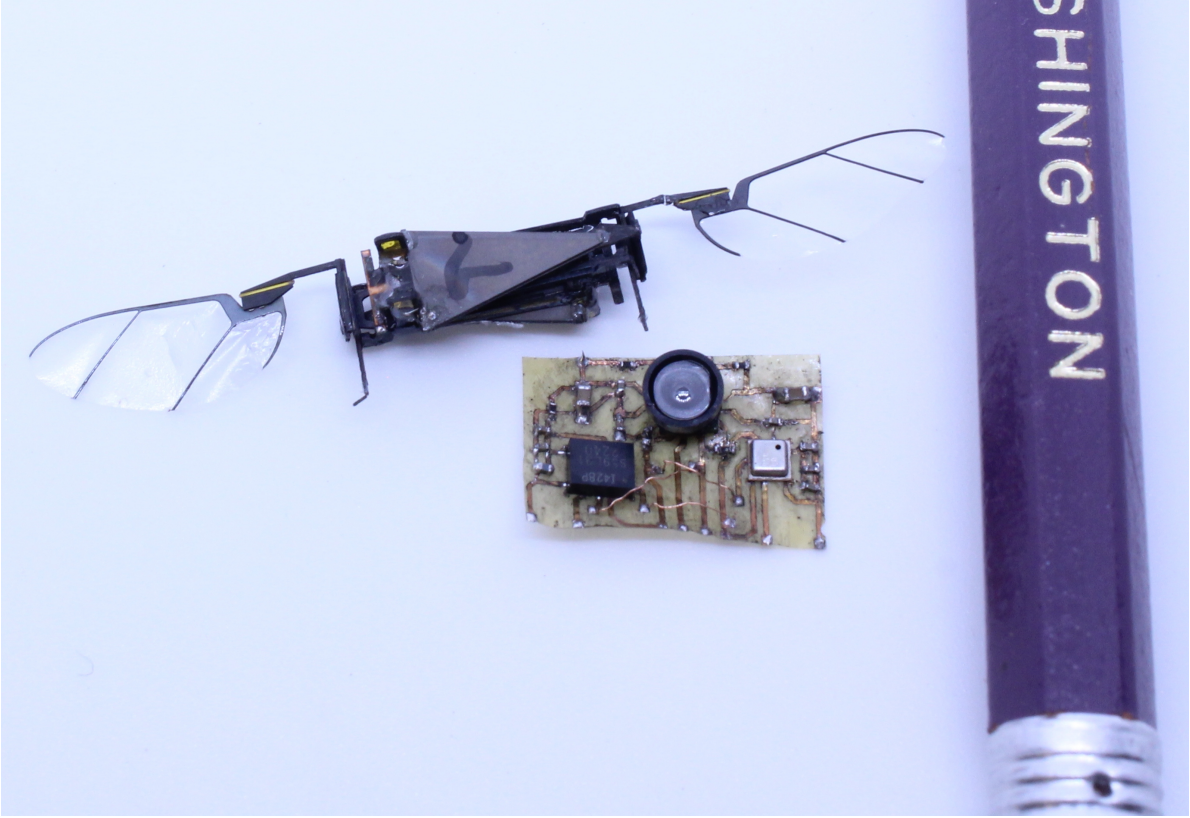


Figure 2.1: The TinySense sensor suite shown next to a U. Washington Robofly and a standard pencil for scale. The width of the sensor suite board is approximately 12 mm and its mass is 78.4 mg.

boost converter are factored in).

Large drones can carry larger sensor payloads such as light-based ranging systems (LiDAR) and the global positioning system (GPS). However, GPS does not work well in indoor environments [46] and small robots are not able to hold emissive sensors due to constraints in size, speed, weight, and power consumption (SSWaP). Recent work has introduced a 187 mg avionics suite that consumes 21 mW [47], considered gyroscope-free avionics [2], and has investigated the effect of wing-induced vibrations [48].

In this paper, we introduce a new avionics system for autonomous hovering flight that is even better tailored in mass and energy consumption for an insect-scale robot (Figure 2.3)

than previous work [47]. To do so, we made two key refinements to lower power and mass. First, we replaced the power-hungry laser rangefinder with a pressure sensor. In recent years, these have attained sufficient precision to be used for the purpose of altitude estimation and control, even for insect-scale robots. Second, we replaced the optic flow sensor with a global shutter camera and computed optic flow on a microcontroller, resulting in a reduction in weight from 97 mg to 44 mg. As in [47], this new sensor suite is able to estimate the crucial variables that are needed for controlled flight: pitch angle, translational velocity, and altitude.

Our contributions, in order of importance, are:

1. A reduction in weight of the sensor suite to 78.4 mg which is less than half of the previous lightest sensor suite [47].
2. Demonstration of accurate estimates of the state variables pitch angle, translational velocity, and altitude using data collected wirelessly from this sensor suite while mounted on a drone.
3. Optic flow estimation on a microcontroller.

Subsequent sections elaborate on the dynamics, sensor configurations and measurement models, and validate our findings with data acquired wirelessly from our sensor suite, which is then analyzed on a desktop computer.

2.2 Dynamics

The basic dynamics of most small hovering devices are unstable in pitch angle and position. This was true for the first robotic flies [49] and EHD thruster actuated robots [50], as well as larger aircraft such as the tailless Delfly Nimble [51]. Techniques to stabilize them passively through modification of wing configuration [52] or using air dampers [53] are either unproven or result in undesirable steady-state oscillations, respectively. Our approach in this work is to

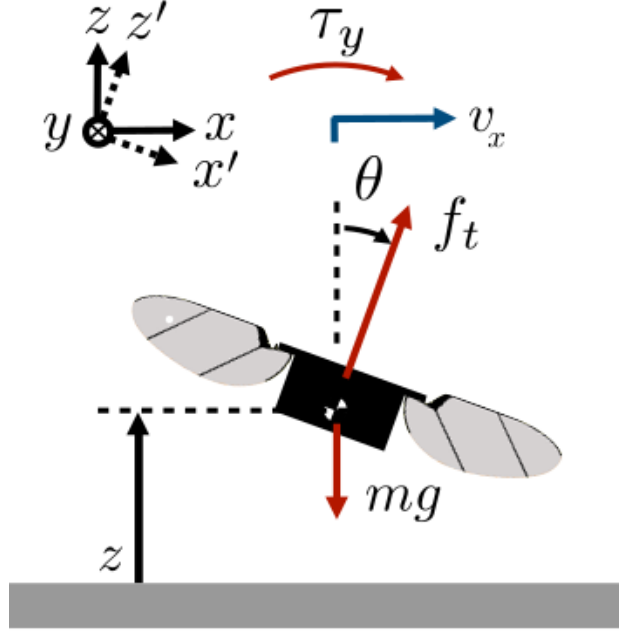


Figure 2.2: Forces and state variables of a small hovering Flying Insect Robot (FIR).

first use linearized dynamics and observation models from which we can construct a Kalman observer of these unstable states so that they can be controlled.

Figure 2.2 depicts forces acting on a flapping-wing robot such as a gnat robot or the Robofly, but the proposed model can apply to almost any small flying aircraft, including EHD-based thrusters [54]. Near hover, the vehicle's dynamics are well approximated by a linear state-space model [53, 55, 56] of the form $\dot{\mathbf{q}} = \mathbf{A}\mathbf{q} + \mathbf{B}\mathbf{u} + \mathbf{G}\mathbf{w}$, where $\mathbf{w}$ is the process noise, assumed to be zero-mean Gaussian white noise with known covariance. Process noise $\mathbf{w}$ is included because it is a necessary consideration for the design of the estimator.

To simplify the analysis, we restrict our attention to motion in the x - z plane; the dynamics in the y - z plane are very similar. We choose a minimal state vector $\mathbf{q} = [\theta, v_x, z]^T$, where θ is the pitch angle (rad), v_x is the vehicle's translational velocity in the world x -direction (m/s), and z is its vertical position or altitude (m) as illustrated in Figure 2.2. When the nonlinear Euler-Lagrange equation dynamics are linearized by taking the Jacobian at $\theta = 0$, thrust

force $f_t = mg$ (N), and $z = z_d$, where z_d is the desired altitude (m), the dynamics matrices are given by:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ g & -\frac{b}{m} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad (2.1)$$

where m is the vehicle’s mass (kg), and b is the vehicle’s translational drag proportionality constant (Ns/m) [56]. Note that because we are concerned in this paper about only estimation, we ignore inputs from the motor, and instead use the measurements of the gyroscope $\mathbf{u} = [\omega_m]$ (rad/s) as the only input.

2.3 *Sensor Suite and Measurements*

Next we consider the sensor suite and state estimator. Our interest is in sensors that are either available commercially off-the-shelf, nearly so, or can be fabricated using the same tools that have been successful building insect robots such as the Robofly [57] and Robobee [10, 11]. A further constraint is that they require extremely low power for both the sensor itself and subsequent transduction and processing. This largely eliminates sensors that emit power, such as radar, sonar, depth cameras that emit structured light, and scanning laser rangefinders [58], but permits passive sensing such as vision. The global positioning system is largely denied indoors and does not provide enough resolution anyway (1–10 m), and, further, requires significant power for signal processing. A lightweight and low-power sensing modality that can provide the necessary information to hover is “optic flow,” which is defined as the velocity of the image (in rad/s) across the camera imager [59–61]. In tandem with other sensors, optic flow can be used to estimate the distance to obstacles as the imager translates through space [62]. Methods for estimating optic flow include calculating luminance gradients locally [63] (Lucas-Kanade), or image-wide [64], as well as iterative search for matching blocks of pixels [65]. Distances can also be estimated with a second camera and stereo vision using line matching [66], but this requires a second camera and a long “baseline” between cameras for accuracy, which adds excessive weight.

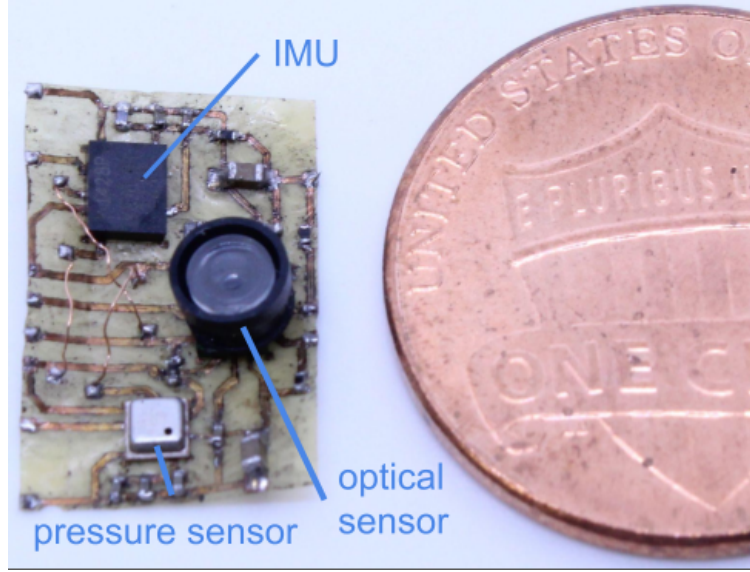


Figure 2.3: Sensor suite (PAG7920LT optical sensor, ICM42688-P IMU, BMP388 pressure sensor) next to US 1 cent coin.

Given these constraints, we propose the following set of onboard sensors: An optical sensor, a pressure sensor, and a gyroscope. We designed a customized microcircuit with the three sensors. (Figure 2.3)

2.3.1 Camera and optic flow

The system employs an optical sensor, the PAG7920LT (PixArt Imaging, Inc., Taiwan), to capture 160×120 QQVGA images with a 120° field of view at 100 Hz over a 32 MHz SPI connection, with which the microcontroller computes optic flow via the Lucas-Kanade Method. The reason we select this camera is that, to our knowledge, it is the only global shutter camera that provides an SPI interface to transmit the image data. Other global shutter cameras require a special interface called MIPI (mobile industry processor interface), which is not available for any microcontroller. The majority of cameras employ a “rolling shutter” in which pixel rows are captured at sequential periods of time. A global shutter

camera, by contrast, captures that all pixels at the same time, which ensures an accurate optic flow measurement.

To minimize the data transmitted wirelessly over Bluetooth from our TinySense sensor suite, it is necessary to perform all optic flow calculations onboard. We wrote a Lucas-Kanade algorithm using the Arduino development environment in C++ to run on NRF52-based Adafruit Express board, chosen for its low size and weight.

The Lucas-Kanade algorithm is performed by calculating the following quantities:

$$J = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \dots & \dots \\ I_x(p_n) & I_y(p_n) \end{bmatrix}, \mathbf{v} = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, b = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \dots \\ -I_t(p_n) \end{bmatrix},$$

where $p_1, p_2, \dots, p_n$ are the pixels in the image, $\mathbf{v}$ is the image flow in pixels per frame, and $I_x(p_i), I_y(p_i), I_t(p_i)$ are the image's horizontal, vertical, and temporal (time) partial derivatives respectively, at pixel p_i . Then, the optic flow is estimated (in pixels/frame) by taking the pseudoinverse $\mathbf{v} = (J^T J)^{-1} J^T b$.

To minimize computation for the limited microcontroller resources, we reduced the image resolution by a factor of 4 (producing a 40×30 resolution) by skipping pixels.

The two most expensive operations involve computing $J^T J$ and $J^T b$. A 40×30 image produces an J matrix with $(40-2) \times (30-2) = 1064$ rows and 2 columns and a b matrix with 1064 rows and 1 column. Multiplying two matrices of size $k \times l$ and $l \times m$ requires at worst kml multiplications. Thus, computing $J^T J$ and $J^T b$ require 4256 and 2128 multiplications, respectively. We neglect the additional operations required to compute their product (2 multiplications) and the 2×2 matrix inverse (6 multiplications and one divide), giving a total operations count of 6384.

We additionally implemented a form of sparse optic flow in which flow vectors were computed for twelve 10×10 patches, which are averaged to produce a final estimate. A 10×10 image produces an J matrix with $(10-2) \times (10-2) = 64$ rows and 2 columns and a b matrix with 64 rows and 1 column. Thus, computing $J^T J$ and $J^T b$ require approximately

256 and 128 multiplications, respectively. With twelve matrices of size 10×10 , the total number of multiplication is $12 \times (256 + 128) = 4608$. This is a reduction of approximately 28%.

To convert the optic flow from pixels/frame to angular velocity (rad/s), we found the calibration constant $\alpha = -0.027$ (rad/pixel) by rotating the TinySense by hand and performing a linear regression against its gyroscope readings.

2.3.2 Pressure sensor

Altitude measurements were conducted by a Bosch BMP388 pressure sensor. This sensor estimates altitude as a proportion of the difference between the air pressure and sea level. We set it for $2\times$ pressure oversampling and an IIR filter coefficient of 3 to maximize resolution. To compensate for variations in local pressure causing inaccurate altitude measurements, the sensor was assigned a bias equal to the average of the first 25 measurements where the sensor system was grounded; this bias was subtracted from every reading.

2.3.3 Gyroscope

For the onboard inertial measurement unit (IMU), we chose the TDK ICM42688-P because it is a recent model that is available in the smallest commercially-available package size of 2.5×3 mm. It includes both a 3-axis gyroscope and a 3-axis accelerometer. In this work, we used only the gyroscope, which measures angular velocity through the detection of Coriolis forces within an electromechanical resonator. We configured the sensor to have a ± 2000 degrees/s full-scale (FS) range and enabled the sensor-provided low-noise mode to improve the sensitivity and minimize noise. During initial experiments, we noticed that the gyroscope's signal was noisier than expected and differed significantly from the Crazyflie's gyroscope, even after applying a digital low-pass filter. In handheld, non-flying experiments, however, the gyroscope proved accurate. We hypothesized that vibrations from the spinning propellers, which were at a higher frequency than the 100 Hz sampling rate, were introducing aliasing. We enabled the sensor's anti-aliasing low-pass filter with a cutoff frequency of 42 Hz,

Table 2.1: Comparison between TinySense and prior work.

	component	size (mm)	mass (mg)	power (mW)
prior work [47]	IMU	$2.5 \times 3 \times 0.91$	14	3
	rangefinder	$4.9 \times 2.5 \times 1.56$	16	5
	optical flow	$5 \times 5 \times 3.08$	97	12
	board/discretes	–	60	–
	total	–	187	21
TinySense	IMU	$2.5 \times 3 \times 0.91$	14	3
	pressure	$2.0 \times 2.0 \times 0.75$	14	2.6
	camera chip	$2.2 \times 2.7 \times 6.9$	24	10
	lens	$3.8 \times 3.8 \times 1.95$	20	–
	board/discretes	–	26.4	–
	total	–	78.4	15.6

which is well below the Nyquist frequency associated with the 100 Hz sampling rate. We found that this setting significantly reduced the gyroscope’s noise and error.

2.4 State Estimation

To implement a Kalman Filter, we must first ascertain whether our system is observable at equilibrium. To show this, we linearize the observation model for the task of hovering. Note that the gyroscope provides a low-noise measurement of angular velocity $\boldsymbol{\omega}$. In cases like this in which the sensor provides more precise information than can be derived knowing the system’s actuated inputs (e.g. torque applied by the aircraft’s rotors or wings), it is common practice to reformulate the Kalman Filter to use the sensor’s readings as “input” $\boldsymbol{u}$, and to ignore actual system input. This has an additional advantage that one less state variable is required for each such measurement. Our system includes gyroscope readings as one of its inputs and hence angular velocity $\boldsymbol{\omega}$ does not appear among the state variables. Our observation model for the suite of sensors is:

$$\boldsymbol{y} = \begin{bmatrix} \Omega_m + n_o \\ z_m + n_p \end{bmatrix}, \quad (2.2)$$

where Ω_m is the optic flow measurements (rad/s) from optic flow sensor, z_m is the measurement (m) from the pressure sensor, and n_o and n_p are the noise terms for the two sensors. We assume noise terms are zero-mean Gaussian white noise, a standard assumption in sensor noise modeling to simplify the analysis. The measurement model for the optic flow in the x direction, as measured by the optic flow camera is a nonlinear function of the state [2]:

$$\Omega_m = \omega_m - \frac{v_x}{z}, \quad (2.3)$$

where ω_m is the angular velocity measurement (rad/s) from gyroscope. The optic flow measurement depends on z . We can take the Jacobian linearization at a desired altitude z_d :

$$\Omega_m(z) = \Omega_m(z_d) + O(z) = \omega_m - \frac{v_x}{z_d} + O(z). \quad (2.4)$$

With this linearization, our measurement model can be cast into the Kalman Filter model, $\mathbf{y} = C\mathbf{q} + D\mathbf{u} + \mathbf{n}$, where

$$C = \begin{bmatrix} 0 & -\frac{1}{z_d} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \quad (2.5)$$

The observability matrix $[C; CA]$ is full rank, meaning that a Kalman Filter will be able to estimate the system's state using the measurements given in Eq. (2.5).

The Kalman Filter observer estimates $\hat{\mathbf{q}}$ by numerically integrating the system:

$$\dot{\hat{\mathbf{q}}} = A\hat{\mathbf{q}} + B\mathbf{u} + K(\mathbf{y} - C\hat{\mathbf{q}} - D\mathbf{u}), \quad (2.6)$$

where K is the Kalman gain.

We calculated optic flow variance R_{n1} to be 0.017 by subtracting its value from that predicted by mocap using Eq. (2.3) during a constant-velocity horizontal translational flight. We calculated the variance of the pressure sensor R_{n2} to be 0.0055 by taking measurements while holding it at a constant altitude for 60 seconds:

$$R_n = \text{diag}(R_{n1}, R_{n2}) = \text{diag}(0.017, 0.0055).$$

To estimate the process noise covariance matrix Q_n , we employed a hybrid optimization approach, combining grid search for coarse parameter initialization, gradient descent for fine-tuning, and manual adjustment to minimize the RMSE between TinySense state estimates and mocap ground truth:

$$Q_n = \text{diag}(0.0124^2, 0.001^2, 0.22^2).$$

For the process noise (disturbance) matrix G , we assumed that the noise enters the system as white noise disturbance adding to the derivatives of the three states θ , v_x , and z . Based on these, the Kalman gain K was computed using the `lqe` command in `python-control` [67]:

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad K = \begin{bmatrix} 0.095 & 0 \\ -1.32 & 0 \\ 0 & 3 \end{bmatrix}.$$

We implemented the Kalman Filter offline in Python by iteratively computing the state estimates $\hat{\mathbf{q}}_{i+1}$ according to $\hat{\mathbf{q}}_{i+1} = \hat{\mathbf{q}}_i + \dot{\hat{\mathbf{q}}}_i \cdot dt$.

2.5 Experiments

2.5.1 Robot platform

We performed experiments using Crazyflie 2.0, a palm-sized quadrotor by Bitcraze (Sweden), in wind-free conditions in an indoor environment. In addition to its built-in gyroscope, the Crazyflie was equipped with the Flow deck v2 consisting of a downward-facing optic flow sensor and a laser rangefinder; it uses an Extended Kalman Filter to estimate its state in flight [68].

2.5.2 Motion capture system

To provide a ground-truth comparison for the sensors, we used a twelve-camera OptiTrack Flex 13 motion capture arena, with spatial accuracy ± 0.20 mm. The Crazyflie drone was fitted with four retro-reflective motion capture markers and its position and orientation were

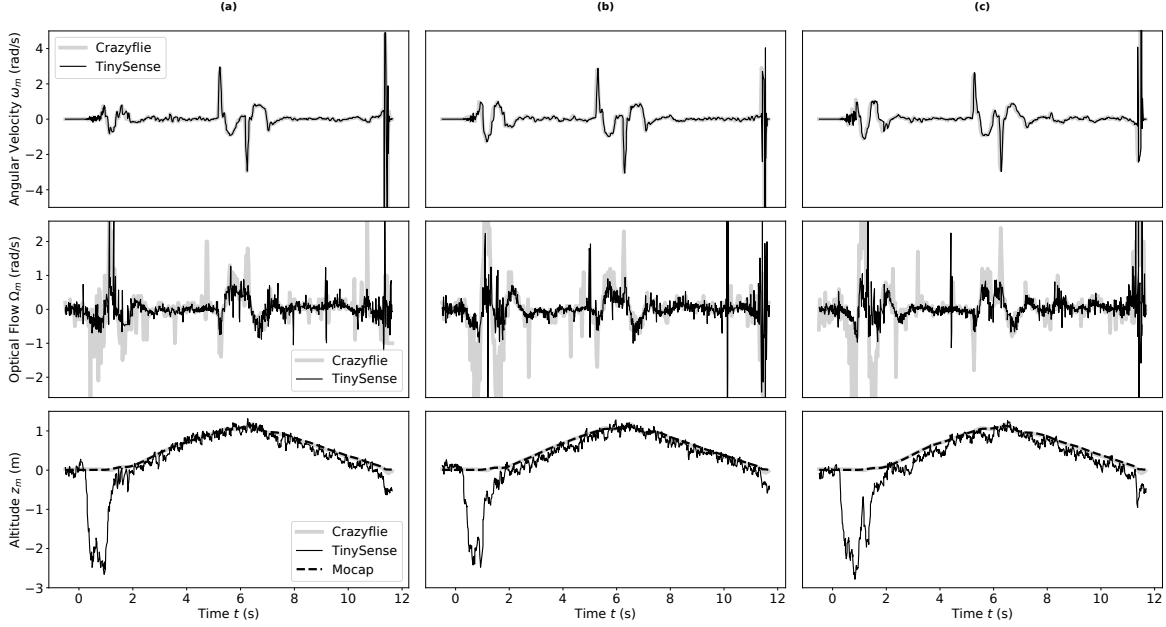


Figure 2.4: Sensor measurements from the onboard gyroscope, optic flow sensor, and pressure sensor of TinySense are compared with those from the Crazyflie for three different flight experiments (a), (b), and (c); trajectories are described in the main text and corresponding state estimates in Fig. 2.5. In the altitude plots, altitude is measured by the Crazyflie’s laser rangefinder, while the TinySense uses a pressure sensor; mocap data is also included for comparison.

recorded at 120 Hz. Velocity was estimated by taking the numerical derivative of the position data in post-processing.

2.5.3 Experiment

We connected our sensor suite to an NRF52840 Express (Adafruit), a board which supports data transmission through UART and 2.4 GHz Bluetooth Low Energy compatibility. We mounted the board as close to the geometrical center of the Crazyflie as possible to maintain a reasonable center of mass and minimize vibration interference to the IMU from the propellers.

Table 2.2: Estimate of optic-flow algorithm power use (FP = floating-point).

update frequency (Hz)	occurrence/update	task	single-cycle ops	int division ops	FP division ops	total cycles/update	total cycles/s (MHz)	power (mW)
100	1	skip frame	8493	0	8	249781	25.0	5.5
	12	LK OF	18434	128	2			
	1	copy data	1200	0	0			

Power was supplied from the Crazyflie’s 250 mAh Lithium Polymer (LiPo) battery.

Each trial consisted of commanding the robot to rise to an altitude of 1 m, translate forward 1 m at 1 m/s, and then descend back to the ground. We ran this experiment three times. We collected measurements from the TinySense’s sensors on a laptop as they were transmitted over Bluetooth at 100 Hz as binary data. Simultaneously, a second computer captured motion capture data at 120 Hz and used a Python-based script to capture sensor measurements and state estimates transmitted from the Crazyflie over Bluetooth at its maximum rate of 30 Hz. Each receiving device independently recorded a UTC timestamp, which was used to synchronize the data. The sensor measurements from the TinySense compared with the Crazyflie, and the motion capture (mocap) system’s measurement for altitude are shown in Figure 2.4.

2.5.4 Computational load

Table 2.2 provides an estimate of the power consumption of the optic flow algorithm, by calculating the number of clock cycles needed for each operation. Our estimate includes the matrix operations described in Section 2.3.1, as well as additional housekeeping operations, and is based on ARM Cortex-M4 instruction timings for addition, subtraction, multiplication, and memory read/write operations; integer divisions require 12 cycles at most and floating-point divisions require 14 cycles. Given that in our current system all sensors update at a rate of 100 Hz, we estimated that the total number of needed cycles per second is ~ 25.0 Mhz.

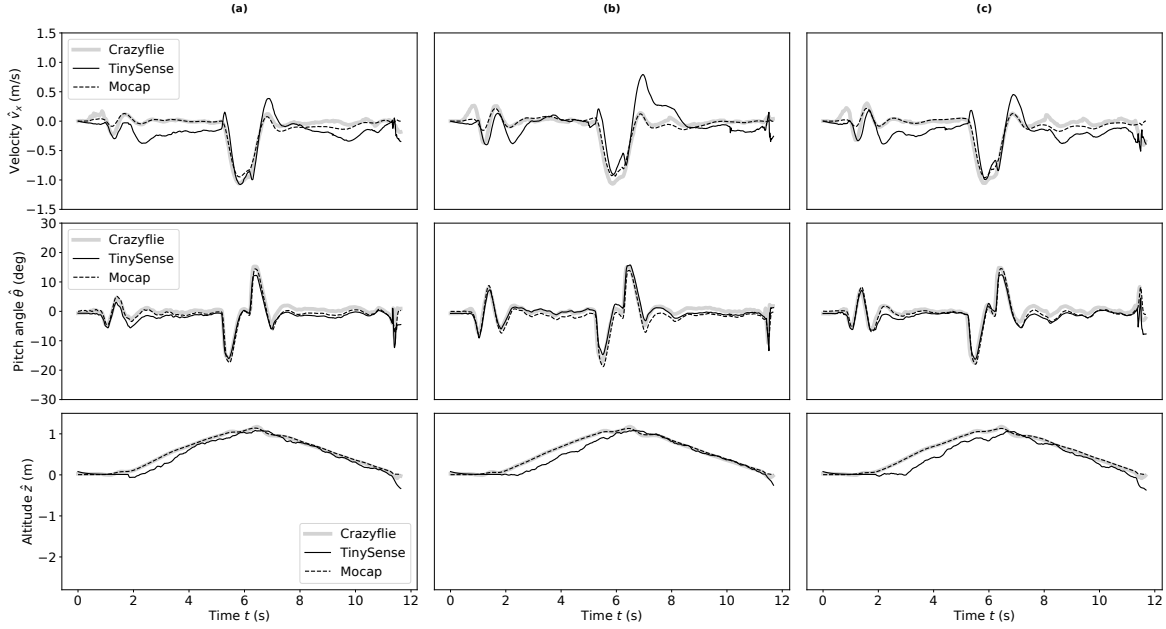


Figure 2.5: Comparison between state estimates from TinySense’s Kalman Filter, Crazyflie, and mocap. We conducted three flight experiments (a), (b), and (c), with the Kalman filter estimates starting at time zero. Pressure sensor measurements were ignored during the first 1.8 s because pressure changes due to the ground effect at takeoff confound readings.

Operating the nRF52840 with an efficiency of $52 \mu\text{A}/\text{MHz}$ and a battery-powered voltage of 4.2 V, we estimate the power usage of the optic flow algorithm to be 5.5 mW.

2.5.5 Results and discussion

Figure 2.5 compares our sensor suite’s state estimate to that of the Crazyflie’s built-in estimator. States derived from the mocap system are considered to be ground truth. For the state estimator, time $t = 0$ corresponds to when the gyroscope reading becomes significantly different from zero, indicating liftoff. Figure 2.4 shows that a pressure increase occurs when the propellers first turn on, which can be observed as a drop in measured altitude. To mitigate unwanted effects of this in the Kalman Filter, we ignored (set to zero) altitude mea-

Table 2.3: Root Mean Squared Error (RMSE) of estimates relative to the motion-capture system.

	v_x (m/s)	θ (deg)	z (m)
Crazyflie	0.075 ± 0.009	1.619 ± 0.267	0.021 ± 0.001
TinySense	0.186 ± 0.015	1.573 ± 0.166	0.136 ± 0.026

measurements z_m during the first 1.8 s after liftoff. Nevertheless, once it is in the air, the altitude measurement matches quite closely with the laser rangefinder on the Crazyflie, though with slightly more noise.

Our system’s estimate compares well with the Crazyflie’s, despite receiving gyroscope readings at only 100 Hz, which is ten times slower [68]. The key metric of pitch angle RMSE is approximately equal (Table 2.3). Large errors in translational velocity v_x appear around $t = 5$ and $t = 6$. These occur after sudden changes in attitude, during which angular velocity exceeds 3 rad/s (Figure 2.4). At these moments, we observed that TinySense optic flow readings are consistently lower than those of the Crazyflie. Lucas-Kanade cannot detect motion accurately if the displacement between frames is greater than a half pixel between image frames [69]. This limits optic flow measurement to $\Omega_m < 0.5\alpha F = 1.4$ rad/s, where $\alpha=0.027$ is the angular increment between pixels and $F=100$ is the frame rate (Hz). We believe that the lateral velocity estimate diverges during these fast maneuvers because it depends on erroneous optic flow readings.

2.6 Conclusion

In the present study, we introduce a new sensor suite with dramatically reduced mass, and associated optic flow and state estimation software that is compatible with the computational constraints of an onboard microcontroller small enough to fly aboard flying insect robots (FIRs) weighing less than a gram. Our proposed avionics suite diverges in two important ways from the 28 g Crazyflie helicopter, the lightest vehicle yet to perform controlled hovering,

and previous work [47]. First, we replaced a power-hungry laser rangefinder with a much more efficient (and slightly less ideal) and lighter pressure sensor. Second, we replaced an off-the-shelf optic flow sensor, which is available only in relatively heavy packages, with a custom camera with a global shutter and a custom-written Lucas-Kanade-based optic flow algorithm running onboard a 10 mg microcontroller small enough to fly onboard an FIR. We estimate the compute power usage is about 5 mW. Our system uses a linear Kalman Filter to estimate pitch angle, translational velocity, and altitude. This can be readily adapted to a nonlinear extended Kalman Filter for more generality [47].

The cumulative weight of this sensor suite is only 78.4 mg, rather less than the 187 mg of previous work [47]. With the addition of a 30 mg microcontroller and crystal, the overall system remains well within the 252 mg payload limit of the 143 mg robotic platform reported in [52]. Our system demonstrates average RMSE values of 0.186 m/s for velocity, 1.573 deg for pitch angle, and 0.136 m for altitude when compared to the motion capture system. In comparison, the Crazyflie platform achieves RMSE values of 0.075 m/s, 1.619 deg, and 0.021 m for the same states during similar flight maneuvers. This shows that our system performs closely to the state-of-the-art Crazyflie in these scenarios. The pressure sensor showed negligible drift over a 60-second period, providing a usable altitude estimate as long as its output during takeoff is ignored. Future improvements include mitigating the lateral velocity errors that occur during rapid rotations that exceed the limit of Lucas-Kanade estimation (Section 2.5.5) by increasing the frame rate or image blurring. Additionally, our patch-based optic flow estimation (Section 2.3.1) may allow outliers to be rejected based on their eigenvalues to eliminate low-contrast patches, reducing noise [69].

Future work will integrate TinySense along with a microcontroller and power hardware onto insect-scale robots such as the Robofly [70] to perform visual navigation [3]. The Robofly performs faster accelerations than the Crazyflie and is a different vibration environment, but is otherwise similar. Previous work shows vibrations do not confound gyroscope readings [48, 71]. Algorithmic changes in optic flow may be needed to accommodate vibration, but our choice of a global shutter camera will mitigate tearing artifacts.

Chapter 3

A GYROSCOPE-FREE VISUAL-INERTIAL FLIGHT CONTROL AND WIND SENSING SYSTEM FOR 10 MG ROBOTS

3.1 Introduction

The idea of extremely small autonomous robots, termed “gnat robots” [72, 73], first gained widespread attention in the 1980’s. To provide a precise definition and terminology, we will define a *Nature-inspired Aerial Ten-milligram robot*, or “NAT robot” to refer to a robot that flies and weighs between 1 and 10 mg. A grain of rice weighs approximately 10 mg, making NAT robots significantly smaller than the 100–600 mg flapping-wing flyers (weighing about the same as 1 to 6 toothpicks) that have been created to date such as the Robofly [4, 36, 57], Robobee [10, 11], Bee+[74], and Softfly [22] (Figure 3.1). By virtue of their small size, NAT robots will have new capabilities. Operating in teams of thousands or millions, they could perform “fast, cheap, and out of control” space missions at dramatically reduced launch cost, serve as autonomous mobile “smart dust” [75] to find hazardous fume sources or map air flow patterns, or collectively perform manipulation tasks on objects larger than themselves [76]. Reduction in size to below 10 mg amplifies many of the scale-dependent benefits of small robots. These include greater deployment numbers at the same cost for improved coverage, and the ability to harvest all needed energy from a greater array of ambient energy sources in the environment. The increasing surface-area-to-volume ratio as scale diminishes favors solar power in small robots, for example [77]. The ability to fly affords important benefits for small scale. These include easily surmounting obstacles [72] and the concomitant ability to come into closer proximity to sensing targets and power sources, lowering sensitivity and conversion efficiency requirements.

When sensing and control of such small flying vehicles is considered, the physics of small

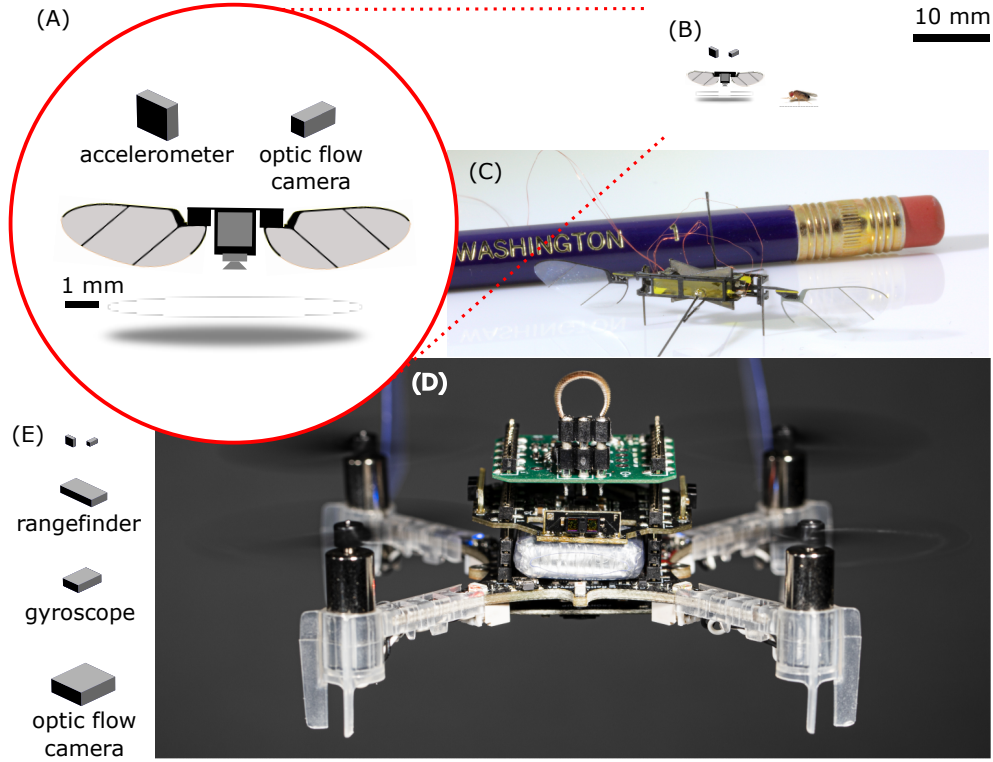


Figure 3.1: The sensor suite and size of the conceptual Nature-inspired Aerial Ten-milligram “NAT” robot compared to other small aircraft. (A) close-up view of the conceptual 10 mg robot and package sizes of its sensor suite made with off-the-shelf parts. Shown approximately to scale on the page are the (B) the NAT robot and its sensors along with the 1 mg fruit fly, (C) the 143 mg U. Washington Robofly [4, 5] with a pencil for scale, and (D) a palm-sized 30 g quad-rotor, the lightest vehicle yet to perform sensor-autonomous hover. Here it is shown equipped with the moth-based odor sensor from [6]. (E) Our proposed hovering sensors are much smaller and more power efficient than those used for autonomous hover in the palm drone (sensor package sizes also shown approximately to scale on the page).

scale takes on a heightened importance [39–42]. For a robot of size scale ℓ (for example its length from wingtip to wingtip), mass varies as ℓ^3 to first order. This implies sensor mass must diminish as scale reduces, which rules out many sensor types for NAT robots and

implies lower precision or update rates for others. It also places a severe limit on sensor and controller power, because battery mass also scales as ℓ^3 . As sensor controller quality degrade with scale, the speed of the dynamics they must control concomitantly increases. Angular acceleration and translational acceleration vary as ℓ^{-1} (if translational velocity is measured in units of body lengths per second, or ℓ/s) [78]. The attitude of small hovering vehicles such as flapping-wing aircraft [49, 51], electrohydrodynamic vehicles [50, 54], and flies [79] is unstable. This implies a size-dependent upper limit on sensor time delay or sampling rate [80]. Together these constitute extreme constraints on the speed, size, weight, and power (SSWaP) of a NAT robot’s avionics system.

We addresses in this article the lowest level in the drone autonomy hierarchy introduced in [43], known as “sensor autonomy.” Sensor autonomy entails the ability to hover in the air stably and is a requisite for higher level tasks such as navigating through confined environments [44, 81–86] and plume source-seeking [6, 87, 88]. Previous work in such avionics or “autopilot” systems geared toward small aerial vehicles has recognized the need for computation- and power-efficient onboard vision. A key constraint is power usage: we assume that sensing and computation for a NAT robot must consume no more than 1 mW. This is 10% of the power to fly, like previous visual flight demonstrations at 1.5 kg [44] and 30 g [45], and comes from scaling the 200 mW required for a 100 mg Robofly [57] down by a factor of ten. This rules out off-board sensor processing because wireless radio transmission consumes tens of mW even for low-rate, low-resolution video [89]. It also rules out emissive sensors such as laser rangefinders [72, 85] and sensors that require significant computation. The latter include signal processing for the Global Positioning System, and localizing relative to a stored map (e.g. Simultaneous Localization and Mapping or SLAM). Flight control using optic flow, a measure of the velocity of motion of visual scenery as the robot or animal moves through it, has been proposed as a low-power alternative [44, 62, 85, 90, 91].

The primary contribution here is an alternative avionics system with dramatically reduced mass and power consumption that is nevertheless able to control such a vehicle. Our sensor suite is notable in that it eschews a heavy and power-hungry gyroscope, relying instead on a

much lighter and more efficient accelerometer. The accelerometer is used to sense airspeed by sensing resultant drag-induced accelerations. Using this airspeed measurement and a model of the aircraft, it is possible to estimate attitude using a Kalman Filter. We combined this with a power-efficient optic flow estimator, allowing the wind vector can be estimated as well. All elements of our system can be made using slightly modified versions of off-the-shelf components, implying that no untested technology is required. The sensors, computations, and controller together weigh approximately 6 mg in total and consume less than 200 μW on an off-the-shelf microcontroller. This is achieved in part by computing exclusively using fast and power-efficient multiply and add operations. We show through analysis, simulation, and validation on a small flying aircraft that the attitude estimate enables fast lateral maneuvers and the wind estimate facilitates wind rejection. Our sensor suite also enables tasks such as plume source finding that rely on knowledge of the wind vector, and compensates for the attitude instability of small hovering aircraft.

3.2 Method

Dynamics model We created a nonlinear flight dynamics model to simulate NAT robot flight motion and to construct the linearized dynamics model used in its state estimator.

For many hovering aircraft, the center of aerodynamic drag force is displaced by a vector $\mathbf{d}_z$ relative to the center of mass (Figure 3.2), which results in a torque-force coupling. If this is the case, a slight modification of our air drag model in Equation (3.5) is needed, which is given by

$$\mathbf{f}_d = b(\mathbf{v}_w - \mathbf{v} - \boldsymbol{\omega} \times \mathbf{d}_z), \quad (3.1)$$

where $\boldsymbol{\omega}$ is the angular velocity vector of the vehicle. The resulting drag torque is given by $\boldsymbol{\tau}_d = \mathbf{d}_z \times \mathbf{f}_d$. Whether or not $\mathbf{d}_z$ is zero-length does not affect either the observability or controllability of our system discussed below, but its effect was included to provide a more faithful model of many real-world aircraft. An aerodynamic rotational damping coefficient c that adds rotational drag according to $\tau_{dy} = -c\omega_y$. It can be shown that $c = bd_z^2 + bd^2$ for

the roll axis and $c = bd_z^2$ for the pitch axis [53], where d is the lateral distance from the center of mass to the center of aerodynamic pressure of the wing (denoted by a dot in Figure 3.2).

To simplify exposition and analysis, we assume that yaw angle is controlled to have a near-zero yaw rate, which has previously been demonstrated on an insect-sized flapping-wing robot [4]. With this assumption and an assumption that the moment of inertia matrix $\mathbf{J}$ is diagonal, cross product terms in the Euler-Lagrange equations (see Supplementary materials) can be neglected. If the attitude remains near upright ($\theta = 0$), then the dynamics reduce to two nearly independent planar systems with three degrees of freedom, that is, motion and rotation in the x - z and y - z planes, each of which can be controlled independently [92, 93]. Hence our nonlinear 2D simulated dynamical system is a simplified version of Equation (3.21) given by

$$\begin{aligned}
 \dot{\theta} &= \omega \\
 \dot{\omega} &= \frac{1}{\mathbf{J}} (b\mathbf{d}_z \times \mathbf{f}_d - c\omega + \tau_y) \\
 \dot{\mathbf{v}} &= \begin{bmatrix} 0 \\ -g \end{bmatrix} + \frac{1}{m}\mathbf{f}_d + \frac{1}{m}\mathbf{R} \begin{bmatrix} 0 \\ f'_t \end{bmatrix} \\
 \dot{\mathbf{p}} &= \mathbf{v} \\
 \dot{v}_w &= 0,
 \end{aligned} \tag{3.2}$$

where $\mathbf{f}_d$ is given in Equation (3.1), $\mathbf{p} = [x, y, z]^\top$ is the position of the center of mass of the robot in world coordinates, and $\mathbf{R}$ is the rotation matrix that relates vectors given in body-attached coordinates to world coordinates. For any vector $\mathbf{v}$ expressed in world coordinates, $\mathbf{v}'$ is the same vector expressed in body-attached coordinates. They are related by $\mathbf{v} = \mathbf{R}\mathbf{v}'$. The wind velocity state v_w (Figure 3.2) is included because the Kalman Filter estimates it. For the 2D case, we use that $\mathbf{R} = [\cos \theta, \sin \theta; -\sin \theta, \cos \theta]$. The gravitational force is given by $\mathbf{f}_g = [0, 0, mg]^\top$.

If thrust is generated by a pair of thrusters, such as two wings or a pair of EHD thrusters, thrust force f_t and control torque τ_y are actuated independently by inverting the relations $f_t = f_1 + f_2$ and $\tau_y = d(f_1 - f_2)$, where f_1 and f_2 are thrust forces applied by the left and

right thruster, respectively. We define the inputs to be

$$\mathbf{u} = [\tilde{\tau}_y, \tilde{f}_t]^T, \quad (3.3)$$

where $\tilde{f}_t$ is any change in thrust relative to the baseline needed to compensate exactly for gravity, scaled to have units of acceleration (ms^{-2}), that is $\tilde{f}_t = f_t/m - g$. Similarly $\tilde{\tau}_y = \tau_y/J$ is the normalized input torque that has units of angular acceleration (rad s^{-2}).

Python-based simulation environment Many elements of our analysis are facilitated by simulation. We created a simulated NAT robot using Python and the Python Control Systems Library [67] using a fixed-step Euler integrator at 200 Hz to solve the 2D version of the Euler-Lagrange dynamics equations in Equation (3.2), the linear feedback controller in Equation (3.20) and the Kalman Filter dynamics in Equation (3.15) in the x - z plane. All software used to create the figures is released as open source on Github using a persistent Digital Object Identifier at <https://doi.org/10.5281/zenodo.7324484>.

For the AO sensor suite, which includes a camera, we additionally constructed a simple visual environment, camera model, and added wind input. We used a ray casting algorithm to read pixel luminance values from a 1D visual image textured on the ground, as depicted in Figure 3.4. The texture is a single line of pixels extracted from a photograph of an outdoor scene. The optic flow measurement Ω_m of our camera was calculated using Equation (3.9) over an array of 35 luminance readings arrayed evenly over a region $\gamma = (-15^\circ, 15^\circ)$ directly below the robot, captured at the update rate of 200 Hz. At the start of the simulation, we computed the value of c in Equation (3.9), which is a measure of the average gradients in the projected image, using $c = -(\mathbf{l}_s^\top \mathbf{l}_s)^{-1}$.

The weighting matrices for the Kalman Filter were derived as follows. The variance of noise n_a in the airspeed estimate is the square of the RMS noise derived in Figure 3.3. We manually tuned the optic flow noise to be much higher than observed in simulation, to provide for conservative performance. Disturbance magnitudes were hand-tuned. For the LQR controller, thrust and torque effort were balanced approximately by power used; state

cost was manually tuned. We computed the gains K and L using the Python Control System Library [67]. Values for these matrices were the same for all simulations. In practice it may be desirable to incorporate an integral action term on θ to reject disturbances from a wire tether or manufacturing irregularity [11, 52].

Robot platform We performed tests on a small (palm-sized) quad-rotor helicopter (Crazyflie 2.1, Bitcraze, Sweden) in an indoor environment, both in wind-free conditions and in the presence of a fan. We collected sensor information over Bluetooth from the helicopter as it flew. The helicopter was equipped with a downward-facing rangefinder, gyroscope, and optic flow camera (the GRO sensor suite) that it used to estimate its velocity and position using an Extended Kalman Filter [68]. A careful analysis of its open-source software, confirmed by its author, shows that it does not use the x - and y - axis measurements of its accelerometer for state estimation in free flight. Our airspeed measurement assumes that $b = 13.2 \times 10^{-3}$ Nm/s, which was empirically tuned and is in a similar range to the 6.6×10^{-3} Nm/s estimated in a system identification effort [94]. We used $m = 30$ g from the datasheet of the device.

For the direct airspeed measurements, no Kalman Filter was used, and the fan was off (Figure 3.5A).

For full state estimation including wind speed estimation (Figure 3.5B), we processed sensor data offline in the Python Control Systems Library [67] using our Kalman Filter. We did not have information about the helicopter’s throttle or torque commands and so assumed they are zero-mean disturbance noise. Experiments consisted of flying the helicopter 1–2 m downwind of a 50 cm diameter box fan at a specified height $z_d = 1$ m. After the fan was powered on, the robot was observed to move downwind a short distance (10–20 cm) before stabilizing to a new location. We ran the Kalman Filter at the maximum data rate that could be achieved over the Bluetooth communication channel (20 Hz). We used the raw optic flow readings from optic flow sensor (PMW3901, Pixart Imaging, Inc., Hsinchu, Taiwan). The manufacturer does not disclose the optic flow algorithm this sensor uses.

3.3 Result

The sensor suite used on the smallest drone yet to perform sensor-autonomous stable hover uses a sensor suite we will term GRO. It combines a gyroscope (which measures angular velocity), a downward-facing rangefinder (which measures distance to the ground), and a downward-facing optic flow camera (which measures angular velocity of visual motion below the aircraft). [68, 95, 96] (Figure 3.1D). Ref. [68] describes the state estimator used on this palm-sized drone (Crazyflie, Bitcraze, Sweden) but with a different sensor suite that uses external position information. That system was subsequently updated use GRO. The rangefinder in GRO can use sonar [95], RADAR, or LIDAR (distance-finding using the time to reflect for light)[68]. It can be shown using the observability criterion [97] that GRO can observe a state given by $\mathbf{q} = [\theta, \omega, v_x, z, v_z]^\top$ [98] (see Figure 3.2). The key downsides of GRO are excessive weight and power consumption. The gyroscope must continually vibrate, making it heavier and consume more power than an accelerometer, which uses a passive mechanical element. After decades of refinement, the lightest commercially-available is the TDK InvenSense ICM-20600, which weighs 15 mg and consumes at least 3 mW. Similarly, rangefinders consume energy because they must emit energy: the lightest (ST Microelectronics VL53L0) weighs approximately 20 mg and consumes 6 mW [99]. Together, the components of a GRO system like the palm drone helicopter (Figure 3.1D) [68] would weigh 134 mg and consume approximately 21 mW (Table 3.2). This far exceeds the capability of a NAT robot.

3.3.1 Sensor suite description

Here we propose a dramatically lighter and lower power alternative to the gyroscope-rangefinder-optic flow (GRO) suite. Our system, which we call AO, consists only of a 3-axis accelerometer and an optic flow camera. These two devices are already available commercially in packages compatible with NAT scale (Figure 3.1). The accelerometer, mCube MC3672 (mCube Inc., San Jose, CA USA) measures approximately 1.1 mm square, weighs 2.0 mg,

and consumes only about $20\ \mu\text{W}$ operating at 210 Hz. As for a camera, the commercially-available OVM6948 (Omnivision Inc., Santa Clara, CA USA) is light enough, measuring only $0.65\times 0.65\times 1.2\ \text{mm}$ and weighing approximately 1 mg including its multi-element lens. This camera consumes approximately 25 mW, exceeding the 1 mW NAT robot power budget. Imaging chips with suitable resolution and size have been demonstrated in the lab, however. A 128×128 pixel camera, operating at 10 frames per second has been created that consumes only $1\ \mu\text{W}$ [100]. Subsequent refinements made a more practical sensor with improved dynamic range and frame rate, at the expense of greater power usage, resulting in a 64×40 sensor with 80 dB dynamic range and $30\ \mu\text{W}$ power consumption at 80 frames per second [101]. These square sensors measure approximately 1 mm on a side and, through wafer thinning, can be made extremely lightweight. Commercial imagers now have the necessary efficiency: the Himax HM series (Tainan City, Taiwan) consumes 40 nW/pixel at 60 frames per second (Figure 3.7), suggesting a power usage of approximately $49\ \mu\text{W}$ for a variant with a 35×35 imaging surface. A pinhole [102] or small spherical [89, 103] lens could also satisfy mass constraints. Compared to GRO, AO cannot observe altitude using a linear observer, but we discuss nonlinear methods in the conclusion that could be used to overcome this.

An airspeed sensor using an accelerometer The fruit fly uses a sense of airspeed derived from the Johnson’s Organs of its antennae, which detect deflections in the antenna induced by wind [104]. Wind sensors that are small enough or could conceptually be reduced in size to suit a NAT robot have been previously demonstrated, but are not available off-the-shelf [105–109].

We build on previous work that used accelerometers to improve drone state estimation [110], and a later nonlinear observability analysis that showed that an accelerometer alone can estimate attitude and velocity [111]. Here we show that accelerometer performance for this task improves with diminishing scale, promising route to attaining a state estimator compatible with the SSWaP constraints of a NAT robot.

Our approach uses an accelerometer to sense airspeed on a hovering robot. Note first

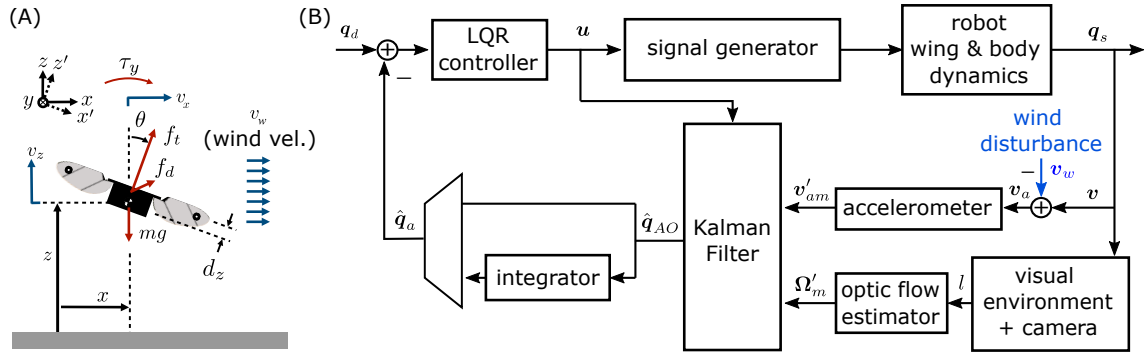


Figure 3.2: Dynamics and flight control architecture overview. (A) Diagram of parameters that describe the dynamics of our model for a small hovering robot. (B) The proposed feedback control architecture is LQG (Linear Quadratic Gaussian), consisting of a Linear Quadratic Regulator for a controller and a Kalman Filter estimator. The sensor suite includes an accelerometer-based drag sensor that measures airspeed v_{am} and a camera that provides a measurement Ω_m of the rate of optic flow. To estimate the states x , y , and z , the estimates of v_x , v_y , and v_z , respectively, are numerically integrated.

that an accelerometer does not measure acceleration directly. Instead it senses the “specific acceleration,” which is the difference between its acceleration and acceleration due to gravity. Formally, the specific acceleration is $\mathbf{a}_s = \frac{1}{m}(\mathbf{f} - \mathbf{f}_g)$, where m is the mass of the rigid body to which the accelerometer is attached, $\mathbf{f}$ is the sum of all forces acting on it, and $\mathbf{f}_g$ is the force of gravity. We assumed that the accelerometer is positioned near the aircraft’s center of mass so that Coriolis accelerations are negligible. Now consider the accelerometer’s reading in free-fall. In the absence of air drag it reads is zero because the only force is gravitation, giving $\mathbf{a}_s = \frac{1}{m}(\mathbf{f}_g - \mathbf{f}_g) = \mathbf{0}$, regardless of orientation [111]. Now incorporate air drag while still in free-fall. The accelerometer reads a nonzero value due to the aerodynamic drag force vector $\mathbf{f}_d$, but gravitation cancels out as above, leaving $\mathbf{a}_s = \frac{1}{m}\mathbf{f}_d$. To model aerodynamic drag, define the airspeed as the difference between flight velocity $\mathbf{v}$ and wind velocity $\mathbf{v}_w$ (Figure 3.2),

$$\mathbf{v}_a = \mathbf{v}_w - \mathbf{v}. \quad (3.4)$$

We can use this information to estimate the airspeed if we have a model for how air drag maps to airspeed, that is, we know $\mathbf{f}_d(\mathbf{v}_a)$, for example from wind tunnel tests [112]), For a small flapping-wing robot, a number of lines of evidence point to a linear model

$$\mathbf{f}_d(\mathbf{v}_a) = b\mathbf{v}_a \quad (3.5)$$

being a very good approximation for aerodynamic drag on flapping-wing hovering aircraft and flies [7, 53, 55, 92, 113, 114]. The quantity b has been measured in a wind tunnel for both forward and lateral wind on the Robobee to be 2×10^{-4} Ns/m [55]. It is 1.1×10^{-7} Ns/m for the 1 mg fruit fly in forward free flight, estimated by measuring flight velocity changes in response to wind gusts on flies whose wind sensing apparatus was ablated [7].

Finally, consider now that the aircraft to which the accelerometer is attached is subject to one additional force, a thrust force $\mathbf{f}_t$ caused by, for example, flapping wings. Then total specific acceleration is given by $\mathbf{a}_s = \frac{1}{m}(b\mathbf{v}_a + \mathbf{f}_t)$.

Now rearrange this equation and express it body-attached coordinates, which are denoted

with a $(')$ to distinguish from world coordinates. The airspeed is given by

$$\mathbf{v}'_a = \frac{m}{b} \mathbf{a}'_s - \frac{1}{b} \mathbf{f}'_t. \quad (3.6)$$

This shows that the airspeed can be measured if the quantities m and b are known, the controller output $\mathbf{f}'_t$ is known, and $\mathbf{a}'_s$ is measured by the accelerometer. This formulation permits airspeed measurements even for aircraft that can actuate forward and lateral forces, which flapping wings are likely capable of [115], as well as advanced rotorcraft. This indicates that the accelerometer is nearly interchangeable with a flow-based sensor such as a whisker/antenna [106, 107] or Pitot tube, provided controller outputs are well-known.

Here, without loss of generality, we assume thrust only acts directly in line with the aircraft's z -axis, that is, $\mathbf{f}'_t = [0, 0, f'_t]^\top$, where f'_t is usually nearly equal to the gravitational force mg . In component form, the airspeed measurement is then

$$\mathbf{v}'_a = \left[\frac{m}{b} a'_{s_x}, \frac{m}{b} a'_{s_y}, \frac{m}{b} a'_{s_z} - \frac{1}{b} f'_t \right]^\top. \quad (3.7)$$

Efficient optic flow estimation using autocorrelation The other sensor modality we include is optic flow from an onboard camera, which measures the speed of visual motion. In the section on state estimation below, we show that a downward-facing optic flow camera can provide additional information about the rate of lateral motion that is necessary to estimate wind.

Here, we begin by considering the computations needed to estimate optic flow from pixel luminance readings taken by a camera attached to the robot. Suppose the camera collects luminance readings $l(\gamma, t)$ (lux) at angles γ across a visual field. (This may be expanded to a two-dimensional surface by incorporating a second Euler angle β , or by specifying a vector direction $\mathbf{s}$ on the unit sphere). The simplest methods to estimate optic flow entail operations on the spatial luminance derivative $l_\gamma = \frac{dl}{d\gamma}$ (lux/rad, or the spatial gradient $\nabla_s l$ in 2D) and the temporal luminance derivative $l_t = \frac{dl}{dt}$ (lux/s) [69]. In practice, the spatial derivative l_γ must be calculated by subtracting luminance readings at two nearby angles γ_1 and γ_2 separated by $\delta\gamma$, such as at adjacent camera pixels, according to $l_\gamma = \frac{1}{\delta\gamma}(l(\gamma_2) - l(\gamma_1))$. The

time derivative is calculated by subtracting successive luminance readings (camera frames) at times t and $t - \delta t$, according to $l_t = \frac{1}{\delta t}(l(t) - l(t - \delta t))$, where δt is the time interval between frames. Assuming the shape of the luminance image remains fixed between frames but moves by a small amount, the measured optic flow along one dimension for a single pixel pair can be computed according to $\Omega_m = -l_t/l_\gamma$. This is a special case of the Lucas-Kanade method [63], which is a simple and accurate method to estimate the average 2D optic flow in an array of pixel readings [69].

In the 2D case, arrays of derivatives are constructed in which each entry corresponds to a pixel in the camera: $\mathbf{l}_s$ consists of pairs of spatial derivatives along two linearly-independent directions (e.g. $[l_\gamma, l_\beta]$), and $\mathbf{l}_t$ consists of temporal derivatives for each pixel. The Lucas-Kanade method for both 1D and 2D estimates optic flow in a least-squares sense according to

$$\Omega_m = -(\mathbf{l}_s^\top \mathbf{l}_s)^{-1} \mathbf{l}_s^\top \mathbf{l}_t. \quad (3.8)$$

In pursuit of minimizing the computation required to estimate optic flow, we consider a simplification inspired by biology that been previously proposed [8] that requires fewer operations, given by

$$\Omega_m = c \mathbf{l}_s^\top \mathbf{l}_t, \quad (3.9)$$

where c is a scalar constant. Because it entails only multiplication, it is often called “autocorrelation,” or “a correlator” for short. In Lucas-Kanade, c is computed each frame according to $c = -(\mathbf{l}_s^\top \mathbf{l}_s)^{-1}$. This more than doubles the multiply-add operations, as well as requiring a division operation to invert the 2×2 matrix. Equation (3.9) is a slight variation on the Hassenstein-Reichardt (H-R) model for insect optic flow processing [116] (see Supplementary Material).

Observing attitude, velocity, and wind speed

We show that the accelerometer-only sensor suite is able to estimate the aircraft attitude and velocities. Then we incorporate the optic flow sensor, creating the AO sensor suite that

can additionally estimate wind speed v_w .

Dynamics model We describe a dynamic model for the aircraft that is an integral part of our Kalman-based state estimator and closed-loop simulation. Figure 3.2 depicts forces acting on a flapping-wing robot such as a NAT robot or the Robofly, but the proposed model can apply to almost any small hovering aircraft, including multi-rotor drones and EHD-based thrusters [54]. For a flapping-wing or rotor-actuated aircraft, air drag is approximately linear with airspeed [7, 55, 117] (Equation (3.5)). For non-flapping-wing devices near hover conditions such as EHD-actuated aircraft [50, 54], drag is dominated by inertial flow, which varies as the square of airspeed instead of linearly with it. For these, a reasonable linear model has $b = 0$.

For control and estimation, linearity of airspeed drag results in a good linear approximation of the vehicle's dynamics [53, 55, 92]. For a state given by

$$\mathbf{q}_s = [\theta, \omega, x, v_x, z, v_z, v_w]^\top, \quad (3.10)$$

and inputs being the body- z -axis translational acceleration and body- y -axis angular acceleration (Equation (3.3)), the linearization of the dynamics in (3.2) has the form $\dot{\mathbf{q}}_s = A\mathbf{q}_s + B\mathbf{u}$; $\mathbf{y} = C\mathbf{q}_s$, where the Jacobian linearization is taken at $\theta = \omega = v_x = v_z = 0$, $f_t = mg$. The position $\mathbf{p} = [x, z]^\top$ has no effect on the Jacobian. The resulting matrices are given by:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{c}{J} & 0 & -\frac{b}{J}d_z & 0 & 0 & \frac{b}{J}d_z \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ g & -\frac{b}{m}d_z & 0 & -\frac{b}{m} & 0 & 0 & \frac{b}{m} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{b}{m} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \quad B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}. \quad (3.11)$$

This linear dynamical system exhibits the growing oscillatory instability that is characteristic of small flapping-wing vehicles [49, 92].

Accelerometer-only observability As above, for simplicity of exposition, we consider a simplified system consisting only of planar, 2D motion as depicted in Figure 3.2. The accelerometer system output is defined as the body x - and z -axis components of the accelerometer’s airspeed measurement (Equation (3.7)), given by

$$\mathbf{y}_A = \begin{bmatrix} v'_{ax} + n_a \\ v'_{az} + n_a \end{bmatrix}, \quad (3.12)$$

where each n is sensor noise added to the reading, which we assume is zero-mean Gaussian white noise. The state vector to be estimated is given by

$$\mathbf{q}_A = [\theta, \omega, v_x, v_z]^\top. \quad (3.13)$$

The accelerometer-only sensor suite is not able to estimate with wind speed v_w , so we assume it is zero and leave it out of the state. The Jacobian linearization of the measurement model (Equation (3.12)), taken at hover, is given by

$$C_A = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \quad (3.14)$$

The dynamics matrix associated with this estimator, A_A , is the A matrix in Equation (3.11) with rows and columns associated with x , z , and v_w removed. The observation matrix $[97]$ $obsv(A_A, C_A)$ is full rank, indicating that the entirety of the state vector $\mathbf{q}_A$ can be observed in a neighborhood of upright hover. Given that the system can be observed, a Kalman Filter-based linear estimator with dynamics of the form

$$\dot{\hat{\mathbf{q}}} = A\hat{\mathbf{q}} + B\mathbf{u} + L(\mathbf{y} - C\hat{\mathbf{q}}), \quad (3.15)$$

can be constructed to estimate the state of the system. In the accelerometer-only sensor suite, the state to be estimated is $\mathbf{q}_A$ (Equation (3.13)).

Using optic flow to additionally observe wind velocity We added a downward-facing camera (pointed in the negative of the body z direction, that is, along $-z'$) to provide a second

measure of lateral velocity that is not affected by wind, unlike the accelerometer. The optic flow camera uses either autocorrelation or Lucas-Kanade to provide a measurement Ω_m of the true optic flow Ω . Formally, optic flow is the time rate of change of a unit vector $\mathbf{s}$ pointed in the direction of a given visual feature as it moves across the visual field [62].

It can be shown that the x' -component (defined in the body frame) of the optic flow sensed by a downward-facing camera is given by [62]

$$\Omega = \omega - \frac{\cos \theta}{z} v'_x. \quad (3.16)$$

This shows that the sensor provides information about the body-frame lateral velocity v'_x , but its output is also affected by other state variables ω , θ , and z . Under conditions when $\omega = \theta = 0$ (as would occur during stable, level flight), the optic flow is simply $\Omega = v_x/z$, which can be used to estimate velocity if the altitude z is known [85]. But this is a restrictive assumption, and current barometric altimeters are too heavy and power hungry for a NAT robot (ST, Bosch, Omron, Murata, and TDK only market $>2 \times 2$ mm packages, e.g. the 2×2 Bosch BMP390 weighs 6.3 mg and consumes 1.1 mW at 50 Hz). The observability criterion given below shows that even under more general conditions, when combined with the accelerometer, the additional information provided by optic flow can be used to estimate and reject the effect of a wind disturbance.

The output “measurement model” for the combined accelerometer-optic flow (AO) sensor suite is given by

$$\mathbf{y}_{AO} = \begin{bmatrix} v'_{ax} + n_a \\ v'_{az} + n_a \\ \Omega + n_o \end{bmatrix} \quad (3.17)$$

where each n is zero-mean Gaussian white sensor noise.

We assume the wind velocity v_w only acts in the world x -direction, which is why it is only a single state. To estimate this quantity, we introduce an augmented state that includes the wind velocity,

$$\mathbf{q}_{AO} = [\theta, \omega, v_x, v_z, v_w]^\top. \quad (3.18)$$

The optic flow measurement depends on altitude, so we take the Jacobian linearization at a specific desired altitude z_d , giving

$$C_{AO} = \begin{bmatrix} 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & -\frac{1}{z_d} & 0 & 0 \end{bmatrix}. \quad (3.19)$$

The Observability Criterion indicates that $\mathbf{q}_{AO}$ is observable, including the wind velocity. As above, we use the Kalman Filter (3.15) to provide the estimate $\hat{\mathbf{q}}_{AO}$.

For some tasks, it is desirable to stabilize to a desired position. To provide a coarse position estimate for use in control, we numerically integrated the estimates $\hat{v}_x$ and $\hat{v}_z$ according to the last line in Equation (3.21) to obtain an augmented state $\hat{\mathbf{q}}_a$ that includes $\hat{x}$ and $\hat{z}$ estimates. These two states are not technically “observable”, meaning their estimates will slowly drift over time as sensor noise numerically accumulates. But their presence allows, for example, performing lateral maneuvers of a desired distance.

State estimator and flight controller

Anticipated sensor noise plays an important role, so we formulate the feedback system as an LQG (Linear Quadratic Gaussian) control problem. The system is under-actuated but the controllability matrix $\text{ctrb}(A, B)$ [97] is full rank, showing that the system is controllable. We constructed a trajectory-following LQR controller to follow a (slowly-varying) desired reference trajectory $\mathbf{p}_d(t) = [x_d(t), z_d(t)]$ using the controller

$$\mathbf{u} = K(\mathbf{q}_d - \hat{\mathbf{q}}). \quad (3.20)$$

The combination Kalman Filter and LQR controller is known as an LQG regulator and is diagrammed in Figure 3.2.

The LQR controller includes an additional component to compensate for wind. To move laterally, LQR controller effectively makes the robot perform “helicopter-like” control, tilting so that the thrust from the wings $\mathbf{f}_t$ takes on a lateral component (Figure 3.2). Assuming

	NAT robot	palm drone	units
mass (m)	10	30,000	mg
air drag coefficient (b)	49.4×10^{-6}	13.2×10^{-3} [94]	Ns/m
acceleration per unit airspeed (b/m)	4.9	0.44	s^{-1}
noise (amplitude spectral density)	not provided	175	$\mu g \text{ Hz}^{-\frac{1}{2}}$
RMS accel. noise, 100 Hz (σ)	4.4	1.75	milli-g
RMS accel. noise, 200 Hz, in flight	0.38	0.15	m/s^2
RMS airspeed noise (σ_a)	0.077	0.34	m/s

Table 3.1: Small scale allows the NAT robot to sense airspeed with lower noise than a palm drone despite more noise in the sensor itself.

lift balances gravity, the lateral thrust is $mg \sin \theta \approx mg\theta$, which can be produced through suitable choice of setpoint θ . In the presence of a non-zero wind estimate $\hat{v}_w$ (only possible with the AO sensor suite) the torque and attitude commands are altered to compensate for the added translational drag force according to $\theta_d = \theta_d - \frac{b\hat{v}_w}{mg}$ and added torque according to $\tilde{\tau}_y = \tilde{\tau}_y - \frac{d_z b \hat{v}_w}{J}$.

Noise scaling in accelerometer-based wind sensing

A key concern in the design of the NAT robot’s sensor suite is that high sensor noise magnitude could result in sluggish performance in the estimator. Or worse, instability if the control loop is operated at a low rate. Typically, smaller and lower-power sensors have lower performance. For example, the 2 mg mCube MC3672 accelerometer on the NAT robot has a $2.5\times$ higher root mean square (RMS) noise than the 30 mg Bosch BMI088 in the drone (Table 3.1). Our results show that despite this, our conceptual NAT robot senses airspeed with a much lower noise magnitude than larger aircraft because of favorable scaling physics. As scale reduces, the acceleration per unit airspeed, which is parameterized by the ratio b/m ,

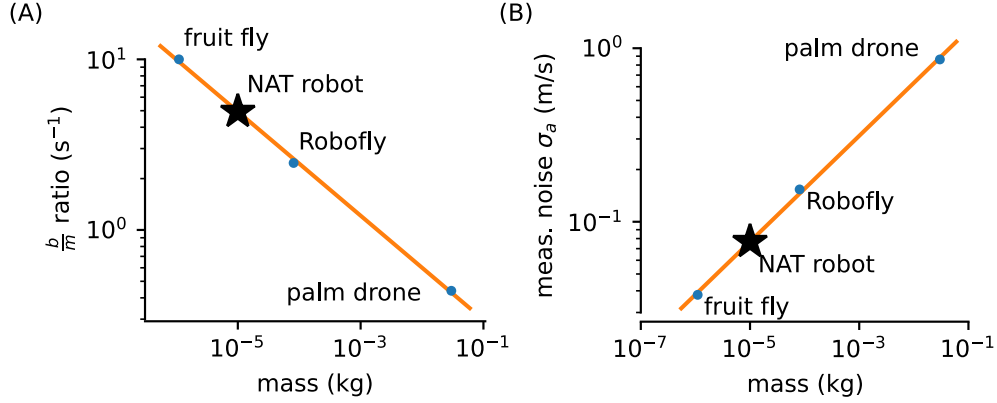


Figure 3.3: Scaling laws in accelerometer-mediated wind sensing reduce the airspeed measurement noise as size reduces. (A) The ratio of damping to mass varies approximately with the inverse square of mass, with a least log-error exponential fit giving $\frac{b}{m} = 0.15m^{-1.9}$. (B) This results in reduced airspeed measurement noise σ_a as size diminishes.

increases as a result of decreasing wing loading with scale [118]. It follows a roughly inverse quadratic trend $\frac{b}{m} = 0.15m^{-1.9}$ over nearly five orders of magnitude (Figure 3.3A). When used as an airspeed sensor, the accelerometer's sensor noise standard deviation is given by $\sigma_a = \frac{m}{b}\sigma$, where σ is the standard deviation of the acceleration measurement. The noise magnitude diminishes as scale reduces (Figure 3.3B). Table 3.1 shows that despite its noisier accelerometer, a NAT robot with a scale-appropriate b/m ratio using the fit exponential given above has nearly $5\times$ lower airspeed sensing noise. Details are provided in the supplementary material.

Closed-loop visual flight control in simulation

Figure 3.4A shows a trajectory of the system in closed-loop using only accelerometer feedback (no vision), indicating the feasibility of trajectory control. The figure shows the effect of realistic conditions in which the initial condition of the estimator does not match the true state.

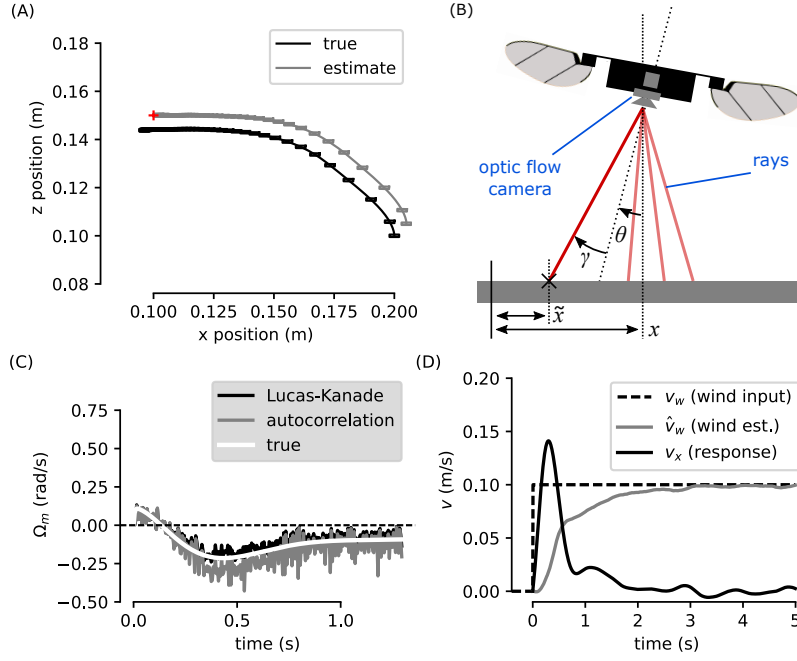


Figure 3.4: Simulation results. (A) Flight path of the accelerometer-only sensor suite along the x and z dimensions. The aircraft performs a lateral maneuver starting at the lower right to a setpoint location (red cross mark) on the upper left. The true trajectory is shown in black; the state estimate (grey) is subject to an error in the initial estimated position, but is able to nonetheless maintain attitude stability. Lines show the path of the center of mass and boxes depict the vehicle’s approximate size and attitude at 200 ms intervals. (B) Diagram of the camera model in the accelerometer-optic flow (AO) sensor suite: luminance readings $l(\gamma)$ are collected uniformly-spaced angles γ (red lines), which intersect a texture projected on the ground below at a position $\tilde{x} = x - z \tan(\gamma + \theta)$. (C) The correlator method produces similar output to Lucas-Kanade-based optic flow estimation, when following the same (arbitrary) trajectory. (D) The simulated AO system, when using autocorrelation-based optic flow estimation, is able to sense and react a wind disturbance. A step change in wind speed causes the aircraft to initially move in the direction of the wind. After about a second, the wind estimate has converged on its true value; the controller responds by commanding the robot to incline itself, resulting in a nearly perfect rejection of the disturbance.

Figure 3.4C shows that the proposed autocorrelation method has similar performance as Lucas-Kanade. Figure 3.4D and 3.8 show that our proposed system under closed-loop LQG control using autocorrelator-based optic flow estimation is able to reject a simulated step change in wind speed in slightly more than a second.

Power and mass

Tables 3.3, 3.4, and 3.5 tally the necessary operations per second and power required to perform flight control using the two optic flow methods. Lucas-Kanade and autocorrelation perform comparably (Figure 3.4) and operate within the required 1 mW power budget when executed on an off-shelf microcontroller, but the autocorrelation method uses approximately a third as much power. Table 3.2 provides the estimated mass and power consumption of the complete accelerometer-optic flow (AO) sensor and control system. Masses of each component were taken using a precision scale with a 0.1 mg resolution. We searched for the lightest gyroscope (TDK InvenSense ICM20600) and accelerometer (Mcube MC3672) commercially available and took estimated power numbers from the corresponding datasheets. Power usage of the AO camera, the optic flow camera, and rangefinder were measured in the citations provided. We have neglected the mass of a flex circuit and additional required discrete components, which tends to add an additional 25% [98]. Even when compared with a version of the drone’s gyroscope-rangefinder-optic flow (GRO) suite with the most recent (lighter) version of each sensor, it is substantially lighter and more power efficient. More details are provided in the supplementary information.

Physical robot validation

To demonstrate that the accelerometer itself can directly measure lateral velocity, we collected data from a 30 g palm-sized four-rotor helicopter (Crazyflie 2.1, Bitcraze, Sweden) (see Methods). No Kalman Filter was used in this test. Figure 3.5A shows that our accelerometer-based lateral velocity estimate corresponds well to the true lateral velocity provided by the palm drone’s full, GRO-based state estimator when undergoing lateral motions along both

	component	mass (mg)	power (μ W)
NAT robot sensor suite (AO)	accelerometer	2.0	20
	optic flow camera	1.0	100 [101]
	microcontroller	3.2	48
	total	6.2	167
palm drone sensor suite (GRO)	gyroscope	14	3,000
	optic flow camera	97	12,000 [98]
	rangefinder	20	6,000 [119]
	microcontroller, computation	2.6	371[98]
	total	133.6	21,371
improvement factor		22$\times$	128$\times$

Table 3.2: Our conceptual accelerometer-optic flow sensor suite is dramatically lighter and more efficient than the state of the art small avionics system.

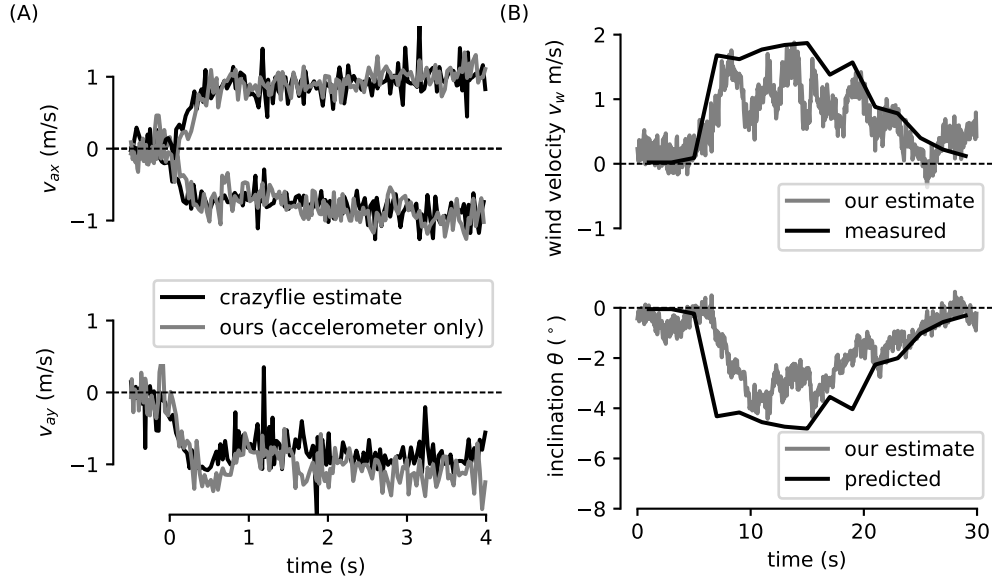


Figure 3.5: Flight tests on a 30 g palm drone show that our accelerometer-vision system can estimate vehicle state and wind speed. (A) We used the accelerometer alone (no Kalman Filter) to measure airspeed v_a in still air ($v_w = 0$), providing an estimate of lateral velocity $v = -v_a$ in both the x and y directions during a rapid lateral maneuver. Estimates are very close to the helicopter’s full GRO-based state estimate. (B) The full accelerometer-optic flow (AO) sensor suite and Kalman Filter estimator are able to estimate the wind velocity induced by a fan with reasonable performance, as well as the aircraft’s attitude.

the x - and y -directions. The inclination of the helicopter is small, just a few degrees, so we are able to compare the body-frame accelerometer-based velocity estimate to the world-frame state estimator’s lateral velocity, even though they are in different frames. The accelerometer-based wind estimate does not appear to be distorted by flight-induced vibrations.

We then constructed a Kalman Filter-based estimator to validate the ability of our accelerometer-optic flow (AO)-based avionics suite to estimate both wind and aircraft attitude. Figure 3.5B shows a comparison of the measured wind speed from a hot-bulb anemometer (Testo 405 hot-wire anemometer) against our system’s estimate. Important features of the wind speed measurement from the anemometer can be observed in our estimate, including onset time, peaks, and the slow decay of the wind after the fan was powered down. Our system’s attitude estimate also follows the prediction by a lateral force balance, $\theta = \arcsin\left(\frac{-bv_w}{mg}\right)$.

Comparison to the fruit fly’s wind gust response

Finally, we compare the performance of our system to behavioral data collected from the fruit fly *Drosophila melanogaster* subject to rapid, impulsive gusts of wind provided by an air piston while they flew along a wind tunnel with a 30 cm square cross-section [7]. A purely passive simulated mass with the same b/m ratio as the fly and our conceptual NAT robot (10.0 s^{-1}) initially moves with the gust before returning to zero velocity. By contrast, both the fly’s mean velocity response ($n = 92$) and our accelerometer-optic flow (AO) flight controller responded initially by moving more vigorously in the direction of wind than the passive particle (Figure 3.6 and 3.9). This indicates that both employ a feedback regulator that senses wind and attempts to regulate flight speed by minimizing airspeed error. Both also exhibit a delayed compensatory response that overshoots, leading to a positive value of v_x for a short period of time after $t > 0.3 \text{ s}$. This is the result of a compensatory visual feedback response that is slower because visual feedback is noisier than wind feedback for both our system, and likely the fly as well [120]. The dynamics are also affected by coupling between forward thrust and the rotational pitch dynamics.

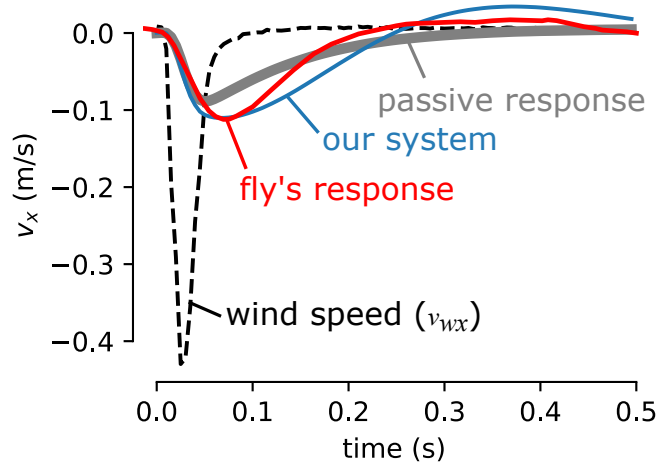


Figure 3.6: Our simulated accelerometer-optic flow (AO) 10 mg aircraft with, when operated in closed-loop (blue), has a velocity response to an impulsive wind gust that closely resembles that of fruit flies. For this simulation, we set $\frac{b}{m}$ to equal to the 10.0 s^{-1} measured in 1 mg fruit flies. Compared to a passive response of a particle suspended in air (grey), both our system (blue) and the fly (red) have similar dynamics, exhibiting a larger impulse response followed a delayed compensatory overshoot beginning after about 0.2 s. The red line is the mean velocity response of fruit flies flying along length of a wind tunnel ($n = 92$, from [7])

3.4 Discussion

The avionics for a 10 mg flying robot is subject to extreme size, speed, weight, and power (SSWaP) constraints. Here we introduce three innovations that drastically lower the mass and power of sensors required to hover while retaining sufficient bandwidth to stabilize. The first is to show that, because acceleration due to air drag on small aircraft is high, an accelerometer constitutes a light, precise, and power-efficient means to sense airspeed on smaller aircraft. This was inspired by previous work showing that an accelerometer can sense airspeed on drones [68, 110, 111]. The second is the use of a fly-inspired autocorrelation algorithm to reduce the computation needed for optic flow processing. Because our Kalman Filter observer also estimates angular velocity, it effectively de-rotates the optic flow measurement. Our entire flight controller operates exclusively using power-efficient multiply and add operations. And the third is to combine these elements to solve the key problems of attitude and wind estimation. By estimating these quantities, the robot is able to make rapid maneuvers and reject the effect of wind. Knowledge of the wind vector also facilitates higher level capabilities such as plume source finding [6]. The stable hovering platform and autocorrelator-based optic flow system described here serves as a platform upon which higher-level “reactive autonomy” tasks can be performed.

Our results show that by using only an accelerometer and camera, the avionics system mass is reduced more than 20-fold and power usage reduces by more than 100-fold compared to previous demonstrated hovering controllers. This work represents an important step toward realizing 10 mg aerial robots first conjectured in [72], as well as significantly reducing sensor mass and power for 100 mg robots like the Robofly [36, 57]. We validated our results on a 30 g palm-sized hovering rotorcraft. Our system was inspired by how the 1 mg fruit fly *Drosophila melanogaster* combines airspeed and visual feedback to navigate wind and exhibits an impulsive wind gust response that closely resembles it.

Compared to [85] and [121], our wind-vision sensor fusion does not require a power-hungry gyroscope to de-rotate the optic flow and can operate at hover rather than stable

forward flight. This makes it far more suitable for very small aircraft, which, like small insects and hummingbirds, must hover because of a low glide ratio [122, 123]. Both our work and [91] propose lightweight sensor systems based on optic flow to estimate attitude and velocity. Ours uses an accelerometer to additionally estimate wind speed, while [91] shows that nonlinear estimation can eliminate inertial sensors entirely in principle, though flight demonstrations required a gyroscope.

The present work is motivated by a larger narrative of advancements in ultra-tiny flying robots. Flapping systems of 1–5 mg have been demonstrated [124–126]. Actuators for small robots are largely electrostatic, requiring high voltages. A controllable coil-based voltage boost converter has been realized [127]; a long series of photovoltaic cells is another option [128]. Milligram batteries have been fabricated by 3D printing [129], photolithography [130], and laser microfabrication [131]. Recent results indicate it may be possible to dispense with the battery altogether using solar cells, which can be extremely thin and have a favorable surface-area-to-volume ratio as scale reduces [77].

The physical sensor suite proposed here satisfies proposed SSWaP constraints (≈ 6 mg, <1 mW, >200 Hz) using an off-the-shelf accelerometer combined with efficient imaging sensor technology that has been demonstrated in the literature, such as sub-100 μ W sensors [100, 101] and commercially in the Himax HM series. Such imagers could be combined with small optics such as the multi-element lens in the ≈ 1 mg Omnivision OVM6948 camera. Cameras with integrated optic flow estimation hardware [132, 133] and event cameras [134] have the potential to reduce both power and latency even further below what is possible with the camera-and-computer architecture considered here.

A linear observer is unable to estimate both the altitude and wind velocity using the accelerometer-optic flow sensor suite introduced here. This is in contrast to GRO, which is able to by virtue of its emissive (power-hungry) rangefinder [95]. A number of methods have been proposed in the literature that would not require additional hardware to control altitude [135]. The first is to take advantage of the ground effect, which is an increase in lift that occurs near the ground, to obtain an altitude equilibrium [84]. Another is to include

a feedback law that varies thrust f'_t , rather than inclination angle θ , in response to optic flow. This can regulate altitude through suitable optic flow setpoint [86, 136]. A third is to detect delay-induced feedback instability in an optic flow-mediated altitude controller [137]. A widely-varying altitude requires modification of our linear estimator to accommodate the associated optic flow nonlinearity in z in Equation (3.16). Two alternatives are an Extended Kalman Filter at the cost of an additional $\approx 200 \mu\text{W}$ [98], or to use a “gain-scheduled” Kalman Filter at negligible additional power.

We have neglected some non-idealities in physical accelerometers such as bias drift [138], which will manifest in a larger magnitude of lateral drift velocity. We anticipate that the reactive-autonomy layer can be used to compensate, or nonlinear methods [139]. For example, an omnidirectional camera could be used to stabilize a single position using visual servoing [140], avoid walls [85, 86, 141], or navigate confined spaces [45, 84, 86, 142, 143]. Chemical sensors can be added to seek chemical plume sources [6] or map them [88].

Our system responds similarly to the fruit fly to gusts of wind in flight. This indicates desirable real-world performance. Our results also provide insight into the mechanisms used by insects for their superlative flight abilities [144, 145]. An open question is how animals like bees and moths stabilize their flight without the gyroscopic halteres carried by dipteran flies [146]. One possibility is sensing twisting motions in the hindwings [147]. Our results suggest what we believe represents an alternative testable hypothesis. We suggest that it is possible that these animals could estimate and therefore stabilize their attitude by sensing airspeed using their wind-sensitive antennae [7, 104]. An airspeed sense mediated by these organs could in principle provide the exact same feedback as the accelerometer-mediated airspeed sensing described here.

Supplementary materials

Remarks regarding autocorrelation-based optic flow estimation The correlator given in Equation (3.9) is a slight variation on the Hassenstein-Reichardt (H-R) model for insect optic flow processing [116]. The classic H-R model predicts turning response of insects

to panoramic visual motion. It uses a first-order low-pass filter instead of a pure delay as part of the derivative computation l_t (above) [148, 149]. If the panoramic image is a pure sinusoid of amplitude A and spatial frequency ω_s moving at speed Ω in rad/s, it can be shown that the output of the correlator in Equation (3.9) approaches $\Omega_m = A^2\omega_s^2\Omega$ [150] as $\delta\gamma\omega_s$ approaches 0. This shows that the correlator’s output increases with increasing optic flow, but also depends on the spatial frequency and amplitude of the image. It was observations of these characteristics in insect behavior that led to introduction of the H-R model [148, 149].

In realistic, broadband images, the correlator responds to the averaged combined effects of the frequency and contrast of the image’s constituent Fourier spectra [150]. In Lucas-Kanade, the scalar c effectively normalizes out these effects by computing the square of the average spatial gradient of the image pixels. For many conditions of interest, however, c does not change quickly, hence we propose to either compute it only once at the outset, or on an intermittent basis so that it adds negligible additional computation load. Like Lucas-Kanade, a correlator can only provide a reasonable estimate if the image has contrast, rather than a uniform brightness.

Supplementary methods

Airspeed sensing noise The RMS accelerometer noise at 100 Hz is provided in the datasheet of mCube MC3672 (low-power mode). We calculated the equivalent quantity for the Bosch BMI088 used on the quad-rotor drone by assuming its noise amplitude spectral density (ASD) is flat (white). The root mean squared (RMS) noise σ for a flat bandwidth f is given by $\sigma = \text{ASD}\sqrt{f}$.

We observed a factor $\alpha = 6.2$ increase in RMS acceleration noise in the helicopter when in flight relative to the datasheet. This is likely the result of rotor and aerodynamics-induced vibration. We assumed that the same amplification factor applies for the NAT robot. To calculate the RMS airspeed noise σ_a [m/s] at the anticipated sampling frequency

of $f = 200$ Hz, we used the equation

$$\sigma_a = \alpha g \frac{\sqrt{f}}{\sqrt{100}} \frac{m}{b} \sigma,$$

where $g = 9.81$ is the gravitational acceleration of the Earth, and m and b are the mass and air drag damping coefficient of the vehicle's dynamics, respectively.

Dynamics model The dynamics given in Equation 3.2 describe the motion of a hovering aircraft like a robot fly or multi-rotor drone constrained to motion in a 2D plane. They are a subset of the full rigid body dynamics in space, which are given by the Euler-Lagrange equations:

$$\begin{aligned}\Sigma \mathbf{f} &= m \dot{\mathbf{v}} \\ \Sigma \boldsymbol{\tau}' &= \mathbf{J} \dot{\boldsymbol{\omega}}' + \boldsymbol{\omega}' \times \mathbf{J} \boldsymbol{\omega}' \\ \dot{\mathbf{R}} &= \mathbf{R} \boldsymbol{\omega}'^\times \\ \dot{\mathbf{p}} &= \mathbf{v}\end{aligned}\tag{3.21}$$

where $\mathbf{f}$ is any force acting on the vehicle, $\boldsymbol{\tau}'$ is any torque acting on the vehicle, $\mathbf{v}$ is the velocity and $\mathbf{p}$ is the position of the center of mass of the robot in world coordinates, $\boldsymbol{\omega}'$ is the angular velocity of the robot, $\mathbf{J}$ is the vehicle's moment of inertia, and $\mathbf{R}$ is the rotation matrix that relates vectors given in body-attached coordinates to world coordinates. For any vector $\mathbf{v}$ expressed in world coordinates, $\mathbf{v}'$ is the same vector expressed in body-attached coordinates. They are related by $\mathbf{v} = \mathbf{R} \mathbf{v}'$. Using the special property of rotation matrices that $\mathbf{R}^{-1} = \mathbf{R}^T$, we can also go in the other direction: $\mathbf{v}' = \mathbf{R}^T \mathbf{v}$. The quantity $\boldsymbol{\omega}'^\times$ is a 3×3 matrix that performs the cross product operation $\boldsymbol{\omega}' \times$.

Computation and power Table 3.3 gives an estimate of the computation load per pixel of the two optic flow methods. For each method, we tallied the number of clock cycles it takes to perform the primary mathematical operations given by Equations (3.8) and (3.9). We neglected edge effects and operations that only needed to be performed once (e.g. multiplication by a constant or division).

This constitutes an approximate measure of the amount of power required. A multiply-accumulate operation typically takes a single cycle for both fixed point and floating point units on common microcontrollers such as the Arm M4 (Arm Technical Reference Manual, available here). Additions also take a single clock cycle, but are much simpler and usually are accompanied by a multiply in a single multiply-accumulate instruction and so have been neglected. Lucas-Kanade requires a single, slower division operation (consuming approximately 10 cycles depending on processor) at the end as part of the 2×2 matrix inverse (Equation (3.8)), but this occurs infrequently and has been neglected.

Table 3.4 gives an estimate of the number of operations required per step for each of the two LQG controllers. For positive or negative unity entries in the A , B , and C matrices given above, we assume that a multiply operation is not required. The number of entries in the possibly-sparse LQG controller gains were estimated by assuming near-zero entries are zero.

Table 3.5 gives estimates of the instruction count and power draw of the computations required for the complete control system. In this table, we assumed that two separate copies of the LQG controller are operating simultaneously, one for each of the x - z and y - z planes, respectively. Based on our simulation, we found that it was possible to operate the controller at a 200 Hz update rate; slower update rates led to instability, likely a result of the increasing speed of rotational dynamics at small scale [78], resulting in instability.

We assumed that optic flow computations operated on luminance readings from a camera with a 35×35 square grid of pixels (1225 pixels), the same number used previously in a palm-sized rotorcraft [68]. These were used to estimate the optic flow along the body x' and y' directions independently by computing responses along rows and columns of the pixel grid, respectively. We assumed a general-purpose microcontroller such as the 2.6 mg Arm M4-based Max32660 (Maxim Integrated, San Jose, CA USA), which draws approximately $86 \mu\text{A}$ per MHz at 1.1 V, or $95 \mu\text{W}/\text{MHz}$. We assume it runs at 1 operation per clock cycle and goes into low-power sleep mode in a few microseconds at each time step after computations have completed.

	Lucas-Kanade	autocorrelation
multiplies per pixel (1D)	2	1
multiplies per pixel (2D)	6	2
RMS error (1D) (rad/s)	0.044	0.074

Table 3.3: A comparison of computation requirements and performance in our simulation of the Lucas-Kanade and correlator method

component	item	operations
controller	nonzero entries in K (Eq. (3.20))	6
	integral action computation	1
	total, controller	7
estimator	nonzero, non-unity entries in A	8
	nonzero, non-unity entries in B	0
	nonzero, non-unity entries in C_{AO}	1
	nonzero entries in L (Eq. (3.15))	9
	total, estimator	18
total (per LQG regulator)		25

Table 3.4: Estimated number of multiply-accumulate operations needed per update for the LQG regulator.

component	Lucas-Kanade	correlator
x' LQG controller	5000	5000
y' LQG controller	5000	5000
x' - y' optic flow estimation	1,470,000	490,000
total operations/s	1,480,000	500,000
computation power consumed (μ W)	141	48

Table 3.5: Estimated computation load (in operations/s and power) for combined optic flow and controller computations.

Supplementary figures

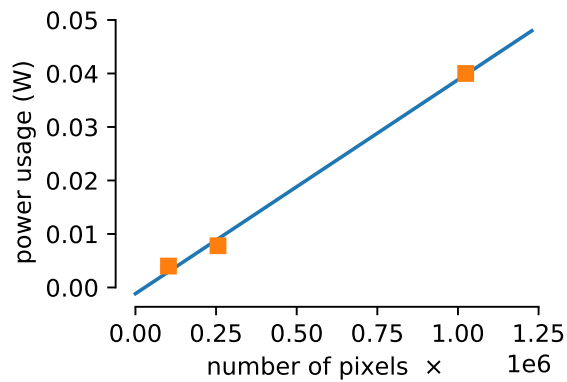


Figure 3.7: Power scaling per pixel for a commercial imaging sensor family (Himax HM series, Tainan City, Taiwan).

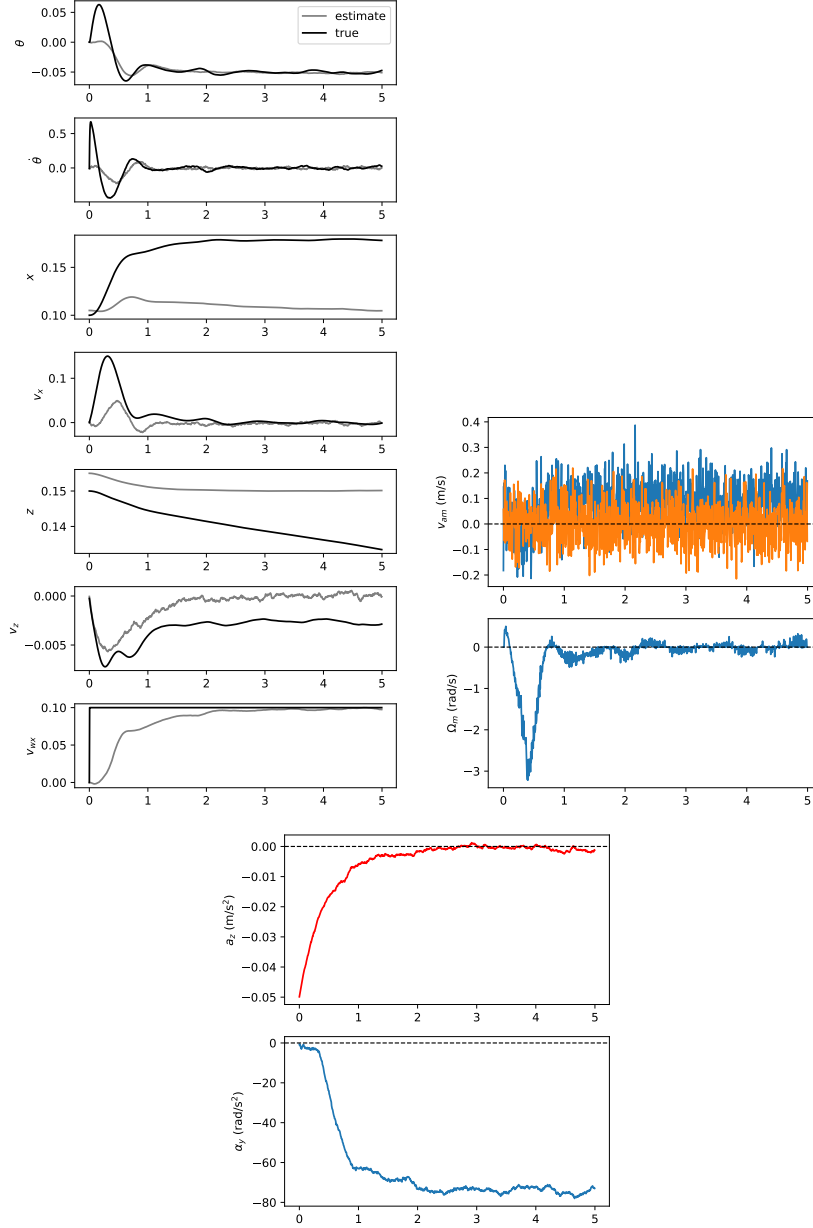


Figure 3.8: States, measurements, and controller outputs (where a_z is z -axis acceleration $\tilde{f}_t$ and α_y is the y -axis angular acceleration $\tilde{\tau}_y$) during response to step input gust. (Figure 3.4D) Units for the state are rad, rad/s, m, m/s, m, m/s, and m/s, respectively.

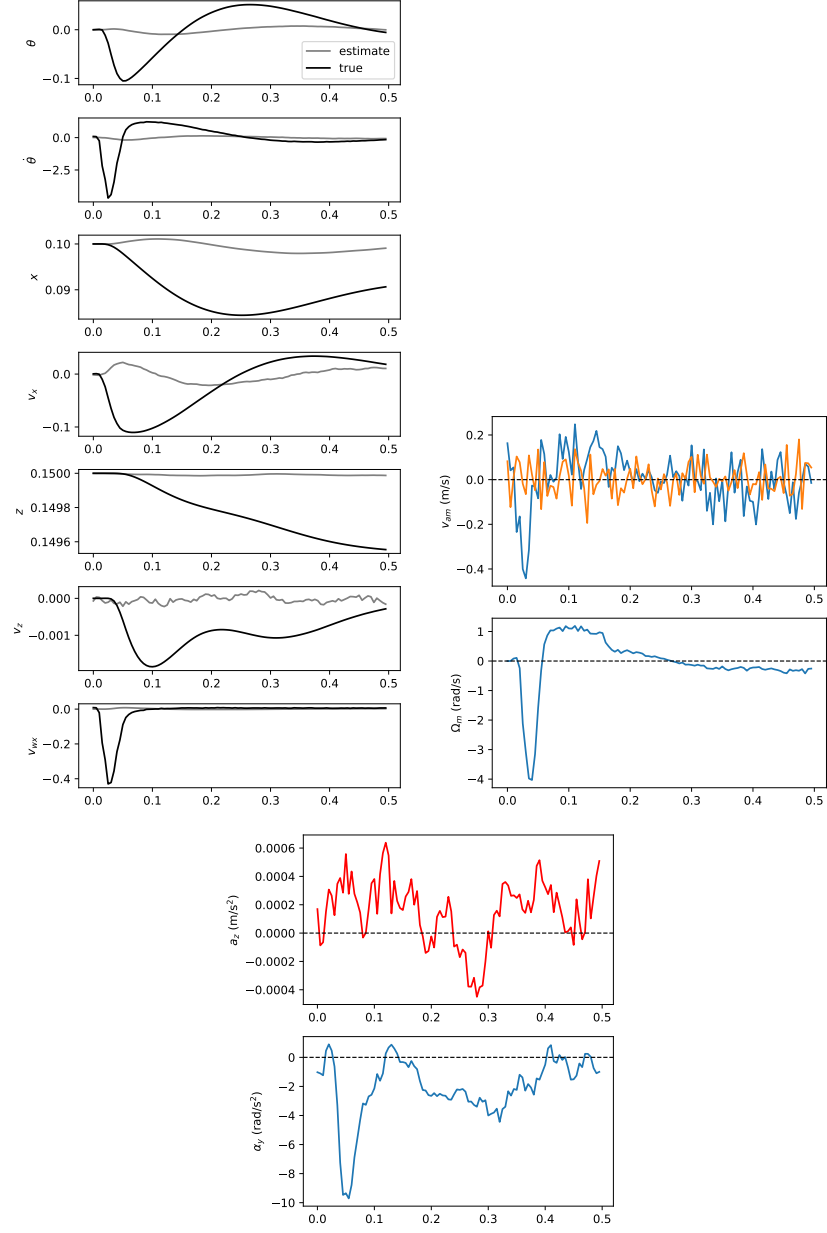


Figure 3.9: States, measurements, and controller outputs (where a_z is z -axis acceleration $\tilde{f}_t$ and α_y is the y -axis angular acceleration $\tilde{\tau}_y$) during simulated response to impulsive gust (Figure 3.6). Units for the state are rad, rad/s, m, m/s, m, m/s, and m/s, respectively.

Chapter 4

VISUAL CONFINED-SPACE NAVIGATION USING AN EFFICIENT LEARNED BILINEAR OPTIC FLOW APPROXIMATION FOR INSECT-SCALE ROBOTS

4.1 *Introduction*

Compared with the large robots, insect-scale robots have advantage of lower cost because the material cost is dramatically reduced and also they can perform many tasks better because of the small size such as locating the leaks of the gas of interest in dense piping infrastructure, space exploration with much lower cost, and replacing fixed-in-place air sensors in urban environments. However, the reason why the insect-scale robots are still in research stage rather than industry deployed is because as the scale reduces dramatically, the sensor feedback quality and energy available for computation also reduces drastically, which makes the realization of insect-scale robots challenging to have the similar performance as the larger robots. For example, larger drones can be equipped with several sensors such as Lidars, thermal sensors and the global positioning system (GPS), but for insect-scale robots, most of those sensors are either too power-hungry or too heavy. Also, especially for indoor environment, the GPS signal cannot be accurate enough [151]. One of our contributions is to solve the challenge for insect-scale robots to do navigation given speed, size, weight, and power (SSWaP) constraints.

Visual information has been used as an important input to robotic systems to do navigation [152], manipulation [153], and human-robot interaction tasks [154]. However, most of the algorithms are for large robots without tight SSWaP constraints. In nature, the fly is good at visual navigation under those constraints. Flies have omnidirectional eyes that can perceive visual information and then do navigation in surroundings [155]. Bio-inspired by

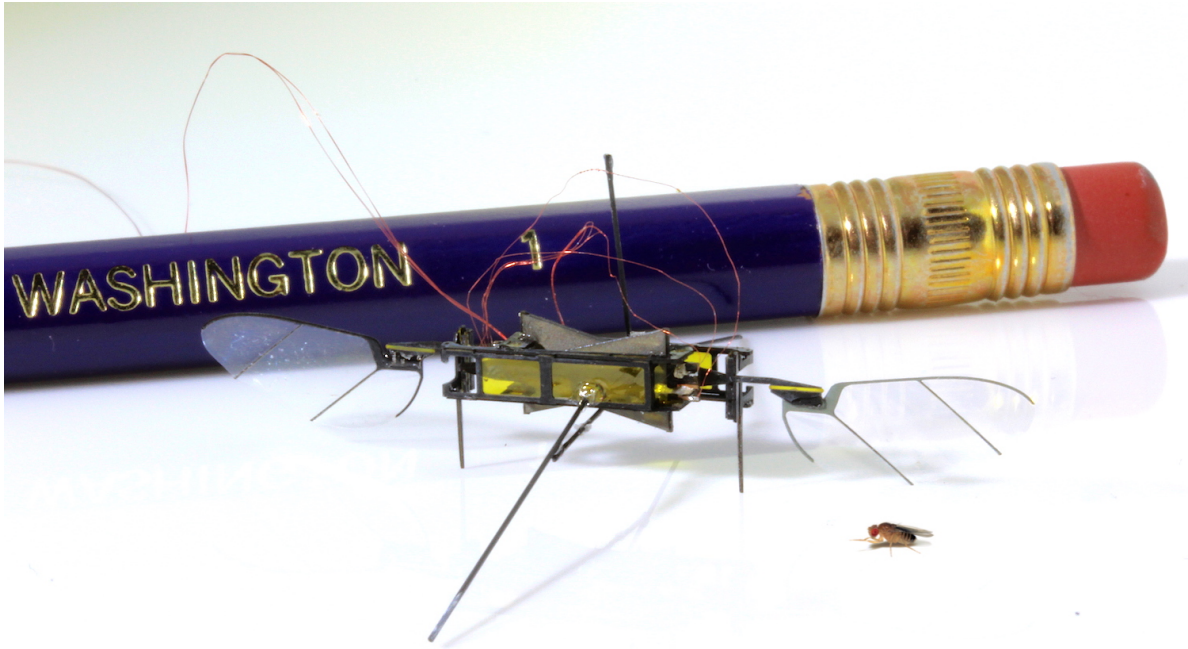


Figure 4.1: The 143 mg U. Washington Robofly (pencil in background for scale) weighs a bit more than a honeybee (101 mg). We aim to perform visual flight control with the very low compute power available on robots of this size and below.

flies, we do visual navigation with insect-scale robot with an omnidirectional camera using an optic flow estimation. Here, we are interested in a corridor following task (Figure 4.2) because it is the essential behavior required for navigating between obstacles in a cluttered environment. A reflexive corridor following controller could operate in real-time, freeing a high-level controller to pursue long range goals such as search or path planning.

Our proposed control architecture takes inspiration from fruit flies. Rather than using state estimation derived from a stored map (SLAM), we rely on optic flow, a measure of the velocity of motion of visual scenery as the robot or animal moves through it. Optic flow requires less computation and provides a crude measure of distances to obstacles [90]. This provides enough information to carry out many tasks, such as source seeking [6] and navigating through indoor and outdoor environments [44]. Lucas-Kanade [63] is a simple and

effective [69] method to estimate optic flow from raw pixels. To reduce computation power still further for application on a gnat robot [156], in this work we employ an insect-inspired simplification. Known as the Hassenstein-Reichard correlator [116], it entails only multiplying derivatives of pixel readings and predicts a number of aspects of insect behavior [149].

We assume our robot must be controlled by a computer carried onboard, rather than a remote computer to which data is transmitted wirelessly, for three important reasons. First and most importantly, many anticipated applications for small flying robots are indoors or in confined spaces, where a consistent wireless channel is not available. Second, even with a reliable channel, wireless transmission consumes excessive power. For example, consider the 71 mg Robofly (Figure 4.1), which weighs less than a honeybee and requires 60 mW to fly [57]. Transmitting low-resolution (160×120 pixel) video at 1–5 frames per second over low power Bluetooth consumes 4–18 mW [89]. Even at this low frame rate, which is too low for many control tasks, wireless transmission represents an excessive fraction of the power needed to simply stay in the air. And matters become far worse as scale diminishes to that of a gnat. Third, an important advantage to small robots is their potential to be deployed in large numbers; this advantage is largely nullified if they are forced share a crowded communication channel and an over-burdened server.

The platform we have in mind for the ensuing analysis and simulation is extremely small and power-constrained flying robots with the size of insects. Such robots, whether actuated by wings or other means such as electrohydrodynamic thrust, are dynamic and unstable as a general rule [49, 92, 157]. For this work, we build on parallel work [2] that introduces an ultra-lightweight sensor suite consisting of an accelerometer and a downward-facing optic flow camera to stabilize these unstable hover dynamics. The sensor suite provides an estimate, and therefore the ability for the robot to control, lateral velocity. This represents the first level of control autonomy, “sensor autonomy,” proposed in the hierarchy introduced in [43]. Here we are concerned with the next level up, in which the robotic agent senses and responds to the external environment. This is known as “reactive autonomy.”

We assume therefore that the task to be solved here is to provide an outer-loop controller

that provides control inputs in the form of desired lateral and rotational velocity values to the inner loop hovering controller. We also assume that the inner-loop controller is capable of controlling the heading velocity $\dot{\phi}$. On a larger drone equipped with a full inertial measurement unit (IMU), this would be done using a feedback control loop in which the angular velocity was measured using a MEMS gyroscope. However, we are targeting extremely small aircraft, which may not be compatible with the power draw (tens of mW) and mass (tens of mg) of current MEMS gyroscope technology. We assume that the $\dot{\phi}$ is instead measured using the downward-facing optic flow camera, trained on rotating imagery in a relatively straightforward extension of the work derived here.

Previous work has introduced a “biologically plausible” means to use motor babbling to learn a simple, bilinear model that approximates optic flow patterns observed by omnidirectional cameras during self motion purely by observing pixel readings and their derivatives. This approach has been used to stabilize attitude [8] as well as pose [140] in simulation and on physical robots [158], also known as “visual servoing.”

Here, our contribution is to show that a similar approach can learn how to stabilize a state of continuous motion down the length of a confined or cluttered corridor-shaped environment.

Some robot navigation works use the sensors like RGB-D camera [159] or Lidars [160]. In the environment where GPS cannot work well, a state estimation with the Lidar and IMU was proposed [161]. However, for insect-scale robots, those sensors are too heavy and power consumption is too high.

Most works in visual navigation are to create a map on surrounding environment using sensors, and simultaneously robots can localize itself in the map anytime. This is the process called Simultaneous Localization and Mapping (SLAM) [162][163]. A SLAM implementation for generating 3D map for unknown environment is introduced in [164]. However, SLAM is a heavy computation process and also require a significant memory, thus for insect-scale robots, it is not ideal to use SLAM to do navigation.

Recently, deep learning approach emerges in robot navigation tasks. Dorbala et al. [165]

proposed a deep learning approach using a convolutional neural network (CNN) to do corridor following task with wheeled robots. Saxena et al. [166] presented an end-to-end deep learning based approach for visual servoing by training CNN on colored images. However, deep learning approach only works when there are sufficient data for training and if the distribution of the test dataset is different from the training dataset, the deep learning approach will not work well. Because deep learning architecture usually has lots of parameters, it may consume significant computation power.

Given the SSWaP constraints, for insect-scale robots, researchers utilized bio-inspired approaches. Fuller and Murray [142] showed that insect-inspired visual autocorrelation can navigate a hovercraft robot through a corridor successfully. Censi et al. [8] proposed a bilinearly estimation approach using optic flow to stabilize visual attitude. To best of our knowledge, There is no research on navigating an insect-scale robot through a corridor, which is the main task of our paper.

4.2 Methodology

In this section, we will introduce the kinematics and autocorrelation-based visual control.

4.2.1 Kinematics and control

We developed a kinematics model for the corridor following task. The dynamics read:

$$\begin{cases} \dot{x} = u_x \cos \phi - u_y \sin \phi, \\ \dot{y} = u_x \sin \phi + u_y \cos \phi, \\ \dot{\phi} = u_\phi. \end{cases} \quad (4.1)$$

,where $u = [u_x, u_y, u_\phi]$ is the control input for the velocity in body frame. And $\dot{x}, \dot{y}$ and $\dot{\phi}$ are the velocity in world frame. To control the dynamics, we use a proportional controller:

$$u_\phi = K_{p\phi}(0 - \hat{\phi}). \quad (4.2)$$

We are mainly interested in getting an estimation $\hat{\phi}$.

4.2.2 Bilinear optic flow approximation

We would like to get estimation of $\hat{\phi}$ using pure optic flow information. We have the precise relation for how luminance intensity varies with time due to camera motion given by Eq. (4.3) [167]:

$$\dot{\mathbf{l}} = (\mathbf{s} \times \nabla_{\mathbf{s}} \mathbf{l}) \cdot \boldsymbol{\omega} + \mu(\mathbf{s}, \mathbf{p}) \nabla_{\mathbf{s}} \mathbf{l} \cdot \mathbf{v}, \quad (4.3)$$

where $\mu(\mathbf{s}, \mathbf{p})$ is the reciprocal of the distance to the visual element, known as the nearness. $\mathbf{s}$ is a continuous index ranging over the “sensual space” $\mathcal{S}$, which is a unit sphere in this case denoted as $\mathbb{S}^2$. Nearness depends on the geometry of the environment and the pose $\mathbf{p}$ (position and orientation) of the robot.

To simplify control, we consider a Taylor series approximation of Eq. (4.3), which is non-linear, in terms only of $\mathbf{l}$ and $\mathbf{u}$, that is, the luminance readings and the control inputs. The intent is to approximate the “average” behavior of Eq. (4.3) across the typical distributions encountered in our environment of interest. These include, in particular, the shape of the nearness function μ , the image contrast, and the pose of the vehicle. The first terms of the Taylor series are given by

$$\dot{\mathbf{l}} \approx A + B\mathbf{u} + C\mathbf{l} + \mathbf{u}M\mathbf{l}, \quad (4.4)$$

where $\mathbf{u} = [\boldsymbol{\omega}, \mathbf{v}]^T \in \mathbb{R}^p$ is the linear and angular velocity vectors of the robot ($p = 6$ for the 3D case; for 2D planar motion, $p = 3$). Assuming $\mathbf{l}$ must be discretized by, for example, a digital imaging surface, then $\mathbf{l} \in \mathbb{R}^n$, and $\nabla_{\mathbf{s}} \mathbf{l}$ is replaced with its discrete 1D approximation $\frac{d\mathbf{l}}{d\gamma} \approx \frac{l_{k+1} - l_{k-1}}{2\delta\gamma}$, which can be represented as a matrix operator $D\mathbf{l}$, so that $M \in \mathbb{R}^{p \times n \times n}$ is the product of some matrix times D . It can be shown that the first three terms, in terms of A , B , and C , are zero, leaving only the last term, which is bilinear in $\mathbf{l}$ and $\mathbf{u}$. The bilinear approximation can be written in terms of 2D matrices as

$$\dot{\mathbf{l}} = \sum_{i=1}^p M_i \mathbf{l} \mathbf{u}_i, \quad (4.5)$$

where the M_i are p different $n \times n$ matrices.

If the M_i 's in Eq. 4.5 above can be found, then they can be used in the to estimate the state of motion $\mathbf{u}$. The least-squares solution is

$$\hat{\mathbf{u}}_i = -((M_i \mathbf{l})^\top M_i \mathbf{l})^{-1} \mathbf{l}^\top M_i \dot{\mathbf{l}}. \quad (4.6)$$

This can be compared to the Lucas-Kanade method of optic flow estimation [63], but where the estimation is of a state geometrical motion rather than simple translational motion relative to a flat surface. It was shown in [8] and subsequent work that it is possible use a simpler estimator that eliminates the matrix inversion step, leaving the bilinear term

$$\hat{\mathbf{u}}_i = -c \mathbf{l}^\top M_i \dot{\mathbf{l}}, \quad (4.7)$$

where c is inverse of the average image contrast: $c = (\mathbb{E}_m\{||\nabla \mathbf{l}||_2^2\})^{-1}$. Eq. (4.7), when approximated using discrete pixels and when the derivative is computed at discrete time increments, is mathematically identical to a Hassenstein-Reichardt correlator in which the lag is a pure delay [8].

Previous work [8, 140, 158] showed that, with mild conditions on the shape of μ , it is possible to derive a controller that can locally drive the robot to a pose in which its luminance readings match those of a goal “snapshot” $\mathbf{g}$ using bilinear estimator. This is known as visual servoing.

For the case of 2D planar motion, for example, $\mathbf{u} = [\dot{\phi}, \dot{x}, \dot{y}]^\top$. To perform visual servoing, that is, driving the pose $\mathbf{p}$ to become coincident with a goal pose $\mathbf{p}_g$, a “goal” luminance snapshot, $\mathbf{l} = \mathbf{g}$ must be available. The proportional controller for a kinematic system (defined only by velocities) is simply $\mathbf{u}_i = \mathbf{l}^\top M_i (\mathbf{g} - \mathbf{l})$. This approach can also stabilize an inertial, dynamic system driven by forces and torques, where $\mathbf{u} = [\boldsymbol{\tau}, \mathbf{f}] \in \mathbb{R}^6$ (3D) or $\in \mathbb{R}^3$ (2D planar motion). A stabilizing proportional-derivative (PD) visual servoing controller has the form $\mathbf{u}_i = \mathbf{l}^\top M_i (\mathbf{g} - \mathbf{l} - k_p \dot{\mathbf{l}})$, where k_p is the proportional feedback gain [140]. In other words, the matrices M_i capture luminance correlations. That is, they encode how luminance readings change with time, and simultaneously and as a result, also the direction of the error gradient between two displaced images if the displacement is small.

4.2.3 Learning the M matrices

We learn the matrices by detecting correlations between pixel readings, their derivatives, and randomly-chosen motor inputs through “motor babbling.” We have $\hat{\mathbf{u}} = c\dot{\mathbf{l}}^\top M \mathbf{l}$. Then [8] shows that $\hat{\mathbf{u}}$ is an unbiased estimator of $\mathbf{u}$.

We would like to learn a matrix M to minimize the expected estimation error $\mathbb{E}\|\dot{\mathbf{l}}^\top M_i \mathbf{l} - \mathbf{u}\|_2^2$, for each component of $\mathbf{u}$, we can learn M_i according to the error function:

$$\mathbb{E}\{(\dot{\mathbf{l}}^\top M_i \mathbf{l} - \mathbf{u}_i)^2\}. \quad (4.8)$$

Given a particular training sample $(\mathbf{u}, \dot{\mathbf{l}}, \mathbf{l})$, M_i can be updated according to a stochastic gradient descent (SGD), which consists in minimizing Eq. (4.9) only with respect to the particular sample. The update rule is

$$\dot{M}_i = \lambda(\dot{\mathbf{l}}^\top M_i \mathbf{l} - \mathbf{u}_i)\dot{\mathbf{l}}\dot{\mathbf{l}}^\top, \quad (4.9)$$

where λ is the learning rate. Through repeated exposures to randomly-chosen inputs (“motor babbling”), this increases the value of elements of M_i in which $\mathbf{u}_i$, $\mathbf{l}$, and $\dot{\mathbf{l}}$ are correlated. It is instructive to think of the product $\mathbf{l}_j \dot{\mathbf{l}}_k$ as the “optic flow”: if the luminance is increasing in time at location j and it is high at a nearby location k , then this corresponds to positive optic flow, for example.

4.2.4 Estimating ϕ

We now expand the space of quantities to be estimated to include not just velocities, but other states that are relevant to the task of corridor following, that is, stabilizing those variables that the inner-loop hovering controller (see Introduction) is not able to estimate on its own. These consist of y , the lateral position of the robot, and ϕ , the heading angle of the robot relative to the axis of a corridor-shaped environment along which it is traveling. Our treatment builds on [142], which indicated that an autocorrelation weighting function exists that can provide an estimate of ϕ for use in control. That work used a frequency-domain

analysis of an autocorrelation scheme that only considered nearest-neighbor pixels. Here we provide a more general solution by training the weighting function in an *abinitio* fashion. This has the benefit that statistics of the scenery as well as the camera calibration itself are all learned at once.

We assume that the environment geometry resembles a narrow corridor, and that the vehicle is equipped with an additional omnidirectional camera. The bilinear model will be used to estimate components of the robot's state that cannot be observed with the sensor suite. We would like to estimate the robot orientation ϕ by using the bilinear optic flow estimate $\hat{\phi}$.

We use a Taylor expansion as before, this time as a three-term expansion in $\mathbf{u}$, $\mathbf{p}$, and $\mathbf{l}$, and have

$$\dot{\mathbf{l}} = M_\omega \mathbf{l} \omega + (M_\phi \mathbf{l} \phi + M_y \mathbf{l} y)^\top \mathbf{v}. \quad (4.10)$$

We linearize at $\mathbf{u}_0 = [\omega, v_x, v_y] = [0, 0.1m/s, 0]$ and $y = 0$, then we have

$$\dot{\mathbf{l}} = M_\phi \mathbf{l} \phi v_x, \quad (4.11)$$

where v_x can be incorporated into M_ϕ , then we can have the estimation $\hat{\phi} = \mathbf{l}^\top M_\phi \dot{\mathbf{l}}$, where M_ϕ can be learned using the method in last section.

4.3 Experiments

4.3.1 Set up the scene

A diagram of the robot, its corridor-shaped environment, and the camera model is shown in Figure 4.2. For all simulation tasks, we extracted a single horizontal line of pixels from a photograph from an outdoor scene and projected these onto both of the corridor walls.

The parameters are:

1. w_d : distance between walls = 0.1 m.
2. i_s : The image size is set to 1 m.

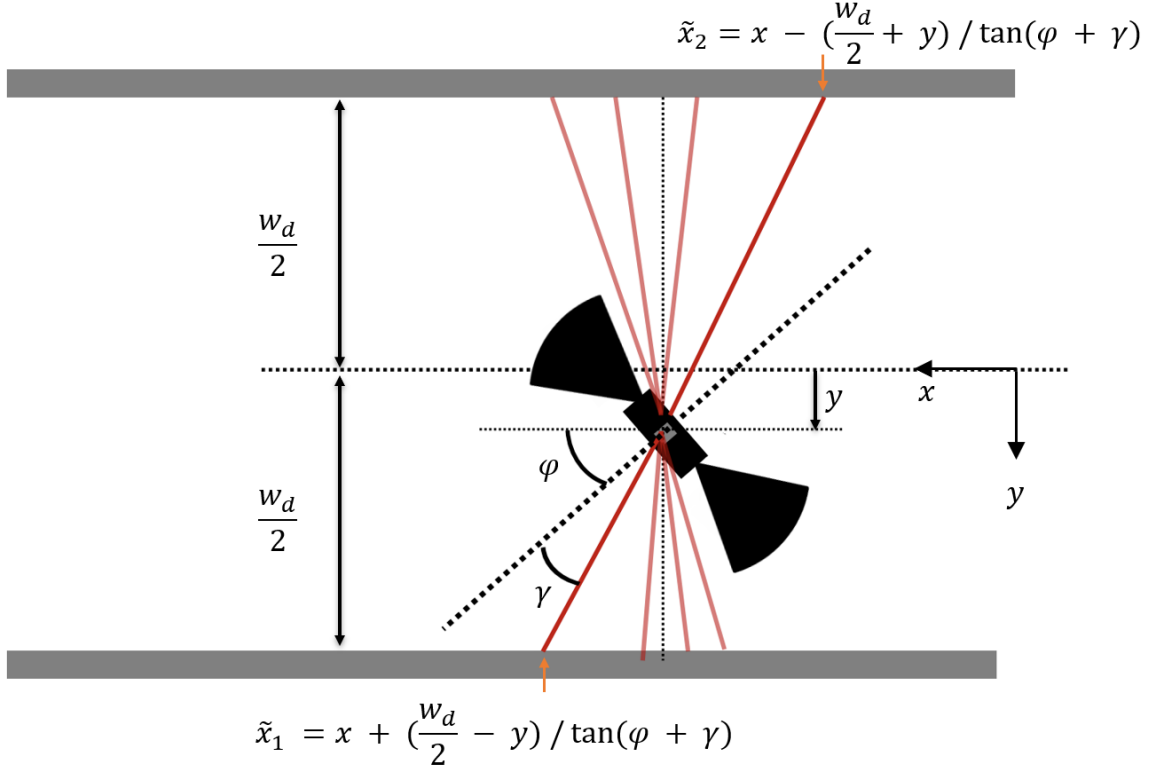


Figure 4.2: A diagram of the robot in its environment (top view). The robot is assumed to move continuously in the positive x -direction ($\dot{x} > 0$). The red lines indicate directions of pixel readings taken by its omnidirectional camera. These are comparable to the ray casting algorithm used in computer graphics nomenclature. The solid dashed line is the heading angle ϕ of the robot. $\tilde{x}_1$ is the position of the intersection for the left wall, $\phi + \gamma < \pi$ and $\tilde{x}_2$ is the position of the intersection for the right wall, $2\pi > \phi + \gamma > \pi$.

3. q : State of the robots. $q = [\phi, \dot{\phi}, x, \dot{x}, y, \dot{y}]$
4. γ : The angle of each ray relative to the heading direction.

The implementation procedure is briefly described as below:

1. The angle for the ray according to the x-axis is $\phi + \gamma$.
2. If $\phi + \gamma < \pi$, then the ray intersects the left wall, and intersection position on the wall is given by

$$\tilde{x}_1 = x + (w_d/2 - y)/\tan(\phi + \gamma). \quad (4.12)$$

If $2\pi > \phi + \gamma > \pi$, then the wall intersection position is

$$\tilde{x}_2 = x - (w_d/2 + y)/\tan(\phi + \gamma). \quad (4.13)$$

3. Then we can use $\tilde{x}$ to get the luminance readings of each ray (corresponding pixel value on the image).
4. We also need to calculate the distance from the robot to the position of the intersection of the ray and the wall . If $\phi + \gamma < \pi$, then we have the distance d is given by

$$d = \sqrt{\frac{w_d/2 - y}{\sin(\phi + \gamma)}}. \quad (4.14)$$

If $2\pi > \phi + \gamma > \pi$, then we have the distance d calculated by Eq. (4.15).

$$d = \sqrt{\left| \frac{w_d/2 + y}{\sin(\phi + \gamma)} \right|}. \quad (4.15)$$

5. We also need to know the direction of each ray to calculate $\nabla_s l$. For each ray, the direction is $(\cos(\phi + \gamma), \sin(\phi + \gamma))$.

4.3.2 Validating the camera model

We constructed a simulation environment in Python. To confirm that our simulation was correctly performing the necessary numerical operations, we confirmed that its outputs satisfied Eq. (4.3) in 2D. We describe briefly this process. The parameters are:

1. $\dot{\mathbf{l}}$: The time derivative of the luminance readings.
2. $\mathbf{s}$: The direction, which can be represented as a vector on the unit sphere. In general, $\mathbf{s} \in S$ is the sensel direction and S is the sensel space.
3. $\nabla_s \mathbf{l}$: The spatial derivative of the luminance readings.
4. $\boldsymbol{\omega}$: The angular velocity.
5. $\mu(\mathbf{s}, \mathbf{p})$: The reciprocal of the distance to the visual element, also called nearness.
6. $\mathbf{v}$: The translational velocity of the robot.

The left hand side of Eq. (4.3) can be calculated by:

$$LHS = \dot{\mathbf{l}} \approx \frac{\mathbf{l}(t - 2dt) - \mathbf{l}(t)}{2dt}. \quad (4.16)$$

As we know the right hand side of Eq. (4.3) is :

$$RHS = (\mathbf{s} \times \nabla_s \mathbf{l}) \cdot \boldsymbol{\omega} + \mu(\mathbf{s}, \mathbf{p}) \nabla_s \mathbf{l} \cdot \mathbf{v}. \quad (4.17)$$

The first term of Eq. (4.17) can be calculated as

$$(\mathbf{s} \times \nabla_s \mathbf{l}) \cdot \boldsymbol{\omega} = \left[(\cos(\phi + \gamma), \sin(\phi + \gamma)) \times \left(\frac{l_{2,i+1} - l_{2,i-1}}{2d\gamma} \cdot (-\sin(\phi + \gamma), \cos(\phi + \gamma)) \right) \right] \cdot \dot{\phi}. \quad (4.18)$$

Note that we have used the two-dimensional version of the cross product “ $\times$ ”, which results in a scalar. This can be shown by promoting the 2D vectors to 3D according to

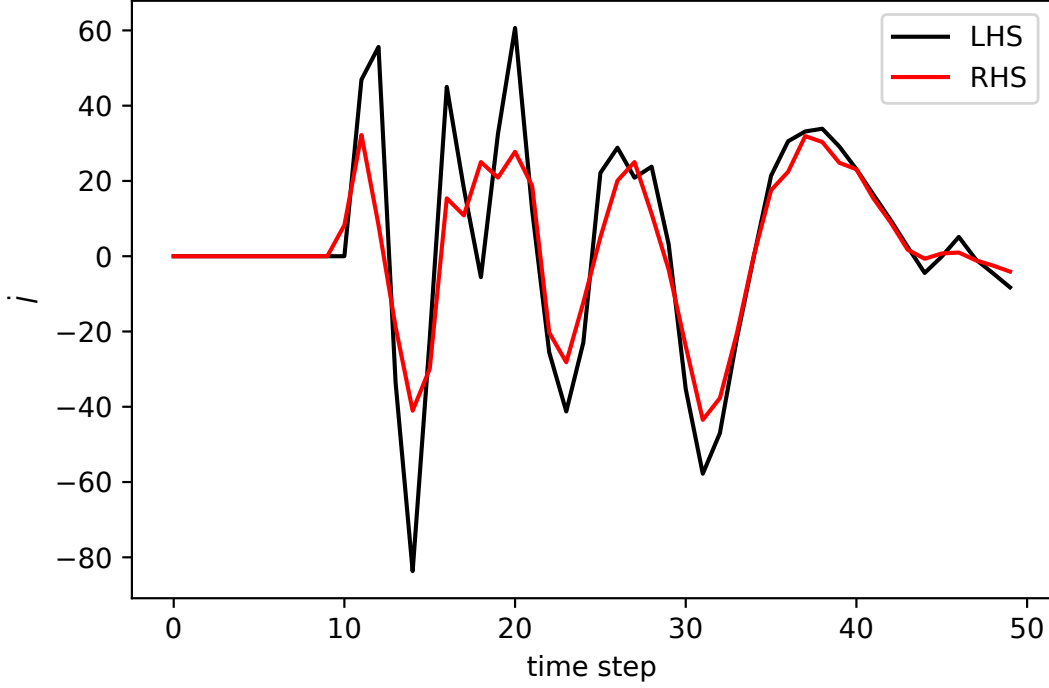


Figure 4.3: Validation of the camera model. We have $\text{RHS} = (\mathbf{s} \times \nabla_{\mathbf{s}} \mathbf{l}) \cdot \boldsymbol{\omega} + \mu(\mathbf{s}, \mathbf{p}) \nabla_{\mathbf{s}} \mathbf{l} \cdot \mathbf{v}$. And we have $\text{LHS} = \dot{l}$. We can see the RHS can follow LHS well.

$[x_1, y_1, 0] \times [x_2, y_2, 0]$, which produces a vector that only has a nonzero value along its z direction; the output of the 2D cross product is the value along the z direction. The second term of Eq. (4.17) can be calculated as

$$\mu(\mathbf{s}, \mathbf{p}) \nabla_{\mathbf{s}} \mathbf{l} \cdot \mathbf{v} = \mu(\mathbf{s}, \mathbf{p}) \frac{l_{2,i+1} - l_{2,i-1}}{2d\gamma} \cdot \left[(-\sin(\phi + \gamma), \cos(\phi + \gamma)) \cdot (v_x, v_y) \right]. \quad (4.19)$$

We select time step $dt = 1/200$ s and pick the ray index 5 (the 6th ray) and then propagate the time for 50 dt to see whether the LHS is equal to RHS on those 50 time points. We set $y = 0.05$ m, $\dot{y} = 0.05$ m/s and set $\phi = 5^\circ$, $\dot{\phi} = 100^\circ/\text{s}$ and $x = 0.25$ m, $\dot{x} = 0.1$ m/s. The validation result is shown in Figure 4.3. We can see they match well.

4.3.3 Learning M_ϕ

Because we only need to estimate ϕ , thus we only need to learn the correlator matrix $M_\phi \in \mathbb{R}^{n \times n}$. We can rewrite learning rule Eq. (4.9) to

$$\dot{M}_\phi = \lambda \phi \dot{\mathbf{l}} \dot{\mathbf{l}}^\top. \quad (4.20)$$

We use a motion babbling strategy to develop the training dataset. For each iteration, we random sample x, y, ϕ and $\dot{\phi}$ in a certain range and use the sampled values to do ray casting to get $\dot{l}$ and l . Then we can calculate $\dot{M}_\phi$ to update M_ϕ by

$$M_\phi = M_\phi + \dot{M}_\phi dt. \quad (4.21)$$

We set $\lambda = 1, dt = \frac{1}{200}$ s, and we sample x around $x = 0.5$ m with range 0.3 m, y around the middle of the corridor $y = 0$ m with range 0.03 m, ϕ around heading forward $\phi = 0^\circ$ with range 30° and $\dot{\phi} = 0$ with range $114.6^\circ/\text{s}$. We trained 20000 iterations to make M_ϕ asymptotically approach a form, up to a constant, that has the appearance shown for example in Figure 4.4. The learned M_ϕ matrix is shown in Figure 4.4. The shape is a skew-symmetric matrix matching the derivation in [8].

4.3.4 Control simulation (stablizing ϕ)

We constructed a simulation of the dynamics, Eq. (4.2), using fixed-step Euler Integration, to explore the stabilization behavior. For the simulation experiment, we set the initial condition $x_0 = 0.2$ m, $y_0 = 0.0$ m, $\dot{x}_0 = 0.1$ m/s, $\dot{y}_0 = 0.0$ m/s, $\dot{\phi}_0 = 0$ and we set different initial conditions for ϕ with a range of initial angles to get different trajectories to see whether ϕ can converge to zero. In this simulation, we only control $\dot{\phi}$ according to Eq. (4.2). We set the simulation time period is 4 s, the time step is $\frac{1}{200}$ s. $K_{p\phi} = 5$.

Figure 4.5 shows that our approach is able to steer ϕ to a within approximately 5° of zero when initiated across a range of initial conditions. Then we plot the ϕ vs time in Figure 4.5, we can see all trajectories converge to somewhere around $\phi = 0$

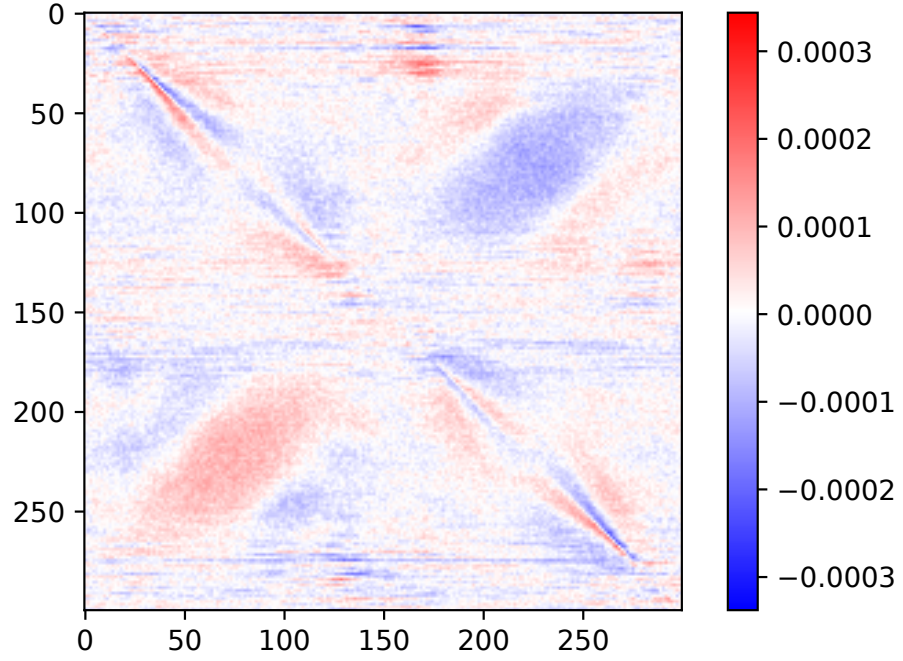


Figure 4.4: The learned M_ϕ matrix. According to the derivation in [8], it should be a skew-symmetric matrix. We can see our simulation matches the derivation.

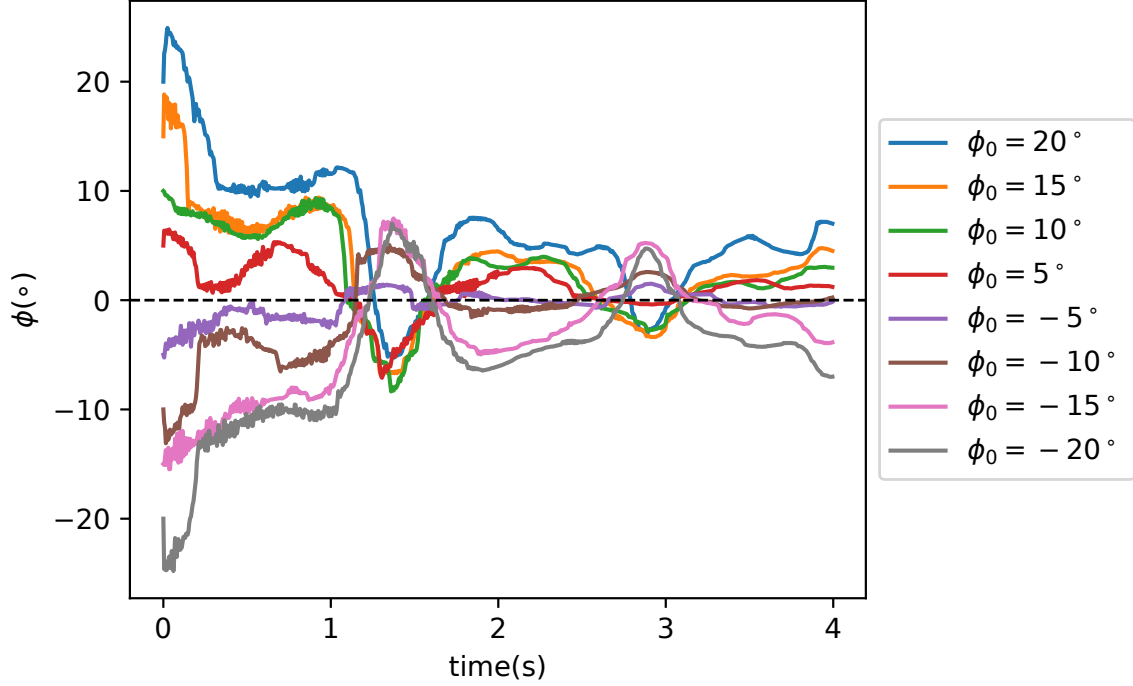


Figure 4.5: ϕ vs time. There are six trajectories with different initial conditions shown in the right bar.

To show that our optic flow estimation can estimate ϕ well, we plot ϕ and $\hat{\phi}$ in the same plot for two trajectories ($\phi_0 = 5^\circ$ and $\phi_0 = 15^\circ$) shown in Figure 4.6 and Figure 4.7.

The Figure 4.8 shows that the robot can navigate through the corridor successfully given the initial heading angle $\phi_0 = 15^\circ$.

4.4 Conclusion

Our paper has shown that using purely the optic flow estimation, we could stabilize the heading ϕ of a simulated insect-scale robot as it moves through a corridor. The bilinear optic flow approximation method we used has the advantage that it is end-to-end: a process of motor babbling allows it to learn pixel correlations regardless of their orientation, so it

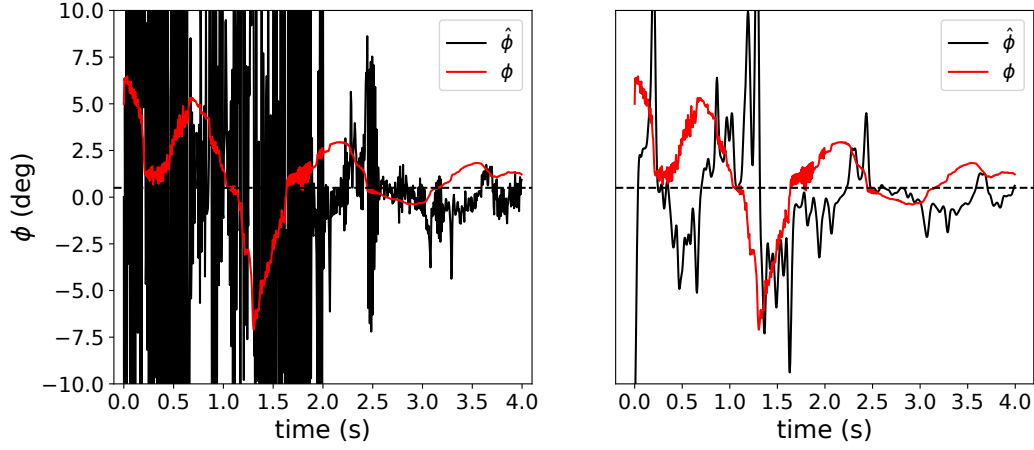


Figure 4.6: ϕ and $\hat{\phi}$ for the trajectory with initial condition $\phi_0 = 5^\circ$. The left figure is plotting ϕ and raw $\hat{\phi}$. The right figure is plotting ϕ and Gaussian filtered $\hat{\phi}$ to remove spurious noise.

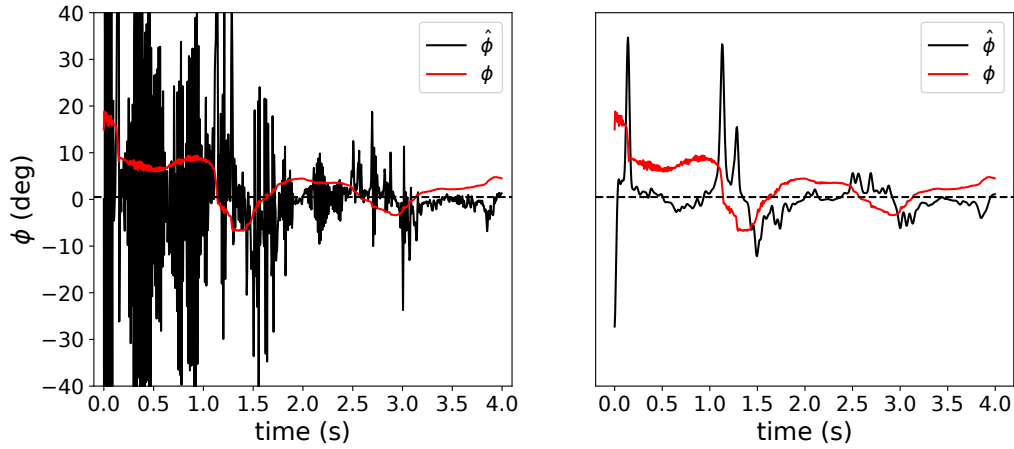


Figure 4.7: ϕ and $\hat{\phi}$ for the trajectory with initial condition $\phi_0 = 15^\circ$. The left figure is plotting ϕ and raw $\hat{\phi}$. The right figure is plotting ϕ and Gaussian filtered $\hat{\phi}$ to remove spurious noise.

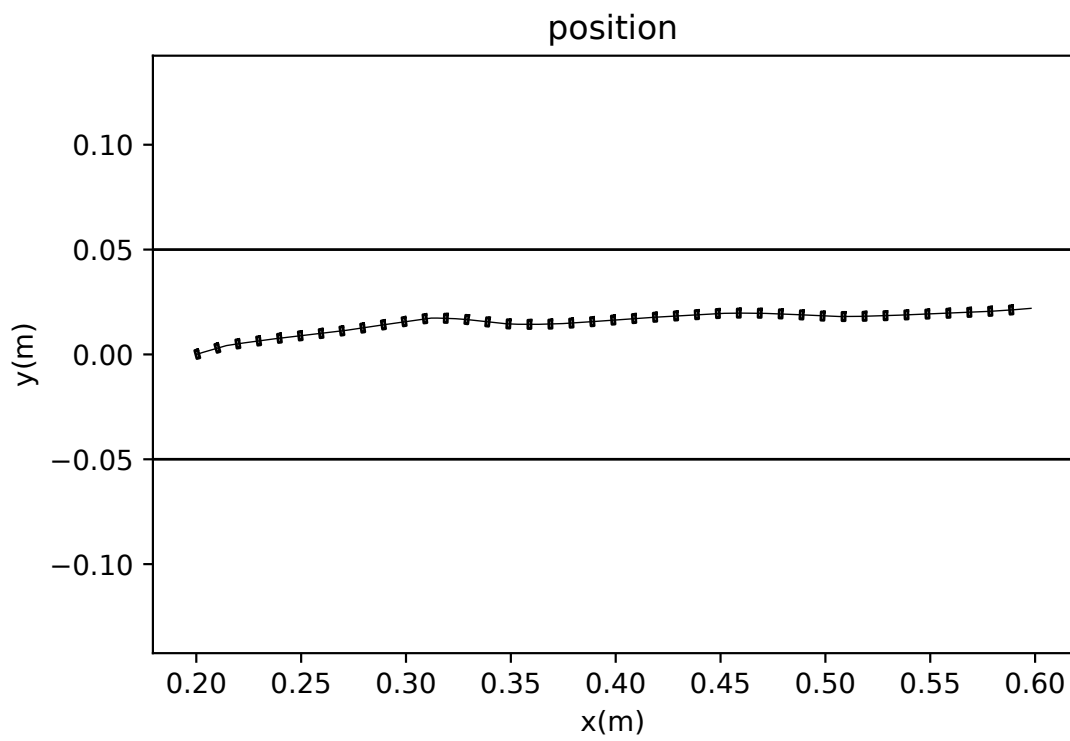


Figure 4.8: An example trajectory of the simulated robot between the corridor walls shows that it is able to align itself with the long axis of the corridor.

therefore requires no camera calibration. On a physical system, the necessary M matrices can be learned by moving the robot around while recording state using a motion capture arena or temporary on-board sensors such as tiny laser rangefinders [168]. The approach is also therefore easy to generalize to 3D flight control.

Our approach builds on an assumption that it is operating as an “outer-loop” controller that sends desired translational and rotational velocities to a fast “inner loop” that takes care of stabilizing the unstable dynamics of such a robot and regulating flight speeds. How the hovering controller is able to operate under the extreme SSWaP constraints of an insect-sized robot are described in greater detail in [2]. We believe our approach is suited to tiny robots because it operates entirely using multiply and add operations, which are relatively fast and efficient computational operations. The learned M matrices are skew-symmetric and largely sparse, so when used in practice we anticipate that near-zero entries will be zeroed out, and only half of the matrix needs to be stored in memory. Further optimizations may be possible by neglecting entries that correspond to pixel readings that are far apart. We anticipate that this will allow our method to do navigation in real time using the constrained capabilities of an onboard, general purpose microcontroller.

The learned M_ϕ appears to be performing the same computational operation as the ϕ estimator derived in [142], which itself was inspired by [169]. All three use a similar method to determine the location of the centroid of the optic flow pattern in the lateral directions. As the robot moves down the corridor, it is subject to an optic flow pattern that is highest in the lateral directions (directly to the left and right when facing forward) and zero straight ahead and behind. Therefore the centroid of this peak represents an indication of the direction one is facing when moving forward. Locating the centroid entails multiplying a function $f(x)$ by x and integrating ($\int xf(x)dx$). A weighting gradation, ranging from positive to zero to negative can be observed along the diagonal of M_ϕ , where the zero is at the lateral ray directions (ray elements 75 and 225 in Figure 4.4).

Chapter 5

TOWARDS DEEP REINFORCEMENT LEARNING FOR ONBOARD CONFINED-SPACE NAVIGATION OF INSECT-SCALE ROBOTS USING MONOCULAR EMBEDDED EVENT-BASED VISION

5.1 *Introduction*

As fabrication technology advances, the question of the right avionics architecture becomes more pressing. Classical approaches used in larger drones, which include a gyroscope and one or more laser-based rangefinders, represent a negligible component of the both weight and power, but as scale diminishes to an insect, their inherent power consumption begins to compare to or even eclipse the power required to remain in the air and the mass of the aircraft. At the scale of the UW Robofly, which weighs 150 mg and consumes approximately 500 mW to fly [15], a 24 mg gyroscope that consumes 1–3 mW (eg. tdk 20600 series) and a laser rangefinder weighing 15 mg and consuming 5 mW, represent significant mass payloads and perhaps a tolerable power payload. However, as scale reduces even further to 10 mg or even 1 mg [73], these sensors will clearly exceed the payload. We explored the elimination of the gyroscope in favor of a much lighter and more power efficient accelerometer [2] and other altitude sensor technology such as a pressure sensor introduced in Chapter 2 [1]. In a future fully-self contained insect robot, it would be desirable to simplify the sensor payload as much as possible, which suggests a heavy reliance on vision. A challenge with vision is its inherent high bandwidth and attendant heavy computation costs. Frame-based cameras struggle here due to motion blur and limited dynamic range, especially under fast ego-motion and challenging lighting, which can destabilize both classical mapping–planning–control pipelines and end-to-end policies that consume images directly. Event cameras offer an attractive al-

ternative: they report microsecond-level brightness changes asynchronously, largely reducing motion blur and operating over high dynamic range—properties that are well aligned with agile flight in confined spaces.

Confined-space navigation is a primary task that is widely applied in autonomous UAVs, while it is still absent in insect-scale robotics research. Our task is to navigate the insect-scale robots through a narrow corridor given only visual input. Visual navigation for insect-scale robots is very challenging because in such a small scale, the speed, size, weight, and power (SSWaP) constraints do not appear to permit visual navigation techniques such as SLAM (Simultaneous Localization and Mapping). We have covered the detailed introduction about the challenges in Chapter 4.

Deep reinforcement learning (RL) is emerging as a powerful paradigm in robotics, delivering strong performance across agile flight. In our setting, model-free RL learns a parametric policy by interacting with the environment, yielding an end-to-end network that maps monocular event-based observations directly to control commands. This approach avoids complicated hand-engineered pipelines and can reduce inference latency. When designed to be lightweight, the policy can run on a microcontroller—enabling fully onboard autonomy for insect-scale robots.

In this paper, we present a deep RL system for onboard confined-space navigation using a monocular embedded event camera, designed for the tight power and compute budgets of insect-scale robots. We trained a lightweight deep RL policy in NVIDIA Isaac Lab with a binary event-mask (BEM) computed from consecutive grayscale frames, and deployed it on the OpenMV Cam RT1062 to perform onboard inference from event-camera data alone. The training and inference pipeline is shown in Figure 5.1. To our knowledge, this is the first learning-based approach to address confined-space navigation for insect-scale robots with an event camera and without any offboard computation (no external link/antenna). Our contributions are:

1. Developed, to the best of our knowledge, the world’s smallest aerial robot platform

with the onboard embedded event camera module.

2. Developed a deep reinforcement learning policy that runs on a microcontroller using only event-camera input.
3. Demonstrated that onboard policy-inference power consumption fits within the insect-scale flight power budget.

5.2 Methodology

5.2.1 Event camera

For insect-scale robots, event camera has the advantages of:

- Low latency: Event cameras conduct the asynchronously detecting and recording the changes in intensity at each single pixel. Each pixel independently triggers an "event" only when an intensity change exceeds a fixed threshold. There is no need to wait for a full frame to be captured, processed, and transmitted. In contrast, conventional frame-based cameras capture entire frames at fixed time intervals(frames per second). Thus, event cameras can achieve much smaller latency than the frame-based cameras.
- No motion blur: In conventional frame-based cameras, motion blur happens when the entire scene is captured over a fixed exposure time. At the same time, objects moving at high speed will create a "smear" as they shift position relative to the camera sensor. If the scene or objects move very quickly and their motion is integrated over the duration of the frame capture, the blurred images will be generated. In contrast, event cameras independently capture changes in intensity in real time and each event is timestamped with microsecond precision, so there is no cumulative exposure time during which motion can blur the image.
- Very low power: Because the event cameras only trigger the events on each pixel independently when there is a significant change in intensity, the pixels triggered and

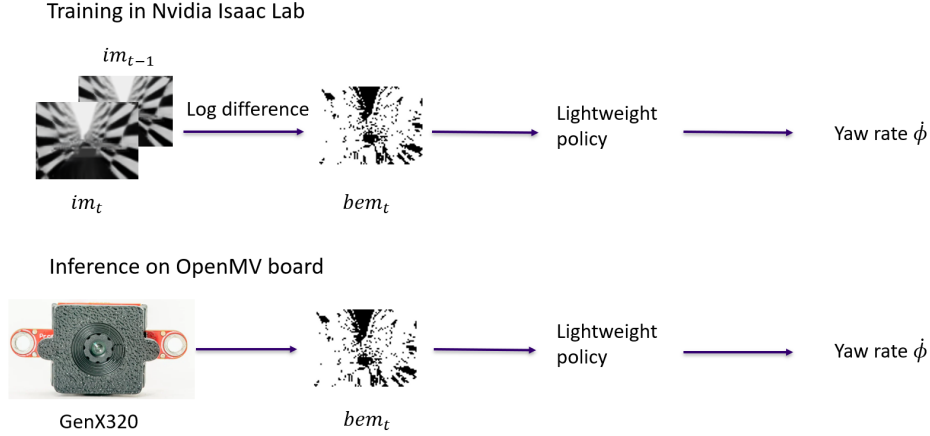


Figure 5.1: Training in simulation and onboard inference on the robot platform. During training, BEMs are computed from consecutive grayscale frames and fed to the policy network. During deployment, BEMs are computed from the GenX320 event camera and fed to the same policy running on the OpenMV Cam RT1062 microcontroller.

the data generated are much less than the conventional cameras. The overall data flow is lower using event cameras so the bandwidth requirements and the computational power are reduced. Thus, the event cameras can be operated in mW-level power.

5.2.2 Deep Reinforcement Learning

Task formulation

We formulate the confined-space navigation problem with purely vision-based sensing as a *partially observable Markov decision process* (POMDP), since the robot does not have access to its full underlying state. The task is to navigate a narrow corridor while avoiding wall collisions, with the robot starting on the centerline at the corridor entrance. For training, we construct a corridor with a 30° bend in NVIDIA Isaac Sim (Figure 5.2), and for evaluation, we design additional test corridors with a straight path, a 15° bend, and a 60° bend (Figure 5.9).

Observation space

Because there is no effective continuous-time event camera simulator [170], we use a *binary event mask* (BEM) as the observation, derived a monocular, forward-facing camera. At each environment step, the policy receives a single-channel BEM computed from the logarithmic brightness change between consecutive grayscale frames. Let $I_t(x, y)$ denote the image intensity at pixel (x, y) and time t ; the BEM is

$$\text{BEM}_t(x, y) = \mathbb{I}(|\log I_t(x, y) - \log I_{t-1}(x, y)| \geq \kappa), \quad (5.1)$$

with threshold $\kappa > 0$ and indicator $\mathbb{I}(\cdot)$. On the first step (no previous frame), a zero mask is returned. The observation at time t is a binary tensor $\mathbf{o}_t \in \{0, 1\}^{H \times W}$ with $H=W=320$, where H and W denote height and width, respectively. No privileged state (e.g., position, velocity, or goal pose) is appended.

Action space

The action space is one-dimensional and controls yaw rate only. The policy outputs a normalized scalar action $a_t \in [-1, 1]$, which is mapped to a physical yaw-rate command

$$\dot{\psi}_t = \omega_{\max} a_t, \quad \omega_{\max} = 1.0 \text{ rad/s}. \quad (5.2)$$

A hierarchical controller tracks fixed body-frame velocity targets $v_x^* = 1.0$ m/s, $v_y^* = 0$, and altitude $z^* = 0.5$ m. It converts these targets to thrust and moments, and applies the policy’s yaw-rate command to regulate heading. Accordingly, the agent learns to steer using vision-only observations while lower-level loops stabilize forward motion and altitude.

Reward function

We use a dense reward that encourages fast, centered flight, plus a sparse collision penalty.

Let $\mathbf{p}_t = (x_t, y_t, z_t)^\top$ be the world position and Δt the control period. The corridor centerline is $y_c(x)$ with unit tangent $\mathbf{t}(x_t) \in \mathbb{R}^2$. Define the planar position $\mathbf{p}_t^{xy} = (x_t, y_t)^\top$,

the signed arc-length increment

$$\Delta s_t = (\mathbf{p}_t^{xy} - \mathbf{p}_{t-1}^{xy})^\top \mathbf{t}(x_t),$$

and the forward speed along the centerline

$$\dot{s}_t = \max\left(0, \frac{\Delta s_t}{\Delta t}\right).$$

Let $e_y = |y_t - y_c(x_t)|$ be the lateral deviation and $\mathbb{I}_{\text{col},t} \in \{0, 1\}$ a collision indicator.

With weights $w_{\text{prog}} > 0$, $w_{\text{center}} < 0$, and collision scale $c_{\text{col}} < 0$, the per-step reward is

$$r_t = \underbrace{w_{\text{prog}} \dot{s}_t}_{\text{forward progress}} + \underbrace{w_{\text{center}} e_y}_{\text{centerline penalty}} + \underbrace{c_{\text{col}} \mathbb{I}_{\text{col},t}}_{\text{collision penalty}}. \quad (5.3)$$

5.2.3 Policy design and training

We learn a vision-only policy $\pi_\theta : \mathbf{o}_t \mapsto a_t^{\text{yaw}}$ that maps the BEM observation $\mathbf{o}_t$ to a single continuous action, the commanded yaw rate a_t^{yaw} . The network uses a lightweight convolutional neural network (CNN) backbone to extract spatial features, followed by multilayer perceptron (MLP) layers that parameterize a Gaussian policy over a_t^{yaw} . We optimize π_θ with the Proximal Policy Optimization (PPO) [171] on parallel simulations using Nvidia Isaac lab. The corresponding environment grid view in NVIDIA Isaac Sim is illustrated in Figure 5.2. The Crazyflie’s onboard camera view is shown in Figure 5.3.

PD controller

In Nvidia isaac lab, the default Crazyflie interface exposes low-level thrust-and-moment control. On the real Crazyflie, the control commands are: forward velocity v_x , lateral velocity v_y , yaw rate $\dot{\psi}$ and altitude z . To align the two, we implemented a PD controller in simulation that maps these setpoints to thrust and body moments so the simulated control matches the real robot. Because the policy’s only action is yaw rate, we tuned its closed-loop response carefully. We first measured the real Crazyflie’s yaw-rate step response (Figure 5.4). We then tuned the simulation controller gains to reproduce this response (Figure 5.5), aligning the simulated dynamics with the hardware.

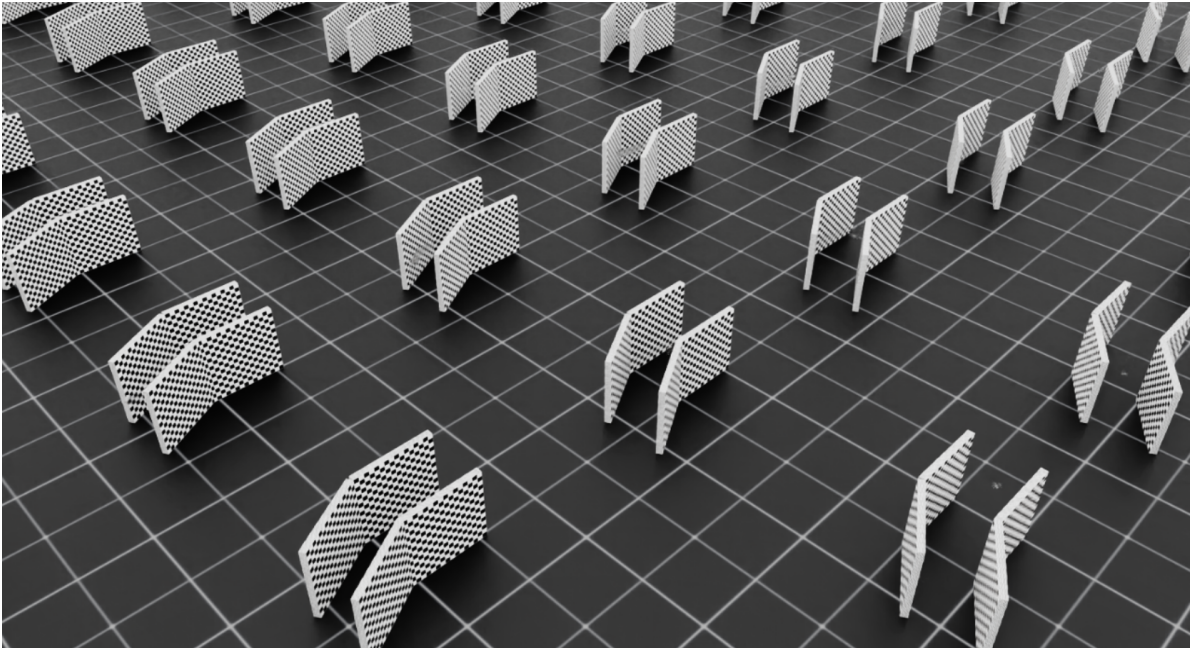


Figure 5.2: Environment grid view in NVIDIA Isaac Sim

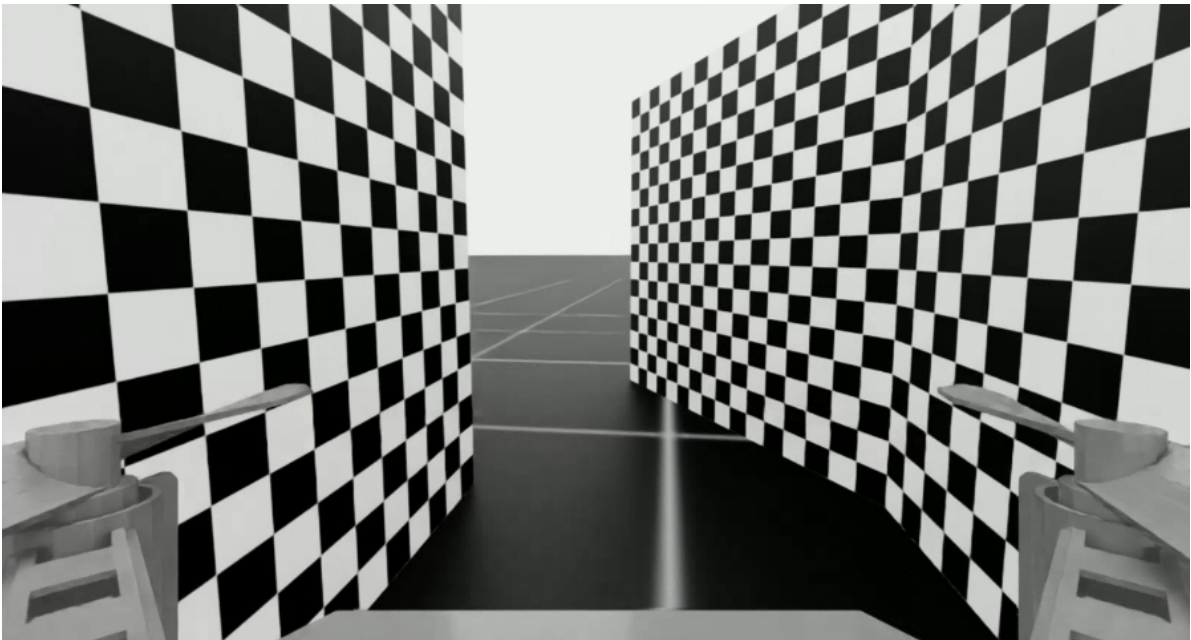


Figure 5.3: Crazyflie's onboard camera view in the checkerboard corridor, simulated in NVIDIA Isaac Sim.

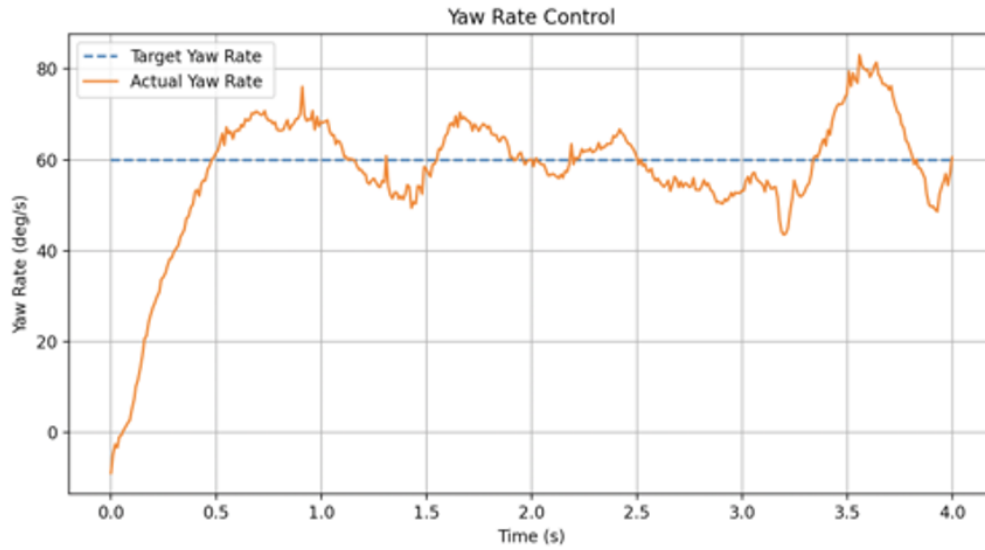


Figure 5.4: The yaw-rate step response from real Crazyflie.

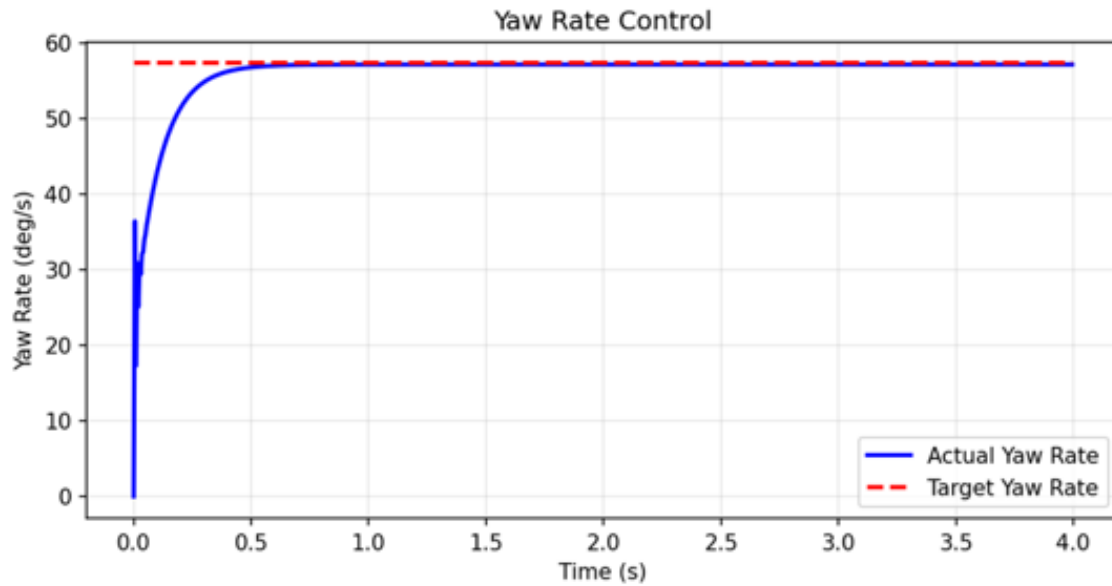


Figure 5.5: The yaw-rate step response in simulation.

TinyML

Deploying a large reinforcement learning model directly onto the OpenMV Cam RT1062 board is infeasible due to the board’s limited memory capacity. To address this constraint, model quantization is employed to reduce the model’s size while preserving its robustness and performance. In our experiments, quantization reduced the file size of a model from 394 KB to 128 KB, enabling successful deployment on the microcontroller. Quantization achieves this reduction by lowering the precision of numerical representations, such as weights and activations, from higher-bit formats (e.g., 32-bit floating-point) to lower-bit formats (e.g., 8-bit integers). This process maps a wide range of values into a smaller, discrete set, significantly reducing memory requirements. To quantize a PyTorch model, it must be converted into a TensorFlow Lite model. Since PyTorch does not natively export to TensorFlow Lite, the model is initially converted into the ONNX format, then into a TensorFlow model, before finally being quantized into TensorFlow Lite. This workflow offers a practical approach for deploying reinforcement learning models on resource-constrained embedded platforms, allowing for real-time inference without exceeding hardware limitations.

5.3 Hardware Platform

We built a robot platform using a brushless Crazyflie, an OpenMV Cam RT1062 board, and a GenX320 event-camera module.

5.3.1 Brushless Crazyflie Platform

The Brushless Crazyflie 2.1 quadrotor, an open-source aerial robotics platform, was employed to evaluate the proposed method. With a total mass of 32 g and a compact form factor, it is particularly suitable for navigation in confined environments. In comparison to the brushed-motor version, the brushless platform incorporates four 08028–10000KV motors, each capable of generating approximately 30 g of thrust. This configuration supports an increased payload capacity (40 g compared to 15 g), improved efficiency, and an extended



Figure 5.6: Integrated robot platform next to a pencil.

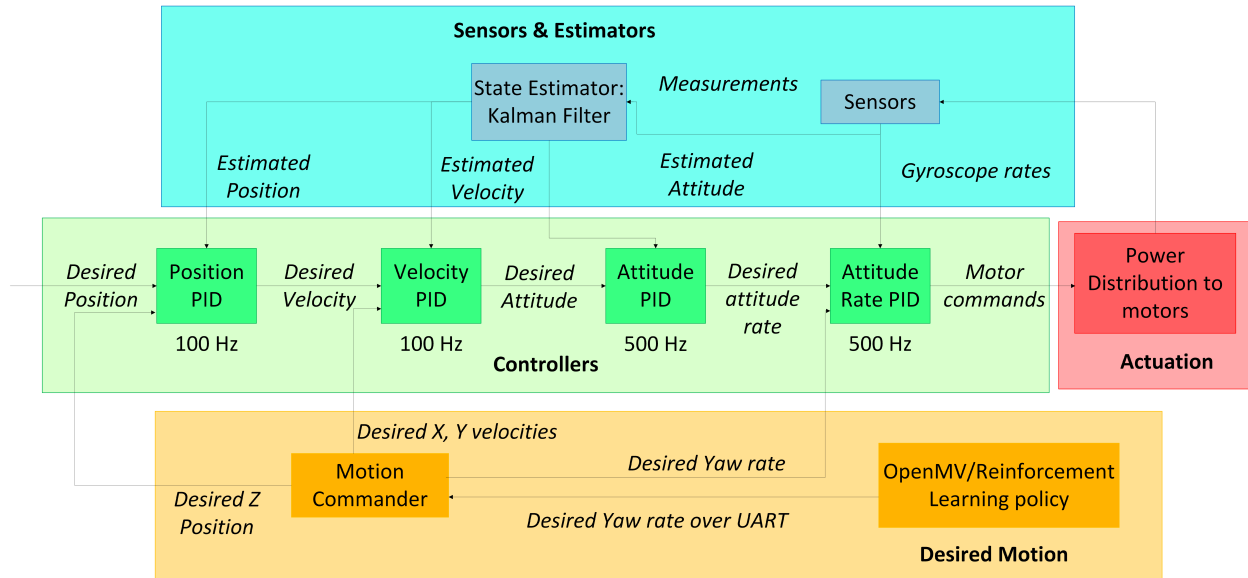


Figure 5.7: Architectural flow of the Crazyflie firmware.

flight time of approximately 10 minutes, representing a three-minute increase relative to the brushed version. These improvements enhance suitability for confined-space navigation, even when carrying additional payloads such as the protective enclosure and the OpenMV camera. Each motor is driven by an integrated 1-cell 5 A electronic speed controller (ESC) running BLHeli_S/Bluejay firmware, while overall flight control is handled by the onboard STM32F405 microcontroller, which generates the corresponding ESC commands. Together, these components establish the foundation for stable flight and system integration.

The firmware architecture, illustrated in Figure. 5.7, incorporates a stabilizer module that governs flight control through multiple interconnected components. Prior to executing control commands, the system processes measurements from the onboard sensor suite, which includes an accelerometer, gyroscope, and pressure sensor for state estimation. In addition, a Flowdeck V2 optical flow sensor provides high-precision displacement measurements relative to the ground at millimeter resolution, further enhancing flight stability.

Sensor measurements are fused using an onboard Kalman filter to generate real-time

estimates of the vehicle state. These estimates are subsequently provided to the state controller, which is implemented as a proportional–integral–derivative (PID) controller within the Crazyflie firmware. The controller computes corrective actions for roll, pitch, and yaw based on the desired setpoints. These control outputs are then converted into motor commands through the embedded flight control logic, ensuring stable tracking of commanded trajectories. The resulting framework provides a reliable baseline for autonomous operation, which is extended in this work through integration with the OpenMV camera for onboard perception and control.

5.3.2 *OpenMV Board with GenX320 Event Camera*

The microcontroller board used in this work is the OpenMV Cam RT1062, a low-power and lightweight platform capable of real-time machine vision. It is integrated with the GenX320 event camera module to enable event-based visual sensing. The GenX320 event-based sensor was selected, as it is light-weight, low power consumption and well-suited for capturing rapid changes in the visual field.

The camera is mounted on top of the drone within a protective enclosure introduced in Section 5.3.4. Communication between the drone and the camera is established via a UART connection between the two devices. The camera is powered through a Li-Po battery connected to the drone’s VCOM and GND pins, providing 3.7 V via the board’s integrated Li-Po support circuitry.

The primary function of the OpenMV board is to acquire event data from the camera, generate the BEM representation, and supply this processed information as input to the deep RL policy. The RL policy subsequently outputs a yaw rate command, which is transmitted to the drone for navigation. This perception and control pipeline operates at approximately 15 Hz, allowing the system to respond rapidly to static obstacles in the environment. Importantly, this loop is decoupled from the drone’s attitude control loop, which operates at a significantly higher frequency of 500 Hz to maintain stable flight. This separation enables the reinforcement learning policy to run independently, without compromising the drone’s

flight stability.

The OpenMV board communicates with the drone by transmitting control commands in a packet structure containing the body-frame velocities v_x and v_y , the altitude z , and the yaw rate $\dot{\psi}$. In this study, the lateral velocities and altitude were fixed, and only the yaw rate was allowed to vary. Although these additional control channels were not utilized, they offer flexibility for future investigations that may require full three-dimensional maneuvering of the drone.

In the context of reinforcement learning, deploying a policy on a real robot requires the input representation to be consistent with that used in simulation. To this end, we employ the BEM representation for event camera data. The BEM converts the asynchronous output of the event camera into a frame-based binary image, where each pixel assumes a value of either 1 or 0. To construct the BEM, raw events were first collected, with each event characterized by the tuple (event type, seconds, milliseconds, microseconds, coordinate x, coordinate y). The event type encodes the polarity of the event, indicating whether a pixel undergoes a positive or negative intensity change. Within a specified temporal window, Δt , events were accumulated per pixel, and the absolute difference between the counts of positive and negative events was computed. A pixel was assigned a value of 1 if this difference was nonzero; otherwise, it remained 0. The resulting binary image thus encapsulates the dynamic changes within the observed environment and serves as the input for our reinforcement learning model.

The primary challenge in this phase was the computationally efficient implementation of the BEM representation. An initial strategy involved iteratively processing individual events to manually accumulate positive and negative events for image construction. However, this approach was computationally intensive and proved unsuitable for real-time applications. To overcome these limitations, the implementation was restructured to fully vectorize the computation using OpenMV’s built-in functions that are optimized to be the fastest they can be, thereby substantially reducing execution time and enabling efficient, real-time generation of BEM representations. Leveraging the OpenMV’s capability for real-time event-based

perception, the subsequent section details how its output interfaces with the Crazyflie’s commander to facilitate autonomous flight control.

5.3.3 *Communication Interface*

With the OpenMV and Crazyflie drone connected via wires, communication occurs through the UART (Universal Asynchronous Receiver/Transmitter) interface. UART was chosen for its simplicity and low-latency operation, enabling reliable board-to-board data transfer. To send control commands, we designed a fixed packet structure consisting of one start byte followed by 8 bytes of data: 2 bytes each for the v_x, v_y, z and $\dot{\psi}$, all represented as `int16_t`. The start byte serves as a marker, ensuring correct packet alignment and preventing corrupted or misinterpreted data. Within the Crazyflie firmware, we implemented two main modifications: a module that waits for and parses the complete packet into usable values, and an interface that forwards the parsed data to the Crazyflie commander. The commander then interprets the values as control setpoints, converting them into motor commands to achieve the desired motion.

5.3.4 *3D Printed Protective Enclosure*

We carefully designed the wiring and the layout using a 3D-printed protective enclosure. The enclosure was built to achieve two primary design aims: (1) securely tethering the OpenMV board to the Crazyflie, and (2) protecting sensitive board components from impact. Additionally, it shields and conceals the wiring between the board and drone while maintaining reliable electrical connections. The enclosure employs a two-part design consisting of: (1) a main body and (2) a cover, fastened together with four screws. The main body directly interfaces with both the board and Crazyflie using four screws, allowing for a tight fit along the sides and back of the board. The cover complements this by securing the front (camera) side and protecting the top of the board, which is otherwise exposed above the Crazyflie’s motor shields and vulnerable to direct impact. Both parts are 3D printed from PLA plastic

with 20% infill, chosen to balance strength and weight. This material also enables simple and reliable threading of screw holes.

5.4 Experiments

5.4.1 Hardware evaluation

We designed a simple demonstration to evaluate our hardware platform in a static obstacle avoidance task. Conceptually, the BEM represents movement information as binary values (1s) at each pixel. In our approach, the BEM is divided into left and right halves to inform directional decisions. To reduce noise from the drone’s propellers, the frame is further partitioned into a top and bottom region, with the bottom region being discarded. Using only the top portion, the number of active pixels (1s) is counted separately for the left and right sides. If the number of active pixels on both sides falls below a predefined threshold, no yaw adjustment is applied. Otherwise, the accumulated counts are compared, and if the difference exceeds a second threshold, a yaw command is issued to steer the drone away from the obstacle. By applying this simple policy, we can achieve basic obstacle avoidance illustrated in Figure. 5.8.

5.4.2 Policy evaluation

We trained the policy in a corridor with a 30° bend and evaluated it on corridors with varying bend angles. The resulting trajectories are shown in Figure 5.9. The drone successfully navigates the straight, 15°, and 30° corridors, but fails to complete the 60° bend. This failure arises because the observations encountered in the 60° corridor deviate too far from the distribution of training data, exceeding the policy’s ability to generalize.

5.4.3 Model deployment

We quantized the model using TensorFlow Lite and deployed it on the OpenMV board. The CPU load breakdown indicates that policy inference requires a reasonable amount of



Figure 5.8: Obstacle avoidance using the onboard GenX320 event-camera module.

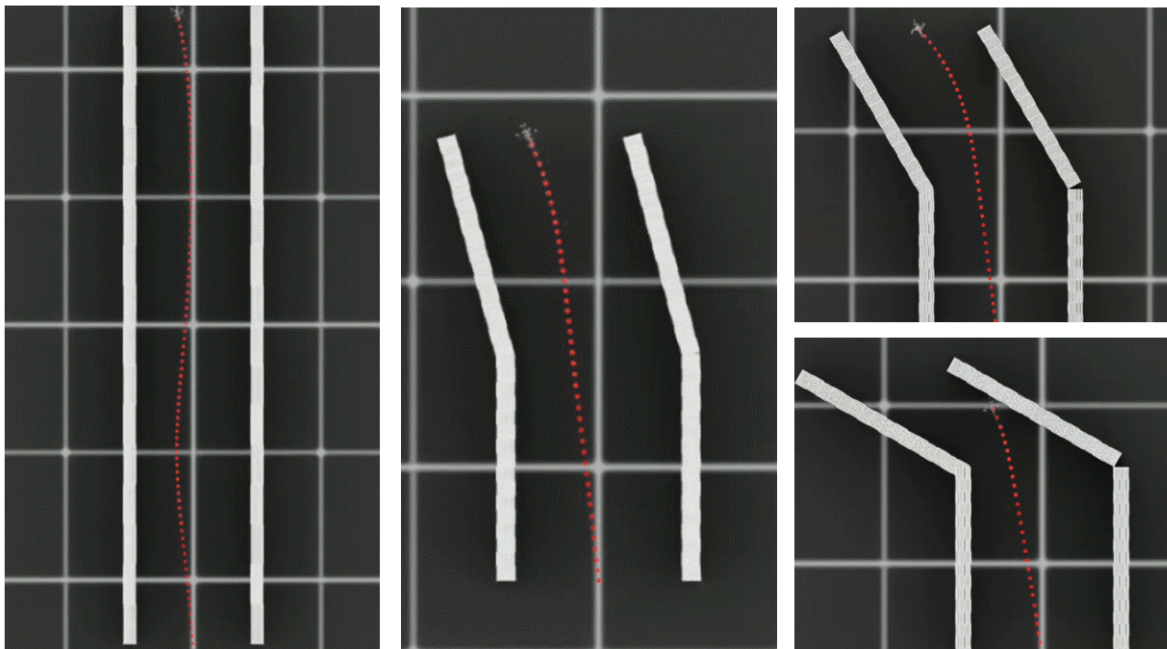


Figure 5.9: Crazyflie trajectory (red dotted) in four corridor geometries. Left: straight (5 m). Right, top to bottom: 15°, 30°, and 60° bends. The bent corridors are composed of two 1.22m straight sections connected at the midpoint.

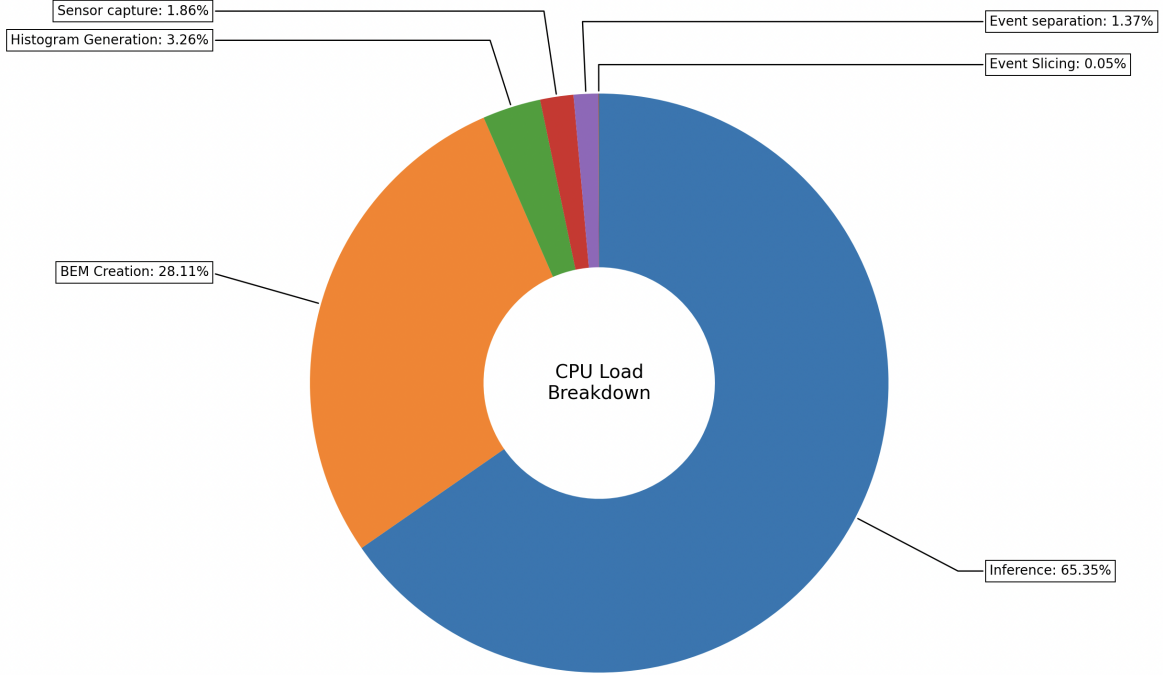


Figure 5.10: CPU load breakdown.

computation. Furthermore, the measured power consumption of policy inference is 1.43 mW at an inference rate of 15 Hz, which is well within the power budget of insect-scale robots.

5.5 Conclusion and Future work

In this work, we introduced the first approach to vision-based confined-space navigation for insect-scale robots using an onboard event camera and deep reinforcement learning. By employing the BEM representation and a lightweight neural policy, we demonstrated successful navigation through narrow corridors in NVIDIA Isaac Sim. We further developed the smallest aerial platform to date with an integrated event camera, implemented the supporting firmware, and deployed the trained policy onboard. Our evaluation showed that policy inference operates within the tight power budget of insect-scale robots, consuming less than 1.5 mW.

Future work will focus on sim-to-real transfer. Although efforts were made to align the simulation with real-world conditions—such as matching wall dimensions, tuning the PD controller, and refining the event data representation—a large gap still remains between simulation and reality. To address this, we plan to apply domain randomization techniques to reduce the simulation-to-reality gap, explore modifications to the policy architecture to improve generalization, and conduct comprehensive evaluations using the physical robot platform we built. We will report these results at a leading robotics venue.

Chapter 6

CONCLUSION AND FUTURE WORK

This thesis investigates the state of art methods in vision and control field to address the challenges of hovering and navigation for insect-scale robots.

For hovering task, we created TinySense, the world’s lightest avionics system for insect-scale robots to do sensing and accurate state estimation. By replacing a power-hungry laser rangefinder with a compact pressure sensor and redesigning the optic flow module around a global-shutter camera with a custom Lucas–Kanade implementation, we reduced the total sensor suite weight to 78.4 mg and power consumption to 15.6 mW—less than half of prior state-of-the-art avionics systems. TinySense demonstrated accurate onboard estimation of pitch angle, translational velocity, and altitude. Real flight experiments with the Crazyflie quadrotor confirmed that TinySense achieves performance comparable to the built-in estimator of a much larger platform, despite operating under strict speed, size, weight, and power (SSWaP) constraints. For future work, integrating TinySense with fully autonomous insect-scale platforms such as Robofly will lead to evaluation in more challenging flight regimes. Beyond hovering, TinySense provides a foundational sensor suite for advancing higher-level autonomy, including tasks such as obstacle avoidance and navigation.

And we then introduce a gyroscope-free, visual-inertial flight control system capable of wind compensation. This system, inspired by the biological fusion of wind and vision sensing in fruit flies, integrates a tiny optic flow camera and accelerometer to maintain stability in response to wind disturbances. By eliminating the need for a gyroscope, we reduce both weight and power consumption, enabling a 10-milligram robot to do stable hovering. Through extensive simulations and physical tests, the system demonstrates effective wind rejection, advancing the field toward NAT robots. Future work will integrate this system with RoboFly

first and conduct real-robot experiments.

For navigation, we first propose to use a biology-inspired approach, which we term the bilinear optic flow approximation, that is more computationally efficient. We build on previous work that has shown that the bilinear approximation can be used for visual navigation. Here, we show that a bilinear approximator can be learned that is able to stabilize the heading of a robot while performing continuous forward motion in a corridor-shaped environment. This is a necessary capability for confined-space navigation that insect-sized robots are likely to perform. In this work, we describe the underlining methodology of the method and built a 2D visual simulation environment and omnidirectional camera model to validate our results. We then explore the use of an unconventional vision sensor—the event camera—combined with a deep reinforcement learning policy for onboard confined-space navigation. To this end, we developed the smallest aerial robot platform equipped with an embedded event camera module. The trained policy was deployed onto the OpenMV board, where we demonstrated that it runs efficiently within the available computational resources and remains well within the power budget of insect-scale robots. Future work will focus on achieving sim-to-real transfer to demonstrate that our deep RL framework can enable confined-space navigation in real flight experiments. At present, we rely on an off-the-shelf board to power the event camera. Moving forward, we plan to design a customized PCB to power the event camera, reducing overall weight and enabling integration of the sensor suite into insect-scale robots such as RoboFly.

BIBLIOGRAPHY

- [1] Zhitao Yu, Joshua Tran, Claire Li, Aaron Weber, Yash P Talwkar, and Sawyer Fuller. Tinysense: A lighter weight and more power-efficient avionics system for flying insect-scale robots. *arXiv preprint arXiv:2501.03416*, 2025.
- [2] Sawyer Fuller, Zhitao Yu, and Yash P Talwkar. A gyroscope-free visual-inertial flight control and wind sensing system for 10-mg robots. *Science Robotics*, 7(72):eabq8184, 2022.
- [3] Zhitao Yu, Gioele Zardini, Andrea Censi, and Sawyer Fuller. Visual confined-space navigation using an efficient learned bilinear optic flow approximation for insect-scale robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4250–4256. IEEE, 2022.
- [4] Yogesh M Chukewad and Sawyer Fuller. Yaw control of a hovering flapping-wing aerial vehicle with a passive wing hinge. *IEEE Robotics and Automation Letters*, 6(2):1864–1871, 2021.
- [5] Daksh Dhingra, Yogesh Madhavrao Chukewad, and Sawyer Fuller. A device for rapid, automated trimming of insect-sized flying robots. *IEEE Robotics and Automation Letters*, 2020.
- [6] Melanie J Anderson, Joseph G Sullivan, Timothy K Horiuchi, Sawyer B Fuller, and Thomas L Daniel. A bio-hybrid odor-guided autonomous palm-sized air vehicle. *Bioinspiration & Biomimetics*, 16(2):026002, 2020.
- [7] Sawyer B. Fuller, Andrew D. Straw, Martin Y. Peek, Richard M. Murray, and Michael H. Dickinson. Flying *Drosophila* stabilize their vision-based velocity con-

- troller by sensing wind with their antennae. *Proceedings of the National Academy of Sciences Plus*, 111(13):E1182–E1191, 2014. doi: 10.1073/pnas.1323529111. URL <http://www.pnas.org/content/111/13/E1182.abstract>.
- [8] Andrea Censi, Shuo Han, Sawyer B Fuller, and Richard M Murray. A bio-plausible design for visual attitude stabilization. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 3513–3520. IEEE, 2009.
- [9] Yogesh M Chukewad, Avinash T Singh, Johannes M James, and Sawyer B Fuller. A new robot fly design that is easier to fabricate and capable of flight and ground locomotion. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4875–4882. IEEE, 2018.
- [10] R. J. Wood. The first takeoff of a biologically inspired at-scale robotic insect. *IEEE Trans. Robotics*, 24(2):341–347, 2008. doi: 10.1109/TRO.2008.916997.
- [11] Kevin Y. Ma, Pakpong Chirarattananon, Sawyer B. Fuller, and Robert J. Wood. Controlled flight of a biologically inspired, insect-scale robot. *Science*, 340(6132):603–607, 2013.
- [12] Sawyer B Fuller. Four wings: An insect-sized aerial robot with steering ability and payload capacity for autonomy. *IEEE Robotics and Automation Letters*, 4(2):570–577, 2019.
- [13] Vamsi Talla, Bryce Kellogg, Benjamin Ransford, Saman Naderiparizi, Shyamnath Golakota, and Joshua R Smith. Powering the next billion devices with wi-fi. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, pages 1–13, 2015.
- [14] Moritz A Graule, Pakpong Chirarattananon, Sawyer B Fuller, Noah T Jafferis, Kevin Y Ma, Matthew Spenko, Roy Kornbluh, and Robert J Wood. Perching and takeoff of

- a robotic insect on overhangs using switchable electrostatic adhesion. *Science*, 352(6288):978–982, 2016.
- [15] Yogesh M. Chukewad, Johannes M. James, Avinash T. Singh, and Sawyer B. Fuller. Robofly: An insect-sized robot with simplified fabrication that is capable of flight, ground, and water surface locomotion. *IEEE Transactions on Robotics*, 37:2025–2040, 2021.
- [16] Joseph Conroy, Gregory Gremillion, Badri Ranganathan, and J Sean Humbert. Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. *Autonomous robots*, 27(3):189–198, 2009.
- [17] Hector D Escobar-Alvarez, Neil Johnson, Tom Hebble, Karl Klingebiel, Steven AP Quintero, Jacob Regenstein, and N Andrew Browning. R-advance: rapid adaptive prediction for vision-based autonomous navigation, control, and evasion. *Journal of Field Robotics*, 35(1):91–100, 2018.
- [18] Christophe De Wagter, Sjoerd Tijmons, Bart DW Remes, and Guido CHE de Croon. Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4982–4987. IEEE, 2014.
- [19] Jean-Christophe Zufferey, Adam Klaptocz, Antoine Beyeler, Jean-Daniel Nicoud, and Dario Floreano. A 10-gram microflyer for vision-based indoor navigation. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3267–3272. IEEE, 2006.
- [20] Daniele Palossi, Antonio Loquercio, Francesco Conti, Eric Flamand, Davide Scaramuzza, and Luca Benini. A 64-mw dnn-based visual navigation engine for autonomous nano-drones. *IEEE Internet of Things Journal*, 6(5):8357–8371, 2019.

- [21] Pakpong Chirarattananon, Kevin Y Ma, and Robert J Wood. Adaptive control of a millimeter-scale flapping-wing robot. *Bioinspiration & biomimetics*, 9(2):025004, 2014.
- [22] YuFeng Chen, Siyi Xu, Zhijian Ren, and Pakpong Chirarattananon. Collision resilient insect-scale soft-actuated aerial robots with high agility. *IEEE Transactions on Robotics*, 37(5):1752–1764, 2021. doi: 10.1109/TRO.2021.3053647.
- [23] Avik De, Rebecca McGill, and Robert J Wood. An efficient, modular controller for flapping flight composing model-based and model-free components. *The International Journal of Robotics Research*, 41(4):441–457, 2022.
- [24] Andrea Tagliabue, Yi-Hsuan Hsiao, Urban Fasel, J Nathan Kutz, Steven L Brunton, YuFeng Chen, and Jonathan P How. Robust, high-rate trajectory tracking on insect-scale soft-actuated aerial robots with deep-learned tube mpc. *arXiv preprint arXiv:2209.10007*, 2022.
- [25] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019.
- [26] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, 2017.
- [27] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.
- [28] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 43(4):572–587, 2024.

- [29] Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28694–28698, 2025.
- [30] Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing with deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1205–1212. IEEE, 2021.
- [31] Yunlong Song, Kexin Shi, Robert Penicka, and Davide Scaramuzza. Learning perception-aware agile flight in cluttered environments. *arXiv preprint arXiv:2210.01841*, 2022.
- [32] Jiaxu Xing, Ismail Geles, Yunlong Song, Elie Aljalbout, and Davide Scaramuzza. Multi-task reinforcement learning for quadrotors. *IEEE Robotics and Automation Letters*, 2024.
- [33] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Tiezhen Wang, et al. Tensorflow lite micro: Embedded machine learning for tinyml systems. *Proceedings of machine learning and systems*, 3:800–811, 2021.
- [34] Miguel Beltrán-Escobar, Teresa E Alarcón, Jesse Y Rumbo-Morales, Sonia López, Gerardo Ortiz-Torres, and Felipe DJ Sorcia-Vázquez. A review on resource-constrained embedded vision systems-based tiny machine learning for robotic applications. *Algorithms*, 17(11):476, 2024.
- [35] Suhan Kim, Yi-Hsuan Hsiao, YuFan Chen, Jie Mao, and YuFeng Chen. Firefly: An insect-scale aerial robot powered by electroluminescent soft artificial muscles. *IEEE Robotics and Automation Letters*, 7(3):6950–6957, 2022.

- [36] Johannes M. James, Vikram Iyer, Yogesh M. Chukewad, Shyamnath Gollakota, and Sawyer B. Fuller. Liftoff of a 190 mg laser-powered aerial vehicle: The lightest wireless robot to fly. In *Robotics and Automation (ICRA), IEEE Int. Conf*, pages 1–8, 2018.
- [37] Noah T. Jafferis, E. Farrell Helbling, Michael Karpelson, and Robert J. Wood. Un-tethered flight of an insect-sized flapping-wing microscale aerial vehicle. *Nature*, 570: 491–495, 2019.
- [38] Takashi Ozaki, Norikazu Ohta, Tomohiko Jimbo, and Kanae Hamaguchi. A wireless radiofrequency-powered insect-scale flapping-wing aerial vehicle. *Nature Electronics*, 4 (11):845–852, 2021.
- [39] W. S. N. Trimmer. Microbots and micromechanical systems. *Sensors and Actuators*, 19:267–287, 1989.
- [40] Gilles Caprari, Thomas Estier, and Roland Siegwart. Fascination of down scaling-alice the sugar cube robot. *Journal of micromechatronics*, 1(3):177–189, 2001.
- [41] Jake J Abbott, Zoltan Nagy, Felix Beyeler, and Bradley J Nelson. Robotics in the small, part i: microbotics. *IEEE Robotics & Automation Magazine*, 14(2):92–103, 2007.
- [42] Igor Paprotny and Sarah Bergbreiter. Small-scale robotics: An introduction. In *workshop at the IEEE International Conference on Robotics and Automation*, pages 1–15. Springer, 2013.
- [43] Dario Floreano and Robert J Wood. Science, technology and the future of small autonomous drones. *Nature*, 521(7553):460–466, 2015.
- [44] Hector D. Escobar-Alvarez, Neil Johnson, Tom Hebble, Karl Klingebiel, Steven A. P. Quintero, Jacob Regenstien, and N. Andrew Browning. R-advance: Rapid adaptive prediction for vision-based autonomous navigation, control, and evasion. *Journal of*

- Field Robotics*, 35(1):91–100, 2018. ISSN 1556-4967. doi: 10.1002/rob.21744. URL <http://dx.doi.org/10.1002/rob.21744>.
- [45] Daniele Palossi, Antonio Loquercio, Francesco Conti, Eric Flamand, Davide Scaramuzza, and Luca Benini. A 64mw dnn-based visual navigation engine for autonomous nano-drones. *IEEE Internet of Things Journal*, 2019.
- [46] Mikkel Baun Kjærgaard, Henrik Blunck, Torben Godsk, Thomas Toftkjær, Dan Lund Christensen, and Kaj Grønbæk. Indoor positioning using gps revisited. In *Pervasive Computing: 8th International Conference, Pervasive 2010, Helsinki, Finland, May 17-20, 2010. Proceedings 8*, pages 38–56. Springer, 2010.
- [47] Yash P Talwkar, Andrew Adie, Vikram Iyer, and Sawyer B Fuller. Towards sensor autonomy in sub-gram flying insect robots: A lightweight and power-efficient avionics system. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9675–9681. IEEE, 2022.
- [48] Aryan Naveen, Jalil Morris, Christian Chan, Daniel Mhrous, E Farrell Helbling, Nak-Seung Patrick Hyun, Gage Hills, and Robert J Wood. Hardware-in-the-loop for characterization of embedded state estimation for flying microrobots. *arXiv preprint arXiv:2411.06382*, 2024.
- [49] Néstor O Pérez-Arancibia, Kevin Y Ma, Kevin C Galloway, Jack D Greenberg, and Robert J Wood. First controlled vertical flight of a biologically inspired microrobot. *Bioinspiration and Biomimetics*, 6(3):036009, Sep 2011. doi: 10.1088/1748-3182/6/3/036009. URL <http://dx.doi.org/10.1088/1748-3182/6/3/036009>.
- [50] Daniel S Drew and Kristofer SJ Pister. First takeoff of a flying microrobot with no moving parts. In *Manipulation, Automation and Robotics at Small Scales (MARSS), 2017 International Conference on*, pages 1–5, 2017.

- [51] Matěj Karásek, Florian T Muijres, Christophe De Wagter, Bart DW Remes, and Guido CHE de Croon. A tailless aerial robotic flapper reveals that flies use torque coupling in rapid banked turns. *Science*, 361(6407):1089–1094, 2018.
- [52] Sawyer B. Fuller. Four wings: An insect-sized aerial robot with steering ability and payload capacity for autonomy. *IEEE Robotics and Automation Letters*, 4(2):570–577, 2019.
- [53] Sawyer B. Fuller, Zhi Ern Teoh, Pakpong Chirarattananon, Néstor O. Pérez-Arancibia, Jack Greenberg, and Robert J. Wood. Stabilizing air dampers for hovering aerial robotics: design, insect-scale flight tests, and scaling. *Autonomous Robots*, February 2017. ISSN 1573-7527. doi: 10.1007/s10514-017-9623-3. URL <http://dx.doi.org/10.1007/s10514-017-9623-3>.
- [54] Hari Krishna Hari Prasad, Ravi Sankar Vaddi, Yogesh M Chukewad, Elma Dedic, Igor Novosselov, and Sawyer B Fuller. A laser-microfabricated electrohydrodynamic thruster for centimeter-scale aerial robots. *PloS one*, 15(4):e0231362, 2020.
- [55] Zhi Ern Teoh, Sawyer B. Fuller, Pakpong C. Chirarattananon, Néstor O. Pérez-Arancibia, Jack D. Greenberg, and Robert J. Wood. A hovering flapping-wing microrobot with altitude control and passive upright stability. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ Int. Conf.*, pages 3209–3216, Vilamoura, Algarve, Portugal, 7–12 October 2012.
- [56] Sawyer B Fuller, Michael Karpelson, Andrea Censi, Kevin Y Ma, and Robert J Wood. Controlling free flight of a robotic fly using an onboard vision sensor inspired by insect ocelli. *Journal of The Royal Society Interface*, 11(97):20140281, 2014.
- [57] Y. M. Chukewad, J. M. James, A. Singh, and S. B. Fuller. Robofly: An insect-sized robot with simplified fabrication that is capable of flight, ground, and water surface locomotion. *IEEE Transactions on Robotics*, 2021. (in press).

- [58] Antoine Beyeler. *Vision-based control of near-obstacle flight*. PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2009.
- [59] Graham K. Taylor and Holger G. Krapp. Sensory systems and flight stability: What do insects measure and why? In J. Casas and S.J. Simpson, editors, *Insect Mechanics and Control*, volume 34 of *Advances in Insect Physiology*, pages 231–316. Academic Press, New York, 2007. doi: 10.1016/S0065-2806(07)34005-8. URL <http://www.sciencedirect.com/science/article/pii/S0065280607340058>.
- [60] Julien Serres and Franck Ruffier. Optic flow-based robotics, 2016.
- [61] Julien R Serres and Franck Ruffier. Optic flow-based collision-free strategies: From insects to robots. *Arthropod structure & development*, 46(5):703–717, 2017.
- [62] Jan J. Koenderink and Andrea J. van Doorn. Facts on optic flow. *Biological Cybernetics*, 56:247–254, 1987.
- [63] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.
- [64] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.
- [65] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [66] C. De Wagter, S. Tijmons, B.D.W. Remes, and G.C.H.E. de Croon. Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system. In *Robotics and Automation (ICRA), 2014 IEEE Int. Conf.*, Hong Kong, China, 2–5 June 2014. IEEE.
- [67] Sawyer Fuller, Ben Greiner, Jason Moore, Richard Murray, René van Paassen, and

- Rory Yorke. The python control systems library (python-control). In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 4875–4881. IEEE, 2021.
- [68] Mark W Mueller, Michael Hamer, and Raffaello D’Andrea. Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadcopter state estimation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1730–1736, May 2015. doi: 10.1109/ICRA.2015.7139421.
- [69] John L Barron, David J Fleet, Steven S Beauchemin, and TA Burkitt. Performance of optical flow techniques. In *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 236–237. IEEE Computer Society, 1992.
- [70] Johannes James, Vikram Iyer, Yogesh Chukewad, Shyamnath Gollakota, and Sawyer 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)*, pages 3587–3594. IEEE, 2018.
- [71] Sawyer B. Fuller, E. Farrell Helbling, Pakpong Chirarattananon, and Robert J. Wood. Using a MEMS gyroscope to stabilize the attitude of a fly-sized hovering robots. In *Micro Air Vehicle (IMAV), 2014 Int. Conf.*, Delft, The Netherlands, August 12-15 2014.
- [72] Anita M Flynn. Gnat robots (and how they will change robotics). In *IEEE Micro Robots and Teleoperators Workshop: An Investigation of Micromechanical Structures, Actuators and Sensors*, volume 1, 1987.
- [73] Rodney A. Brooks and Anita M. Flynn. Fast, cheap, and out of control: A robot invasion of the solar system. *Journal of the British Interplanetary Society*, 42:478–485, 1989.

- [74] Xiufeng Yang, Ying Chen, Longlong Chang, Ariel A Calderón, and Néstor O Pérez-Arancibia. Bee+: A 95-mg four-winged insect-scale flying robot driven by twinned unimorph actuators. *IEEE Robotics and Automation Letters*, 4(4):4270–4277, 2019.
- [75] Joseph M Kahn, Randy H Katz, and Kristofer SJ Pister. Next century challenges: mobile networking for “smart dust”. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278, 1999.
- [76] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758, 2014.
- [77] Nishant Elkunchwar, Suvesha Chandrasekaran, Vikram Iyer, and Sawyer B. Fuller. Toward battery-free free flight: Duty cycled recharging of small drones. In *Int. Conf. Robotics and Intelligent Systems (IROS)*, 2021.
- [78] Vijay Kumar and Nathan Michael. Opportunities and challenges with autonomous micro aerial vehicles. *The International Journal of Robotics Research*, 31(11):1279–1291, 2012.
- [79] Leif Ristroph, Attila J. Bergou, Gunnar Ristroph, Katherine Coumes, Gordon J. Berman, John Guckenheimer, Z. Jane Wang, and Itai Cohen. Discovering the flight autostabilizer of fruit flies by inducing aerial stumbles. *Proceedings of the National Academy of Sciences*, 107(11):4820–4824, Mar 2010. doi: 10.1073/pnas.1000615107. URL <http://dx.doi.org/10.1073/pnas.1000615107>.
- [80] Leif Ristroph, Gunnar Ristroph, Svetlana Morozova, Attila J. Bergou, Song Chang, John Guckenheimer, Z. Jane Wang, and Itai Cohen. Active and passive stabilization of body pitch in insect flight. *Journal of The Royal Society Interface*, 10(85), 2013. doi: 10.1098/rsif.2013.0237. URL <http://rsif.royalsocietypublishing.org/content/10/85/20130237.abstract>.

- [81] J.-C. Zufferey and D. Floreano. Fly-inspired visual steering of an ultralight indoor aircraft. *IEEE Transactions on Robotics*, 22(1):137–146, Feb. 2006. doi: 10.1109/TRO.2005.858857.
- [82] Jean-Christophe Zufferey, Adam Klaptocz, Antoine Beyeler, Jean-Daniel Nicoud, and Dario Floreano. A 10-gram microflyer for vision-based indoor navigation. In *Intelligent Robots and Systems (IROS), 2006 IEEE/RSJ Int. Conf.*, pages 3267–3272, 2006.
- [83] J. Serres, D. Dray, F. Ruffier, and N. Franceschini. A vision-based autopilot for a miniature air vehicle: joint speed control and lateral obstacle avoidance. *Autonomous Robots*, 25:103–122, 2008.
- [84] Joseph Conroy, Gregory Gremillion, Badri Ranganathan, and J. Humbert. Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. *Autonomous Robots*, 27:189–198, 2009. ISSN 0929-5593. URL <http://dx.doi.org/10.1007/s10514-009-9140-0>. 10.1007/s10514-009-9140-0.
- [85] Antoine Beyeler, Jean-Christophe Zufferey, and Dario Floreano. Vision-based control of near-obstacle flight. *Autonomous Robots*, 27:201–219, 2009. ISSN 0929-5593. URL <http://dx.doi.org/10.1007/s10514-009-9139-6>. 10.1007/s10514-009-9139-6.
- [86] Chelsea M Sabo, Alex Cope, Kevin Gurney, Eleni Vasilaki, and James Marshall. Bio-inspired visual navigation for a quadcopter using optic flow. In *AIAA Infotech@Aerospace*, page 0404. 2016.
- [87] Patrick P Neumann, Victor Hernandez Bennetts, Achim J Lilienthal, Matthias Bartholmai, and Jochen H Schiller. Gas source localization with a micro-drone using bio-inspired and particle filter-based algorithms. *Advanced Robotics*, 27(9):725–738, 2013.
- [88] Javier Burgués, Victor Hernández, Achim J Lilienthal, and Santiago Marco. Smelling nano aerial vehicle for gas source localization and mapping. *Sensors*, 19(3):478, 2019.

- [89] Vikram Iyer, Ali Najafi, Johannes James, Sawyer Fuller, and Shyamnath Gollakota. Wireless steerable vision for live insects and insect-scale robots. *Science Robotics*, 5(44), 2020. doi: 10.1126/scirobotics.abb0839. URL <https://robotics.sciencemag.org/content/5/44/eabb0839>.
- [90] Jan J. Koenderink. Optic flow. *Vision Research*, 26(1):161–180, 1986.
- [91] Guido CHE de Croon, Julien JG Dupeyroux, Christophe De Wagter, Abhishek Chatterjee, Diana A Olejnik, and Franck Ruffier. Accommodating unobservability to control flight attitude with optic flow. *Nature*, 610(7932):485–490, 2022.
- [92] Sawyer B. Fuller, Michael Karpelson, Andrea Censi, Kevin Y. Ma, and Robert J. Wood. Controlling free flight of a robotic fly using an onboard vision sensor inspired by insect ocelli. *J. Royal Society Interface*, 11(97), August 2014. doi: 10.1098/rsif.2014.0281. URL <http://rsif.royalsocietypublishing.org/content/11/97/20140281.abstract>.
- [93] Zhan Tu, Fan Fei, Jian Zhang, and Xinyan Deng. An at-scale tailless flapping-wing hummingbird robot. i. design, optimization, and experimental validation. *IEEE Transactions on Robotics*, 2020.
- [94] Julian Förster. System identification of the crazyflie 2.0 nano quadrocopter. B.S. thesis, ETH Zurich, 2015.
- [95] Dominik Honegger, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1736–1741. IEEE, 2013.
- [96] Hanna Müller, Daniele Palossi, Stefan Mach, Francesco Conti, and Luca Benini. Fünfliber-drone: A modular open-platform 18-grams autonomous nano-drone. In *De-*

- sign*, *Automation Test in Europe Conference Exhibition (DATE 2021)*, pages 12–11, 2021.
- [97] Bernard Friedland. *Control system design: an introduction to state-space methods*. Courier Corporation, 2012.
- [98] Yash Talwekar, Vikram Iyer, and Sawyer B. Fuller. Towards sensor autonomy in subgram flying insect-robots: A lightweight and power-efficient avionics system. In *IEEE Int. Conf. Robotics and Autonomous Systems (ICRA)*, 2022.
- [99] E. Farrell Helbling, Sawyer B. Fuller, and Robert J. Wood. *Altitude Estimation and Control of an Insect-Scale Robot with an Onboard Proximity Sensor*, pages 57–69. Springer International Publishing, Cham, 2018. ISBN 978-3-319-51532-8. doi: 10.1007/978-3-319-51532-8_4.
- [100] Scott Hanson, ZhiYoong Foo, David Blaauw, and Dennis Sylvester. A 0.5 v sub-microwatt cmos image sensor with pulse-width modulation read-out. *IEEE Journal of Solid-State Circuits*, 45(4):759–767, 2010.
- [101] Meng-Ting Chung and Chih-Cheng Hsieh. A 0.5 v 4.95 μ w 11.8 fps pwm cmos imager with 82db dynamic range and 0.055% fixed-pattern noise. In *2012 IEEE International Solid-State Circuits Conference*, pages 114–116. IEEE, 2012.
- [102] Sivakumar Balasubramanian, Yogesh M. Chukewad, Johannes M. James, Geoffrey L. Barrows, and Sawyer 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)*, pages 1297–1302, 2018.
- [103] P.-E.J. Duhamel, N.O. Pérez-Arancibia, G.L. Barrows, and R.J. Wood. Biologically inspired optical-flow sensing for altitude control of flapping-wing microrobots. *Mechatronics, IEEE/ASME Transactions on*, 18(2):556–568, 2013. ISSN 1083-4435. doi: 10.1109/TMECH.2012.2225635.

- [104] Dietrich Burkhardt and Michael Gewecke. Mechanoreception in *Arthropoda*: the chain from stimulus to behavioral pattern. *Cold Spring Harbor Symposium on Quantitative Biology*, 30:601–614, 1965.
- [105] Petros Argyrakis, Alister Hamilton, Barbara Webb, Yaxiong Zhang, Theophile Gonos, and Rebecca Cheung. Fabrication and characterization of a wind sensor for integration with a neuron circuit. *Microelectron. Eng.*, 84(5-8):1749–1753, May 2007. ISSN 0167-9317. doi: 10.1016/j.mee.2007.01.174. URL <http://dx.doi.org/10.1016/j.mee.2007.01.174>.
- [106] Andrea Tagliabue, Aleix Paris, Suhan Kim, Regan Kubicek, Sarah Bergbreiter, and Jonathan P. How. Touch the wind: Simultaneous airflow, drag and interaction sensing on a multirotor, 2020.
- [107] Sawyer B. Fuller, Alexander Sands, Andreas Haggerty, Michael Karpelson, and Robert J. Wood. Estimating attitude and wind velocity using biomimetic sensors on a microrobotic bee. In *Robotics and Automation (ICRA), 2013 IEEE Int. Conf.*, pages 1374–1380, Karlsruhe, Germany, 6–10 May 2013.
- [108] Mahdi M. Sadeghi, Rebecca L. Peterson, and Khalil Najafi. Hair-based sensors for micro-autonomous systems. In Thomas George, M. Saif Islam, and Achyut Dutta, editors, *Proc. SPIE 8373, Micro- and Nanotechnology Sensors, Systems, and Applications IV, 83731L*, volume 8373, page 83731L. SPIE, 2012. doi: 10.1117/12.919860. URL <http://link.aip.org/link/?PSI/8373/83731L/1>.
- [109] Keshav Rajasekaran, Hyung Dae Bae, Sarah Bergbreiter, and Miao Yu. 3d printed bio-inspired hair sensor for directional airflow sensing. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8945–8950. IEEE, 2020.
- [110] Philippe Martin and Erwan Salaün. The true role of accelerometer feedback in quadro-

- tor control. In *2010 IEEE international conference on robotics and automation*, pages 1623–1629. IEEE, 2010.
- [111] Robert C Leishman, John C Macdonald, Randal W Beard, and Timothy W McLain. Quadrotors and accelerometers: State estimation with an improved dynamic model. *IEEE Control Systems*, 34(1):28–41, 2014.
- [112] Zhi Ern Teoh and Robert J. Wood. A flapping-wing microrobot with a differential angle-of-attack mechanism. In *Robotics and Automation (ICRA), 2013 IEEE Int. Conf.*, 2013.
- [113] B. Cheng and X. Deng. Translational and rotational damping of flapping flight and its dynamics and stability at hovering. *IEEE Transactions on Robotics*, 27(5):849–864, 2011.
- [114] Tyson L Hedrick, Bo Cheng, and Xinyan Deng. Wingbeat time and the scaling of passive rotational damping in flapping flight. *Science*, 324(5924):252–255, Apr 2009. doi: 10.1126/science.1168431. URL <http://dx.doi.org/10.1126/science.1168431>.
- [115] Michael W Oppenheimer, Isaac E Weintraub, David O Sigthorsson, and David B Doman. Control of a minimally actuated biomimetic vehicle using quarter-cycle wingbeat modulation. *Journal of Guidance, Control, and Dynamics*, pages 1–10, 2015.
- [116] B. Hassenstein and W. Reichardt. Systemtheoretische analyse der zeit-, reihenfolgen- und vorzeichenbewertung bei der bewegungsperzeption des rüsselkafers chlorophanus. *Zeitschrift Fur Naturforschung*, 11b:513–524, 1956.
- [117] Marcus Greiff. Modelling and control of the crazyflie quadrotor for aggressive and autonomous flight by optical flow driven state estimation. Master’s thesis, 2017.
- [118] Joel M Grasmeyer, Matthew T Keennon, et al. Development of the black widow micro air vehicle. *Progress in Astronautics and aeronautics*, 195:519–535, 2001.

- [119] E Farrell Helbling, Sawyer B Fuller, and Robert J Wood. Altitude estimation and control of an insect-scale robot with an onboard proximity sensor. In *Int. Symp. Robotics Res.*, 2015.
- [120] Jeremy E Niven, John C Anderson, and Simon B Laughlin. Fly photoreceptors demonstrate energy-information trade-offs in neural coding. *PLoS Biol*, 5(4):e116, Apr 2007. doi: 10.1371/journal.pbio.0050116. URL <http://dx.doi.org/10.1371/journal.pbio.0050116>.
- [121] Adam J. Rutkowski, Mikel M. Miller, Roger D. Quinn, and Mark A. Willis. Egomotion estimation with optic flow and air velocity sensors. *Biological Cybernetics*, 104(6):351–367, Jun 2011. doi: 10.1007/s00422-011-0440-z. URL <http://dx.doi.org/10.1007/s00422-011-0440-z>.
- [122] Sighard F Hoerner. *Fluid-dynamic drag: practical information on aerodynamic drag and hydrodynamic resistance*. Hoerner Fluid Dynamics, Midland Park, NJ, 1965.
- [123] Stephen Vogel. *Comparative Biomechanics: Life’s Physical World*. Princeton University Press, 2003.
- [124] Jeffrey S Pulskamp, Ronald G Polcawich, Ryan Q Rudy, Sarah S Bedair, Robert M Proie, Tony Ivanov, and Gabriel L Smith. Piezoelectric pzt mems technologies for small-scale robotics and rf applications. *MRS bulletin*, 37(11):1062–1070, 2012.
- [125] Xiaojun Yan, Mingjing Qi, and Liwei Lin. Self-lifting artificial insect wings via electrostatic flapping actuators. In *2015 28th IEEE International Conference on Micro Electro Mechanical Systems (MEMS)*, pages 22–25. IEEE, 2015.
- [126] Palak Bhushan and Claire J Tomlin. Design of the first sub-milligram flapping wing aerial vehicle. In *2019 IEEE 32nd International Conference on Micro Electro Mechanical Systems (MEMS)*, pages 2–5, 2019.

- [127] J. M. James and S. B. Fuller. A high-voltage power electronics unit for flying insect robots that can modulate wing thrust. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2021.
- [128] Seth Hollar, Anita Flynn, Colby Bellew, and KSJ Pister. Solar powered 10 mg silicon robot. In *The Sixteenth Annual International Conference on Micro Electro Mechanical Systems, 2003. MEMS-03 Kyoto. IEEE*, pages 706–711. IEEE, 2003.
- [129] Ke Sun, Teng-Sing Wei, Bok Yeop Ahn, Jung Yoon Seo, Shen J Dillon, and Jennifer A Lewis. 3d printing of interdigitated li-ion microbattery architectures. *Advanced materials*, 25(33):4539–4543, 2013.
- [130] JB Bates, NJ Dudney, B Neudecker, A Ueda, and CD Evans. Thin-film lithium and lithium-ion batteries. *Solid state ionics*, 135(1-4):33–45, 2000.
- [131] Mihai Duduta, Sebastien de Rivaz, David R Clarke, and Robert J Wood. Ultra-lightweight, high power density lithium-ion batteries. *Batteries & Supercaps*, 1(4):131–134, 2018.
- [132] Geoffrey L Barrows and C Neely. Mixed-mode vlsi optic flow sensors for in-flight control of a micro air vehicle. In *Critical Technologies for the future of Computing*, volume 4109, pages 52–63. SPIE, 2000.
- [133] Kyuseok Lee, Seokjun Park, Sung-Yun Park, Jihyun Cho, and Euisik Yoon. A 272.49 pj/pixel cmos image sensor with embedded object detection and bio-inspired 2d optic flow generation for nano-air-vehicle navigation. In *2017 Symposium on VLSI Circuits*, pages C294–C295. IEEE, 2017.
- [134] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *IEEE Transactions on*

- Pattern Analysis and Machine Intelligence*, 44(1):154–180, 2022. doi: 10.1109/TPAMI.2020.3008413.
- [135] Julien R Serres and Franck Ruffier. Optic flow-based collision-free strategies: From insects to robots. *Arthropod structure & development*, 46(5):703–717, 2017.
 - [136] Nicolas Franceschini, Franck Ruffier, and Julien Serres. A bio-inspired flying robot sheds light on insect piloting abilities. *Current Biology*, 17:1–7, 2007.
 - [137] Guido CHE de Croon. Monocular distance estimation with optical flow maneuvers and efference copies: a stability-based strategy. *Bioinspiration & biomimetics*, 11(1): 016004, 2016.
 - [138] Xiangyu Wu and Mark W Mueller. Using multiple short hops for multicopter navigation with only inertial sensors. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8559–8565. IEEE, 2020.
 - [139] Floris van Breugel. A nonlinear observability analysis of ambient wind estimation with uncalibrated sensors, inspired by insect neural encoding. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 1399–1406. IEEE, 2021.
 - [140] Shuo Han, Andrea Censi, Andrew D. Straw, and Richard M. Murray. A bio-plausible design for visual pose stabilization. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5679–5686, San Francisco, CA, 25–30 September 2010. IEEE.
 - [141] GCHE de Croon, C De Wagter, and T Seidl. Enhancing optical-flow-based control by learning visual appearance cues for flying robots. *Nature Machine Intelligence*, 3(1): 33–41, 2021.
 - [142] Sawyer B. Fuller and Richard M. Murray. A hovercraft robot that uses insect-inspired visual autocorrelation for motion control in a corridor. In *Robotics and Biomimetics*

- (*ROBIO*), *2011 IEEE Int. Conf.*, pages 1474–1481, Phuket, Thailand, 7–11 December 2011.
- [143] Zhitao Yu, Yash Talwekar, and Sawyer B. Fuller. Visual confined-space navigation using an efficient learned bilinear optic flow approximation for insect-scale robots. In *IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, 2022.
 - [144] Michael H. Dickinson, Fritz-Olaf Lehmann, and Sanjay P. Sane. Wing rotation and the aerodynamic basis of insect flight. *Science*, 284(5422):1954–1960, 1999.
 - [145] Pan Liu, Sanjay P Sane, Jean-Michel Mongeau, Jianguo Zhao, and Bo Cheng. Flies land upside down on a ceiling using rapid visually mediated rotational maneuvers. *Science advances*, 5(10):eaax1877, 2019.
 - [146] G. Fraenkel and J. W. S. Pringle. Biological sciences: Halteres of flies as gyroscopic organs of equilibrium. *Nature*, 141:919–920, May 1938. doi: 10.1038/141919a0.
 - [147] AL Eberle, BH Dickerson, Per G Reinhall, and TL Daniel. A new twist on gyroscopic sensing: body rotations lead to torsion in flapping, flexing insect wings. *Journal of the Royal Society Interface*, 12(104):20141088, 2015.
 - [148] W. Reichardt. *Movement Perception in Insects*, chapter Movement perception in Insects, pages 465–493. Academic Press, New York, 1969.
 - [149] E. Buchner. *Photoreception and Vision in Invertebrates*, chapter Behavioral Analysis of Spatial Vision in Insects, pages 561–621. Plenum, 1984.
 - [150] Ron O. Dror, David C. O’Carroll, and Simon B. Laughlin. Accuracy of velocity estimation by Reichardt Correlators. *Journal of the Optical Society of America A*, 18(2): 241–252, February 2001.
 - [151] Antoine Beyeler, Jean-Christophe Zufferey, and Dario Floreano. Vision-based control of near-obstacle flight. *Autonomous robots*, 27(3):201–219, 2009.

- [152] Quentin Bateux, Eric Marchand, Jürgen Leitner, François Chaumette, and Peter Corke. Training deep neural networks for visual servoing. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3307–3314. IEEE, 2018.
- [153] Abdulrahman Al-Shanoon and Haoxiang Lang. Robotic manipulation based on 3-d visual servoing and deep neural networks. *Robotics and Autonomous Systems*, page 104041, 2022.
- [154] Lei Shi, Cosmin Copot, and Steve Vanlanduit. A bayesian deep neural network for safe visual servoing in human–robot interaction. *Frontiers in Robotics and AI*, 8, 2021.
- [155] Graham K Taylor and Holger G Krapp. Sensory systems and flight stability: what do insects measure and why? *Advances in insect physiology*, 34:231–316, 2007.
- [156] Anita M Flynn. Gnat robots (and how they will change robotics). 1987.
- [157] Daniel S Drew, Brian Kilberg, and Kristofer SJ Pister. Future mesh-networked pico air vehicles. In *Unmanned Aircraft Systems (ICUAS), 2017 International Conference on*, pages 1075–1082, 2017.
- [158] Andrea Censi and Richard M Murray. Bootstrapping bilinear models of simple vehicles. *The International Journal of Robotics Research*, 34(8):1087–1113, 2015.
- [159] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Robotics Research*, pages 235–252. Springer, 2017.
- [160] Subramanian Ramasamy, Roberto Sabatini, Alessandro Gardi, and Jing Liu. Lidar obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid. *Aerospace Science and Technology*, 55:344–358, 2016.
- [161] Adam Bry, Abraham Bachrach, and Nicholas Roy. State estimation for aggressive

- flight in gps-denied environments using onboard sensing. In *2012 IEEE International Conference on Robotics and Automation*, pages 1–8. IEEE, 2012.
- [162] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.
- [163] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.
- [164] Abraham Bachrach, Ruijie He, and Nicholas Roy. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4):217–228, 2009.
- [165] Vishnu Sashank Dorbala, AH Abdul Hafez, and CV Jawahar. A deep learning approach for robust corridor following. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3712–3718. IEEE, 2019.
- [166] Aseem Saxena, Harit Pandya, Gourav Kumar, Ayush Gaud, and K Madhava Krishna. Exploring convolutional networks for end-to-end visual servoing. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3817–3823. IEEE, 2017.
- [167] Andrea Censi and Richard M Murray. Bootstrapping sensorimotor cascades: a group-theoretic perspective. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2056–2063. IEEE, 2011.
- [168] E. Farrell Helbling, Sawyer B. Fuller, and Robert J. Wood. Pitch and yaw control of a robotic insect using an onboard magnetometer. In *Robotics and Automation (ICRA), 2014 IEEE Int. Conf.*, pages 5516–5522, 2014.
- [169] James Sean Humbert and Andrew Maxwell Hyslop. Bioinspired visuomotor convergence. *IEEE Transactions on Robotics*, 26:121–130, 2010.

- [170] Anish Bhattacharya, Marco Cannici, Nishanth Rao, Yuezhan Tao, Vijay Kumar, Nikolai Matni, and Davide Scaramuzza. Monocular event-based vision for obstacle avoidance with a quadrotor. *arXiv preprint arXiv:2411.03303*, 2024.
- [171] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.