NEUROMORPHIC SENSING AND CONTROL OF AUTONOMOUS MICRO-AERIAL VEHICLES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Taylor S Clawson

December 2019

NEUROMORPHIC SENSING AND CONTROL OF AUTONOMOUS MICRO-AERIAL VEHICLES

Taylor S Clawson, Ph.D.

Cornell University 2019

Recent developments in manufacturing, processing capabilities, and sensor design point to a future in which sensor-equipped robots will assist humans by autonomously performing difficult, hazardous tasks such as visually inspecting utilities and infrastructure, locating a missing person, or monitoring potentially hazardous environmental conditions. To achieve this, perception algorithms must be developed which use data from on-board exteroceptive sensors to gather information on targets while navigating autonomously through unknown environments. Often, these tasks are best performed by highly maneuverable aerial robots which can easily avoid obstacles and access unique vantage points, but also require computationally efficient perception algorithms and power-efficient sensors due to their limited power budgets. Insect-scale flapping-wing robots represent an extreme, motivating example of aerial robots which, although highly maneuverable due to their extreme reduction in size and weight, also require high frequency sensing and control loops that operate on just milliwatts of power. However, conventional sensing and perception frameworks operate using too much data to operate at the high frequencies required by these robots. Energy-efficient, biologically inspired neuromorphic processors and sensors present a potential solution to this challenge. Neuromorphic chips and their software analog, spiking neural networks (SNNs), can be trained to approximate arbitrary functions while learning and adapting online.

Neuromorphic cameras consume only milliwatts of power despite operating with microsecond temporal precision. However, existing neuromorphic perception algorithms either collect only sparse information about the environment or do not account for a moving sensor and are thus inapplicable to flapping-wing robot navigation.

This work presents a framework of computationally efficient methods for neuromorphic perception and control to enable autonomous obstacle avoidance and target detection using highly agile micro aerial vehicles (MAVs). The SNN-based control method presented here is developed using a comprehensive, full-envelope flapping-wing flight dynamics model also presented in this work. This approach models flapping flight based on blade-element theory and is used to determine a broad class of set points, trim conditions, and quasi-steady maneuvers including coordinated turns. The model, analysis, and stability results are successfully validated experimentally for both stable and unstable modes. The SNN-based control approach is shown to be capable of controlling maneuvers including takeoff, landing, and coordinated turns throughout the flight envelope while adapting online to account for unmodeled parameter variations and disturbances. Computationally efficient neuromorphic visual perception techniques for obstacle avoidance and target detection are also developed comprising methods for dense optical flow estimation, dense monocular depth estimation, and independent motion detection. These methods leverage the high sensing rate of neuromorphic cameras to enable simple, linear assumptions which lead to improved accuracy and reduced computational cost compared with existing methods. In total, the methods presented here represent a computationally efficient framework for target tracking and obstacle avoidance with autonomous micro aerial vehicles.

BIOGRAPHICAL SKETCH

Taylor Clawson is a PhD student in the Laboratory for Intelligent Systems and Controls (LISC) at Cornell University. He received the B.S. degree in Mechanical Engineering from Utah State University while working as a part-time software developer. During his senior year, he and his team took first place in a national Air Force Research Lab (AFRL) design competition and appeared subsequently in Popular Science. Upon graduating, he was employed at Northrop Grumman as an engineer and software developer for two years, where he worked on national defense systems. His research interests include numerical optimization, biologically-inspired spiking neural network control methods, flight dynamics, and visual perception. He is a 2018 Cornell College of Engineering Commercialization Fellow and the Entrepreneurial Lead for a 2018 National NSF I-Corps Team.

Dedicated to my loving and supportive family,

especially my wonderful, patient wife Jess.

ACKNOWLEDGEMENTS

This work was supported by the Office of Naval Research Grant N00014-17-1-2614 and by the National Science Foundation Grant IIP-1838470.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Significant recent developments in manufacturing, processing capabilities, and sensor design have created the foundation of a future in which sensor-laden robots assist human activities by autonomously performing tasks in remote and hazardous environments. Such tasks frequently involve gathering information about a target. For example, visually inspecting utilities and infrastructure, locating a missing person in need of assistance, or monitoring potentially hazardous environmental conditions in a specific area. To enable robots to reliably perform these tasks in unknown, cluttered environments requires tightly integrated perception and control methods which maximize the information obtained about a target while maintaining stability in the presence of potential disturbances and unexpected conditions.

The ability of a robot to gather information on a target in cluttered environments increases significantly if it can be made more maneuverable. For instance, flying robots such as quadcopters can view targets from additional perspectives which are not available to ground-based robots. However, the maneuverability of a robot depends not only on the physical capabilities of the robot, but also on control methods which are effective throughout the robot's operational envelope. Such methods must be based on accurate models of the robot dynamics to guarantee stability and performance during maneuvers required for obstacle avoidance and information gathering. This is especially true for small, agile robots due to the rapid timescales of their dynamics, which can quickly lead to instability and crashes if unaccounted for.

Recently, micro aerial vehicles (MAVs) have been developed on the insect-

scale, which have the potential advantages demonstrated by their biological counterparts of accessing small, confined spaces and flying safely even in the event of collisions. These insect-scale flapping-wing robots represent an extreme, motivating example of highly maneuverable robots which, when equipped with power-efficient exteroceptive sensors, can effectively gather information on targets in cluttered environments which may be impassible to larger robots and humans. However, autonomous flight at the insect scale poses several challenges which must first be overcome. Insect-scale flapping robots are challenging to stabilize due to their fast dynamics, sensitivity to unmodeled parameter variations such as wing asymmetries, and the periodic nature of their control input. On-board controllers should be capable of adapting to unexpected disturbances while operating on just milliwatts of power. Similarly, on-board sensors such as cameras must operate on the same power budget while maintaining a sufficiently high sensing rate to cope with the rapid movement of the robot during flight.

Neuromorphic, or event-based hardware presents a potential solution to these challenges through energy-efficient, adaptable hardware inspired by biological nervous systems. Energy efficient neuromorphic chips have been developed which mimic networks of biological neurons and can be trained to approximate arbitrary functions while rapidly adapting online to improve performance. The term "Neuromorphic sensors" encompasses many different classes of sensors, all of which are similarly inspired by biological nervous systems. Generally, neuromorphic sensors communicate information through events that indicate the change in a measured value, such as the light intensity in a vision sensor or an acoustic signal [67] for a spatial audition sensor. These sensors eliminate the redundancy in measured and transmitted data by signaling only

changes in the measured value, typically in an asynchronous fashion and at a far lower power requirement than their more traditional counterparts. In particular, neuromorphic cameras have been demonstrated which consume only milliwatts of power while measuring changes in the camera field of view (FOV) with microsecond precision [12,25,66,83,121]. Despite the high temporal resolution of these cameras, they typically generate on the order of 1/10th the amount of data produced by a traditional frame-based camera operating at 30Hz.

This work presents a framework of computationally-efficient methods for neuromorphic perception and control to enable autonomous obstacle avoidance and target detection in agile mobile robotic platforms. To facilitate the application of these methods to insect-scale flapping-wing robots, a method for modeling the full-envelope flight dynamics of flapping wing aerial vehicles is also presented. Although there exists a large variety of robot and actuation designs, some commonalities have begun to emerge in recent years because of the effectiveness and feasibility of minimally actuated flapping flight control. Agile full-envelope flight, however, requires the development of comprehensive dynamic models able to capture all dominant flight modes and regimes of the robot. This paper presents a novel approach for modeling the full-envelope dynamics of periodic flapping flight based on blade-element theory. By this approach, a broad class of set points, trim conditions, and quasi-steady maneuvers, including coordinated turns are determined. A new approach based on dominant eigenplanes is also developed for analyzing the stability and dominant linear modes of the periodic dynamic model. Using the dominant eigenplanes, the new model, analysis, and stability results are successfully validated experimentally for both stable and unstable modes, including flight conditions leading to rapidly uncontrolled tumbling.

An adaptive flight control method using spiking neural networks (SNNs) is developed based on the full-envelope flight dynamics model. This adaptive SNN control approach is shown to be capable of controlling maneuvers throughout the flight envelope of a flapping-wing robot. Furthermore, it is capable of rapidly adapting online to unmodeled parameter variations such as asymmetries in the physical parameters of the robot wings and actuators. The controller is shown to adapt rapidly during a simulated flight test and requires a total of only 800 neurons, allowing it to be implemented with minimal power requirements in neuromorphic hardware. The control method is then extended to enable full-envelope flight control, including takeoff, landing, and coordinated turns.

Finally, computationally-efficient visual perception methods for autonomous obstacle avoidance and target tracking with event cameras are developed. First, an approach for event-based dense optical flow estimation is developed, which can be used directly for obstacle avoidance or motion-based image segmentation. The approach presented here demonstrates improved accuracy over existing methods while being implementable entirely through efficient linear convolutions. The underlying assumptions from the optical flow method are then extended to enable the computation of dense monocular depth estimation. In contrast to existing methods, the depth estimation method computed by this approach remains accurate near the focus of expansion (FOE), which corresponds to the direction of travel, and is thus directly usable for real time obstacle avoidance in autonomous aerial vehicles. To facilitate the detection of moving targets in the FOV of a translating and rotating event camera mounted to an aerial vehicle, an approach for independent motion detection is presented. This approach uses the estimated depth of points in the FOV to segment station-

ary and moving objects while accounting for the motion of the camera. In total, the methods presented here represent a computationally-efficient framework for real-time obstacle avoidance and target tracking with autonomous aerial vehicles using a single event-based camera.

CHAPTER 2

MODELING AND ANALYSIS OF MINIMALLY ACTUATED FULL-ENVELOPE FLAPPING-WING FLIGHT DYNAMICS

2.1 Introduction

New fabrication techniques have recently enabled the miniaturization of flapping micro aerial vehicles (MAVs) to insect-scale robots that present many advantages and potential future applications [1, 3, 24, 41, 43, 51, 55, 58, 61, 62, 68, 69, 113]. Besides being lower in cost and allowing access to highly confined spaces, these insect-scale robots are safer to operate near people and more agile than larger MAVs [43]. As the scale of the robot decreases, however, conventional propulsion methods, such as propellers, become extremely inefficient because low Reynold's numbers and increased losses in the electromagnetic motors cause significant reductions in lift-to-drag ratio [43, 55]. Flapping-wing robots provide a biologically inspired alternative that, especially at very small size and weight, increases both maneuverability and survivability and replicates the power-efficient and agile flight observed in many insects.

Although many flapping flight controllers have been successfully demonstrated in recent years, their effectiveness has been largely confined to the hovering regime and has otherwise been hindered by the lack of accurate full-envelope flight dynamics models and open-loop stability results [16, 17, 19, 21, 27,31,34,47,80,82]. Historically, full-envelope modeling and analysis of conventional fixed-wing or rotary-wing flight dynamics have enabled a detailed understanding and analysis of all vehicle flight modes and, as a result, the development of full-envelope robust and stable flight controllers [40, 77, 99, 100]. As for

conventional aircraft, a full-envelope flapping-wing flight model must capture the vehicle's dominant dynamics for all possible quasi-steady flight conditions, also known as set points [99,100].

Existing flapping-wing flight models to date have assumed prescribed kinematic wing motions that could not always be accurately tracked by flapping wings and required precise measurements of wing kinematics [32,80] or aerodynamic forces [16,88]. As a result, existing approaches cannot be extended to full-envelope flight dynamics modeling of flapping robots, such as minimally actuated designs, that lack direct control authority over wing pitch angles.

The approach presented in this paper enables the accurate modeling and analysis of flapping-wing robots by capturing the dominant forces produced by periodic flapping wing motions using a blade-element approach originally proposed for insect flight modeling in [28]. The approach accounts for all six body degrees-of-freedom (DOF) as well as the two rotational DOF of each wing, by assuming a quasi-steady aerodynamic model of forces and moments that has been shown to significantly lower the computational complexity when compared to computational fluid dynamics (CFD) approaches [2,30,41,79,91,92,103,109,111,114,118].

Many flapping-wing flight models also ignore the instantaneous coupling between the wings and the body by averaging the aerodynamic forces over a flapping period [46,104]. So called stroke-averaged models, for example, average an assumed wing kinematic trajectory or measure average forces experimentally. Consequently, the stability analysis of these models fails to account for the effects of perturbations in the body configuration on the wing dynamics and, by extension, on the aerodynamic forces on the system. Conversely,

(a) (b)

Figure 2.1: The method presented in this paper is directly applicable to dual-wing, minimally actuated designs such as (a) and can be easily extended to accommodate designs with fixed tails (b) and multiple pairs of wings.

the approach developed in this paper can lead to more accurate stability analysis and results by including the passive dynamics of wing pitching and their instantaneous coupling with the body dynamics.

Previous stability and dominant-mode results were also limited to hovering set points or ascending flight [14,15,54,65,88,102,105,115,117,124]. Recent studies extended these approaches to forward level flight but were unable to provide useful results for other flapping-flight regimes [39, 102, 116, 117]. In addition to providing a full-envelope dynamic model that is experimentally validated in a broad range of stable and unstable modes, the approach presented in this paper also enables the determination of the robot full flight envelope and all of the quasi-steady set points in it. By analyzing the dynamic model at a broad range of set points, including hovering, forward flight, steady turns, ascending, and descending flight, the dominant modes and corresponding stability results are determined throughout the robot flight envelope.

The new dynamic modeling and analysis approach presented in this paper is successfully demonstrated both numerically and experimentally on the two-wing minimally actuated insect-scale flapping robot known as as the RoboBee [42, 68]. It is also directly applicable to many other popular flapping robots

such as the Cox piezo flier [24], the CMU flapping-wing robot [57, 58], the AFRL piezo-driven flapping-wing robot [3], and many others insect-like robots [63, 98, 126]. As explained in Section 2.3, the approach can be easily extended to other flapping-wing configurations characterized by additional wings (Fig. 2.1) [86] or rigid tails [89]. Because it accounts for passive wing-pitch dynamics, the approach can be applied to robots characterized by piezoelectric, electromagnetic, and motor-driven drive mechanisms [24,42,57,58,63,68,81,89,98,126].

Finally, although flapping-robot designs employ a variety of control strategies, some commonalities have begun to emerge in recent years demonstrating the effectiveness and feasibility of minimally actuated control of flapping-wing flight [24,119]. Other robot designs operate both wings at a fixed frequency and control pitch and roll torque by varying the mean stroke angle and stroke amplitude of each wing independently, as shown in Figs. 2.3b and 2.3c, substantially improving power efficiency [107]. The modeling approach presented in this paper can accommodate all of these actuator designs, as well as emerging ones recently proposed by the authors [34, 68, 80]. Additionally, the flapping-wing yaw control method presented in Section 2.3 avoids drawbacks such as undesirable vibrations by allowing the robot to operate at a fixed frequency without requiring wing rotation stops.

2.2 Problem Formulation and Assumptions

This paper presents an approach for modeling and analyzing the full-envelope 3D flight dynamics of a class of flapping-wing robots that are minimally-actuated, have a single pair of periodically-flapping wings, and have a single

rigid frame. For simplicity, and as an example case, the model presented here has no additional fixed wings, tails or other fixed aerodynamic surfaces. The model easily includes robots with additional wing pairs or tails, shown in Fig. 2.1, as discussed in section 2.3. This paper develops an approach for deriving flapping robot dynamic models in standard form,

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t\right], \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state, $\mathbf{u} \in \mathbb{R}^m$ is the control input, $\mathbf{p} \in \mathbb{R}^l$ are the physical parameters of the robot, and the initial conditions, $\mathbf{x}_0$, are known. The present paper does not include feedback control. The model has a 12-dimensional configuration space, which is represented with variables which specify the geometry of the robot at any instant. Three configuration variables specify body translation, three specify body position, and three for each wing specify the wing orientation (for a total of six variables for the wings). Kinematic constraints on the wings reduce the system degrees of freedom down to 8, which are represented by the dynamic state variables comprising the robot body position and orientation as well as the pitch angle of each wing. These degrees of freedom are represented in the generalized coordinate vector $\mathbf{q} \in \mathbb{R}^{(n/2)}$ which, together with its derivative, comprise the state vector $\mathbf{x} = [\mathbf{q} \quad \dot{\mathbf{q}}]^T$. The position of the robot body and its yaw angle comprise the ignorable coordinates or symmetry variables, so called because any solution to (2.1) is valid for any choice of values in these variables.

Four right-handed reference frames, shown in Fig. 2.2, are used to describe the motion of the robot. The inertial frame $\mathcal{F}_f$ is fixed to the ground and does not move, the body frame $\mathcal{F}_b$ is fixed to the robot body, and the right and left wing reference frames $\mathcal{F}_r$ and $\mathcal{F}_l$ are fixed to the respective wings. The orthonormal basis corresponding to each reference frame is defined as $\{\mathbf{e}_1^i, \mathbf{e}_2^i, \mathbf{e}_3^i\}$, where the

subscript of the reference frame is substituted for i. The left and right wings are modeled as thin, rigid plates of negligible thickness, and are denoted by the rigid objects $\mathcal{L}$ and $\mathcal{R}$, respectively. The wings are attached to the robot body, which is modeled as a 3D rigid object $\mathcal{B}$, at the point A, which is located at a distance d from the body center of gravity G in the $-\mathbf{e}_3^b$ direction.

The model presented here is applicable to *minimally-actuated* flapping-wing robots, for which only the right and left wing stroke angles are actuated, and where wing pitch varies passively. In other words, the non-zero elements in the Jacobian $\partial\mathbf{f}/\partial\mathbf{u}$ are only in the rows corresponding to the governing equations for the right and left stroke angle accelerations ($\ddot{\phi}_r$ and $\ddot{\phi}_l$), with

$$\operatorname{rank}\left(\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)}{\partial \mathbf{u}}\right) = 2, \qquad \forall\ \mathbf{x},\ \mathbf{u},\ \mathbf{p},\ t \tag{2.2}$$

The three translational degrees of freedom in the system are represented by the inertial coordinates x, y, and z of the body center of gravity, G, defined with respect to $\mathcal{F}_f$ as $\mathbf{r}_G = x\mathbf{e}_1^f + y\mathbf{e}_2^f + z\mathbf{e}_3^f$. The three rotational degrees of freedom of the robot body are represented by three Euler angles: yaw ϕ, roll θ, and pitch ψ. The sequence of Euler angle rotations from the inertial frame to the body frame begins with a rotation about $\mathbf{e}_3^f$ by ϕ, followed by a rotation about an inertial axis coincident with the intermediate $\mathbf{e}_1^b$ axis by θ, and finally a rotation about $\mathbf{e}_2^b$ by ψ. The orientation of each wing relative to the body is defined using the nominal stroke plane, defined as the set of all points $\mathbf{r}_P$ such that $\mathbf{e}_3^b \cdot (\mathbf{r}_P - \mathbf{r}_A) = 0$. As shown in Fig. 2.2, the orientation of the right wing relative to the body frame is given by three Euler angles: the stroke angle $\phi_r \in [-\pi,\ \pi]$, stroke-plane deviation $\theta_r \in [-\pi,\ \pi]$, and the wing pitch angle $\psi_r \in [-\pi,\ \pi]$. The orientation of $\mathcal{F}_l$ relative to the body frame is defined similarly, but begins with a rotation about $\mathbf{e}_3^b$ by π so that $\mathbf{e}_2^l$ points in the positive span-wise direction of the left wing.

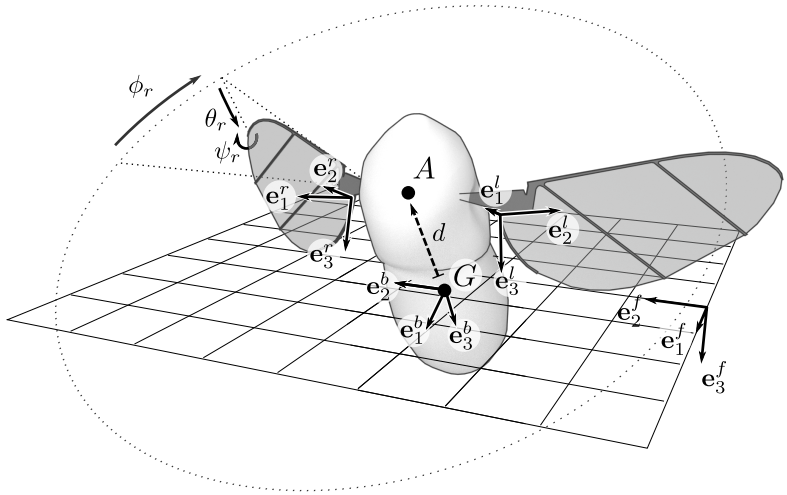


Figure 2.2: Wing Euler angles and reference frames for the robot body and each wing are shown on the robot, as well as the body center of gravity and wing rotation point.

The stroke angle of each wing (ϕ_r and ϕ_l) is assumed to be determined entirely by the control input and the actuator dynamics and does not depend on other state variables. The stroke-plane deviation angles (θ_r and θ_l) are assumed to be uniquely determined from the corresponding stroke angle (ϕ_r or ϕ_l) and physical parameters $\mathbf{p}$ through a continuous and differentiable mapping Γ:

$$\theta_r = \Gamma(\phi_r, \mathbf{p}), \qquad \theta_l = \Gamma(\phi_l, \mathbf{p}) \tag{2.3}$$

This mapping is typically the same for both the left and right wings and is determined by the geometry of the actuators and hinge mechanisms particular to the design of a given robot. The wing pitch angles (ψ_r and ψ_l) are functions of the control input $\mathbf{u}$ and the other robot state variables, i.e. they are determined by dynamics instead of kinematics and thus are included in the state vector $\mathbf{x}$. The complete state vector is written in terms of the generalized configuration coordinates as

$$\mathbf{x} = \begin{bmatrix} \mathbf{q} & \dot{\mathbf{q}} \end{bmatrix}^T, \qquad \mathbf{q} = \Big[\underbrace{x \quad y \quad z}_{\text{body position}} \quad \underbrace{\phi \quad \theta \quad \psi}_{\text{body rotation}} \quad \underbrace{\phi_r \quad \psi_r \quad \phi_l \quad \psi_l}_{\text{wing rotation}} \Big]^T \tag{2.4}$$

The control input directly affects the amplitude and mean offset of the right and left wing stroke angles through a forcing function on the actuator dynamics described in section 2.3. It comprises the amplitude input u_a, the pitch input u_p, the yaw input u_y, and the roll input u_r

$$\mathbf{u}(t) = \begin{bmatrix} u_a(t) & u_p(t) & u_y(t) & u_r(t) \end{bmatrix}^T \tag{2.5}$$

The model developed here is used to analyze the open loop dynamics of insect-scale flapping-wing flight at M quasi-steady set points $\mathcal{P} = \{(\mathbf{x}^*(t), \mathbf{u}^*)_i : i = 1, \ldots, M\}$. Each set point comprises a constant control input $\mathbf{u}^*$ and a periodic trajectory $\mathbf{x}^*(t)$, which is a solution to (2.1). The non-ignorable dynamic variables in $\mathbf{x}^*(t)$, which comprises the complete state vector except for the body position and yaw angle and is denoted by $\boldsymbol{\chi}^*(t)$, must be periodic with the flapping period T:

$$\boldsymbol{\chi}^*(t + jT) - \boldsymbol{\chi}^*(t) = \mathbf{0}, \qquad \forall j \in \mathbb{N} \tag{2.6}$$

The concept of steady flight is defined for this paper as flight where the non-ignorable dynamic variables are periodic with the flapping period T. The analysis is performed within the steady flight envelope, which is the set of conditions where steady flight is possible within the range of allowable physical parameters of the system. In this paper, a method for computing the steady flight envelope based on the physical parameters of the system is shown, a method of solving for the set points within the flight envelope is presented, and the dynamic stability and dominant modes of motion are analyzed at several representative points. Additionally, a simple method for yaw control is proposed and the model is validated against experimental data collected from open loop tests with a physical flapping wing robot.

2.3 Flight Dynamics Modeling

We use the Newton-Euler equations for linear and angular momentum balance as applied to the body and to each flapping wing to derive the equations of motion for the system. The model developed here accounts for the coupling between body dynamics and passive wing pitching dynamics in minimally-actuated flapping-wing robots to facilitate accurate stability analysis and flight envelope computations. The forces acting on the system are written in terms of the mass m of the robot body, the masses m_l and m_r of the left and right wings, the gravity vector $\mathbf{g}$, and the total aerodynamic forces $\mathbf{F}_l$ and $\mathbf{F}_r$ acting on the left and right wings. Following previous models of insect flight such as [14], the model presented here neglects aerodynamic forces acting on the robot body and interactions between the wings. This assumption is motivated by previous work on modeling insect flight which has shown that the net aerodynamic forces and moments over a stroke are accurately computed by properly tuning the coefficients of the rigid-wing model [2]. External moments on the system include rotational damping moments $\mathbf{M}_{rd,l}$ and $\mathbf{M}_{rd,r}$ from the left and right wings. The external moments depend on the position vectors $\mathbf{r}_{GP_l}$ and $\mathbf{r}_{GP_r}$ from G to the centers of pressure P_l and P_r of the left and right wings, and the position vectors $\mathbf{r}_{GL}$ and $\mathbf{r}_{GR}$ from G to the centers of gravity L and R of the left and right wings. Linear momentum balance for the system is written in terms of these forces, the acceleration $\mathbf{a}_G$ of the robot body center of gravity G located at $\mathbf{r}_G$, and the accelerations $\mathbf{a}_L$ and $\mathbf{a}_R$ of the centers of gravity of the left and right wings

$$(m + m_l + m_r)\mathbf{g} + \mathbf{F}_r + \mathbf{F}_l = m\mathbf{a}_G + m_l\mathbf{a}_L + m_r\mathbf{a}_R \tag{2.7}$$

Angular momentum balance of the entire system about G is

$$\mathbf{M}_{rd,l} + \mathbf{r}_{GP_l} \times \mathbf{F}_l + \mathbf{r}_{GL} \times m_l\mathbf{g} + \mathbf{M}_{rd,r} + \mathbf{r}_{GP_r} \times \mathbf{F}_r + \mathbf{r}_{GR} \times m_r\mathbf{g} = \dot{\mathbf{H}}_b + \dot{\mathbf{H}}_l + \dot{\mathbf{H}}_r \tag{2.8}$$

where $\dot{\mathbf{H}}_b$, $\dot{\mathbf{H}}_l$, and $\dot{\mathbf{H}}_r$ are the time rate of change in angular momentum of the body, left, and right wings, respectively, about G. Note that G is a non-inertial point, so the acceleration of each point relative to the inertial frame must be included in the angular momentum terms as shown in (2.9)-(2.11). Each of these vector derivatives can be written using the inertia of the corresponding rigid body about its own center of gravity ($\mathbf{I}_b$, $\mathbf{I}_l$, and $\mathbf{I}_r$) and the angular rates of each rigid body ($\boldsymbol{\omega}_b$, $\boldsymbol{\omega}_l$, and $\boldsymbol{\omega}_r$),

$$\dot{\mathbf{H}}_b = \cancelto{0}{\mathbf{r}_{GG}} \times m_b \mathbf{a}_B + \mathbf{I}_b \dot{\boldsymbol{\omega}}_b + \omega_b \times \mathbf{I}_b \omega_b, \tag{2.9}$$

$$\dot{\mathbf{H}}_r = \mathbf{r}_{GR} \times m_r \mathbf{a}_R + \mathbf{I}_r \dot{\boldsymbol{\omega}}_r + \omega_r \times \mathbf{I}_r \omega_r \tag{2.10}$$

$$\dot{\mathbf{H}}_l = \mathbf{r}_{GL} \times m_l \mathbf{a}_L + \mathbf{I}_l \dot{\boldsymbol{\omega}}_l + \omega_l \times \mathbf{I}_l \omega_l \tag{2.11}$$

where the aerodynamic forces and moments ($\mathbf{F}_r$, $\mathbf{F}_l$, $\mathbf{M}_{rd,r}$, and $\mathbf{M}_{rd,l}$) are derived in Section 2.4.

Independent equations for the stroke angles, stroke-plane deviation angles, and wing pitch angles are obtained in order to model the motion of the wings. As noted, the flight model in this paper assumes no direct control authority over the wing pitch angles, which are each modeled as a degree of freedom. To determine their values, angular momentum balance is computed for each wing about the wing attachment point A in the span-wise direction ($\mathbf{e}_2^l$ and $\mathbf{e}_2^r$),

$$\mathbf{e}_2^l \cdot (\mathbf{M}_{rd,l} + \mathbf{r}_{AP_l} \times \mathbf{F}_l + \mathbf{r}_{AL} \times m_l \mathbf{g} + \mathbf{M}_k) = \mathbf{e}_2^l \cdot (\mathbf{r}_{AL} \times m_l \mathbf{a}_L + \mathbf{I}_l \dot{\boldsymbol{\omega}}_l + \omega_l \times \mathbf{I}_l \omega_l) \tag{2.12}$$

$$\mathbf{e}_2^r \cdot (\mathbf{M}_{rd,r} + \mathbf{r}_{AP_r} \times \mathbf{F}_r + \mathbf{r}_{AR} \times m_r \mathbf{g} + \mathbf{M}_k) = \mathbf{e}_2^r \cdot (\mathbf{r}_{AR} \times m_r \mathbf{a}_R + \mathbf{I}_r \dot{\boldsymbol{\omega}}_r + \omega_r \times \mathbf{I}_r \omega_r) \tag{2.13}$$

where $\mathbf{r}_{AP_l}$ and $\mathbf{r}_{AP_r}$ are the position vectors from A to the centers of pressure of each wing, $\mathbf{r}_{AL}$ and $\mathbf{r}_{AR}$ are the position vectors from A to the centers of gravity of each wing, and $\mathbf{M}_{k,l} = -k_w \psi_l$ and $\mathbf{M}_{k,r} = -k_w \psi_r$ are moments caused by torsional springs in the hinges of each wing with spring constant k_w.

The stroke-plane deviation angles of the wings are constrained by the geometry of the wing hinge and actuator assembly, characterized by the mapping Γ in (2.3). In many common flapping-wing designs, there is no stroke plane deviation and thus $\Gamma(t, \mathbf{p}) = 0$. Alternative forms of Γ include common wing kinematic forms found in insect flight literature. One such form from [14] characterizes oscillations in the stroke-plane deviation,

$$\Gamma(t, \mathbf{p}) = \theta_0 + \theta_m \cos(N \cdot \frac{2\pi}{T} t + \delta_\theta) \tag{2.14}$$

which depends on the nominal offset θ_0, the deviation amplitude ϕ_m, the phase shift δ_θ, and $N \in \{1,\ 2\}$. A single vertical oscillation occurs per stroke with $N = 1$, whereas $N = 2$ results in the wing tip tracing a figure-eight motion.

The stroke angle for each wing is modeled as a decoupled second-order linear system. Previous studies have demonstrated that linear models can capture key aspects of the wing and actuator dynamics, and eliminate the need to accurately measure a large number of physical parameters of the hinge mechanism [41]. Note that the validity of decoupling these equations from external forces depends on the advance ratio, defined as the ratio of the flight speed to the mean wingtip velocity, remaining small. Thus, the assumption becomes invalid as the advance ratio approaches unity. However, with only rare exceptions, the advance ratio for flapping-wing flight remains less than unity even when the robot or insect is flying at its top speed [36]. The model includes the parameters ω_w and ζ_w, which can be chosen to match the natural frequency and damping ratio of the physical system

$$\ddot{\phi}_r(t) + 2\zeta_w \omega_w \dot{\phi}_r(t) + \omega_w^2 \phi_r(t) = f_r[\mathbf{u}(t), \mathbf{p}] \tag{2.15}$$

$$\ddot{\phi}_l(t) + 2\zeta_w \omega_w \dot{\phi}_l(t) + \omega_w^2 \phi_l(t) = f_l[\mathbf{u}(t), \mathbf{p}] \tag{2.16}$$

The forcing functions $f_r(\mathbf{u}, \mathbf{p})$ and $f_l(\mathbf{u}, \mathbf{p})$ represent the actuator drive signals and

are determined by the forcing frequency ω_f and the control inputs $\mathbf{u}$. The form of the forcing functions used in this work are given in (2.17) and (2.18). In general, the chosen functional form of the forcing function, such as the split-cycle approach from [33, 68, 80, 95], depends on the control inputs and the actuator design for a given robot.

An alternate functional form for the drive signals is proposed here, which is capable of affecting roll, pitch, and yaw torques by adjusting only the mean stroke angle and stroke amplitude of both wings. In this Raibert-like control method, pitch and roll are controlled similarly to many existing designs, such as [68], but yaw torque is controlled by adjusting the mean stroke angle of the right and left wings in opposite directions as shown in Fig. 2.3. This method generates a yaw torque by affecting the relative phase shift between the pitch angles of the left and right wings [87], as will be shown in section 2.8. This method is advantageous because it allows the system to flap at the resonant frequency of the wing and actuator assembly to increase efficiency and does not require wing rotation stops, which can cause unwanted vibrations and thus further reduces efficiency [3, 122]. The amplitude of both forcing functions used in this method are affected equally by u_a. The roll input u_r increases the amplitude of the force on one wing while decreasing the amplitude on the other. The forcing frequency ω_f is typically fixed near the resonant frequency of the actuator assembly to increase efficiency. The pitch input u_p biases the stroke angles symmetrically, and the yaw input u_y biases one stroke angle forwards and the other rearwards. In terms of these control inputs, the forcing functions are

$$f_r(\mathbf{u}(t), \mathbf{p}) = \frac{u_a(t) - u_r(t)}{2} \sin(\omega_f t) - u_p(t) - u_y(t) \tag{2.17}$$

$$f_l(\mathbf{u}(t), \mathbf{p}) = \frac{u_a(t) + u_r(t)}{2} \sin(\omega_f t) - u_p(t) + u_y(t) \tag{2.18}$$

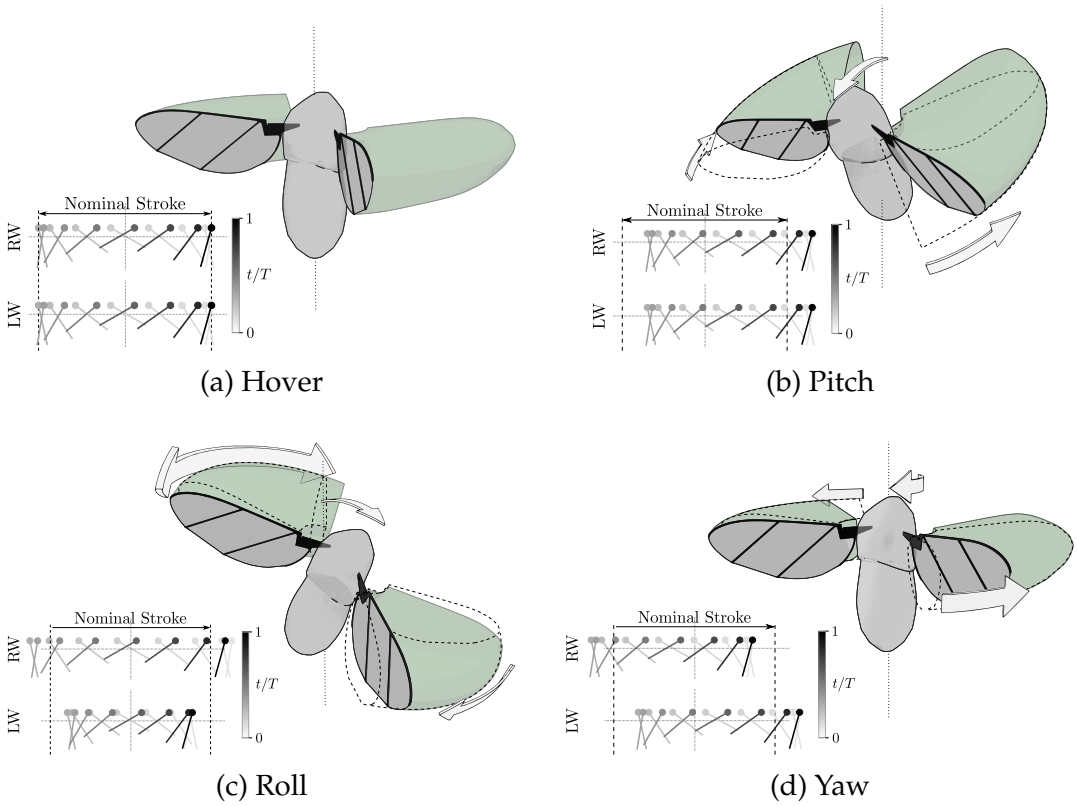


Figure 2.3: Body torque is controlled by introducing a bias into the wing stroke for pitch control, using asymmetric stroke amplitudes for roll control, and introducing opposite biases to the right and left wings (RW and LW) for yaw control. The insets show lateral views of each wing during a flapping period T for each control torque.

The method proposed here assumes the wing pitch kinematics ψ_r and ψ_l are monotonic in the corresponding mean stroke angle ($\bar{\phi}_r$ and $\bar{\phi}_l$), which can be achieved by designing an actuator which follows a stroke-plane trajectory such as (2.14). Under this assumption, a positive change to the mean stroke angle of both wings creates a positive yaw torque on the body by simultaneously creating a net positive aerodynamic force $\mathbf{F}_N(t)$ in the $\mathbf{e}_1^b$-direction on the left wing and a net negative aerodynamic force on the right wing during a complete wing stroke. It will be shown that the integral of $\mathbf{F}_N(t)$ on the right wing in the $\mathbf{e}_1^b$ direction over the total period T of a stroke is monotonic in the mean stroke angle, and a similar argument can be made for the left wing.

For the right wing, the net aerodynamic force in the $\mathbf{e}_1^b$-direction over the course of a wing stroke is

$$F_1 \triangleq \int_0^{T/2} \mathbf{F}_N(t) \cdot \mathbf{e}_1^b dt + \int_{T/2}^{T} \mathbf{F}_N(t) \cdot \mathbf{e}_1^b dt \tag{2.19}$$

It will be shown that F_1 is a strictly decreasing function of $\bar{\phi}_r$ by showing that individually, $F_{1,d} \triangleq \int_0^{T/2} \mathbf{F}_N(t) \cdot \mathbf{e}_1^b dt$ and $F_{1,u} \triangleq \int_{T/2}^{T} \mathbf{F}_N(t) \cdot \mathbf{e}_1^b dt$ are both strictly decreasing functions of $\bar{\phi}_r$. It is conventient to split the wing stroke into the downstroke $\mathcal{T}_d = [0,\ T/2)$, during which the stroke angle $\phi_r(t) \in [-\pi/2,\ \pi/2]$ is strictly decreasing and the upstroke $\mathcal{T}_u = [T/2,\ T)$, during which the stroke angle $\phi_r(t)$ is strictly increasing. For this analysis, assume hovering flight with no body velocity.

Restricting the analysis to hovering flight, it is reasonable to assume that the wing pitch $\psi_r(t) \in [-\pi/2, 0]$ for all $t \in \mathcal{T}_d$ and that $\psi_r(0) = \psi_r(T/2) = 0$. With these assumptions, the force $\mathbf{F}_N(t) \cdot \mathbf{e}_1^b$ is a strictly decreasing function of the angle of attack $\alpha_r \in [-\pi/2,\ \pi/2]$. For small stroke-plane deviations, the right wing angle of attack is approximately

$$\alpha_r \approx -\text{atan}[\cos(\psi_r)/\sin(\psi_r)] = \psi_r + \pi/2, \qquad \forall\ t \in \mathcal{T}_d \tag{2.20}$$

If ψ_r is a strictly decreasing function of $\bar{\phi}_r$, then, by extension, α_r is strictly decreasing in $\bar{\phi}_r$ for all $t \in \mathcal{T}_d$. Any positive changes to $\bar{\phi}_r$ will increase $F_{1,d}$ and any negative changes to $\bar{\phi}_r$ will decrease $F_{1,d}$ and $F_{1,d}$ is a strictly decreasing function of $\bar{\phi}_r$. Similarly for the upstroke, $F_{1,u}$ is a strictly decreasing function of $\bar{\phi}_r$. Assuming that $\psi_r(t) \in [0, \pi/2]$ for all $t \in \mathcal{T}_u$ and that $\psi_r(T/2) = \psi_r(T) = 0$. With the small stroke-plane deviation assumption, the angle of attack during the upstroke is $\alpha_r \approx \psi_r - \pi/2$ for all $t \in \mathcal{T}_u$. If ψ_r is a strictly decreasing function of $\bar{\phi}_r$, then by extension α_r is strictly decreasing in $\bar{\phi}_r$ for all $t \in \mathcal{T}_u$. Thus, $F_{1,u}$ is also a strictly decreasing function of $\bar{\phi}_r$, as is F_1.

To extend the model to designs with more than one wing pair, the equations of motion are to be augmented to account for the two additional degrees of freedom in each additional wing. Note that the model presented in this paper assumes no aerodynamic interactions between wings, which may not be valid for robots with many closely-spaced wing pairs. For each additional wing, the robot equations of motion are augmented with two scalar equations to account for the wing stroke and pitch angles, similarly to the equations developed for a single pair of wings in this section. The state vector must also be augmented to include the stroke and pitch angle for each wing, along with their derivatives. To extend the model in this paper to flapping-wing robots with fixed, rigid tails, the tail planform is modeled as a flat plane rigidly attached to the body of the robot. The equations for linear and angular momentum balance of the robot about its center of gravity developed in this section are modified to include the aerodynamic forces and moments acting on the tail, with the forces acting at the center of pressure on the tail. The aerodynamic forces, moments, and center of pressure for the tail planform are computed using the blade element method in Section 2.4. The fixed tail planform does not impart any additional degrees of freedom to the system, and so the number of equations of motion and state vector remain as presented in this section.

Equations (2.7) - (2.13), (2.15), and (2.16) constitute the equations of motion of the robot in standard form (2.1). Using the definition of the state $\mathbf{x}$ and generalized configuration coordinates $\mathbf{q}$ from (2.4), it can be seen that the equations are linear in $\ddot{\mathbf{q}}$, and can be expressed in terms of the mass matrix $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{(n/2)\times(n/2)}$, the nonlinear terms $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{(n/2)}$, and the input matrix $\mathbf{B}(t) \in \mathbb{R}^{(n/2)\times m}$ as follows:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{B}(t)\mathbf{u} \tag{2.21}$$

Solving Equation (2.21) for $\ddot{\mathbf{q}}(t)$, an expression is obtained for the state derivative $\dot{\mathbf{x}}(t)$

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{x}(t) - \begin{bmatrix} \mathbf{0} \\ \mathbf{M}(\mathbf{q})^{-1}\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}(\mathbf{q})^{-1}\mathbf{B}(t) \end{bmatrix} \mathbf{u}(t) \tag{2.22}$$

Thus, the dynamics take the affine form,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x}, t)\mathbf{u}(t) \tag{2.23}$$

where all expressions for kinematic terms in equations (2.7) - (2.13) are shown in Appendix A.1.

2.4 Blade-Element Calculations of Aerodynamic Forces and Moments

The modeling of wing aerodynamic forces and moments greatly influences the accuracy of the flight dynamics. The model of wing aerodynamic forces for flapping-wing robots is complicated due to the periodic nature of flapping and the presence of unsteady aerodynamic effects caused by the acceleration of the wing during stroke reversal. During flapping, wings periodically experience high angles of attack, stall, and high rates of rotation. Therefore, many studies have been performed to characterize the aerodynamic forces and moments during flight, including experiments on both insects and robotic wings [111] and numerical experiments [103]. Alternatively, simplified models which capture the most significant forces and moments have been proposed in [2, 32, 92, 111]. When properly tuned, these so-called *quasi-steady* models can accurately predict the net forces and moments acting on the wings throughout a wing stroke [75]

without the high computational cost associated with computation fluid dynamics (CFD) approaches. Thus, this paper uses a quasi-steady model to compute the aerodynamic forces during flight. Quasi-steady models [2, 111] obtain local wing lift and drag forces by computing forces acting on differential elements of the wing, as shown in Fig. 2.4, and integrating the forces over the entire wing surface. For simplicity, only the translational aerodynamic forces are included in the model presented here. The forces are dependent on lift and drag coefficients $C_L(\alpha)$ and $C_D(\alpha)$ that each depend on the angle of attack α_i for the ith wing. The equations for the lift and drag coefficients determined from experiments and numerical calculations are,

$$C_L(\alpha_i) = C_{L0} \sin(2\alpha_i) \tag{2.24}$$

$$C_D(\alpha_i) = C_{D0} - C_{D1} \cos(2\alpha_i) \tag{2.25}$$

as shown in [30, 111]. What follows is the computation of aerodynamic forces and moments for each wing i, where i is substituted by r or l for the right or left wing, respectively. A point Q is defined to lie in the span-wise center of each differential element on the $\mathbf{e}_2^i$ axis. The angle of attack α_i is defined as the angle between the velocity $\mathbf{v}_Q$ relative to the surrounding fluid and $\mathbf{e}_3^i$ in the plane normal to the span-wise direction of the wing $\mathbf{e}_2^i$

$$\alpha_i = -\text{atan}(\mathbf{v}_Q^T \mathbf{e}_1^i / \mathbf{v}_Q^T \mathbf{e}_3^i) \tag{2.26}$$

The differential lift force $d\mathbf{F}_L$ acting on a differential element of the wing is,

$$d\mathbf{F}_L = \left(\frac{1}{2} \rho \mathbf{v}_Q^T \mathbf{v}_Q C_L(\alpha_i) c(y) dy \right) \mathbf{e}_L \tag{2.27}$$

where ρ is the density of the surrounding fluid, $c(y)$ is the chord length of the element, and dy is the span-wise width of the element. The lift force acts in the $\mathbf{e}_L$ direction, which is normal to the relative velocity $\mathbf{v}$:

$$\mathbf{e}_L = -\cos(\alpha_i)\mathbf{e}_1^i - \sin(\alpha_i)\mathbf{e}_3^i \tag{2.28}$$

Integrating (2.27) along the wing span, the total lift force acting on the wing is

$$\mathbf{F}_L = \left(\frac{1}{2}\rho C_L(\alpha_i)\int_{y_0}^{y_f} \mathbf{v}_Q^T \mathbf{v}_Q c(y)dy\right)\mathbf{e}_L \tag{2.29}$$

The local velocity can also be written as the sum of the velocity $\mathbf{v}_A$ at the hinge point A and the velocity $\mathbf{v}_{AQ}$ of the differential element relative to the hinge, such that $\mathbf{v} = \mathbf{v}_A + \mathbf{v}_{AQ}$. Substituting this expression for local velocity into (2.29) yields,

$$\mathbf{F}_L = \frac{1}{2}\rho C_L(\alpha_i)\left(\mathbf{v}_A^T\mathbf{v}_A\int_{y_0}^{y_f} c(y)dy + \int_{y_0}^{y_f}(2\mathbf{v}_A^T\mathbf{v}_{AQ} + \mathbf{v}_{AQ}^T\mathbf{v}_{AQ})c(y)dy\right)\mathbf{e}_L \tag{2.30}$$

where both integrals in (2.30) depend on the wing geometry and $\mathbf{v}_Q$, and thus must be repeatedly evaluated based on the robot operating conditions, leading to burdensome computations. Using the notation $v_{A_1} \triangleq \mathbf{v}_A^T\mathbf{e}_1^i$ for the wing-frame components of the velocity and $\omega_{w_1} \triangleq \boldsymbol{\omega}_w^T\mathbf{e}_1^i$ for the wing-frame components of the wing angular rate, the integrals can be decomposed and simplified to,

$$\mathbf{F}_L = \frac{1}{2}\rho C_L(\alpha_i)\left(\mathbf{v}_A^T\mathbf{v}_A C_1 + 2(v_{A_3}\omega_{w_1} - v_{A_1}\omega_{w_3})C_2 + (\omega_{w_1}^2 + \omega_{w_3}^2)C_3\right)\mathbf{e}_L \tag{2.31}$$

where,

$$C_1 \triangleq \int_{y_0}^{y_f} c(y)dy, \qquad C_2 \triangleq \int_{y_0}^{y_f} yc(y)dy, \qquad C_3 \triangleq \int_{y_0}^{y_f} y^2c(y)dy \tag{2.32}$$

Similarly, the drag is computed following the same procedure,

$$\mathbf{F}_D = \frac{1}{2}\rho C_D(\alpha)\left(\mathbf{v}_A^T\mathbf{v}_A C_1 + 2(v_{A_3}\omega_{w_1} - v_{A_1}\omega_{w_3})C_2 + (\omega_{w_1}^2 + \omega_{w_3}^2)C_3\right)\mathbf{e}_D \tag{2.33}$$

where the drag force acts in the direction given by the unit vector,

$$\mathbf{e}_D = -\sin(\alpha_i)\mathbf{e}_1^i + \cos(\alpha_i)\mathbf{e}_3^i \tag{2.34}$$

which represents the direction of the velocity of the surrounding fluid relative to the wing. The total aerodynamic force acting on the wing is simply the summation of the lift and drag, $\mathbf{F}_i = \mathbf{F}_L + \mathbf{F}_D$.

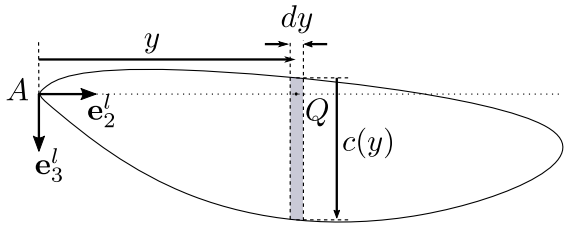


Figure 2.4: A 2D view of the left wing showing a single differential blade element used for the calculation of aerodynamic forces on the wing.

The lift and drag forces both act at the wing's center of pressure. The position vector of the left wing center of pressure relative to the wing hinge as used in (2.12) and (2.13) is,

$$\mathbf{r}_{AP_i} = Y_{cp}\mathbf{e}_2^i + Z_{cp}(\alpha_i)\mathbf{e}_3^i \tag{2.35}$$

where Y_{cp} and Z_{cp} are the span-wise and chord-wise locations of the center of pressure, respectively. Previous studies in fruit flies [31] and robotic wings [114] have found that Y_{cp} remains relatively constant with angle of attack and that Z_{cp} varies with the angle of attack according to the relationship,

$$Z_{cp}(\alpha_i) = \int_{y_0}^{y_f} c(y)\left(\frac{0.82|\alpha_i|}{\pi} + 0.05\right)dy \tag{2.36}$$

found empirically [31].

Finally, local aerodynamic forces also causes rotational damping acting about the span-wise direction of the wing ($\mathbf{e}_2^i$) [2, 31, 114]. The rotational damping moment $\mathbf{M}_{rd}$ is found by integrating the local drag on a rectangular differential element of the wing in both the chord-wise and span-wise directions

$$\mathbf{M}_{rd,i} = \left(\frac{1}{2}\rho C_D(\pi/2)\int_{z_0}^{z_1}\int_0^R |\mathbf{v}_Q^T\mathbf{e}_1^i|\mathbf{v}_Q^T\mathbf{e}_1^i z\,dr\,dz\right)\mathbf{e}_2^i \tag{2.37}$$

Together, the equations (2.31), (2.33), (2.35), and (2.37) describe the aerodynamic forces and moments acting on the wings during flight and the locations of the center of pressure. They are used in the equations of motion (2.23) to determine

the response of the robot to a given set of initial conditions and sequence of control inputs.

2.5 Steady Maneuvers

The model developed in Sections 2.3-2.4 is used here to analyze the open loop dynamics of insect-scale flapping-wing flight, including the range of conditions in which steady flight is possible, the dominant linear modes of flight, and its open-loop stability. The analysis is completed at M set points $\mathcal{P} = \{(\mathbf{x}^*(t), \mathbf{u}^*)_i : i = 1, \ldots, M\}$, each comprising a constant control input $\mathbf{u}^*$ and a periodic trajectory $\mathbf{x}^*(t)$, which is a solution to (2.1) that also satisfies the constraints defined for the maneuver. The definition of the state which is used for the analysis presented in this section comprises the Euler angles and angular rates for the right and left wings, the inertial coordinates of the body x, y, and z, the body Euler angles ϕ, θ, and ψ, the components of the robot velocity in the body frame u, v, and w, and the components of the robot angular rate in the body frame p, q, and r. The complete state of the flapping-wing robot is

$$\mathbf{x} = \begin{bmatrix} \phi_r & \psi_r & \phi_l & \psi_l & \dot{\phi}_r & \dot{\psi}_r & \dot{\phi}_l & \dot{\psi}_l & x & y & z & \phi & \theta & \psi & u & v & w & p & q & r \end{bmatrix}^T \tag{2.38}$$

Each set point falls within the steady flight envelope. This paper develops a method for computing the steady flight envelope based on the physical parameters of the system and for computing the robot set points within the flight envelope. The dynamic stability and dominant modes of motion are analyzed in Section 2.7. This analysis of linear modes provides a foundation for stabilizing controller design and an understanding of the range of flight conditions which

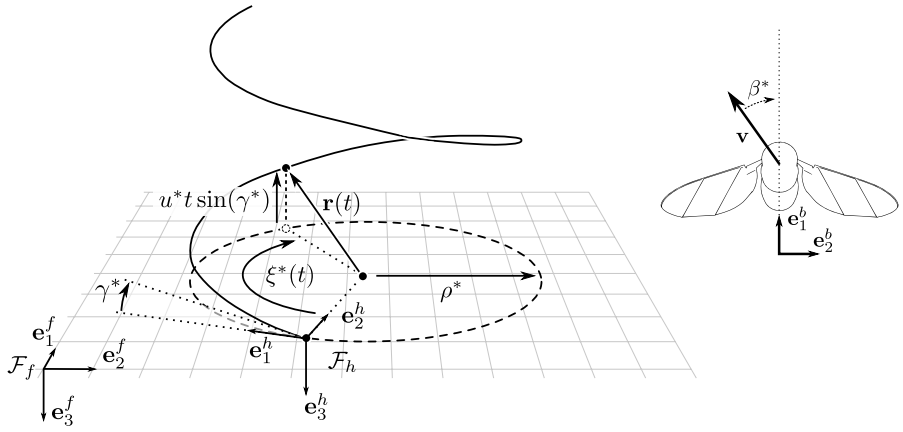


Figure 2.5: The trajectory followed by a robot during a steady coordinated turn.

require an active controller for stable flight.

Earlier work has examined the unstable modes of flapping-wing flight in hovering, vertical, and forward flight [14, 88, 116, 124], but here a more general analysis is presented which includes a range of climb angles and turning flight. The steady coordinated turn is the most general steady maneuver for the flapping-wing model, where a steady maneuver is defined as a solution to (2.1) for which the body-frame components of $\mathbf{v}$ and $\boldsymbol{\omega}$ have a minimum period of T. After defining the trajectory constraints for the steady coordinated turn, the important special cases of longitudinal flight, lateral flight, and hovering are defined. In longitudinal flight, the robot motion is confined to the robot's sagittal plane, defined as the plane which passes through G and is normal to $\mathbf{e}_2^b$. In lateral flight, motion is confined to the robot's coronal plane, defined as the plane which passes through G and is normal to $\mathbf{e}_1^b$. Hovering flight is the special case where $\mathbf{x}(t+T) - \mathbf{x}(t) = \mathbf{0}$ for all t.

The command vector $\mathbf{y}^*$ is used to define the parameters of a coordinated turn, and comprises the commanded speed $u^* \in \mathbb{R}^+$, the commanded climb angle $\gamma^* \in \mathbb{R}$, a commanded sideslip angle $\beta^* \in \mathbb{R}$, and the commanded turn rate

$\dot{\xi}^* \in \mathbb{R}$. The commanded speed u^* is the non-negative speed of the robot. The commanded climb angle γ^* is the angle measured from the ground plane to the body velocity vector $\mathbf{v}$. The sideslip angle $\beta^* \in \mathbb{R}$ is the angle measured from the body velocity vector $\mathbf{v}$ to the sagittal plane of the robot body. The commanded turn rate $\dot{\xi}^*$ is the rotation rate of $\mathcal{F}_b$ about $\mathbf{e}_3^f$, and is equivalent to the yaw rate, with the command vector written as

$$\mathbf{y}^* = \begin{bmatrix} u^* & \gamma^* & \dot{\xi}^* & \beta^* \end{bmatrix}^T \tag{2.39}$$

When the turn rate $\dot{\xi}^*$ is non-zero, it can be used in conjunction with the commanded speed to define a turning radius ρ^*

$$\rho^* = \frac{u^*}{\dot{\xi}^*} \tag{2.40}$$

The general case of a steady coordinated turn is shown in Fig. 2.5, including important reference frames and command variables.

2.5.1 Flapping-wing Robot Set Points

To determine the feasibility of these maneuvers, a direct transcription method is used to convert the ODE root finding problem into a set of algebraic equations to be solved numerically. This method was more successful in practice than alternative methods, such as shooting methods which solve the root finding problem using only the end point constraint. The robot's trajectory $\mathbf{x}(t)$ is discretized in time on the interval $[0,\ T]$ using a step size of Δt, such that $\mathbf{x}_k = \mathbf{x}(k\Delta t)$ where $k \in \mathbb{N}$. Using the Hermite Simpson rule, the trajectory is constrained to satisfy the dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$ at each point $\mathbf{x}_k$ as well as at the midpoints between the discretized points, denoted by $\bar{\mathbf{x}}_{k+1} = \mathbf{x}(k\Delta t + \Delta t/2)$. The constraints for each

point can be written as

$$\mathbf{0} = \bar{\mathbf{x}}_{k+1} - \frac{1}{2}(\mathbf{x}_{k+1} + \mathbf{x}_k) - \frac{\Delta t}{8}(\mathbf{f}_k - \mathbf{f}_{k+1}) \tag{2.41}$$

$$\mathbf{0} = \mathbf{x}_{k+1} - \mathbf{x}_k - \frac{\Delta t}{6}(\mathbf{f}_{k+1} + 4\bar{\mathbf{f}}_{k+1} + \mathbf{f}_k) \tag{2.42}$$

The dynamic constraints $\mathbf{c}_d$ for the entire interval $[0, T]$ can be expressed as a set of N_d linear equations written as,

$$\mathbf{c}_d(\mathbf{x}, \mathbf{u}) \triangleq \mathbf{G}\mathbf{x} + \mathbf{H}\mathbf{d}(\mathbf{x}, \mathbf{u}) \tag{2.43}$$

where,

$$\mathbf{d} \triangleq \Delta t \begin{bmatrix} \mathbf{f}_1^T & \bar{\mathbf{f}}_2^T & \mathbf{f}_2^T & \mathbf{f}_3^T \dots \mathbf{f}_M^T \end{bmatrix}^T \tag{2.44}$$

and expressions for the constant matrices $\mathbf{G} \in \mathbb{R}^{N_d \times N_d}$, $\mathbf{H} \in \mathbb{R}^{N_d \times N_d}$ are given in Appendix A.3. The dynamic constraints given in (2.43) constrain the state at every discretized point in time t_k except for the initial conditions at $t = 0$. Maneuver constraints are also used, and are only satisfied when the state at the end of a period is equal to the desired state $\mathbf{x}^*$:

$$\mathbf{c}_m(\mathbf{x}) \triangleq \mathbf{x}(T) - \mathbf{x}^*(T) \tag{2.45}$$

The constraints for each maneuver will be specified in the following sections. Once the constraints are defined, a numeric solver can be used to find the roots of the constraint function

$$\mathbf{c}(\mathbf{x}, \mathbf{u}) \triangleq \begin{bmatrix} \mathbf{c}_d^T(\mathbf{x}, \mathbf{u}) & \mathbf{c}_m^T(\mathbf{x}, \mathbf{u}) \end{bmatrix}^T \tag{2.46}$$

2.5.2 Maneuver Definitions

Steady maneuvers are defined by choosing the appropriate definition of the desired state $\mathbf{x}^*$ for each maneuver. The desired state is first defined for a steady

coordinated turn, followed by the definition of the desired state for other steady maneuvers. The wing state $\mathbf{x}_w \in \mathbb{R}^8$, comprising the Euler angles and angular rates of both wings, is constrained to be periodic with the flapping period T for every steady maneuver. Otherwise, any deviation from a periodic wing state will cause a change in aerodynamic forces and torques, preventing the conditions in (2.45) from being satisfied. In a steady coordinated turn, the velocity $\mathbf{v}$ and body angular rate $\boldsymbol{\omega}_b$ vectors are constrained to rotate about $\mathbf{e}_3^f$ at the desired turn rate $\dot{\xi}^*$. This rotation is computed using the rotation matrix $\mathbf{R}^* \triangleq \mathbf{R}(\mathbf{e}_3^f, T\dot{\xi}^*)$, where the general expression for computing the terms of the rotation matrix is given in Appendix A.2. Constraints on the body Euler angles and the body position vector follow from the constraints on body angular rate and velocity, and are written using an intermediate frame $\mathcal{F}_h$ which coincides with the inertial frame $\mathcal{F}_f$ rotated about $\mathbf{e}_3^b$ by the yaw angle $\phi(0)$ at the beginning of the maneuver. The desired state for a steady coordinated turn is,

$$\mathbf{x}^*(T) = \begin{bmatrix} [\mathbf{x}_w(0)]^T & [\mathbf{R}^*\boldsymbol{\omega}_b(0)]^T & [\mathbf{R}^*\mathbf{v}(0)]^T & [\boldsymbol{\Theta}^*(T)]^T & [\mathbf{r}^*(T)]^T \end{bmatrix}^T \tag{2.47}$$

where the desired orientation is $\boldsymbol{\Theta}^*(T) = \boldsymbol{\Theta}(0) + [T\dot{\xi}^* \; 0 \; 0]^T$, the desired position is,

$$\mathbf{r}^*(T) = \mathbf{r}(0) - u^* T \sin(\gamma^*)\mathbf{e}_3^f + \rho^* \cos(\gamma^*)(\mathbf{e}_2^h - \mathbf{R}^*\mathbf{e}_2^h) \tag{2.48}$$

and where $\mathbf{e}_2^h = \mathbf{R}[\mathbf{e}_3^f, \phi(0)]\mathbf{e}_2^f$.

Steady longitudinal flight is a special case of a steady coordinated turn corresponding to a nonzero commanded speed u^*, zero turn rate $\dot{\xi}^*$, and zero sideslip angle β^*. In other words, steady longitudinal flight is a steady coordinated turn where the command input satisfies the conditions

$$u^* > 0, \qquad \dot{\xi}^* = 0, \qquad \beta^* = 0 \tag{2.49}$$

The desired state for a steady coordinated turn given in (2.47) must be rewritten for longitudinal flight, because the desired position (2.48) is written in terms of the turn radius, which is $\rho^* = \infty$ for longitudinal flight. The desired position is decomposed into its $\mathbf{e}_1^h$ and $\mathbf{e}_2^h$ components

$$\mathbf{r}^*(T) = \mathbf{r}(0) - u^* T \sin(\gamma^*)\mathbf{e}_3^f + \rho^* \cos(\gamma^*)[\mathbf{e}_2^h - \mathbf{e}_2^h \cos(\dot{\xi}^* T) + \mathbf{e}_1^h \sin(\dot{\xi}^* T)] \tag{2.50}$$

The desired position can now be reformulated in terms of the commanded speed u^* by subtracting $\mathbf{r}(0)$ from both sides and taking the inner product of the result with $\mathbf{e}_1^h$ to obtain

$$[\mathbf{r}^*(T) - \mathbf{r}(0)] \cdot \mathbf{e}_1^h = \rho^* \cos(\gamma^*) \sin(\dot{\xi} T) \tag{2.51}$$

The Taylor Series expansion of $\sin(\dot{\xi}T)$ is then used along with (2.40) to obtain

$$[\mathbf{r}^*(T) - \mathbf{r}(0)] \cdot \mathbf{e}_1^h = u^* T \cos(\gamma^*)(1 - \frac{(\dot{\xi}T)^2}{3!} + \frac{(\dot{\xi}T)^4}{5!} + \ldots) \tag{2.52}$$

Applying the longitudinal flight constraint $\dot{\xi} = 0$ yields the final expression for the desired position in the $\mathbf{e}_1^h$ direction:

$$[\mathbf{r}^*(T) - \mathbf{r}(0)] \cdot \mathbf{e}_1^h = u^* T \cos(\gamma^*) \tag{2.53}$$

Taken together with the $\mathbf{e}_3^f$ component of the desired position from (2.48) gives the desired position for longitudinal flight:

$$\mathbf{r}^*(T) = \mathbf{r}(0) - u^* T \sin(\gamma^*)\mathbf{e}_3^f + u^* T \cos(\gamma^*)\mathbf{e}_1^h \tag{2.54}$$

Together with the remaining constraints from (2.47) and the longitudinal flight constraint $\dot{\xi}^* = 0$, the desired state for steady longitudinal flight is

$$\mathbf{x}^*(T) = \begin{bmatrix} \mathbf{x}_w^T(0) & \boldsymbol{\omega}_b^T(0) & \mathbf{v}^T(0) & \boldsymbol{\Theta}^T(0) & [\mathbf{r}^*(T)]^T \end{bmatrix}^T \tag{2.55}$$

Steady lateral flight is similar to steady longitudinal flight, but with a sideslip angle $\beta = \pi/2$. Thus, the command input for steady lateral flight must

satisfy

$$u^* > 0, \qquad \dot{\xi}^* = 0, \qquad \beta^* = \frac{\pi}{2} \tag{2.56}$$

The desired state for lateral flight is identical to the longitudinal case, except that the velocity vector must be pointed in the $\mathbf{e}_2^h$ direction:

$$\mathbf{r}^*(T) = \mathbf{r}(0) - u^* T \sin(\gamma^*)\mathbf{e}_3^f + u^* T \cos(\gamma^*)\mathbf{e}_2^h \tag{2.57}$$

Hovering flight is a special case of longitudinal flight where $u^* = 0$. In hovering, the state is perfectly periodic in T and the desired state is thus

$$\mathbf{x}^*(T) = \mathbf{x}(0) \tag{2.58}$$

2.6 Flight Envelopes

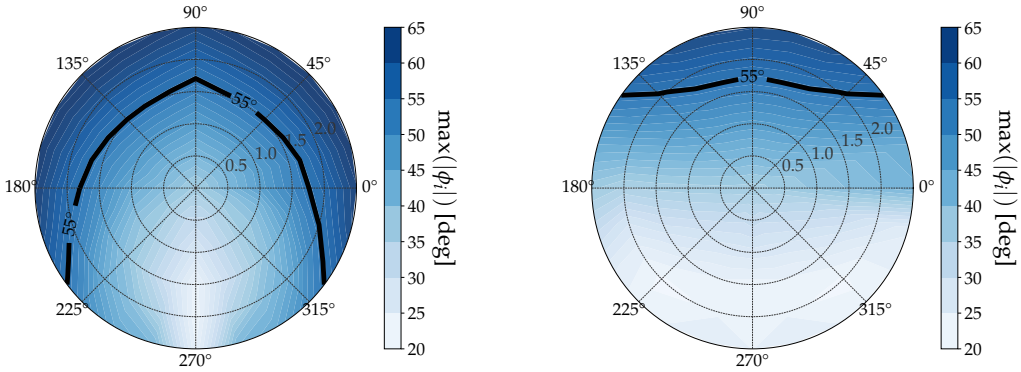
The physical parameters of the robot limit the range of static and dynamic conditions within which steady flight is possible. This range of conditions, called the *steady flight envelope*, is used in this paper to provide reasonable boundaries on the set points to be analyzed for stability and other flight characteristics. Furthermore, by analyzing the sensitivity of the flight envelope boundaries to the physical parameters of a robot, the flight envelope can be a useful tool to aid in the design of new robots to ensure that they are capable of achieving their performance requirements. In designing autonomous controllers for flapping-wing robots, the steady flight envelope defines the range of conditions within which a suitable flight controller should be capable of maintaining the robot at a set point. Once a robot is built and operational, the flight envelope is used to determine what conditions the robot can safely operate in without damaging the actuators.

It is assumed that the primary limiting factors for most flapping wing robots are the joint limits affecting the maximum wing stroke angle required to achieve a desired set point. To determine boundaries of the steady flight envelope, it is necessary to evaluate the wing trajectories required to achieve steady flight at a given set point. For the desired set point, the equations of motion (2.59) are solved to determine the maximum stroke angle, $\max(|\phi_i|)$, required for steady flight for each wing i. The maximum stroke angle required for longitudinal and lateral flight is plotted as a function of the commanded speed u^* and climb angle γ^* in Fig. 2.6. The radial distance from the center of the plot indicates the commanded speed u^* and the angular coordinate indicates the commanded climb angle γ^* as shown in Fig. 2.7. A contour on each plot is highlighted corresponding to the assumed maximum allowable stroke angle of each wing.

Figure 2.6a shows that the imposed joint limits constrain the maximum flight speed in ascending flight to approximately 1.5m/s, with larger flight speeds possible in descending flight. For lateral flight, the maximum stroke angle primarily limits the maximum vertical speed of flight, and is only a weak function of lateral velocity. This is noticeably different from the longitudinal case, in which the non-zero mean pitch angle required for forward flight limits longitudinal flight speed more significantly.

2.7 Stability and Dominant Linear Modes

With the exception of hovering flight, all steady maneuver set points are *quasistatic* equilibria: the body-frame velocities, body-frame angular rates, and wing state variables are constant, but other elements of the state vector are al-



(a) Longitudinal flight envelope

(b) Lateral flight envelope

Figure 2.6: The maximum stroke amplitude required to achieve a desired set point in both longitudinal and lateral flight

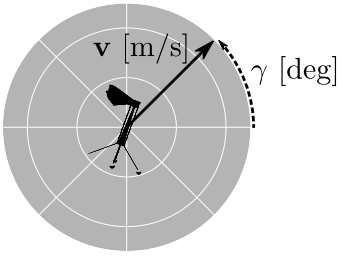


Figure 2.7: A key for reading the flight envelope plots, where the speed is the radial coordinate and the climb angle is the angular coordinate.

lowed to change over time. Using the state vector $\mathbf{x}$ defined in (2.38), which expresses the robot velocity and angular rate in the body frame, only four of the state variables will have a non-zero rate of change during a general coordinated turn. These so-called *ignorable* state variables comprise the yaw angle ϕ and the body position coordinates x, y, and z. The remaining *non-ignorable* state variables are independent of the body position or yaw angle, so any solution to (2.1) found at one position and yaw angle is equally valid for any other position and yaw angle.

Due to the periodic nature of flapping-wing flight, stability is analyzed by first discretizing (2.23) by sampling at an interval equal to the flapping period

T. Adopting the notation $t_k \triangleq kT$ and $\mathbf{x}_k \triangleq \mathbf{x}(t_k)$ for any non-negative integer k, the sampled system is,

$$\mathbf{x}_{k+1} = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}) + \mathbf{x}_k \tag{2.59}$$

where

$$\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}) \triangleq \int_{t_k}^{t_{k+1}} \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t]dt \tag{2.60}$$

An equilibrium point consists of a point in state space $\mathbf{x}^*$ and a corresponding constant control input $\mathbf{u}^*$. An equilibrium point must satisfy

$$\mathbf{h}(\mathbf{x}^*, \mathbf{u}^*, \mathbf{p}) = \mathbf{0} \tag{2.61}$$

The Jacobian matrix $\nabla\mathbf{h}|_{\mathbf{x}^*, \mathbf{u}^*}$ is used to analyze the linear modes of the system and its stability at a given equilibrium point. The sampled equations of motion (2.59) are approximated by the linear system

$$\mathbf{x}_{k+1} = \nabla\mathbf{h}|_{\mathbf{x}^*, \mathbf{u}^*}\mathbf{x}_k \tag{2.62}$$

Eigendecomposition is used to express the solution to (2.62) in terms of the eigenvalues $\lambda_i = \sigma_i \pm i\omega_i$ and eigenvectors $\mathbf{v}_i = \mathbf{u}_i \pm i\mathbf{w}_i$ of the Jacobian,

$$\mathbf{x}_k = \sum_{i=1}^{n} \kappa_i \lambda_i^k \mathbf{v}_i \tag{2.63}$$

where $\kappa_i \in \mathbb{R}$ are coefficients that depend on the initial conditions $\mathbf{x}_0$.

The linearized equations of motion are analyzed here using parameters from the flapping-wing robot in [68], including a flapping frequency of $T = 120$Hz. Throughout the flight envelope, there are two dominant linear modes which may be either stable or unstable depending on the regime of flight:

1. A generally oscillatory motion dominated by pitching and forward velocity terms called the *longitudinal mode*.

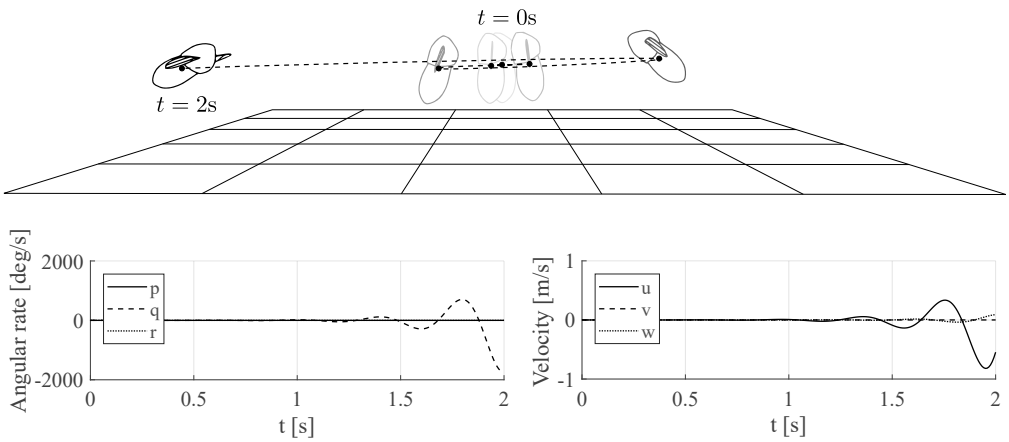


Figure 2.8: The unstable longitudinal mode in hovering flight depicted as a trajectory along with the body-frame components of the velocity and angular rate.

2. A generally oscillatory motion dominated by lateral velocity and body roll oscillations called the *lateral mode*, which also couples with yaw and pitch in forward flight.

All other modes damp out relatively quickly and are stable throughout the flight envelope. Thus, the discussion in this paper will focus exclusively on the longitudinal and lateral modes outlined above. Figures 2.8 and 2.9 illustrate the unstable motions associated with these modes at the hovering equilibrium point. The trajectories plotted in these figures are solutions to (2.62) given initial conditions that lie in the spaces of the longitudinal and lateral eigenvectors ($\mathbf{v}_o$ and $\mathbf{v}_a$) respectively. As plotted in these figures, the longitudinal mode is generally characterized by oscillations in pitch rate q and forward velocity u. The lateral mode near hovering is characterized by oscillations in lateral body velocity v and roll rate p [39]. As forward flight speed increases, the lateral mode also couples with pitch rate q, and yaw rate r.

To visualize changes in the eigenvalues for the longitudinal mode throughout the flight envelope, the longitudinal damping ratio $\zeta_o = -\cos[\angle \ln(\lambda_o)]$

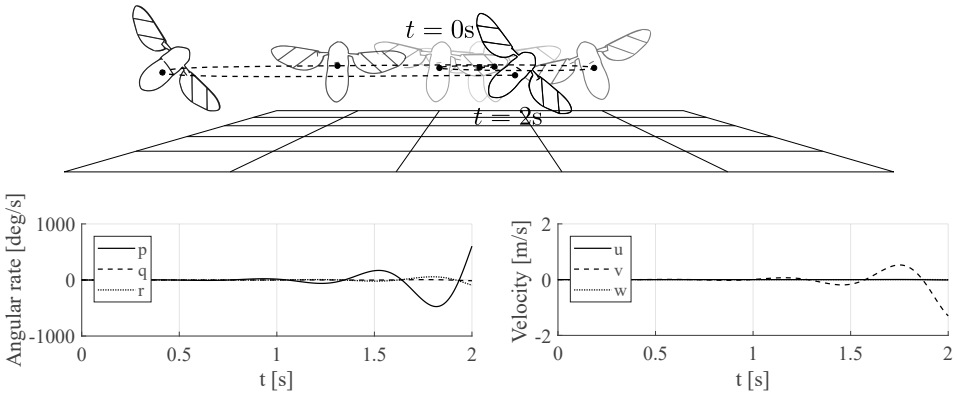
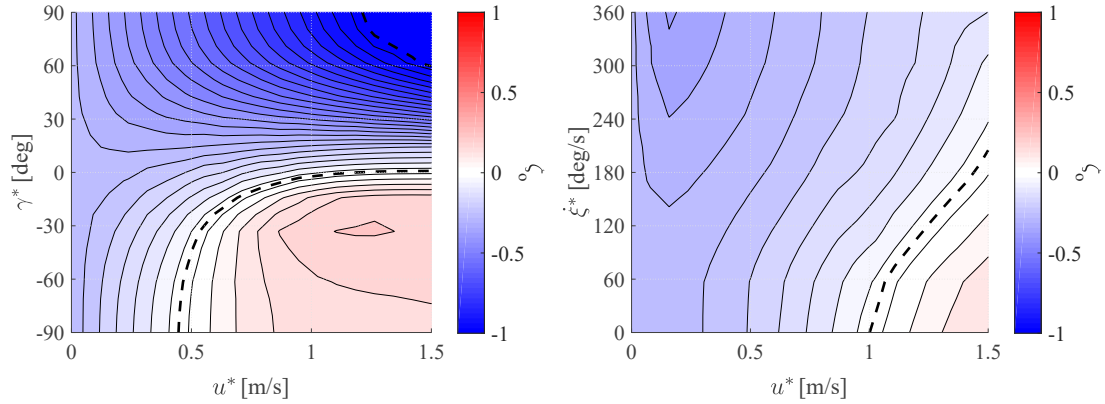


Figure 2.9: The unstable lateral mode in hovering flight depicted as a trajectory along with the body-frame components of the velocity and angular rate.

(where $\angle$ denotes the angle in the complex plane, e.g. $\angle\lambda_i = \text{atan}(\omega_i/\sigma_i)$) is plotted for longitudinal flight as a function of flight speed u^* and climb angle γ^* in Fig. 2.10a (with $\dot{\xi}^* = \beta^* = 0$) and for a level coordinated turn as a function of u^* and turn rate $\dot{\xi}^*$ in Fig. 2.10b (with $\gamma^* = \beta^* = 0$). Near hovering flight and in most regimes of low-speed flight and climbing flight, the longitudinal mode is unstable, having two complex conjugate eigenvalues each with a magnitude greater than unity. As indicated by the red region in Fig. 2.10a, the longitudinal mode becomes a stable, underdamped mode at moderate flight speeds when $\gamma^* \leq 0$. A third regime exists for nearly vertical high-speed flight, outlined in the upper-right hand corner of Fig. 2.10a approximately where $\gamma^* > 60^\circ$ and $u^* > 1.3$m/s. In this regime, the longitudinal mode is characterized by an unstable exponentially divergent motion and has two distinct purely real eigenvalues. For level turning flight, the longitudinal mode is unstable except for low turn rates at high speed ($\dot{\xi}^* < 120^\circ$/s and $u^* > 1$m/s), where it is characterized by underdamped stable oscillations, plotted as the red area in Fig. 2.10b. The transition from unstable to stable flight occurs at higher flight speeds as $\dot{\xi}^*$ increases.

Similarly, the eigenvalues of the lateral mode throughout the flight envelope



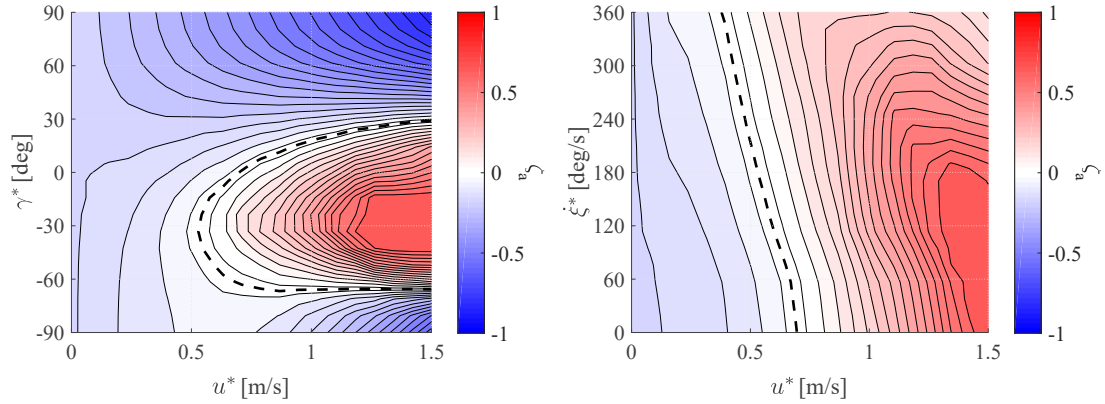
(a) In longitudinal flight

(b) In level turning flight

Figure 2.10: The damping ratio of the longitudinal mode in longitudinal flight level turning flight, with unstable regions highlighted in blue and stable regions in red.

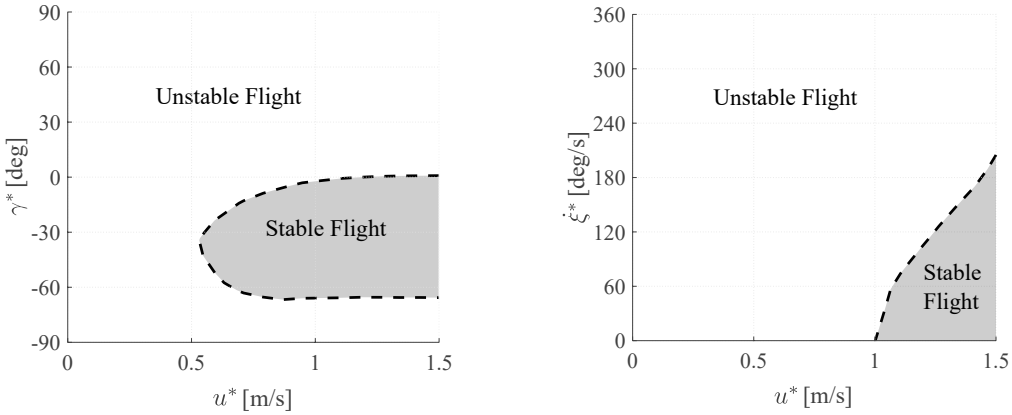
are visualized by plotting the lateral damping ratio $\zeta_a = -\cos[\angle \ln(\lambda_a)]$ in Fig. 2.11. Stability of the lateral mode in the longitudinal flight regime is divided into two regions, plotted in Fig. 2.11a. Throughout the majority of the flight envelope, the lateral mode is unstable, having two complex conjugate eigenvalues with magnitude greater than unity. Above moderate forward flight speeds, the lateral mode is characterized by stable underdamped solutions, highlighted by the red region in Fig. 2.11a. The stability boundary for the lateral mode in level turning flight, plotted in Fig. 2.11b, trends in the opposite direction of the longitudinal mode stability boundary, with the transition from unstable to stable flight occurring at lower flight speeds as $\dot{\xi}^*$ increases.

An important special case of longitudinal flight is level forward flight, where $u^* > 0$ and $\gamma^* = 0$. In level forward flight, the lateral mode is stable above flight speeds of $u^* \approx 0.7$ m/s, and the longitudinal mode is stable above flight speeds of $u^* \approx 1.0$ m/s. At forward flight speeds greater than 1.0 m/s, all linear modes of forward level flight are stable. The regions of stability for all regimes of longitudinal flight and level turning flight are found by taking the union of



(a) In longitudinal flight (b) In level turning flight

Figure 2.11: The damping ratio of the lateral mode in longitudinal flight and level turning flight, with unstable regions highlighted in blue and stable regions in red.



(a) Longitudinal flight (b) Level turning flight

Figure 2.12: Regions of stable and unstable flight in steady longitudinal flight and steady level turning flight, where the stable regions are simply the union of the stable regions from Figs. 2.10 and 2.11.

the stable regions for both the longitudinal and lateral modes from Figs. 2.10 and 2.11. The result of this union is plotted in Fig. 2.12.

In stable flight regimes, the longitudinal and lateral modes exhibit different motions compared with their motions near hovering (Figs. 2.8 and 2.9). Figures 2.13 and 2.14 illustrate the stable motions associated with the longitudinal and lateral modes for forward level flight with $u^* = 1.5\text{m/s}$. Compared with the

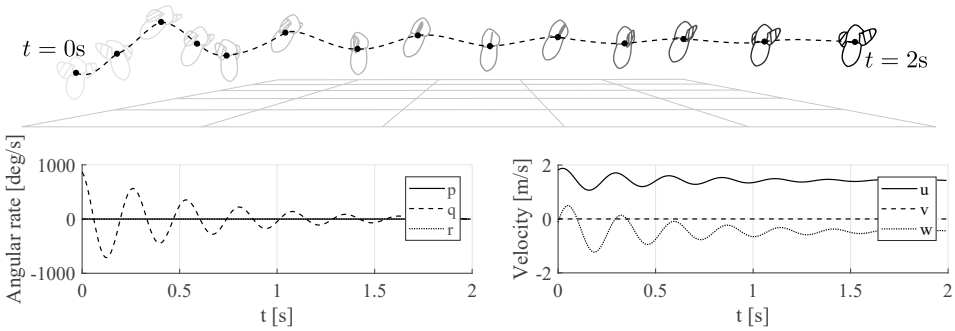


Figure 2.13: The longitudinal mode in forward level flight at 1.5m/s depicted as a trajectory along with the body-frame components of the velocity and angular rate.

mode shape at hovering, the longitudinal mode in forward flight couples more strongly with the $\mathbf{e}_3^b$-component of body velocity, w, such that the magnitude of oscillations in w is roughly twice the magnitude of the oscillations in u. For the flight speed plotted (u^* = 1.5m/s), the longitudinal mode is lightly damped, requiring on the order of hundreds of wingbeats to damp out to steady state. In forward flight, the lateral mode resembles the dutch roll mode present in fixed-wing aircraft [100], exhibiting coupled rolling and yawing motions that result in passively stable banked turns. In forward flight, the oscillations in p and r in the lateral mode are approximately 180° out of phase, whereas the same oscillations are nearly in phase with one another near hovering flight (see Fig. 2.9). The lateral mode in forward flight is more highly damped than the longitudinal mode and settles to steady-state in under 100 wingbeats.

2.8 Experimental Results and Validation

Validating the model against experimental data is complicated by the unstable nature of flapping-wing flight near hovering and its sensitivity to initial conditions and disturbances. One way to validate the model is to examine the dom-

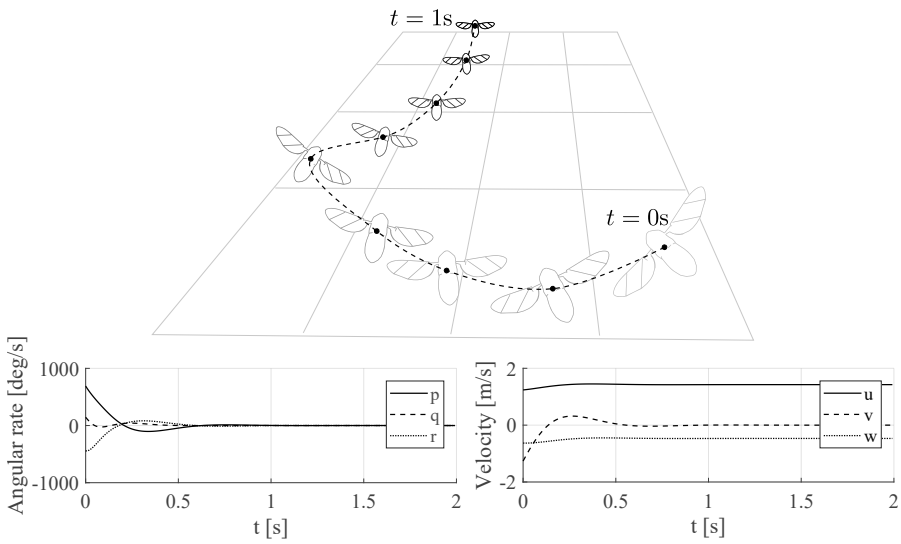
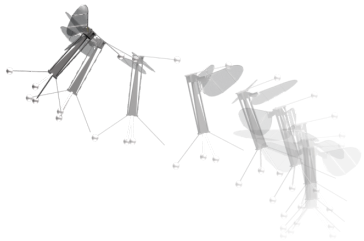


Figure 2.14: The lateral mode in forward level flight at 1.5m/s depicted as a trajectory along with the body-frame components of the velocity and angular rate.

inant linear modes present near hovering in experimental data and to compare them with the modes predicted by the model. Several open loop flight trials were conducted using a flapping-wing robot operating near hover. For each test, the robot began with zero body velocity and was oriented vertically. The constant control signal sent to the robot during each trial corresponds with the signal to approximate hovering flight. Fifteen tests were conducted in this manner, with a sequence of images from two trials shown in Figs. 2.15a and 2.16a. Figures 2.15b and 2.16b show a sequence of images rendered from the model using the same initial conditions and parameters from the experimental trial. The system is sensitive to initial conditions and disturbances caused by the thin wire tether used to send power and control signals to the robot, which resulted in significant variations between successive tests. Although it is impossible to exactly recreate the set of initial conditions and disturbances present in each experimental trial, the model shows strong qualitative similarity to the trajectories from the open loop flight experiments.

(a)



(b)

Figure 2.15: A visual comparison of the open loop longitudinal instability in the physical robot (a) and the model (b), showing qualitatively similar behavior.

(a)

(b)

Figure 2.16: A second comparison of the open loop instability in the physical robot (a) and the model (b), showing qualitatively similar behavior. Video available at `https://youtu.be/qj-VbJiVVk4`

Dynamic Mode Decomposition [64] is used to extract the dominant linear modes of the robot dynamics from the collected data. Assuming the dynamics are approximately linear near hovering, the data points at one time step are related to the data points from the previous time step by the dynamics matrix $\mathbf{A}$ through the relationship,

$$\mathbf{X}_2 = \mathbf{A}\mathbf{X}_1 \tag{2.64}$$

where the data collected from the fifteen experimental trials consists of N data points $\mathbf{x}_i \in \mathbb{R}^n$ that are arranged into the matrices

$$\mathbf{X}_1 = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{N-1} \end{bmatrix}, \qquad \mathbf{X}_2 = \begin{bmatrix} \mathbf{x}_2 & \mathbf{x}_3 & \dots & \mathbf{x}_N \end{bmatrix} \tag{2.65}$$

The eigenvalues and eigenvectors of $\mathbf{A}$ correspond to the linear modes of the physical robot near hovering flight, and can be compared to the linear modes of the model obtained from the Jacobian matrix $\nabla \mathbf{h}|_{\mathbf{x}^*,\mathbf{u}^*}$ to evaluate the similarity

of the model to the physical robot. The matrix $\mathbf{A}$ is computed from the collected data using Singular Value Decomposition (SVD),

$$\mathbf{A} = \mathbf{X}_2 \mathbf{V} \mathbf{\Sigma}^{-1} \mathbf{U}^* \tag{2.66}$$

where $*$ denotes the conjugate transpose, $\mathbf{U} \in \mathbb{C}^{n \times r}$, $\mathbf{\Sigma} \in \mathbb{C}^{r \times r}$, $\mathbf{V} \in \mathbb{C}^{N \times r}$, and $r \leq n$ is the rank of the SVD approximation to $\mathbf{X}_1$. Noise in the data is suppressed by restricting the rank r so that the decomposition includes only the dominant singular modes.

Two of the eigenvectors of the $\mathbf{A}$ matrix computed from the experimental data correspond to the longitudinal and lateral modes found in the model. The longitudinal mode is comprised predominantly of terms corresponding to the state variables ψ, v_x, and $d\psi/dt$. Let $\mathbf{v}_o \in \mathbb{R}^3$ and $\tilde{\mathbf{v}}_o \in \mathbb{R}^3$ be the vectors consisting of the entries of the longitudinal mode eigenvectors corresponding to these terms from the model and the data, respectively. The real and imaginary parts of these vectors span planes which pass through the origin and have the normal vectors given by,

$$\mathbf{n}_o = \text{Re}(\mathbf{v}_o) \times \text{Im}(\mathbf{v}_o), \qquad \tilde{\mathbf{n}}_o = \text{Re}(\tilde{\mathbf{v}}_o) \times \text{Im}(\tilde{\mathbf{v}}_o) \tag{2.67}$$

where $\text{Re}(\cdot)$ and $\text{Im}(\cdot)$ denote the real and imaginary parts of the vector, respectively. The experimental trajectories lie near the plane normal to $\tilde{\mathbf{n}}_o$, as shown in Fig. 2.18b. This occurs because the longitudinal mode dominates the evolution of ψ, v_x, and $d\psi/dt$ near hovering, which implies that the trajectories of these state variables must remain near the space spanned by the real and imaginary parts of the longitudinal mode eigenvector. Similarly, the lateral mode contains dominant terms corresponding to the state variables θ, v_y, and $d\theta/dt$, implying that the trajectories of these variables must remain near the space spanned by the real and imaginary parts of the lateral mode eigenvector. Let $\mathbf{v}_a \in \mathbb{R}^3$ and

$\tilde{\mathbf{v}}_a \in \mathbb{R}^3$ be the vectors consisting of the entries of the lateral mode eigenvectors corresponding to these terms from the model and the data, respectively. The real and imaginary parts of these vectors span planes which pass through the origin and have normal vectors given by

$$\mathbf{n}_a = \mathrm{Re}(\mathbf{v}_a) \times \mathrm{Im}(\mathbf{v}_a), \qquad \tilde{\mathbf{n}}_a = \mathrm{Re}(\tilde{\mathbf{v}}_a) \times \mathrm{Im}(\tilde{\mathbf{v}}_a) \tag{2.68}$$

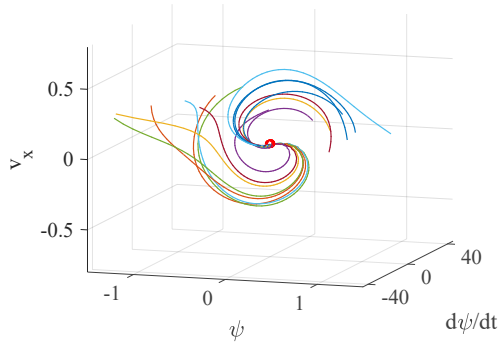
The trajectories from the experimental data are plotted in the longitudinal and lateral subspaces in Fig. 2.17. The planes normal to $\tilde{\mathbf{n}}_o$ and $\tilde{\mathbf{n}}_a$ are plotted in Figs. 2.18a and 2.18b alongside the trajectories, where it is clear that the trajectories lie near to the planes defined by these modes.

To assess how well the dominant modes of the model match the dominant linear modes of the physical robot flight dynamics near hovering, the orientation of the longitudinal and lateral mode planes is compared between the model and the data by computing the dot product of their normal vectors from (2.67) and (2.68). The planes are well-aligned between the model and the data, as is shown by the values of the dot products, which are very nearly unity:

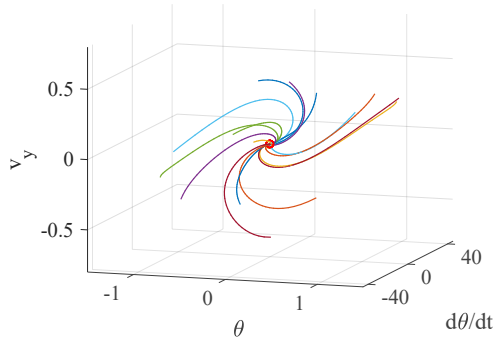
$$\mathbf{n}_o^T \tilde{\mathbf{n}}_o = 0.98, \qquad \mathbf{n}_a^T \tilde{\mathbf{n}}_a = 0.99 \tag{2.69}$$

Additionally, the natural frequency $\omega_i = |\ln(\lambda_i)/T|$ and damping ratio $\zeta_i = -\cos(\angle \ln(\lambda_i))$ of each mode computed from the model is compared with the same value computed from the data, shown in Table 2.1.

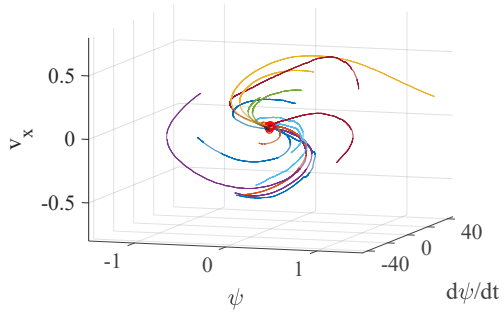
As discussed in section 2.7, an important aspect of flapping-wing flight predicted by the model is the coupling between roll and yaw within the lateral mode. Experimental trials were conducted with a flapping-wing robot to validate this behavior. The robot was clamped vertically and its wings were commanded to flap at a constant amplitude corresponding to hovering flight, while



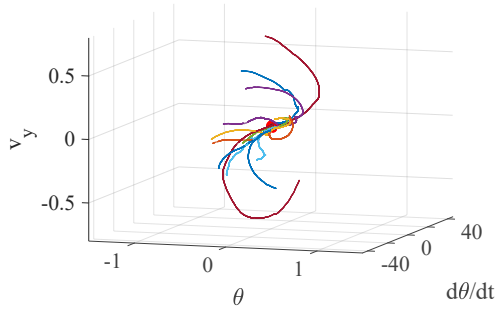
(a) Simulated trajectories in the longitudinal subspace



(b) Simulated trajectories in the lateral subspace

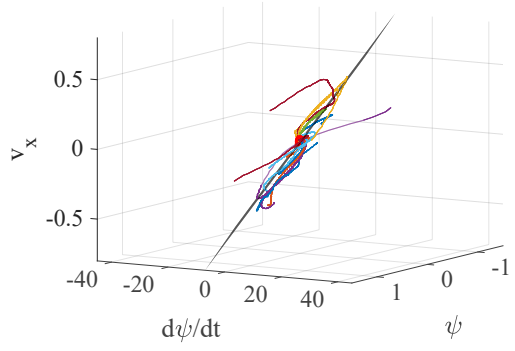


(c) Experimental trajectories in the longitudinal subspace

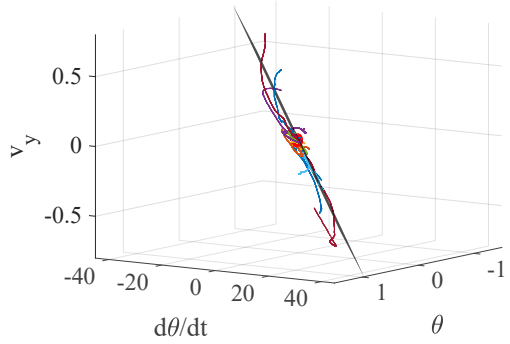


(d) Experimental trajectories in the lateral subspace

Figure 2.17: Trajectories plotted in state space obtained through simulations using the model, shown in (a) and (b) are qualitatively similar to trajectories obtained from the experiment shown in (c) and (d)



(a) The plane defined by the longitudinal mode



(b) The plane defined by the lateral mode

Figure 2.18: Phase portraits of experimental flight trajectories taken near open loop hovering lie near the dominant eigenplane defined by the linear and lateral modes, shown as black lines in plots (c) and (d)

	ω [rad/s]	ζ
Longitudinal (data)	22.8	-0.35
Longitudinal (model)	19.0	-0.17
Lateral (data)	11.1	-0.67
Lateral (model)	14.3	-0.61

Table 2.1: Comparison of the natural frequency and damping ratio of the dominant modes between the model and the data.

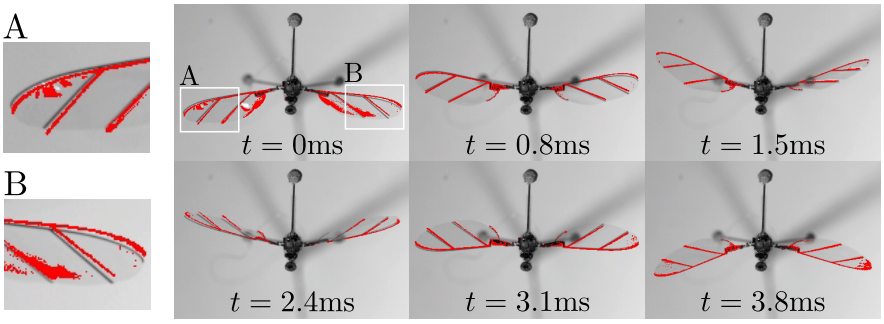


Figure 2.19: A comparison between the wing trajectories with a 0.5m/s headwind at a 0° sideslip angle (shown in grayscale) and at a 30° sideslip angle (shown in red)

a high-speed camera mounted above captured wing stroke and pitch angles. To measure the effects of forward flight on wing pitch angles, ten baseline trials were conducted with a headwind of $v = 0.5$m/s along the $\mathbf{e}_1^b$ axis. Ten more trials were conducted at the same wind speed, but at a sideslip angle of $\beta = 30°$. Figure 2.19 shows several still frames from the baseline, with the wing from the sideslip case overlayed in red. These images show that the right and left wing pitch angles are affected asymmetrically when the sideslip angle between the $\mathbf{e}_1^b$ axis and the wind direction is non-zero.

The projected angle between the first wing spar and the leading edge is used to compute the wing pitch angle from the high-speed video using the known geometry of the wing. Figure 2.20 shows how the mean pitch angles $\bar{\psi}_r$ and $\bar{\psi}_l$ computed from the video change between the baseline case with a direct headwind and the 30° sideslip case, with the mean values across all trials shown in

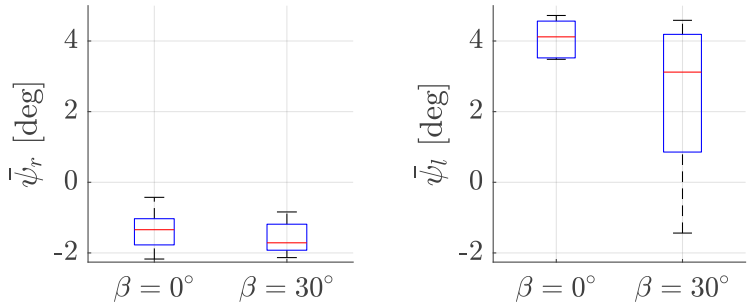


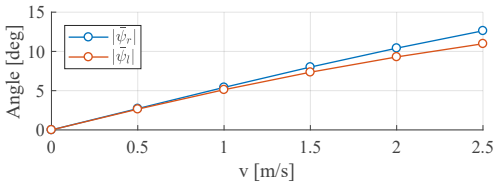
Figure 2.20: The mean right and left wing pitch angles ($\bar{\psi}_r$ and $\bar{\psi}_l$) over ten trials for the cases of a 0.5m/s headwind ($\beta = 0°$) and the same wind speed at a $\beta = 30°$ sideslip angle

red. As the wind direction shifts to the side and the sideslip angle increases, the magnitude of the left wing mean pitch angle decreases towards zero and the right wing mean pitch angle remains largely unchanged. For a given wind speed, the model predicts that the magnitude of the mean pitch angles ($|\bar{\psi}_r|$ and $|\bar{\psi}_l|$) for both wings are largest when there is a direct headwind, i.e. $\beta = 0°$. The magnitude of the mean pitch angles decrease for both wings asymmetrically as the sideslip angle increases, with $|\bar{\psi}_l|$ decreasing more than $|\bar{\psi}_r|$, matching the experimental test results shown in Fig. 2.20.

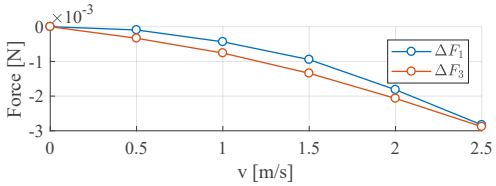
To further explore the lateral coupling with increased forward flight speed, the model is used to plot the mean pitch angles for the right and left wing as a function of increasing wind speed at $\beta = 30°$, with the results shown in Fig. 2.21a. These mean wing pitch values directly affect the stroke-averaged aerodynamic forces for each wing, and thus the stroke-averaged aerodynamic roll and yaw torques acting on the robot body in flight. The relative stroke-averaged aerodynamic forces between the right and left wings are given by

$$\Delta F_i \triangleq \int_0^T [\mathbf{F}_r(t) - \mathbf{F}_l(t)] \cdot \mathbf{e}_i^b dt, \qquad i = 1, 2, 3 \tag{2.70}$$

For a positive sideslip angle β, figure 2.21b shows that both ΔF_1 and ΔF_3 de-



(a) Absolute value of the mean wing pitch



(b) The difference between right wing and left wing forces

Figure 2.21: As the headwind velocity increases with a sideslip angle $\beta = 30°$, the mean pitch angle of the right wing is more affected than that of the left wing, resulting in less force in the $\mathbf{e}_3^b$ direction on the right side than the left and a larger negative force in the $\mathbf{e}_1^b$ direction on the right than the left

crease as a function of increasing wind speed, resulting in positive roll and yaw torque acting on the robot body. This explains the increased coupling between roll and yaw observed in the lateral mode, as any perturbation to lateral velocity v_y induces sideslip, which simultaneously generates restorative roll and yaw torque on the robot body.

2.9 Conclusion

The potential of flapping-wing robots can only be fully realized through a foundational understanding of flapping-wing flight dynamics and maneuvers, enabling the design of full-envelope flight control. This paper presented the derivation of a nonlinear flight dynamics model for minimally-actuated flapping-wing robots which captures the six body degrees of freedom as well as two rotational degrees of freedom in each wing during flight. A simplified yaw control method was proposed which maintains a constant flapping frequency during maneuvers for more power-efficient flight. As a basis for understanding the dominant linear modes of flight, detailed definitions were presented for quasi-steady aerodynamic maneuvers along with a method for solving the

equations of motion for quasi-steady set points.

Steady flight envelopes for flapping-wing flight, given in terms of wing stroke angle limits imposed by the robot geometry, were presented using the set point solutions based on the maneuver definitions. The quasi-steady maneuver set points were used to analyze the dominant linear modes and the stability of flight for a specific flapping-wing robot, which revealed stable flight regimes at moderately high speeds of forward level and descending flight. The mode shapes of the two dominant linear modes throughout the flight envelope were shown for both hovering and forward level flight. Finally, the model was validated against experimental data from a physical flapping-wing robot, which showed good agreement between the dominant modes in the model and the experimental data. Finally, experimental results were presented validating the model's predicted coupling between roll and yaw.

CHAPTER 3

ADAPTIVE FULL-ENVELOPE SPIKING NEURAL CONTROL

3.1 Introduction

Insect-scale flapping robots have recently demonstrated the ability to hover and perform basic trajectory following and other maneuvers through the implementation of modular PID and adaptive controllers [18,53,68]. Due to their size and weight, these robots have a wide variety of potential applications, including search and rescue and remote monitoring in confined spaces or hazardous environments. Generating lift through flapping as opposed to more traditional fixed-wing or rotary designs is inherently more efficient at this scale [68]. Additionally, it provides a great deal of agility, as can be observed in many flying insects [30,45].

Stabilizing and controlling these robots remains a challenging problem, however, due to a number of factors. A successful controller must be robust to uncertain physical parameters caused by manufacturing errors such as wing asymmetries, which result in a noticeable torque bias during flight that rapidly destabilizes the robot. Additionally, actuator dynamics vary between robots, leading to variations in flapping kinematics. Analytic models of aerodynamic forces are imperfect, and their accuracy can degrade during rapid maneuvers or in other poorly modeled flight regimes. The controller must be able to react quickly to stabilize the system's naturally fast, unstable dynamic modes. Additionally, the model and control law must account for the periodic flapping nature of the system. This periodicity means that the system has limit cycles instead of equilibrium points, and that linearizations thus result in time-varying

instead of time-invariant systems. These systems are also severely underactuated, possessing between 10 and 12 degrees of freedom but only 3 or 4 control inputs. This limits the effectiveness of some common techniques such as feedback linearization.

Recent work has shown that spiking neural networks (SNNs) are capable of being used as adaptive controllers for a wide variety of systems. They have been used to stabilize longitudinal fixed-wing aircraft dynamics [44], perform trajectory following on robot arms [29], perform navigation and control for terrestrial insect-like robots [71, 123], and stabilize simplified longitudinal dynamics for insect-scale flapping robots [19]. Building on these previous efforts, the work presented here develops an adaptive SNN controller capable of stabilizing hovering flight of a simulated flapping insect-scale robot. The robot model used in this work computes aerodynamic forces on the wings during flight with blade-element theory and includes 6 degrees of freedom in the main robot body [20].

An important consideration for the design of an adaptive controller for insect-scale flight is the maximum allowable rate of adaptation, which affects how quickly the controller can account for an unmodeled parameter variation online. In general, the adaptation rate for insect-scale flight control must be quite fast to correct for instabilities which can otherwise lead to a crash in just a few hundred milliseconds during open loop flight. However, instability can be induced in the closed-loop system by the controller if the adaptation rate is set too high [60, 120], although there currently is a lack of analytical quantification of the relationship between the rate of adaptation, the transient response, and the dominant timescales of open-loop instability [60]. Nonetheless, there is some evidence that SNN-based controllers may be able to adapt online more

quickly than their conventional counterparts without inducing instability from rapidly changing control gains. For instance, in [19], an SNN-based controller was used to successfully stabilize a simple 2D model of flapping-wing flight through rapid online adaptation from a random initialization of the network. The adaptive SNN-based controller developed here is capable of rapid adaptation while controlling the full 3D model of flapping-wing flight presented in Chapter 2.

SNN-based controllers have several additional desirable properties for controlling insect-scale flapping robots. Hardware implementations of SNNs have shown to be very power efficient [10], an important consideration given the small power and weight budgets for robots at this scale. For instance, the power budget for the RoboBee is around 20mW [68], most of which is required to power the actuators. Additionally, event-based sensors are available which yield low-latency feedback on the order of 15μs [66]. The output from these sensors must be processed before being used with most existing control algorithms however, thus reducing the inherent benefits of low latency and low power consumption. Integrating the event-based output from these sensors with SNN-based control algorithms has the potential to provide low latency and low power feedback control if SNN control algorithms can be developed which demonstrate robust performance for complex unstable systems. To this end, this paper presents an adaptive SNN-based controller capable of stabilizing a realistic model of an insect-scale flapping robot while adapting for unknown parameter variations.

Due to a lack of sufficient rotational damping, open loop hovering flight is unstable for many insect-scale flapping robots, including the RoboBee shown in

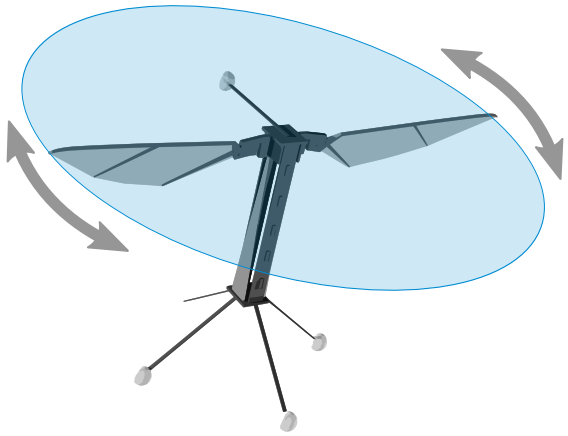


Figure 3.1: The RoboBee, with the wing stroke plane shown in blue.

Fig. 3.1 [46,110]. In systems such as the RoboBee, open loop flight tests often result in a crash in less than 0.3 seconds, leaving very little time for a poorly tuned controller to adapt to uncertain physical parameters. If the adaptive controller is not initially robust to uncertainties in physical parameters prior to any online learning, it will be unable to stabilize hovering flight long enough to learn the parameters of the robot. To achieve this, a portion of the SNN control signal is tuned offline to approximate a linear controller which was developed for the idealized robot model. A separate term in the control law is used to adapt online to any errors caused by parameter variations in the system.

Several distinct control methods have been implemented to stabilize hovering flight for flapping-wing robots previously. A hierarchical approach to designing flight controllers for insect-scale flapping robots proposed in [27]. The design calls for a stabilizing controller to compute desired forces and torques, which are passed on to a wing trajectory controller, responsible for generating periodic signals to be sent to the wings. This modular approach has been extended to several control methods for insect-scale flapping robots.

A modular PID approach was used to stabilize hovering and perform basic

trajectory following for the RoboBee, shown in Fig. 3.1, in [68]. A similar approach in [17] used the same modular architecture, but included adaptive terms to estimate wing torque biases and improve tracking performance. This design was augmented with an iterative learning scheme to perch on vertical surfaces in [18]. The adaptive approach in [16] is an extension of [18], but also accounts for wind disturbance rejection based on feedback estimates.

The SNN controller proposed here is shown to be capable of stabilizing an insect-scale flapping robot in the presence of unknown parameter variations, achieving improved performance over the linear controller on which a portion of the SNN control law is based. This is demonstrated by controlling a simulation of the RoboBee which has been previously validated against experimental data [20].

3.2 Dynamic Model

The SNN control method developed here is applicable to linear dynamic systems of the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state, $\mathbf{u} \in \mathbb{R}^m$ is the control input, $\mathbf{A}$ is the dynamics matrix, and $\mathbf{B}$ is the control input matrix. As an example case, the control design is applied here to the flapping-wing robot design presented in Chapter 2.

For the class of flapping-wing robots considered here, thrust is provided by two wings mounted to the top of the main body, as shown in Fig. 3.1. Each wing is independently actuated, and only the stroke angle of each wing can be directly controlled. Piezoelectric actuators drive transmissions that flap the

wings to generate lift. Torque control is achieved by changing the wings' relative stroke amplitudes for roll and the mean stroke angle for pitch, as shown in Fig. 2.3. This follows the control strategy described in Section 2.3, in particular (2.17) and (2.18).

Just as for many biological insects, the robot is unstable in hovering flight [46, 110] as described in Section 2.7. The instability arises because of net drag forces acting high above the robots center of mass. This unstable mode arises in both the longitudinal and lateral directions, destabilizing the robot in both pitch ψ and roll θ. The dynamics are neutrally stable in yaw, so for simplicity, yaw control is not considered in the design of the adaptive SNN hovering controller. This assumption is later relaxed for the development of a full-envelope SNN controller, where yaw control is necessary to execute coordinated turns.

The SNN control design is applied here to the flight model described in Section 2.3. If the equations of motion given in (2.23) can be linearized about hovering without significant loss of accuracy, then control design is greatly simplified. Linearization about a single point in state space will yield a poor model for control design however, as the wing states vary greatly during flight. On the other hand, linearizing about the periodic hovering trajectory should provide a good approximation for relatively small deviations from hovering. This will yield a linear time varying solution in terms of deviation from the hovering trajectory. The steady hovering trajectory is characterized by a time-varying state denoted by $\mathbf{x}^*(t)$ and a constant control input $\mathbf{u}^*$. As described in Section 2.5, the values for this trajectory can be computed by solving (2.23) for the initial conditions and constant control inputs that produce a periodic trajectory so that $\mathbf{x}^*(t) = \mathbf{x}^*(t + T)$, where the period $T = 120$Hz is equal to the flapping frequency.

After defining the state deviation $\tilde{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}^*(t)$ and the control deviation $\tilde{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}^*$, the linearized equations of motion are given by

$$\dot{\tilde{\mathbf{x}}} = \mathbf{A}(t)\tilde{\mathbf{x}} + \mathbf{B}(t)\tilde{\mathbf{u}} \tag{3.2}$$

where, for $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$,

$$\mathbf{A}(t) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}^*(t),\mathbf{u}^*}, \quad \mathbf{B}(t) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{x}^*(t),\mathbf{u}^*} \tag{3.3}$$

Due to the periodic nature of flapping flight, the model (3.2) is periodic with period T, such that

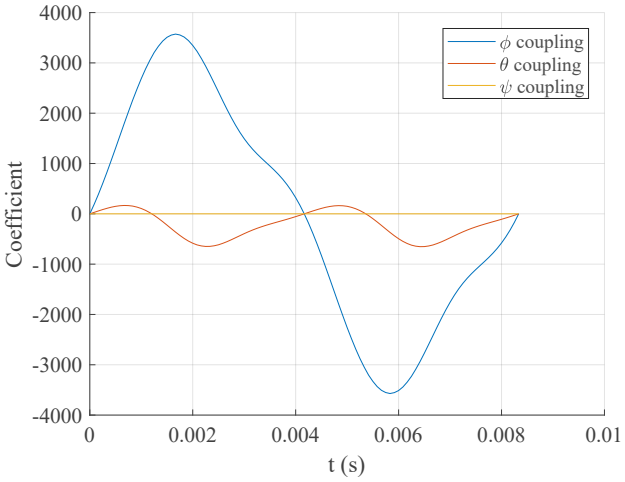
$$\mathbf{A}(t) = \mathbf{A}(t+T), \quad \mathbf{B}(t) = \mathbf{B}(t+T)$$

The linear time-varying model shown in (3.2) is far more suitable for controller design than the full non-linear time-varying equations of motion. The equations in this form do not accurately represent the system responsiveness to all control inputs however, even near the equilibrium trajectory. This is caused by the inclusion of the actuator dynamics in the full equations of motion (2.23). Linearizing the equations of motion directly, for instance by measuring the value of $\dot{\mathbf{x}}$ in response to small perturbations in $\mathbf{x}$ and $\mathbf{u}$, captures the response of the system to an instantaneous control input, which depends on the transient wing response instead of the steady state response. The transient response is, in some cases, significantly different than the steady state response and thus causes the linearized control input matrix $\mathbf{B}(t)$ to inaccurately represent the system response to control inputs.

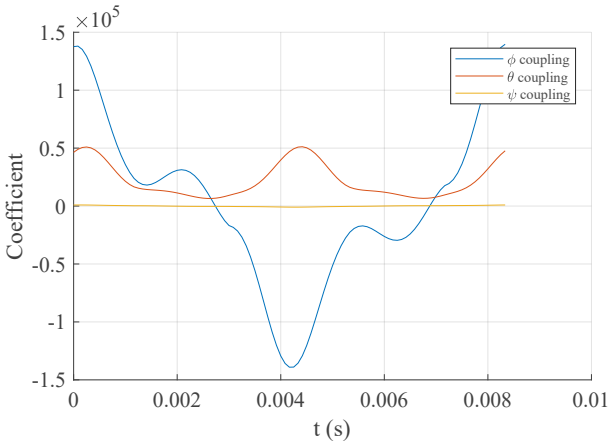
The most noticeable case of the transient wing response differing from the steady state response is after a step change to the roll input u_r, which induces roll torque in the steady state, but also couples with yaw torque in the transient response. The instantaneous response of the wings to a step change in roll input

u_r is, briefly, quite different from the steady state response. As one wing accelerates and the other decelerates, a yaw torque is generated and the body very slightly rolls opposite the desired direction. Flapping one wing faster than the other is known to theoretically cause yaw torque [68,80]. The impact on the control input matrix $\mathbf{B}(t)$ is significant. The instantaneous response to a roll control input is captured by the linearization, which then indicates that a roll control works in the opposite of the desired direction and causes significant yawing. Figure 3.2a plots the coefficients of the $\mathbf{B}(t)$ matrix that couple the roll input u_r to the body Euler angle accelerations when the $\mathbf{B}(t)$ matrix is calculated using the linearization shown in (3.2).

Since the desired effect of the linearization is to capture the response of the body due to the steady state motion of the actuators, the steady state wing motion must be computed for any control input used when numerically computing the Jacobian shown in (3.3). The state vector must first be divided into the wing states $\mathbf{x}_w$ and the body states $\mathbf{x}_b$. The steady state wing motion can then be computed by integrating the equations of motion for the wing states, $\dot{\mathbf{x}}_w = \mathbf{f}_w(\mathbf{x}, \mathbf{u}, t)$, while assuming the body remains in the equilibrium trajectory defined by $\mathbf{x}_b^*$. Applying this to the linearization results in a much improved representation of the LTV system. The coefficients of the $\mathbf{B}(t)$ matrix computed using the steady state wing trajectories are shown in Fig. 3.2b. These now correctly indicate that a positive roll control input will result in a positive roll torque. The yawing effect remains, but if the control is held constant throughout the flapping period, the yaw torque generated on the upstroke will be negated by the yaw torque generated on the downstroke.



(a) Using the transient wing response



(b) Using the steady state wing response

Figure 3.2: The coefficients of the control input matrix which couple the roll input u_r to body Euler angles differ significantly when considering the steady-state response instead of the transient response.

3.3 Proportional Integral Filter (PIF) Compensator Reference Control Design

A successful controller design for flapping wing robots must stabilize a non-linear system using periodic control inputs in the presence of unmodeled aerodynamic forces and uncertain parameters. Blade element theory gives a good

estimate of aerodynamic forces during flight, but does not account for many unsteady effects that are present during flapping flight. Additionally, physical parameters vary between robots due to manufacturing defects, which creates noticeably different flight characteristics which must be accounted for by the controller. The fast dynamic response of flapping flight at an insect scale creates additional challenges. Small body oscillations during flapping create an undesirable high-frequency component in the control law which, if applied to the system without filtering, could damage actuators and excite undesirable modes of the system dynamics. The SNN controller design which will stabilize flight in the presence of these uncertainties and disturbances must first approximate a suitably robust reference control design, after which it will adapt online for improved performance.

Proportional-integral-filter compensation [99, p.528] is an approach that addresses the issues mentioned above and can be tuned to be suitably robust to both parameter variations and unmodeled disturbances. This approach augments the state error $\tilde{\mathbf{x}}$ with both the control error $\tilde{\mathbf{u}}$ and the integral of the output error $\boldsymbol{\xi}(t) = \boldsymbol{\xi}(0) + \int_0^t \tilde{\mathbf{y}}(\tau)d\tau \in \mathbb{R}^r$. Including the control input $\mathbf{u}$ in the state vector introduces a low-pass filter on the resulting control law which is incorporated into the optimization problem. Steady-state error due to constant disturbances is addressed through the output integral $\boldsymbol{\xi}$. The resulting controller is used to stabilize the robot about a pre-computed trajectory consisting of a possibly time-varying state $\mathbf{x}^*(t)$ and a constant control input $\mathbf{u}^*$. Control gains are chosen to minimize the quadratic cost function,

$$J = \lim_{t_f \to \infty} \frac{1}{2t_f} \int_0^{t_f} \{\boldsymbol{\chi}^T(t)\mathbf{Q}'\boldsymbol{\chi}(t) + \dot{\mathbf{u}}^T(t)\mathbf{R}'\dot{\mathbf{u}}(t)\}dt \tag{3.4}$$

where the augmented state vector is $\chi = [\tilde{\mathbf{x}}^T(t)\ \tilde{\mathbf{u}}^T(t)\ \boldsymbol{\xi}^T(t)] \in \mathbb{R}^q$ and the matrix,

$$\mathbf{Q}' = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Q}_2 \end{bmatrix}$$

weights state error $\tilde{\mathbf{x}}$ through $\mathbf{Q}_1$, control error $\tilde{\mathbf{u}}$ through $\mathbf{R}_1$, and the integral of the output error $\boldsymbol{\xi}$ through $\mathbf{Q}_2$. The cost function (3.4) corresponds to the dynamic constraint given by,

$$\dot{\chi}(t) = \mathbf{A}'(t)\chi(t) + \mathbf{B}'(t)\chi(t) \tag{3.5}$$

where the dynamic matrix $\mathbf{A}'(t)$ and input matrix $\mathbf{B}'(t)$ for the augmented state equation are given by,

$$\mathbf{A}'(t) = \begin{bmatrix} \mathbf{A}(t) & \mathbf{B}(t) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{H}_x & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{B}'(t) = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \\ \mathbf{0} \end{bmatrix} \tag{3.6}$$

and the matrix $\mathbf{H}_x$ relates the state to the output.

For simplicity, a subset $\mathbf{x}_c$ of the complete state vector is used for the controller design. The wing states, body position, and yaw are all ignored. The state subset includes the roll θ, pitch ψ, roll rate $\dot{\theta}$, pitch rate $\dot{\psi}$, body x velocity v_x, the body y velocity v_y, and the body z velocity v_z. The output for the controller design is defined to contain only roll and pitch, so that

$$\mathbf{x}_c = \begin{bmatrix} \theta & \psi & \dot{\theta} & \dot{\psi} & v_x & v_y & v_z \end{bmatrix}^T$$
$$\mathbf{y} = \begin{bmatrix} \theta & \psi \end{bmatrix}^T$$

In the dynamics matrix $\mathbf{A}'(t)$ and input matrix $\mathbf{B}'(t)$ in (3.5), only the coefficients corresponding directly to this subset of state variables are used.

Given a cost function of the form (3.4), the minimizing control law is

$$\begin{aligned} \dot{\mathbf{u}}(t) &= -\mathbf{K}(t_p)\boldsymbol{\chi}(t) \\ \dot{\mathbf{u}}(t) &= -\mathbf{R}'^{-1}\mathbf{B}'^{T}\mathbf{S}(t_p)\boldsymbol{\chi}(t) \end{aligned} \tag{3.7}$$

where the period time $t_p = t \bmod T$ is the amount of time elapsed since the beginning of the current flapping period. The control law given in (3.7) gives an optimal policy for the time derivative of the control law. Equation (3.7) must be integrated to find the control input $\mathbf{u}(t)$ to be provided to the plant. The gain $\mathbf{K}(t)$ is dependent on $\mathbf{S}(t)$, which is the periodic solution to the matrix Riccati equation

$$\begin{aligned} \dot{\mathbf{S}}(t) = &- \mathbf{A}'^{T}(t)\mathbf{S}(t) - \mathbf{S}(t)\mathbf{A}'(t) - \\ &\mathbf{Q}' + \mathbf{S}(t)\mathbf{B}'(t)\mathbf{R}'^{-1}\mathbf{B}'^{T}(t)\mathbf{S}(t) \end{aligned} \tag{3.8}$$

A closed-form solution to the continuous periodic Riccati is not available, but its solution can be found by integrating (3.8) until it satisfies $\mathbf{S}(t) = \mathbf{S}(t+T)$ within a specified tolerance. The final period of the resulting solution can then be used to calculate the periodic gain $\mathbf{K}(t)$. This computation is performed offline for each trajectory, but can be computationally expensive. Care must be taken to ensure that the resulting solution is symmetric, which can be enforced during integration by periodically replacing the calculated solution $\mathbf{S}_c(t)$ with the symmetric result $\mathbf{S}(t) = 1/2[\mathbf{S}_c(t) + \mathbf{S}_c^T(t)]$. Additional methods are available which can reduce the computation time, including general methods for numerical solutions to differential equations [9] as well as algorithms specifically for the Riccati equation [99].

Computing the solution to the periodic Riccati equation (3.8) is in general quite inefficient. This in turn makes it difficult to fine-tune the augmented state

and input weighting matrices $\mathbf{Q}'$ and $\mathbf{R}'$. An alternative is to average the dynamics over an entire flapping period and then neglect the motion of the wings. This so-called stroke-averaging technique is commonly used as a method of simplifying the dynamics of insect flight for analysis [88, 110]. Although this is a difficult task for the complete equations of motion (2.23) due to the passive degree of freedom in wing pitch, the linear time-varying system (3.2) can easily be stroke-averaged. Beginning with the augmented dynamic equation (3.5), the stroke-averaged model takes the form,

$$\dot{\boldsymbol{\chi}}(t) = \bar{\mathbf{A}}'\boldsymbol{\chi}(t) + \bar{\mathbf{B}}'\dot{\mathbf{u}}(t) \tag{3.9}$$

where

$$\bar{\mathbf{A}}' = \frac{1}{T}\int_0^T \mathbf{A}'(t)dt, \quad \bar{\mathbf{B}}' = \frac{1}{T}\int_0^T \mathbf{B}'(t)dt \tag{3.10}$$

The frequency of the control gains computed in (3.7) is much higher than the bandwidth of the actuators. The settling time for the stroke angle in response to a step input is typically at least 5 complete flapping periods. Thus, the stroke-averaged control inputs computed using the model (3.9) will yield similar performance to the time-varying model (3.5) in practice.

Because the stroke-averaged model is time-invariant, the optimal control law becomes,

$$\begin{aligned} \dot{\mathbf{u}}(t) &= -\mathbf{R}'^{-1}\mathbf{B}'^{T}\mathbf{S}\boldsymbol{\chi}(t) \\ &= -\bar{\mathbf{K}}\boldsymbol{\chi}(t) \end{aligned} \tag{3.11}$$

and the solution to the Riccati equation $\mathbf{S}$ is found by solving the algebraic Riccati equation

$$\mathbf{0} = -\bar{\mathbf{A}}'^{T}\mathbf{S} - \mathbf{S}\bar{\mathbf{A}}' - \mathbf{Q}' + \mathbf{S}\bar{\mathbf{B}}'\mathbf{R}'^{-1}\bar{\mathbf{B}}'^{T}\mathbf{S} \tag{3.12}$$

The algebraic Riccati equation can be solved much more efficiently than the differential Riccati equation (3.8). Thus, using the stroke-averaged linearized

equations of motion significantly simplify the problem of finding the optimal control gains for the control law.

3.4 Adaptive SNN Control Design

The control signal can be provided entirely by populations of spiking neurons using an adaptation of the approach presented in [29]. Each individual population of neurons is structured as a single-layer feed forward network. A total of four separate neuron populations are used to compute the complete control signal, which is the summation of two terms. One term approximates the signal from a linear controller that is designed to stabilize the ideal model [20], and the other is an adaptive term to account for parameter variations in the robot. A total of 800 neurons are used across all four populations.

The PIF compensator described in the previous section is used as the reference linear controller for the SNN approximation. Integrating (3.11) yields the reference control law

$$\mathbf{u}_{PIF}(t) = \int_0^t -\mathbf{K}\boldsymbol{\chi}(\tau)d\tau \tag{3.13}$$

For simplicity, the control input error and integral of the output error are both assumed to be zero for the SNN approximation. The most significant terms in the PIF control signal are dependent on the state error $\tilde{\mathbf{x}}(t)$ only.

To formulate the PIF approximation term $\mathbf{u}_0$ in terms of spiking neurons, the Neural Engineering Framework (NEF) [35, 101] is used. The NEF provides a framework for representing algorithms in terms of spiking neuron models, and includes methods for both encoding time-varying vectors in populations

of neurons and for decoding the output of spiking neurons into a continuous signal.

The state $\mathbf{x}(t)$ is encoded in a population of neurons as a sequence of spikes from each neuron. The nonlinear spiking neuron model of the ith neuron within the population is $G_i : \mathcal{J} \times \mathcal{T} \to \mathbb{R}$, which maps the input current $J_i \in \mathcal{J} \subset \mathbb{R}$ and time $t \in \mathcal{T} \subset \mathbb{R}_0^+$ to the spike train $r_i \in \mathbb{R}$,

$$r_i = G_i(J_i(\mathbf{x}), t) \tag{3.14}$$

The total current J_i into neuron i is written in terms of a gain term $\alpha_i \in \mathbb{R}$, the neuron's preferred direction vector $\mathbf{e}_i \in \mathbb{R}^n$, and a fixed background current $J_i^{bias} \in \mathbb{R}$,

$$J_i(\mathbf{x}) = \alpha_i \mathbf{e}_i^T \mathbf{x}(t) + J_i^{bias} \tag{3.15}$$

The nonlinear spiking neuron model G_i used in this work is the leaky integrate-and-fire model [52]. This models the voltage V_i across the membrane of a neuron as an RC circuit governed by the first-order ODE

$$\tau_{RC} \frac{dV_i}{dt} = -V_i(t) + R_i J_i(\mathbf{x}), \qquad V_i(0) = V_0 \tag{3.16}$$

where τ_{RC} is the decay time constant and R_i is the resistance in the circuit and accounts for the passive "leak" of the current. The spike train r_i of the neuron can be written as a summation of Dirac delta functions indexed by k. A spike time t_{ik} is defined when the voltage reaches a threshold V_{th}, following which the membrane potential V_i is set to the reset value $V_r < V_{th}$. Expressed formally,

$$t_{ik} = \{\tau : \ V_i(\tau) = V_{th}\} \tag{3.17}$$

$$\lim_{t \to t_{ik}^+} V_i(t) = V_r \tag{3.18}$$

Following a spike, neuronal activity is held at the reset value V_r for a refractory period τ_{ref}, after which the membrane potential $V_i(t)$ again follows (3.16).

The spiking activity contained in r_i can be expressed using the spike times t_{ik} as,

$$r_i = \sum_k \delta_i(t - t_{ik}) \tag{3.19}$$

where $\delta_i(t)$ denotes the Dirac delta function. To decode a continuous estimate of the state $\mathbf{x}(t)$ from the spiking activity r_i, the filtered postsynaptic activity is used. In this model, synapses act as linear filters $h_i(t)$ on the spike trains. This can be explicitly stated using the postsynaptic time constant τ_{PSC} as $h_i(t) = (1/\tau_{PSC})e^{-t/\tau_{PSC}}$. The activity a_i of neuron i is thus the summation of impulse responses from linear filters, which can also be written as the convolution of the synaptic filters $h_i(t)$ with the spike train,

$$\begin{aligned} a_i(\mathbf{x}, t) &= h_i(t) * G_i(J_i(\mathbf{x}), t) \\ &= \sum_k h_i(t - t_{ik}) \end{aligned} \tag{3.20}$$

where "$*$" denotes the convolution operator. The activity of the population of neurons can be used to represent the PIF control signal through the use of carefully chosen linear decoders $\mathbf{d}_i \in \mathbb{R}^m$,

$$\mathbf{u}_0(t) = \sum_i a_i(\mathbf{x}, t)\mathbf{d}_i \tag{3.21}$$

The linear decoders which give the optimal representation of the PIF control signal are found using standard least squares optimization over a set $\mathcal{B}$ of sampled data,

$$\mathbf{d}_i = \underset{\mathbf{m}_i}{\operatorname{argmin}} \int_{\mathbf{x}\in\mathcal{B}} \| \mathbf{g}(\mathbf{x}) - \sum_i \bar{a}_i(\mathbf{x})\mathbf{m}_i \|^2 \, d\mathbf{x} \tag{3.22}$$

where $\mathbf{g}(\mathbf{x})$ is the output of the PIF control law for a given point in state space and $\bar{a}_i(\mathbf{x})$ is the time-averaged steady-state neuron activity for the same point. The

elements of $\mathcal{B}$ are sampled from a subset of the state space near the hovering condition, where the neuron population is expected to be capable of approximating the PIF control law.

The domain $\mathcal{A} \subset \mathbb{R}^n$ is a subset of the state space near the hovering condition. The sampled data includes all points in state space where the neuron population is expected to be capable of approximating the PIF control law. The desired output of the neuron population at any point $\mathbf{x}$ is the output of the PIF control law at the same point.

The control input $\mathbf{u}_0(t)$ closely approximates the signal from the PIF compensator, which is designed to control a linearized model of a flapping-wing robot [20]. In reality however, the robot is subject to manufacturing defects which are not present in the model. These defects include variations in the wing hinges and asymmetries between the actuators that result in anomalous torque generation during flight. An adaptive control input $\mathbf{u}_{adapt}(t)$ accounts for this and other unmodeled dynamics present in the system. The computation of $\mathbf{u}_{adapt}(t)$ follows a direct adaptive method similar to that outlined in [29, 93, 96].

The adaptive term $\mathbf{u}_{adapt}(t)$ contains three scalar quantities: an amplitude input $u_a(t)$, a pitch input $u_p(t)$, and a roll input $u_r(t)$,

$$\mathbf{u}_{adapt}(t) = \begin{bmatrix} u_a(t) & u_p(t) & u_r(t) \end{bmatrix}^T \tag{3.23}$$

Each of these scalars are computed from a separate population of neurons, as shown in Fig. 3.3. Each population is trained online to zero out an error signal that depends on the error $\Delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_{ref}(t)$ between the state $\mathbf{x}(t)$ and the reference state $\mathbf{x}_{ref}(t)$, a constant gain term $\alpha \in \mathbb{R}_0^+$, and the constant vector $\mathbf{\Lambda} \in \mathbb{R}^n$

$$E(t) = \mathbf{\Lambda}^T[\Delta\mathbf{x}(t) + \alpha\Delta\dot{\mathbf{x}}(t)] \tag{3.24}$$

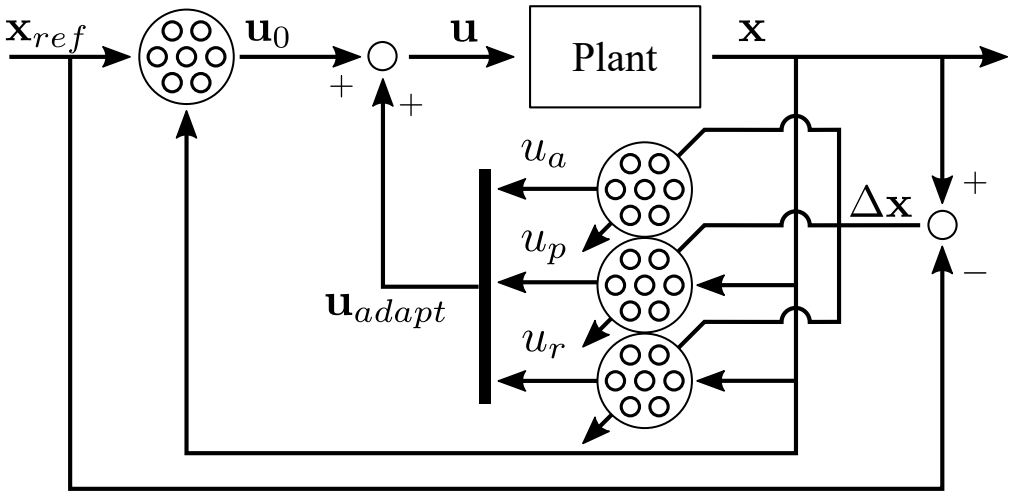


Figure 3.3: The SNN control architecture, where clusters of circles represent neuron populations.

In general, variations in actuator performance between robots result in constant biases to the effective gain of the amplitude input and state-dependent biases to the effective gain of the pitch and roll inputs. The adaptive amplitude input $u_a(t)$ is computed by decoding the neuron activity through linear decoders $d_{a,i} \in \mathbb{R}$,

$$u_a(t) = \sum_i a_{a,i}(t) d_{a,i}(t) \tag{3.25}$$

where the activity of the neurons in this population is

$$a_{a,i}(t) = h_i(t) * G_i(J_i^{bias}, t) \tag{3.26}$$

This can be written more compactly by defining the activity for the amplitude population as $\mathbf{a}_a = [a_{a,1},\ a_{a,2},\ \ldots,\ a_{a,N}]^T$ and the corresponding population decoders $\mathbf{d}_a = [d_{a,1},\ d_{a,2},\ \ldots,\ d_{a,N}]^T$. The amplitude control signal (3.25) can now be rewritten as

$$u_a(t) = \mathbf{a}_a^T(t)\mathbf{d}_a(t) \tag{3.27}$$

The linear decoders $\mathbf{d}_a(t)$ for the amplitude population are continuously updated based on an adaptation rate $\gamma \in \mathbb{R}_0^+$, the population activity, and the error

signal (3.24),

$$\dot{\mathbf{d}}_a(t) = \gamma \mathbf{a}_a(t) E(t) \tag{3.28}$$

For the amplitude input $u_a(t)$, the vector $\mathbf{\Lambda}$ is chosen so that the error function is only a function of the body z velocity error $\Delta v_z(t)$ and its derivative.

The adaptive pitch input $u_p(t)$ and adaptive roll input $u_r(t)$ are computed similarly to the adaptive amplitude input (3.27). Pitch and roll torques experienced during flight are in general a function of the state, because drag on the wings varies as a function of the robot's angular and linear velocity. This is accounted for in both $u_p(t)$ and $u_r(t)$ by incorporating the state in the neural activity, as shown in (3.20)

$$u_p(t) = \mathbf{a}_p^T(\mathbf{x}, t)\mathbf{d}_p(t) \tag{3.29}$$

$$\dot{\mathbf{d}}_p(t) = \gamma \mathbf{a}_p(\mathbf{x}, t) E(t) \tag{3.30}$$

The pitch input is computed to reduce the error in body x velocity $v_x(t)$ by choosing $\mathbf{\Lambda}$ in the error function (3.24) so that the error depends only on $v_x(t)$ and its derivative.

The roll input $u_r(t)$ is computed identically to the pitch input (3.29), except that $\mathbf{\Lambda}$ in the error function is chosen so that the error depends only on the body y velocity $v_y(t)$ and its derivative,

$$u_r(t) = \mathbf{a}_r^T(\mathbf{x}, t)\mathbf{d}_r(t) \tag{3.31}$$

$$\dot{\mathbf{d}}_r(t) = -\gamma \mathbf{a}_r(\mathbf{x}, t) E(t) \tag{3.32}$$

The sign difference between the right hand side of (3.30) and (3.32) results from differences in sign convention between the two control inputs.

In summary, the adaptive amplitude input (3.27) is designed to zero out any error in body z velocity, the pitch input (3.29) is designed to zero out error in

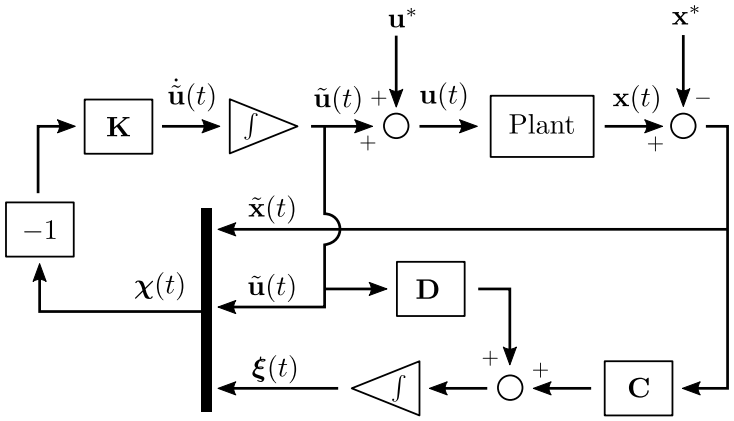


Figure 3.4: Block diagram of the PIF compensator.

the body x velocity, and the roll input (3.31) will zero out the error in body y velocity. The complete control signal sent to the plant is the sum of (3.21) and (3.23),

$$\mathbf{u}(t) = \mathbf{u}_0(t) + \mathbf{u}_{adapt}(t) \tag{3.33}$$

The first term $\mathbf{u}_0(t)$ is trained offline to control the ideal model, and the adaptive term $\mathbf{u}_{adapt}(t)$ is trained online to minimize the steady-state error $\Delta\mathbf{x}(t)$ caused by parameter variations and constant disturbances.

3.5 Full-Envelope SNN Control Design

The PIF gain matrix $\mathbf{K} \in \mathbb{R}^{m\times(n+m+r)}$ can be partitioned into $\mathbf{K}_1 \in \mathbb{R}^{m\times n}$, $\mathbf{K}_2 \in \mathbb{R}^{m\times m}$, and $\mathbf{K}_3 \in \mathbb{R}^{m\times r}$. Using these partitioned matrices, the PIF control law (3.11) can be written as

$$\dot{\tilde{\mathbf{u}}}(t) = -\mathbf{K}\boldsymbol{\chi}(t) = -\mathbf{K}_1\tilde{\mathbf{x}}(t) - \mathbf{K}_2\tilde{\mathbf{u}}(t) - \mathbf{K}_3\boldsymbol{\xi}(t) \tag{3.34}$$

The third term, $\mathbf{K}_3\boldsymbol{\xi}(t)$ is the integral term designed to minimize steady-state error and account for discrepancies between the linearized model and the physical plant. Since the SNN controller will later be modified to include an adaptive

term designed to account for the same errors, this term is neglected in the PIF approximation. Neglecting this term, the transfer function for the PIF control law becomes

$$\mathbf{G}(s) \triangleq \frac{\mathbf{U}(s)}{\mathbf{X}(s)} = -(s\mathbf{I} + \mathbf{K}_2)^{-1}\mathbf{K}_1 \tag{3.35}$$

This transfer function is simply a low-pass filtered gain matrix. Using the final value theorem, the steady-state PIF gain matrix is

$$\mathbf{G}(0) = -\mathbf{K}_2^{-1}\mathbf{K}_1 \triangleq \mathbf{K}_{ss} \tag{3.36}$$

A single-layer feed-forward SNN is used to approximate the PIF control law over the entire flight envelope, using $\mathbf{y}^*$ as the input to the network and taking the output of the network as the approximate gain matrix. The transfer function for the SNN from the vector of spike trains $\mathbf{P}(s)$, generated by the nodes in the network, to the post synaptic current $\mathbf{R}(s)$ can be written as

$$\frac{\mathbf{R}(s)}{\mathbf{P}(s)} = H(s)\mathbf{W} \tag{3.37}$$

The PIF gains are approximated by tuning the network output weights $\mathbf{W}$ so that the steady-state post synaptic current matches the steady-state PIF gain matrix $\mathbf{K}_{ss}$ for all desired values of the command input $\mathbf{y}^*$. The post-synaptic filter $H(s)$ can then be tuned to match the gain of the low-pass filter from the original PIF control law.

3.5.1 Adaptive SNN-based Hovering Control Results

Closed loop flight test simulations were conducted using both the SNN controller (3.33) and the PIF compensator (3.13) to control the model from Chapter 2. Compared to previous stroke-averaged models, this model more accurately

represents the coupling between various degrees of freedom in the robot by using blade element theory to compute instantaneous aerodynamic forces during flapping. The model was compared with experimental data to validate its effectiveness at representing the flight dynamics of the physical robot. To verify the ability of each controller to compensate for parameter variations in the model, wing asymmetry was introduced in both simulations by applying a static amplitude bias between the right and left wings of the model for roll and a static mean stroke angle offset for pitch. In both cases, the controllers were commanded to control hovering flight.

The attitude and body velocity during the flight simulation using the PIF compensator are shown in Fig. 3.5 and Fig. 3.6. Although the integral term in the PIF compensator does work to eliminate steady-state error in v_x and v_y as shown in Fig. 3.6, it does so slowly, and at the end of the 6 second test flight the robot maintains a significant non-zero velocity. Due to the positional drift, the robot continues yawing throughout the simulation as shown in Fig. 3.5. However, the PIF compensator is able to quickly stabilize pitch and roll to near zero.

The same initial conditions and wing biases were used to simulate another closed loop flight, this time using the adaptive SNN (3.33) to control the simulated robot. The parameters for the adaptive input $\mathbf{u}_{adapt}(t)$ were tuned using a grid parameter search. The search varied the parameters of the decoder update laws to find the values that lead to the smallest total deviation from the starting hover position in meters. The attitude and body velocity from a trial with simulated wing bias and random initial conditions are shown in Fig. 3.7 and Fig. 3.8, respectively. During this simulated flight test, the RoboBee remained

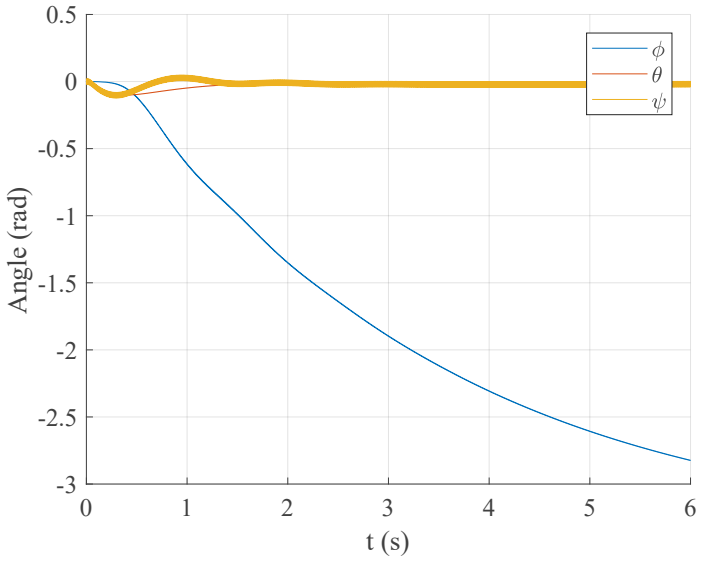


Figure 3.5: For comparison with the SNN controller, the PIF compensator quickly stabilizes roll θ and pitch ψ, but the yaw angle ϕ is not stabilized.

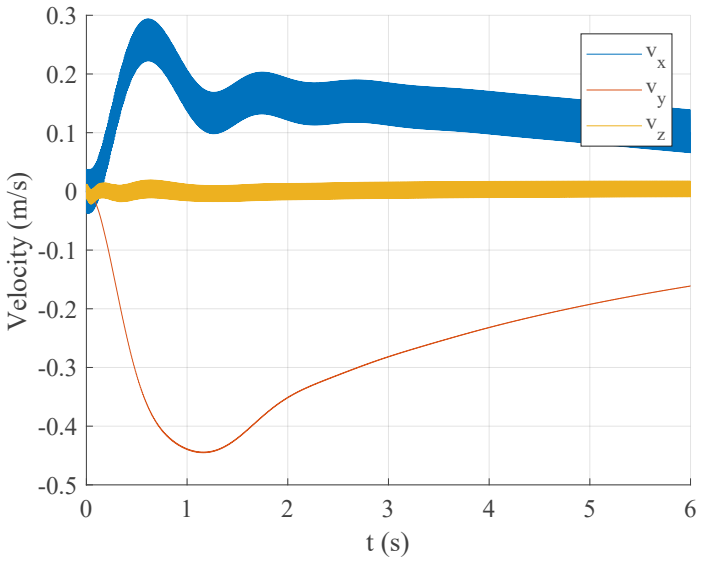


Figure 3.6: For comparison with the SNN controller, the PIF compensator successfully stabilizes v_z, but only slowly stabilizes v_x and v_y, resulting in significant drift from the initial position.

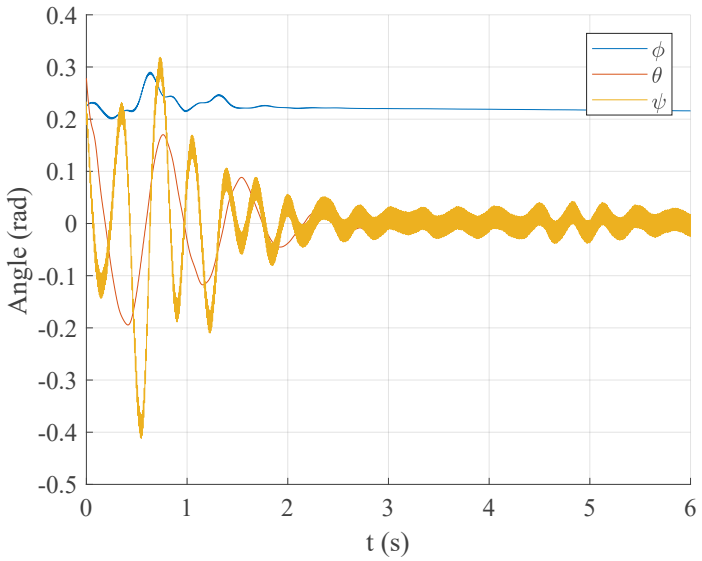


Figure 3.7: During a simulated hovering flight, the adaptive SNN successfully stabilizes roll θ and pitch ψ about zero despite parameter variations in the model, while the yaw angle ϕ stabilizes at a constant non-zero value.

within 10cm of its starting position.

The body attitude and velocity are both stabilized by the adaptive SNN controller. The yaw angle ϕ settles at a non-zero constant in Fig. 3.7 and the pitch angle ψ and roll angle θ both stabilize near zero. The yaw angle was not included in the set point and thus was not expected to reach zero, since the controller lacks direct yaw control authority. All components of the body velocity stabilize near zero as shown in Fig. 3.8 after approximately 3 seconds. Comparing the RoboBee attitude and velocity when controlled by the PIF compensator (Fig. 3.5 and Fig. 3.6) with the attitude and velocity when controlled by the adaptive SNN (Fig. 3.7 and Fig. 3.8) demonstrates the ability of the adaptive SNN to account for the unmodeled torque disturbances caused by the simulated wing bias.

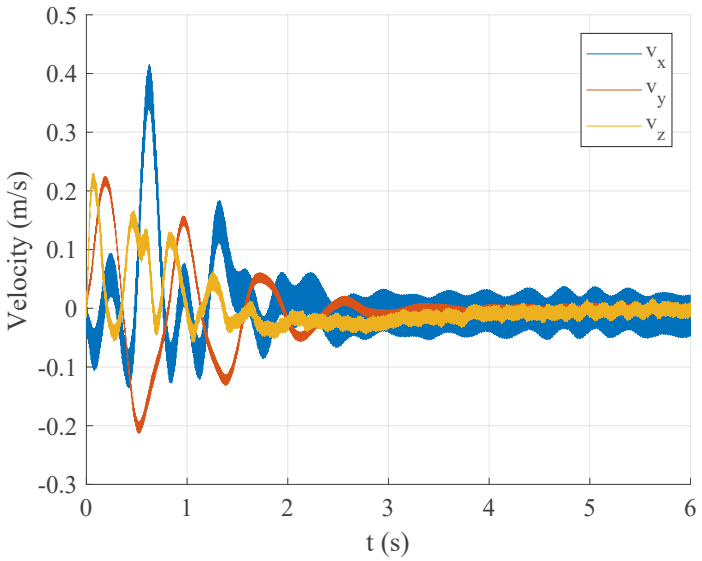


Figure 3.8: During a simulated hovering flight, the adaptive SNN successfully stabilizes all components of the body velocity near zero despite parameter variations.

3.5.2 Full-envelope SNN-based Flight Control Results

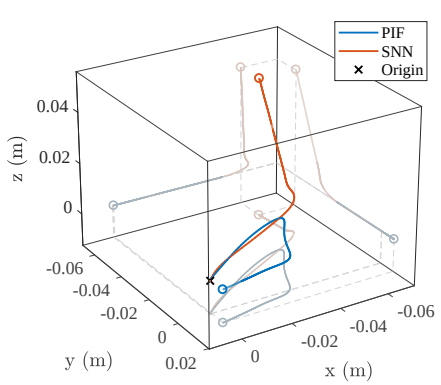
Here, the performance of the full-envelope SNN controller is compared against a gain-scheduled PIF compensator for the task of controlling a simulated flapping-wing robot as it performs a variety of maneuvers throughout the flight envelope. The PIF controller provides a stabilizing control law for the RoboBee over the entire flight envelope, including hovering, vertical flight, longitudinal flight, and steady climbing turns. The SNN controller is trained offline to approximate the PIF control law over the flight envelope. The closed-loop performance for several maneuvers, shown in Table 3.1, is shown here for the SNN controller, comparing it in each case to the closed-loop PIF performance.

Trial 1 tests the ability of the SNN controller to recover from an initial disturbance and maintain hovering flight. Figure 3.9 compares the closed-loop response of the PIF-controlled robot and the SNN-controlled robot with the same

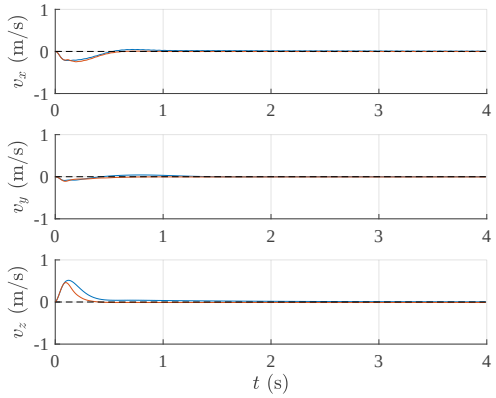
Trial	Maneuver	Figure
1	Hovering	3.9
2	Take-off	3.10
3	Steady lateral flight	3.11
4	Steady climbing turn	3.12
5	Up-and-over	3.13
6	Sinusoidal climb angle	3.14
7	Orbit around target	3.15

Table 3.1: A summary of the different trials used to test the performance of the SNN controller.

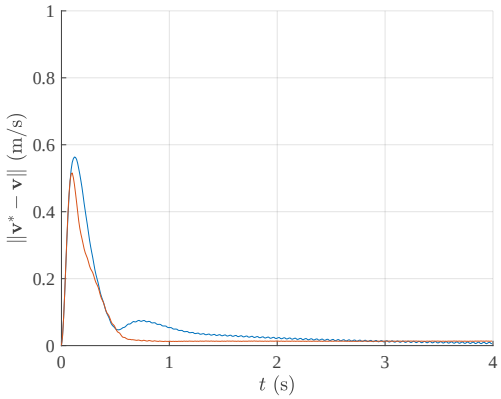
initial conditions. Although the trajectories of each appear dissimilar, they each drift by only a few centimeters from the origin. The velocity tracking performance is good for both controllers, as shown in Fig. 3.9b. The control inputs provided by the SNN controller are shown in Fig. 3.9d, in which they are compared to the inputs the PIF controller would have provided given the same state information. This plot shows that the primary difference between the two controllers is the additional low-pass filtering of the control signal caused by the synapses on the SNN controller.



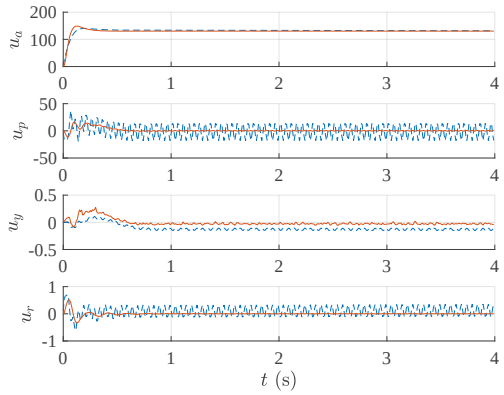
(a) Trajectories



(b) Velocity tracking performance



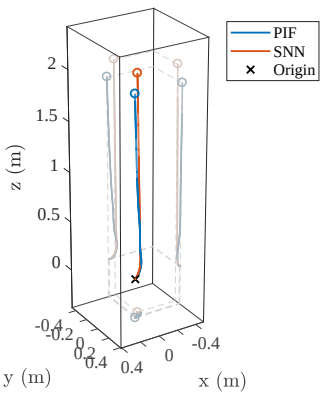
(c) Norm of velocity tracking error.



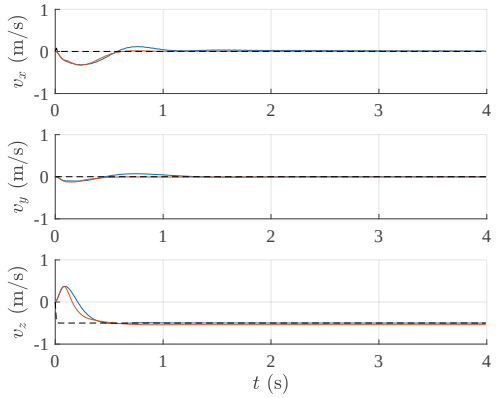
(d) Comparison between PIF and SNN control signals

Figure 3.9: The RoboBee attempting to hover after an initial disturbance. Both controllers yield very similar velocity tracking performance. The synapses on the SNN controller have been tuned to filter out the high frequency component of the control signal present in the signal from the PIF.

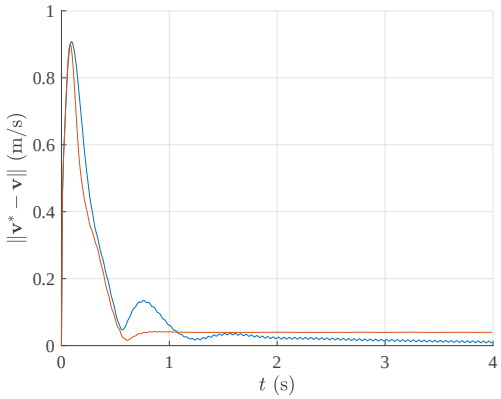
Trial 2 (shown in Fig. 3.10) shows the closed-loop performance of both controllers performing a vertical take-off maneuver at $v^* = 0.5\text{m/s}$ after an initial disturbance. The SNN controller again performs very similarly to the PIF controller.



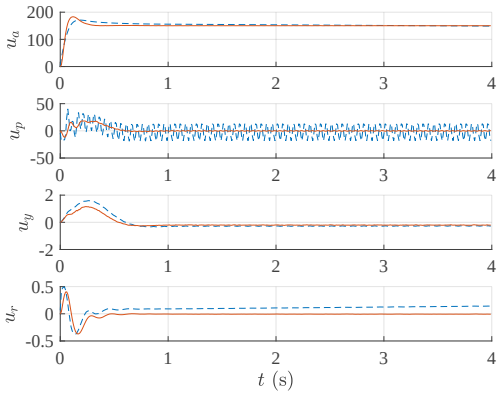
(a) Trajectories



(b) Velocity tracking performance



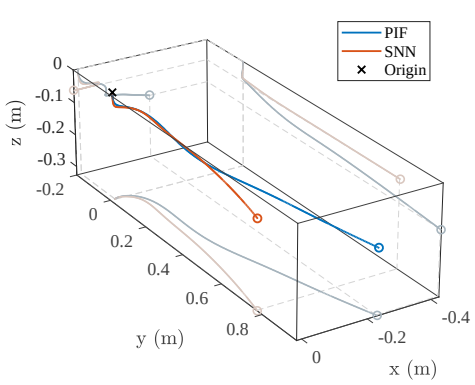
(c) Norm of velocity tracking error.



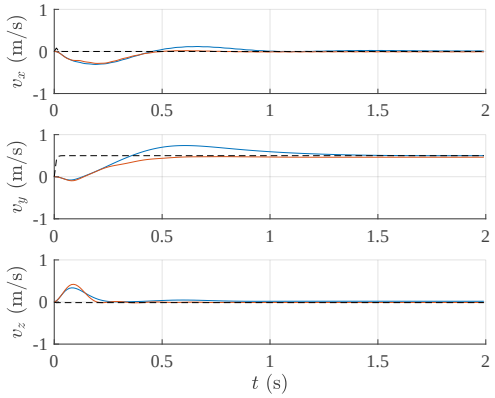
(d) Comparison between PIF and SNN control signals

Figure 3.10: Closed-loop response of the RoboBee performing a vertical takeoff maneuver after an initial disturbance. Both controllers again perform very similarly, and both maintain small velocity tracking errors.

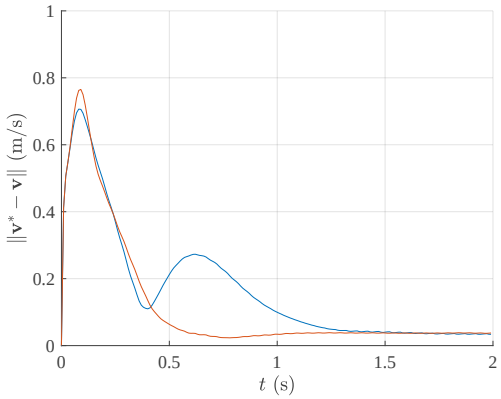
Trial 3 (Fig. 3.11) shows the closed-loop performance of both controllers performing steady-level flight with $v^* = 0.5\text{m/s}$ and $\beta^* = 90°$ after an initial disturbance. The control input from the SNN closely follows the input that would have been provided by the PIF, and the controller successfully executes the maneuver with similar tracking performance to the PIF.



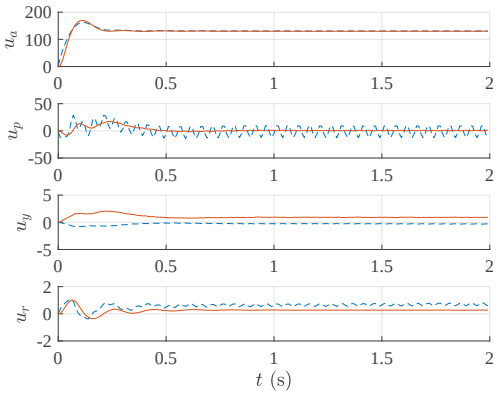
(a) Trajectories



(b) Velocity tracking performance



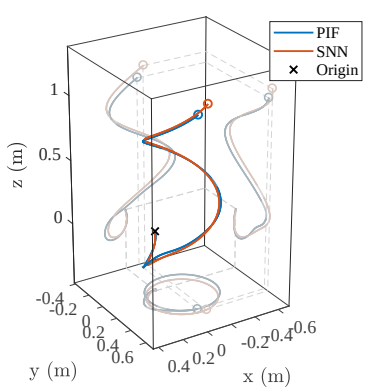
(c) Norm of velocity tracking error.



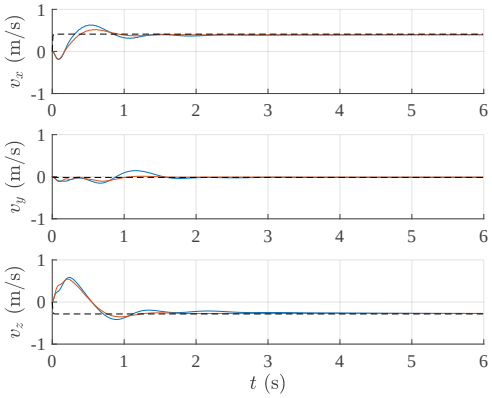
(d) Comparison between PIF and SNN control signals

Figure 3.11: Closed-loop response of the RoboBee in steady level flight with a sideslip angle of $\beta = 90°$ after an initial disturbance. Both controllers again perform very similarly and maintain small velocity tracking errors.

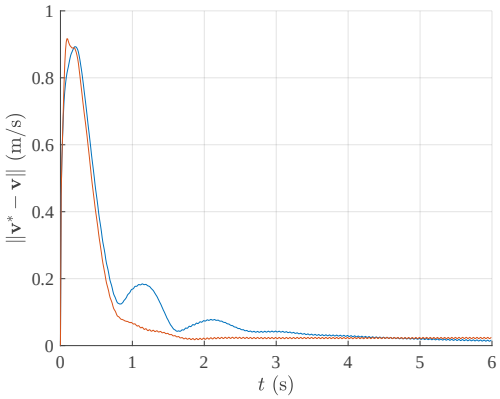
Trial 4 (Fig. 3.12) shows the closed-loop performance of both controllers performing a steady climbing turn with $v^* = 0.5\text{m/s}$, $\gamma^* = 30°$, and $\dot{\xi}^* = 90°\text{s}^{-1}$ after an initial disturbance. The SNN controller provides control inputs that are very similar to the PIF and successfully executes the maneuver with very similar velocity tracking performance (shown in Fig. 3.12b) to the PIF.



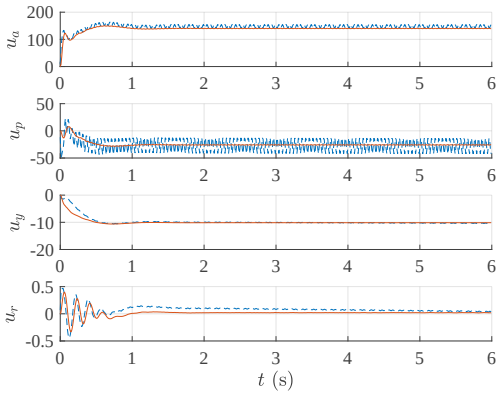
(a) Trajectories



(b) Velocity tracking performance



(c) Norm of velocity tracking error.



(d) Comparison between PIF and SNN control signals

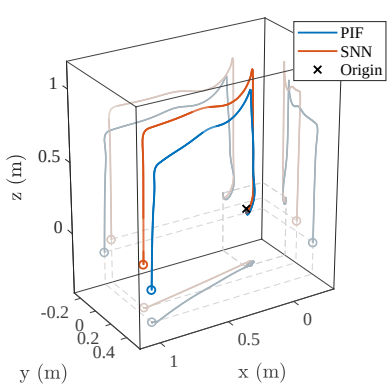
Figure 3.12: Closed-loop response of the RoboBee performing a steady climbing turn after an initial disturbance, with $v^* = 0.5\text{m/s}$, $\dot{\xi}^* = 90°\text{s}^{-1}$, and $\gamma^* = 30°$. Both controllers again perform very similarly, and both maintain small velocity tracking errors.

Trial 5 (Fig. 3.13) shows the ability of the SNN controller to cope with rapidly changing command inputs. The robot is commanded to climb vertically from $t = 0$ to $t = 2$, then to fly forward in steady level flight from $t = 2$ to $t = 4$, and finally to descend vertically from $t = 4$ to $t = 6$. In each case, the desired speed

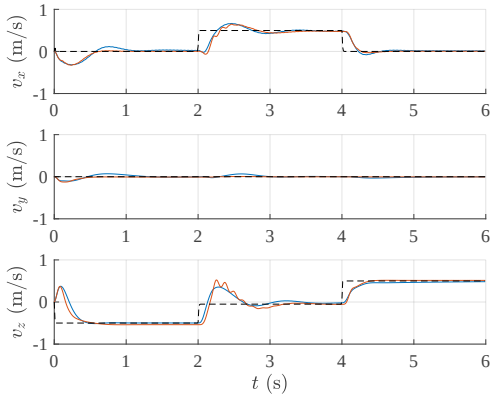
is $v^* = 0.5\text{m/s}$. In other words,

$$\mathbf{y}^* = \begin{cases} \begin{bmatrix} 0.5 & \pi/2 & 0 & 0 \end{bmatrix}^T, & \text{if } 0 < t \leq 2 \\ \begin{bmatrix} 0.5 & 0 & 0 & 0 \end{bmatrix}^T, & \text{if } 2 < t \leq 4 \\ \begin{bmatrix} 0.5 & -\pi/2 & 0 & 0 \end{bmatrix}^T, & \text{if } t > 4 \end{cases} \tag{3.38}$$

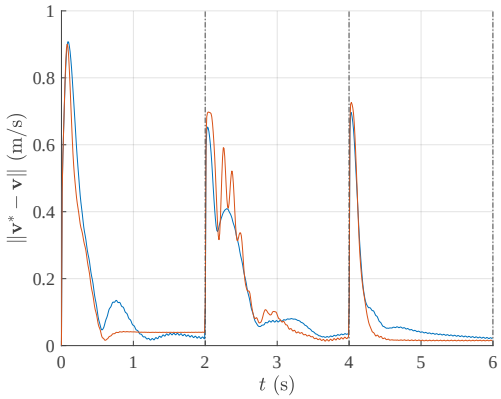
The SNN successfully performs the maneuver, obtaining very similar tracking performance to the PIF and generating similar control inputs.



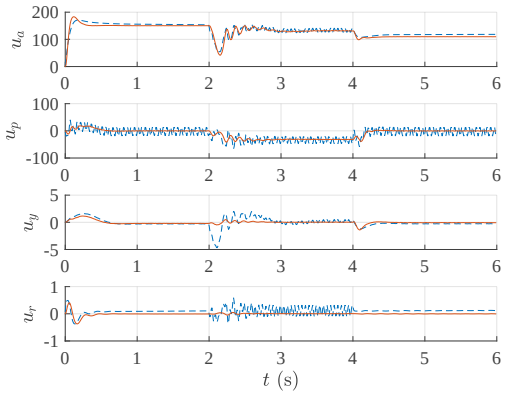
(a) Trajectories



(b) Velocity tracking performance



(c) Norm of velocity tracking error. Vertical lines indicate command input changes.

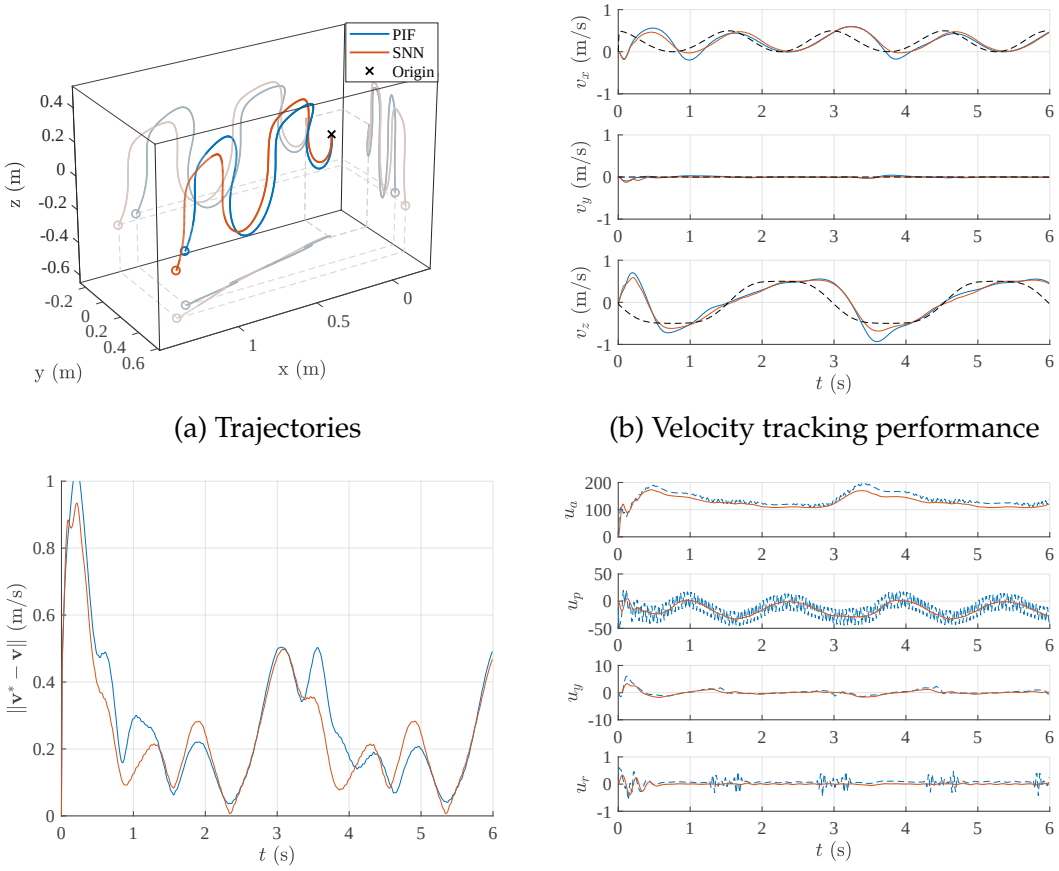


(d) Comparison between PIF and SNN control signals

Figure 3.13: Closed-loop response of the RoboBee performing three maneuvers in quick succession. For the first two seconds, the robot is commanded to climb vertically with $v^* = 0.5\text{m/s}$ and $\gamma^* = 90°$. From $t = 2\text{s}$ to $t = 4\text{s}$, it is commanded to fly forward with $v^* = 0.5\text{m/s}$, and $\gamma^* = 0$. From $t = 4\text{s}$ to $t = 6\text{s}$, it is commanded to fly vertically downward with $v^* = 0.5\text{m/s}$ and $\gamma^* = -90°$. Both controllers again perform very similarly, and both maintain small velocity tracking errors after a short transient response following each command input change.

To demonstrate another example of a changing command input, trial 6 (Fig. 3.14) shows the closed-loop performance of the SNN-controlled robot with a constant commanded speed of $v^* = 0.5\text{m/s}$ and a sinusoidal climb angle following $\gamma^* = (\pi/2)\sin(2\pi t/3)$. The SNN-controlled robot executes the maneuver with very similar tracking error to the PIF-controlled robot, as seen in Fig. 3.14b.

Again, the control inputs are also similar with the exception of the additional low-pass filteing from the SNN synapses.



(a) Trajectories

(b) Velocity tracking performance

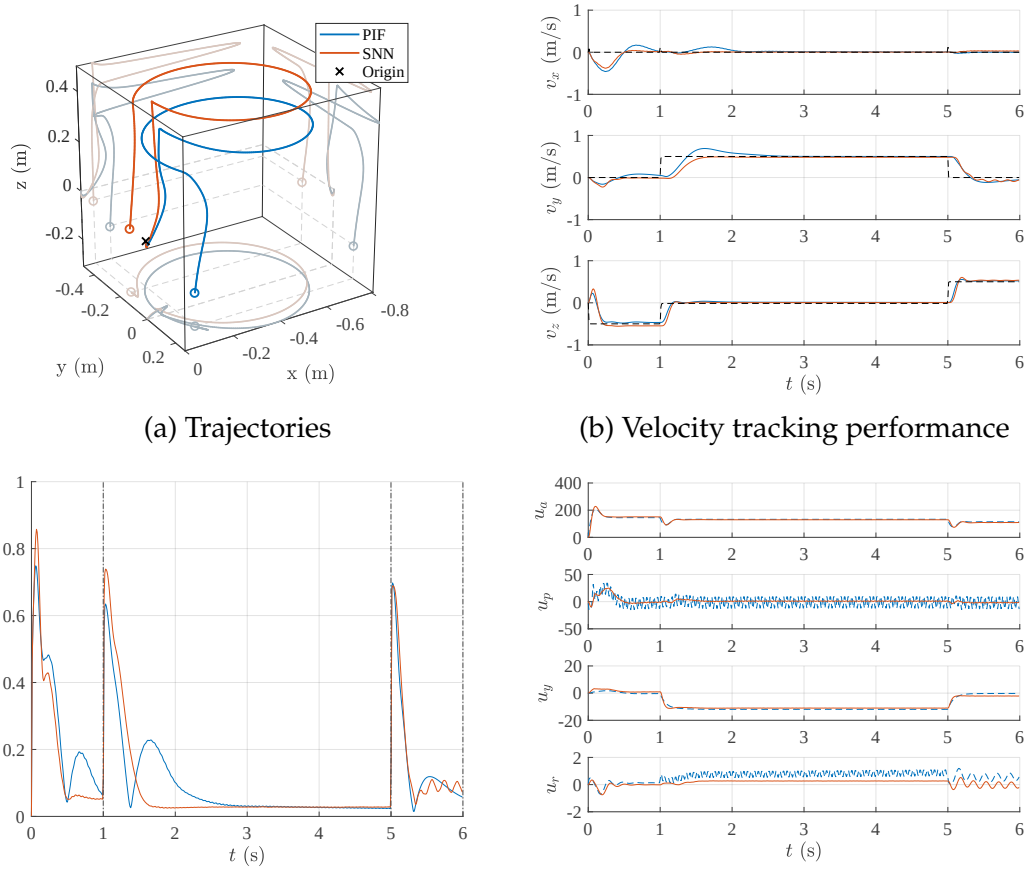
(c) Norm of velocity tracking error.

(d) Comparison between PIF and SNN control signals

Figure 3.14: Closed-loop response of the RoboBee performing a maneuver with a time-varying sinusoidal path angle, $v^* = 0.5\text{m/s}$ and $\gamma^* = (\pi/2)\sin(2\pi t/3)$. Both controllers are able to perform the maneuver, and again have similar control inputs throughout the simulation. Neither controller is able to perfectly follow the target velocity however, and the tracking error in this trial never reaches a value as small as the other trials.

Trial 7 (Fig. 3.15) simulates a flight pattern that might occur during in a sensing application. In this trial, the robot is commanded to take-off vertically for 1 second with $v^* = 0.5\text{m/s}$ and $\gamma^* = 90°$, after which it executes a steady-level turn with $v^* = 0.5\text{m/s}$, $\dot{\xi} = 90°\text{s}^{-1}$, and $\beta = 90°$. Due to the sideslip angle in

this maneuver, the robot remains facing towards the center of the circle while executing the turn. This would allow, for example, a front-facing camera on the RoboBee to collect information from multiple perspectives on a target located at the center of the circle. After completing the circle, the robot is commanded to land again with $v^* = 0.5\text{m/s}$ and $\gamma^* = -90°$. As in the other trials, the SNN maintains similar velocity tracking performance to the PIF, and successfully executes the maneuver.



(a) Trajectories

(b) Velocity tracking performance

(c) Norm of velocity tracking error.

(d) Comparison between PIF and SNN control signals

Figure 3.15: Closed-loop response of the RoboBee circling a target while maintaining a heading such that the RoboBee is pointed towards the center of the turn at all times. Both controllers again perform very similarly, and both maintain small velocity tracking errors.

3.6 SNN-based Control Conclusions

Control using SNNs has several potential benefits, including the potential for low latency control loops and low power consumption. This is especially true when considering integrating SNN controllers with event-based sensors. This paper presents an SNN-based control algorithm that is shown to be capable of stabilizing an accurate model of an unstable insect-scale flapping robot despite unknown parameter variations. One term in the control law approximates the control signal from a linear controller designed for the idealized model, while another term adapts online to account for parameter variations that are unknown *a-priori*. The controller is able to adapt to the parameter variations within 3 seconds and successfully stabilizes hovering flight. This demonstrates the ability of SNN controllers to stabilize inherently unstable systems that require rapid, accurate feedback control.

CHAPTER 4

EVENT-BASED VISUAL PERCEPTION

4.1 Introduction

Significant recent developments in manufacturing, processing capabilities, and sensor design form the foundation for a future in which robots autonomously assist human efforts in remote and hazardous environments. A necessary component of this vision is the development of algorithms which can interpret the environment in real time, allowing robots to autonomously navigate an unknown space while avoiding obstacles and gathering information on targets in the environment. Such algorithms, together with the sensors they are paired with, should be computationally efficient to enable their use in real time on power-constrained mobile platforms.

Many algorithms have been developed to enable obstacle avoidance and target tracking with conventional frame-based cameras. These include methods for localization and mapping [94,112], obstacle avoidance techniques based on the perceived motion of objects in the scene [49,76,125], and target tracking methods based on deep neural networks [50,72]. These techniques have been successfully demonstrated in a variety of applications, but are fundamentally limited by the large amount of redundant data which they must process from conventional frame-based cameras. For instance, to enable rapid and agile flight, many birds and insects can perceive visual changes at up to 150Hz or more [11,56,97]. However, many frame-based perception algorithms are limited to operating closer to 30Hz due to the large amount of redundant data which must be processed from each subsequent frame from a conventional camera.

Recently, event-based sensors have been developed which eliminate the transmission of redundant data by measuring and transmitting only changes in an underlying signal as opposed to absolute values.

Event-based, or neuromorphic sensors encompasses many different classes of sensors, all of which are inspired by biological nervous systems. Generally, event-based sensors communicate information through events that indicate the change in a measured value, such as the light intensity in a vision sensor [12, 25, 66, 83, 121] or an acoustic signal [67] for a spatial audition sensor. These sensors reduce the redundancy in measured and transmitted data by signaling only changes in the measured value, typically in an asynchronous fashion and at a far lower power requirement than their more traditional counterparts.

Event-based vision sensors are particularly interesting for many robotics applications due to their low latency, low power requirements, and high temporal resolution. The earliest event-based vision sensor was demonstrated in 1994 [70]. Modern sensors have relatively low spatial resolution compared to frame-based cameras, but have temporal resolution on the order of 1μs with latency on the order of 1ms or less [66, 83]. Neuromorphic sensors are a natural fit for many robotics applications due to their high temporal resolution, low power requirements, and ability to easily extract motion from an image, but have only recently become commercially available. Some examples demonstrating their potential in robotics applications include tracking a moving ball on a field for a robotic "goalie" arm [26], balancing a pencil using event-based vision sensors to track its position [23], and several examples of obstacle avoidance and target tracking for ground-based robots [73, 74] and aerial vehicles with constrained motion [78]. Recently, event-based cameras have been used for visual odometry

and state estimation on quadcopters in static environments [13, 85, 108].

One group of methods of particular interest for the development of obstacle avoidance and target tracking is optical flow, in which the velocity of points in the image plane is to be estimated. Several event-based optical flow methods have been developed to-date, including sparse methods which track the motion of a few key feature points [48] and dense methods which estimate the flow over the entire field of view. Several algorithms exist to compute optical flow from the output of a neuromorphic vision sensor, which can in turn be used together with more traditional target tracking and obstacle avoidance methods. Gradient-based methods are capable of computing optical flow in real time on embedded systems [6,7,22], but compute only the component of the flow velocity normal to an edge. Other methods track contours or other salient points over time, but require more complex algorithms, usually involve solving optimization problems online, and thus cannot generally be run in real time on embedded hardware [4, 5, 85]. Another recent development does not compute optical flow directly, but is capable of tracking lines in the scene over time, however it is limited to straight lines with no rotational motion in the image plane [37].

Another visual perception task which is critical for mapping and obstacle avoidance is depth estimation, where the distances of points in the field of view from the camera are to be estimated. Depth estimates can be computed from stereo vision, where two cameras with overlapping fields of view are used to observe points simultaneously from different perspectives, or from multi-view stereo, where a single moving camera is used. In some mobile robotics applications, such as the flapping-wing robots described in Chapter 2, the multiple cameras required by stereo vision techniques are in conflict with the extreme

power and weight limits of the platform. The majority of the work in event-based multi-view stereo is limited to estimating the depth of a set of sparse features in the field of view [84]. It is also possible to compute depth from optical flow measurements, but this approach has so far only been demonstrated for estimating the distance of objects away from the direction of travel [125].

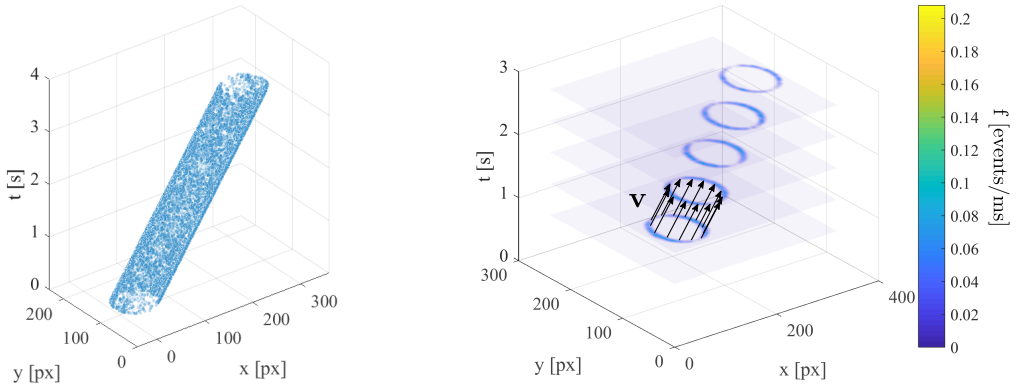
Target tracking from a moving platform and obstacle avoidance using neuromorphic vision sensors has only been explored in a limited fashion to-date. Target tracking can be performed relatively simply with a stationary camera, as the camera naturally detects only moving objects [26, 59]. In the presence of camera motion, the problem becomes more difficult and has only been demonstrated on ground robots and robotic arms using methods which are not directly extensible to aerial vehicles [73, 74, 106].

Here, computationally-efficient methods visual perception methods for autonomous obstacle avoidance and target tracking with event cameras are developed. First, a an approach for event-based dense optical flow estimation is developed, which demonstrates improved accuracy over existing methods while being implementable entirely through efficient linear convolutions. The method is then extended to compute dense monocular depth estimates from a moving camera. In contrast to existing methods, the depth estimates computed by this approach remain accurate near the direction of travel, and are thus directly usable for real time obstacle avoidance in autonomous aerial vehicles. The monocular depth estimates are then used directly to detect moving targets in the field of view of a translating and rotating event camera on an aerial vehicle. In total, the methods presented here represent a computationally-efficient framework for real-time obstacle avoidance and target tracking with autonomous aerial ve-

hicles using a single event-based camera.

4.2 Firing Rate-based Optical Flow

Event-based sensors have much higher temporal resolution and sensing rate than traditional frame-based cameras, providing the potential for accurate optical flow estimation without the need to track motion over large distances between frames. However, because event-based cameras do not measure the absolute brightness of the image, traditional optical flow algorithms cannot be used directly with event-based sensors. Here a new event-based optical flow method is developed using information about the spatio-temporal firing rate of events. The results are compared with existing methods, showing that the proposed *event-density method* is computationally efficient, allowing for real time operation, and offers improved accuracy despite being developed using simple linearized assumptions.



(a) (b)

Figure 4.1: Events (a) generated when a horizontally translating circle crosses the camera FOV are used to compute the spatio-temporal firing rate (b). The ground truth optical flow $\mathbf{v}$ follows structures in the firing rate.

An important assumption which allows for the measurement of optical flow in a conventional camera is that the total derivative of brightness I in the image with respect to time is equal to zero,

$$[\nabla_{\boldsymbol{\eta}} I(x, y, t)]^T \mathbf{v}(x, y, t) = 0 \tag{4.1}$$

where $\boldsymbol{\eta} = [x\ y\ t]^T$ and

$$\mathbf{v}(\boldsymbol{\eta}) = \begin{bmatrix} v_x(\boldsymbol{\eta}) & v_y(\boldsymbol{\eta}) & 1 \end{bmatrix}^T \tag{4.2}$$

This implies that the motion of $\mathbf{p}$ must lie within a level set of brightness in the image.

Although this assumption is commonly used for computer vision tasks with conventional cameras, event-based cameras do not directly measure the brightness, and reconstructing it from the events the camera produces is a complex task and computationally-intensive task in and of itself [4]. Neuromorphic cameras generate sequences of events $\mathbf{e}_i = [\boldsymbol{\eta}_i^T,\ p_i]^T$ due to changes in brightness in the scene. Events are generated when the log of the intensity I at a given pixel location changes beyond a fixed threshold θ. Each event comprises the $x_i \in \mathcal{X} \subset \mathbb{R}$ and $y_i \in \mathcal{Y} \subset \mathbb{R}$ pixel coordinates of the event, the time $t_i \in \mathcal{T} \subset \mathbb{R}^+$ of the event, and the polarity $p_i \in \{-1, 1\}$ of the event. The polarity of the event indicates the sign of the intensity change:

$$p_i = \begin{cases} 1 & \text{if } \ln[I(x_i, y_i, t_i)] - \ln[I(x_i, y_i, t_{i-1})] \geq \theta \\ -1 & \text{if } \ln[I(x_i, y_i, t_i)] - \ln[I(x_i, y_i, t_{i-1})] \leq -\theta \end{cases} \tag{4.3}$$

The firing rate, approximately the number of events in a given period of time, can be defined in terms of the log brightness $I' \triangleq \ln(I)$ as

$$f(\boldsymbol{\eta}) = \frac{1}{\theta} \frac{\partial I'}{\partial t} \tag{4.4}$$

In order to construct an analogous constraint to (4.1) which applies to neuromorphic cameras, some vector $\mathbf{n} \in \mathbb{R}^3$ must be found which is written in terms of f instead of I, such that

$$[\mathbf{n}(\boldsymbol{\eta})]^T \mathbf{v}(\boldsymbol{\eta}) = 0 \tag{4.5}$$

It will be shown that a definition for $\mathbf{n}$ which satisfies (4.5) is

$$\mathbf{n} \triangleq \nabla_{\boldsymbol{\eta}} f \tag{4.6}$$

Using this definition in (4.5) yields a relationship between the partial derivatives of f and the motion field:

$$\frac{\partial f}{\partial x} v_x + \frac{\partial f}{\partial y} v_y = -\frac{\partial f}{\partial t} \tag{4.7}$$

Under the assumption of constant brightness, the firing rate can be expressed in terms of the spatial derivatives of brightness which, together with (4.4), can be substituted into (4.7) to prove that (4.6) satisfies (4.5). It is possible to write the constant brightness assumption in terms of the log of intensity as

$$[\nabla_{\boldsymbol{\eta}} I'(\boldsymbol{\eta})]^T \boldsymbol{\xi}(\boldsymbol{\eta}) = 0 \tag{4.8}$$

Together with (4.4), this yields

$$\frac{\partial I'}{\partial x} v_x + \frac{\partial I'}{\partial y} v_y + f\theta = 0 \tag{4.9}$$

Solving for f yields

$$f = -\frac{1}{\theta}\left(\frac{\partial I'}{\partial x} v_x + \frac{\partial I'}{\partial y} v_y\right) \tag{4.10}$$

Substituting (4.4) into the left-hand side of (4.7) and (4.10) into the right-hand side yields

$$\frac{\partial}{\partial x}\left(\frac{1}{\theta}\frac{\partial I'}{\partial t}\right) v_x + \frac{\partial}{\partial y}\left(\frac{1}{\theta}\frac{\partial I'}{\partial t}\right) v_y = \frac{\partial}{\partial t}\left[\frac{1}{\theta}\left(\frac{\partial I'}{\partial x} v_x + \frac{\partial I'}{\partial y} v_y\right)\right] \tag{4.11}$$

Given the assumption that $\partial/\partial t(v_x) = \partial/\partial t(v_y) = 0$:

$$\frac{1}{\theta}\frac{\partial}{\partial x}\left(\frac{\partial I'}{\partial t}\right)v_x + \frac{1}{\theta}\frac{\partial}{\partial y}\left(\frac{\partial I'}{\partial t}\right)v_y = \frac{1}{\theta}\frac{\partial}{\partial t}\left(\frac{\partial I'}{\partial x}v_x + \frac{\partial I'}{\partial y}v_y\right) \tag{4.12}$$

$$\frac{\partial}{\partial x}\left(\frac{\partial I'}{\partial t}\right)v_x + \frac{\partial}{\partial y}\left(\frac{\partial I'}{\partial t}\right)v_y = \frac{\partial}{\partial t}\left(\frac{\partial I'}{\partial x}\right)v_x + \frac{\partial}{\partial t}\left(\frac{\partial I'}{\partial y}\right)v_y \tag{4.13}$$

$$\frac{\partial^2 I'}{\partial x \partial t}v_x + \frac{\partial^2 I'}{\partial y \partial t}v_y = \frac{\partial^2 I'}{\partial t \partial x}v_x + \frac{\partial^2 I'}{\partial t \partial y}v_y \tag{4.14}$$

Applying symmetry of second derivatives yields the final result,

$$\frac{\partial^2 I'}{\partial x \partial t}v_x + \frac{\partial^2 I'}{\partial y \partial t}v_y = \frac{\partial^2 I'}{\partial x \partial t}v_x + \frac{\partial^2 I'}{\partial y \partial t}v_y \tag{4.15}$$

which shows that the proposed definition for $\mathbf{n}$ given by (4.6) satisfies the constraint (4.5) under the stated assumptions.

Because (4.5) is a single scalar equation for each coordinate $\boldsymbol{\eta}$, an additional assumption is required to compute both v_x and v_y. A second assumption, which will be shown to work well in practice, is that the orientation $\psi = \text{atan}(f_y/f_x)$ of the firing rate's spatial gradient remains constant along the trajectory of each moving point. These two assumptions are used to find the direction of the velocity at each point $\boldsymbol{\eta}$ in the spatio-temporal volume by solving

$$\mathbf{v}(\boldsymbol{\eta}) = \arg\min_{\tilde{\mathbf{v}}} \left\{ \left[\psi(\boldsymbol{\eta} + \tilde{\mathbf{v}}dt) - \psi(\boldsymbol{\eta})\right]^2 + \left[f(\boldsymbol{\eta} + \tilde{\mathbf{v}}dt) - f(\boldsymbol{\eta})\right]^2 \right\} \tag{4.16}$$

The expression above can be significantly simplified by linearizing both ψ and f about $\boldsymbol{\eta}$. The linearization will be accurate for a sufficiently small dt, which is possible due to the high sensing rate of event cameras. Linearizing (4.16) yields

$$\mathbf{v}(\boldsymbol{\eta}) = \arg\min_{\tilde{\mathbf{v}}} \left\{ \left[\left(\nabla_{\boldsymbol{\eta}}\psi(\boldsymbol{\eta})\right)^T \tilde{\mathbf{v}}dt \right]^2 + \left[\left(\nabla_{\boldsymbol{\eta}} f(\boldsymbol{\eta})\right)^T \tilde{\mathbf{v}}dt \right]^2 \right\} \tag{4.17}$$

If $\nabla_{\boldsymbol{\eta}} f = \mathbf{0}$, $\nabla_{\boldsymbol{\eta}}\psi = \mathbf{0}$, or if the gradients are parallel, there are infinitely many solutions for $\mathbf{v}$. Otherwise, the solution to the minimization is any vector which lies in the null space of both $\nabla_{\boldsymbol{\eta}} f$ and $\nabla_{\boldsymbol{\eta}}\psi$,

$$\hat{\mathbf{v}}(\boldsymbol{\eta}) = \frac{(\nabla_{\boldsymbol{\eta}}\psi(\boldsymbol{\eta})) \times (\nabla_{\boldsymbol{\eta}} f(\boldsymbol{\eta}))}{\|(\nabla_{\boldsymbol{\eta}}\psi(\boldsymbol{\eta})) \times (\nabla_{\boldsymbol{\eta}} f(\boldsymbol{\eta}))\|} \tag{4.18}$$

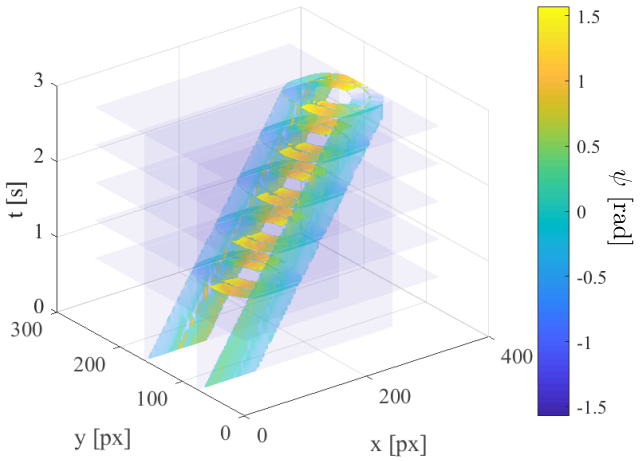


Figure 4.2: The orientation of the firing rate gradient, corresponding to the example shown in Fig. 4.1.

where $\times$ denotes the vector cross product. From (4.2), the magnitude of $\mathbf{v}$ should be set such that the third component is equal to unity. Thus, the velocity at each point in the volume is equal to the unit directional vector $\hat{\mathbf{v}}$ normalized by its third component $\hat{\mathbf{v}}_3$:

$$\mathbf{v}(\boldsymbol{\eta}) = \frac{\hat{\mathbf{v}}(\boldsymbol{\eta})}{\hat{\mathbf{v}}_3(\boldsymbol{\eta})} \tag{4.19}$$

4.3 Event-based Dense Monocular Depth Estimation

In robotics applications, the velocity and angular rate of the camera are typically known from other on-board sensors and estimation techniques. In this case, the gradient-based dense optical flow method presented in Section 4.2 can be adapted to provide dense depth estimates of static objects in an environment from monocular camera. Because the depth estimation method presented here is based on the same linear assumptions as the optical flow method, it is computationally simple, entirely implementable through convolutions in real time.

The camera used for depth estimation is modeled using the pin-hole camera model as shown in Fig. 4.3. Every stationary point Q in the camera field of view (FOV) is projected onto the image plane at a corresponding point P. The position of Q relative to the camera origin C expressed in the camera frame $\mathcal{F}_c$ is $\mathbf{q} = [q_x \; q_y \; q_z]^T$. The position of P is written in terms of $\mathbf{q}$ as,

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} \frac{\lambda}{q_z} q_x \\ \frac{\lambda}{q_z} q_y \\ \lambda \end{bmatrix} \tag{4.20}$$

where $\lambda \in \mathbb{R}^+$ is the camera focal length. Any relative motion between the camera and Q causes the position of P to change according to,

$$\dot{\mathbf{p}} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} \frac{\lambda}{q_z}\dot{q}_x - \frac{\lambda}{q_z^2}\dot{q}_z q_x \\ \frac{\lambda}{q_z}\dot{q}_y - \frac{\lambda}{q_z^2}\dot{q}_z q_y \\ 0 \end{bmatrix} \tag{4.21}$$

where $\dot{(\cdot)}$ denotes the time derivative. The velocity of P depends on $\dot{\mathbf{q}}$, which expressed in terms of the camera velocity $\mathbf{v}_c$ and angular rate $\boldsymbol{\omega}_c$ is,

$$\dot{\mathbf{q}} = \mathbf{v}_c - \boldsymbol{\omega} \times \mathbf{q} \tag{4.22}$$

where $\times$ denotes a cross product. Evaluating equation (4.22) and substituting into (4.21) yields the final expression for the velocity of the projected point in the image plane,

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \frac{1}{q_z}\mathbf{G}\mathbf{v}_c + \mathbf{H}\boldsymbol{\omega}_c \tag{4.23}$$

where the matrices $\mathbf{G}$ and $\mathbf{H}$ depend exclusively on intrinsic camera parameters:

$$\mathbf{G} \triangleq \begin{bmatrix} \lambda & 0 & -p_x \\ 0 & \lambda & -p_y \end{bmatrix}, \qquad \mathbf{H} \triangleq \begin{bmatrix} \frac{p_x p_y}{\lambda} & -(\lambda + \frac{p_x^2}{\lambda}) & p_y \\ (\lambda + \frac{p_y^2}{\lambda}) & -\frac{p_x p_y}{\lambda} & -p_x \end{bmatrix} \tag{4.24}$$

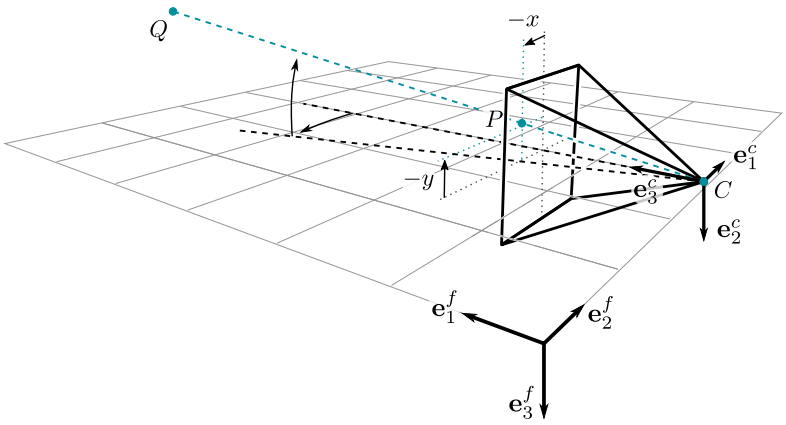


Figure 4.3: The pinhole camera model

This expression relates the depth of objects in the scene to the camera velocity and the optical flow (v_x and v_y) on the camera image plane.

The assumption that flow is oriented along level sets of the firing rate f, used previously in the development of optical flow, can now be used together with (4.23) to estimate the depth of stationary points in the scene. The constant firing rate constraint is written as,

$$\begin{bmatrix} f_x & f_y \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = -f_t \tag{4.25}$$

where the subscript on f denotes a partial derivative, e.g. $f_x = \partial f / \partial x$. Substituting the expressions for the image plane velocities (4.23) into (4.25) yields

$$\frac{1}{q_z} \begin{bmatrix} f_x & f_y \end{bmatrix} \mathbf{G}\mathbf{v}_c + \begin{bmatrix} f_x & f_y \end{bmatrix} \mathbf{H}\boldsymbol{\omega}_c = -f_t \tag{4.26}$$

Solving (4.26) for depth yields:

$$q_z = -\frac{\begin{bmatrix} f_x & f_y \end{bmatrix} \mathbf{G}\mathbf{v}_c}{f_t + \begin{bmatrix} f_x & f_y \end{bmatrix} \mathbf{H}\boldsymbol{\omega}_c} \tag{4.27}$$

4.4 Event-based Independent Motion Detection

To track moving targets through an environment, it is necessary to first detect the targets using a method which either detects patterns of colors and shapes via object detection or instead detects patterns of movement through the environment. Here, a method is developed which detects any motion relative to the environment in FOV of a rotating and translating event camera.

As with the depth estimation method described in Section 4.3, the camera is modeled using the pin-hole camera model shown in Fig. 4.3 and it is assumed that the state of the camera is known. First, the method described in Section 4.3 is used to estimate the depth q_z associated with each event $\boldsymbol{\eta_i} = [x_i,\ y_i,\ t_i]$, where x_i and y_i are the image-plane coordinates of the event. Using the estimated depth, the position $\mathbf{q}_i$ of the point Q in the world frame corresponding to the point $\boldsymbol{\eta_i}$ in the image plane is computed as,

$$\mathbf{q}_i = w(x_i, y_i, t_i) = \mathbf{R} \begin{bmatrix} x_i \frac{q_z(x_i, y_i, t_i)}{\lambda} \\ y_i \frac{q_z(x_i, y_i, t_i)}{\lambda} \\ q_z(x_i, y_i, t_i) \end{bmatrix} + \mathbf{r}_c(t_i) \tag{4.28}$$

where $\mathbf{R} \in \mathrm{SO}(3)$ is a rotation matrix from the camera frame to the world frame and $\mathbf{r}_c$ is the position of the camera expressed in the world frame. The density ρ of points in the world frame is then computed by convolving a Gaussian kernel with Dirac delta functions δ. The Dirac delta functions are located at the points $\mathbf{q}_i$ corresponding to events which occurred within a given time window defined by the parameters $\tau_2 > \tau_1 > 0$,

$$\rho(\mathbf{q}, t) = \sum_{\mathbf{q}_i \in Q_1} \exp[-(\mathbf{q} - \mathbf{q}_i)^T \boldsymbol{\Sigma}^{-1} (\mathbf{q} - \mathbf{q}_i)] \tag{4.29}$$

where,

$$\mathcal{Q}_1 = \{\mathbf{q}_i = w(x_i, y_i, t_i) \mid t_i \in [t - \tau_2,\ t - \tau_1)\} \tag{4.30}$$

the diagonal matrix $\mathbf{\Sigma} = \mathbf{I}_3\sigma_k$, $\mathbf{I}_3 \in \mathbb{R}^{3\times3}$ is the identity matrix, and $\sigma_k > 0$ defines the width of the kernel. In practice, the event density can be efficiently approximated by computing the sum in (4.29) only for the subset of points $\mathbf{q}_i$ which lie within a distance $2\sqrt{\sigma_k}$ of $\mathbf{q}$. This subset can be found efficiently using a k-d tree [8].

Stationary objects in the camera FOV are located in regions of high event density ρ, because the world-frame point estimates $\mathbf{q}_i$ associated with points on the object will accumulate in the same location over time, distributed according to the estimation error. World-frame point estimates associated with points on moving objects however, will follow the motion of the object throughout the world, and thus do not accumulate at a single world-frame location over time. Therefore, point estimates $\mathbf{q}_i$ which correspond to points on moving objects generally exist in regions of low world-frame event density and high values of the inverse density function h,

$$h(x_i, y_i, t) = \exp[-\rho(\mathbf{q}_i, t)/\alpha], \qquad \forall \mathbf{q}_i \in \mathcal{Q}_2 \tag{4.31}$$

where the parameter α controls the smoothness of the function and

$$\mathcal{Q}_2 = \{\mathbf{q}_i = w(x_i, y_i, t_i) \mid t_i \in [t - \tau_1,\ t]\} \tag{4.32}$$

4.5 Optical Flow Results

To demonstrate the ability of the firing rate-based optical flow method to compute both components of the optical flow vector accurately, the method is

demonstrated on an example with a diagonally-translating square in Fig. 4.4. While many existing event-based optical flow methods can only compute the normal component of the optical flow [90], the firing rate-based method presented here computes both components of the flow around the entire perimeter of the square. The average angular error (AAE) $\tilde{\phi}$ is a common metric used to evaluate the accuracy of optical flow methods. It measures the average angular difference between the computed optical flow vectors $(v_x,\ v_y)$ and the ground truth optical flow $(v_x^*,\ v_y^*)$ at all points where the computed optical flow is non-zero:

$$\tilde{\phi} = \cos^{-1}\left[\frac{v_x v_x^* + v_y v_y^*}{\sqrt{v_x^2 + v_y^2}\sqrt{(v_x^*)^2 + (v_y^*)^2}}\right] \tag{4.33}$$

By only computing the component of optical flow which is locally normal to any of the translating edges of the square, existing methods explored in [90] effectively have an AAE of approximately 45°. By contrast, the AAE of the optical flow computed for the translating square with firing rate-based method presented here is only 14.2°.

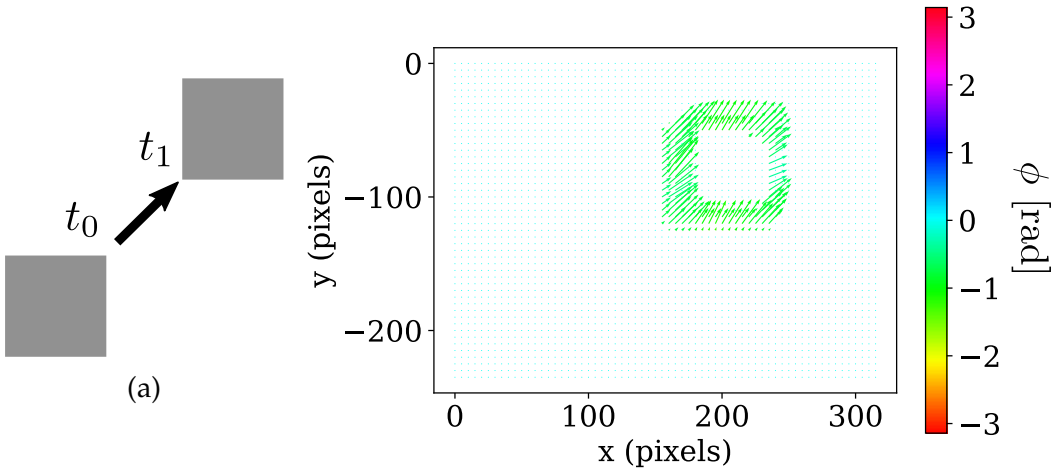


Figure 4.4: The firing rate-based optical flow method computes the optical flow vectors (b) along the entire edge of a diagonally-translating block (a) with an AAE of only 14.2°.

Although it is useful to compare the firing rate-based optical flow method with existing event-based optical flow algorithms, it is also useful to compare the performance of the method presented here with commonly used frame-based optical flow methods used in the majority of computer vision and robotics applications today. The Farneback algorithm [38] is an algorithm for dense optical flow estimation which is still widely in use today. The Farneback algorithm is compared to the firing rate-based method developed here for the case of the three diagonally translating shapes shown in Fig. 4.5. As reported in Table 4.1, the firing rate-based method more than halves the AAE of the Farneback algorithm despite running four times faster in a single-threaded implementation on an Intel® Xeon® E5-2623 3GHz CPU. Both algorithms have very low error rates on the translating sphere shown in Fig. 4.5. However, the firing rate-based method does not compute flow at two extreme edges of the sphere due to the

	Farneback	Firing Rate-based Flow
AAE [degrees]	38.7	14.9
Approximate operating frequency [Hz]	30	120

Table 4.1: The proposed firing rate-based optical flow method presented here is approximately four times faster than the commonly used Farneback algorithm and has less than half of the angular error.

low number of events in those locations. The firing rate-based method however is able to compute the flow along the straight edges of the translating cubes more accurately than the Farneback algorithm, which leads to its overall lower error rate.

4.6 Depth Estimation Results

The depth estimation method presented here is used to estimate the depth of stationary objects in two different examples, shown in Figs. 4.7 and 4.9. In the first example, a camera views several stationary objects while translating laterally at a constant speed, as shown in Fig. 4.6. As can be seen in Fig. 4.7, the largest errors are observed at the top of the blue sphere. This is due to the lack of events in that region, which cause a poor depth estimate. The depth error is computed as a percentage difference between the estimated depth q_z and the ground truth depth d:

$$\tilde{d} = \left| \frac{q_z - d}{d} \right| \tag{4.34}$$

The depth estimate elsewhere in the field of view remains near 5%, with a median depth error of 5.9%. This compares favorably with existing monocular event-based depth estimation algorithms, which report median depth errors of 9.4% when viewing the straight walls of a 4m-wide corridor [125].

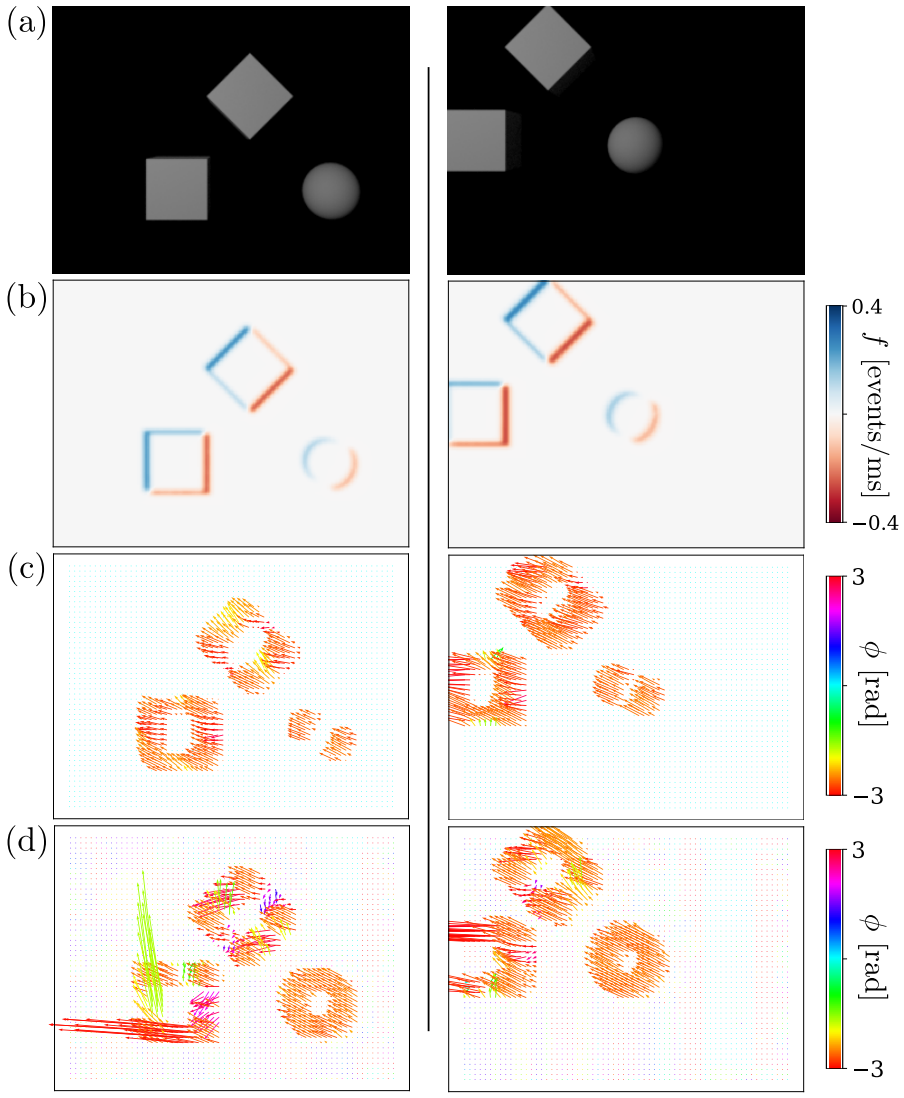


Figure 4.5: The firing rate-based optical flow algorithm developed here is compared with the well-known Farneback algorithm. The scene consists of three shapes translating at an angle through the FOV. The firing rate-based optical flow (c) has a lower average angle error than Farneback (d), especially on the flat edges of the cubes, but does not compute flow at two edges of the sphere due to a low firing rate in that region (b).

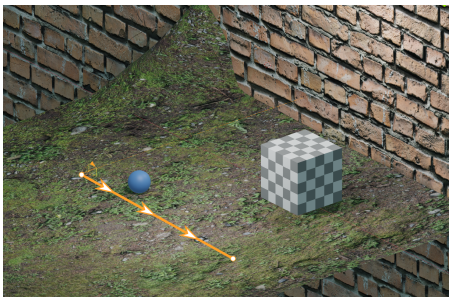


Figure 4.6: The depth estimation algorithm is applied to the data taken from the FOV of a laterally translating camera (shown in yellow), as it observes an otherwise static scene.

The depth estimation method presented here is also examined on a more complex example in which a camera flies forwards and laterally through a set of obstacles in Fig. 4.9, with the scene and camera trajectory shown in Fig. 4.8. The depth estimation error is higher in this example, primarily due to the large distance to the background of the image, which is over 35m away in places. The error shown in Fig. 4.9 is high for the sky, because the true depth is infinite in that region. Overall, the median depth error in this example is 9.3%.

This example demonstrates the ability of the proposed algorithm to estimate scene depth near the focus of expansion (FOE). The FOE is the point in the image plane where all ground truth optical flow vectors due to camera translational motion intersect, and corresponds to direction of travel projected into the image plane. The FOE is a singularity where no translational optical flow exists. Near the FOE, the signal-to-noise ratio of most algorithms becomes too low to be of practical use in obstacle avoidance. For instance, the monocular depth estimation used in [125] is limited to the sides of the quadcopter traveling forwards down a corridor. With the method presented here, the region near the focus of expansion which results in unusable depth estimates is limited to approximately 25° of the horizontal FOV.

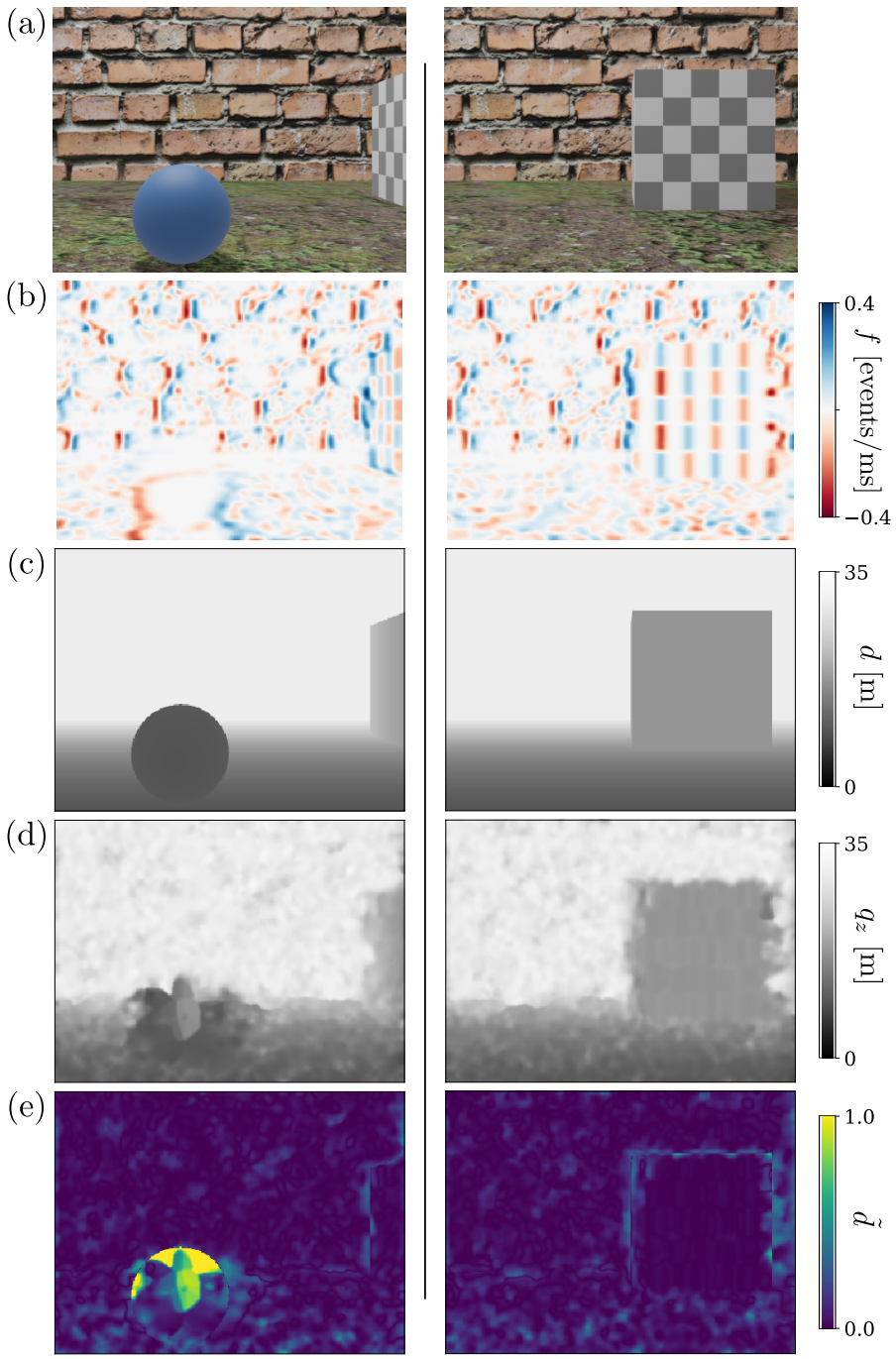


Figure 4.7: The depth estimation algorithm is applied to the data taken from the FOV of a laterally translating camera (a). As shown by the depth estimation error (e), the depth estimate (d) is generally within 5% of the ground truth depth (c), except near the top of the blue sphere, where a lack of contrast yields insufficient information to obtain any estimate in that region, as seen by the correspondingly low firing rate value (b).

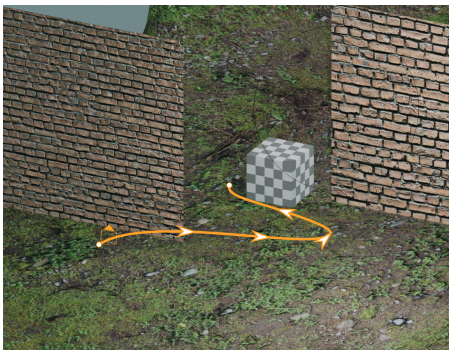


Figure 4.8: The depth estimation algorithm is applied to the data taken from the FOV of a longitudinally and laterally translating camera (shown in yellow), as it observes an otherwise static scene.

4.7 Independent Motion Detection Results

The independent motion detection method presented here is used to detect a bouncing ball in the field of view of a translating camera in Fig. 4.10. This task is challenging due to the translational motion of the camera, which creates relative motion between the camera and every point in its FOV, including both stationary and moving objects. By comparing the depth estimates at each time step to the world-frame estimates of points in the FOV, as described in Section 4.4, the inverse world-frame point density function h is obtained. As shown in Fig. 4.10, high values of h correspond to moving objects in the FOV which are moving relative to the fixed world frame.

It is interesting to note that not only the ball in Fig. 4.10 is detected by this method, but also the moving shadows cast by the ball as it moves through the environment. In the first frame shown in the example, the shadow of the ball is visible just below it and to the left against the brick wall. As the ball approaches the ground, the shadow on the ground becomes visible and is detected by the algorithm as well.

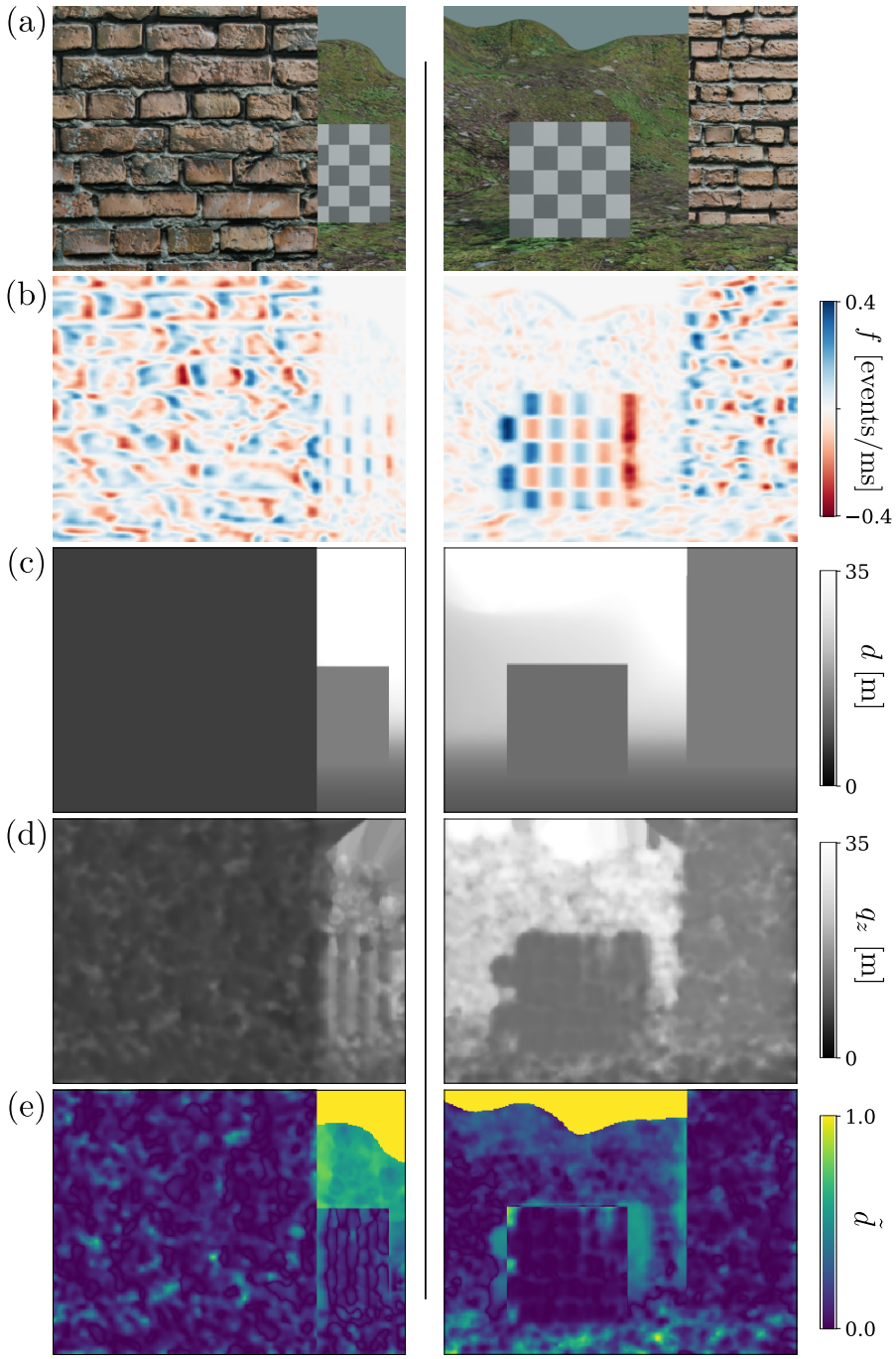


Figure 4.9: The depth estimation algorithm is applied to the data taken from the FOV of camera as it moves forward and laterally through an environment (a). The depth estimation error percentage (e) is high at large distances, such as the sky at the top of the image. For closer objects, the depth estimate (d) is generally within 5-10% of the ground truth (c).

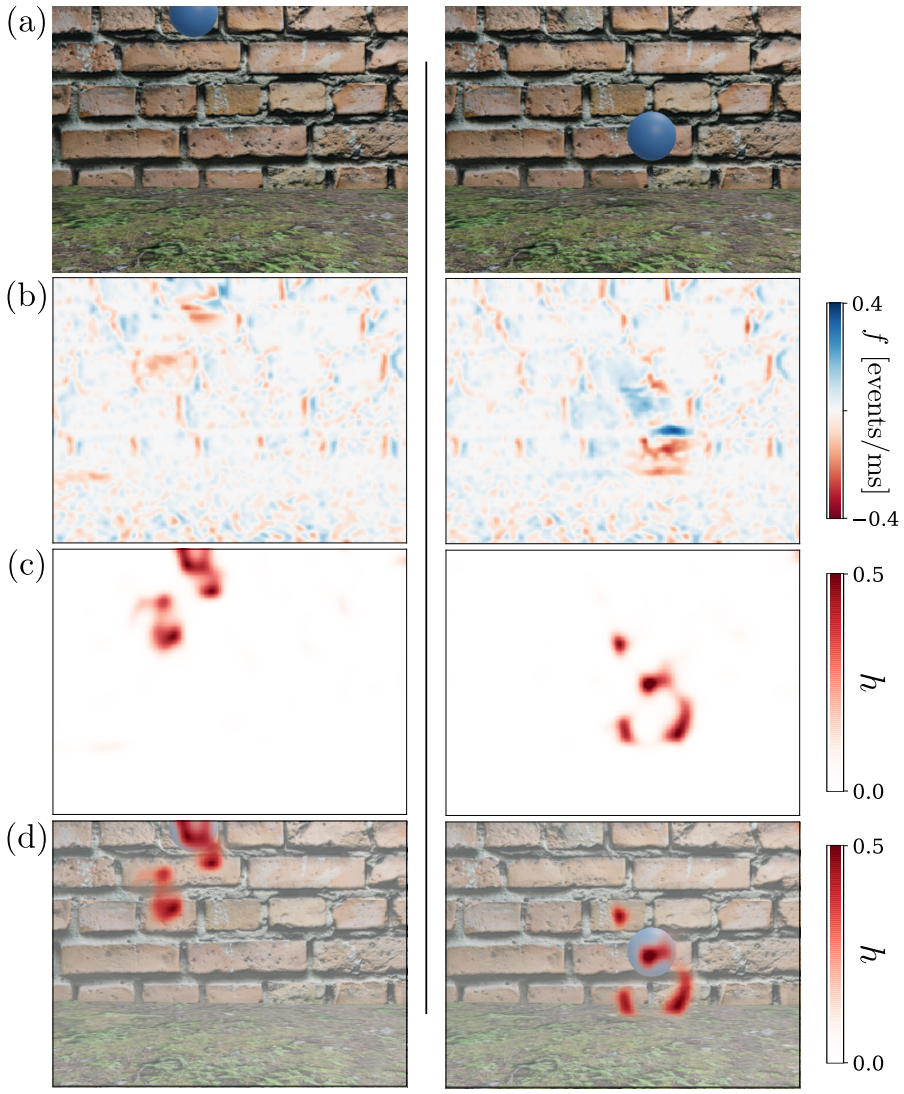


Figure 4.10: The independent motion detection algorithm is used to detect a moving ball in the FOV of a moving camera (a). The firing rate (b) yields depth estimates, which are projected into the world frame to determine the inverse world-frame point density function h (c). High values of h correspond to movement in the world frame. The algorithm detects both the moving ball and its moving shadows (d).

Additionally, the independent motion detection method presented here is used to detect a moving butterfly in the FOV of a translating camera in Fig. 4.11. The difficulty in this particular scenario is that the butterfly has little relative velocity to the camera, as it translates together with the camera in the same direction at the same speed. Despite this complication, the algorithm presented here is nonetheless able to consistently detect the motion of the butterfly, as shown by the high values of h at the location of the butterfly in the FOV in Fig. 4.11.

4.8 Conclusions

A visual perception framework was presented for event-based cameras which takes advantage of the high sensing rate of event cameras to enable computationally efficient, complimentary methods for optical flow estimation, depth estimation, and independent motion detection from a monocular camera. The optical flow method is shown to have reduced error rates compared to commonly used event-based and frame-based dense optical flow methods, with reduced computational cost compared to common frame-based methods. The depth estimation method leverages the assumptions developed for the optical flow algorithm to enable dense monocular depth estimation with applicability to obstacle avoidance and mapping. Finally, a method is developed which uses the depth estimates to detect objects which are moving relative to the world frame from point of view of a translating and rotating event camera. Together, these three methods form a foundation for autonomous obstacle avoidance and target tracking in power- and weight-constrained mobile robotics applications.

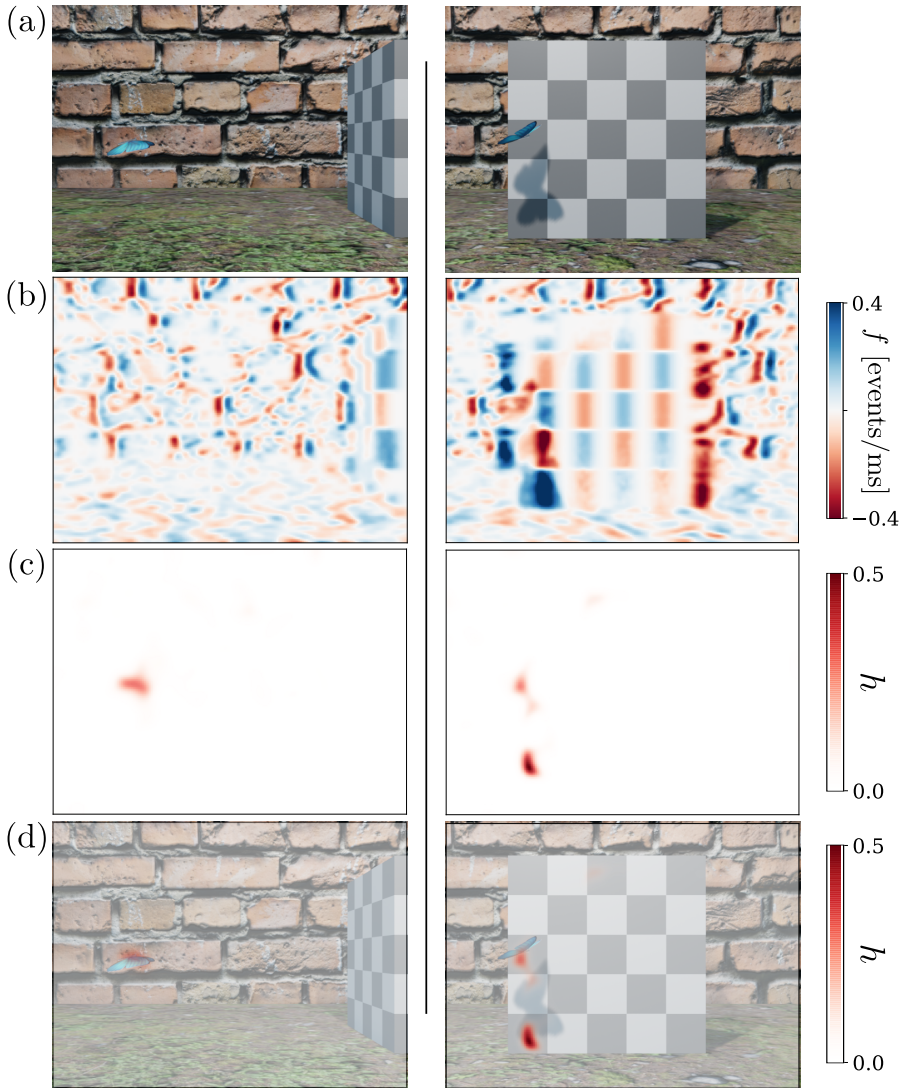


Figure 4.11: The independent motion detection algorithm is used to detect a butterfly in the FOV of a moving camera, which moves with almost zero relative velocity with the camera (a). The firing rate (b) yields depth estimates, which are projected into the world frame to determine the inverse world-frame point density function h (c). High values of h correspond to movement in the world frame. The algorithm detects the butterfly despite no relative motion between the camera and the butterfly (d).

CHAPTER 5

CONCLUSIONS

The potential of flapping-wing robots can only be fully realized through a foundational understanding of flapping-wing flight dynamics and maneuvers, enabling the design of full-envelope flight control. This work presented the derivation of a nonlinear flight dynamics model for minimally-actuated flapping-wing robots which captures the six body degrees of freedom as well as two rotational degrees of freedom in each wing during flight. A simplified yaw control method was proposed which maintains a constant flapping frequency during maneuvers for more power-efficient flight. As a basis for understanding the dominant linear modes of flight, detailed definitions were presented for quasi-steady aerodynamic maneuvers along with a method for solving the equations of motion for quasi-steady set points.

Steady flight envelopes for flapping-wing flight, given in terms of wing stroke angle limits imposed by the robot geometry, were presented using the set point solutions based on the maneuver definitions. The quasi-steady maneuver set points were used to analyze the dominant linear modes and the stability of flight for a specific flapping-wing robot, which revealed stable flight regimes at moderately high speeds of forward level and descending flight. The mode shapes of the two dominant linear modes throughout the flight envelope were shown for both hovering and forward level flight. Finally, the model was validated against experimental data from a physical flapping-wing robot, which showed good agreement between the dominant modes in the model and the experimental data. Finally, experimental results were presented validating the model's predicted coupling between roll and yaw.

This work also developed a framework of computationally-efficient methods for neuromorphic perception and control to enable autonomous obstacle avoidance and target detection in flapping-wing robots and other agile mobile robotic platforms. An adaptive flight control method using spiking neural networks (SNNs) was developed based on the full-envelope flapping-wing flight dynamics model. The controller was demonstrated to be capable of accounting for parametric variations in the model through rapid online adaptation, which resulted in improved stabilizing performance in hovering flight compared with a more conventional linear optimal controller. The control method was also extended to perform quasi-steady maneuvers such as coordinated turns, take-off, and landing throughout the flight envelope. It was demonstrated that it could effectively control these maneuvers with reduced error compared with a gain-scheduled optimal controller.

Finally, computationally-efficient neuromorphic visual perception methods were developed for dense optical flow estimation, dense depth estimation, and independent motion detection. These methods use simple, linearized assumptions about the firing rate of events, which work well due to the high sensing rate of event-based cameras. The firing rate-based optical flow method is shown to have reduced angular error and improved computational performance compared to existing methods. It is shown that the underlying assumptions used for the optical flow method also enable dense depth estimation from a moving event camera. Finally, the depth estimates were used in a new event-based independent motion detection approach which facilitates the detection of moving targets in the FOV of a translating and rotating event camera.

In total, the methods presented here represent a computationally-efficient

framework for real-time obstacle avoidance and target tracking with autonomous aerial vehicles using a single event-based camera.

APPENDIX A

CHAPTER 1 OF APPENDIX

A.1 Kinematic Terms

The position vectors used in the equations of motion are,

$$\mathbf{r}_{GA} = -d\mathbf{e}_3^b \tag{A.1}$$

$$\mathbf{r}_{AL} = l_1\mathbf{e}_2^l + h_1\mathbf{e}_3^l \tag{A.2}$$

$$\mathbf{r}_{AR} = l_1\mathbf{e}_2^r + h_1\mathbf{e}_3^r \tag{A.3}$$

where $d \in \mathbb{R}$, $l_1 \in \mathbb{R}$, and $h_1 \in \mathbb{R}$ are distance parameters that that are determined by the robot geometry. The angular rates of the body, left and right wings with respect to the fixed frame are,

$$\boldsymbol{\omega}_b = \dot{\phi}\mathbf{e}_3^f + \dot{\theta}\mathbf{e}_1^{f'} + \dot{\psi}\mathbf{e}_2^b \tag{A.4}$$

$$\boldsymbol{\omega}_l = \dot{\phi}_l\mathbf{e}_3^b + \dot{\theta}_l\mathbf{e}_1^{l'} + \dot{\psi}_l\mathbf{e}_2^l + \boldsymbol{\omega}_b \tag{A.5}$$

$$\boldsymbol{\omega}_r = \dot{\phi}_r\mathbf{e}_3^b + \dot{\theta}_r\mathbf{e}_1^{r'} + \dot{\psi}_r\mathbf{e}_2^r + \boldsymbol{\omega}_b \tag{A.6}$$

where the basis vectors of the intermediate frames are $\mathbf{e}_1^{f'} = \mathbf{R}(\mathbf{e}_3^f, \phi)\mathbf{e}_1^f$, $\mathbf{e}_1^{l'} = \mathbf{R}(\mathbf{e}_3^b, \phi_l)\mathbf{e}_1^b$, and $\mathbf{e}_1^{r'} = \mathbf{R}(\mathbf{e}_3^b, \phi)\mathbf{e}_1^b$. The rotation matrices, denoted with $\mathbf{R}$, are described in Appendix A.2. The relative velocity vectors are obtained by using the expressions for the position vectors and angular rate vectors for the corresponding rigid bodies:

$$\mathbf{v}_{GA} = \boldsymbol{\omega}_b \times \mathbf{r}_{GA} \tag{A.7}$$

$$\mathbf{v}_{AL} = \boldsymbol{\omega}_l \times \mathbf{r}_{AL} \tag{A.8}$$

$$\mathbf{v}_{AR} = \boldsymbol{\omega}_r \times \mathbf{r}_{AR} \tag{A.9}$$

The acceleration of the body center of gravity G, the left wing center of gravity L, and the right wing center of gravity R are written in terms of (A.1) - (A.9):

$$\mathbf{a}_G = \ddot{x}\mathbf{e}_1^f + \ddot{y}\mathbf{e}_2^f + \ddot{z}\mathbf{e}_3^f \tag{A.10}$$

$$\mathbf{a}_L = \mathbf{a}_G + \dot{\boldsymbol{\omega}}_b \times \mathbf{r}_{GA} + \boldsymbol{\omega}_b \times \mathbf{v}_{GA} + \dot{\boldsymbol{\omega}}_l \times \mathbf{r}_{AL} + \boldsymbol{\omega}_l \times \mathbf{v}_{AL} \tag{A.11}$$

$$\mathbf{a}_R = \mathbf{a}_G + \dot{\boldsymbol{\omega}}_b \times \mathbf{r}_{GA} + \boldsymbol{\omega}_b \times \mathbf{v}_{GA} + \dot{\boldsymbol{\omega}}_r \times \mathbf{r}_{AR} + \boldsymbol{\omega}_r \times \mathbf{v}_{AR} \tag{A.12}$$

A.2 Rotation Matrices

A rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$ can be computed the axis of rotation $\mathbf{n} \in \mathbb{R}^3$ and the angle of rotation $\eta \in \mathbb{R}$, where $\|\mathbf{n}\| = 1$ and η is expressed in radians. The rotation matrix is,

$$\mathbf{R}(\mathbf{n}, \eta) = [1 - \cos(\eta)]\mathbf{n}\mathbf{n}^T + \cos(\eta)\mathbf{I}_3 + \sin(\eta)S(\mathbf{n}) \tag{A.13}$$

where $\mathbf{I}_3 \in \mathbb{R}^{3\times3}$ is the identity matrix and S denotes the skew-symmetric matrix:

$$S(\mathbf{n}) = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix} \tag{A.14}$$

A.3 Dynamic Constraint Matrices

Expressions for the matrices in (2.43) are obtained by writing the sequence of equations from (2.41) and (2.42) at every collocation point as a set of linear ex-

pressions, which results in,

$$\mathbf{G} \triangleq \begin{bmatrix} -\frac{1}{2}\mathbf{I} & \mathbf{I} & -\frac{1}{2}\mathbf{I} & \mathbf{0} & \mathbf{0} & \dots \\ -\mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & -\frac{1}{2}\mathbf{I} & \mathbf{I} & -\frac{1}{2}\mathbf{I} & \\ \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{I} & \\ \vdots & & & & & \ddots \end{bmatrix} \tag{A.15}$$

and

$$\mathbf{H} \triangleq \begin{bmatrix} \frac{1}{8}\mathbf{I} & \mathbf{0} & -\frac{1}{8}\mathbf{I} & \mathbf{0} & \mathbf{0} & \dots \\ \frac{1}{6}\mathbf{I} & \frac{2}{3}\mathbf{I} & \frac{1}{6}\mathbf{I} & \mathbf{0} & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & \frac{1}{8}\mathbf{I} & \mathbf{0} & -\frac{1}{8}\mathbf{I} & \\ \mathbf{0} & \mathbf{0} & \frac{1}{6}\mathbf{I} & \frac{2}{3}\mathbf{I} & \frac{1}{6}\mathbf{I} & \\ \vdots & & & & & \ddots \end{bmatrix} \tag{A.16}$$

BIBLIOGRAPHY

[1] John D Albertson, Tierney Harvey, Greg Foderaro, Pingping Zhu, Xiaochi Zhou, Silvia Ferrari, M Shahrooz Amin, Mark Modrak, Halley Brantley, and Eben D Thoma. A mobile sensing approach for regional surveillance of fugitive methane emissions in oil and gas production. *Environmental science & technology*, 50(5):2487–2497, 2016.

[2] A Andersen, U Pesavento, and Z Jane Wang. Unsteady aerodynamics of fluttering and tumbling plates. *Journal of Fluid Mechanics*, 541:65–90, 2005.

[3] Michael L Anderson. Design and control of flapping wing micro air vehicles. Technical report, AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING AND MANAGEMENT, 2011.

[4] Patrick Bardow, Andrew J Davison, and Stefan Leutenegger. Simultaneous optical flow and intensity estimation from an event camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 884–892, 2016.

[5] Francisco Barranco, Cornelia Fermüller, and Yiannis Aloimonos. Contour motion estimation for asynchronous event-driven cameras. *Proceedings of the IEEE*, 102(10):1537–1556, 2014.

[6] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-based visual flow. *IEEE transactions on neural networks and learning systems*, 25(2):407–417, 2014.

[7] Ryad Benosman, Sio-Hoi Ieng, Charles Clercq, Chiara Bartolozzi, and Mandyam Srinivasan. Asynchronous frameless event-based optical flow. *Neural Networks*, 27:32–37, 2012.

[8] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[9] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010.

[10] Kwabena Boahen. A neuromorph's prospectus. *Computing in Science & Engineering*, 19(2):14–28, 2017.

[11] Jannika E Boström, Marina Dimitrova, Cindy Canton, Olle Hstad, Anna Qvarnström, and Anders Ödeen. Ultra-rapid vision in birds. *PloS one*, 11(3):e0151099, 2016.

[12] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

[13] Andrea Censi and Davide Scaramuzza. Low-latency event-based visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 703–710. IEEE, 2014.

[14] Song Chang and Z Jane Wang. Predicting fruit fly's sensing rate with insect flight simulations. *Proceedings of the National Academy of Sciences*, 111(31):11246–11251, 2014.

[15] Bo Cheng and Xinyan Deng. Translational and rotational damping of flapping flight and its dynamics and stability at hovering. *IEEE Transactions on Robotics*, 27(5):849–864, 2011.

[16] Pakpong Chirarattananon, Yufeng Chen, E Farrell Helbling, Kevin Y Ma, Richard Cheng, and Robert J Wood. Dynamics and flight control of a flapping-wing robotic insect in the presence of wind gusts. *Interface Focus*, 7(1):20160080, 2017.

[17] Pakpong Chirarattananon, Kevin Y Ma, and Robert J Wood. Adaptive control of a millimeter-scale flapping-wing robot. *Bioinspiration & Biomimetics*, 9(2):025004, 2014.

[18] Pakpong Chirarattananon, Kevin Y Ma, and Robert J Wood. Fly on the wall. In *Biomedical Robotics and Biomechatronics (2014 5th IEEE RAS & EMBS International Conference on*, pages 1001–1008. IEEE, 2014.

[19] Taylor S Clawson, Silvia Ferrari, Sawyer B Fuller, and Robert J Wood. Spiking neural network (snn) control of a flapping insect-scale robot. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3381–3388. IEEE, 2016.

[20] Taylor S Clawson, Sawyer B Fuller, Robert J Wood, and Silvia Ferrari. A blade element approach to modeling aerodynamic flight of an insect-scale robot. In *American Control Conference (ACC), 2017*, pages 2843–2849. IEEE, 2017.

[21] Taylor S Clawson, Terrence C Stewart, Chris Eliasmith, and Silvia Ferrari. An adaptive spiking neural controller for flapping insect-scale robots. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE, 2017.

[22] Jörg Conradt. On-board real-time optic-flow for miniature event-based vision sensors. In *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, pages 1858–1863. IEEE, 2015.

[23] Jörg Conradt, Matthew Cook, Raphael Berner, Patrick Lichtsteiner, Rodney J Douglas, and T Delbruck. A pencil balancing robot using a pair of aer dynamic vision sensors. In *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pages 781–784. IEEE, 2009.

[24] Adam Cox, Daniel Monopoli, Dragan Cveticanin, Michael Goldfarb, and Ephrahim Garcia. The development of elastodynamic components for piezoelectrically actuated flapping micro-air vehicles. *Journal of Intelligent Material Systems and Structures*, 13(9):611–615, 2002.

[25] Eugenio Culurciello and Andreas G Andreou. Cmos image sensors for sensor networks. *Analog Integrated Circuits and Signal Processing*, 49(1):39–51, 2006.

[26] T Delbruck and Patrick Lichtsteiner. Fast sensory motor control based on event-based hybrid neuromorphic-procedural system. In *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pages 845–848. IEEE, 2007.

[27] Xinyan Deng, Luca Schenato, and S Shankar Sastry. Flapping flight for biomimetic robotic insects: Part ii-flight control design. *IEEE Transactions on Robotics*, 22(4):789–803, 2006.

[28] Xinyan Deng, Luca Schenato, Wei Chung Wu, and S Shankar Sastry. Flapping flight for biomimetic robotic insects: Part i-system modeling. *IEEE Transactions on Robotics*, 22(4):776–788, 2006.

[29] Travis DeWolf, Terrence C Stewart, Jean-Jacques Slotine, and Chris Eliasmith. A spiking neural model of adaptive arm control. In *Proc. R. Soc. B*, volume 283, page 20162134. The Royal Society, 2016.

[30] 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.

[31] William B Dickson, Andrew D Straw, Christian Poelma, and Michael H Dickinson. An integrative model of insect flight control. In *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit*, pages 31–38, 2006.

[32] David Doman, Michael Oppenheimer, and David Sigthorsson. Dynamics and control of a minimally actuated biomimetic vehicle: Part i-aerodynamic model. In *AIAA Guidance, Navigation, and Control Conference*, page 6160, 2009.

[33] David Doman, Michael Oppenheimer, David Sigthorsson, Isaac Weintraub, and Ben Perseghetti. Wing velocity control system for testing body motion control methods for flapping wing mavs. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 332, 2013.

[34] David B Doman, Michael W Oppenheimer, and David O Sigthorsson. Wingbeat shape modulation for flapping-wing micro-air-vehicle control during hover. *Journal of guidance, control, and dynamics*, 33(3):724–739, 2010.

[35] Chris Eliasmith and Charles H Anderson. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2004.

[36] Charles P Ellington. The novel aerodynamics of insect flight: applications to micro-air vehicles. *Journal of Experimental Biology*, 202(23):3439–3448, 1999.

[37] Lukas Everding and Jörg Conradt. Low-latency line tracking using event-based dynamic vision sensors. *Frontiers in Neurorobotics*, 12:4, 2018.

[38] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.

[39] Imraan Faruque, Kenneth MacFarlane, and J Humbert. Reduced-order forward flight dynamics models for dipteran insects. In *AIAA Guidance, Navigation, and Control Conference*, page 4978, 2012.

[40] Silvia Ferrari and Robert F Stengel. Online adaptive critic flight control. *Journal of Guidance Control and Dynamics*, 27(5):777–786, 2004.

[41] Benjamin M Finio, Néstor O Pérez-Arancibia, and Robert J Wood. Sys-

tem identification and linear time-invariant modeling of an insect-sized flapping-wing micro air vehicle. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1107–1114. IEEE, 2011.

[42] Benjamin M Finio and Robert J Wood. Open-loop roll, pitch and yaw torques for a robotic bee. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 113–119. IEEE, 2012.

[43] Dario Floreano and Robert J Wood. Science, technology and the future of small autonomous drones. *Nature*, 521(7553):460, 2015.

[44] Greg Foderaro, Craig Henriquez, and Silvia Ferrari. Indirect training of a spiking neural network for flight control via spike-timing-dependent synaptic plasticity. In *49th IEEE Conference on Decision and Control (CDC)*, pages 911–917. IEEE, 2010.

[45] Steven N Fry, Rosalyn Sayaman, and Michael H Dickinson. The aerodynamics of free-flight maneuvers in drosophila. *Science*, 300(5618):495–498, 2003.

[46] 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.

[47] John C Gallagher, David B Doman, and Michael W Oppenheimer. The technology of the gaps: an evolvable hardware synthesized oscillator for the control of a flapping-wing micro air vehicle. *IEEE Transactions on Evolutionary Computation*, 16(6):753–768, 2012.

[48] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3867–3876, 2018.

[49] Matthew A Garratt and Javaan S Chahl. Vision-based terrain following for an unmanned rotorcraft. *Journal of Field Robotics*, 25(4-5):284–301, 2008.

[50] Jake Gemerek, Silvia Ferrari, Brian H Wang, and Mark E Campbell. Video-guided camera control for target tracking and following. *IFAC-PapersOnLine*, 51(34):176–183, 2019.

[51] Jake R Gemerek, Silvia Ferrari, and John D Albertson. Fugitive gas emission rate estimation using multiple heterogeneous mobile sensors. In *2017 ISOCS/IEEE International Symposium on Olfaction and Electronic Nose (ISOEN)*, pages 1–3. IEEE, 2017.

[52] Wulfram Gerstner and Werner Kistler. Spiking neuron models cambridge university press, 2002.

[53] MA Graule, P Chirarattananon, SB Fuller, NT Jafferis, KY Ma, M Spenko, R Kornbluh, and RJ Wood. Perching and takeoff of a robotic insect on overhangs using switchable electrostatic adhesion. *Science*, 352(6288):978–982, 2016.

[54] 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, 2009.

[55] E Farrell Helbling and Robert J Wood. A review of propulsion, power, and control architectures for insect-scale flapping-wing vehicles. *Applied Mechanics Reviews*, 70(1):010801, 2018.

[56] Jean Hendricks. Flicker thresholds as determined by a modified conditioned suppression procedure. *Journal of the experimental analysis of behavior*, 9(5):501–506, 1966.

[57] Lindsey Hines. Design and control of a flapping flight micro aerial vehicle. 2012.

[58] Lindsey Hines, Domenico Campolo, and Metin Sitti. Liftoff of a motor-driven, flapping-wing microaerial vehicle capable of resonance. *IEEE Transactions on Robotics*, 30(1):220–232, 2014.

[59] Gereon Hinz, Guang Chen, Muhammad Aafaque, Florian Röhrbein, Jörg Conradt, Zhenshan Bing, Zhongnan Qu, Walter Stechele, and Alois Knoll. Online multi-object tracking-by-clustering for intelligent transportation system with neuromorphic vision sensor. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 142–154. Springer, 2017.

[60] Naira Hovakimyan, Chengyu Cao, Evgeny Kharisov, Enric Xargay, and Irene M Gregory. L 1 adaptive control for safety-critical systems. *IEEE Control Systems Magazine*, 31(5):54–104, 2011.

[61] Noah T Jafferis, Mario Lok, Nastasia Winey, Gu-Yeon Wei, and Robert J Wood. Multilayer laminated piezoelectric bending actuators: design and manufacturing for optimum power density and efficiency. *Smart Materials and Structures*, 25(5):055033, 2016.

[62] Noah T Jafferis, Michael J Smith, and Robert J Wood. Design and manufacturing rules for maximizing the performance of polycrystalline piezoelectric bending actuators. *Smart Materials and Structures*, 24(6):065023, 2015.

[63] 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 1–8. IEEE, 2018.

[64] J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*, volume 149. SIAM, 2016.

[65] Bin Liang and Mao Sun. Nonlinear flight dynamics and stability of hovering model insects. *Journal of The Royal Society Interface*, 10(85):20130269, 2013.

[66] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128×128 120 db 15μs latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576, 2008.

[67] Shih-Chii Liu, André van Schaik, Bradley A Minch, and Tobi Delbruck. Asynchronous binaural spatial audition sensor with $2 \times 64 \times 4$ channel output. *IEEE transactions on biomedical circuits and systems*, 8(4):453–464, 2014.

[68] 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.

[69] Kevin Y Ma, Samuel M Felton, and Robert J Wood. Design, fabrication, and modeling of the split actuator microrobotic bee. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1133–1140. IEEE, 2012.

[70] Misha Mahowald. *An analog VLSI system for stereoscopic vision*, volume 265. Springer Science & Business Media, 1994.

[71] Pinaki Mazumder, D Hu, I Ebong, X Zhang, Z Xu, and Silvia Ferrari. Digital implementation of a virtual insect trained by spike-timing dependent plasticity. *INTEGRATION, the VLSI journal*, 54:109–117, 2016.

[72] Niall McLaughlin, Jesus Martinez del Rincon, and Paul Miller. Video person re-identification for wide area tracking based on recurrent neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[73] Moritz B Milde, Hermann Blum, Alexander Dietmüller, Dora Sumislawska, Jörg Conradt, Giacomo Indiveri, and Yulia Sandamirskaya. Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog/digital neuromorphic processing system. *Frontiers in neurorobotics*, 11:28, 2017.

[74] Diederik Paul Moeys, Federico Corradi, Emmett Kerr, Philip Vance, Gautham Das, Daniel Neil, Dermot Kerr, and Tobi Delbrück. Steering a predator robot using a mixed frame/event-driven convolutional neural network. In *Event-based Control, Communication, and Signal Processing (EBCCSP), 2016 Second International Conference on*, pages 1–8. IEEE, 2016.

[75] Toshiyuki Nakata, Hao Liu, and Richard J Bomphrey. A cfd-informed quasi-steady model of flapping-wing aerodynamics. *Journal of fluid mechanics*, 783:323–343, 2015.

[76] Randal C. Nelson and Jhon Aloimonos. Obstacle avoidance using flow field divergence. *IEEE Transactions on pattern analysis and machine intelligence*, 11(10):1102–1106, 1989.

[77] Robert C Nelson et al. *Flight stability and automatic control*, volume 2. WCB/McGraw Hill New York, 1998.

[78] Thomas Netter and Nicolas Francheschini. A robotic aircraft that follows terrain using a neuromorphic eye. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 129–134. IEEE, 2002.

[79] Stephen M Nogar, Abhijit Gogulapati, Jack J McNamara, Andrea Serrani, Michael W Oppenheimer, and David B Doman. Control-oriented modeling of coupled electromechanical-aeroelastic dynamics for flapping-wing vehicles. *Journal of Guidance, Control, and Dynamics*, 40(7):1664–1679, 2017.

[80] Michael W Oppenheimer, David B Doman, and David O Sigthorsson. Dynamics and control of a biomimetic vehicle using biased wingbeat forcing functions. *Journal of guidance, control, and dynamics*, 34(1):204–217, 2011.

[81] Michael W Oppenheimer, David Sigthorsson, David B Doman, and Isaac Weintraub. Wing design and testing for a tailless flapping wing micro-air vehicle. In *AIAA Guidance, Navigation, and Control Conference*, page 1271, 2017.

[82] 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*, 38(7):1187–1196, 2015.

[83] Christoph Posch, Teresa Serrano-Gotarredona, Bernabe Linares-Barranco, and Tobi Delbruck. Retinomorphic event-based vision sensors: bioinspired cameras with spiking output. *Proceedings of the IEEE*, 102(10):1470–1484, 2014.

[84] Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. Emvs: Event-based multi-view stereo—3d reconstruction with an event camera in real-time. *International Journal of Computer Vision*, 126(12):1394–1414, 2018.

[85] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Real-time visualinertial odometry for event cameras using keyframe-based nonlinear optimization. In *British Machine Vis. Conf.(BMVC)*, volume 3, 2017.

[86] Charles Richter and Hod Lipson. Untethered hovering flapping flight of a 3d-printed mechanical insect. *Artificial life*, 17(2):73–86, 2011.

[87] Leif Ristroph, Gordon J Berman, Attila J Bergou, Z Jane Wang, and Itai Cohen. Automated hull reconstruction motion tracking (hrmt) applied to sideways maneuvers of free-flying insects. *Journal of Experimental Biology*, 212(9):1324–1335, 2009.

[88] 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):20130237, 2013.

[89] Michelle H Rosen, Geoffroy le Pivain, Ranjana Sahai, Noah T Jafferis, and Robert J Wood. Development of a 3.2 g untethered flapping-wing platform for flight energetics and control experiments. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3227–3233. IEEE, 2016.

[90] Bodo Rueckauer and Tobi Delbruck. Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor. *Frontiers in neuroscience*, 10:176, 2016.

[91] David Russell and Z Jane Wang. A cartesian grid method for modeling multiple moving objects in 2d incompressible viscous flow. *Journal of Computational Physics*, 191(1):177–205, 2003.

[92] Sanjay P Sane and Michael H Dickinson. The control of flight force by a flapping wing: lift and drag production. *Journal of experimental biology*, 204(15):2607–2626, 2001.

[93] Robert M Sanner and J-JE Slotine. Gaussian networks for direct adaptive control. *IEEE Transactions on neural networks*, 3(6):837–863, 1992.

[94] Stephen Se, David Lowe, and Jim Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 2, pages 2051–2058. IEEE, 2001.

[95] David Sigthorsson, Michael W Oppenheimer, David B Doman, and Isaac Weintraub. Wing flexibility induced control reversal for flapping wing vehicles: Observation and evaluation. In *AIAA Guidance, Navigation, and Control Conference*, page 1273, 2017.

[96] Jean-Jacques E Slotine and Weiping Li. On the adaptive control of robot manipulators. *The international journal of robotics research*, 6(3):49–59, 1987.

[97] Mandyam V Srinivasan and Miriam Lehrer. Temporal acuity of honeybee vision: behavioural studies using moving stimuli. *Journal of Comparative Physiology A*, 155(3):297–312, 1984.

[98] Erik Steltz, Robert J Wood, Srinath Avadhanula, and Ronald S Fearing. Characterization of the micromechanical flying insect by optical position sensing. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1252–1257. IEEE, 2005.

[99] Robert F Stengel. *Optimal control and estimation*. Courier Corporation, 2012.

[100] Robert F Stengel. *Flight dynamics*. Princeton University Press, 2015.

[101] Terrence C Stewart and Chris Eliasmith. Large-scale synthesis of functional spiking neural circuits. *Proceedings of the IEEE*, 102(5):881–898, 2014.

[102] Mao Sun. Insect flight dynamics: stability and control. *Reviews of Modern Physics*, 86(2):615, 2014.

[103] Mao Sun and Jian Tang. Unsteady aerodynamic force generation by a model fruit fly wing in flapping motion. *Journal of experimental biology*, 205(1):55–70, 2002.

[104] Mao Sun, Jikang Wang, and Yan Xiong. Dynamic flight stability of hovering insects. *Acta Mechanica Sinica*, 23(3):231–246, 2007.

[105] Mao Sun and Yan Xiong. Dynamic flight stability of a hovering bumblebee. *Journal of experimental biology*, 208(3):447–459, 2005.

[106] Valentina Vasco, Arren Glover, Elias Mueggler, Davide Scaramuzza, Lorenzo Natale, and Chiara Bartolozzi. Independent motion detection with event-driven cameras. In *Advanced Robotics (ICAR), 2017 18th International Conference on*, pages 530–536. IEEE, 2017.

[107] Hamid R Vejdani, David B Boerma, Sharon M Swartz, and Kenneth S Breuer. The dynamics of hovering flight in hummingbirds, insects and bats with implications for aerial robotics. *Bioinspiration & biomimetics*, 14(1):016003, 2018.

[108] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Hybrid, frame and event based visual inertial odometry for robust, autonomous navigation of quadrotors. *arXiv preprint arXiv:1709.06310*, 2017.

[109] Z Jane Wang. Two dimensional mechanism for insect hovering. *Physical review letters*, 85(10):2216, 2000.

[110] Z Jane Wang. Insect flight: From newton's law to neurons. *Annual Review of Condensed Matter Physics*, 7:281–300, 2016.

[111] Z Jane Wang, James M Birch, and Michael H Dickinson. Unsteady forces and flows in low reynolds number hovering flight: two-dimensional computations vs robotic wing experiments. *Journal of Experimental Biology*, 207(3):449–460, 2004.

[112] Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.

[113] John P Whitney, Pratheev S Sreetharan, Kevin Y Ma, and Robert J Wood. Pop-up book mems. *Journal of Micromechanics and Microengineering*, 21(11):115021, 2011.

[114] JP Whitney and RJ Wood. Aeromechanics of passive rotation in flapping flight. *Journal of Fluid Mechanics*, 660:197–220, 2010.

[115] Jiang Hao Wu and Mao Sun. Floquet stability analysis of the longitudinal dynamics of two hovering model insects. *Journal of The Royal Society Interface*, 9(74):2033–2046, 2012.

[116] Yan Xiong and Mao Sun. Dynamic flight stability of a bumblebee in forward flight. *Acta Mechanica Sinica*, 24(1):25–36, 2008.

[117] Na Xu and Mao Sun. Lateral dynamic flight stability of a model bumblebee in hovering and forward flight. *Journal of theoretical biology*, 319:102–115, 2013.

[118] Sheng Xu and Z Jane Wang. A 3d immersed interface method for fluid–solid interaction. *Computer Methods in Applied Mechanics and Engineering*, 197(25-28):2068–2086, 2008.

[119] Takashi Yasuda, Isao Shimoyama, and Hirofumi Miura. Microrobot actuated by a vibration energy field. *Sensors and Actuators A: Physical*, 43(1-3):366–370, 1994.

[120] Tansel Yucelen and Wassim M Haddad. Low-frequency learning and fast adaptation in model reference adaptive control. *IEEE Transactions on Automatic Control*, 58(4):1080–1085, 2012.

[121] Kareem A Zaghloul and Kwabena Boahen. Optic nerve signals in a neuromorphic chip ii: Testing and results. *IEEE Transactions on Biomedical Engineering*, 51(4):667–675, 2004.

[122] Jian Zhang and Xinyan Deng. Resonance principle for the design of flapping wing micro air vehicles. *IEEE Transactions on Robotics*, 33(1):183–197, 2017.

[123] Xu Zhang, Ziye Xu, Craig Henriquez, and Silvia Ferrari. Spike-based indirect training of a spiking neural network-controlled virtual insect. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 6798–6805. IEEE, 2013.

[124] Yanlai Zhang and Mao Sun. Dynamic flight stability of a hovering model insect: lateral motion. *Acta Mechanica Sinica*, 26(2):175–190, 2010.

[125] Simon Zingg, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. Mav navigation through indoor corridors using optical flow. In *2010 IEEE International Conference on Robotics and Automation*, pages 3361–3368. IEEE, 2010.

[126] Yang Zou, Weiping Zhang, and Zheng Zhang. Liftoff of an electromagnetically driven insect-inspired flapping-wing robot. *IEEE Transactions on Robotics*, 32(5):1285–1289, 2016.