A neural network model of an insect's wing hinge reveals how steering muscles control flight

Thesis by
Johan Matthijs Melis

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2023
Defended December 16, 2022

Johan Matthijs Melis
ORCID: 0000-0001-8966-9496

"I love deadlines. I like the whooshing sound as they fly by." - Douglas Adams

ACKNOWLEDGEMENTS

After more than a decade of studying insect flight, I am happy to have achieved this milestone. Throughout my PhD, I have encountered numerous set-backs, dead ends, failed experiments and countless hours of trouble-shooting. Recurring themes were an overly optimistic planning and flies that would not cooperate. Despite the woes of a PhD, I learned and developed new methods in robotics, machine learning, control theory and even some neuroscience. Although my stubbornness and perseverance were important factors in the success of my thesis, there are several people who helped me to get to the end.

First of all, I would like to thank Michael Dickinson, for giving me the time, support and freedom to try out crazy ideas, and learn and develop new techniques. While encountering set-backs, Michael has always trusted in my ability to figure things out. Throughout my PhD, I could always rely on Michael's advice, knowledge, and mentorship.

With respect to the experiments of this thesis, I heavily relied on the work of two former post-docs in the Dickinson lab: Theodore Lindsay and Florian Muijres. I would like to thank them both for teaching me the intricacies of GCaMP imaging and high-speed videography, and their mentorship and friendship.

I would also like to thank all other members of the Dickinson lab, present and past, for their help, advice, and friendship. First, I would like to thank Ivo, for all the lengthy lunch breaks at Ernie's with all the scientific, and not so scientific, discussions. Will for his unfaltering help to solve all kind of experimental problems. Tarun for testing Deeplabcut and useful discussions about neural networks. Lilian for her administrative help and appreciation of Dutch coffee. And Ainul and Irene, for making my PhD a lot more fun.

The mountains, forests, deserts, and coastlines of California, Hawaii, Montana, Nevada, Utah, Colorado, and Wyoming, have been a welcome distraction from my research. I would like to thank my friends, Gautam, Daniel, and Simon, for exploring the outdoors with me, and hopefully we will finally make our trip to Alaska!

Being away from the rainy Netherlands was not always easy, especially when I could not attend the legendary Stomwijzer Christmas dinners. I would like to thank my Dutch friends for making me feel back home immediately, when I was in the Low

Countries. Especially during the pandemic, regular calls with Thijs were always a welcome catch up on politics, chess, and life in general.

During the last part of my PhD, I met Laura, and since then my research productivity went down, while my knowledge of historical facts and random places in Los Angeles significantly increased. Thank you, for all the fun we had and will continue to have!

Last but not least, I would like to thank my parents and brothers, Karel and Chris. I have always been able to chase my interest in science and engineering, thanks to the unrelenting support and love of my parents. Thank you! I hope that you will enjoy this work, despite the fact that it is not about bees, hummingbirds, bats, or any other "more interesting animal".

ABSTRACT

The flight system of the fly is remarkable. A fly can execute an escape maneuver in milliseconds, compensate for wing damage when half of the wing is missing, fly in turbulent conditions, and migrate over large distances. While there are many factors that contribute to the robustness and versatility of insect flight, it is the mechanical encoding of wing motion in the wing hinge that allows flies to rapidly and accurately change wing motion over a large dynamic range. The wing hinge consists of several hardened skeletal elements, named sclerites, and a set of twelve steering muscles are attached to some of these components within the exoskeleton. Due to the anatomical complexity and minute size of the sclerites, the way in which the steering muscles alter the mechanical encoding of wing motion in the hinge is poorly understood.

Using genetically encoded calcium indicators and high-speed videography, is is possible to simultaneously image steering muscle activity and wing motion. In order to extract wing pose from the high-speed video frames, an automated tracking algorithm was developed, that used a neural network and model fitting to accurately reconstruct the wing kinematics. The synchronous recordings of wing motion and steering muscle activity were used to train a convolutional neural network that learned to accurately predict the wing kinematics from muscle activity patterns. After training, the convolutional neural network was used to perform virtual experiments, revealing how the steering muscles regulate wing motion. Correlation analysis revealed that the 12 steering muscles have highly correlated activity. The correlation of muscle activity can be approximated well by a 12D-plane, in which all activity has to reside.

To study the function of the sclerites, a bottleneck was introduced in the convolutional neural network. The bottleneck consists of five neurons, or latent parameters, four parameters corresponding to the state of the different sclerites, on which the steering muscles act, and one parameter representing the wingbeat frequency. This so called latent network predicts both the changes in wing motion and muscle activity patterns as a function of sclerite state. The predicted wing motion as a function of sclerite state matches with previous anatomy and electrophysiology studies for the basalare, first axillary and third axillary sclerites. The fourth axillary sclerite has not been studied before, but shows an antagonistic relationship between the $hg_{1,2}$ and $hg_{3,4}$ muscles, resulting in a strong decrease and increase, respectively, of stroke amplitude, deviation and wing pitch angles.

By replaying the wing kinematics of the virtual experiments on a dynamically scaled robotic fly, a model of the aerodynamic and inertial control forces as a function of steering muscle activity was constructed. This control force model was subsequently integrated in a state-space system of fly flight, which in turn was integrated in a model predictive control simulation that was used to simulate free flight maneuvers. The body motion, steering muscle activity, and wing kinematics of the model predictive control simulations were strikingly similar to the recorded maneuvers of free-flying flies.

The integrative, multi-disciplinary approach that was used to reveal the mechanical logic of the wing hinge, and the control problem that a fly needs to solve to stay airborne, are both unprecedented in prior literature. The methodologies and models of this study will be a valuable resource in future research on how the fly's nervous system controls the complex behavior that is flight.

PUBLISHED CONTENT AND CONTRIBUTIONS

Melis, JM, T Lindsay, and MH Dickinson (2018). "Mapping steering muscle activity to 3-dimensional wing kinematics in fruit flies". In: *Integrative and Comparative Biology* 58.S1, E152. DOI: 10.1093/icb/icy001.
J.M.M developed the methodologies and software, performed experiments and analysis, and gave the presentation.

Cherin, Emmanuel et al. (2017). "Acoustic behavior of Halobacterium salinarum gas vesicles in the high-frequency range: experiments and modeling". In: *Ultrasound in medicine & biology* 43.5, pp. 1016–1030. DOI: 10.1016/j.ultrasmedbio.2016.12.020.
J.M.M. developed and performed the finite-element simulations of nano-scale gas vesicle bucling.

Maresca, David et al. (2017). "Nonlinear ultrasound imaging of nanoscale acoustic biomolecules". In: *Applied physics letters* 110.7, p. 073704. DOI: 10.1063/1.4976105.
J.M.M. developed and perfomred the finite-element simulations of nano-scale gas vesicle bucling.

Muijres, Florian T, Nicole A Iwasaki, et al. (2017). "Flies compensate for unilateral wing damage through modular adjustments of wing and body kinematics". In: *Interface Focus* 7.1, p. 20160103. DOI: 10.1098/rsfs.2016.0103.
J.M.M. developed the quasi-steady aerodynamic model.

Muijres, Florian T, Michael J Elzinga, et al. (2014). "Flies evade looming targets by executing rapid visually directed banked turns". In: *Science* 344.6180, pp. 172–177. DOI: 10.1126/science.1248955.
J.M.M. developed the Kalman filter, quasi-steady aerodynamic model, and animations.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

NOMENCLATURE

α. Angle-of-attack.

$\bar{c}$. Mean wing chord.

$\bar{P}^{*}_{ind}$. Mean specific aerodynamic power to overcome induced drag.

$\bar{P}^{*}_{pro}$. Mean specific aerodynamic power to overcome profile drag.

$\bar{U}$. Stroke-averaged absolute wing tip velocity.

$\hat{r}^{3}_{3}(S)$. Non-dimensional moment of third wing area.

ν. Kinematic viscosity.

Φ. Stroke extent in radians.

ρ. Air density.

AR. Aspect ratio.

C^{Pro}_{D}. Profile drag coefficient.

g. Gravitational acceleration.

Hz. Hertz, cycles per second.

m. Mass of the fly.

mm. Millimeter.

n. Wingbeat frequency.

p_w. Wing loading.

pO_2. Percentage of atmospheric oxygen.

R. Wing length.

Re. Reynolds number.

S. Wing area.

CFD. Computational Fluid Dynamics.

CNN. Convolutional Neural Network.

CPU. Central Processing Unit.

DLM. Dorso-Longitudinal Muscles.

DOF. Degrees Of Freedom.

DVM. Dorso-Ventral Muscles.

EKF. Extended Kalman Filter.

FT. Force-Torque.

GCaMP. Green fluorescent Calmodulin M13 Protein.

GECI. Genetically Encoded fluorescent Calcium-Indicators.

GPU. Graphics Processing Unit.

GUI. Graphical User Interface.

ICP. Iterative Closest Point.

LED. Light Emitting Diode.

LEV. Leading Edge Vortex.

MPC. Model Predictive Control.

MSE. Mean Squared Error.

NURBS. Non-Uniform Rational B-splines.

PID. proportional-integral-derivative.

PSO. Particle Swarm Optimization.

PSO. Particle Swarm Optimization.

PWM. pulse-width modulation.

ReLU. Rectified Linear activation Unit.

RGB. Red Green Blue.

SELU. Scaled Exponential Linear Unit.

SRF. Strokeplane Reference Frame.

STL. Standard Tessellation Language.

TEV. Trailing Edge Vortex.

UKF. Unscented Kalman Filter.

VNC. Ventral Nerve Cord.

Chapter 1

INTRODUCTION

One of the earliest fossils of a winged insect, a mayfly, is more than 300 million years old (Figure 1.1). The most recent studies date the first flying insects in the Carboniferous, between 350 and 400 million years ago. Being able to fly is a significant evolutionary advantage, and to this day, flying insects are the most abundant and species-rich taxon within the metazoan animals. Although modern mayflies are remarkably similar to their fossilized ancestors (Almudi et al., 2019), insects have continued to evolve their flight capabilities throughout the eras (Dudley, 2002). Flies in particular have evolved many novel adaptations of their flight system, leading to miniaturization and extreme aerial maneuverability. The focus of this thesis is on the result of millions of years of optimizing insect flight: the flight system of the fly. In particular, this research will investigate how the steering muscles, wing hinge, wing kinematics, and aerodynamics are integrated in the control system of the fruit fly: *Drosophila melanogaster*.

Figure 1.1: Oldest winged insect fossil, *Lithomantis carbonarius*, estimated to be 325 million years old (Ross, 2017). Scale bar 5 mm.

1.1 Stretch-activated muscles enabled the miniaturization of insects

In contrast to birds, bats, and pterosaurs, insects do not have muscles in their wings. After eclosion from the pupae, insect wings harden and form a flexible lightweight structure. Changes in orientation or shape of the wing are actuated through the wing hinge. In insects, including flies, the wings are powered indirectly through deformation of the thorax. Two sets of power muscles, the dorsal longitudinal and dorso-ventral muscles (DLMs and DVMs), are oriented approximately orthogonal within the thorax (Figure 1.2). Power muscles are stretch-activated, which means that the stretching of the muscle triggers contraction without the need of an action potential from the motor neuron. The approximately orthogonal orientation of the power muscles generates a self-sustained oscillation, i.e. the contraction of the DLMs stretches and activates the DVNs, and subsequent contraction of the DVMs stretches and activates the DLMs, etc. Using this mechanism, some midges can achieve wingbeat frequencies of 1000 Hz , (Sotavalta, 1953).

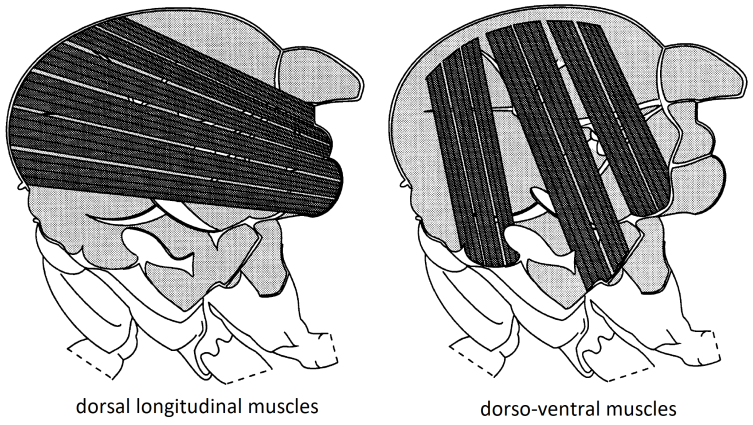

Figure 1.2: Cross-section of a blowfly's thorax with the dorsal longitudinal and dorso-ventral muscles, modified from (Dickinson and Tu, 1997).

Insects without stretch-activated power muscles, such as dragonflies and locusts, are limited in the wingbeat frequencies they can achieve. Conventional neurally activated muscles rely on the rapid release of calcium ions within the muscle fibers to trigger contraction. The muscle can only relax, and be ready to contract again, once the calcium level is brought back to the baseline level. Calcium ions are stored in the sarcoplasmic reticulum and uptake of the ions across the membrane of the internal storage organelle is an energy-intensive process. To supply the energy required for rapid calcium cycling, additional mitochondria are needed. The higher the contraction frequency of the muscle is, the lower the volume ratio between

muscle fibers and other cellular components such as mitochondria and sarcoplasmic reticulum becomes. This means that fast muscles are limited w.r.t the contractile force they can generate. For example, the cross-sectional area of the rattle-muscles of a rattlesnake, vibrating at 100 Hz, consists of only 30% muscle fibers (Iwamoto, 2011). The stretch-activated muscles of a bee contract at 230 Hz but their cross-sectional area contains 50% muscle fibers.

The development of stretch-activated muscles was a significant event in the evolution of insect flight. Having a relatively high power output at high wingbeat frequencies allowed insects to become small. The fossil record shows that insects in the Carboniferous and Permian periods had large wing lengths, up to 350 mm (Figure 1.3). During the Triassic, Jurassic and Cretaceous, the maximum wing length decreased to 70 mm, which is comparable to the largest insects alive today. Historically, these large wing lengths have been linked to high oxygen levels in the atmosphere. Statistical analysis by Clapham and Karr, 2012 shows that oxygen levels explain trends in wing length only until the end of the Jurassic, however. After the Jurassic, the evolution of aerial predators, such as birds and bats, provides a likely explanation for the decrease in wing length. Although insects with large wings have conventional, neurally activated, twitch-type muscles, the vast majority of insect species relies on stretch-activated muscles. Most insects are quite small; the average wing length of an insect is two orders of magnitude smaller than the maximum wing length. During the evolution of insects, stretch-activated muscles developed four times independent of each other, which suggests that the ability to fly at high wingbeat frequencies is crucial for the adaptive radiation of insects (Dudley, 2002).

1.2 Aerodynamic power requirements of insect flight

High wingbeat frequencies are an aerodynamic necessity for small insects. Before analyzing why this is the case, let me first define some basic aerodynamic terms. A wing generates both lift and drag forces. The shape of the cross-section of the wing is largely what determines the ratio of lift and drag forces and is called an *airfoil*. As lift and drag are dependent on factors such as wing orientation and air velocity, it is easier to quantify the aerodynamic characteristics of an airfoil or a wing using *dimensionless* numbers. A dimensionless number is a coefficient that has no measurement units. The advantage of a dimensionless number is that the performance of a certain airfoil shape can be compared across all flight conditions. Lift on a wing for example, can be characterized by the lift coefficient:

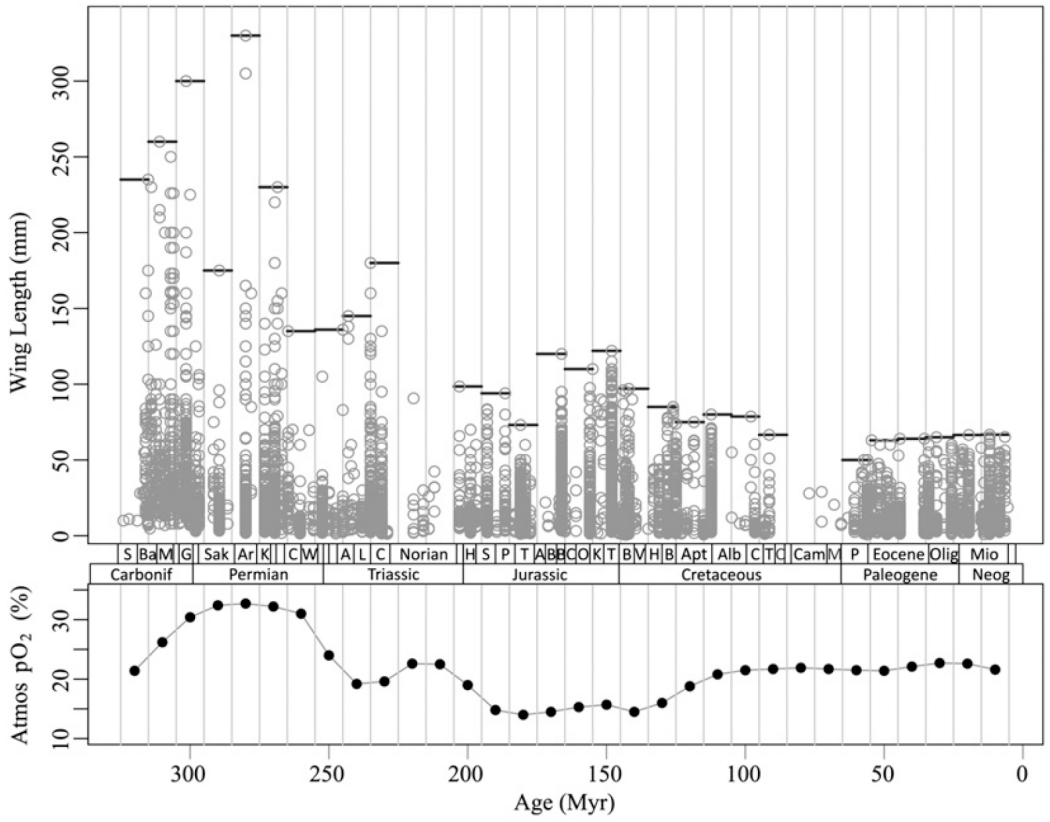

Figure 1.3: Fossil-record of insect wing length and atmospheric pO_2 (Clapham and Karr, 2012).

$$C_L = \frac{2L}{\rho U^2 S}, \tag{1.1}$$

where L is the lift force, ρ the air density, U the air velocity, and S the wing area. Similarly the drag characteristics of a wing can be expressed as the drag coefficient:

$$C_D = \frac{2D}{\rho U^2 S}, \tag{1.2}$$

where D is the drag force. Both the lift and drag force are dependent on the *angle-of-attack*, α, which is the angle between the chord line of the wing and the incoming air velocity.

The Reynolds number, Re, a dimensionless parameter that represents the ratio between inertial and viscous forces in a fluid, can be written as:

$$Re = \frac{\bar{U} \cdot \bar{c}}{\nu} = \frac{4 \cdot \Phi \cdot R^2 \cdot n}{\nu \cdot AR}, \tag{1.3}$$

where $\bar{U}$ is the stroke-averaged absolute wing tip velocity, $\bar{c}$ the mean chord, ν the kinematic viscosity, Φ the stroke extent in radians, R the wing length, n the wingbeat frequency, and AR the aspect ratio of the wing (Charles Porter Ellington, 1984b). The aspect ratio is defined as:

$$AR = \frac{2 \cdot R}{\bar{c}}, \tag{1.4}$$

and quantifies the slenderness of the wing. The Reynolds numbers of insects span several orders of magnitude. For example, the small *trichogrammatid* wasp has a wing length of 200 µm, a wingbeat frequency of 1000 Hz, and a Reynolds number of $Re = 2.5$. A hawkmoth is a relatively large insect and has a wing length of 50 mm, a wingbeat frequency of 26 Hz, and a Reynolds number of $Re = 5400$. *Drosophila melanogaster* have a wingbeat frequency of 200 Hz, a wing length of 2.5 mm, and a Reynolds number of $Re = 110$. From these three examples it becomes clear that there is an inverse relation between wing length and wingbeat frequency. This trend is not only present in insects but in all other flying animals such as birds and bats.

The origin of the inverse trend between wing length and wingbeat frequency lies in the relationship between the Reynolds number and profile drag. Profile or skin-friction drag is the drag generated by deceleration of air in the boundary layer around the wing. Profile drag is generally independent of lift generation and for a laminar boundary layer it can be approximated as:

$$C_D^{Pro} \approx \frac{7}{\sqrt{Re}}, \tag{1.5}$$

where C_D^{Pro} is the profile drag coefficient.

Besides profile drag there is a second source of drag on the wings: *induced drag*. The induced drag is the component of the pressure force on the wing that is parallel to the air velocity vector. In most insects, the cross-section of the wing can be approximated by a flat plate. Using the flat-plate approximation, one can assume that the pressure force will be normal to the wing surface. Now it is possible to compute the lift and drag components of the pressure force using the angle-of-attack, α, i.e. the angle between the cross-section of the wing and the air velocity vector.

In aircraft aerodynamics the angle-of-attack is usually small, as a high angle-of-attack ($\alpha > 15°$) results in the flow separating from the wing, a phenomenon named

stall. Flow separation generates low lift and strong drag forces that will slow a plane down until it falls out of the sky. In conventional aircraft aerodynamics, induced drag is usually associated with the tip vortex on the wing. During flight, a strong pressure differential exists between the bottom and top of the wing. At the tips of the wing, this pressure differential generates a strong tip vortex. The tip vortex has a detrimental effect on the lift generation of the wing, as it changes the flow field around the wing resulting in a lower *effective angle-of-attack*. For most wings, there is a linear relationship between the angle-of-attack and the lift generated at a given airspeed. A reduction in the effective angle-of-attack means lower lift generation. To provide weight support, a pilot would need to raise the angle-of-attack thereby increasing the lift-induced drag. To reduce the lift-induced drag, aircraft wings have a high aspect ratio such that the detrimental effects of the tip vortex only affect the outboard part of the wing.

In insect flight typical angle-of-attacks are around 45° and large areas with flow separation occur on the wing. The wings are still capable of generating strong lift forces, however, due to the presence of a leading edge vortex (LEV). At high angles-of-attack, the airflow immediately separates from the wing at the leading edge. The separation of the airflow creates a strong low pressure area on the leading edge of the wing, which bends the airflow back towards the wing. Before the trailing edge, the airflow reattaches to the wing. The LEV merges with the tip vortex and the low pressure area beneath the combined vortex is beneficial for the lift production. Therefore, in this dissertation, I consider the induced drag to be the component of the pressure force parallel to the air velocity vector, whereas the lift is the component of the pressure force perpendicular to the air velocity vector.

The aerodynamic power required to overcome the induced drag can be obtained using the Rankine-Froude estimate:

$$\bar{P}^*_{ind} = \frac{\bar{P}_{ind}}{m \cdot g} = \sqrt{\frac{2 \cdot p_w}{\rho \cdot \Phi \cdot AR}}, \tag{1.6}$$

with $\bar{P}^*_{ind}$ as the mean induced power, m the mass of the fly, g the gravitational acceleration, $p_w = \frac{m \cdot g}{S}$ the wing loading, ρ the air density, and S the wing area. The mean specific aerodynamic power required to overcome profile drag can be estimated by:

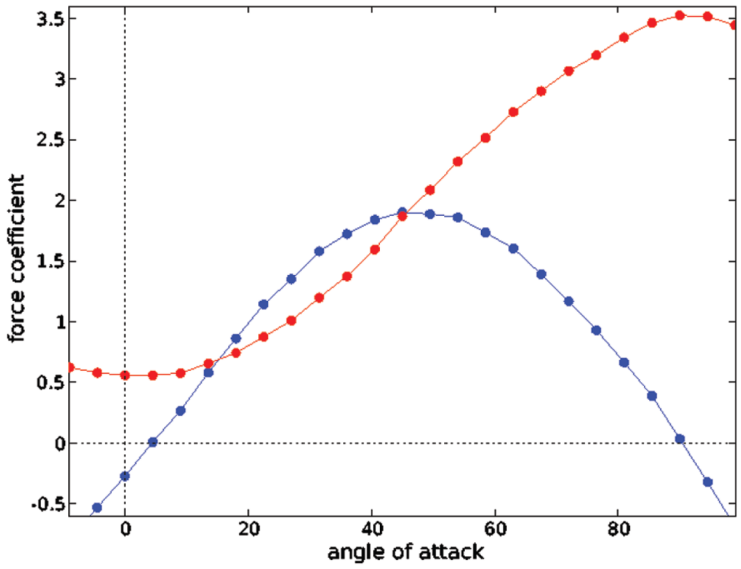

Figure 1.4: Lift and drag coefficients for a fruit fly wing with LEV attached. Note that the maximum lift coefficient (blue) occurs at $\alpha = 45°$ and the maximum drag coefficient (red) at $\alpha = 90°$. The wing does not stall for any angle-of-attack (Dickinson, Lehmann, and Sane, 1999).

$$\bar{P}^*_{pro} = \frac{\rho \cdot n^3 \cdot \Phi^3 \cdot R^3 \cdot \hat{r}_3^3(S) \cdot \pi^3}{2 \cdot p_w}, \tag{1.7}$$

with $\hat{r}_3^3(S)$ as the non-dimensional moment of third wing area. One can calculate the means specific aerodynamic power by adding the induced and profile drag power:

$$\bar{P}^*_{aero} = \bar{P}^*_{pro} + \bar{P}^*_{ind}. \tag{1.8}$$

I assume that the mass of a hovering insect scales with the wing length cubed, (Muijres, Elzinga, Melis, et al., 2014):

$$m = 68.5 \cdot R^3. \tag{1.9}$$

To compute the lift force on the wings the *quasi-steady* assumption is often used, meaning that the lift-force can be computed using the instantaneous values for airspeed and angle-of-attack (Weis-Fogh, 1973). Any time-dependent aerodynamic effects such as wake-capture (the forces caused by a flapping wing when it interacts with the wake shed during the previous stroke) are ignored, however for hovering

flight the quasi-steady lift force can explain more than 90% of weight support. The wingbeat-averaged quasi-steady lift force can be computed using:

$$\bar{L} = \frac{1}{2} \cdot \rho \cdot n^2 \cdot \Phi^2 \cdot R^2 \cdot S \cdot \hat{r}_2^2(S) \cdot \pi^2 \cdot \bar{C}_L, \tag{1.10}$$

where Φ corresponds to the stroke-extend in radians, $\hat{r}_2^2$ to the non-dimensional second moment of area, and $\bar{C}_L$ to the mean lift coefficient during the wingbeat. In hovering flight, the mean lift force needs to equal the insect's weight:

$$\bar{L} = m \cdot g, \tag{1.11}$$

where g is the gravitational acceleration, $g = 9.8\,\mathrm{m/s^2}$.

In Figure 1.5 the relation between mean specific aerodynamic power, wing length and aspect-ratio was investigated using the functions described above. Figure 1.5A shows several power curves over a range of wing length values ($100\mu m - 1m$) and the required wingbeat frequency to ensure weight support. The aspect-ratio and mean lift coefficient of the wing are constant per power curve. For each power curve, the minimum power value was found and plotted against aspect-ratio and mean lift coefficient, see (Figure 1.5B). The analysis shows that minimum specific aerodynamic power is a compromise between minimizing profile drag and induced drag. Optimal wingbeat frequencies range between 100 and 300 Hz. A fruit fly flaps its wings around 200 Hz and has a wing length of 2.5 mm, a data point that is close to some of the minimum power values found. The power curve with the lowest power requirements has a low aspect-ratio and high mean lift coefficient. A fruit fly wing has an aspect-ratio of 4 and mean lift coefficient of 2, which is close to the minimum power point. Low aspect-ratio wings and high lift (and drag) coefficients are trends that are generally observable in smaller insects (Charles Porter Ellington, 1984c), (Liu and Aono, 2009).

1.3 Aerodynamic mechanisms in insect flight

A persistent popular myth is that bumblebees should not be able to fly according to aerodynamic theory (S. P. Sane, 2017). Bumblebees can fly however and aerodynamicists are perfectly capable of computing the lift and drag forces on its wings. Most of the aerodynamic forces during insect flight can be calculated using a relatively simple set of equations, i.e. the quasi-steady model. The only thing that is different between insect flight and conventional aircraft aerodynamics is that some

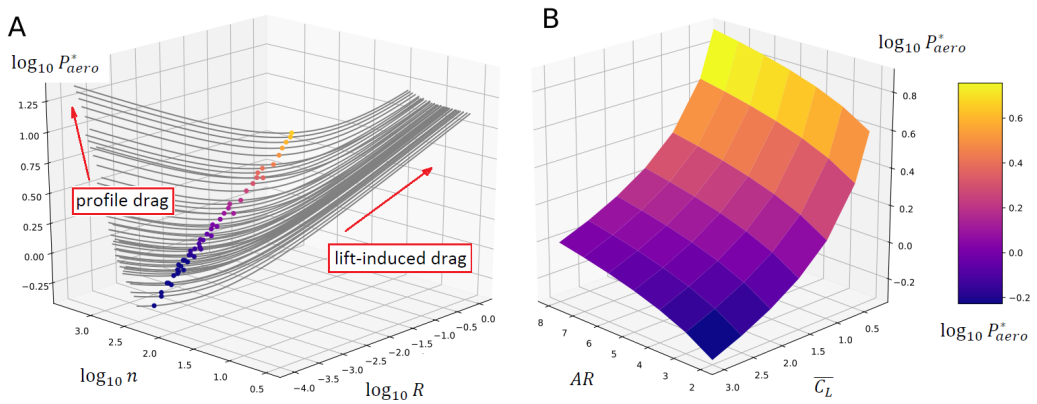

Figure 1.5: Relations between the required aerodynamic power for hovering flight and several parameters. A: Mean specific aerodynamic power requirements as a function of wingbeat frequency n and wing length R, plotted on a log scale. The grey trajectories correspond to different mean lift coefficients and aspect-ratio (see B). Directions of increasing profile and induced drag are marked with red arrows. Minimum power locations are marked with colored dots using the color map in B. B: Minimum specific aerodynamic power plotted against aspect-ratio AR and mean lift coefficient $\bar{C}_L$. Values correspond to the minimum power points in A.

common assumptions and simplifications about conventional aerodynamics are no longer applicable.

The analysis of the minimum required aerodynamic power for hovering flight shows a drive for smaller insects to increase the wingbeat-averaged lift coefficient. The simplest way to obtain a higher lift coefficient is to increase the angle of attack. As discussed previously, for high angles of attack a LEV will form on a flapping wing. The presence of a LEV on the wing increases the lift coefficient in two ways. First, the LEV makes it possible to reach high angles of attack without stalling the airflow on the wing. Second, the core of the LEV has a low pressure area that adds to the total pressure force on the wing (Figure 1.6).

LEVs are ubiquitous in insect flight and several bat and bird species also use LEVs to boost lift production during flight (Ellington et al., 1996), (Videler, Stamhuis, and Povel, 2004), (Muijres, Johansson, et al., 2008). Most insects would not be able to fly at all without the high lift coefficients that are obtained via the LEV. This reliance on the LEV begs the question of how reliable the formation and temporal stability of a LEV is. LEVs are not uncommon in aircraft aerodynamics, a striking example are the delta wings of the supersonic *Concorde*. At take-off and landing,

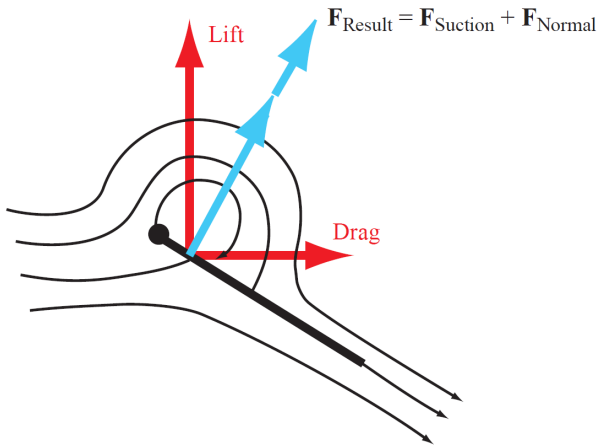

Figure 1.6: The total pressure force, F_{Result}, on a wing with LEV present consist of the pressure force that a flat-plate wing would experience without a LEV, F_{Normal} and a suction force, $F_{Suction}$, that is generated by the low pressure core of the LEV. The total pressure force can be split in the lift and drag component which are orthogonal and parallel to the air velocity, respectively (Dickinson and Gotz, 1993).

the delta wings are at a relatively high angle of attack and a LEV will form. Similar to insect wings, the presence of a LEV prevents the wing from stalling and boosts the lift coefficient. The sweep-back angle of the leading edge of delta wings ensures that the LEV will merge with the tip vortex (Figure 1.7).

When a LEV forms at the leading edge of the wing, the airflow separates from the wing surface. At the boundary between the free-stream airflow and the stagnated air on top of the wing there exists a strong gradient in air velocity. This so-called *shear layer* sheds *vorticity*, i.e. small vortices or *eddies*, into the LEV. The shedding of vorticity into the LEV makes the vortex grow in intensity until the vortex *bursts*. When a vortex bursts, the internal flow transitions from *laminar* to *turbulent* flow. The turbulent vortex is larger in size than the laminar vortex and the suction force is reduced. In aircraft with straight wings, i.e. not swept back as on delta-wing aircraft, vortex bursting will result in the detachment of the LEV and subsequent stalling of the wing (Anderson, 2009). The sweep angle of delta wings makes the LEV merge with the tip vortex, which means that vorticity gets transported from the LEV and tip vortex into the wake, helping to ensure that the LEV does not burst. An insect wing does not have a strong sweep angle however, which makes it vulnerable to vortex bursting and LEV detachment.

The stability of the LEV on an insect wing is not achieved by the sweep angle of

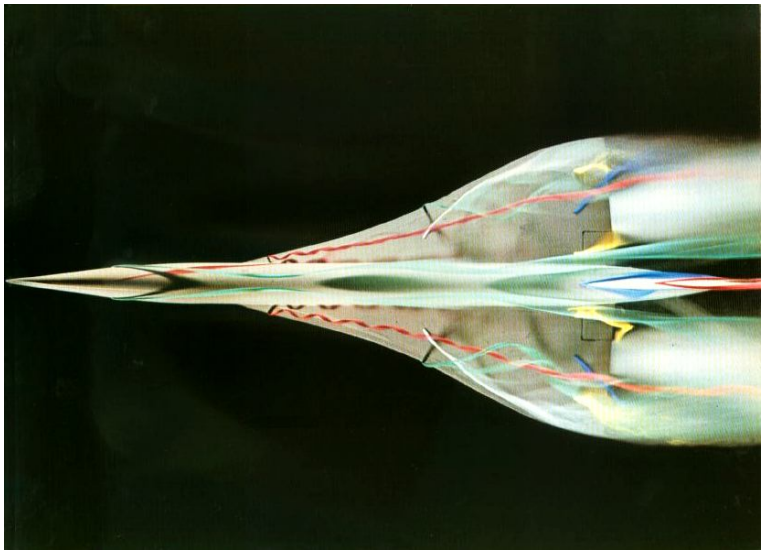

Figure 1.7: Visualization of the LEV on a water-tunnel model of the Concorde in landing configuration (Werlé, 1975).

the leading edge. How do insects stabilize the LEV on the wing and prevent it from stalling? One of the hypotheses was that the duration of the wing stroke is short enough to avoid vortex-bursting altogether (C. Ellington, 1995). This so-called *delayed stall* mechanism would mean that LEVs are unstable on insect wings but that the wingbeat frequencies are high enough to ever encounter the negative effects of vortex-bursting. However, this hypothesis does not explain how the continuously rotating seeds of maple trees can sustain a stable LEV (Lentink, Dickson, et al., 2009).

What mechanism does stabilize LEVs on insect wings? In a flow visualization study using a robotic fly wing, Lentink and Dickinson systematically investigated the effect of three wing motion parameters on LEV stability: the Reynolds number, stroke amplitude, and the aspect-ratio of the wing (Lentink and Dickinson, 2009). The aspect-ratio is defined as a dimensionless number, the so-called Rossby number:

$$Ro = R/\bar{c}. \tag{1.12}$$

Similarly, the stroke amplitude is expressed as the dimensionless stroke amplitude:

$$A^* = \Phi R/\bar{c}. \tag{1.13}$$

Figure 1.8 shows the LEV stability for eight different combinations of Ro, Re, and A^*. The dimensionless stroke amplitude and the Rossby number were tested for values corresponding to fly flight and infinity. In case of the Rossby number, a finite number corresponds to a rotating wing and an infinite value to a translating wing. An infinite value for the dimensionless stroke amplitude means that the stroke duration is infinite. The two different values of the Reynolds number correspond to fruitfly ($Re = 110$) and housefly ($Re = 1400$). A Reynolds number below 1000 means that the flow is fully laminar, whereas turbulent flow is possible at Reynolds numbers greater than 1000. In a similar way, the values picked for the dimensionless stroke amplitude can distinguish between the delayed-stall mechanism and continuous LEV stability. Finally, the Rossby number distinguishes between translating and rotating motion of the wing. The conclusion of the LEV stability analysis is that the Rossby number is the determining factor for whether a LEV stays attached to the wing or not. When the Reynolds number allows for turbulent flow on the house fly wing, the LEV will burst at the outboard section of the but stays attached. LEV stability is independent of the dimensionless stroke amplitude and remains attached for both a flapping and continuously rotating wing. However, when the wing motion transitions from rotating to translating motion, the LEV will detach and the wing will stall. The LEV is stabilized by centripetal and Coriolis accelerations that the airflow experiences when close to the wing surface. A Rossby number of $Ro = 3$ guarantees a stable LEV while a higher number can result in LEV detachment. An investigation of the Rossby number in different flying animals shows that many species can sustain a stable LEV during flight over a large range of Reynolds numbers, from a fruit fly to a mute swan.

Most insects make use of a LEV to boost lift production, however in flapping flight the reciprocating motion means that the wing experiences low air velocities during stroke reversal. The wingbeat-averaged lift coefficient of a flapping wing will therefore be significantly lower than for a continuously rotating wing. A second aerodynamic mechanism makes use of the rotation around the wing's longitudinal axis during stroke reversal to produce lift. This so-called *rotational lift* is similar to the Magnus effect, i.e. a spinning cylinder or sphere can generate positive or negative lift depending on the direction of the airflow (Seifert, 2012). As insect wings are flat, this effect is more complicated and the instantaneous orientation of the wing affects lift and drag forces. Rotational lift generated by an insect wing rotating around its longitudinal axis is called the Kramer effect. The Kramer effect can be described in the quasi-steady formulation as:

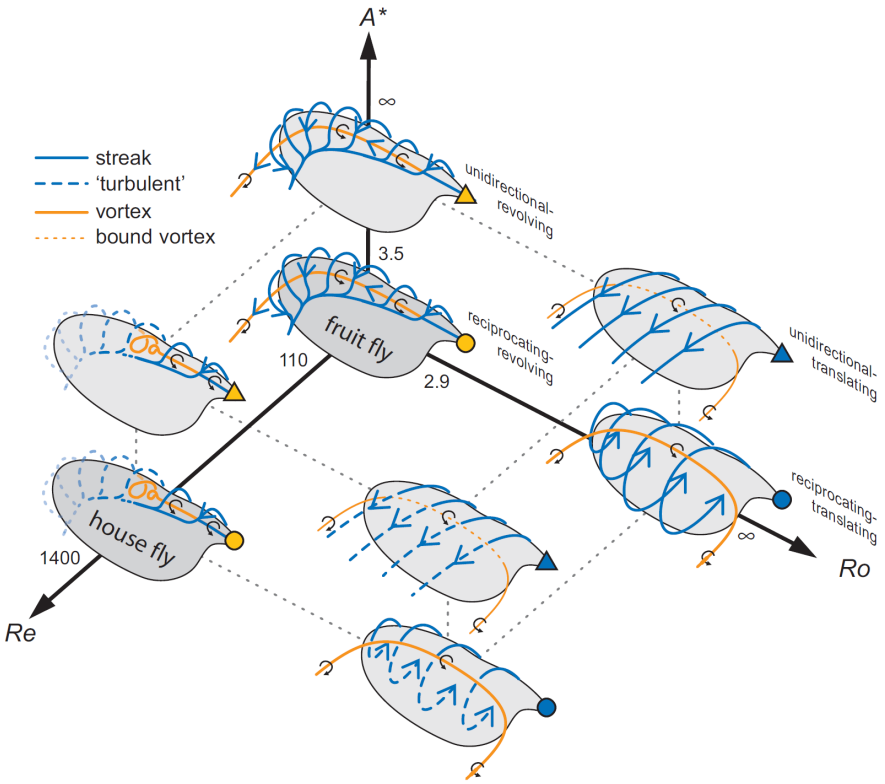

Figure 1.8: LEV stability for different Reynolds numbers, Re, Rossby numbers, Ro, and dimensionless stroke amplitude, A^*. Solid blue lines indicate laminar streamlines while dashes blue lines correspond to turbulent streamlines. The core of the vortex is shown by orange solid lines while orange dashed lines indicate a bound vortex. A bound vortex means that there is no LEV present on the wing and that any lift is generated by the wing surface alone. The detachment of the LEV means that the wings with a bound vortex are in stall conditions (Lentink and Dickinson, 2009).

$$F_{rot} = C_{rot}\rho\sqrt{S_{xx}S_{yy}}\omega_{pitch}\omega_{stroke}, \tag{1.14}$$

where F_{rot} is the rotational force, C_{rot} the rotational lift coefficient, S_{xx} and S_{yy} the second moment of area around the x and y axes, ω_{pitch} the angular velocity around the longitudinal axis of the wing and ω_{stroke} the angular velocity in the strokeplane (Figure 1.9). The second moment of area can be calculated by:

$$S_{xx} = \int_0^R y^2 dA, \tag{1.15}$$

and,

$$S_{yy} = \int_{c_1}^{c_2} x^2 dA, \tag{1.16}$$

with x and y as coordinates in the wing reference frame, dA an infinitesimal wing area, R the wing length, and c_2 and c_1 the locations of the leading and trailing edge of the wing respectively (Figure 1.9).

Using dynamically scaled robotic experiments, Sane and Dickinson (2002), found the rotational lift coefficient to be 2.08 for insect wings. Most versions of the quasi-steady model assume the wing to be flat and the rotational force is therefore always normal to the wing. The location of the center of pressure on the chord varies with angle-of-attack however and was empirically found to be:

$$x_{cp}(\alpha) = (c_2 - c_1)\left[0.82\frac{|\alpha|}{\pi} + 0.05\right], \tag{1.17}$$

where $c_2 - c_1$ indicates the chord length at location r_y and x_cp the x coordinate of the center of pressure at r_y.

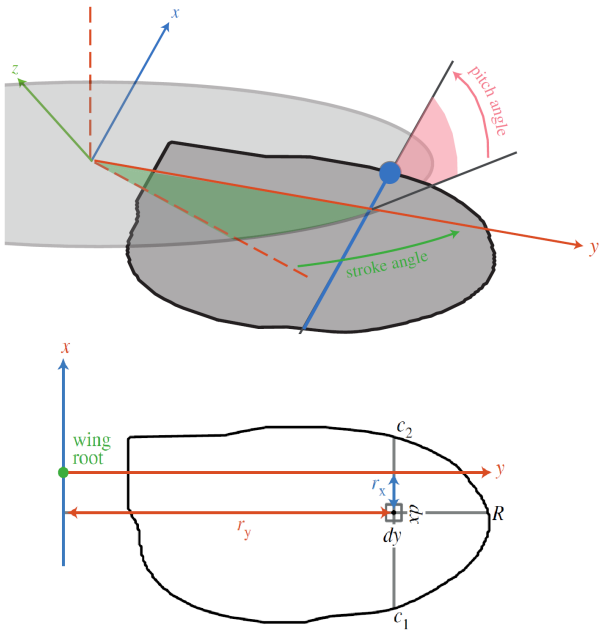

Figure 1.9: Wing-based reference frame definition of the quasi-steady model with wing pitch angle and stroke angle (Veen, Leeuwen, and Muijres, 2019).

A recent Computational Fluid Dynamics (CFD) study by, van Veen, van Leeuwen and Muijres (2019), found that there is a second rotational lift mechanism. This

so-called *pitch-rate rotational force* relies only on wing pitch rotation and does not require stroke velocity. The quasi-steady formulation for the pitch-rate rotational force is:

$$F_{pitch} = C_{rot}\rho S_{|x|x}\omega_{pitch}^2, \tag{1.18}$$

with F_{pitch} normal to the wing surface and $S_{|x|x}$ the asymmetric second moment of area around the x-axis, which can be calculated by:

$$S_{|x|x} = \int_0^R x|x|dA. \tag{1.19}$$

Rotational forces are important during stroke reversal when lift produced by the LEV and wing is small. Insects with high wingbeat frequencies and therefore low wingbeat amplitudes tend to rely more on rotational forces (Altshuler et al., 2005). For example, the southern house mosquito, *Culex quinquefasciatus*, flaps its wings back-and-forth at 717 Hz and has a stroke-extend of 44° and relies for approximately 50% on rotational lift for weight support during hovering, (Bomphrey et al., 2017). In *Drosophila*, the wingbeat frequency is lower (200 Hz) and the stroke-extend therefore higher (150°) which makes the role of rotational lift smaller: approximately 10% during hovering flight.

When a wing starts moving from standstill it will shed a vortex from the trailing edge, the so-called *starting vortex*. The circulation of the starting vortex is opposite to the bound circulation on the wing. Lift is directly related to the circulation on a wing:

$$L' = \rho\Gamma V, \tag{1.20}$$

where L' is the lift per unit span of the wing, V the freestream air velocity, and Γ the circulation per unit span. A vortex close to the wing with the same sign of circulation increases the total circulation on the wing. The strength of the circulation increase is inversely proportional to the distance of the vortex to the wing:

$$\Delta\Gamma = \int_0^c \frac{\Gamma_{vortex}\left(\frac{c}{2} - x\right)}{4\pi\left(d + \frac{c}{2} - x\right)}dx, \tag{1.21}$$

where $\Delta\Gamma$ is the increase in circulation, c the chord length, Γ_{vortex} the circulation of the vortex and d the distance between the circulation centers of the vortex and the wing. When the vortex has opposite circulation compared to the wing, the total circulation on the wing decreases as does the lift on the wing. The starting vortex has opposite circulation w.r.t. the wing and therefore has a negative effect on the lift production. As the wing moves away from the starting vortex, this negative effect on lift diminishes. This delay in circulation growth is known as the *Wagner effect* (Figure 1.10) (Wagner, 1925). Because the wing accelerates from standstill twice during a wingbeat, the Wagner effect negatively impacts lift generation in flapping flight.

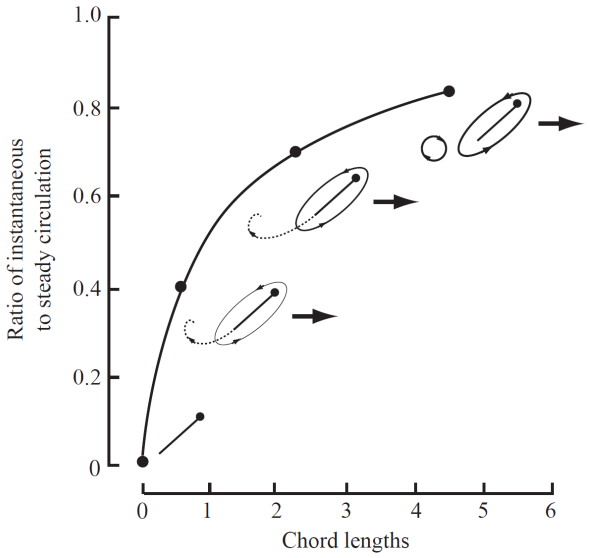

Figure 1.10: Wagner effect. Delay in circulation growth as a function of chord lengths travelled from the starting vortex for an instantaneously accelerated airfoil travelling at constant speed (S. P. Sane, 2003).

Some insects get around the Wagner effect using the *clap-and-fling* mechanism. At the end of the dorsal wing stroke, the left and right wing clap together and subsequently get peeled apart at the start of the next wing stroke. During the clap phase, the LEV and the Trailing Edge Vortex (TEV) shed from the wing (Figure 1.11). The wake between the LEV and TEV hits the wing, pushing the left and right wing against each other. When the wings clap together, the air between the wings gets pushed downwards and generates an upward force. Both the shed LEV and TEV move away from the wing quickly in the downward wake. Similarly, the wakes of the left and right wing create an upward force when the wings clap together and deflect the momentum of the wakes downwards. The fling phase starts

with the leading edges of the wings moving apart. Air rushes in over the leading edge and forms an LEV. When the trailing edges of the wing peel apart another downward wake occurs, strengthening the LEVs on both wings. The advantage of the clap-and-fling mechanism is that the starting vortex does not form as the shed vorticity of both wings have opposite signs and cancel out each other. Additionally the downward wakes of the clap-and-fling phases provide upward thrust. Although some insects employ the clap-and-fling mechanism in free flight, for many insects, like mosquitoes, the stroke amplitude required for the wings to touch is unfeasible. While fruit flies regularly employ clap-and-fling under tethered flight conditions, in free flight clap-and-fling is rare.

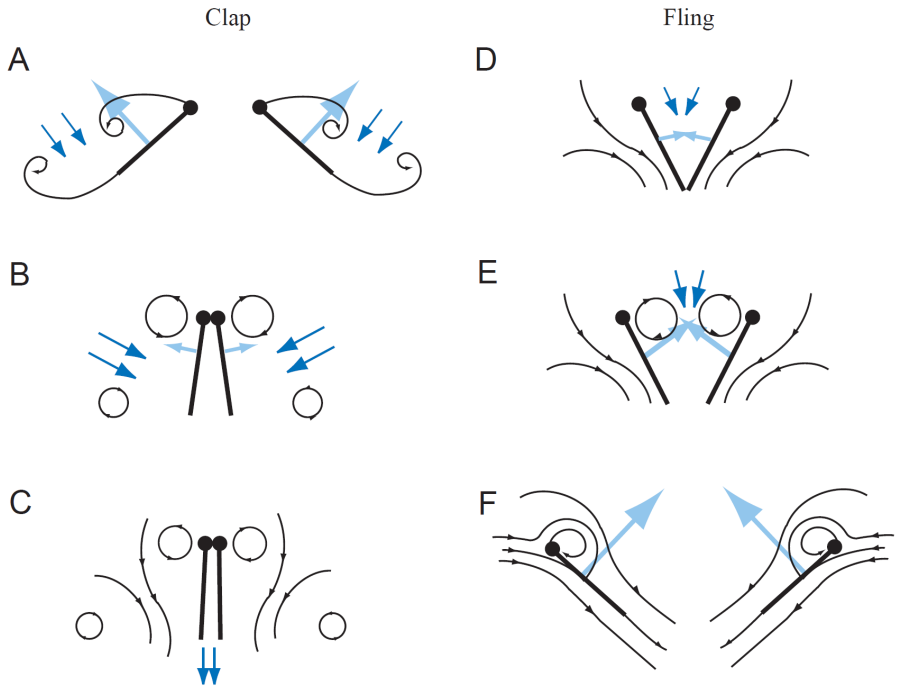

Figure 1.11: Clap-and-fling mechanism. The bold black lines show a cross-section of the wing and the thin black lines correspond to flow lines. Light blue arrows show the forces on the wings and dark blue arrows the induced flow. A: The wings approach each other and touch first at the leading edge. B: LEV and TEV vortices shed from the wing and induce strong flow that claps the wings together. C: The clap pushes out the air between the wings, in combination with the deflected induced flow, creating a strong downward flow. D: Leading edges of both wings move apart and air rushes into the gap. E: LEVs form on both wings and induce a downward flow. F: Fully developed circulation of the LEV and wing generates strong lift forces (Weis-Fogh, 1973).

The Reynolds number for most insects is generally low compared to other flying

animals such as birds and bats. As a consequence, viscous forces play an important role in insect flight. One aerodynamic effect that becomes more significant for low Reynolds numbers is *added mass*. Added mass can be viewed as additional inertia due to the air that gets dragged along with the wing (Figure 1.12). Fly wings are extremely lightweight, one fruit fly wing weighs approximately 0.15% of the body mass (Charles Porter Ellington, 1984a). Due to this low weight, the mass of the air that moves with the wing is significant. The volume of air that gets accelerated with the wing depends on many factors such as the angle-of-attack, wing shape, and the magnitude and (rotational) direction of the acceleration. A quasi-steady formulation of the added mass is described by:

$$F_{AM} = \rho \dot{\omega} S_{cy} C_{F_{AM}}(\alpha), \tag{1.22}$$

with $\dot{\omega}$ as the angular acceleration and S_{cy} as the chord-based second moment of area:

$$S_{cy} = \int_0^R c(y)^2 y dy, \tag{1.23}$$

where $c(y)$ is the chord length at spanwise section y. The added mass coefficient, $C_{F_{AM}}$, can be split into an x, y, and z component. Lift and drag forces are in the $x-z$ plane and the y component can be ignored. The coefficients for the x and z directions are:

$$C_{Fx_{AM}} = C_{Fx\alpha_{AM}} cos(\alpha), \tag{1.24}$$

and

$$C_{Fz_{AM}} = C_{Fz\alpha_{AM}} sin(\alpha), \tag{1.25}$$

where $C_{Fx\alpha_{AM}}$ and $C_{Fz\alpha_{AM}}$ were found to be 0.104 and 0.96 respectively.

A recent CFD study by (Veen, Leeuwen, Oudheusden, et al., 2022) computed the forces from the Wagner effect, and added mass and translational forces during the start of the wing stroke. A quasi-steady formulation for the Wagner effect can be expressed as:

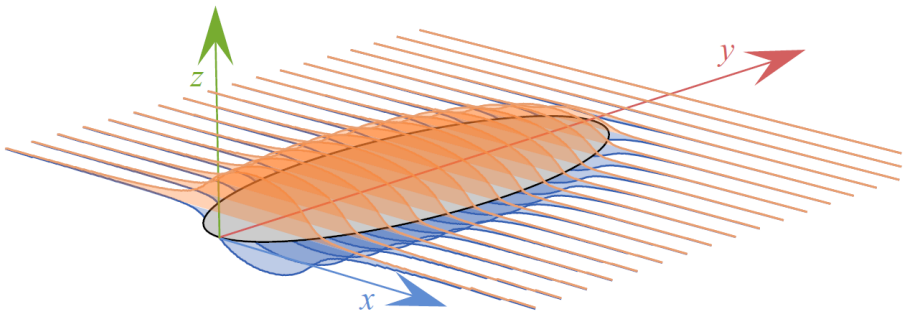

Figure 1.12: Visualization of the fluid layer accelerated by an elliptical wing, (Veen, Leeuwen, Oudheusden, et al., 2022).

$$F_{WE} = \frac{1}{2}\rho\omega\sqrt{\dot{\omega}}S_{WE}C_{F_{WE}}(\alpha), \tag{1.26}$$

where S_{WE} is the wing geometry scaling parameter of the Wagner effect:

$$S_{WE} = \int_0^R \sqrt{c(y)^3 y^3}dy, \tag{1.27}$$

and the Wagner effect coefficients for the x and z directions:

$$C_{Fx_{WE}}(\alpha) = C_{Fx\alpha_{WE}}cos(\alpha), \tag{1.28}$$

and

$$C_{Fz_{WE}}(\alpha) = C_{Fz\alpha_{WE}}sin(\alpha), \tag{1.29}$$

where $C_{Fx\alpha_{WE}}$ and $C_{Fz\alpha_{WE}}$ are 0 and -1.02 respectively. The translational lift and drag forces are rewritten as a function of ω:

$$F_{trans} = \frac{1}{2}\rho\omega^2 S_{yy}C_{F_{trans}}(\alpha), \tag{1.30}$$

with:

$$C_{Fx_{trans}}(\alpha) = A_{Fx\alpha_{trans}}\alpha^2 + B_{Fx\alpha_{trans}}\alpha + C_{Fx\alpha_{trans}}, \tag{1.31}$$

and

$$C_{Fz_{trans}}(\alpha) = C_{Fz\alpha_{trans}} cos(\alpha), \tag{1.32}$$

with $A_{Fx\alpha_{trans}} = 8.5{\times}10^{-5}$, $B_{Fx\alpha_{trans}} = -1.2{\times}10^{-2}$, $C_{Fx\alpha_{trans}} = 0.41$ and $C_{Fz\alpha_{trans}} = 3.13$.

With quasi-steady terms for translational, added mass and Wagner effect forces it is possible to compute the total aerodynamic force during a wingstroke (excluding rotational forces). The total aerodynamic force is simply the sum of all three effects:

$$F_{total} = F_{trans}(\alpha, \omega, S_{yy}) + F_{AM}(\alpha, \dot{\omega}, S_{cy}) + F_{WE}(\alpha, \omega, \dot{\omega}, S_{WE}). \tag{1.33}$$

Analyzing the development of the total aerodynamic force during wingbeats of mosquitoes and fruit flies shows that the acceleration of the wing is such that the added mass and Wagner effect forces cancel each other out (Figure 1.13). Mosquitoes have a higher wingbeat frequency and lower stroke extend than fruit flies and the acceleration forces (Wagner effect + added mass) have a larger effect on lift and drag relative to the translational force. The cancellation of the Wagner effect by the added mass force explains why most flying insects do not regularly employ the clap-and-fling mechanism: the delay in circulation growth is compensated by added mass forces. Clap-and-fling also increases mechanical wear on the wings and requires more muscle power to reach the required dorsal stroke extend.

1.4 Wing motion patterns are mechanically encoded in the wing hinge

Many of the aerodynamic mechanisms rely on precise control of parameters like the angle of attack and timing of wing rotation. The power muscles deform the thorax and this deformation gets transformed into a precise and highly stereotyped wing motion pattern via the wing hinge. Exactly how the wing hinge can transform the simple up-and-down motion of the thorax into a complex 4D wing motion pattern has remained a mystery after more than a century of research.

Several studies have tried to infer the mechanical workings of the wing hinge by carefully analyzing its anatomy (Williams and Williams, 1943), (J. Pringle, 1949), (Boettiger and Furshpan, 1952), (John William Sutton Pringle, 1957), (Wisser and Nachtigall, 1984), (Miyan and Ewing, 1985), (Ennos, 1987). In this section I will discuss some of the hypotheses that are based on careful dissection of the wing hinge and thorax. Although informative, one has to keep in mind that these studies only provide limited understanding of the mechanics of the wing hinge, as they are

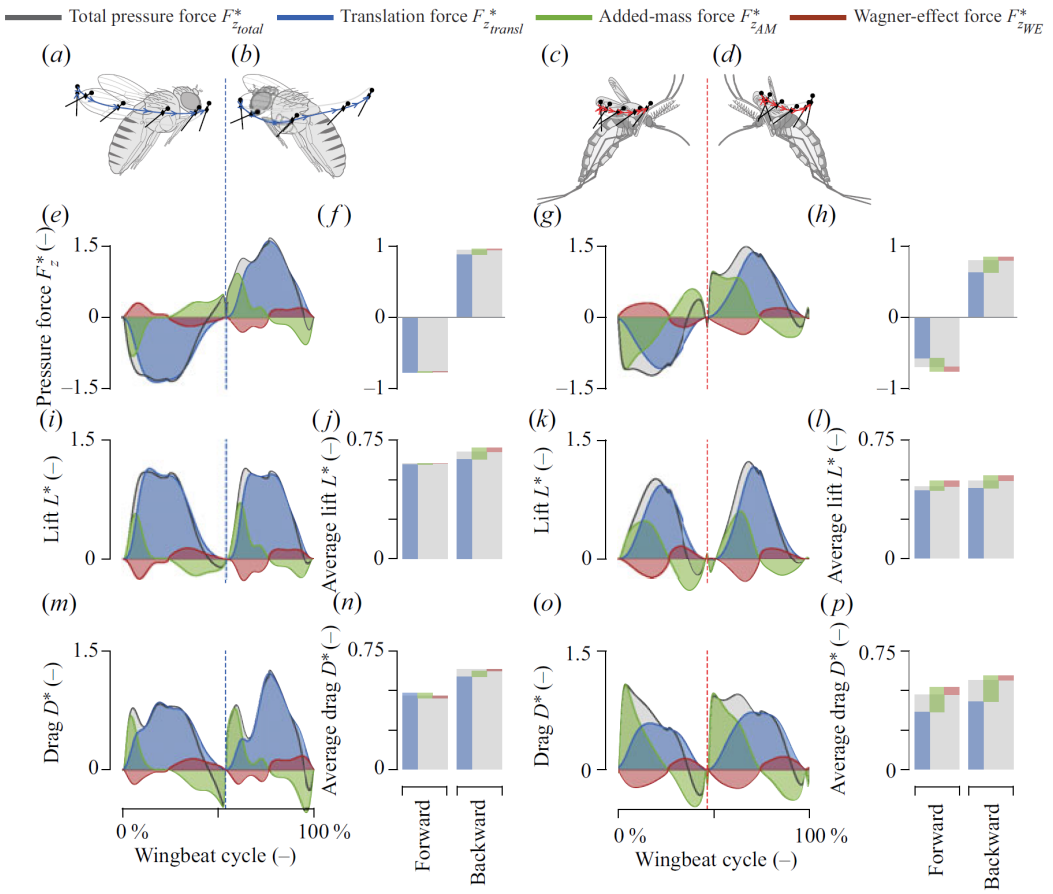

Figure 1.13: Total aerodynamic force during the wingbeat of a fruit fly and mosquito. A: Total pressure force (grey), translation force (blue), added mass force (green). and Wagner effect force (red) during the downstroke of a fruit fly wingbeat. The pressure force F^*_Z is normal to the wing surface and the lift and drag forces are in the vertical and horizontal plane respectively. B: Forces during the upstroke of the fruit fly wingbeat. C,D: Forces during the down-and-upstroke of a mosquito wingbeat (Veen, Leeuwen, Oudheusden, et al., 2022).

obtained from deceased animals. The wing hinge is difficult to image during flight, as it is small and some parts are internal to the thorax. More recent studies have used techniques such as micro Computerized Tomography (μCT) and Scanning Electron Microscopy (SEM) scans to obtain the detailed 3D morphology of the thorax, musculature, and the wing hinge (Fabian, Schneeberg, and Beutel, 2016), (Deora, Gundiah, and Sane, 2017). Again, these studies are on deceased animals and only give limited insight on the wing hinge mechanics. An *in-vivo* X-ray tomography of a tethered *Calliphora vicina* blowfly is the state-of-the-art study on the mechanics of the wing hinge in flight (Walker, Schwyn, et al., 2014). Although impressive,

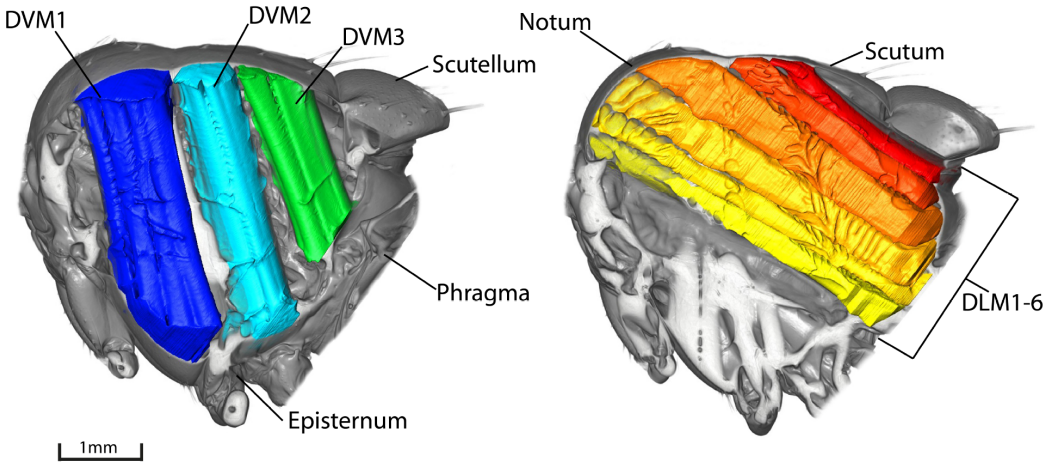

Figure 1.14: Orientation of the DVMs and DLMs in the thorax of the blowfly *Calliphora vicina*. Five regions of the exo-skeleton of the thorax are indicated: notum, scutum, scutellum, phragma, and episternum (Page, 2018).

the spatial and temporal resolution of these results are not sufficient to resolve the internal motion of the wing hinge. Besides the lack of resolution, the recordings were made from a single individual fly that was spinning at a fast rate whilst being subjected to damaging radiation. The hypotheses on the mechanical workings of the wing hinge I will discuss in this section are, therefore, based on earlier anatomy studies.

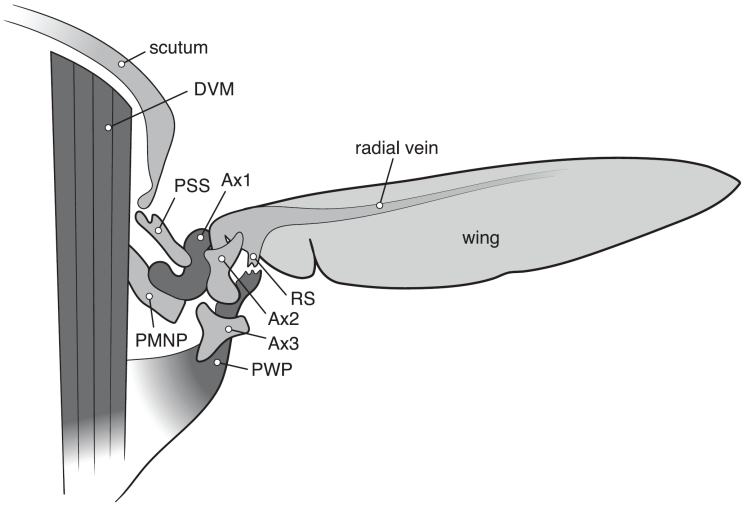

Figure 1.15: Schematic of a cross-section of the wing, with: Parascutal Shelf (PSS), Post-Medial Notal Process (PMNP), first axillary sclerite (Ax1), second axillary sclerite (Ax2), third axillary sclerite (Ax3), Radial Stop (RS), and Pleural Wing Process (PWP) (Hedenström, 2014).

The DVMs and DLMs act on the notum and scutum at the top of the thorax (Figure 1.14). Due to the difference in orientation of the muscles, the DVMs and DLMs have opposite effects on the motion of the scutum. The DVMs attach to the episternum and are oriented vertically in the thorax, moving the scutum down when activated. DLMs attach to the phragma of the metathorax at a more horizontal orientation, moving the scutum up when activated. Motion of the scutum is transferred to the wing via a piece of the exo-skeleton of the thorax, or *cuticle*, named the parascutal shelf. The wing hinge consists of several hardened interlocking skeletal elements named *sclerites*. Two of these sclerites, the first and second axillary sclerites, transfer the motion from the parascutal shelf to the radial vein of the wing (Figure 1.15). The second axillary sclerite is rigidly attached to the radial vein and functions as a fulcrum on the pleural wing process.

The transfer of mechanical power from the scutum to the wing via the parascutal shelf, first and second axillary sclerites was described in (Boettiger and Furshpan, 1952). In the study, Boettiger and Furshpan made use of carbon tetrachloride, CCl_4, to anaesthetize flies during flight, such that the wings are extended. What they found was that the wings could toggle between two extremes: the maximum dorsal or maximum ventral stroke extent. When pushing on the thorax with a brush, the experimenters could make the wings switch between the these two positions. During the switch, the wings moved with the changes in wing pitch angle one would observe in flight. This bifurcation led to the hypothesis that the wing hinge is a bi-stable mechanism and that contraction of the power muscles would switch the wing between the two stable points. This so-called *click-mechanism* reveals a process of mechanical encoding by the wing hinge, in which the wing motion pattern is integral to the hinge morphology. In the click-mechanism, the power muscles provide the force required to switch between the two stable states. However in (MIYAN and EWING, 1985) argued that the click-mechanism was an artefact of the CCl_4 anaesthesia. Due to the heightened amount of tension in the flight muscles of the fly under CCl_4 anaesthesia, the wings move between the stroke extremes. Under normal flight conditions, the acceleration profile of the wings differs considerably from the click mechanism, making it unlikely that the wings are moving between two bi-stable states. However, these results do not refute the basic observation of (Boettiger and Furshpan, 1952) that the pattern of wing motion is encoded mechanically by the morphology of the hinge.

A prominent anatomical feature within a fly's wing hinge is the radial stop, a

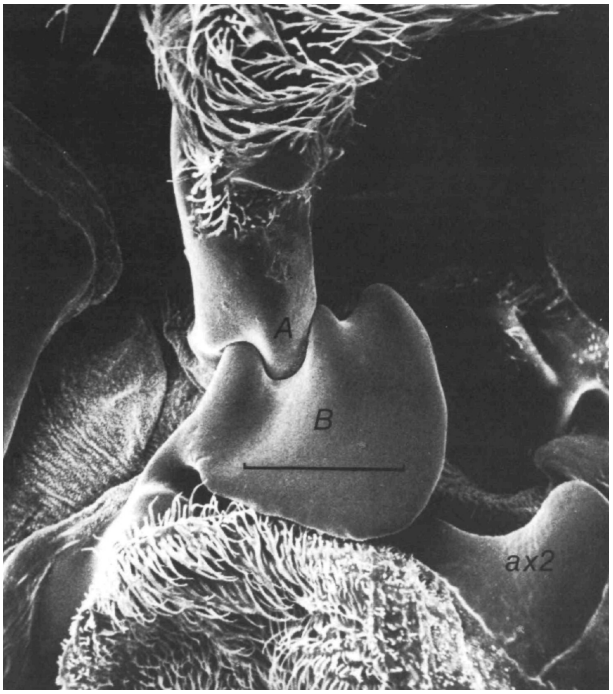

Figure 1.16: SEM image showing the radial stop (A), pleural wing process (B) and the second axillary sclerite (ax2) in *Calliphora erythrocephala*, (PFAU, 1987).

protrusion of the radial vein that has a groove in the middle (Figure 1.16). Beneath the radial stop is a protrusion of the pleural wing process with one ore two grooves, depending on the fly species (Miyan and Ewing, 1985). It has been proposed that the radial stop can engage with the pleural wing process at each wingbeat, by locking into one of the grooves during the downstroke. According to this hypothesis, different configurations in which the radial stop and pleural wing process can be engaged are referred to as *gears* (Figure 1.17). When the radial stop engages with the pleural wing process, the ventral stroke amplitude is significantly reduced (Nalbach, 1989, Walker, Thomas, and Taylor, 2012). The number of gear modes depend on the number of grooves in the pleural wing process. Several studies have reported on the wing gearing mechanism in flies, (PFAU, 1987), (A. Wisser, 1988), (Nalbach, 1989), (Walker, Thomas, and Taylor, 2012), however a high-speed videography study (M.H. Dickinson, personal observations) of wing gearing in *Drosophila* could not observe engagement of the radial stop with the pleural wing process during long tethered flight bouts. It remains unclear to what extent wing gearing is used by different fly species and under which flight conditions.

An intriguing part of the wing hinge is the basalare sclerite; a nail-like structure wedged in the episternal cleft (Figure 1.18). During the wingbeat, the episternal

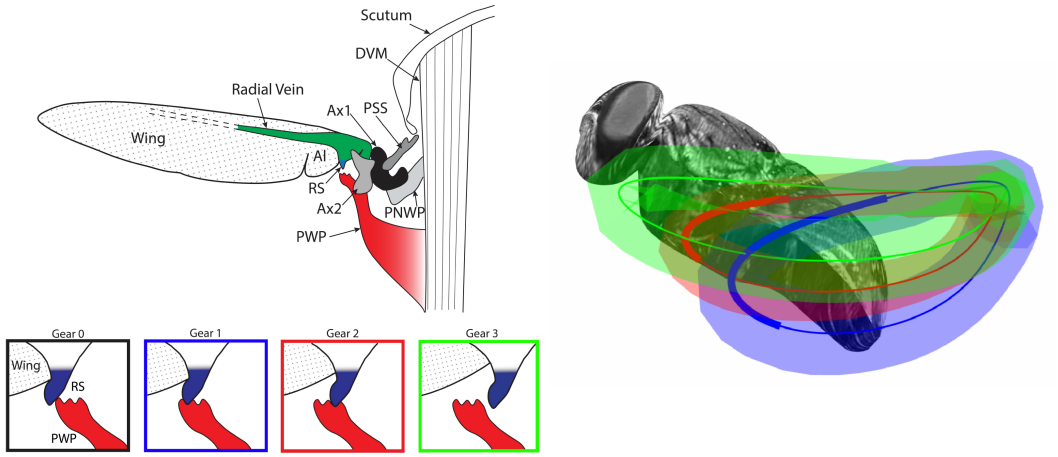

Figure 1.17: Wing gearing in *Calliphora vicina*, four gear modes have been identified: gear 0 brings the wing motion to a complete stop, gear 1 (blue) results in the lowest amplitude, gear 2 (red) an intermediate amplitude, and gear 3 (green) the highest amplitude (Nalbach, 1989, Page, 2018).

cleft opens and closes as the anterior and dorsal episternum plates slide over each other. The shape of the head of the basalare is such that the basalare lever arm will swing back and forth within the thorax, when the episternal cleft opens and closes.

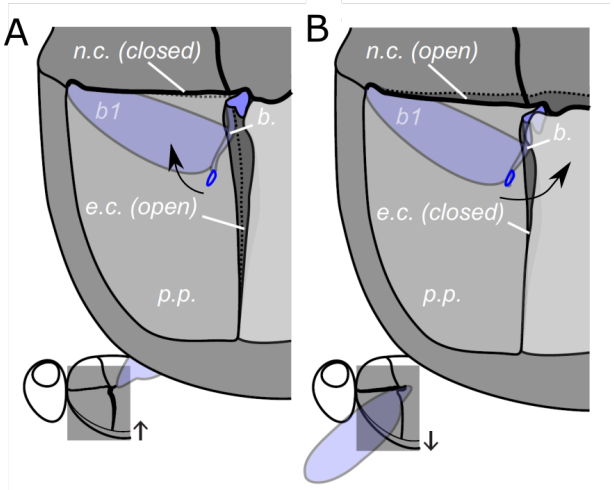

Figure 1.18: Basalare sclerite (b) with b_1 muscle and pleural plate (p.p). A: Basalare at the start of the downstroke, the episternal cleft (e.c.) is open, and the notopleural cleft (n.c.) is closed. Motion of the basalare during the downstroke is indicated by the black arrow. B: Basalare at the start of the upstroke, the e.c. is closed and the n.c. is open. Adapted from (Walker, Schwyn, et al., 2014).

Attached to the basalare is the *basalare tendon* which ends at the radial vein of the

wing (Figure 1.19). The location of the basalare tendon on the basalare lever arm helps to transmit forces to the radial vein. A total of three muscles attach to the basalare; I will discuss in the next section how these muscles affect the motion of the basalare.

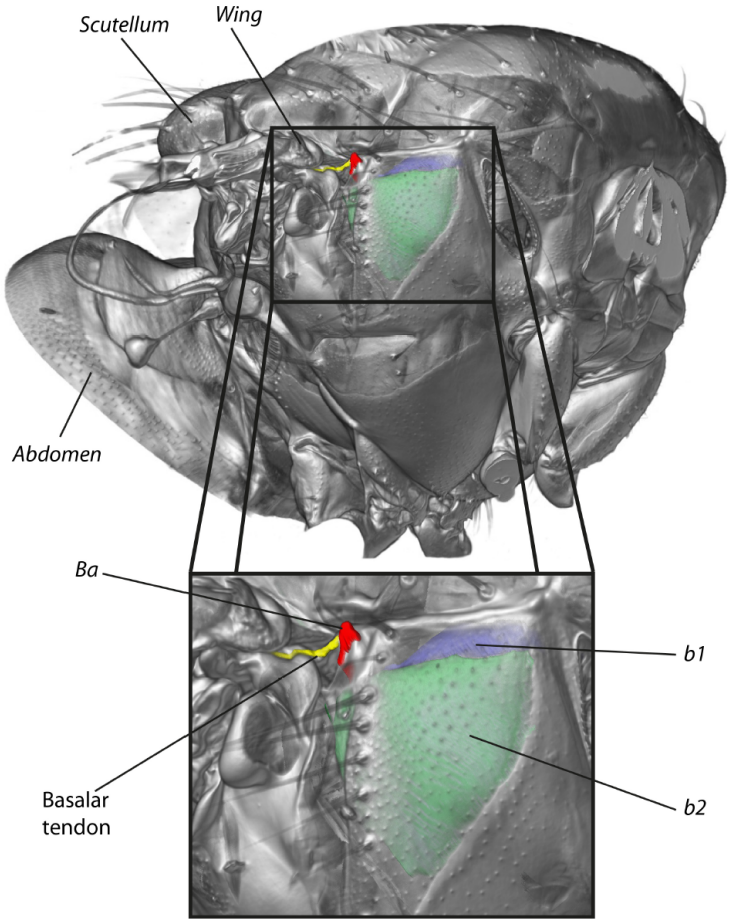

Figure 1.19: μCT image of the basalare (Ba) and basalar tendon with two basalare muscles, b_1 and b_2 (Page, 2018).

Anterior to the scutum is the *scutellum*, a saddle-like structure that is coupled to the scutum via a flexible part of the cuticle that functions as a hinge (Figure 1.20). The scutellum protrudes in the scutellar lever arm on both sides of the body, forming a horseshoe-shaped structure that functions to transmit forces from the DLMs to the wings on both sides of the body. During the wingbeat, the scutellar lever arm rotates up and down in the same direction as the scutum. The scutellar lever arm splits into two branches, which each act on a different axillary sclerite: the first and fourth axillary sclerites (Figure 1.21). During the downstroke the scutellar lever arm will push the first axillary sclerite anterior and dorsal while during the upstroke the

lever arm will pull posterior and ventral on the first axillary sclerite. It is difficult to derive the translation and rotation of the first axillary sclerite as a consequence of the scutellar lever arm motion, as the sclerite has multiple possible rotation points. It is speculated that the motion of the scutellar lever arm rotates the radial vein along its longitudinal axis via rotation of the first and second axillary sclerite.

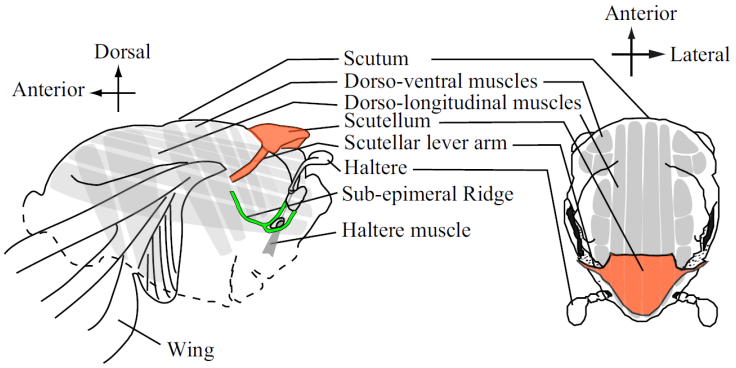

Figure 1.20: Scutellum and the scutellar lever arm in orange (Deora, Singh, and Sane, 2015).

The second branch of the scutellar lever arm is the fourth axillary sclerite. In this section, I will describe a hypothesis formulated by (Miyan and Ewing, 1985), which is based on anatomical observations. However, more research is required to confirm this hypothesis. The fourth axillary sclerite consists of two tubular structures that are coupled by flexible membranes to the cuticle of the fly. At the top of the fourth axillary sclerite, the two tubes fuse into the scutellar lever arm. Over the length of the fourth axillary sclerite, the two tubes form a helical shape. At the bottom of the fourth axillary sclerite, the two tubes merge and fuse into the third axillary sclerite. The helicity of the tubes turns the sclerite into a spindle that rotates the third axillary sclerite anterior under tension and posterior under compression. The third axillary sclerite acts on the radial vein of the wing but is also coupled to the posterior veins of the wings. Rotation of the third axillary sclerite is therefore expected to affect the orientation of the posterior part of the wing and is likely to affect the wing pitch angle and shape of the wing during the wingbeat.

Summarizing, there are four possible *mechanical pathways* through which the contraction of the power muscles is transformed into wing motion. Again, the mechanical pathways are based on the hypotheses of Miyan and Ewing (1985). Figure 1.22 shows the four mechanical pathways in a simplified schematic of the wing hinge. The first mechanical pathway relies on the opening and closing of the episternal

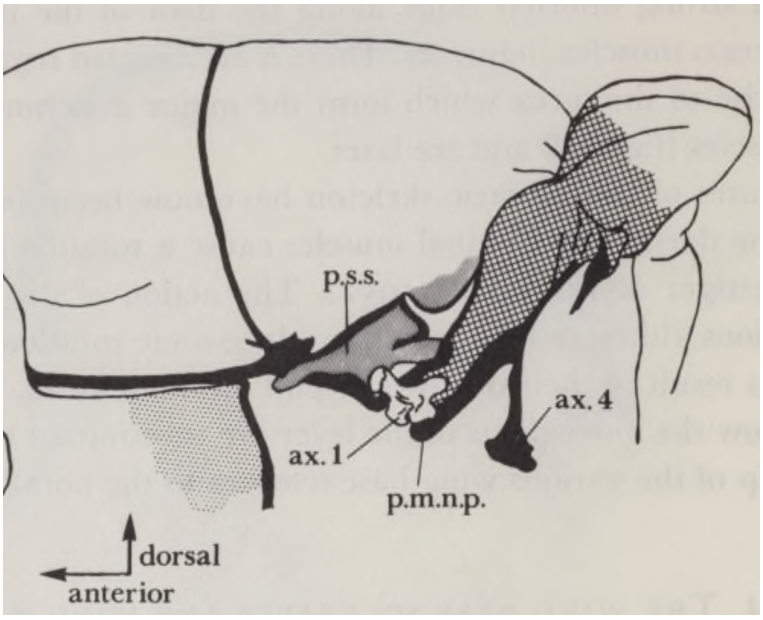

Figure 1.21: Scutellar lever arm and the first (ax1) and fourth (ax4) axillary sclerites. The scutellar lever arm acts on ax1 via the post-median notal process (p.m.n.p.) and the scutum via the parascutal shelf (p.s.s.) (Miyan and Ewing, 1985).

cleft and the oscillatory motion of the basalare sclerite which is wedged in the cleft. Motion of the basalare sclerite is transferred to the radial vein of the wing via the basalare tendon. The second pathway starts with the up-and-down motion of the scutum, which subsequently transfers this motion to the radial vein via the parascutal shelf and the first axillary sclerite. This pathway is thought to be responsible for the main back-and-forth motion of the wing during the wingbeat. The third pathway starts with the upward motion of the scutum during the downstroke, which causes the scutellum to rotate clockwise and the scutellar lever arm to move anterior. A notch in the scutellar lever arm, the post-median notal process, interacts with the first axillary sclerite and makes it rotate. During the upstroke the opposite happens and the scutellar lever arm moves posterior and results in rotation of the first axillary sclerite in opposite direction. The fourth pathway affects the posterior part of the wing via the third and fourth axillary sclerites. Due to the motion of the scutellar lever arm, the spindle in the fourth axillary sclerite comes under compression during the downstroke and under tension during the upstroke. The axial rotation of the fourth axillary sclerite makes the third axillary sclerite, which is coupled to the fourth, moves anterior during the downstroke and posterior during the upstroke.

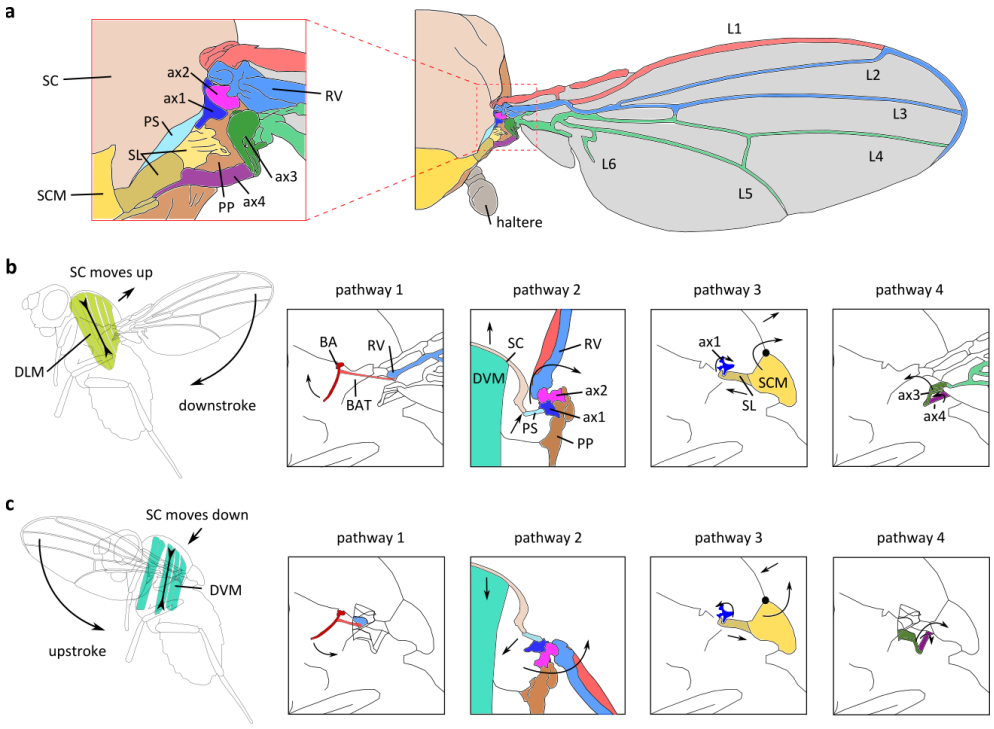

Figure 1.22: Four mechanical pathways that transform thorax deformation into wing motion. A: Top view of the thorax, hinge, and wing of *Drosophila melanogaster*, with the scutum (SC), axillary sclerites (ax1-4), parascutal shelf (PS), radial vein (RV), pleural process (PP), scutellar lever arm (SL), scutellum (SCM), haltere, and wing veins L1-6. B: Pathways during the downstroke. 1: basalare (BA) moves anterior and pulls on the basalare tendon (BAT) which is connected to the RV. 2: the SC moves up and pulls the PS upwards which causes the ax1 and ax2 to tilt on the PP. The RV is rigidly connected to ax2 and will move the wing downwards during the downstroke. 3: contraction of the DLMs makes the SCM turn clockwise and moves the SL anterior. A notch in the SL interacts with ax1 and rotates the sclerite clockwise. 4: with the anterior motion of the SL, ax4 comes under compression resulting in an anterior rotation of the ax3. C: Pathways during the upstroke. 1: the BA moves anterior and tension on the BAT is released. 2: the SC moves downward, pushing on the PS, causing ax1 and ax2 to rotate anti-clockwise and generating the upward motion of the RV. 3: anti-clockwise rotation of the SCM moves the SL posterior and causes anti-clockwise rotation of ax1. 4: posterior motion of the SL puts ax4 in tension and causes posterior rotation of ax3.

1.5 Flight behavior is controlled by a sparse set of steering muscles

As described above, the intricate wing motion pattern of flies is mechanically encoded in the wing hinge and is generated by the asynchronous contraction of the power muscles. Although the motor neurons of the power muscles can control wingbeat frequency by regulating the level of Ca^{2+} ions in the muscles, it can not

generate asymmetric changes in wing motion. A set of small *steering muscles* that act on the sclerites can change the configuration of the wing hinge and thereby change how the deformation of the thorax is converted into wing motion. In *Drosophila* there are a total of 17 steering muscles per side, 12 muscles act directly on the sclerites of the wing hinge and the remaining 5 muscles can change the stiffness of the thorax. In this section I will focus on the 12 direct steering muscles as these muscles are responsible for the rapid aerial maneuvers which are characteristic for fly flight.

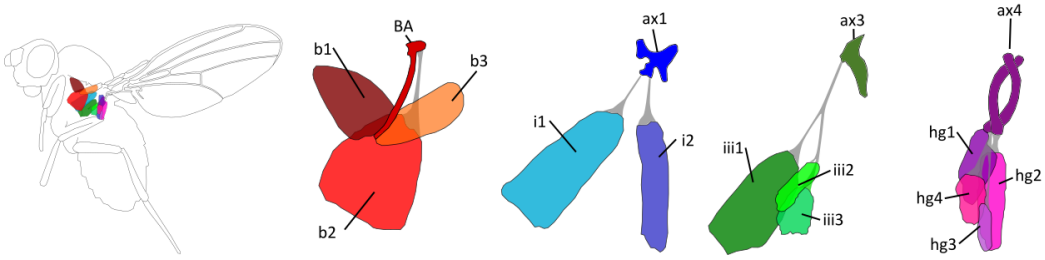

Figure 1.23: Schematic overview of the direct steering muscles and their orientation within the thorax. The muscles are grouped per sclerite: basalare (BA), first axillary (ax1), third axillary (ax3), and fourth axillary (ax4). Tendons are depicted in grey.

The 12 direct steering muscles act on four sclerites in the wing hinge: the basalare, first, third, and fourth axillary sclerites. Three muscles are attached to the basalare sclerite: b_1, b_2, and b_3 (Figure 1.23). Two muscles attach to the first axillary sclerite: i_1 and i_2. The third axillary sclerite has three muscles attached via a single shared tendon: iii_1, iii_2, and iii_3. Finally, the fourth axillary sclerite has four muscles attached: hg_1, hg_2, hg_3, and hg_4. The hg-nomenclature is a remnant of the German term for the fourth axillary: *hinter Gelenkforsatz*.

Each steering muscle is innervated by one single motorneuron, which means that the activity of just 24 motorneurons, 12 on each side, controls all flight behavior. This is in stark contrast to most vertebrate muscles, which can have hundreds of motorneurons per muscle, each controlling just a small fraction of the muscle fibers within the muscle. Having multiple motorneurons per muscle is an advantage for modulating the force output of muscle contraction. Using *population encoding*, vertebrate muscles can turn on only a fraction of the muscle fibers in a muscle to produce a weak force. Insects can modulate muscle output by changing the level of depolarization of the muscle fiber membranes and thereby the amount of Ca^{2+} ions in the cytoplasm.

Although the steering muscles can generate graded muscle twitches, 24 motorneurons to control all flight behavior constitutes an extremely sparse system. Flight behavior ranges from escape maneuvers lasting only 30 ms (Muijres, Elzinga, Melis, et al., 2014) to trimming left and right wing motion for long bouts of straight flight (Leitch et al., 2021). Once wings are formed after eclosion from the pupae, insects cannot repair their wings. Insects will need to compensate any force and torque imbalances due to damage, which often requires highly asymmetric wing motion patterns, (Muijres, Iwasaki, et al., 2017). The mechanical encoding of wing motion in the wing hinge helps to reduce the information that the motorneurons need to transfer to create coordinated and accurate wing motion patterns. Understanding how flies can control such a wide variety of flight scenarios with a small number of neurons could provide valuable insights for robust flight control with limited computational resources.

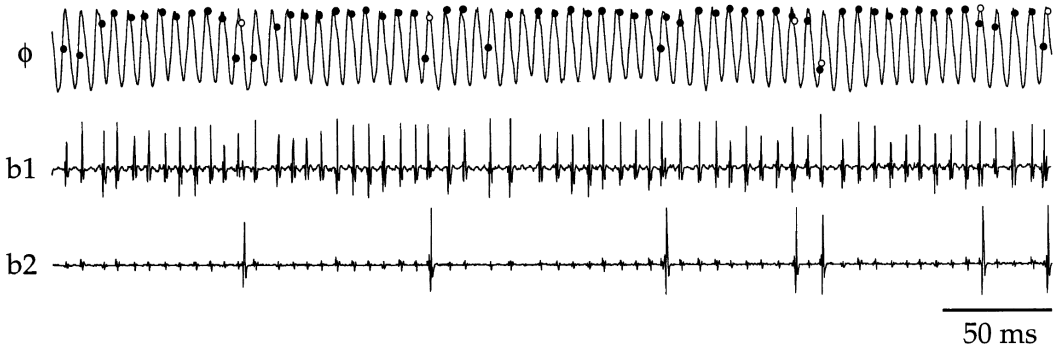

Figure 1.24: Simultaneous electrophysiology recordings from the b_1 and b_2 muscles in *Calliphora vicina*. The stroke amplitude, ϕ, of the wingtip is displayed such that dorsal stroke reversal correspond to the maxima and ventral stroke reversal to the minima and the action potentials of the b_1 and b_2 muscles are shown by solid and open dots respectively. Amplified voltage recordings for the b_1 and b_2 muscles are shown on the same time-scale (Tu and Dickinson, 1996)

One of the most accurate techniques to study steering muscle activity is *electrophysiology*. Two sharp electrodes are inserted into the ventral side of the thorax, such that the electrodes are not in the path of the wing. One electrode functions as the ground or reference while the other electrode is inserted into the steering muscle. The number of steering muscles that are accessible is limited, as some muscles are too close to the wing hinge such that electrode placement would obstruct wing motion. Electrophysiological recordings from the b_1 and b_2 muscles suggest that, based on their activity, there are two types of muscles: tonic and phasic muscles (Tu and Dickinson, 1996). The b_1 motorneuron typically fires an *action potential*

once per wingbeat, which is *tonic activity* (Figure 1.24). An action potential is a rapid depolarization of the cell membrane followed by a slower return to baseline (around 1 ms duration). The b_2 muscle is usually quiescent but can turn on in bursts of activity, i.e. *phasic activity*. The different activity patterns result in differences in muscle physiology. Tonic activity of the b_1 muscle requires the rapid re-uptake Ca^{2+} ions and a significant fraction of the muscle cells consists of sarcoplasmic reticulum and mitochondria. The strength of b_1 muscle twitches is therefore relatively weak. Phasic muscles, like the b_2 do not have to fire every wingbeat and therefore have a larger fraction of contractile fibers.

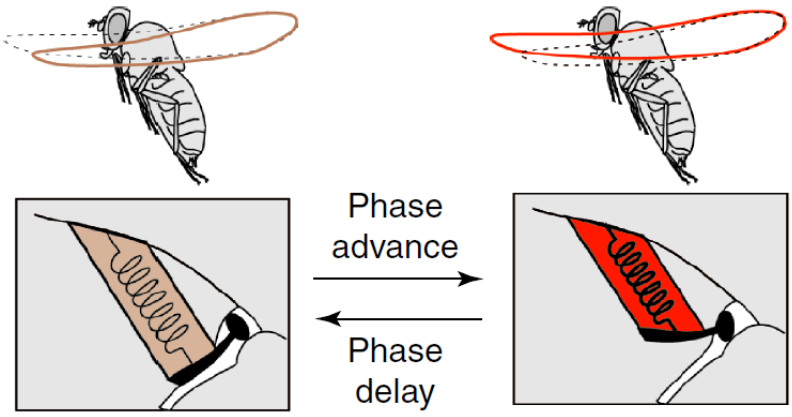

Figure 1.25: Delayed firing of the b_1 motorneuron results in a lower stiffness of the b_1 muscle, posterior motion of the basalare and decrease in stroke amplitude. Advanced firing of the b_1 motorneuron results into a higher stiffness of the muscle, anterior motion of the basalare and an increase in stroke amplitude (Michael Dickinson, 2006).

In electrophysiology recordings from the b_1 muscle (Tu and Dickinson, 1994) found that the timing of *action potentials* from the motor neuron within the wingbeat is important. During flight, the b_1 muscle fires typically once per wingbeat at dorsal stroke reversal. The b_1 motorneuron can advance or delay firing an action potential. This phase shift has an effect on wing motion: advanced firing of the motorneuron results in an increase in wing stroke amplitude and delayed firing results in a decrease in amplitude (Figure 1.25). In an experiment that measured the work output of the b_1 steering muscle under different frequencies of electrode activation, Tu and Dickinson, 1994, found that the muscle performs negative work at activation frequencies of the same order of magnitude as the wingbeat frequency. This means that activation of the muscle does not result in the generation of positive mechanical work, but rather an increase or decrease in the dynamic stiffness of the

muscle depending on the phase. The b_1 muscle can thus be seen as a *programmable spring* with the activation phase controlling the spring stiffness. Phasic muscles, such as the b_2, presumably operate like conventional muscles and a single action potential in the motorneuron results in a muscle twitch that generates positive work.

Besides the b_1 and b_2 motorneurons, there are several other steering muscles that can be accessed by electrodes during tethered flight, at least in larger fly species such as *Calliphora*: i_1, iii_1, and iii_2, (Balint and Dickinson, 2001). Simultaneous recording of muscle activity using electrodes and the wing motion with high-speed cameras has revealed several correlations (Figure 1.26). From the high-speed videos, the 3D wingtip trajectory can be reconstructed and the instantaneous wingtip position can be described by two angles: the stroke angle ϕ within the strokeplane and the deviation angle θ that gives the elevation angle above the strokeplane. The major effect of b_1 and b_2 activity seems to be the increase of the deviation angle during the downstroke, where $b2$ activity results in a larger increase than $b1$ activity. An interesting pattern is observed between the i_1 and the iii_2 muscles: when one of the two muscles is active the other is quiescent. Multiple muscles show increased activity when the iii_2 muscle is active, namely the b_1, b_2, and iii_1 muscles. Two different muscle activity *modes* have been identified: mode 1 is the increased activity of the i_1 muscle and decreased activity of the iii_2 muscle, and mode 2 is an increased activity of the iii_2 muscle and quiescence of the i_1 muscle. The effect of mode 1 on wing motion is a decrease in ventral stroke amplitude, while mode 2 has the opposite effect: an increase in ventral stroke amplitude.

The electrophysiology recordings of 5 of the 12 steering muscles suggest interesting patterns, however one needs to know the simultaneous activity patterns of all 12 muscles to decipher how the steering muscles control flight behavior. A relatively recent technique, developed by a former post-doc in the Dickinson lab; Theodore Lindsay, makes use of the powerful tool of *Drosophila* genetics to visualize steering muscle activity (Lindsay, Sustar, and Dickinson, 2017). Using the genetically-encoded fluorescent calcium indicator (GCaMP) it is possible to measure the calcium concentration in the muscle via fluorescence intensity. The GCaMP molecules that are expressed in the steering muscles, are bright enough to see through the cuticle of the fly. With GCaMP it is possible to simultaneously image the activity of all 12 steering muscles in a tethered flying fly via an epi-fluorescence microscope.

As was found in the electrophysiology experiments, the muscle fluorescence showed that there are roughly two physiological classes of steering muscles: tonic and phasic.

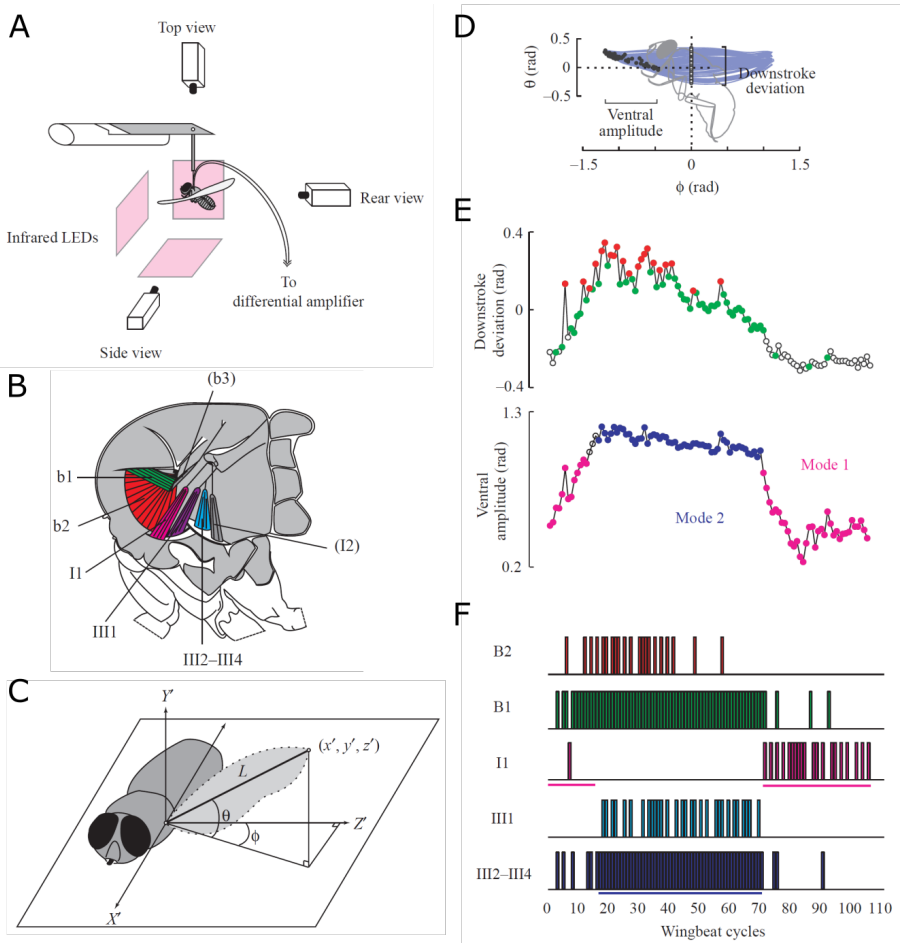

Figure 1.26: Simultaneous recording of wing motion, flight forces, and electrical activity of 5 steering muscles. A: Experimental setup with a blow fly, *Calliphora vicina*, tethered to a piezo-electric crystal that can detect vertical flight forces. A total of 5 pairs of electrode wires are connected to a differential amplifier. Three high-speed cameras record wing motion with infrared backlighting from three orthogonal angles. B: Overview of the 5 muscles that are being recorded and 2 muscles (grey) that are not being recorded. C: Stroke angle, ϕ, and deviation angle, θ, relative to the strokeplane (x, y). D: Wing kinematic keypoints: ventral amplitude and downstroke deviation. E: Downstroke deviation: b_1 activity is marked in green and b_2 in red. Ventral amplitude: mode 1 is marked in pink, mode 2 in blue. F: Activity of the 5 steering muscles during the experiment, action potentials are marked by the vertical bars (Balint and Dickinson, 2004).

Each sclerite has at least one tonic and one phasic muscle attached, which suggests that the ability to actuate the sclerites rapidly (phasic) and for longer durations (tonic) is important for flight control. The tonic muscles are the b_1, b_3, i_2, iii_3, and hg_4 muscles and the phasic muscles are the b_2, i_1, iii_1, iii_2, hg_1, hg_2, and hg_3 muscles.

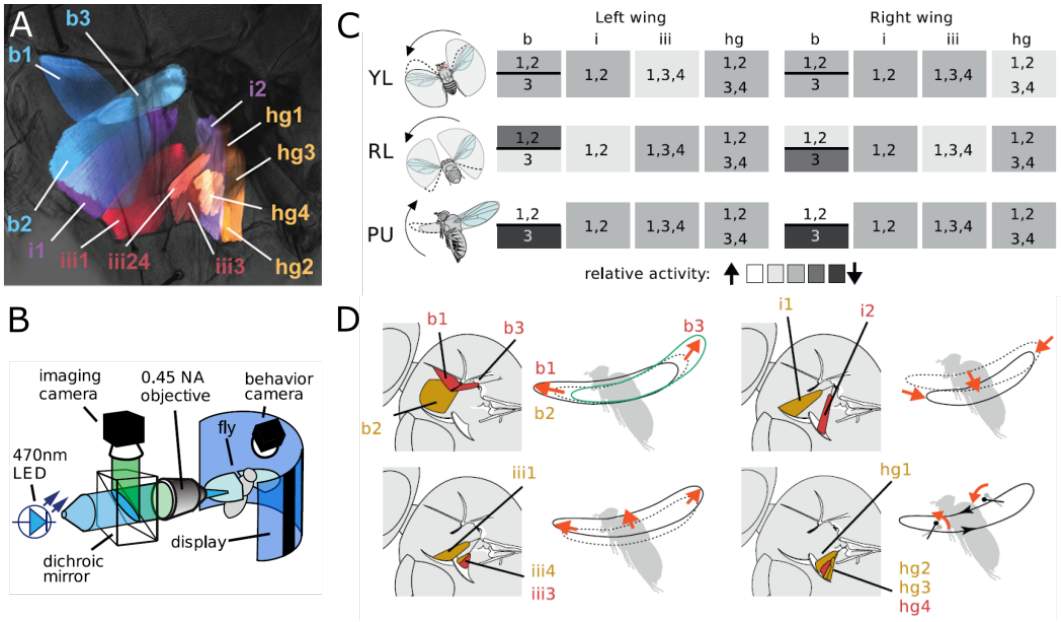

Figure 1.27: Imaging of muscle activity via GCaMP fluorescence. A: Confocal microscopy image of the steering muscles with colors based on the sclerite: basalare (blue), first axillary (purple), third axillary (red), and fourth axillary (yellow). B: Schematic overview of the setup with an epi-fluorescent microscope, LED display and a behavior camera measuring wing stroke amplitude. C: Muscle activity for the left and right wing in response to: yaw left (YL), roll left (RL), and pitch up (PU) stabilization reflexes. Muscle activity is shown for the left and right wing, grouped per sclerite, and a gray-scale color coding indicates relative muscle activity (white = more active, black = less active). D: Hypothesized effect on wing motion for each sclerite. The baseline wing kinematics have been marked by the dotted lines while the solid lines and orange arrows show the effect of muscle activation. In the case of the b_3 muscle a green line shows the wing motion pattern upon activation. Depicted muscles are either tonic (red) or phasic (yellow) (Lindsay, Sustar, and Dickinson, 2017).

By presenting rotating patterns on a LED screen surrounding the tethered fly, Theodore Lindsay elicited stabilization reflexes while recording the muscle activity of the steering muscles and wing stroke amplitudes of the left and right wing (Figure 1.27). Rotation axes of the LED patterns were around the roll, pitch, and yaw axes, and the stabilization reflex consists of changes in wing motion that rotate the fly so as to follow the direction of the visual stimulus. The muscle activity of the stabilization reflexes was very stereotypical and showed that different sclerites are involved in different maneuvers. For example, the *yaw left reflex* shows an increase in activity of the *iii* muscles of the left wing and a decrease in *hg* muscle activity of the right wing. The *roll left* reflex shows decreased activity of the b_1 and b_2

muscles and an increase in activity of the b_3 and i muscles for the left wing. For the right wing, the b_3 muscle has decreased activity while the b_1, b_2, and iii muscles have increased activity. With the exception of the basalare muscles, the muscle activity of the stabilization reflexes seems to be grouped per sclerite. This led the authors to speculate that different components of the wing motion are controlled by different sclerites. According to this hypothesis, the b-muscles control the ventral and dorsal stroke amplitude, the i-muscles decrease the deviation angle of the wing when activated, the iii-muscles increase the deviation angle upon activation, and the hg-muscles control the angle-of-attack of the wing.

Besides actuating wing motion during flight, the steering muscles are used for non-flight behavior as well. The iii_1 muscle is crucial for wing extension and folding, and therefore shows strong activity during flight starts and stops. When flies are not flying, they spend a significant amount of time grooming, i.e. cleaning their eyes and wings. Some of the steering muscles are used during the grooming of the wings, lowering the wing to make it accessible to the legs of the fly. Another non-flight behavior that involves the steering muscles is *singing*. Part of the mating ritual of flies is the *courtship* phase, in which the male fly chases a female fly and extends one of his wings outward, (Bastock and Manning, 1955), (Agrawal, Safarik, and Michael Dickinson, 2014). Once the wing is extended, the fly uses its power and steering muscles to vibrate the wing at approximately 200 Hz with a small amplitude, creating a *song* that is audible to the female fly, (A. W. Ewing and Bennet-Clark, 1968). Some of the steering muscles, b_1 and b_2, are not active, while other steering muscles are always active during song, b_3, i_1, hg_{1-4}, (O'Sullivan et al., 2018). Variation in the song is achieved by the remaining muscles: i_2, iii_{1-3}. The multi-functionality of the steering muscles is a common phenomenon in nature and has to be kept in mind when analyzing a biological system.

1.6 Research outline

The research presented in this thesis started with the hypothesized effect of steering muscle activity on wing motion proposed by Theodore Lindsay, (Lindsay, Sustar, and Dickinson, 2017). Although ventral stroke amplitudes of the left and right wings were recorded by a behavioral camera, the limited temporal resolution (30 Hz) and the projection of complex 3D wing motion into one parameter, provided insufficient data to confirm the hypothesis. To test the proposed effects of steering muscle activity on wing motion, I built a setup that combined the GCaMP fluorescence imaging with a high-speed videography setup that images the wing motion from

three orthogonal directions. The large difference in temporal resolution of muscle imaging and high-speed videography required the setup to be highly automated. Chapter 2 describes the details of the experimental rig and the collected dataset.

Before analyzing the dataset, I needed to extract the left and right wing pose from the high-speed videos. As the three high-speed cameras were recording at 15,000 frames per second, the total number of high-speed video frames is more than 23 million. It is obvious that an automated algorithm is required to extract wing pose from the high-speed videos. There were several existing methods to extract the wing pose from multiple camera views, however an artefact of tethered flight, clap-and-fling, would require too many user interventions to be practical. I, therefore, developed a novel tracking algorithm, named FlyNet, that combined pose prediction by a neural network and pose refinement via 3D model fitting. In Chapter 3, I will discuss the workflow of FlyNet.

In Chapter 4, the recorded muscle activity and wing kinematic traces, tracked by FlyNet, are combined in a coupled dataset. This coupled dataset is subsequently used to train a convolutional neural network to predict wing motion from muscle activity patterns. The trained network can accurately predict wing motion and is used to study how steering muscle activity affects wing motion.

Wing kinematic patterns predicted by the trained neural network were replayed on a dynamically scaled flapping wing robot. The dynamically scaled robot can measure aerodynamic forces and torques, and in combination with the inertial forces and torques computed with the Newton-Euler equations, a map from steering muscle activity to control force-torque was created, as shown in Chapter 5.

In Chapter 6, the map of Chapter 5 is incorporated in a state-space system of fly flight. The state-space system is subsequently embedded in a control loop. By using model predictive control, various free flight maneuvers were simulated. The conclusions of the research in Chapters 2-6 will be discussed in Chapter 7.

Chapter 2

SIMULTANEOUS RECORDING OF MUSCLE ACTIVITY AND WING MOTION

The mechanical complexity of the wing hinge makes it impossible to infer the effect on wing motion of the steering muscles from anatomy alone. Experiments in which steering muscle activity was measured using electrodes while simultaneously imaging the wing motion with high-speed cameras yielded data on the changes in wing kinematics elicited by some muscles. Unfortunately, many of the muscles are situated at locations which are impossible to access with electrodes during flight.

With the development of *Drosophila melanogaster* as a model-species in genetics research and neuroscience, a new technique to measure steering muscle activity in flight has become available: Genetically Encoded fluorescent Calcium-Indicators (GECI). Whenever a muscle fiber gets activated by a motor neuron, its membrane potential rises and calcium ions, Ca^{2+}, are released from the sarcoplasmic reticulum into the cytoplasm. The Ca^{2+} ions will bind to actin filaments, and by doing so enable the contraction of the muscle fibers. After contraction of the muscle fibers, ion pumps transfer the Ca^{2+} ions back into the sarcoplasmic reticulum. During the course of a muscle twitch, the Ca^{2+} concentration will rise and decay with the contraction and relaxation of the muscle.

Calcium ion concentration within the cytoplasm is a good measure of muscle force. By genetically expressing GECI in the steering muscles of the fly, it is possible to image the activity of all steering muscles simultaneously. The changes in fluorescence of GECI molecules during a muscle twitch are bright enough to be captured via an epi-fluorescent microscope through the *cuticle* of a flying fly. With this technique, I was able to image the input and output of the wing hinge simultaneously. The control input to the wing hinge is the activity pattern of the 12 direct steering muscles and the output is the 3D motion pattern of the wing. This chapter describes the experimental setup and the methods that were used to capture a large dataset of muscle activity and coupled wing kinematics of flies under a wide variety of flight conditions.

2.1 Imaging muscle activity in flying flies

One of the most powerful tools in *Drosophila* genetics is the GAL4-UAS system. In order to express reporter molecules such as the Green Fluorescent Protein (GFP) or GECI in specific cells, two genetically modified fly mutants are required. The first genetic variant, or *line*, has a mutation that expresses the yeast protein GAL4 in a specific group of cells. The second line has a UAS region in front of the reporter gene that the experimenter wishes to express. When the GAL4 and UAS-lines are crossed, the progeny will express the reporter gene in the specific cells (Figure 2.1). During development, the GAL4 protein will be produced in the desired cells. The GAL4 protein binds to the UAS region of the genome and subsequently activates the transcription of the gene following the UAS site. Transcription of the gene results in the production of the desired protein in the specified cells.

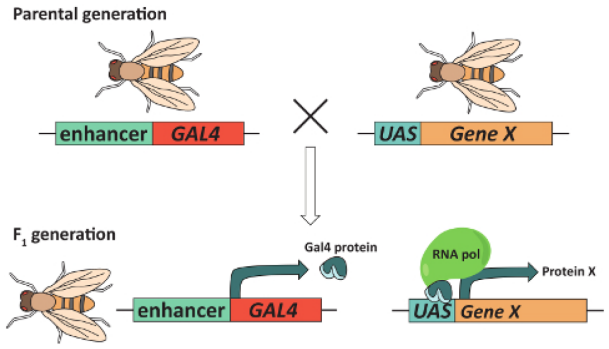

Figure 2.1: Crossing a GAL4 line with a UAS line results in expression of gene X in the progeny. During development the GAL4 region produces the GAL4 protein which can bind to the UAS region. The binding of GAL4 to UAS results in the recruitment of RNA polymerase which reads the DNA strand of gene X and subsequently produces protein X (Kelly, Elchert, and Kahl, 2017).

Using the GAL4-UAS system it is possible to express GECI molecules in the cells of the steering muscles. Geneticists have developed thousands of different GAL4-lines that have GAL4 present in specific groups of cells. This specificity varies but can be sparse enough to target individual neurons or muscles. All direct steering muscles and some leg, neck, and haltere muscles can be targeted by the GAL4-line $R22H05$. I crossed $R22H05$ with the UAS-line that expresses the GECI protein: *GCaMP7f*. The progeny of the $R22H05 \times GCaMP7f$ cross has $GCaMP7f$ in the steering muscles.

The synthetic molecule GCaMP is a combination of three proteins: circularly permuted GFP, calmodulin, and M13. GFP is a protein that turns ultraviolet and

blue light into green light through fluorescence. The gene for producing GFP was isolated from a jellyfish and is widely used in genomics. Calmodulin is a protein that is ubiquitous in all eukaryotes and is crucial in intracellular signaling and smooth muscle contraction. Finally, the M13 protein is a peptide in the myosin light-chain kinase. The protein myosin is the active and moving part during muscle contraction. One of the steps in the myosin light-chain kinase is the binding of calmodulin to the M13 peptide in the myosin light-chain. Researchers have managed to genetically isolate the M13 peptide from the myosin protein. The GCaMP protein was created by merging the genetic codes for GFP, calmodulin, and M13 and inserting this gene into the *Drosophila* genome, (Nakai, Ohkura, and Imoto, 2001).

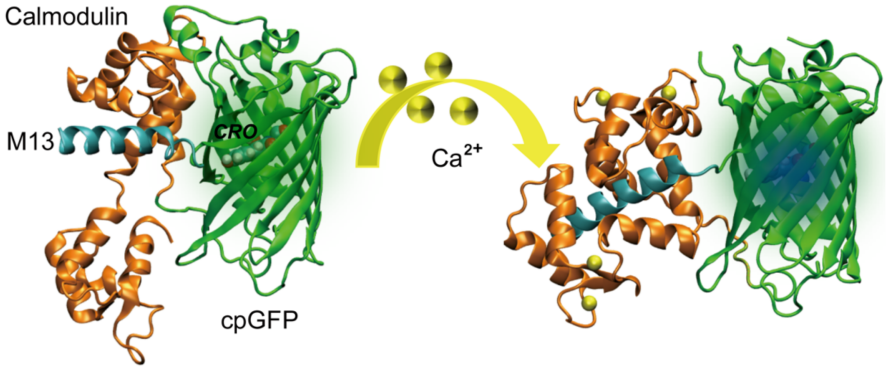

Figure 2.2: The GCaMP molecule consists of calmodulin, M13, and GFP proteins. When the calmodulin protein binds to 4 Ca^{2+} ions, it undergoes a conformational change and acts on the M13 and GFP molecules, changing the exposure of the chromophore in the GFP molecule and increasing its fluorescence (Tang and Fang, 2019).

The Ca^{2+} concentration within a resting muscle cell is low: $10^{-8}M$. When a muscle gets activated, the sarcoplasmic reticulum releases Ca^{2+} into the cytoplasm and the calcium concentration rises to $10^{-6}M$. Calmodulin has 4 sites that can bind with Ca^{2+} ions. The probability that all 4 sites bind with calcium ions increases with Ca^{2+} concentration. When 4 calcium ions bind to calmodulin it undergoes a conformational change and will interact with the GFP protein through the M13 peptide. Circularly permuted GFP has low fluorescence in its resting state because the chromophore is protonated due to interaction with water molecules. Water molecules can access the chromophore as the surrounding alpha helix is porous due to the circular permutation. When calmodulin undergoes a conformational change it will tighten the alpha helix, leading to the deprotonation of the chromophore and a

rapid increase in fluorescence. The chromophore gets excited by a peak wavelength of 480 nm (blue light) and has a peak emission wavelength of 510 nm (green light).

Over the last two decades GCaMP has been optimized via targeted mutagenesis to fluoresce brighter, faster, and with a higher signal-to-noise ratio (Akerboom et al., 2012). The fluorescence signal of GCaMP has a characteristic *kernel* in response to a single muscle twitch or an action potential in a neuron: a fast exponential increase in fluorescence (ON) followed by a slow exponential decrease in fluorescence (OFF) (Figure 2.3). For our experiments, I selected the GCaMP version with the fastest ON-OFF characteristics: GCaMP7f. GCaMP7f has a half-rise time of 75 ms and a half-decay time of 580 ms in dissociated neurons, (Dana et al., 2019). GCaMP fluorescence is, however, dependent on the thermal and chemical properties of the cells in which they are expressed and the temporal characteristics will vary between species. It takes approximately 1 second for the GCaMP fluorescence to return to baseline after a single stimulus. This relatively long fluorescence response can result in temporal summation of GCaMP fluorescence as the steering muscles are active on a wingbeat-to-wingbeat basis ($\sim$ 200 Hz).

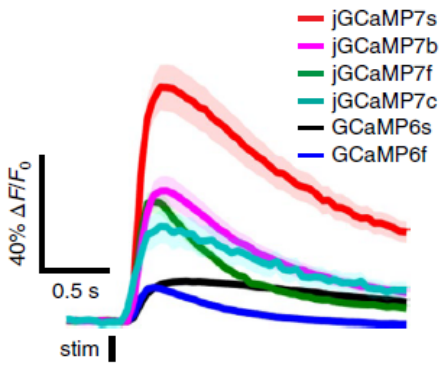

Figure 2.3: Fluorescence response, $\Delta F/F_0$, for different GCaMP versions for one action potential (stim) in a dissociated neuron. $\Delta F/F_0$ measures the instantaneous fluorescence intensity divided by the baseline fluorescence (Dana et al., 2019).

To image GCaMP fluorescence, I built an epi-fluorescent microscope. A 470 nm Light Emitting Diode (LED) provides the excitation photons for GCaMP fluorescence. The excitation light-beam gets reflected via a dichroic mirror onto a 4× objective, which focuses the blue light on the thorax of the fly. When blue light excites the GCaMP molecules inside the steering muscles, fluorescent green light spreads in all directions and some of it travels back through the cuticle and the objective and passes through the dichroic mirror onto an imaging camera (Figure

2.4). Although the captured fluorescent light has a much lower intensity then the excitation light, the dichroic mirror is only translucent to a narrow band of wavelengths centered around 530 nm. This specificity of the dichroic mirror makes it possible to image muscle activity in bright light conditions.

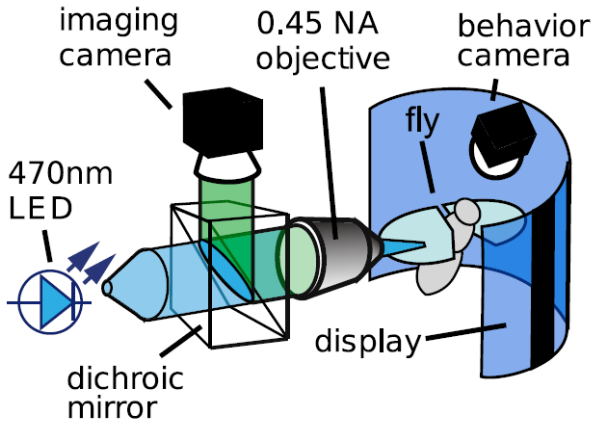

Figure 2.4: Epi-fluorescent microscope for imaging muscle activity in flying flies. Blue light from a 470 nm LED passes trough a dichroic mirror and 0.45 NA objective onto the thorax of a fly. The blue light excites the GCaMP molecules in the steering muscles and the fluorescent green light passes back through the objective and the dichroic mirror onto an imaging camera (Lindsay, Sustar, and Dickinson, 2017).

In order to keep a fly at the focal point of the epi-fluorescent microscope they need to be fixed in place. Flies can be anaesthetized by cooling them down to 4 C° on a cold plate. Using UV-curable glue, the fly is glued to a tungsten tether at the top of the thorax (Figure 2.5). The front and middle pair of legs can obstruct the view of the steering muscles and are therefore cut off at the coxa joint. A fly uses its hind legs to help steering during flight and removal will result in poor (tethered) flight behavior. After removing the front and middle pair of legs and gluing the fly to the tether, flies are given 5 minutes to recover from the procedure. Depending on how well fed and healthy the flies are, they can perform tethered flight for 30 minutes to several hours.

When the fly is positioned in front of the epi-fluorescent microscope the wing will obstruct the view twice during a wingbeat. This is an issue for the automated analysis of the fluorescence signal, as it is difficult to determine when the wing is blocking the view. In order to obtain a stable image, both the blue LED and the camera need to be strobed at the wingbeat frequency of the fly. To measure the phase and frequency of the wing motion I used the so-called *wingbeat analyzer*, (Dickinson,

Figure 2.5: Fruit fly glued to a tungsten tether.

Lehmann, and Gotz, 1993). A wingbeat analyzer consists of an InfraRed (IR) LED, two IR-sensitive diodes, a wing-stroke mask, and an amplifier. The IR LED is positioned above the fly such that the two wings cast a shadow on the wing-stroke mask beneath the fly (Figure 2.6). During the wingbeat, the fly's wings will cast a shadow on different sections of the wing-stroke mask. The wing-stroke mask is designed such that the wing shadow will block more IR-light at the ventral stroke reversal compared to the dorsal stroke reversal. Two IR-sensitive diodes underneath the wing-stroke mask measure the light intensity and the amplifier increases the voltage of the signal (± 10 V).

Using a Teensy microcontroller and the wing shadow signal from the wingbeat analyzer, the blue LED is strobed for 1 ms at dorsal stroke reversal. The Teensy microcontroller also strobes a machine vision camera (FLIR Blackfly S USB3) at half the frequency of the LED, such that each image frame contains two subsequent blue light pulses. Strobing both the camera and the LED has the advantage that the frame-to-frame variation in light intensity is minimal and slight variations in GCaMP fluorescence can be picked up by the camera. A second advantage of the strobing at dorsal stroke reversal is that the thorax deformation is in the same state during each frame. This will ensure that the outline of the steering muscles does not change during flight, irrespective of the wingbeat frequency.

2.2 High-speed videography of wing motion

A fly typically flaps its wings back and forth 200 times per second. To resolve the wing motion with sufficient resolution an imaging camera needs to record at a

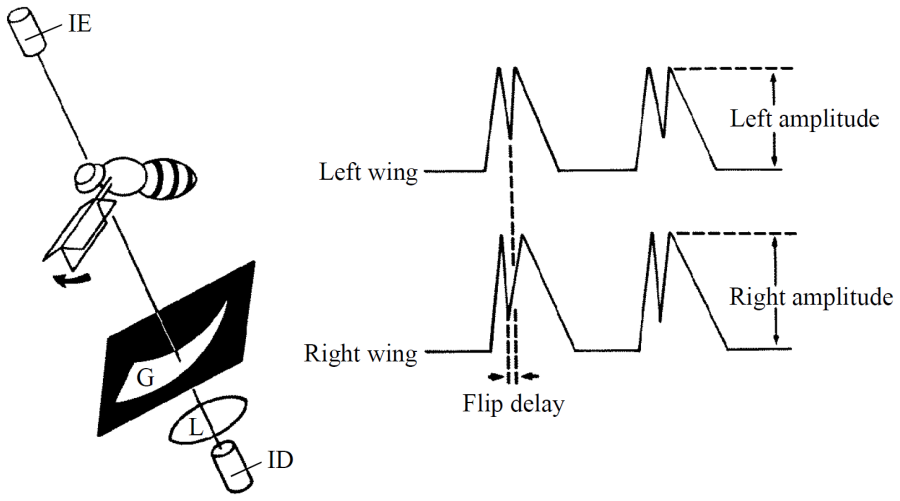

Figure 2.6: Wingbeat analyzer consisting of: an IR LED (IE), wing-stroke mask (G), lens (L), and an IR-sensitive diode (ID). The wing shadow generates a characteristic waveform (double peak) during ventral stroke reversal in which the peak voltage corresponds to the ventral stroke extent (Dickinson, Lehmann, and Gotz, 1993).

minimum of 5000 fps, a frame rate that only high-speed cameras can achieve. The accurate capture of the 3D motion pattern of the fly's wing requires a minimum of three high-speed cameras at orthogonal angles. Sufficient illumination is required to image the fly at these high frame rates. Flies rely extensively on their visual system to stabilize their flight. Too much illumination can blind flies or affect flight behavior. In order to avoid detrimental effects on flight behavior, IR illumination is preferable as flies cannot see at these wavelengths.

The high-speed camera setup I built consists of three Photron SA5 high-speed cameras positioned at orthogonal angles. During the experiment, the cameras are recording continuously at a frame rate of 15, 000 frames per second and with a shutter speed of 33 microseconds (Figure 2.7). The shutter signal of one of the cameras is used to synchronize the other two cameras. Each camera has a telecentric lens (Edmund optics Platinum TL) with 0.5× magnification and a working distance of 175 mm. I backlit the fly using an IR LED (850 nm) with a set of diffuse collimating lenses that provide a uniform background for each of the cameras (Thor labs: 850 nm mounted LED + collimation lens). The combination of a telecentric lens with collimated back-lighting yields sharp images of the contours of the fly. A telecentric lens has its focal point at infinity and the collimating lens projects IR light at infinity. The parallel back-lighting means that most of the light is projected onto the lens and is therefore relatively efficient. An issue with a previous high-speed

videography setup was the excessive heat produced by arrays of IR LEDs, which required continuous cooling of the setup (Muijres, Elzinga, Melis, et al., 2014).

Figure 2.7: Frame triplet taken at: 15000 fps, shutter time $1/30000$ seconds, resolution 256 by 256 pixels, bit-depth 8 bit.

The Photron SA5 camera has a buffer of 8 GigaByte (GB), meaning that at $15,000$ fps, a resolution of 256 by 256 pixels, and a bit-depth of 8 bit, one can record 8 seconds of high-speed video. For the three cameras, it takes approximately 30 minutes to transfer the buffer to a hard-drive. In order to capture muscle activity changes, it is necessary to electronically trigger the recording of high-speed videos. The buffer of each camera was split into 8 partitions of 1 second recording time each. During an experiment, the high-speed cameras are recording continuously and storing the last second of high-speed video to a rolling buffer on a currently active partition. The trigger mode is set to center, meaning that when a trigger pulse comes in, the rolling buffer continues to record for a half a second and then freezes the recording. After triggering, the rolling buffer starts at the next partition until a new trigger pulse arrives. Once all 8 partitions have been filled, the downloading of the videos to a hard-drive automatically starts. To make the video transfer as fast as possible, the videos were saved in the MRAW-format.

To be able to reconstruct 3D wing motion, one needs to know the position and orientation of the high-speed cameras. A 3D camera calibration is performed using a translucent acrylic cube with circles at known locations. During the calibration procedure, a snapshot is taken from each camera view. A custom Matlab script automatically extracts the image coordinates (u, v) of the circle-centers and using the Direct Linear Transformation (DLT), and computes the rotation matrix and translation vector for each camera. The DLT method also computes the magnification and distortion of the lens (Kwon, 1998). As the lens is telecentric, the distortion of the lens is minimal and the projection is uniform over the image plane. One

can use the rotation matrix, translation vector and magnification to compute the *world-to-camera* (w2c) and *camera-to-world* (c2w) projection matrices. The w2c matrix projects 3D coordinates to the image plane (u, v-coordinates) and has the following structure:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} c_u & 0 & 0 & 0 \\ 0 & c_v & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} & T_x \\ R_{yx} & R_{yy} & R_{yz} & T_y \\ R_{zx} & R_{zy} & R_{zz} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{2.1}$$

where c_u and c_v are the scaling factors for the u and v coordinates, R_{ij} the rotation matrix, and T_i the translation vector. The c2w matrix can be obtained by taking the Moore-Penrose pseudoinverse of the w2c matrix:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = pinv\left(\begin{bmatrix} c_u & 0 & 0 & 0 \\ 0 & c_v & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} & T_x \\ R_{yx} & R_{yy} & R_{yz} & T_y \\ R_{zx} & R_{zy} & R_{zz} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \right)^T \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \tag{2.2}$$

In case of the w2c matrix, the 3D coordinate is undetermined and can be anywhere along the line spanned by the focal point and the x, y, z coordinates from the w2c matrix. To reconstruct the 3D point from u, v coordinates a minimum of two camera views is required for triangulation.

2.3 Real-time processing of muscle activity

The limited buffer memory of the high-speed cameras means that the recording of high-speed video needs to be triggered only when muscle activity changes. Therefore continuous monitoring of steering muscle fluorescence is required. The machine vision camera records fluorescence images at half the wingbeat frequency, which can result in frame rates up to 120 fps. A machine vision algorithm that can extract muscle activity at these high frame rates is required to capture the relatively fast rise of GCaMP fluorescence when a steering muscle is activated.

In a previous study, an automated method to extract muscle activity from steering muscles with overlapping Regions Of Interest (ROIs) was developed (Lindsay, Sustar, and Dickinson, 2017). The 12 direct steering muscles of a fly are located relatively close to the exo-skeleton, but some muscles overlap with another in the side view of the thorax. This means that an increase in fluorescence within the

contour of a muscle might not necessarily correspond to that muscle. The *unmixing* method of Theodore Lindsay, resolves this issue by using a 3D model of the steering muscles in combination with an optical model of the lens to simulate the fluorescence image that a muscle activity pattern would generate. By solving the inverse problem, one can obtain the muscle activity from a fluorescence image and distinguish which muscles are active for overlapping ROIs.

The 3D model was obtained by staining the muscles of a fly with *Phalloidin* and capturing a *z-stack* of images ($\Delta z = 10\,\mu m$) using a confocal microscope (Figure 2.8). By tracing the contours of the muscles in each image of the z-stack, a 3D model of each muscle was obtained. The optical model of the lens, the so-called *pointspread* function, was found by placing fluorescent particles of 10 µm diameter on a glass plate in front of the fluorescence microscope. By moving a fluorescent particle in and out of focus, in steps of 10 µm over a range of [−100 µm,100 µm] respectively, one can measure the degree of spreading or blurriness of a point source. Both the muscle model and the pointspread function are converted into 3D voxel space with 10 µm spacing. The fluorescence image of a muscle at a certain activity level can be obtained by convolving the 3D muscle model with the pointspread function.

The muscle fluorescence simulation can be written as a linear system:

$$Au = I_S, \tag{2.3}$$

where A is the muscle fluorescence model, u the muscle activity vector and I_S the vectorized simulated fluorescence image. Muscle activity is obtained by solving the inverse problem:

$$u = pinv(A)^T I_F, \tag{2.4}$$

where I_F is the vectorized fluorescence image and $pinv$ the Moore-Penrose pseudo-inverse. The pseudo-inverse of A can be computed before the experiment, which enables real-time computation of u during an experiment.

At the start of each experiment, the fly is positioned in front of the objective and the translational manipulators move the microscope into focus. The exact orientation of the thorax of the fly is hard to control and it is therefore unlikely that the 3D muscle model matches the view of the microscope. To solve this alignment issue, a Graphical User Interface (GUI) allows the user to manipulate a contour model of the

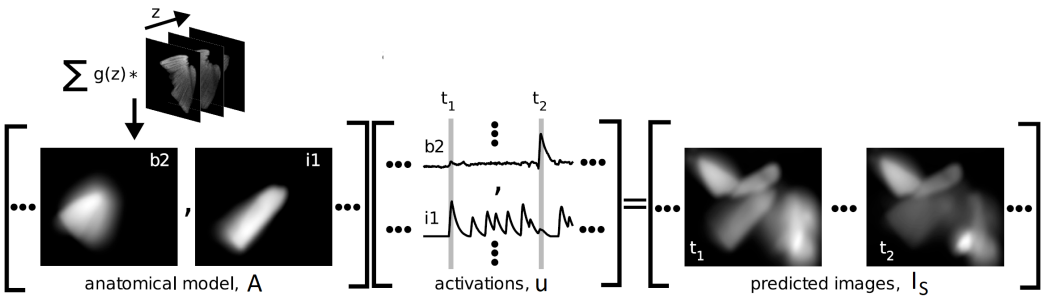

Figure 2.8: A z-stack of Phalloidin stained images and the pointspread function of the imaging lens form the basis of the anatomical model A. Predicted fluorescence images, I_F, can be obtained by multiplying muscle activities u with the A (Lindsay, Sustar, and Dickinson, 2017).

steering muscles which overlays the image. A set of three control points determines the size, position, orientation and shear of the muscle contours. The three control points are used to compute the *affine transformation* that the 3D muscle model has to undergo to match up with the muscle contours in the image. By applying the affine transformation before computing the pseudo-inverse of the muscle model, one can obtain accurate muscle fluorescence unmixing for flies at any orientation where the steering muscles are visible.

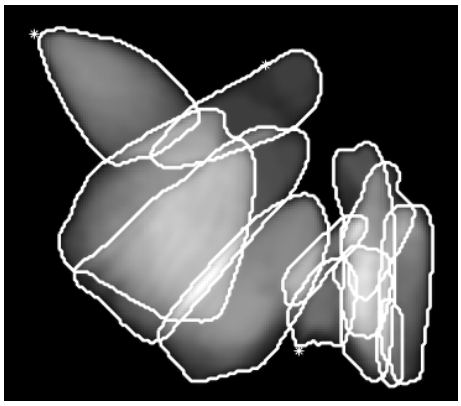

Figure 2.9: Muscle contours and control points (*) for the steering muscles.

The real-time muscle unmixing model allows activation of the high-speed cameras on muscle activity. Because the strength of fluorescence can differ per muscle and exact lighting conditions, it is necessary to scale the individual muscle activity traces. By using the mean and standard deviation of a muscle's activity pattern the trace can be *z-score* normalized, i.e. subtracting the mean from the signal and dividing by the standard deviation. Fluorescence of the steering muscles peaks at flight initiation

and stops and is minimal during periods when the fly is not flying. To normalize the muscle activity to fluorescence intensity during flight, the periods when the fly is not flying need to be removed from the z-score calculation. The wingbeat analyzer signal that is used to strobe the blue LED and microscope camera is quiescent when the fly is not flying. During the experiment, a rolling buffer of 30 seconds of muscle activities is kept and the mean and standard deviation are computed over the buffer. At the same time, the Teensy microcontroller that controls strobing sends a boolean value whether the fly is flying to the computer that performs the real-time muscle unmixing. A timer on the computer keeps track of the time since the last flight start and gets reset to zero when the fly is not flying.

With the flight timer and z-score normalized muscle activity, it is possible to reliably trigger the high-speed cameras on muscle twitches. At the start of an experiment, the user can select the muscle of interest and the gradient threshold for which high-speed video recording will be triggered. A gradient-threshold is used instead of an absolute threshold as it is a more reliable method to capture significant changes in muscle activity and wing motion during the 1 second high-speed videos. The gradient of muscle activity is computed over the last 5 frames in the rolling buffer. Whenever the absolute gradient value exceeds the user-defined threshold and the fly has been flying for more than 30 seconds, a trigger command will be send to the Teensy microcontroller. The Teensy microcontroller will subsequently send a TTL pulse to the high-speed cameras, saving a 1 second high-speed video centered around the muscle twitch on the current partition. After 8 triggers all paritions in the high-speed camera buffer are filled and the experiment is automatically stopped. The high-speed cameras will automatically start transferring their buffers to the hard-drive of a designated computer that controls the high-speed cameras.

2.4 Visual feedback through a flight simulator

The steering muscles of a fly control all flight behavior; from subtle changes in wing motion trim to rapid escape maneuvers. Real-time monitoring of muscle activity allows us to capture high-speed video of relatively fast changes in wing motion. Waiting for spontaneous muscle twitches does not guarantee that all possible muscle activity patterns will be captured over time. The constraints of tethered flight deprive a fly of almost all sensory feedback, except for vision. Halteres, small club-shaped organs that beat in counter-phase to the wing, are the fly's equivalent of an Inertial Measurement Unit (IMU). Fields of strain-detecting sensors at the base of the haltere, called *campaniform sensilla*, can detect the minute Coriolis forces that occur when

a fly rotates. Flies can sense apparent wind using two sets of antennae, called *Johnston's organs*, on their head. The lack of mechanosensory feedback in tethered flight means that flies rely almost exclusively on visual feedback.

One of the most important forms of visual feedback for flies is *optic flow*. Optic flow is the apparent motion of surfaces and edges on the retina as a consequence of the relative motion between the observer and its environment. A fly has specialized groups of neurons on its retina to observe translational and rotational optic flow, (Schnell et al., 2010). Optic flow is important in flight, as it is difficult to sense important flight parameters such as ground speed and drift without vision.

To stabilize flight, insects have fast *optomotor reflexes* that can be triggered by optic flow patterns (Figure 2.10). As a fly is fixed in tethered flight, these optic flow patterns cannot be induced by the fly's own motion. Instead, the surrounding of the fly has to move. Early research on the optic flow made use of rotating drums with stripe patterns, but nowadays LED displays are the norm (Reiser and Dickinson, 2008).

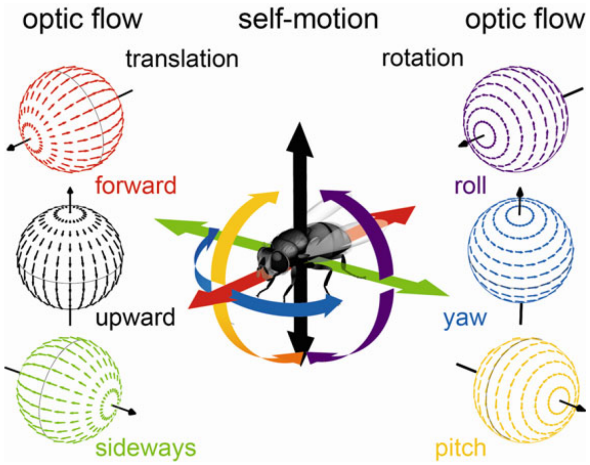

Figure 2.10: Optic flow patterns induced by translational motion: forward, sideways, upward, and rotational motion: roll, pitch, yaw (Egelhaaf, 2013).

To elicit strong optomotor reflexes, and steering muscle activity, I built a LED display surrounding the fly. The LED display consists of 24 horizontal times 5 vertical 8×8 green LED panels. The dichroic mirror of the epi-fluorescent microscope for muscle imaging is transmissive for green light. Having a large LED display with fluctuating green light patterns illuminate the fly, is not ideal when trying to capture small intensity changes in muscle fluorescence. To minimize disturbances of the muscle

imaging, a red stage light filter (Roscolux 26: light red) was placed in front of the LED panels. Shifting the LED panels to red light has the additional advantage that it improves the fly's vision. Unlike vertebrates, flies require red light to regenerate the photo-sensitive retinal (Arshavsky, 2010). Blue light converts *trans*-retinal into *cis*-retinal and this molecular change can be sensed by neurons on the retina. To convert *cis*-retinal back to *trans*-retinal, the retina needs to be exposed to orange/red light. In the outdoor environment this condition is automatically satisfied as sunlight contains both blue and red light. In the muscle-imaging setup, the blue LED used for fluorescence excitation rapidly depletes the *trans*-retinal in the eyes. The red light of the LED display helps to balance the *trans* and *cis*-retinal concentrations.

To detect optic flow, flies need to track the texture of their visual surround. A pattern with lots of features in the full field of view of the fly mimics the optic flow patterns of free flight. A so-called *starfield* pattern induces strong optomotor reflexes. The starfield pattern is generated by an algorithm in Python where a 3D volume is filled with blobs at random positions. A virtual fly and virtual LED display are positioned in the middle of this 3D volume and the algorithm computes the projection of the blobs on the LED display from the perspective of the fly. To create visual stimuli, the fly and the LED display translate or rotate at a given velocity through the 3D volume at a certain frame rate. For each frame, the algorithm computes the projection of the blobs on the LED screen. Using this method, I created 6 translational optic flow patterns (forward, backward, sideward left, sideward right, downward, and upward flight) and 6 rotational optic flow patterns (roll left, roll right, pitch left, pitch right, yaw left, and yaw right). Fly vision is much faster than human vision ($\sim$ 25 Hz) and a minimum refresh rate of 60 Hz is required to ensure smooth perceived motion. The optic flow patterns are therefore designed to run at 60 fps and have a maximum angular velocity of $10^{\circ}\ s^{-1}$ on the fly's retina for both rotational and translational patterns.

In free flight, flies continuously adapt their wing motion to stabilize flight and compensate for drift. While the halteres provide sensory feedback to recover from disturbances such as wind gusts and turbulence, the visual system predominantly controls wing adjustments to compensate for rotational and translational drift. In absence of visual feedback, left and right wing motion tends to become highly asymmetric. Slight asymmetries in wing motion can cause strong roll and yaw torques. Highly asymmetric wing kinematics are therefore undesirable, as these conditions will never be experienced in free flight.

In order to provide visual feedback, the stroke extent of the left and right wings are tracked via a machine vision camera. The machine vision camera records the top view of the fly at 30 fps. A prism (90% transmission, 10% reflection) splits the IR-illumination from below between the high-speed camera and the flight behavior camera. A real-time algorithm automatically extracts the left and right wing amplitude from the wing shadow (Figure 2.11). The difference between the left and right wing angles is used to control the yaw position of a vertical black stripe (8 pixels wide) on a red background. This so-called *stripe fixation* is a very robust behavior in flies, in which they will adjust the left and right stroke amplitude to keep the stripe in front of them. The visual feedback provided by the dark stripe ensures that wing motion is relatively symmetric and that the observed muscle activity corresponds, as closely as possible to free flight conditions.

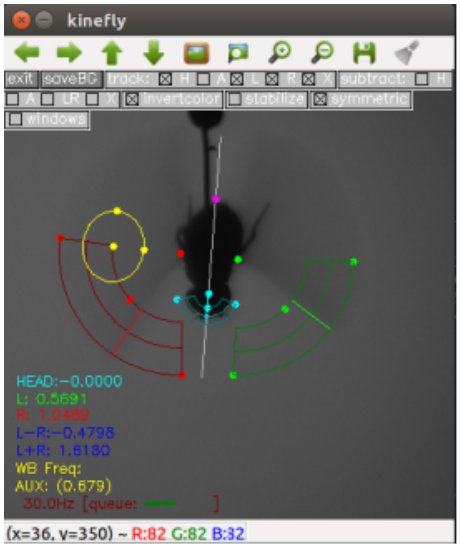

Figure 2.11: Snapshot of the Kinefly system. The green and red contours can extract the left and right stroke angle respectively. Wingbeat frequency can be measured using the yellow contour (Suver et al., 2016).

During the experiments, the flies will be in *closed loop* mode for 60 seconds interspersed by an *open loop* presentation of a randomly selected translational or rotational optic flow pattern for 5 seconds. The LED panels are updated at 60 fps by a *panel controller* which is driven by an ATmega8 MCU (Atmel Corp.), (Reiser and Dickinson, 2008). Commands from the experimental computer are sent via an USB cable to the panel controller. During the experiment, three different types of commands are used: start open-loop mode, start closed-loop mode and select a display pattern (optic flow or stripe fixation). The display patterns are stored

on a SD-card that serves as the memory of the panel controller. While the panel controller commands during the experiment are saved, it was outside the scope of the research to correlate optomotor stimuli to changes in muscle activity and wing kinematics.

2.5 Combining muscle imaging and high-speed videography in one setup

The experimental setup consists of three components that have to be integrated to collect the full range of muscle activity and wing motion a fly employs during flight: high-speed videography, real-time muscle imaging, and the flight simulator. Coordinating multiple processes and saving all the data in real-time is a demanding task for any type of software. A state-of-the-art program that allows to run hundreds of processes in parallel is the Robotic Operating System (ROS), (Stanford Artificial Intelligence Laboratory et al., 2018). As its name suggests, ROS is used to control and test robots. ROS can process, analyze, and store data streams from multiple sensors and send out commands to micro controllers, all simultaneously. At the start of the experiment, the ROS environment called the *roscore* is launched. Within the roscore, there are two types of processes, or *nodes*: (1) a publisher node that publishes a message and (2) a subscriber node that is activated when a new message is published. Sensors like cameras are typically publisher nodes, that publish a message containing image data whenever a new frame has been recorded. A subscriber node to the camera message can come into action and apply an algorithm on the new image, extracting optic flow for example. In a typical application such as an autonomous car, the resulting optic flow map gets published by a dedicated publisher and another subscriber can infer the vehicle's speed from the optic flow map. This network of publishers and subscribers is flexible and easy to use, while the under-the-hood software of ROS allocates CPU resources such that all nodes operate in parallel.

Synchronization is one of the most important aspects of the experiment. The frame rates of the high-speed cameras (15,000 fps) and the fluorescence microscope (100 fps) are two orders of magnitude apart. Key to synchronizing the high-speed video data and the muscle imaging is the Teensy 3.2 microcontroller that controls the strobing of the fluorescence camera and the blue LED. Using the *rosserial* package, the Teensy can be turned into a publisher node in the roscore environment. Similarly, the software running on the Teensy microcontroller can subscribe to nodes in the roscore environment. The Teensy microcontroller takes in the synchronization signal of the high-speed cameras and counts the number of frames since the start of

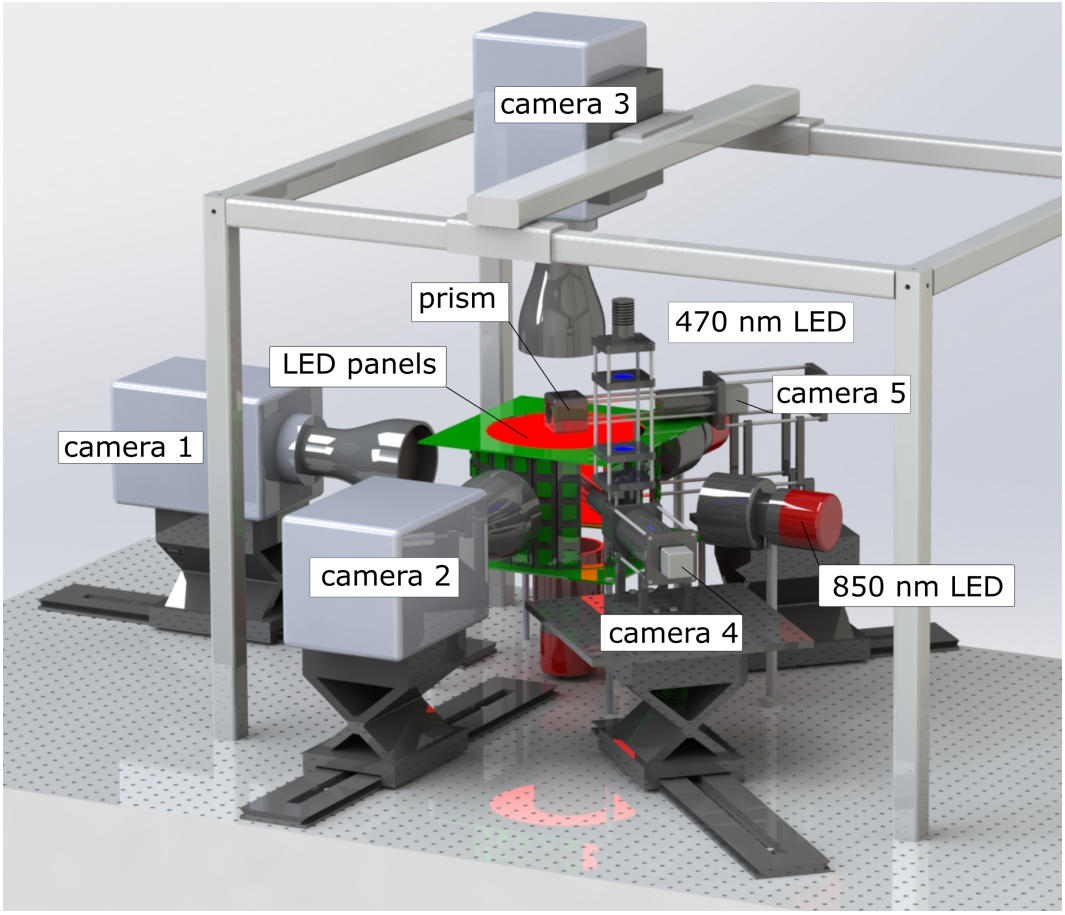

Figure 2.12: Rendering of the experimental setup. High-speed cameras 1-3 run continuously at 15,000 fps with a shutter time of 33 μs and IR (850 nm) backlighting. An epi-fluorescent microscope images steering muscle fluorescence from the left side of the the fly at half the wing beat frequency (camera 4). Red LED panels surrounding the fly can display open and closed-loop stimuli, where visual feedback is enabled by a behavior camera (camera 5) that images the top view of the fly via a prism.

the experiment using a *long* integer called the *frame count*. Synchronization of the fluorescence images and high-speed video frames is achieved by adding a timestamp to each fluorescence frame. The muscle fluorescence camera is strobed such that the shutter is open for two consecutive blue light pulses. As soon as the shutter closes, the camera will read out the chip and send a frame to the experimental computer. At the start of each blue light pulse, the Teensy microcontroller will publish the current frame count. Whenever the camera publishes a frame, a subscriber to the camera topic will add the latest frame count to the image as a timestamp. After extracting the muscle activity from the fluorescence image, the image timestamp is

added to the muscle activity values as well. When a TTL pulse triggers the recording of a high-speed video, the Teensy microcontroller will publish a message with the frame count at the instant that the TTL pulse arrived. The trigger timestamps and the muscle activity timestamps allow me to synchronize high-speed video data and muscle activity.

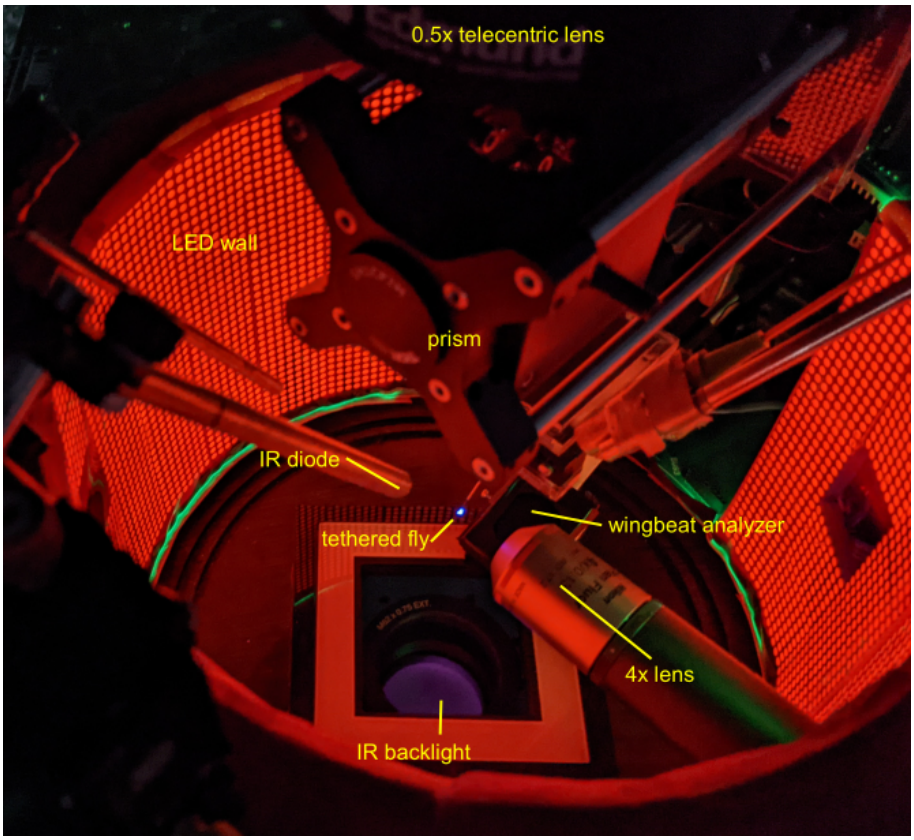

Figure 2.13: Image of the inside of the setup during an experiment.

Before the automated recording of wing motion and muscle activity can start, the user needs to perform several actions. First, the tethered fly needs to be positioned in focus of all three high-speed cameras. The position of the fly can be manipulated in three orthogonal directions using linear motion stages. Once the fly is in focus, the epi-fluorescence microscope can be moved via linear motion stages to get the steering muscles in focus. To extract muscle activity from the fluorescence images, it is necessary to scale and align the ROIs to the steering muscles and compute the affine transformation of the 3D muscle model. The alignment of the muscle ROIs is done via a Graphical User Interface (GUI) in which the muscle contours are displayed over a live view of the fluorescence camera and the user can drag the three control points (Figure 2.14). Muscle activity of the last 30 seconds is displayed at

four panels and helps the user to assess the data collected. The GUI is implemented in Python, using the PyQt and Pyqtgraph packages. For the closed-loop visual feedback, it is necessary to align the left and right wing ROIs in the Kinefly GUI. The final step that the user has to take is selecting a muscle and activity threshold that triggers the recording of high-speed videos. The real-time muscle activity display can help the user to select an interesting muscle and set the activation threshold to a level that ignores minor changes in activity but is low enough to be triggered by major changes. Once the user has determined the trigger settings, they can click the *start recording* button which starts the automated experiment. The open and closed-loop visual stimuli will be displayed on the LED display and the recording of a high-speed video whenever the muscle activity exceeds the trigger threshold. If the trigger threshold is too high, the user can adjust the threshold during the experiment. The experimental script keeps track of the number of triggers and the experiment is automatically stopped when 8 triggers are reached. Alternatively, the experiment is also stopped when the duration exceeds 30 minutes.

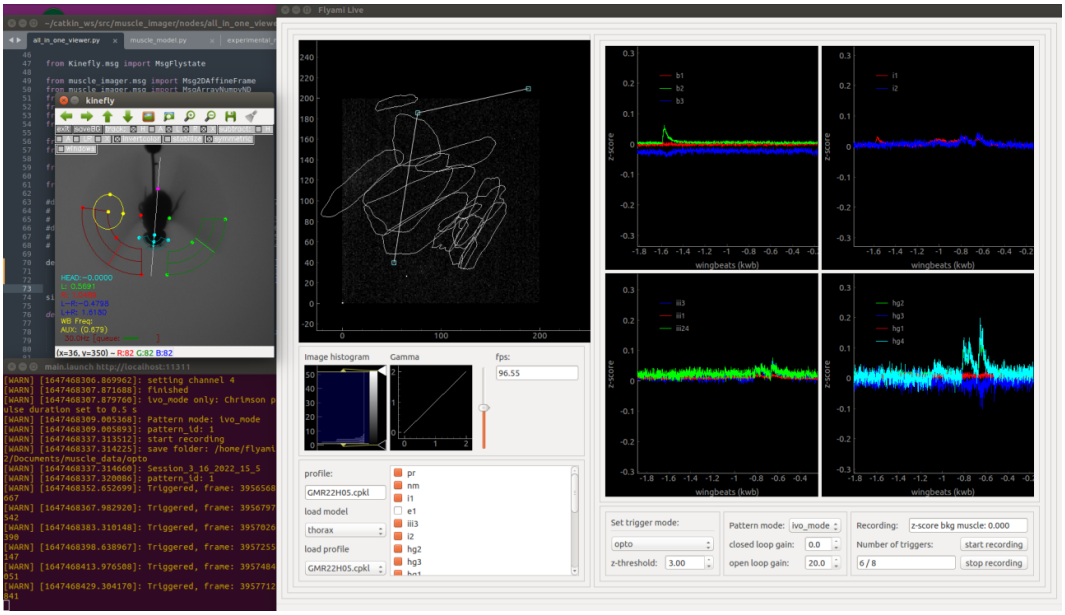

Figure 2.14: Snapshot of the GUI during an experiment.

Once the experiment ends, the high-speed cameras will automatically transfer the high-speed videos to a dedicated computer. The relevant data of the muscle imaging needs to be saved to a hard drive during the experiment however, as the available Random Access Memory (RAM) is insufficient. ROS has a dedicated package, named *rosbag*, that saves all published topics during an experiment. Saving the

fluorescence images at > 100 fps via rosbag, turned out to be problematic however, because the images could not be written to file fast enough. Using the h5py library in python, it is possible to write all data to an hdf5-file during the experiment. At the start of the experiment, an hdf5 file is created and data can be written to the file during the experiment. The hdf5 file records a variety of topics: fluorescence images (100 fps), kinefly images (30 fps), muscle activity, left and right stroke extend from kinefly, trigger timestamps, trigger threshold, randomly selected open-loop patterns (index), flight state (flying or not flying), and the affine transformation parameters (uv-coordinates of control points). Each topic contains the frame count timestamp, such that all data can be synchronized. The hdf5 file is closed automatically at the end of the experiment.

Using this automated process of data collection, I collected a large dataset of high-speed videos and muscle fluorescence data. I tried to record data from at least 5 flies with a specific trigger muscle (Table 2.1). This was successful for all steering muscles except the iii_1 and iii_2 muscles. In almost all experiments, iii_1 activity was only observed during flight starts and stops. The iii_2 muscle is active during flight, however, changes in muscle activity tend to be gradual. In total, I collected 479 high-speed videos from 82 flies.

trigger muscle	fly 1	fly 2	fly 3	fly 4	fly 5	fly 6	fly 7	fly 8	fly 9	fly 10
b_1	5	2	8	6	8	8	-	-	-	-
b_2	4	8	4	7	7	6	1	1	-	-
b_3	8	2	8	7	3	7	8	-	-	-
i_1	8	3	1	2	8	8	5	8	5	-
i_2	6	8	5	6	3	4	8	2	2	-
iii_1	4	-	-	-	-	-	-	-	-	-
iii_2	8	3	8	6	-	-	-	-	-	-
iii_3	8	8	8	8	7	2	8	8	3	8
hg_1	6	8	8	4	5	1	3	-	-	-
hg_2	8	8	8	8	5	2	5	6	-	-
hg_3	8	8	8	1	5	8	8	7	-	-
hg_4	8	5	8	8	7	4	-	-	-	-

Table 2.1: Table of high-speed videos recorded for different trigger muscles. Note that each cell is the number of videos recorded during an experiment from a single unique fly.

Chapter 3

ACCURATE 3D POSE RECONSTRUCTION OF WING MOTION

The full dataset consists of 527 seconds of high-speed video per camera and contains 23.7 million frames. It is obvious that an automated method is required to extract wing and body pose from the images. There are several methods to get the position and orientation of the wing from multiple camera views: hull reconstruction, 3D model fitting, and neural network prediction. In previous free flight experiments (Zabala et al., 2009), (Muijres, Elzinga, Melis, et al., 2014), (Muijres, Elzinga, Iwasaki, et al., 2015), (Muijres, Iwasaki, et al., 2017), the 3D model fitting methodology of (Fontaine et al., 2009) was used. Although the automated tracking method works well for free flight experiments, it requires manual annotation of the wing and body pose for 5 subsequent frames. After manual annotation, the method can track body and wing motion using an *Unscented Kalman Filter* (UKF) and *Iterative Closest Point* (ICP). For the ICP method to work, it needs an initial pose estimate that is close to the actual pose of the body or wing. In the method developed by Fontaine et al., 2009, this initial pose estimate is provided by the UKF, which relies on the tracking results of previous frames. When the wing is not visible in one or more camera views, the ICP method might not converge on a pose close to the true solution. Once an erroneous pose vector was found, the initial pose vectors predicted by the UKF for subsequent frames deviate and the tracking method will *loose track*. Once the method looses track, the values of the tracker will be far from the actual solutions. Only manual re-tracking of the body and wing pose can restore the tracking method to converge on the true pose vectors.

A large difference between the experiments in this study and previous studies is the tethered flight condition. Although tethered flight constrains the orientation and position of the body, it is actually more difficult to track wing motion using the Fontaine et al., 2009 method. In tethered flight, the wings clap together at the end of the upstroke for most wingbeats. Clap-and-fling is rare in free flying *Drosophila*; however, it is ubiquitous in tethered flight. Once the wings clap together, the ICP method has to "decide" which wing to follow and there is a high probability that the method will loose track. With a wingbeat frequency of 200 wingbeats per second, the amount of manual re-tracking required becomes impractical. A new *time-independent* tracking method is required to track the wing motion in the dataset.

In this chapter, I will discuss the existing methods used for wing pose reconstruction followed by the time-independent tracking algorithm I developed for tethered flight. The developed methodology consists of two steps: a *pose prediction* step by a trained Convolutional Neural Network (CNN) and a *pose refinement* step by a Particle Swarm Optimization (PSO). As the method does not use the tracking results of previous frames, converging on an erroneous pose does not affect future solutions. After the automated tracking method has run through a high-speed video, there are two post-processing steps: first the effect of outliers is reduced by Kalman filtering and secondly the wing pose is converted into Tait-Bryan angles in the strokeplane reference frame to increase the interpretability of the wing motion data.

3.1 Hull reconstruction, 3D model fitting and neural networks

In the last three decades, the availability and affordability of high-speed cameras and computers has dramatically increased. With this improved hardware, it is relatively easy to collect large quantities of data. There are however surprisingly few machine vision algorithms to automatically extract kinematic data from videos of animal motion. In this section, I will discuss three different algorithms that can track wing motion in insect flight based on high-speed video from multiple angles.

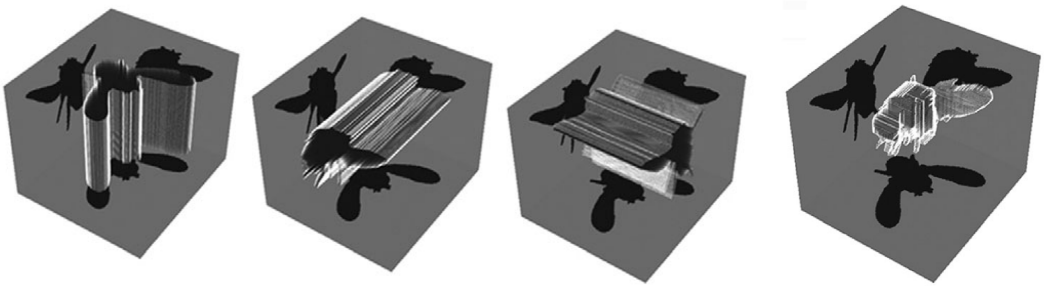

Figure 3.1: The 3D-shape of the fly is carved out in voxel-space by removing all voxels that project to background pixels for each view (Ristroph, Berman, et al., 2009).

Hull reconstruction

Ristroph, Berman, et al., 2009, developed a *3D hull reconstruction* for automated wing tracking. The high-speed videos of free flying flies are capttured at three orthogonal angles and back-lit by IR-panels. A DLT camera calibration provides the camera-to-world and world-to-camera matrices of the three cameras. The camera-to-world matrices are used for *voxel carving*, which forms the basis of the 3D hull reconstruction method. Before the voxel carving can start, a binary mask is obtained

for each frame in which the background (white) pixels get a value of 0 and the foreground (black) pixels get a value of 1. This *foreground segmentation* is performed by setting an intensity threshold and assigning pixels above the threshold as foreground pixels. The voxel carving starts by creating a voxel cube with sufficiently large dimensions that it contains the projected image of each camera view and the voxel spacing corresponds to the distance between pixels on the camera chip times the magnification. For each frame triplet, the values of the voxels in the voxel cube are first set to 1. Subsequently, for each view the camera-to-world matrix is used to find all voxels that correspond to a pixel value of 0 in the binary mask (Figure 3.1). All voxels that project to a pixel value of 0 in one or more camera views are set to 0. The remaining voxels that are 1 form the hull reconstruction of the fly.

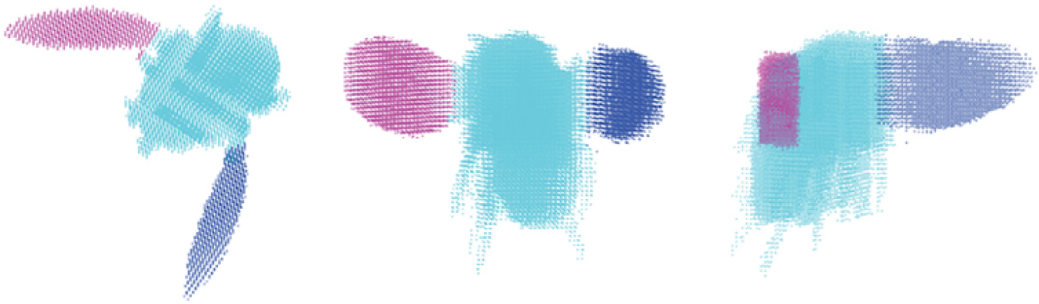

Figure 3.2: A k-means clustering algorithm segments the voxel space into body, left wing and right wing clusters (Ristroph, Berman, et al., 2009).

Once the hull reconstruction has been obtained, the voxels are segmented into 3 segments: body, left wing and right wing. Using *k-means clustering* it is possible to reliably segment body and wings. The position of the body is found by computing the center of mass from the body voxels. Orientation of the body and wings is subsequently found by performing Principal Component Analysis (PCA) on the segmented voxels. Three principal components are computed per voxel segment and the first principal component corresponds to the longitudinal axis of the body or wing. The second and third principal components do not reliably yield the rotation angle of the body or wing around the rotational axis however. Using geometric information of how the body and wings are coupled helps to establish the correct rotation angle. In case of the wings, the projected voxels form a parallelepiped shape and it is assumed that the largest diagonal plane forms the (flat plate) wing. To determine the rotation angle of the body, along the longitudinal axis, and the correct sign of the longitudinal axis, i.e. pointing to the head, the body voxels

are segmented into 3 sub-segments via k-means clustering. The 3 sub-segments correspond roughly to the head, thorax, and abdomen of the fly, and their centers of mass form an arc. The plane which is spanned by the 3 centers of mass is used to compute the rotation angle. It is assumed that the sub-segment with the least amount of voxels corresponds to the head.

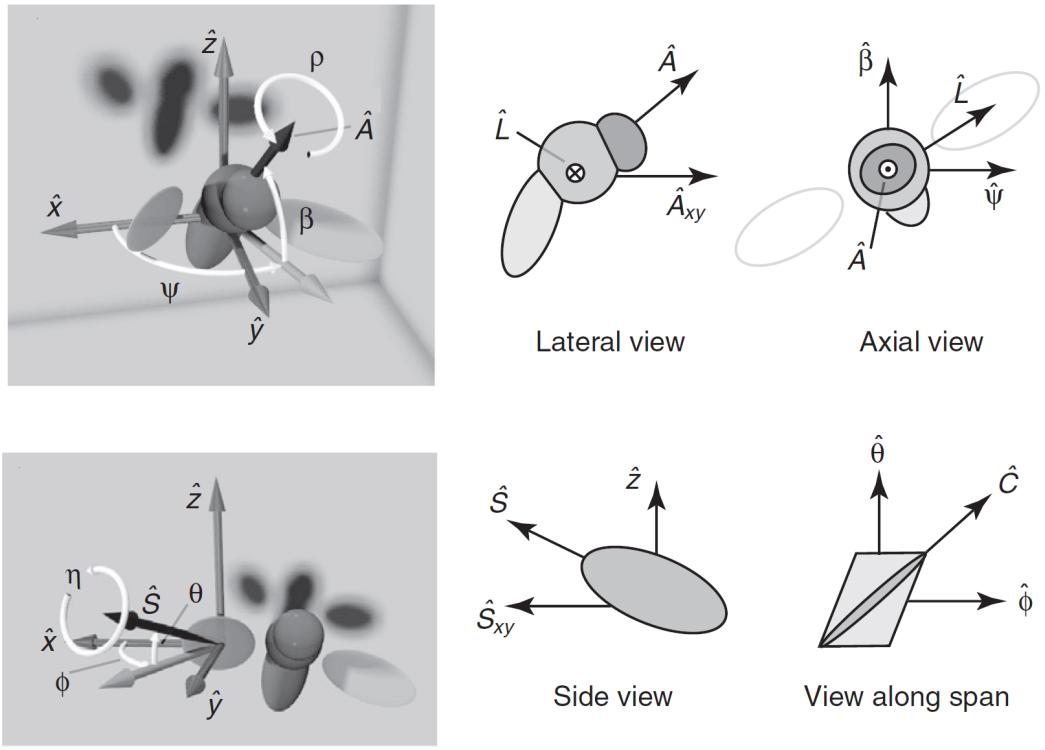

Figure 3.3: Definitions of Euler angles describing body orientation (ψ, β, ρ) and wing orientation (ϕ, θ, η) w.r.t. the longitudinal axis of the body or wing segments (Ristroph, Berman, et al., 2009).

Using the longitudinal axes and the rotation angle of the symmetry axis around it, Ristroph, Berman, et al., 2009 defined the orientation of a segment w.r.t. to the world reference frame via three Euler angles, Figure 3.3. The method is time-independent and has been used successfully in several free flight studies: (Bergou et al., 2010), (Ristroph, Bergou, Ristroph, et al., 2010), (Ristroph, Bergou, Guckenheimer, et al., 2011), (Beatus, Guckenheimer, and Cohen, 2015) and (Whitehead et al., 2015). Bomphrey et al., 2017, an adapted a version of the tracking method to track wing camber by recording the wing motion of a mosquito from 8 different angles. With only three orthogonal angles and telecentric lenses, some of the geometric assumptions of the method fall apart, however. For example, a large number of *false voxels* are introduced when the body occludes the wing in one or more views. For most

orientations of the fly this is likely to happen for several frames in each wingbeat. The parallelepiped that is used to find the largest diagonal plane corresponding to the wing surface becomes a rectangular box when telecentric lenses are used, as the back-lighting and the acceptance angle of the lens are parallel. Using this method, the wing pitch angle cannot be reliably found in the experimental setup of this thesis.

3D model fitting

The 3D model fitting method of Fontaine et al., 2009 relies on a *B-spline* model of a fly. B-spline surfaces capture the essential geometry of the fly with a minimal number of parameters. The body model consists of three rotationally symmetric revolved B-spline surfaces of the head, body and abdomen (Figure 3.4). A single closed B-spline forms the model of the wing, as the wings are assumed to be flat plates. For simplicity, the three body components are assumed to form one rigid body. The 3D model of the fly consists of three rigid bodies and the tracking problem is to resolve the position and orientation of each body.

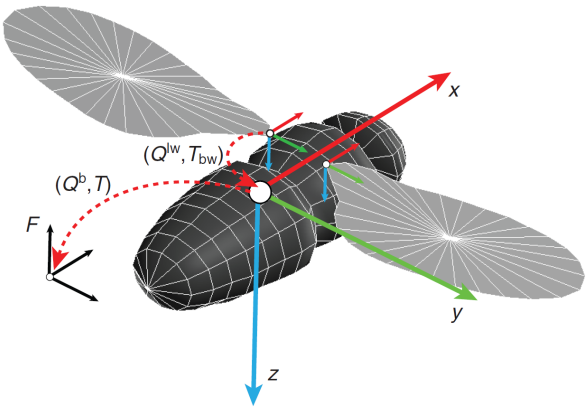

Figure 3.4: Definition of pose vectors describing position and orientation of the body and wings. The position of the body is described by translation vector T and the orientation of the body reference frame (x, y, z) relative to world frame F by quaternion Q^b. Orientation of the left wing is given by the quaternion Q^{lw} and the (fixed) joint position by T_{bw}, both in the body reference frame. A similar definition is used for the right wing (Fontaine et al., 2009).

The orientation of the 3D model is described using *quaternions*. A quaternion is a 4D vector that gives the rotation of one reference frame w.r.t. another reference frame as a 3D rotation axis and a rotation around that axis. The formulation of a quaternion, q, is:

$$q = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2)e_x \\ \sin(\theta/2)e_y \\ \sin(\theta/2)e_z \end{bmatrix} \tag{3.1}$$

where e_x, e_y, and e_z are the components of the normalized rotation axis and θ is the rotation around this axis in radians. By definition, the norm of a quaternion has to be 1: $\|q\| = 1$. Although a quaternion requires an additional parameter to represent orientation, quaternions are mathematically preferable over Euler angles. For some orientations, Euler angles can experience *gimbal-lock*. Gimbal-lock occurs when there are multiple (infinite) combinations of Euler angles that achieve the same orientation. These non-unique solutions for the same orientation are problematic when used in a cost function for optimization, for example.

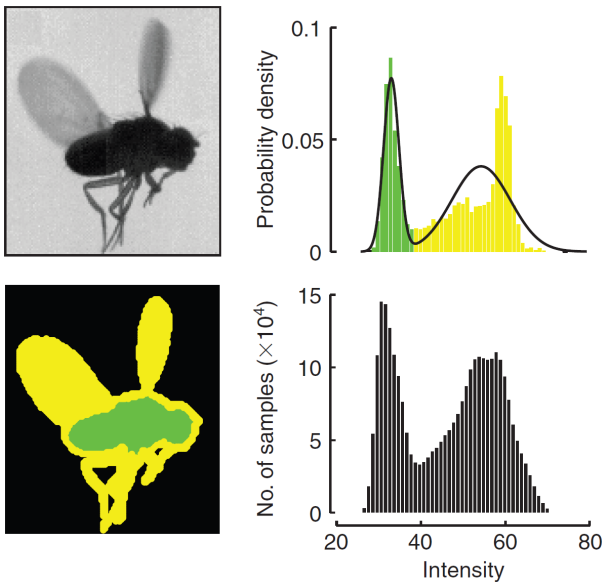

Figure 3.5: Segmentation of the image in background (black), wing (yellow), and body (green) pixels by background subtraction and intensity thresholding. The intensity threshold to separte body and wing pixels is found by fitting two normal distributions to the probability distribution of the pixel intensities and searching for the minimum between the two normal distributions (Fontaine et al., 2009).

Before starting the automated tracking, the user has to scale the B-spline model to the dimensions of the fly in the camera views using a dedicated GUI. Once the model is scaled appropriately, the median image is computed per camera view for the whole sequence. By subtracting the median image from the video, a *background*

subtraction is performed. The background subtraction removes background pixels from the resulting frames. Foreground segmentation between body and wing pixels is performed by fitting two Gaussian distributions to the probability density of the intensity values in a frame and finding the minimum value in between the two distributions (Figure 3.5). This minimum value is subsequently used for intensity thresholding.

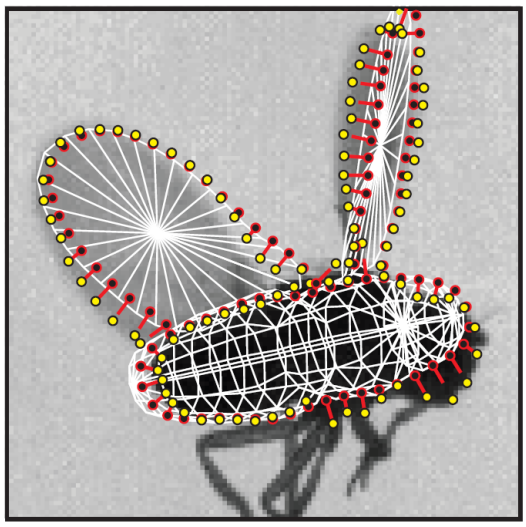

Figure 3.6: Correspondences between image contours (yellow) and model contours (red) (Fontaine et al., 2009).

With the body and wing pixels known, the algorithm extracts the contours of the body and wings from the two masks with a closed-contour B-spline. The body and wing contours in each camera view are used in the ICP algorithm. With the body and wings at a certain initial pose, the ICP algorithm first projects the fly model onto each camera view. A new set of closed-contour B-spline curves is fit to the projected body and wing models. Now for a fixed number of points, N, on the closed-contour B-splines derived from the image, the normal is computed. A search is performed along the normal vector to find the intersection with the B-spline curves from the model projection. After the line-search, a set of *correspondences* is found between the image and model contours of the body and wing for each camera view (Figure 3.6). The squared distance between two corresponding points is used as an error metric. By minimizing the sum of the squared distance of all correspondences, it is possible to find the pose vector for the model that fits the 3D model the best in all camera views.

The ICP algorithm does not always converge onto the optimal solution, however.

If the initial pose vector is too far from the optimal pose vector, the optimization will converge towards a local optimimum. To ensure that the initial pose vector of the model is close enough to the globally optimal solution, an UKF is used to predict the pose of the model using prior (tracked) pose data. Kalman filters use a model of the system dynamics and measurements of the actual dynamics to predict the future behavior of the system. An UKF is a Sigma-Point Kalman Filter, that uses so-called Sigma-points to sample the state of a non-linear system. The UKF prediction provides a good initial estimate, even when the wing is obstructed in some of the camera views.

The rotational symmetry of the body model makes it impossible to find the roll angle. A set of constraints is applied during the ICP optimization to tie the body roll angle to the joint positions of the left and right wing. An additional constraint is that the norm of the quaternions has to be 1.

Before the automated tracker can analyze a high-speed video sequence, the user needs to scale the 3D model and track the pose vector of the fly in the first 5 frames. Verification of the 3D model fitting and human tracked frames of the body orientation, shows that the tracker is within 5° of the human tracked angles. The method has been used in several free flight studies: (Zabala et al., 2009), (Muijres, Elzinga, Melis, et al., 2014), (Muijres, Elzinga, Iwasaki, et al., 2015), (Muijres, Iwasaki, et al., 2017) and (Veen, Leeuwen, and Muijres, 2020). Analyzing high-speed videos of tethered flies with 3D model fitting turned out to be problematic however. Manual re-tracking of the wings during clap-and-fling is extremely tedious.

DeepLabCut

A relatively recent method for tracking animal motion is named *DeepLabCut*, (Mathis et al., 2018). DeepLabCut relies on a deep residual neural network to trace key-points in images. The method is based on the tracking of human poses from single images using deep neural networks (Pishchulin et al., 2016). While human pose tracking relies on a dataset of over 25,000 annotated images to train the neural network, DeepLabCut makes use of *transfer learning* to drastically reduce the required training data. Manual annotation of tens of thousands of images is time consuming and often unfeasible for researchers. Transfer learning relies on the fact that many aspects of machine learning problems are the same, i.e. feature extraction and object detection from images. By copying parts of a neural network that is trained on one dataset and integrating it in a new neural network that is used for a

different task, the required training set for the new neural network can be drastically reduced.

In the case of DeepLabCut, the architecture of the *ResNet50* network has been copied. ResNet50 is a residual neural network of 50 layers that is designed for the ImageNet dataset, J. Deng et al., 2009. ImageNet is a vast dataset of images, > 3.2 million, containing manually labelled objects that span thousands of categories. Although the goal of the ResNet50 network is to predict the correct label for objects in the images, the *convolutional layers* of the network are primarily involved in feature extraction. Categorization happens primarily in the last layer of the ResNet50 network: the fully connected or *dense* layer. By only copying the convolutional layers of the trained ResNet50 network, it is possible to *inherit* the feature extraction capabilities.

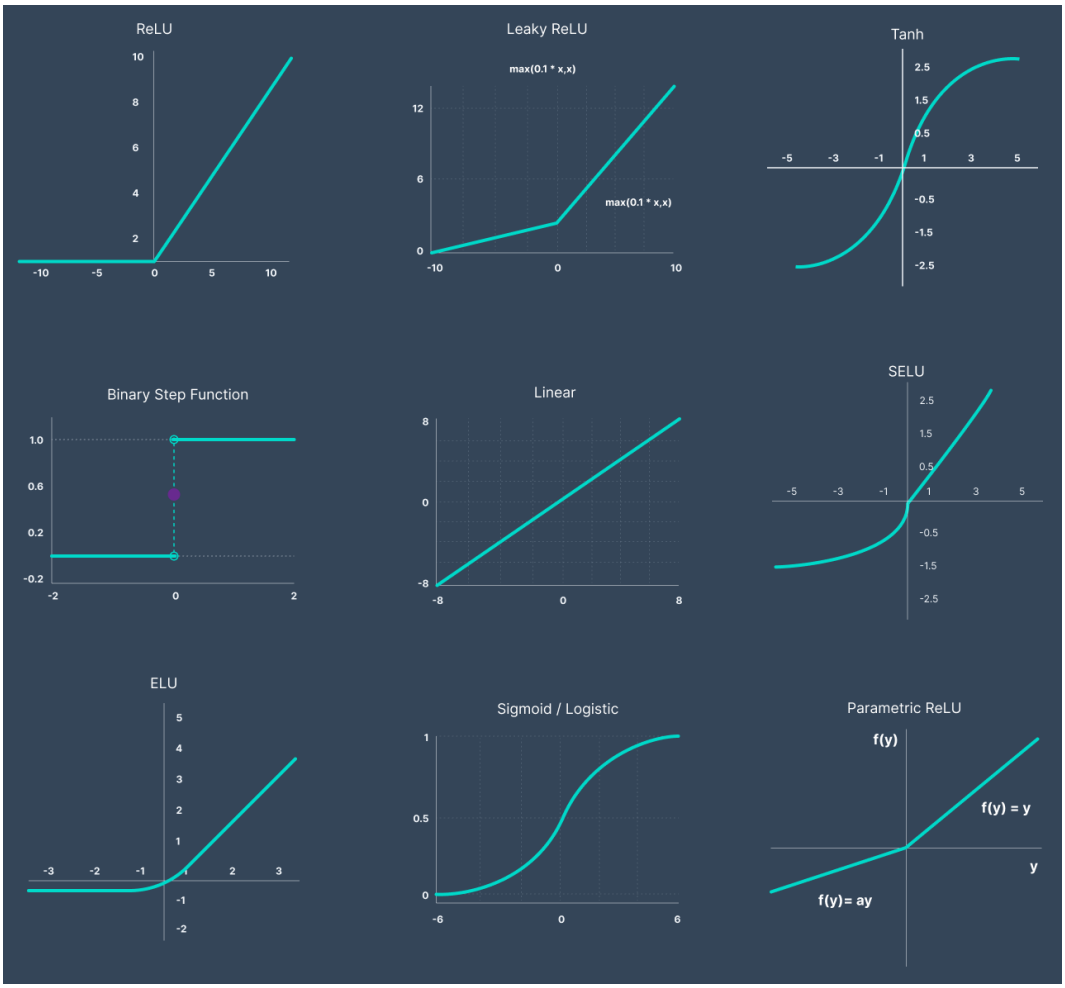

Figure 3.7: Examples of some activation functions for neural networks (V7, 2022).

Before focusing on the specific architecture of the DeepLabCut network, it is informative to go over the different types of layers that have been used. Convolutional layers scan over the image with filters of a certain window size. The filters are called kernels and the weights of these kernels are initialized randomly and subsequently updated by back-propagation of the prediction error. During the convolutional operation, the pixels in the filter window are multiplied with the weights of the kernel and summed. The resulting scalar value is projected onto an *activation function*. When the activation function is linear, the convolutional output is multiplied by 1. Another activation function that is often used is the Rectified Linear activation Unit (ReLU), which returns a 0 when the input value is below 0 and returns the input value when it is above 0. In the case that an image has Red Green Blue (RGB) color channels, the same convolutional filter is applied over all three channels and all the values are summed and mapped onto the activation function, resulting in a single scalar output. Similarly, convolutional layers can be stacked upon each other and the number of kernels of the previous layer amounts to the number of input channels for the next layer.

The layer that can perform complex inference is the fully connected or dense layer. A dense layer consists of a number of predefined *neurons* that all take in the same input vector. The input vector is subsequently multiplied with a weight vector in each neuron and summed into a single scalar value. This scalar value is mapped onto an activation function, which can have a *bias* value that can shift the *decision point* of the activation function from 0 to the bias value. When a linear activation function is used in a dense layer, training the layer is similar to performing linear regression on the data. When the activation function is non-linear, a non-linear regression is performed. The complexity of the functions that can be learned by dense layers increases with the number of neurons used and the number of concatenated dense layers. In practice, one or two dense layers with sufficient neurons can approximate almost any function intrinsic to the dataset.

To reduce the size of image data, *pooling* layers are often used. Similar to a convolutional layer, a pooling layer scans the image with a certain window size and stride. Instead of performing a convolution, the max or mean value of the window is computed. In contrast to convolutional layers, pooling layers do not sum over the kernel dimension.

A problem that is often encountered in deep neural networks is the *vanishing gradient* problem. When multiple convolutional layers are stacked upon each other,

the error update is multiplied by the weights of the convolutional layer during the back-propagation phase. As the weights are often between -1 and $+1$, the error update becomes smaller and smaller. This results in very small updates of the first convolutional layers of a deep CNN during the training phase. As a consequence, the vanishing gradient problem reduces the effectiveness of deep neural networks and adding layers might *decrease* network performance. Pooling layers can help to reduce the vanishing gradient problem. Another remedy is to *rescale* the intermediate data between layers via a *batch-normalization layer*. To speed up training, data is usually presented in batches during the training phase. In the case of image data, a random selection of images and associated labels are stacked together in one batch and fed through the neural network. The average prediction error is computed for the whole batch and used to update weights during the back-propagation phase. A batch-normalization layer computes the mean and standard deviation of the output data of the previous layer. The data is rescaled using the batch mean and standard deviation such that the output of the batch-normalization layer is centered around 0 and has a standard deviation of 1. Rescaling between layers keeps the gradient update at similar magnitude between layers and allows for much deeper neural networks.

When the stride of a convolutional or pooling layer is more than 1, the data dimensions of the output decrease. This compression of data is often used to shrink the number of connections to a dense layer at the end of the network. Expansion of data can be achieved with *deconvolution layers*, which multiply a scalar value with a kernel with a given window size and stride. After multiplication with the kernel for each input channel, the data is summed over the channel dimension and mapped onto an activation function. Deconvolution layers are typically used in a network that predicts a probability heatmap of key-point locations or segmented images.

Now that the different layer types in DeepLabCut have been explained, we can focus on the network architecture. The ResNet50 architecture consists of several *residual blocks* (Figure 3.8). A residual block consists of two convolutional layers with 3×3 windows and a stride of 1, immediately followed by a batch-normalization layer and a ReLU activation function. A bypass of the input to the residual block gets added to output of the residual block, followed by another ReLU activation function. The bypass or residual helps to combat the vanishing gradient problem by providing a shortcut for back-propagating the error update. After the ResNet50 layers, there are several deconvolution layers that expand the network back to the image size. In the

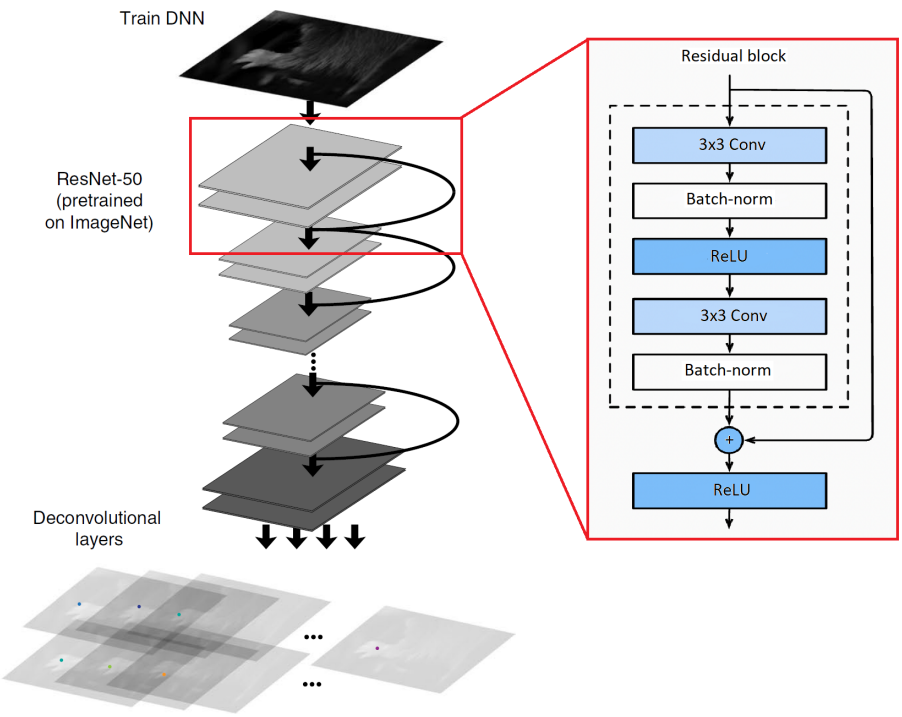

Figure 3.8: Architecture of the DeepLabCut network, the top part of the network consists of the convolutional layers of the ResNet50 network pretrained on the ImageNet dataset, followed by a number of deconvolutional layer that produce a heatmap of the probability of keypoint positions. The ResNet50 network consists of multiple stacked residual blocks, which consists of convolutional layers with 3×3 filters, batch-normalization, ReLU activation, and a by-pass that adds on the input of the residual block to the output (Mathis et al., 2018).

last deconvolution layer, each kernel predicts the location of one key-point. The ResNet50 layer weights are loaded from a network trained on the ImageNet data while the deconvolution layers are initialized with random weights.

To train DeepLabCut on a specific tracking problem, the user needs to annotate keypoints on images of the dataset via a GUI. Depending on the complexity and required accuracy of the tracking problem, the user needs to manually annotate several hundreds to thousands of images to provide a large enough training set. By rotating and flipping the images and labels, the training set can be *augmented*. The labeled dataset is split between a training set (95%) and a validation set (5%). After several hundred-thousands of epochs of training on the dataset, the training and validation errors stabilize around a minimum value. The validation error approaches the human annotation error, a result to be expected as the network cannot surpass the accuracy of the training data.

The trained DeepLabCut network performs well on several example tracking problems (Mathis et al., 2018). For example, several keypoints can be tracked accurately on a fly walking around and laying eggs in an acrylic cube (Figure 3.9). Trained on a dataset of 580 human annotated frames, the network can predict keypoints in the validation set with a root mean square error of 4.2 pixels.

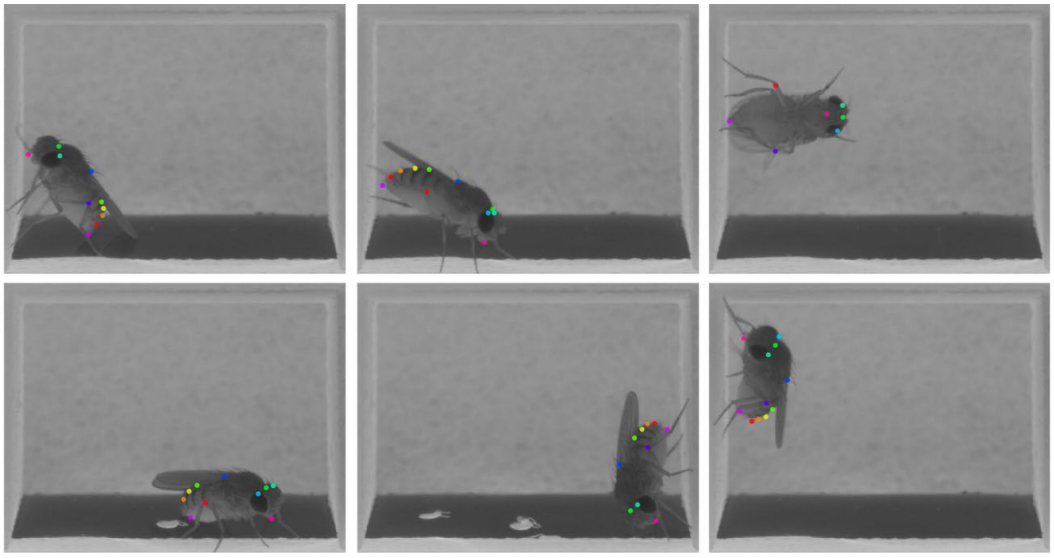

Figure 3.9: Tracking of keypoints on *Drosophila melanogaster* walking around and laying eggs in an acrylic cube (Mathis et al., 2018).

DeepLabCut is designed to track 2D keypoints on images, and some additional processing is required to track 3D wing motion from multiple camera views. The most straightforward method to obtain 3D keypoints is by training DeepLabCut on tracking 2D keypoints in each camera view and subsequently triangulate the 3D keypoints from the predicted 2D keypoints in all views. In Karashchuk et al., 2021, the legs of walking *Drosophila melanogaster* were filmed from multiple viewpoints (6), tracked in each view using DeepLabCut and subsequently triangulated. A minimum of 2 camera views is required to perform the triangulation, and the accuracy of the triangulation improves with the number of camera views in which keypoints are visible.

3.2 FlyNet: Combining neural network prediction with 3D model fitting

The methodologies discussed in the previous section form the state-of-the-art in markerless tracking of animal motion. At the end of 2017, after analyzing some initial high-speed videos of tethered flight, it became clear that hull-reconstruction and 3D model fitting would not suffice to track wing motion accurately in my exper-

imental setup. At this point, it was decided to design and program a new automated tracker that is time-independent from scratch. I tested several methodologies: performing 3D model fitting with global optimization methods like PSO and genetic algorithms, 3D model fitting on the hull-reconstructed voxel clouds and direct prediction of the wing pose vectors by several CNN architectures. These techniques all lacked the ability to track wing motion accurately and reliably during the wingbeat.

The most promising method was the direct prediction of wing pose vectors by CNN networks. Predictions of wing motion would be remarkably accurate for high-speed video sequences that were present in the training set. For videos of different flies (at similar position and orientation within the images), the performance of the trained CNN would be highly erratic however. When wing motion would change significantly during a high-speed video sequence, i.e. when steering muscles change their activity pattern, the predicted pose would remain close to the labeled wing motion pattern. In short, the training set does not contain sufficient variation in wing motion and fly shapes and orientations to *generalize* over all video sequences in the dataset.

The manual annotation of wing pose vectors from frame-triplets was performed using a dedicated GUI, in which the user can rotate and translate a 3D model of a fly projected on top of the images. By visually matching the contours of the 3D model and the underlying images, an accurate annotation of the body and wing pose vectors was obtained. Using the GUI, an initial manually annotated dataset of around 1000 frame-triplets and associated pose vectors was obtained. On average a full wingbeat consists of 75 frame-triplets, and this manually annotated dataset contains only 10 wingbeats from 10 different flies. It is therefore not surprising that the trained CNN does not generalize well over videos with varying wing motion patterns.

In an attempt to increase the training set with a large variety of flies and wing motion, *artificial training data* was created using the VTK library in Python. A 3D model of a fly would be scaled and colored with some random variation of typical scale and color of real flies in the actual dataset. Subsequently, the fly model was positioned and oriented according to the pose vectors in the manually annotated dataset with some random variation added. Using the DLT-calibration of the high-speed cameras, snapshots of the 3D fly model are taken from each camera view. These artificial images are coupled to the body and wing pose vectors that have been used for the positioning the 3D model. Repeating this process 100,000 times yields an artificial dataset that has a wide variety wing motion patterns and zero error in the

pose registration. After training the network on the artificial dataset, the prediction error on an artificial validation set was low. Unfortunately, the network cannot achieve a similar performance on predicting the wing motion in video sequences of real flies.

In September 2018, when DeepLabCut became available, Tarun Sharma, a PhD student in the Dickinson lab, tested its performance in tracking 10 keypoints on each wing in each camera view and subsequently triangulating these keypoints. Using around 1000 manually annotated frame-triplets with keypoints, it was possible to accurately predict 2D keypoints during most frames in a wingbeat. When keypoints were only visible in one or two views due to obstruction by the body, the triangulation became problematic. DeepLabCut would still provide an estimate of the location of the 2D keypoints in the obstructed views, but the prediction error was too large to get an accurate location of the 3D keypoint. Of the 10 keypoints on the wing, only the wingtip keypoint could be reliably triangulated during the full wingbeat. This was insufficient to derive variables like the wing pitch angle with sufficient accuracy.

The number of manually annotated frames that are required to accurately predict wing motion with a neural network is likely to be much larger than the thousand manually annotated frames. Most predictions of the network are close to the actual pose of the wings. With some refinement of the predicted pose, it is possible to perform accurate tracking of the wing motion. The refinement step can be performed by 3D model fitting. Instead of a Kalman filter, a trained CNN can predict the initial pose required for the 3D model fitting. The advantage of CNN prediction of the initial pose vector is that it is independent of previous frames. With CNN prediction and 3D model fitting refinement, it is possible to accurately track wing motion with a limited training set. In the remainder of this chapter, I will discuss the details of this tracking method, named *FlyNet*.

3.3 FlyNet: Body and wing pose vectors

At the core of the model fitting method is a 3D model of a fly. The 3D model is based on images from a tethered flying fly from different angles. Contours were extracted from the images to determine aspect ratios and curvatures of the head, thorax and abdomen. The 2D contours were converted into a 3D surface via Non-Uniform Rational B-splines (NURBS). Like B-splines, NURBS surfaces can be manipulated by changing the locations of control points (in 3D). By collapsing control points at the edges of a NURBS-sheet into each other, one can create a NURBS-sphere.

Three NURBS-spheres were created for the head, thorax and abdomen and the control points were manipulated to reshape the spheres according to the aspect ratios and curvatures found in the images.

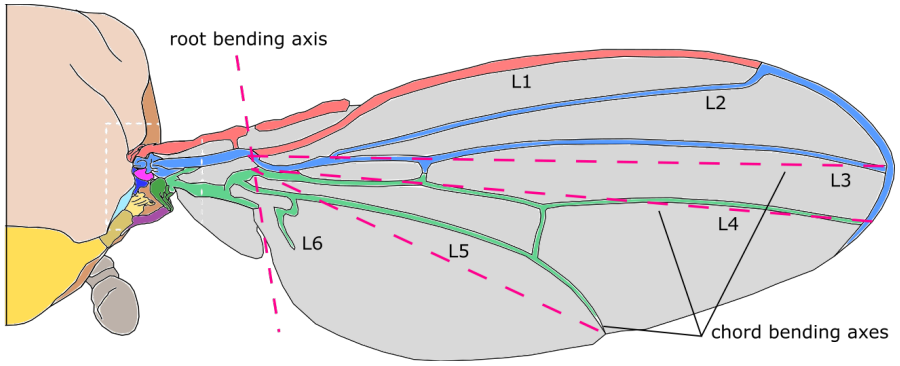

Figure 3.10: Wing veins, L1-6. The bending axes are marked by the striped magenta lines.

The wings of the fly are thin and can be assumed to be flat. Observations from high-speed videos show that the wings can deform significantly during flight, however. Deformation seems to occur along three different *hinge lines* that roughly follow three veins on the wing: L3, L4, and L5 veins (Figure 3.10). The model of the wing consists of four rigid panels connected by three hinge lines. To reduce the number of parameters representing wing deformation, it is assumed that the deformation angle along each hinge line is equal (Figure 3.11). The deformation angle, ξ, is the sum of the local deformation angle of each hinge line: $\xi = \xi/3 + \xi/3 + \xi/3$. Based on the manual annotation of wing pose, the assumption for uniform distribution of deformation holds for most frames in the dataset.

In most tracking methods, the wing hinge is assumed to be a ball-joint with 3 degrees of freedom (DOF). The wing hinge is complex, however, and the ball-joint assumption might not represent wing motion accurately. Another assumption that is often made in tracking methods is that the wing is a rigid, flat plate. Besides chordwise deformation, which is captured by the angle ξ, the wing can also bend spanwise at a flexible section close to the root of the wing (Figure 3.10). It is difficult to model spanwise bending of the wing, and the exact kinematics of the wing hinge and; therefore I have chosen to ignore these deformations in my wing model. Instead, the wing root and hinge are not included in the wing model; rather the origin of the wing is the intersection between the radial vein and the spanwise bending axis. Wing motion is described by a translation and rotation of the origin relative to the world reference frame.

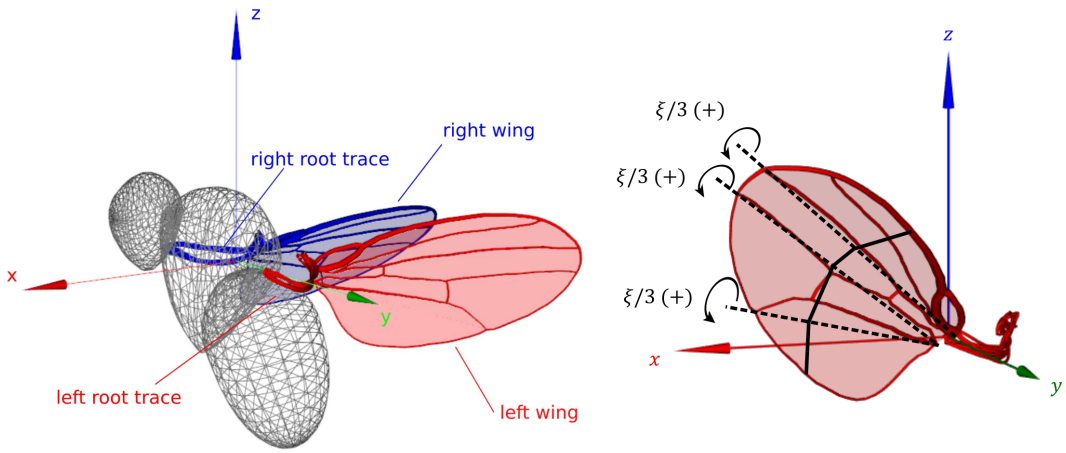

Figure 3.11: 3D model of a fly. The grey wireframes show the shape of the head, thorax and abdomen models. Left and right wing models are shown in red and blue respectively. The path of the left and right wing root traces are plotted in a reference frame that is fixed to the thorax model. Chordwise deformation angle ξ is displayed by the local deformation angles, $\xi/3$, around the 3 hinge lines (dotted lines).

The wing pose vector consists of 8 values: 4 parameters for a quaternion describing the orientation of the leading edge panel of the wing relative to the world reference frame, 3 parameters for a translation vector describing the distance between the wing root and the origin of the world reference frame, and the wing deformation parameter ξ. For the head, thorax and abdomen pose vectors there are 7 parameters each: a quaternion and a translation vector. The complete fly pose is described by a vector with 37 parameters: head (7), thorax (7), abdomen(7), left wing (8), and right wing (8). Besides the pose vector there is also a scaling vector that consists of the 5 parameters that scale the body and wing components independently.

3.4 FlyNet: Predicting pose with a Convolutional Neural Network

The 3D model of the fly can be used to manually annotate the fly pose in each frame triplet. A module in the FlyNet GUI, which was constructed in Python with the PyQt and PyQtGraph packages, allows the user to load the 3D fly model, scale the different model components and rotate, translate, and deform the model components. The GUI uses the DLT-calibration of the high-speed cameras to project a wireframe of the fly model onto the three camera views. The user can select a model component to: scale, rotate, translate, and deform. By adjusting the pose vector of each component until it matches the images in each view, an accurate label can be found for each frame triplet (Figure 3.12). Using this GUI, I collected a

manually annotated dataset of 1000 frame triplets and associated pose vectors.

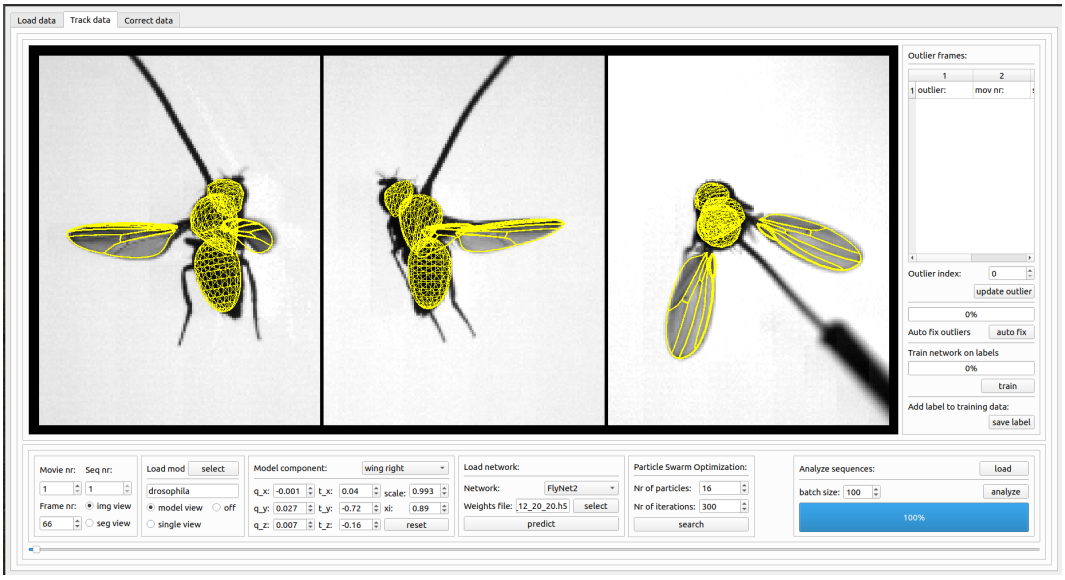

Figure 3.12: FlyNet GUI with 3D model (yellow wireframe) projected on the camera views.

With the annotated dataset it is possible to train a CNN to predict the pose vector. The input of the network is a frame triplet of three 256×256 pixel images. Neural networks work best with data centered around 0 and a standard deviation of 1. The image data has a bit-depth of 8 and needs to be rescaled by dividing by 255. After normalization, the images are cropped into 224×224 pixel images that are centered around the thorax of the fly in each view. The cropping removes unnecessary pixels and makes sure that the fly's body is in the middle of the image. After cropping the image, the next data pass is to a convolutional block that sequentially analyzes the cropped images of the camera views. The CNN is implemented in Python using the Tensorflow library with Graphics Processing Unit (GPU) support.

The CNN architecture consists of three convolutional blocks, one for each camera view. Most of the components of the convolutional block are explained in the previous section about DeepLabCut: convolutional layer (`Conv2D`), batch normalization (`BatchNormalization`), and max pooling (`MaxPooling2D`). In this case, however, a dropout layer (`Dropout`) is added, which randomly turns off a certain fraction of the input to the next layer. By turning off certain kernels or nodes for the following layer at random, the network becomes more robust and generalizes better over the dataset. When a training set is relatively small, as is the case for our manually annotated dataset, the network may learn that a certain combination of features in

the image corresponds to a certain pose. Under these conditions, however, some of these features may be meaningless, however, for example, the orientation of one of the legs might be associated with a certain pose. By randomly turning off this feature during the training, the network is forced to learn which features are meaningful and which are not. Shown below is the code that creates the convolutional blocks of the network:

```
branches = []
for n in range(self.N_cam):
    bn = Lambda(lambda x: x[:,:,:,:,n])(input_mdl)
    bn = Conv2D(32,kernel_size=(7,7),strides=2,activation='selu')(bn)
    bn = BatchNormalization()(bn)
    bn = MaxPooling2D(pool_size=(2,2),strides=2)(bn)
    bn = Dropout(0.1)(bn)
    bn = Conv2D(64,kernel_size=(1,1),strides=1,activation='selu')(bn)
    bn = BatchNormalization()(bn)
    bn = Dropout(0.1)(bn)
    bn = Conv2D(64,kernel_size=(5,5),strides=2,activation='selu')(bn)
    bn = BatchNormalization()(bn)
    bn = MaxPooling2D(pool_size=(2,2),strides=2)(bn)
    bn = Dropout(0.1)(bn)
    bn = Conv2D(128,kernel_size=(1,1),strides=1,activation='selu')(bn)
    bn = BatchNormalization()(bn)
    bn = Dropout(0.1)(bn)
    bn = Conv2D(128,kernel_size=(3,3),strides=2,activation='selu')(bn)
    bn = BatchNormalization()(bn)
    bn = MaxPooling2D(pool_size=(2,2),strides=2)(bn)
    bn = Dropout(0.1)(bn)
    bn = Conv2D(256,kernel_size=(1,1),strides=1,activation='selu')(bn)
    bn = BatchNormalization()(bn)
    bn = Dropout(0.1)(bn)
    bn = Conv2D(256,kernel_size=(3,3),strides=3,activation='selu')(bn)
    bn = BatchNormalization()(bn)
    bn = Dropout(0.1)(bn)
    bn = Conv2D(1024,kernel_size=(1,1),strides=1,activation='selu')(bn)
    bn = BatchNormalization()(bn)
```

```
    branches.append(bn)
conc = Concatenate()(branches)
conc = Flatten()(conc).
```

where the `Lambda`, `Concatenate`, and `Flatten` layers perform the necessary reshaping of the data. The activation function that is used is the so called Scaled Exponential Linear Unit (SELU) (Figure 3.7). In Tensorflow, the convolutional parameters are coded as:

```
Conv2D(32,kernel_size=(7,7),strides=2,activation='selu'),
```

with `Conv2D` as a 2D convolutional layer function, the first entry of the function is the number of kernels used (`32`), the second entry is the kernel window size (`kernel_size=(7,7)`), the third entry specifies the stride length (`strides=2`) in both dimensions and the final entry is the activation function (`activation='selu'`). Similar conventions are used for the max pooling layer:

```
MaxPooling2D(pool_size=(2,2),strides=2),
```

with the first entry being the pool size (`pool_size=(2,2)`) and stride in both dimensions (`strides=2`).

Batch size is an important parameter for the batch normalization layer. If the batch size is too small, the batch might not contain the range of variability of the full dataset. This is problematic as the rescaling of the data by batch normalization depends on the data in the batch. During training, a batch size of 50 samples is used, in which each batch consists of randomly selected samples from the training set. The average wingbeat consists of 75 high-speed frames and this batch size is sufficient to reliably contain the full range of wing motion.

The dropout layer has a single parameter, the dropout rate, which is the fraction of data that are turned off. After some testing, a dropout rate of 0.1 was chosen as it gave the lowest validation error. Lower dropout rates reduced the training error but increased validation error. Higher dropout rates increased the training error and therefore also the obtainable validation error.

I tested various CNN architectures and found that the relatively simple and shallow network presented in the previous paragraphs provides the most accurate and robust

performance. Surprisingly, the pre-trained ResNet network did not generalize well on the test set. An explanation could be that deep neural networks have too many degrees of freedom, and overfit on the training data. Relatively small and shallow networks force the network to generalize over the training set.

After the convolutional head, fully connected layers learn how to predict the pose vectors from the image features extracted by the convolutional layers. The definition of the pose vector of the 3D model components is such that they can be divided into *independent components*. For example, the wing pose vector consists of 3 independent components: the quaternion, the translation vector and the deformation angle ξ. In total, the model pose vector consists of 12 independent components (Figure 3.13). Splitting the prediction of the network into independent components helps generalization, as it is more difficult for the network to associate certain combinations of wing and body pose vectors with a set of features in the image. For example, a fly in the training set might have symmetric left and right wing stroke amplitude during an annotated wingbeat. The network might infer from the annotated wingbeat that left and right wing motion has to be symmetric, even if asymmetric wing motion is possible.

With the architecture of FlyNet defined, training on the dataset of manually annotated frames begins by splitting the data into a training set that contains approximately 95% (950 samples) of the annotated frames, while the remaining 5% (50 samples) was reserved for validation. Samples in the validation set come from the wingbeat of a fly that was not shown in the training set, such that the validation is representative of tracking new data. The network was trained for 1000 epochs with a batch size of 50 and a dropout rate of 0.1 and the mean squared error (MSE) as the loss function. A learning rate of $1.0 \cdot 10^{-4}$ was chosen with a decay of $1.0 \cdot 10^{-7}$. The decay value was subtracted from the learning rate after each epoch, resulting in a linear decrease of the learning rate during training. Using a Nvidia Geforce GTX 1080 graphics card with 8 GB of memory, it took approximately 24 hours to train FlyNet. During the training phase, the MSE for the independent components of the pose vector decayed up to 400 epochs and remained constant afterwards (Figure 3.14). Stabilization of the validation error indicates that the network's performance would not improve with more epochs, even though the training error continues to decline. The weights of the trained network were saved in an hdf5 file and loaded in the FlyNet GUI for the prediction step.

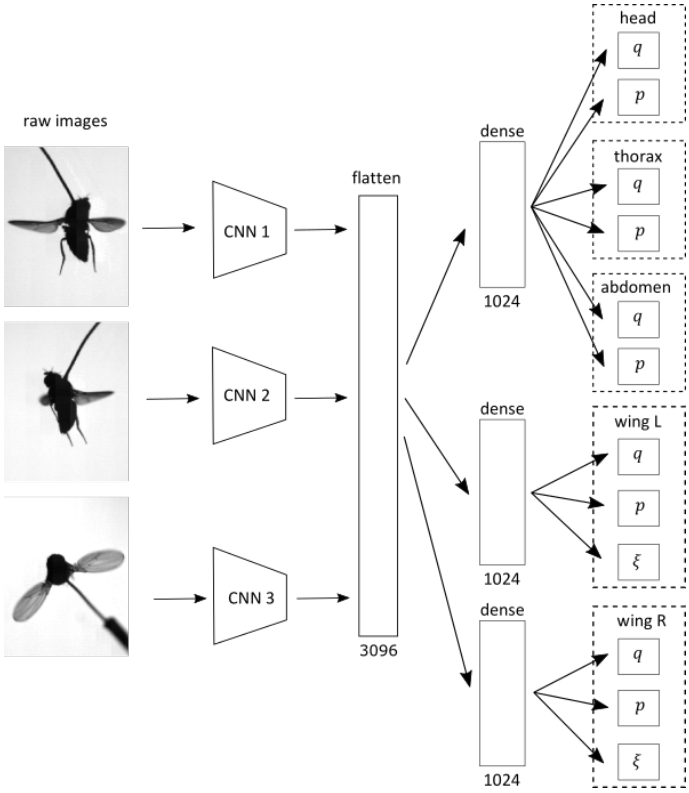

Figure 3.13: FlyNet CNN architecture: the outputs of the convolutional heads (CNN 1-3) per camera view are flattened, 3 fully connected layers (dense) with 1024 neurons and SELU activation each, represent the body, left, and right wing. The body layer predict 6 independent components: the quaternion, q, and the position, p, of the head, thorax, and abdomen. Left and right wing layers predict 3 independent components each: wing quaternion, position, and deformation angle ξ.

3.5 FlyNet: Refining pose with Particle Swarm Optimization

The neural network predictions of body and wing pose were close to the actual wing pose in the images, but not accurate enough to study the resulting aerodynamic forces. Annotating more images might resolve this issue, but it is unclear how many frames were required. Instead of improving the neural network prediction, one could also *refine* the predicted pose using 3D model fitting. This approach avoids an extensive manual annotation process while still maintaining the benefits of time-independent tracking.

Similar to Fontaine et al., 2009, image segmentation is required to create binary masks of the body and wing pixels. Instead of a median image for background subtraction, I took a minimum pixel intensity image over an image batch of 100 frames. Because the body is stationary, subtracting the median or minimum image

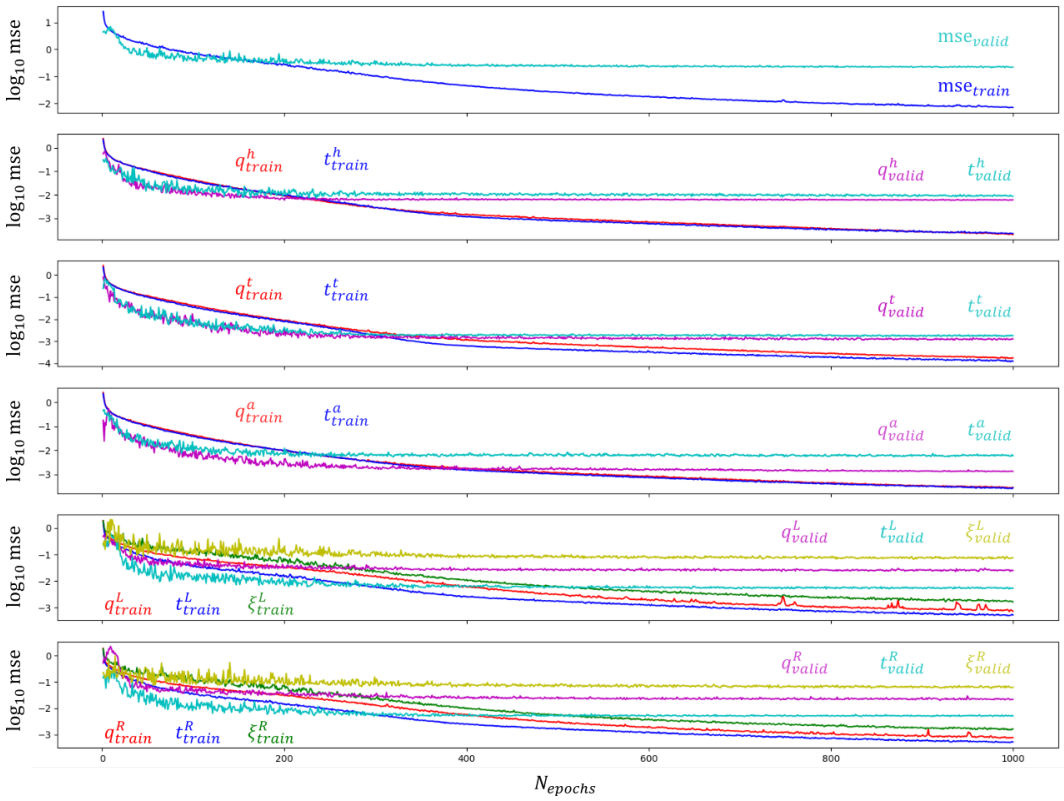

Figure 3.14: Training and validation MSE over 1000 epochs of training of FlyNet. The first row contains the total training and validation loss plotted against a log scale. Separate training and validation errors are plotted for the head (q^h, t^h), thorax (q^t, t^t), abdomen (q^a, t^a), left wing (q^L, t^L, ξ^L), and the right wing (q^R, t^R, ξ^R).

from a frame would result in the removal of the body. Instead, the body pixels were found by the simple threshold that the intensity had to be above 200. The wing pixels are found by subtracting the minimum image from the frame and subsequently selecting all pixels that had an intensity above 10. Subtracting the minimum image removed all *stationary pixels* and resulted in the selection of all pixels corresponding to moving parts (Figure 3.15).

Besides the binary masks of the body and wing pixels, 3D model fitting requires a fly model with each component scaled as closely as possible to the actual fly. The GUI used for annotating the fly pose in frames permits the scaling of model component independently. Before the automated tracking can start, the user has to scale the model components accordingly.

Instead of the ICP algorithm used by Fontaine and co-workers (2009), the 3D model fitting in my approach, uses *area matching* as a measure of how well the 3D model

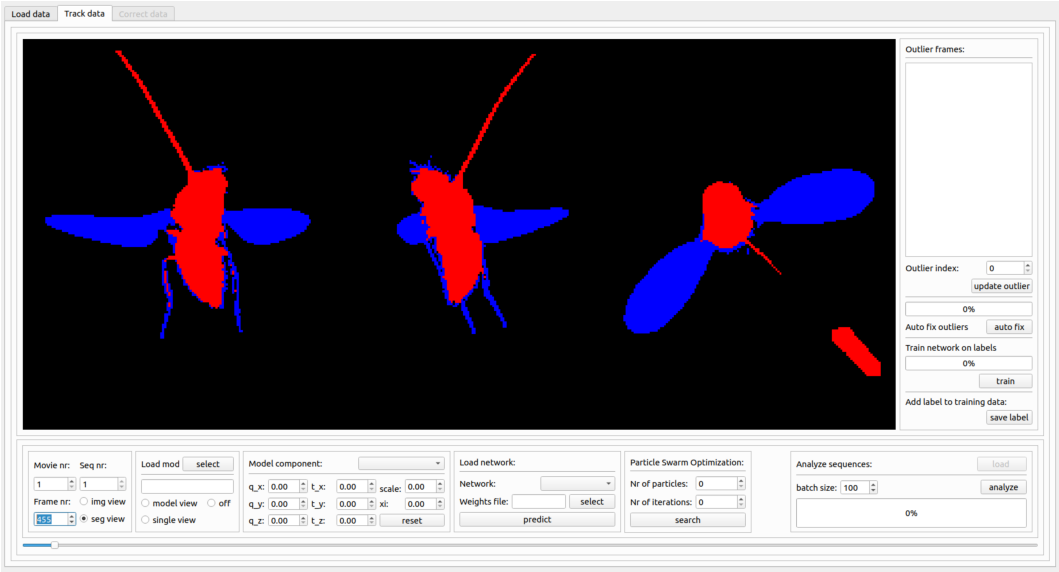

Figure 3.15: Segmented frames with body pixels in red and wing pixels in blue.

aligns with the images. The advantage of area matching is that it is a simple and computationally efficient method. Disadvantages are that the cost function is not continuous and gradient-based optimization algorithms will not work. Instead, a gradient-free optimization method is required to match the projected area of the 3D model and the frames.

A simple and robust gradient-free optimization algorithm is particle swarm optimization (PSO). PSO is a global optimization function, meaning that it has the ability to find the global minimum, even when the cost function is complex and contains multiple local minima. A global optimization function is not guaranteed to converge on the global minimum in a finite number of iterations, but for complex problems such as area matching, it is one of the few optimization methods available.

The PSO method relies on a swarm of particles that move through the state-space of the optimization problem. All particles have a position and a velocity in the state-space. The state vector consists of all variables that affect the cost function. At the start of the PSO algorithm, the particles in the swarm are given random positions and random velocity vectors. For a set number of iterations, the PSO algorithm will update the position and velocity of each particle according to the following equations:

$$x_i^k = x_i^{k-1} + v_i^k, \tag{3.2}$$

$$v_i^k = wv_i^{k-1} + c_1 r_1 (p_i^{best} - x_i^{k-1}) + c_2 r_2, (g^{best} - x_i^{k-1}), \tag{3.3}$$

where i is the particle index, x_i^k is the particle position at iteration k, v_i^k is the particle velocity, r_1 and r_2 are random weight vectors drawn from a normal distribution, p_i^{best} is the personal best of the particle, g^{best} is the global best for the whole particle swarm, and w, c_1, and c_2 are weights. A particle's personal best is the position with the lowest value for the cost function in the travel history of the particle. The global best is the position with minimum cost value in the travel history of all particles of the swarm. At each iteration, the cost function is evaluated for the current position of each particle in the swarm. Subsequently, the cost function values of the personal and global best are compared to the cost function evaluations for the current positions. If the current cost function of a particle position is lower than the personal or global best, these values and the associated position vectors will be updated. Additionally, for each iteration the random vectors r_1 and r_2 are updated by randomized weights drawn from a normal distribution.

The velocity update of each particle consists of three parts. The inertia term, $w \cdot v_i^{k-1}$, is the previous velocity multiplied with weight w. The second term points towards the personal best of the particle with some added random noise and multiplied by the scaling factor, c_1. The third term is a vector pointing towards the global best of the swarm with added random noise and multiplied by scaling factor, c_2. During the search, particles will keep updating their personal and global best positions. The two vectors pointing towards the personal and global best positions help the particles to converge towards a minimum value. Scaled randomization of the last two terms of the velocity update turn this into a stochastic search, and thus insuring that the particles will sample the space near a minimum more often. The inertia term controls how quickly the swarm converges to a global best. If the inertia term is small, the swarm will converge quickly and risks getting stuck in a local minimum. Setting the inertia term too high, however, may result in no convergence at all or convergence only after a large number of iterations.

In my particular application, the cost function that is evaluated by the PSO algorithm is based on the projected images of the 3D model. In order to speed up evaluation, the PSO algorithm and the cost function were implemented in C++. One of the big advantages of C++ is its multi-threading ability. The PSO algorithm lends itself well for parallel processing, as the cost function evaluations of all particles can be executed at the same time. Using the std-library in C++17, a separate thread for

evaluating the cost function can be assigned to each particle. Especially on central processing units (CPU) with multiple cores, multi-threading can significantly speed up computation. The cost function relies on several matrix operations that have been implemented using the Armadillo library. To integrate the C++ code within the FlyNet GUI that is written in Python, the Boost library was used to make several C++ functions callable in Python.

The Standard Tessellation Language (STL) model components of the 3D fly model consisted of a list of vertices and triangles with vertex index numbers. To position and orient the vertices of the model components, one needs to multiply the vertices with a *transformation matrix*. The transformation matrix, M, is computed using the pose vector of a component:

$$M = \begin{bmatrix} 2q_0^2 - 1 + 2q_1^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 & T_x \\ 2q_1q_2 + 2q_0q_3 & 2q_0^2 - 1 + 2q_2^2 & 2q_2q_3 - 2q_0q_1 & T_y \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & 2q_0^2 - 1 + 2q_3^2 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.4}$$

with q_0, q_1, q_2, q_3, representing the elements of the quaternion, and T_x, T_y, T_z representing the elements of the translation vector. When multiplying the vertices with the transformation matrix, an additional element with value 1 has to be added to the vertices to make the multiplication compatible:

$$\begin{bmatrix} p_x^u \\ p_y^u \\ p_z^u \\ 1 \end{bmatrix} = M \begin{bmatrix} p_x^o \\ p_y^o \\ p_z^o \\ 1 \end{bmatrix}, \tag{3.5}$$

where p_x^u, p_y^u, p_z^u are the update vertices and p_x^o, p_y^o, p_z^o the original vertices. For each body component and the leading edges of the wings, a matrix multiplication with the matrix M is sufficient to update the position of the vertices. In case of the remaining panels of the wing, however, a consecutive multiplication of transformation matrices is required. The second panel after the leading edge panel needs to undergo an additional rotation to incorporate the effect of the deformation angle. Rotation of the second panel along the hinge-line with angle $\xi/3$ can be represented by a quaternion:

$$q_{panel1} = \begin{bmatrix} \cos(\xi/6) \\ 0 \\ \sin(\xi/6) \\ 0 \end{bmatrix}. \tag{3.6}$$

Computing the transformation matrix for this quaternion gives the orientation of the second panel relative to the leading edge panel, M_{1-2}. Subsequent multiplication with the transformation matrix of the leading edge panel, M_1 puts the panel in the correct pose:

$$\begin{bmatrix} p_x^u \\ p_y^u \\ p_z^u \\ 1 \end{bmatrix} = M_1 M_{1-2} \begin{bmatrix} p_x^o \\ p_y^o \\ p_z^o \\ 1 \end{bmatrix}. \tag{3.7}$$

In a similar way, the transformation updates of panel 3 requires multiplication by 3 transformation matrices, $M_1 \cdot M_{1-2} \cdot M_{2-3}$ and panel 4 by 4 matrices, $M_1 \cdot M_{1-2} \cdot M_{2-3} \cdot M_{3-4}$. The pose vectors describing the transformation matrices of the left and right wing panels are given in Table 3.1.

pose	variable	M^L_{1-2}	M^L_{2-3}	M^L_{3-4}	M^R_{1-2}	M^R_{2-3}	M^R_{3-4}
q_0	$\cos(\xi/6)$	1	1	1	1	1	1
q_x	$\sin(\xi/6)$	0	-0.0596	-0.362	0	0.0596	0.362
q_y	$\sin(\xi/6)$	1	0.998	0.932	1	0.998	0.932
q_z	$\sin(\xi/6)$	0	0	0	0	0	0
t_x	s	0	-0.0867	0	0	-0.0867	0
t_y	s	0	0.0145	0	0	-0.0145	0
t_z	s	0	0	0	0	0	0

Table 3.1: Definition of the pose vectors describing the transformation matrices between the left, M^L, and right, M^R, wing panels. The pose column describes the different pose parameters, the variable column states the variable with which the remaining elements of the row are multiplied, with s as the scaling factor of the model component.

After updating the vertices of a model component with the transformation matrix, it can be projected to the three camera views using the world-to-camera matrices. A binary image of the projected model can be created by iterating over the projected triangles of the model and determining which pixels are within the triangle. A fast

method to compute whether a pixel is within the triangle makes use of *barycentric coordinates* (Figure 3.16). Given a triangle with corners A, B, and C specifying the u, v-coordinates in the image frame, the barycentric coordinates (r, s) of a point P are given by:

$$\begin{aligned} cc &= (C - A) \cdot (C - A), \\ bc &= (B - A) \cdot (C - A), \\ pc &= (P - A) \cdot (C - A), \\ bb &= (B - A) \cdot (B - A), \\ pb &= (P - A) \cdot (B - A), \end{aligned} \tag{3.8}$$

and

$$\begin{aligned} r &= \frac{bb \cdot pc - bc \cdot pb}{cc \cdot bc - bc \cdot bc}, \\ s &= \frac{cc \cdot pb - bc \cdot pc}{cc \cdot bc - bc \cdot bc}. \end{aligned} \tag{3.9}$$

A point is within the triangle when the following conditions are met:

$$\begin{aligned} r &>= 0, \\ s &>= 0, \\ r + s &< 1. \end{aligned} \tag{3.10}$$

A binary image of the projected triangles is created using barycentric; coordinates, for each triangle a bounding box is computed and all the pixels within the bounding box are tested for being inside or outside the triangle. If a pixel is within the triangle its value is set to 1. By iterating over all triangles in the model, a binary image is constructed.

With binary masks of the high-speed video frames and the projected 3D model, the cost function can be computed. For each model component, the cost function uses the *symmetric difference*, Δ, and the *union*, $\cup$. In order to speed up the computation of the symmetric difference and union, the binary images are converted into *bitsets*. Using the `dynamic_bitset` class of the Boost library, the union and difference for binary

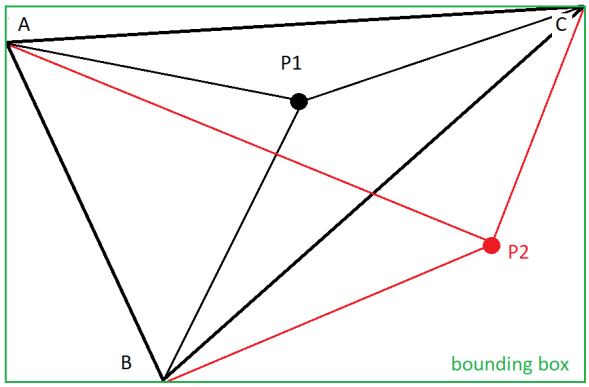

Figure 3.16: Barycentric coordinates of points P1 and P2 with respect to the triangle ABC. The bounding box of ABC is marked in green.

images of all three views can be computed within 10 µs. This is orders of magnitudes faster than a `for` loop in C++ can achieve. A `dynamic_bitset` vectorizes the binary image into a single bit-vector. Computing the union and symmetric difference are bitwise operations, which are extremely fast on a CPU. As the cost function is being computed over all three views simultaneously, the binary images can be concatenated into a single `dynamic_bitset` of $256 \times 256 \times 3 = 196,608$ bits.

The pose vector consists of 37 dimensions, which is a large parameter-space to search. For example, a grid search with 2 possible values per parameter would require the investigation of 2^{37} different configurations, requiring more than 137 billion cost function evaluations. The complexity of the PSO search can be greatly reduced by splitting the pose vector into different components: head, thorax, abdomen, left wing, and right wing. In my implementation, there are five separate PSO searches with 7 or 8 state parameters each. Although the pose vectors of the five model components are independent, the cost function evaluation is not, as different model components can overlap in the projected view of the 3D model. To address this dependence, the PSO algorithm updates the pose of one randomly selected component at the time. Because the accuracy of wing motion tracking is more important than body motion the probability of selecting the left and right wing is $1/3$ and the probability of selecting a body component is $1/9$.

The cost function per model component is given by:

$$C_j = \frac{I^b \Delta I_j^p}{I^b \cup I_j^p}, \tag{3.11}$$

for body components, and

$$C_j = \frac{I^w \Delta I_j^p}{I^w \cup I_j^p}, \tag{3.12}$$

for wing components. Here, C_j, the cost function for model component j, I^b, and I^w the dynamic bitsets of the body and wing pixels respectively, and, I_j^p the dynamic bitset of the model component projections in all 3 views, and Δ and $\cup$ are the bitwise symmetric difference and union operators, respectively. During the PSO search, the personal best of a particle consists of a 5 element cost vector with the minimum cost value per component and, with the pose vectors per component corresponding to the minimum cost values. Similarly, the global best has a 5 element cost vector with corresponding component pose vectors. Because only one model component is evaluated per iteration, the personal and global best pose vectors are corresponding to minimum cost values of different iterations.

The C++ implementation of the PSO algorithm described in the previous paragraph is wrapped into several Python functions, using the Boost library, that are callable by FlyNet. Before the tracking can start, some PSO parameters need to be set: the number of particles, the number of iterations, and the standard deviation of the normal distribution, which is used to perturb the initial state and the PSO search parameters. A large number of particles and/or iterations increases the probability that the PSO search converges to a global minimum, but at the cost of longer run times. As a rule of thumb, a PSO algorithm should have at least twice the number of particles as the number of states. I installed FlyNet on computers with 16 and 32 CPU cores, and the fastest execution times I achieved was when using 1 particle per core. Because the number of states per component is 8 at most, 16 particles are sufficient to search the parameter space. The number of particles was therefore set to the number of CPU cores available. After testing several numbers of iterations, I was found that the PSO algorithm was likely to converge on the true pose within 300 iterations. In cases where the PSO algorithm did not converge to the true state in 300 iterations, it would not converge after 1000 or 2000 iterations either. Whether the PSO algorithm converges to the true pose thus depends on how close the initial pose vectors of the particles are to the true pose.

The initial particle pose vectors were created by adding random noise to the predicted pose vector from the CNN. Similarly, the particle velocities were randomly sampled from a normal distribution. By specifying the standard deviation of the normal

distribution used to generate the random noise vectors, one can specify how close the particles search around the initial pose. For FlyNet, a standard deviation of 0.1 was chosen, which means that most particles will search around the initial pose value.

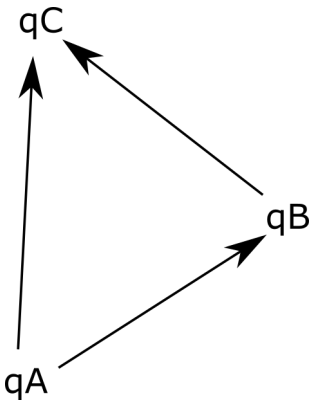

Figure 3.17: Quaternion multiplication of q_A and q_B yields quaternion q_C.

The variables in the pose vector are all linear, except for the quaternion which is non-linear. A unit quaternion, $\|q\| = 1$, can be seen as a point on a 4D sphere with radius 1. When updating the quaternion according to equations 3.2 and 3.3, the result is likely to be not a unit quaternion, however. Instead of the vector difference pointing towards the local and global best, a *quaternion difference* is required to make the particles move on the 4D sphere with radius 1. Determining the quaternion difference requires application of the concepts of quaternion multiplication and the quaternion conjugate, (Kuipers, 1999). In quaternion multiplication, two quaternions, q_A and q_B, are multiplied forming quaternion q_C:

$$q_C = q_A \otimes q_B = \begin{bmatrix} q_{A0} & -q_{Ax} & -q_{Ay} & -q_{Az} \\ q_{Ax} & q_{A0} & q_{Az} & -q_{Ay} \\ q_{Ay} & -q_{Az} & q_{A0} & q_{Ax} \\ q_{Az} & q_{Ay} & -q_{Ax} & q_{A0} \end{bmatrix} \begin{bmatrix} q_{B0} \\ q_{Bx} \\ q_{By} \\ q_{Bz} \end{bmatrix}, \tag{3.13}$$

where $\otimes$ indicates quaternion multiplication. The quaternion conjugate describes the opposite rotation from the quaternion:

$$q^* = \begin{bmatrix} q_0 \\ -q_x \\ -q_y \\ -q_z \end{bmatrix}. \tag{3.14}$$

The quaternion difference can be derived using the following property of the quaternion conjugate:

$$q^* \otimes q = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.15}$$

which makes sense given that the quaternion conjugate opposes the rotation of the quaternion resulting in no rotation. Suppose the particle position is given by quaternion q_A and the personal best is q_C. The difference between q_A and q_C is described by quaternion q_B (Figure 3.17). Using the quaternion conjugate, the quaternion difference can be derived from quaternion multiplication:

$$\begin{aligned} q_C &= q_A \otimes q_B, \\ q_A^* \otimes q_C &= q_A^* \otimes q_A \otimes q_B, \\ q_B &= q_A^* \otimes q_C. \end{aligned} \tag{3.16}$$

Although the quaternion difference describes the rotation required to move from one orientation to another, quaternion algebra is still required to compute the velocity and position updates of the PSO algorithm. The temporal derivative of a quaternion, $\dot{q}$, is the quaternion multiplication with the angular velocity, ω:

$$\dot{q} = \frac{1}{2} q^* \otimes \omega = \frac{1}{2} \begin{bmatrix} q_0 & q_x & q_y & q_z \\ -q_x & q_0 & q_z & -q_y \\ -q_y & -q_z & q_0 & q_x \\ -q_z & q_y & -q_x & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{3.17}$$

Angular velocity can be extracted from the difference quaternion using the following equations:

$$
\begin{aligned}
\hat{\omega} &= 2\arccos(q_0), \\
\|e\| &= \sqrt{q_x^2 + q_y^2 + q_z^2}, \\
\omega_x &= \hat{\omega} \cdot q_x / \|e\|, \\
\omega_y &= \hat{\omega} \cdot q_y / \|e\|, \\
\omega_z &= \hat{\omega} \cdot q_z / \|e\|.
\end{aligned}
\tag{3.18}
$$

The angular velocity does not require quaternion algebra to be added or subtracted and can therefore be used in equation 3.3. After computing the angular velocity update with equation 3.3, the particle quaternion needs to be updated according to the following formula:

$$
\begin{aligned}
q_0 &= \cos(\|\omega\|/2), \\
q_x &= \sin(\|\omega\|/2) \cdot \omega_x / \|\omega\|, \\
q_y &= \sin(\|\omega\|/2) \cdot \omega_y / \|\omega\|, \\
q_z &= \sin(\|\omega\|/2) \cdot \omega_z / \|\omega\|.
\end{aligned}
\tag{3.19}
$$

As mentioned before, unit quaternions reside on a 4D sphere with radius 1. The definition of a quaternion allows for a *duality*, however; a rotation, θ, around axis e corresponds to the same orientation as a rotation, $-\theta$, around axis $-e$. This duality is undesirable as it can confuse the PSO algorithm. To exclude quaternion duality from the analysis, only positive rotations around the quaternion rotation axis are allowed by using the norm $\|\omega\|$ in equation 3.19.

To keep the particles searching in *feasible space*, the search space needs to be bounded. The unit quaternion constraint, $\|q\| = 1$, which is enforced by normalization after every quaternion update, ensures that all particles search in feasible orientation space. For the translation vectors, a bounding box of ±3 mm in the world reference frame guarantees that particles do not stray too far from feasible solutions. When the update of particle position moves outside the bounding box, the particle is clamped to the bounding box border. The personal and global best vector will subsequently pull the particle back from the bounding box border. A similar bounding box is defined for the deformation angle, ξ, with borders at $\pm\pi/2$.

During the refinement step, in my implementation the PSO algorithm ran for a fixed number of iterations, 300, and after that the global best pose was used as the *refined*

pose. With 300 iterations and 16 or more particles, the PSO algorithm converged on the true (manually annotated) pose if the CNN prediction was sufficiently close. Expanding the number of particles and iterations did guarantee convergence on the true solution, at least not for runs using 256 particles and 2000 iterations. The easiest way to improve tracking performance was through manual annotation of problematic frames and retraining of the CNN on the expanded dataset. Over the course of the experiments, the manually annotated dataset increased from approximately 1000 labels to 2500 labels. With this expanded training set, it was rare that FlyNet could not track a frame and it was usually only unusual patterns of wing motion (e.g. flight stops) that were not captured accurately.

The computational performance of FlyNet varied depending on the computer I used, but is primarily dependent on the CPU. Prediction of the initial pose vector by the CNN took less than 100 ms for a batch of 100 frame triplets. The refinement step took around 80 ms per frame triplet on a 32 core AMD Ryzen Threadripper CPU, and it took approximately 20 minutes to analyze a high-speed video sequence of 1.1 seconds. This processing speed was sufficient to analyze the full dataset of 479 videos with a desktop computer.

An overview of the FlyNet workflow is shown in (Figure 3.18). The neural network prediction and PSO refinement complement each other: the neural network prediction ensures time-independent tracking while the PSO refinement reduces the amount of manual annotation that is required to attain the required accuracy. FlyNet is specifically designed to track the body and wing motion of tethered fruit flies. The methodology could be used for accurate 3D pose reconstruction from multiple camera views in general, however. It is relatively easy to import STL models of other insects in the FlyNet GUI, and it might be valuable to evaluate FlyNet's tracking performance on other species.

3.6 FlyNet: Smoothing pose with Kalman Filters

After automated tracking by the FlyNet algorithm, a matrix of 37×16375 elements describes the pose vector during a high-speed video. The time-independence of the FlyNet algorithm is useful as it does not propagate tracking errors over time, but ignoring the temporal aspects of body and wing motion means that some information is discarded. The temporal information can be used to improve the accuracy of pose reconstruction and reduce the effects of outliers. In this section, I will discuss how I used linear and extended Kalman filters (EKF) to smooth the pose traces and

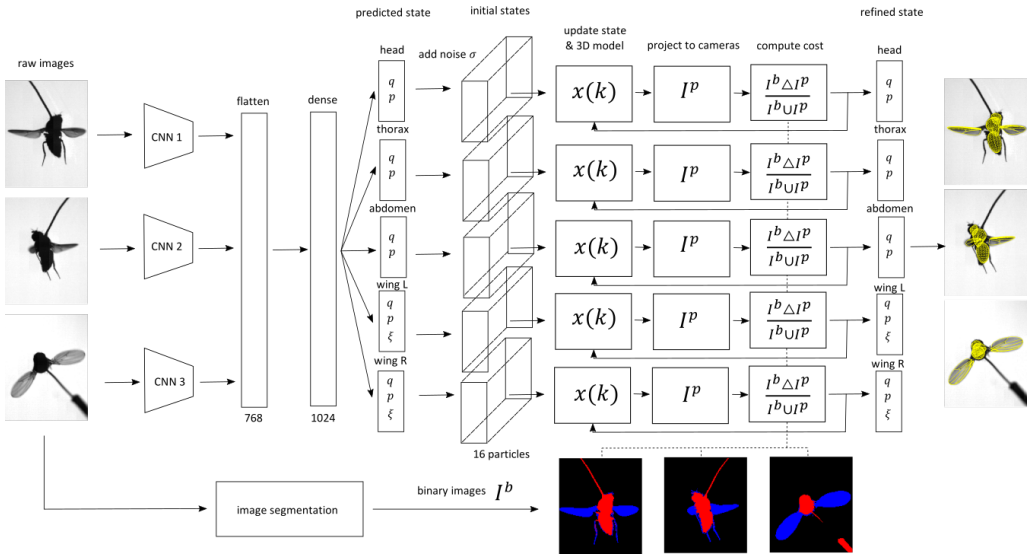

Figure 3.18: Workflow FlyNet: the convolutional blocks per camera view are indicated by CNN1-3, the predicted state vector consists of 5 components with a quaternion, q, translation vector, t and deformation angle ξ. The refinement step starts by randomly selecting a model component and subsequently modifying the pose according to the PSO update. The updated 3D model the gets projected on the camera views and a binary image, I^p, is computed. The symmetric difference and union of the projected model image and the segmented body or wing, binary images are computed and used to calculate the: cost, personal, and global best values of the particles are updated and the next PSO iteration starts, after 300 iterations the global best pose is the refined pose.

dampen the effect of outliers.

Kalman filters can use noisy measurements over time in combination with a dynamical model of the measured system to predict the future state of the system, (Stratonovich, 1959, Kalman, 1960). To filter the pose traces, the Kalman filter was used as a *smoother* and the emphasis is not on accurately predicting the future state but on estimating the most likely value for a sample given the time history of the entire trace and the uncertainties of the measurements and system model.

There are different versions of the Kalman filter, dependent on whether the dynamic model of the system is linear, non-linear, or unknown. Similar to the PSO update, the dynamic model of quaternion propagation through time is non-linear and requires an EKF as a smoother. The dynamic models of position and the deformation angle are linear and can be filtered by a linear Kalman filter. However, because a linear dynamic model in an EKF corresponds to linear Kalman filtering, it is easier to incorporate the quaternion, position, and deformation angle in a single dynamic

model and filter the pose vector with an EKF. In the following section the EKF algorithm will be presented (Grewal and Andrews, 2014).

A Kalman filter consists of a prediction phase and a measurement update phase. The predicted state, $\hat{x}_{k+1}$, is given by:

$$\hat{x}_{k+1} = f(x_k) + w_k, \tag{3.20}$$

with k as the sample number, $f(x_k)$ the non-linear dynamic model and w_k the system noise. System noise is assumed to be normal distributed with covariance matrix Q:

$$w_k = \mathcal{N}(0, Q_k). \tag{3.21}$$

In a similar way, the measurement model is given as:

$$z_k = Hx_k + v_k, \tag{3.22}$$

with z_k as the measurement, H as the measurement model, and v_k the measurement noise that is assumed to be normal distributed with covariance matrix R:

$$v_k = \mathcal{N}(0, R_k). \tag{3.23}$$

In the EKF definition, the non-linear system model is linearized around state x_k:

$$\hat{x}_{k+1} = \frac{\partial f(x_k)}{\partial x} x_k + w_k = \Phi_k x_k + w_k, \tag{3.24}$$

with $\Phi_k = \frac{\partial f(x_k)}{\partial x}$ as the Jacobian matrix. The *a priori* estimate, indicated by (-), is given by:

$$x_{k+1}^- = \Phi_k x_k^+, \tag{3.25}$$

where (+) stands for the filtered state of the previous sample. The predicted error covariance, P_k^-, is computed using:

$$P_k^- = \Phi_k P_k^+ \Phi_k^T + Q_k, \tag{3.26}$$

with P_k^+ as the filtered error covariance from the previous sample. After the prediction phase, the measurement update phase starts by computing the filter gain, K_k:

$$K_k = P_k^- H^T [H P_k^- H^T + R_k]^{-1}. \tag{3.27}$$

Using the filter gain, the state is updated by:

$$x_k^+ = x_k^- + K_k [z_k - H x_k^-], \tag{3.28}$$

and the updated error covariance

$$P_k^+ = [I - K_k H] P_k^-, \tag{3.29}$$

where I is the identity matrix. By alternating between the prediction and update phase, the Kalman filter removes noise from the traces by comparing measurements and dynamic model-based predictions. The model based predictions in the Kalman filter create a temporal lag in the filtered traces. The Rauch-Tung-Striebel smoother is most commonly used for removing this lag with a *backward pass*. After the *forward pass* by the EKF described above, the Rauch-Tung-Striebel smoother will start at the last sample in a sequence. The smoothed state, $\hat{x}_k$ is updated according to:

$$\hat{x}_k = x_k^+ + A_k (\hat{x}_{k+1} - x_{k+1}^-), \tag{3.30}$$

where x_k^+ and x_{k+1}^- are the saved values from the forward pass and

$$A_k = P_k^+ \Phi_k^T P_{k+1}^-, \tag{3.31}$$

is constructed with the saved error covariances of the forward pass. Additionally, the smoothed error covariance can be computed in the backward pass using:

$$\hat{P}_k = P_k^+ + A_k (\hat{P}_{k+1} - P_{k+1}^-). \tag{3.32}$$

A user can control the degree of smoothing by setting the covariance matrices Q and R. In most cases the covariance matrices are unknown and R is typically set

as the identity matrix. By setting the diagonal values of the Q matrix, the user can set the degree of confidence in the dynamic model relative to the measurements. A smaller value of Q means more confidence in the dynamic model and as a result a stronger smoothing of the trace.

In my implementation, the dynamic model used to filter body and wing pose describes the relation between: position, velocity, and acceleration for the translation vector, the orientation, angular velocity, and angular acceleration for the quaternion, and the deformation angle. For example, the state vector, x_k, for the left wing is given by:

$$x_k = [\dot{\omega}_x\ \dot{\omega}_y\ \dot{\omega}_z\ \omega_x\ \omega_y\ \omega_z\ q_0\ q_x\ q_y\ q_z\ a_x\ a_y\ a_z\ \ddot{\xi}\ v_x\ v_y\ v_z\ \dot{\xi}\ p_x\ p_y\ p_z\ \xi]^T, \quad (3.33)$$

where v is the velocity, a the acceleration, ω the angular velocity, $\dot{\omega}$ the angular acceleration, $\dot{\xi}$ and $\ddot{\xi}$ the first and second temporal derivatives of ξ, respectively. The state vector is used to compute the Jacobian of the system update equation. Unfortunately, the full Jacobian, Φ, is too large to display and is therefore split into two independent blocks: Φ^q and $\Phi^{t,\xi}$. The first block of the system matrix is given by:

$$\Phi^q = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\Delta t & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \Delta t & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \Delta t & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
-q_x\frac{\Delta t^2}{4} & -q_y\frac{\Delta t^2}{4} & -q_z\frac{\Delta t^2}{4} & -q_x\frac{\Delta t}{2} & -q_y\frac{\Delta t}{2} & -q_z\frac{\Delta t}{2} & 1 & -\omega_x\frac{\Delta t}{2} & -\omega_y\frac{\Delta t}{2} & -\omega_z\frac{\Delta t}{2} \\
q_0\frac{\Delta t^2}{4} & q_z\frac{\Delta t^2}{4} & -q_y\frac{\Delta t^2}{4} & q_0\frac{\Delta t}{2} & q_z\frac{\Delta t}{2} & -q_y\frac{\Delta t}{2} & \omega_x\frac{\Delta t}{2} & 1 & -\omega_z\frac{\Delta t}{2} & \omega_y\frac{\Delta t}{2} \\
-q_z\frac{\Delta t^2}{4} & q_0\frac{\Delta t^2}{4} & q_x\frac{\Delta t^2}{4} & -q_z\frac{\Delta t}{2} & q_0\frac{\Delta t}{2} & q_x\frac{\Delta t}{2} & \omega_y\frac{\Delta t}{2} & \omega_z\frac{\Delta t}{2} & 1 & -\omega_x\frac{\Delta t}{2} \\
q_2\frac{\Delta t^2}{4} & -q_x\frac{\Delta t^2}{4} & q_0\frac{\Delta t^2}{4} & q_2\frac{\Delta t}{2} & -q_x\frac{\Delta t}{2} & q_0\frac{\Delta t}{2} & \omega_z\frac{\Delta t}{2} & -\omega_y\frac{\Delta t}{2} & \omega_x\frac{\Delta t}{2} & 1
\end{bmatrix}, \quad (3.34)$$

which corresponds to the first 10 values of the state vector: $[\dot{\omega}_x\,\dot{\omega}_y\,\dot{\omega}_z\,\omega_x\,\omega_y\,\omega_z\,q_0\,q_x\,q_y\,q_z]$. As the frame rate of the high-speed cameras is 15,000 fps, the time step is $\Delta t = 1/15000s$. The second block is given by:

$$\Phi^{p,\xi} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \Delta t & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \Delta t & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \Delta t & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Delta t & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{\Delta t^2}{2} & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{\Delta t^2}{2} & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.35}$$

corresponding to the remaining values of the state vector $[a_x\, a_y\, a_z\, \ddot{\xi}\, v_x\, v_y\, v_z\, \dot{\xi}\, p_x\, p_y\, p_z\, \xi]$. This tracking process does not involve any temporal derivatives and the measurement matrix need only to select the state variables that have been tracked. The measurement matrix is given by:

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.36}$$

As mentioned, the measurement covariance matrix, R, is set to identity. The diagonal values of Q are set to 10^{-5} for the head, thorax, and abdomen, and are set to 10^{-2} for the left and right wings.

Pose vectors of each model component are filtered by a forward and backward pass. Besides filtering the pose vectors, the Kalman filter yields the (angular) velocities and accelerations of the model components.

3.7 FlyNet: Representing wing motion in the Strokeplane Reference Frame

Although mathematically preferable, quaternions do not provide an intuitive sense of wing motion. A more intuitive definition of wing motion is a set of Tait-Bryan angles relative to a thorax-fixed reference frame. Muijres and co-workers (2014) defined a reference frame at a fixed angle of 45° relative to the body's longitudinal axis. In case of tethered flight, however, the head and abdomen can be out of the symmetry plane for prolonged times. This is problematic, as the body's longitudinal axis does not necessarily align with the symmetry plane of the thorax. Using the pose vector of the thorax is problematic as well, as the tracking of the thorax quaternion is not sufficiently accurate.

Instead of determining the longitudinal axis, a reference frame can be derived from the positions of the left and right wing roots. Because the fly is tethered, the thorax is fixed in space and the wing roots move in a c-shaped trajectory around the thorax (Figure 3.19). By performing PCA on the left and right root traces, a strokeplane reference frame (SRF) may be conveniently defined. The first principal component (PC) corresponds to the y-axis of the SRF, the second PC forms the x-axis, and the third PC the z-axis. Whereas the PCs describe the orthogonal axes of the SRF, they do not give consistent directions. To create a reference frame that has the x-axis pointing forward and the z-axis pointing upward, the thorax pose vector is used to establish directionality.

With the SRF defined, the pose of the left and right wings can be described in the SRF. Using the normalized axes of the SRF, it is possible to compute the quaternion of the SRF, q_{SRF}. The left and right wing quaternions relative to the SRF can be obtained by multiplying the left and right wing quaternions with the conjugate, q^*_{SRF}. The left and right wing root traces can be expressed relative to the SRF by subtracting the origin location of the SRF and subsequently multiplying with the rotation matrix of q^*_{SRF}.

The mechanics of the wing hinge are complex and the root of the wing is not realistically approximated as a single point. This is why I choose to not define any constraints between the wing and thorax in FlyNet. To simplify the analysis of wing motion, however, the motion of the wing root will be ignored in the remainder of the thesis. It could be that there is valuable information about the mechanics of the wing hinge in the wing root trajectories, but the added complexity of wing root motion makes interpreting the wing kinematics less intuitive. For analysis purposes, the wing joint is assumed to be a ball joint on a fixed position on the thorax with

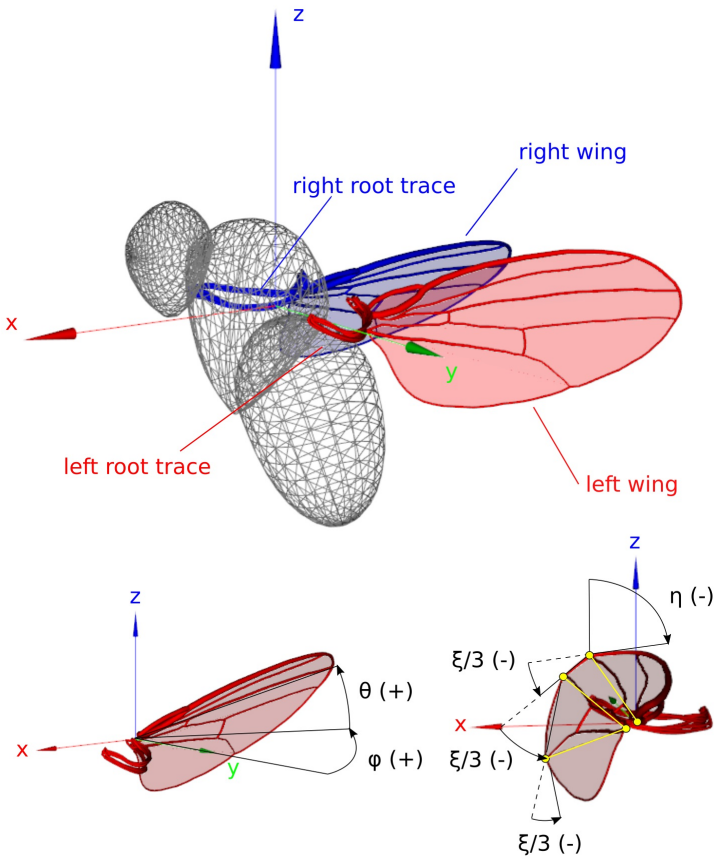

Figure 3.19: Strokeplane reference frame (x, y, z), is computed via PCA from the left and right root traces. The stroke (ϕ) and deviation (θ) angles describe the orientation of the wing tip relative to the y-axis and the $x - y$ plane, respectively. Wing pitch (η) and deformation (ξ) angles give the orientation of the leading edge panel relative to the $z - axis$ and the rotation angle along 3 hinge lines (yellow), respectively. The rotation sign is indicated by (+) and (−).

3 degrees of freedom. From an aerodynamics perspective, this assumption is not likely to change computed or measured aerodynamic forces substantially, as the arm of the wing root is relatively small and does not increase wing velocity significantly.

The orientation of the wing is described by three Tait-Bryan angles: the stroke angle, ϕ, that describes the angle between the y-axis of the SRF and the projection of the wingtip on the x-y plane, the deviation angle, θ, which is the angle between the wingtip and the x-y plane and the wing pitch angle, η, which specifies the orientation of the leading edge panel of the wing with respect to the z-axis (Figure 3.19). Wing shape is described by ξ and remains unaltered from the FlyNet definition. The wing kinematic angles of the right wing are similarly defined such that the signs

are symmetric with respect to the left wing. Euler and Tait-Bryan angles are non-commutative, meaning that the order of rotations matters. The rotation convention used is first ϕ, then θ and finally η. Using quaternions, this can be described as:

$$q_{wingL} = q_\eta \otimes q_\theta \otimes q_\phi = \begin{bmatrix} \cos(\eta/2) \\ 0 \\ \sin(\eta/2) \\ 0 \end{bmatrix} \otimes \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2) \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} \cos(\phi/2) \\ 0 \\ 0 \\ \sin(\phi/2) \end{bmatrix}. \tag{3.37}$$

For the right wing kinematic angles, the signs of ϕ and θ are negative, following the definitions of right-handed rotations:

$$q_{wingR} = q_\eta \otimes q_\theta \otimes q_\phi = \begin{bmatrix} \cos(\eta/2) \\ 0 \\ \sin(\eta/2) \\ 0 \end{bmatrix} \otimes \begin{bmatrix} \cos(-\theta/2) \\ \sin(-\theta/2) \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} \cos(-\phi/2) \\ 0 \\ 0 \\ \sin(-\phi/2) \end{bmatrix}. \tag{3.38}$$

With the wing kinematic angles defined, one can parse the left and right wing motion into distinct wingbeats. A wingbeat is defined as the time between two subsequent dorsal stroke reversals. To find the dorsal stroke reversal point, I computed the derivative of the stroke angle, $\partial\phi/\partial t$. Both the dorsal and ventral stroke reversal points can be found at $\partial\phi/\partial t = 0$. Therefore, the condition that $\phi > 0$ is used to find the dorsal reversal times of the left and right wings in each video sequence. The dorsal reversal time of the left wing does not necessarily align with the right wing, and can be up to 3 frames apart. To remedy the phase difference, I computed an average time point between the dorsal reversals of the left and right wing and subsequently rounded to the closest frame time point.

Per video sequence, the average dorsal stroke reversal times were determined and used to parse out the wingbeats in a sequence. Parameters such as frequency period and frame numbers were computed per wingbeat. All wing kinematic angles during the wingbeat period plus temporal data were stored per wingbeat in a hdf5-file.

3.8 FlyNet: Encoding wing motion with Legendre polynomials

Depending on the activation level of the power muscles, wingbeat frequencies can vary between 150 Hz and 250 Hz. This variation in wingbeat duration makes it difficult to compare wing kinematics between different high-speed video sequences.

At a wingbeat frequency of 200 Hz, 75 high-speed video frames are captured per wingbeat. As there are four kinematic angles per wing, there are 300 data points representing wing motion during a wingbeat. In order to reduce the number of data points per wingbeat and allow for comparison between wingbeats of different periods, I used Legendre polynomials to parameterize wing kinematic traces.

Legendre polynomials are well suited for fitting wing kinematics, as they can fit non-periodic traces and asymmetric wave forms. Fourier series, that are often used for fitting wave forms, require periodic boundary conditions and are biased to symmetric wave forms when a low number of harmonics is used. Similar to Fourier series, Legendre polynomials form a complete and orthogonal system.

An easy way to generate a Legendre polynomial is using Rodrigues' formula:

$$P_n(x) = \frac{1}{2^n} \sum_{k=0}^{n} \binom{n}{k}^2 (x-1)^{n-k}(x+1)^k, \tag{3.39}$$

where n is the order of the Legendre polynomial and x ranges between $[-1, 1]$. A *Legendre basis* of order n is formed by all polynomials from order 0 up to order n (Figure 3.20). By specifying sample points in the range $x = [-1, 1]$ and computing the values for each polynomial in the Legendre basis, a *Vandermonde* matrix is created. The Vandermonde matrix, X, can be used in a least-squares fit of a wing kinematic trace, Y:

$$\beta = (X^T X)^{-1} X^T Y, \tag{3.40}$$

where β is a vector of $n + 1$ coefficients corresponding to the order of the Legendre basis. With sufficient coefficients it is possible to fit any trace throughout a wingbeat without error. When using to many coefficients however, the *Runge phenomenon* can occur, which means that the polynomial fit shows high frequency oscillations around the edges of the wingbeat. The Runge phenomenon may be remediated by lowering the order of the Legendre basis and by imposing boundary conditions at the start and end of the wingbeat.

Boundary conditions on polynomial fits can be imposed using *restricted least-squares*, (Dykstra, 1983). The restricted least-squares fit, β^*, makes use of the least squares fit, β, and restriction matrix, R, and restriction vector r:

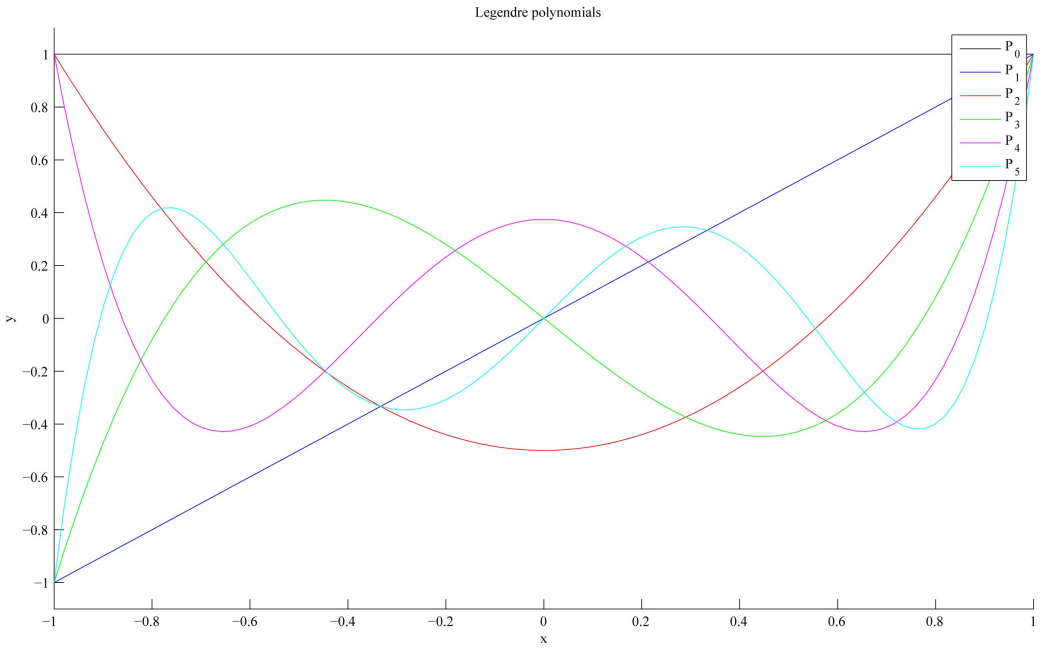

Figure 3.20: Legendre basis of $n = 5$ with the Legendre polynomials given by P_n.

$$\beta^* = \beta - (X^T X)^{-1} R^T (R(X^T X)^{-1} R^T)^{-1} (R\beta - r). \tag{3.41}$$

To reduce the Runge phenomenon, the restriction matrix and vectors are chosen such that the transition between subsequent wingbeats is continuous up to the fourth derivative. The j^{th} derivative of a Legendre polynomial can be computed by:

$$P_n^j(x) = n \cdot P_{n-1}^{j-1} + x \cdot P_{n-1}^j. \tag{3.42}$$

Using the equation above, Legendre bases of the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} derivative are computed. The 1^{st}, 2^{nd}, 3^{rd}, and 4^{th} derivatives of the actual wing kinematic trace are computed over a 9 frame time window centered around the start and end time points of the wingbeat. By requiring that the Legendre fit matches the actual values of the wing kinematic trace and the derivatives, it can be used as the restriction vector and restriction matrix respectively:

$$R\beta = r. \tag{3.43}$$

The restriction matrix, R, contains the Legendre basis and the derivative Legendre bases up to the fourth derivative for $x = -1$ and $x = 1$ (start and end of the

wingbeat). Matching values of the wing kinematic trace and the 1^{st}, 2^{nd}, 3^{rd}, and 4^{th} derivatives are contained in the restriction vector r. Inserting R and r in the restricted least-squares solution of equation 3.41 gives the Legendre coefficients that make the transitions between the previous and next wingbeat continuous up to the fourth derivative.

With the restricted least-squares solution, one can fit with higher order Legendre polynomials before the Runge-phenomenon starts to occur. Using higher order polynomials is preferable as it allows for a more accurate fit. For each wing kinematic angle, I tested various number of Legendre polynomials. The stroke angle, ϕ, could be fitted accurately with 16 polynomials. Deviation, θ, and deformation, ξ, angles require 20 polynomials each. The wing pitch angle, η, needed 24 polynomials for accurate fitting. A total of 80 Legendre coefficients describes the motion of a single wing during a wingbeat.

Chapter 4

CONVOLUTIONAL NEURAL NETWORK REGRESSION BETWEEN MUSCLE ACTIVITY AND WING MOTION

The workflow of the experiments and analysis so far consists of the simultaneous recording of wing motion and muscle fluorescence, reconstructing body and wing pose from the three orthogonal views using FlyNet, smoothing pose using an EKF, extracting four wing kinematic angles per wing and parsing a video sequence into separate wingbeats, and fitting Legendre polynomials to the wing kinematic traces via restricted least-squares. To construct a model of how the steering muscles affect wing motion, one needs to find correlations between muscle fluorescence and changes in wing motion. Finding these correlations is difficult for several reasons. The temporal resolution of the muscle fluorescence recordings and the high-speed videos differs by two orders of magnitude, the kernel of GCaMP fluorescence takes more than a second to return to baseline and the changes in wing motion can be subtle yet aerodynamically significant (Figure 2.3).

In neuroscience, a common methodology to increase the temporal resolution of fluorescence traces is by deconvolving the fluorescence signal with an estimated function of the GCaMP fluorescence kernel. When the estimated fluorescence kernel matches well with actual fluorescence kernel, the deconvolution operation yields clear peaks corresponding to muscle twitches. It is difficult to estimate the GCaMP fluorescence kernel, however, as it depends on temperature, chemical composition of the cytoplasm, and genetic expression levels. If the kernel function that is used for the deconvolution operation is different from the actual GCaMP fluorescence kernel, the deconvolved signal can contain a highly inaccurate measure of muscle activity. Because of the high uncertainty of deconvolution results, another method to remediate the difference in temporal resolution is required.

In this chapter, I will discuss how a CNN can be used to learn the actual GCaMP fluorescence kernel and perform non-linear regression between muscle activity and wing motion. The CNN regression is surprisingly accurate; for most flies in the dataset, the wing motion during the high-speed videos can be reconstructed accurately from the muscle fluorescence of the 12 steering muscles. CNN prediction of wing motion deviates from the tracked motion when the fly is flying at low, wingbeat

frequencies or has stopped flying, or when the strobing of the muscle fluorescence camera was erroneous. By performing a correlation analysis on muscle activity, I found that muscle activity is highly correlated and that muscle activity resides on a 12D plane. Using the 12D-plane constraint, I explored the predicted wing motion corresponding to maximum coordinated muscle activity patterns. Comparison of the CNN-predicted wing motion for maximum coordinated muscle activity patterns and previous studies shows that the steering muscles that are accessible to electrode recording, have similar effects on wing motion as that are predicted by my model. For the steering muscles that are not accessible to electrodes, the CNN makes novel predictions about their effects on wing motion.

4.1 Scaling of muscle activity and wing motion

Before a CNN can be trained on the muscle and wing motion data, the dataset needs to be prepared. The magnitude of the muscle activity patterns found by the muscle unmixing algorithm varies, as GCaMP expression, the translucency of the thorax, and blue light illumination strength, all of which can vary across preparations. During each experiment, all fluorescence images, muscle unmixing output, and flight state parameters were saved to an hdf5-file. An experimental session typically lasted for at least 15 minutes, which means that it is likely that the full activity spectrum of most steering muscles was captured during an experiment. The activity range could subsequently be used to compute a normalized activity function for each steering muscle making it possible to compare muscle activity across experiments.

To rescale muscle activity traces, the mean, μ, and standard deviation, σ, were computed for the periods when the fly was flying. A boolean value named `fly_flying` with a timestamp was saved to an hdf5-file during the experiment, and could subsequently be used to determine the flight bouts. Including data points when the fly was not flying is problematic, as several tonic muscles exhibit constant activity during flight. For all steering muscles, except the b_2 and iii_1 muscles, the activity is rescaled such that a value of 0 corresponds to $(\mu - 2\sigma)$ and 1 to $(\mu + 2\sigma)$. In case of the b_2 and iii_1 muscles, the muscles are mostly quiescent during flight, to such an extent that in some experiments these muscles did not fire at all. To control for complete quiescence, the values of b_2 and iii_1 muscles are not rescaled if the standard deviation was below 0.01. If the standard deviation was above this threshold, the b_2 and iii_1 traces will be rescaled such that a value of 0 corresponds to (μ) and a value of 1 to $(\mu + 3\sigma)$.

The parsing of the wing kinematic traces into wingbeats and the subsequent fitting of Legendre polynomials constitutes most of the work for rescaling wing motion. Although the wing kinematics have been tracked for both the left and right wing, steering muscle activity was only recorded from the left side of the thorax. For each wingbeat, the left wing motion was described by a vector of 80 Legendre coefficients. The values of the Legendre coefficients correspond to radians, which can range between $[-\pi, \pi]$. Therefore, the scaling of the Legendre coefficients is achieved by simply dividing by π.

The resampled muscle activity is coupled to the Legendre coefficients and the wingbeat frequency. This coupled data forms the basis of the dataset used to train and validate the CNN. The input data of the network consists of 13 variables: the activity values of the 12 steering muscles and the wingbeat frequency. Wingbeat frequency was scaled subtracting 150 Hz from the frequency value and subsequently dividing by 100 Hz, such that a value of 0 corresponds to 150 Hz and a value of 1 to 250 Hz. The output data of the network are the 80 normalized Legendre coefficients.

Although neural networks are capable of handling outliers in the dataset, too many outlying data points can result in decreased performance. In my analysis, outliers were removed by thresholds on muscle activity, wingbeat frequency, and Legendre coefficients. A wingbeat is removed from the dataset when the normalized muscle activity was lower than −0.5 or larger than 1.5, the normalized wingbeat frequency was lower than 0 or larger than 1 and if any of the normalized Legendre coefficients were not inside ranges: $-1/3 <= C_\theta <= 1/3$, $-2/3 <= C_\eta <= 2/3$, $-2/3 <= C_\phi <= 2/3$, and $-1/3 <= C_\xi <= 1/3$. After removing all outliers, the dataset consisted of 72,219 wingbeats.

It is common practice in machine learning to split a dataset into a training and validation set (Figure 4.1). For the validation set, I selected the first 30 wingbeats of each high-speed video sequence (10868 wingbeats). The remaining wingbeats in the high-speed video were used for the training set (61351 wingbeats).

4.2 Architecture, training and validation of the CNN regression

The GCaMP fluorescence kernel spreads the fluorescence signal of a muscle twitch out over time. Multiple successive muscle twitches lead to temporal summation of the fluorescence signal and this effect makes it difficult, not impossible, to identify the timing of individual muscle twitches. Besides the low temporal resolution of GCaMP signals, the small changes in muscle fluorescence yield a relatively low

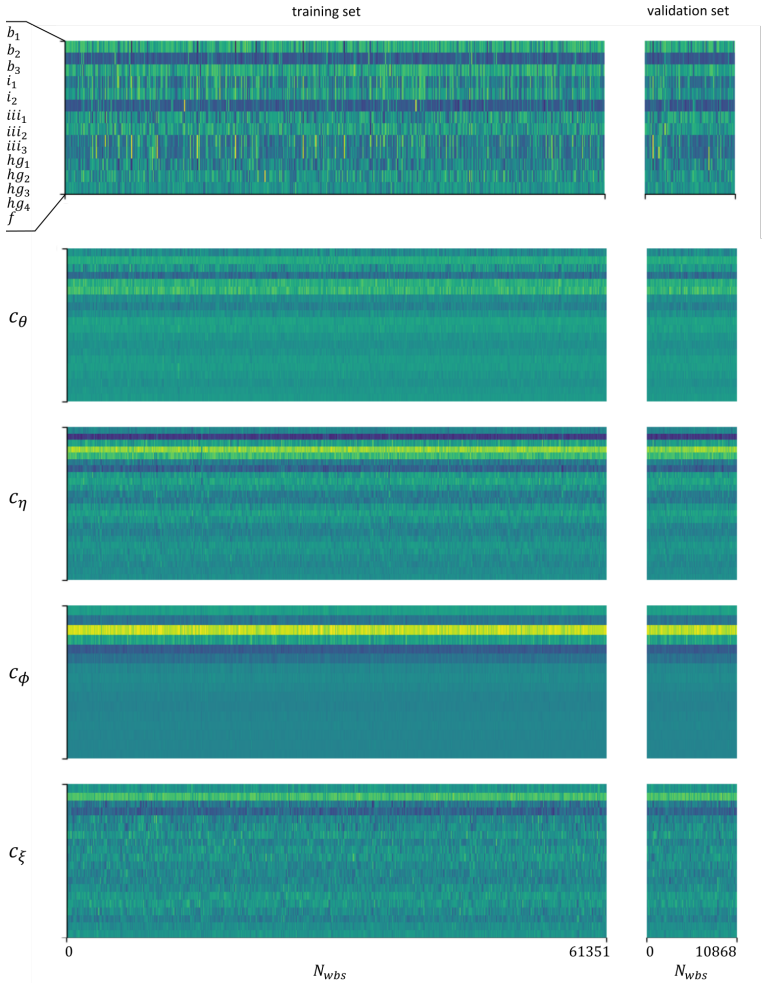

Figure 4.1: Overview of the training (N_{wbs} = 61351) and validation set (N_{wbs} = 10868). The input consists of 13 variables: normalized muscle activity and normalized wingbeat frequency. Output consists of 80 Legendre coefficients: c_θ, c_η, c_ϕ and c_ξ.

signal-to-noise ratio.

A CNN is well suited to extract relevant information from a complex and noisy signal. In the first layer of the CNN, the network learns the GCaMP fluorescence kernel via a number of convolutional kernels over a fixed time window. By specifying a large time window of muscle activity, the CNN has more information to separate signal from noise. A large time window also makes it easier for the network to remember specific patterns in muscle activity that are unique to a recording. Learning unique muscle activity patterns is undesirable, as it is unlikely to occur in unseen data. By specifying a large time window, the CNN is more likely to overfit. If the time window is too small however, the amount of information might not be sufficient to

identify a muscle twitch.

The length of the time window which was found to work well consisted of 9 wingbeats or approximately 45 ms. A shorter time window would not contain sufficient information and prediction accuracy would be worse as a consequence. Longer time windows improved the training error, but the validation error tended to be higher, especially for very long windows (> 50 wingbeats).

The CNN architecture I constructed consisted of 4 layers: 2 convolutional layers and 2 dense layers. Input of the network consisted of a 13×9 matrix of normalized muscle activity and normalized frequency of 9 subsequent wingbeats. As output of the network is a prediction of the 80 normalized Legendre coefficients corresponding to the first wingbeat in the 9 wingbeat time window. Before the input data is fed into the first convolutional layer, Gaussian noise with a standard deviation of 0.05 was added to the input matrix. Gaussian noise helps the CNN to generalize on the data and makes the network more robust to noise in the fluorescence recordings. The first convolutional layer of the CNN consisted of 64 kernels with a 1×9 kernel window, a 1×9 stride ans SELU activation (Figure 4.2). A total of 256 kernels was used for the second layer, with a 13×1 kernel window and stride, and SELU activation. The output of the second layer Was a vector with 256 elements. A fully connected dense layer of 1024 virtual neurons takes in the output of the second layer and applies a SELU activation for each virtual neuron. The last layer of the CNN has 80 neurons with a linear activation function, corresponding to the 80 normalized Legendre coefficients.

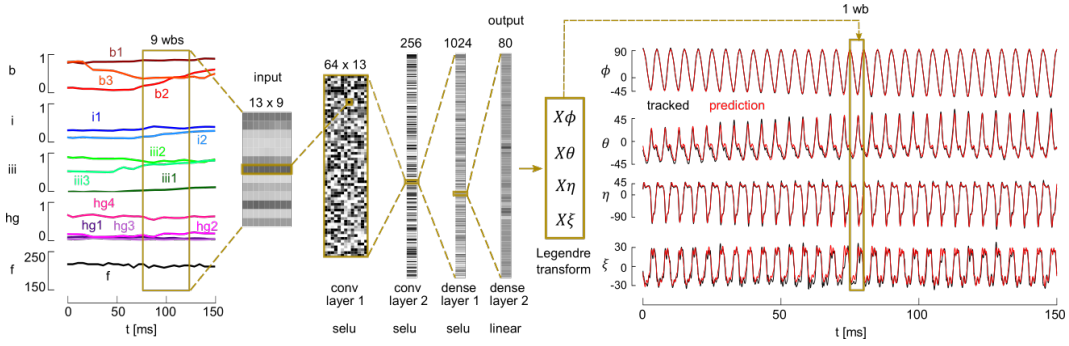

Figure 4.2: Accurate prediction of wing motion by a CNN using muscle fluorescence and wingbeat frequency over a time window of 9 wingbeats.

The CNN was trained on the training set for a total of 1000 epochs and a batch size of 100 samples. After every epoch, the network was evaluated using the validation set. The learning rate was set as 10^{-4} with a decay of 10^{-7} per epoch and MSE

as a loss function. After the first 200 epochs, the validation error stabilized around $10^{-3.7}$ and while the training error continues to decline ($10^{-3.33}$ after 1000 epochs) (Figure 4.3).

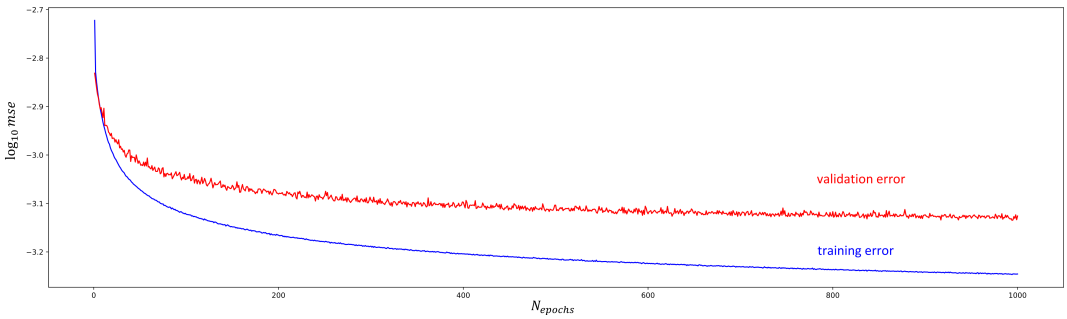

Figure 4.3: Training and validation error of muscle-activity-to-wing-motion CNN.

After training the network, the prediction performance was tested on all videos in the dataset. Examples of the CNN prediction performance are given in (Figure 4.4, 4.5) and Appendix A. For most sequences, the wing motion predicted from muscle activity was within ±2° of the tracked wing motion. This accuracy was quite remarkable, given the complex waveforms of the wing kinematic angles and the sparse nature of the input data. An explanation for the achieved prediction accuracy might be that the orthogonal Legendre decomposition encodes much of the complexity of the wing motion.

Most video sequences shows that muscle fluorescence changes typically last for approximately 50 wingbeats, before returning to baseline. In free flight, one would expect shorter bouts of similar muscle activity, as most maneuvers last between 10 and 30 wingbeats, (Muijres, Elzinga, Melis, et al., 2014, Muijres, Elzinga, Iwasaki, et al., 2015). Although the GCaMP fluorescence kernel can partly explain the slow return to baseline fluorescence, the changes in wing motion typically last around 50 wingbeats as well. This helps to explain why the trained CNN is so accurate in predicting wing motion; the sustained changes in wing motion make the limited temporal resolution of GCaMP imaging less important.

4.3 Virtual experiments on the muscle-to-wing manifold

The trained CNN can be interpreted as an *in-silico* model of the wing hinge. Inputs of this *in-silico* wing hinge are the fluorescence values of the 12 steering muscles and the wingbeat frequency, akin to the input of the steering and power muscles. Output of the *in-silico* wing hinge is the full 3D motion pattern and shape of the

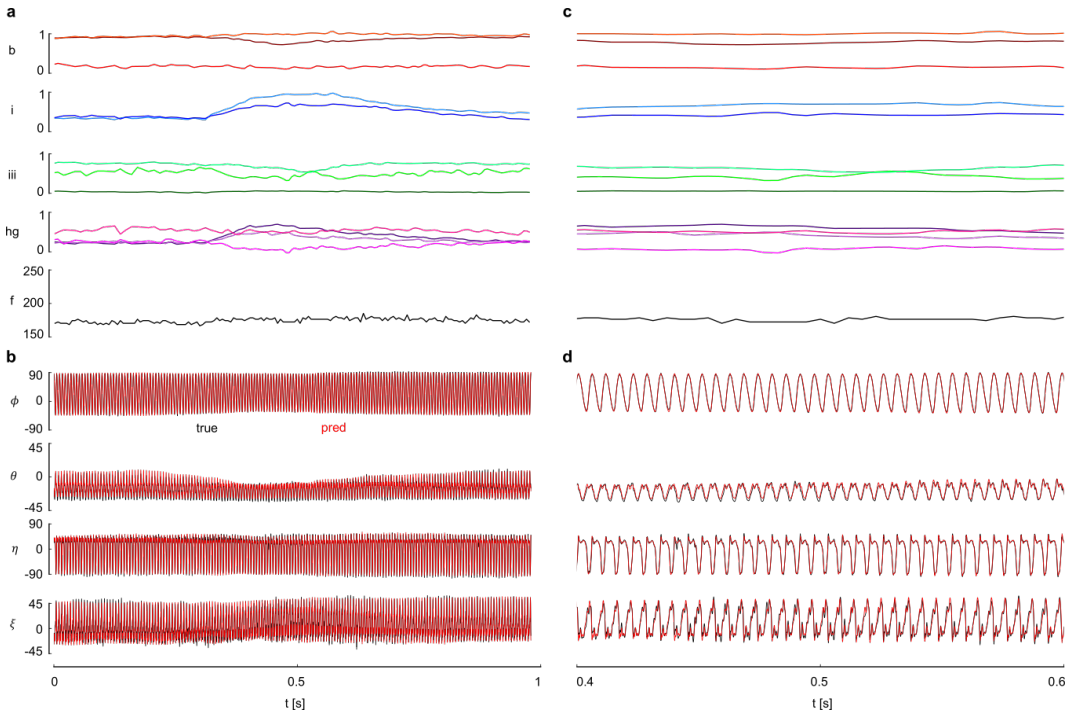

Figure 4.4: Muscle activity and (predicted) wing motion during a high-speed video sequence. A: Muscle activity and wingbeat frequency. B: Tracked (black, true) and predicted (red, pred) wing motion. C: Close-up between 0.4 and 0.6 seconds of A. D: Close-up of B.

wing. The remarkable accuracy of the predicted wing motion using seen and unseen muscle activities inspires confidence in the CNN as a model of the wing hinge. One observation that can be concluded from the accuracy of the trained CNN, is that steering muscle activity and wingbeat frequency provide sufficient information to predict wing motion.

Besides predicting wing motion for observed muscle activity, the *in-silico* wing hinge can also be used to conduct virtual experiments. A naive way of using the CNN predictions would be to vary activity of the b_1 muscle while keep all other input variables constant. However, from the video sequences, I observed that there are no significant independent changes in muscle activity. Instead, the activity patterns within the set of 12 muscles are highly conserved. When changing the muscle activity of only one muscle at a time, the muscle activity pattern is outside the data-subspace on which the CNN was trained. Giving an input that deviates significantly from the patterns that were used to train the CNN requires the network to extrapolate. Extrapolation is unreliable, especially in case of a non-linear regression.

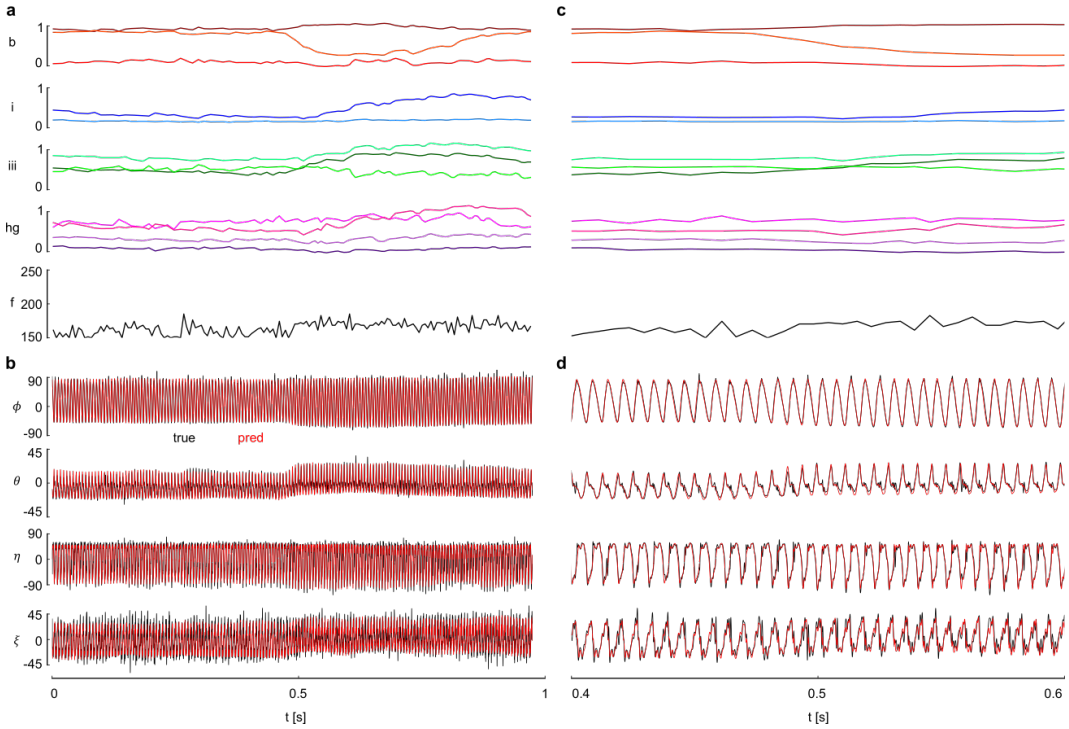

Figure 4.5: Muscle activity and (predicted) wing motion during a high-speed video sequence. A: Muscle activity and wingbeat frequency. B: Tracked (black, true) and predicted (red, pred) wing motion. C: Close-up between 0.4 and 0.6 seconds of A. D: Close-up of B.

In order to get reliable wing motion predictions from the trained CNN, the input data needs to reside within the data-subspace of the training set. To find this subspace, linear models have been fitted to the muscle activity (Figure 4.6). Because the GCaMP fluorescence kernel consists of a sharp rise followed by a more gradual exponential decay, the domain of the linear model consists of all wingbeats in which a selected muscle had activity with a positive fluorescence gradient (> 0.005) during the 9 wingbeat time window. The gradient condition helps to select wingbeats where the muscle of interest was likely to be active. Without this condition, it is impossible to find correlation trends between muscles, as the decay phase of the GCaMP kernel contains more uncorrelated data points and obscures most of the trends. In Table 4.1 the slopes found by a linear model fit for each muscle are presented together with the number of wingbeats that were used for the fit.

The correlation analysis of muscle activity shows that for all steering muscles there is little to no correlation with wingbeat frequency. For the remainder of this thesis, it is therefore assumed that frequency is independent of steering muscle activity. The

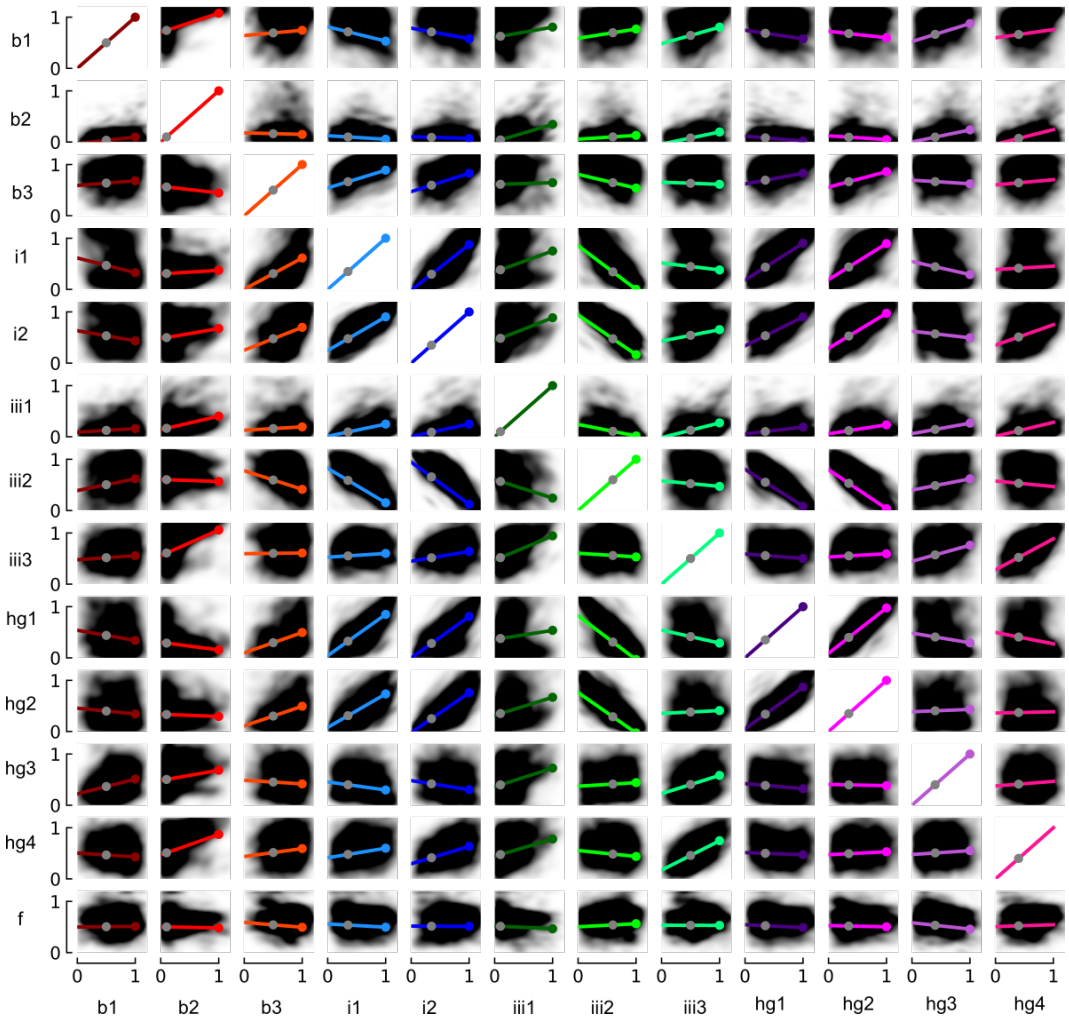

Figure 4.6: Correlation analysis of muscle activity and wingbeat frequency. For each column, a linear model has been fitted to all wingbeats in the dataset for which the muscle fluorescence gradient exceeds a threshold of 0.005. The linear model are plotted over the dataset (black dots) by lines, the baseline muscle activity by a grey dot and the maximum muscle activity by a colored dot.

independence of steering muscle activity from wingbeat frequency is an important observation about the mechanics of the wing hinge and the neural circuitry of the flight system. It suggests that the steering muscles do not need to change their activity pattern as a function of wingbeat frequency, which implies that the configuration changes of the wing hinge by the steering muscles are laregely independent of power muscle activity. Independence from wingbeat frequency also simplifies the neural circuitry required to tune the motor neurons of the steering muscles.

Using the muscle correlation coefficients, it is possible to specify muscle activity

	b_1	b_2	b_3	i_1	i_2	iii_1	iii_2	iii_3	hg_1	hg_2	hg_3	hg_4
b_1	1	0.38	0.09	−0.27	−0.15	0.21	0.18	0.32	−0.16	−0.09	0.38	0.16
b_2	0.12	1	−0.05	−0.07	−0.01	0.34	0.08	0.20	−0.09	−0.07	0.24	0.25
b_3	0.09	−0.14	1	0.36	0.36	0.04	−0.27	−0.03	0.22	0.33	−0.07	0.10
i_1	−0.30	0.06	0.60	1	0.86	0.41	−0.86	−0.17	0.73	0.73	−0.29	0.05
i_2	−0.20	0.19	0.43	0.65	1	0.45	−0.78	0.21	0.56	0.72	−0.16	0.40
iii_1	0.06	0.26	0.04	0.23	0.23	1	−0.23	0.23	0.12	0.17	0.18	0.25
iii_2	0.24	−0.03	−0.34	−0.65	−0.79	−0.35	1	−0.08	−0.71	−0.78	0.26	−0.09
iii_3	0.08	0.50	−0.01	0.07	0.23	0.46	−0.07	1	−0.09	0.09	0.29	0.60
hg_1	−0.20	−0.16	0.39	0.77	0.73	0.16	−0.86	−0.29	1	0.90	−0.24	−0.25
hg_2	−0.11	−0.05	0.37	0.62	0.71	0.34	−0.80	0.02	0.79	1	0.01	0.01
hg_3	0.30	0.20	−0.08	−0.15	−0.14	0.35	0.07	0.32	−0.09	−0.02	1	0.06
hg_4	−0.08	0.40	0.14	0.17	0.36	0.33	−0.11	0.58	−0.06	0.08	0.07	1
f	0.02	0.00	−0.09	−0.06	0.00	−0.02	0.04	−0.01	−0.04	−0.02	−0.08	0.04
N	7606	4325	8884	7252	6949	599	10772	8727	7457	7374	9154	13669

Table 4.1: Linear models are fitted to a subset (N) of muscle activity based on a gradient threshold (> 0.005). One linear model is fitted per column to the muscle specified on top and the indices correspond to the slopes found by the linear model.

inputs that are within the data subset that was used to train the CNN. Besides the slopes of the muscle correlations, a baseline muscle activity pattern is required as a bias vector. Baseline muscle activity is determined by looking for videos in the dataset that contain no significant changes in muscle activity or wing motion, and subsequently determining the average activity over those sequences. The baseline muscle activity is given in Table 4.2.

b_1	b_2	b_3	i_1	i_2	iii_1	iii_2	iii_3	hg_1	hg_2	hg_3	hg_4	f
0.5	0.1	0.5	0.35	0.35	0.1	0.6	0.5	0.35	0.35	0.4	0.4	0.5

Table 4.2: Baseline muscle activity

With the baseline muscle activity, it is possible to predict a baseline wingbeat using the trained CNN. For each muscle and the wingbeat frequency, the baseline muscle activity value is kept constant over the 9 wingbeat time window. Feeding this 13×9 matrix into the trained CNN yields the baseline wingbeat (Figure 4.7).

In a similar way, the effect of maximum activation of a certain muscle can be studied. The slopes found by the linear models of (Figure 4.6) and Table 4.1 form a 12D surface. This surface can be anchored to the baseline muscle activity pattern (Table 4.2). As an example, one can start from the baseline muscle activity and move over the 12D surface towards a point with maximum b_1 activity ($b_1 = 1$). The muscle

activity pattern at $b_1 = 1$ on the 12D surface forms the maximum muscle activity pattern for b_1. In Figure 4.7 the baseline and maximum muscle activity patterns for all steering muscles are shown as well as the CNN predictions of the associated wing motion.

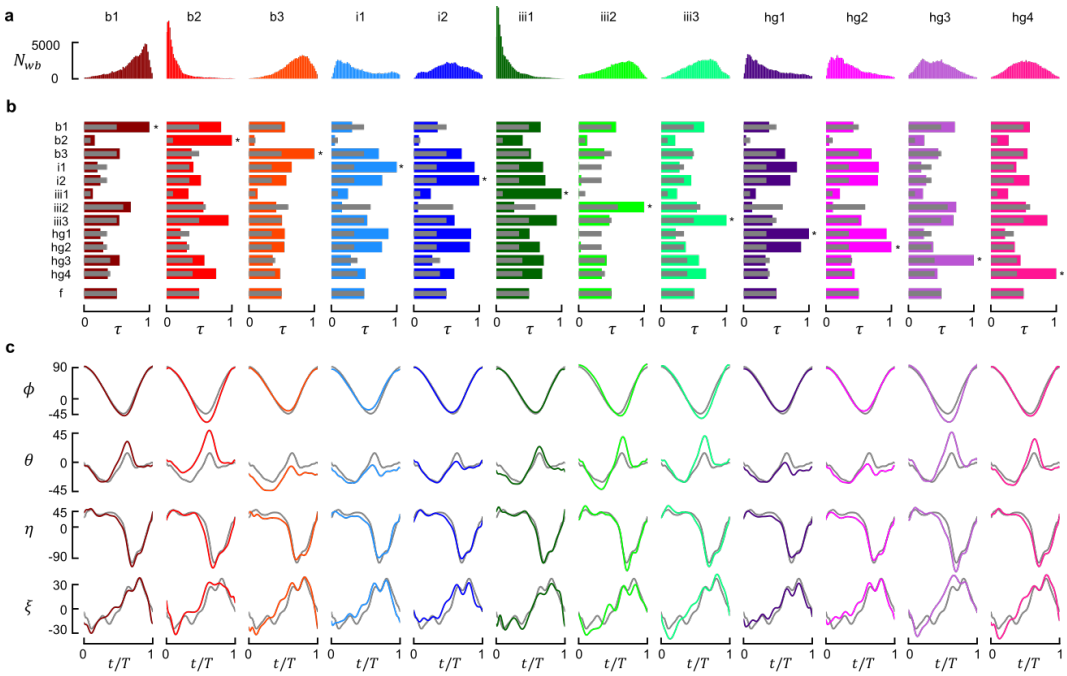

Figure 4.7: Wing kinematics for maximum muscle activity patterns. A: Distribution of muscle activity (50 bins) in the dataset (N_{wb} is number of wingbeats). B: Maximum muscle activity patterns, baseline activity is depicted in gray while the colored bars indicate the maximum muscle activity patterns and maximum muscle activity is marked by *. C: Predicted wing kinematics for the muscle activity patterns in B. over one wingbeat t/T.

4.4 Verification of the predicted muscle function

The found trends in muscle activity patterns and predicted wing motion in the previous section need to be verified by comparison to previous physiological studies. For some of the steering muscles, in particular the hg-muscles, there are no studies of their effects on wing motion. In this section, I will discuss the effects on wing motion by muscle activity and how these effects relate to prior studies.

The distribution of muscle activity in (Figure 4.7A) overlaps mostly with the results found by Lindsay, Sustar, and Dickinson, 2017. In Lindsay, Sustar and Dickinson (2017), the tonic muscles are b_1, b_3, i_2, iii_3, hg_4, and the phasic muscles are b_2, i_1, iii_1, iii_2, hg_1, hg_2, hg_3. The distribution of muscle activity in (Figure 4.7A) shows

that my results largely replicated these findings, with a few subtle differences. The b_1, b_3, iii_2, iii_3, and hg_4 muscles clearly exhibited tonic activity, whereas the b_2, iii_1, hg_1, and hg_2 muscles exhibited phasic patterns of activity. A few muscles (i_1, i_2, and hg_3) showed a mixture of tonic and phasic activity. These discrepancies are not unexpected, given that the amount of data collected by Lindsay and co-workers (2017) far exceed what I was able to collect due the the constraints associated with collecting high-speed video data.

Activity of the b_1 and b_2 muscles results in an increase in stroke amplitude and deviation during ventral stroke reversal (Figure 4.7C). A similar effect on wing motion has been observed in Dickinson and Tu, 1997, Balint and Dickinson, 2001 and Balint and Dickinson, 2004. Besides changes in stroke and deviation angle, the wing pitch and deformation angle advance during ventral stroke reversal. Advanced firing of the b_1 motor neuron results in advanced rotation of the wing, Dickinson and Tu, 1997. Delayed firing of the b_1 motor neuron has an opposite effect on the wing–a decrease in ventral stroke amplitude and deviation, and delayed wing pitch rotation. When both the b_1 and b_2 muscles are active, a strong increase in stroke and deviation angles plus an advance in wing pitch rotation occur. The CNN predictions in (Figure 4.7C) and the measured effects on wing motion in blowflies (Dickinson and Tu, 1997), (Balint and Dickinson, 2001), (Balint and Dickinson, 2004) are remarkably close.

The muscle activity correlations analysis in (Figure 4.6) and Table 4.1 show some surprisingly strong correlations. When inspecting the activity of the b_3, i_1, i_2, hg_1, and hg_2 muscles, one can see strong correlated trends, whereas the iii_2 muscle is anti-correlated with the previously mentioned muscles. An increase in b_3, i_1, i_2, hg_1, and hg_2 activity and a decrease in iii_2 activity results in a decrease in stroke and deviation angle at both ventral and dorsal stroke reversal, in combination with a reduced wing pitch and deformation angle during the downstroke (Figure 4.7C). The reduced wing pitch angle means a higher angle-of-attack during the downstroke. At the start of the upstroke, the wing pitch angle reaches a minimum, sometimes resulting in negative angles-of-attack. In case of increased b_3, i_1, i_2, hg_1, and hg_2 activity with the decreased iii_2 activity, the dip in wing pitch angle is reduced compared to the baseline wingbeat, lowering the wing pitch rotational velocity during ventral stroke reversal.

When iii_2 activity increases and b_3, i_1, i_2, hg_1, and hg_2 activity decreases, the stroke and deviation angle increase during ventral and dorsal stroke reversal, while wing

pitch and deformation angle increase during the downstroke. Angle of attack is increased during the downstroke and the dip in the wing pitch angle at the start of the upstroke is deeper than the baseline wingbeat, resulting in a larger wing pitch rotational velocity during ventral stroke reversal. The deviation angle shows a dip during mid-downstroke, creating a more circular wing trajectory than the U-shaped trajectory of the baseline wingbeat.

Although not all muscles of the b_3, i_1, i_2, hg_1, hg_2 - iii_2 combination have been recorded in Balint and Dickinson, 2004, the observed correlations and effects on wing motion are very similar to the *mode 1* and *mode 2* behavior in (Figure 1.26). Mode 1 in Balint and Dickinson, 2004, corresponds to an increase in i_1 activity and an absence of iii_{24} activity. The iii_4 muscle exists in a blowfly but not in a fruit fly, the iii_2 muscle in *Drosophila* is comparable in orientation and attachment to the *iii*-sclerite, however. Mode 2 shows an absence of i_1 activity and increased iii_{24} activity. The observed stroke and deviation angles in Balint and Dickinson, 2004 for mode 1 and 2, correspond well to the wing kinematics found in (Figure 4.7C). Besides the i_1 and iii_2 muscles, there are four other muscles involved in mode 1 and 2.

Differences in the relative activity of the b_3, i_1, i_2, hg_1, and hg_2 muscles result in relatively subtle changes in wing motion. Strong b_3 activity for example, results in a low deviation angle during the complete wingbeat. When i_2 is more active than the other four muscles, however, the deviation angle is only lower during ventral stroke reversal. The wing motion for maximum i_1, hg_1 or hg_2 activity is quite similar and the effects on deviation angle are between maximum b_3 and i_2 activity.

The iii_1 muscle shows strong fluorescence peaks during flight starts and stops and is likely involved with the unfolding and folding of the wings. During flight itself, the muscle undergoes more gradual changes in activity. The linear fits in Figure 4.6 and Table 4.1 are not necessarily a good representation of iii_1 activity, as some correlations show a bi-modal activity pattern. In Figure 4.7C, the primary effect of iii_1 activity is a lowering of the deviation angle during dorsal stroke reversal. One has to be cautious when evaluating the effect of iii_1 on wing motion, as the number of wingbeats with significant iii_1 activity during flight was small.

The effects on wing motion of the iii_3, hg_3 and hg_4 muscles have not been researched in electrophysiology studies. When looking at the effects of iii_3, hg_3 and hg_4 activity in Figure 4.7C, one can see that the muscles have similar effects on wing motion. For all three muscles, a strong increase in stroke and deviation angle can be observed

during ventral stroke reversal. Wing pitch rotation is advanced with respect to the baseline wingbeat and the wing pitch angle is lower during the upstroke (resulting in a lower angle-of-attack). The effect on ventral stroke and deviation angle are the strongest for hg_3 activity, followed by iii_3 and hg_4 activity. There is a strong similarity in wing motion for maximum b_2 and hg_3 activity. In fact, b_2 activity is correlated with iii_3, hg_3 and hg_4 activity. The b_2 muscle shows highly phasic activity, while the iii_3, hg_3 and hg_4 muscles have more tonic behavior. Similar to the mode 1 and mode 2 behavior for the b_3, i_1, i_2, hg_1, hg_2 - iii_2 combination, one could propose that there is a *mode 3* as well. Mode 3 involves the b_2, iii_3, hg_3 and hg_4 muscles, resulting in strong increases in stroke amplitude, high deviation angles, and advanced wing rotation during ventral stroke reversal. As with mode 1 and mode 2, the relative activity of the involved muscles determines the exact wing motion pattern.

Interestingly, the hg_3 and hg_4 muscles have independent activity (Figure 4.6, Table 4.1). A similar independence of muscle activity with similar effects on wing motion can be observed between the iii_2 and iii_3 muscles. I speculate that having multiple muscles with similar effects on wing motion makes it easier to switch between different motor programs in subsequent wingbeats. It is difficult to grasp the significance of the sometimes subtle differences in wing motion patterns for the maximum muscle activity wingbeats in Figure 4.7C. In the following chapter, the aerodynamic force production of all wing motion in Figure 4.7C will be investigated, shining a light on the more subtle differences in wing kinematics.

In summary, the CNN-predicted effects on wing motion for b_1, b_2, i_1, and iii_2 activity correspond well to the results of previous electrophysiology studies. This overlap provides confidence in the accuracy of the CNN predictions. The correlation analysis of muscle activity shows that changes in wing motion require an increase or decrease in activity of most steering muscles. Actuation of wing motion requires coordinated steering muscle activity, with the activity being approximately confined to a 12D plane. The correlation analysis in Figure 4.6 and the CNN-predicted wing motion in Figure 4.7 will be valuable for investigating the neural wiring of the ventral nerve cord and how the observed motor programs are encoded in the nervous system of the fly.

4.5 Deciphering sclerite functionality with an autoencoder

The strong correlations between steering muscles, described in the previous section, show that analyzing the effects on wing motion of individual muscles using the CNN is not feasible, given the strong intrinsic correlations. Therefore, I will take a different approach, which makes use of the fact that there are four sclerites in the wing hinge with steering muscles attached, and understanding the functional role of each sclerite is a critical step in determining how muscle activity translates into changes in wing motion. The trained CNN of the previous section is not suitable for this investigation, as the input requires correlated muscle activity patterns that involve all four sclerites. In this section, I will train a CNN using an *autoencoder* architecture to learn how the activity of all the muscles attached to a particular sclerite is correlated to changes in wing motion. An autoencoder representing independent sclerite states will provide more information on the function of the wing hinge.

An autoencoder predicts the input of the network, but contains a bottleneck that forces the network to learn how to represent the input data by a small number of parameters: the latent variables. The parameters in the bottleneck span the latent space. Training the autoencoder with a bottleneck is roughly equivalent to performing a (non-linear) PCA on a dataset. In this study, I implement a novel form of a typical autoencoder, so that besides predicting the input, the 80 Legendre coefficients of wing motion are predicted as well.

The architecture of the sclerite function autoencoder is given in Figure 4.8. Input to the autoencoder is split into 5 streams, grouping the muscle activity per sclerite and a separate wingbeat frequency stream. Each stream uses two convolutional layers, similar to Figure 4.2, followed by a dense layer of 512 neurons and a final layer of a single neuron with linear activation. The 5 neurons of the 5 streams form the latent space, projecting the muscle activity to a single variable per sclerite. After the latent space, the network is split into two streams: one stream uses a *decoder* to predict the input to the network, the second stream uses two dense layers to predict the Legendre coefficients of the wing motion. A back-propagation stop is placed between the latent space and the muscle activity decoder layers, such that the latent space is only trained on correlations with wing motion. The muscle activity decoder layers are still functional, as the decoder predicts the correlation between the latent space and muscle activity without the noise of the input to the network.

After training the sclerite function autoencoder (1000 epochs, batch size 100, learn-

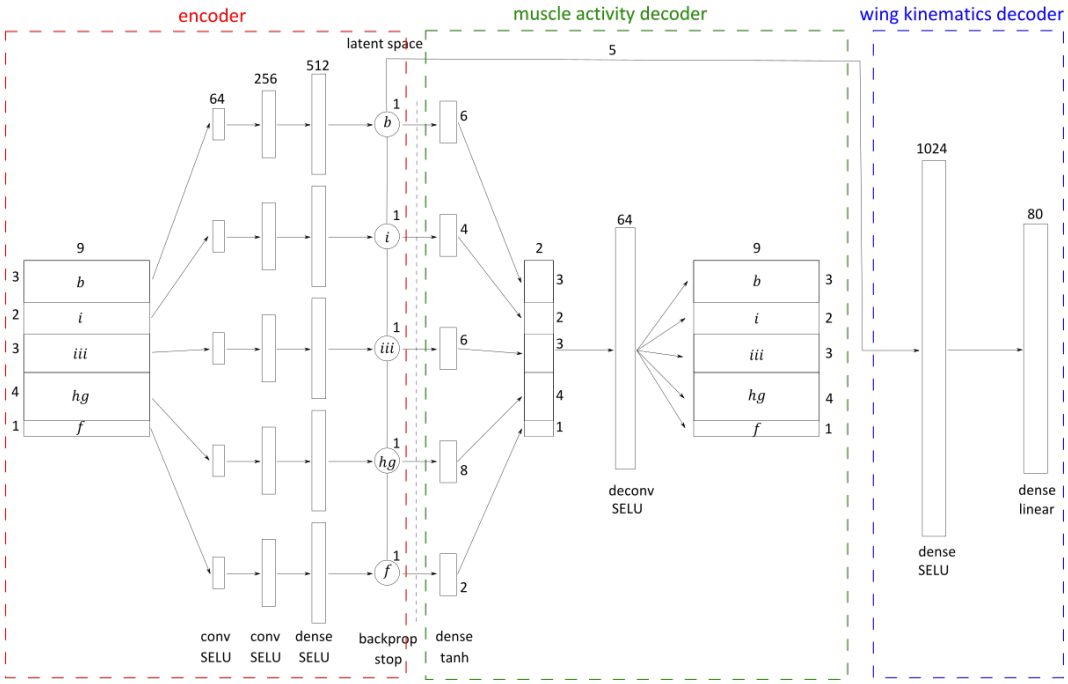

Figure 4.8: Architecture of the sclerite autoencoder. Input is the 13×9 muscle activity matrix while the outputs are the muscle activity matrix and the Legendre coefficients of wing motion. The latent space consists of 5 parameters representing the b, i, iii, hg sclerites and the wingbeat frequency, f. Convolutional (conv), deconvolutional (deconv), and fully connected (dense) layers have been used with linear, SELU and tanh activation functions. A back-propagation stop has been placed between the latent space and the muscle activity decoder section of the network.

ing rate $10^{-}4$, decay $10^{-}7$) on the muscle activity and wing motion dataset, the network predicts three different vectors/matrices: the latent space vector (5×1), the muscle activity matrix (13×9), and the Legendre coefficients (80×1). It is important to note that the prediction error of the wing kinematics by the sclerite function autoencoder is worse than the network in Figure 4.2, as the bottleneck restricts the amount of information that can be used for the prediction.

In Figure 4.9, only the decoder sections of the network are used to predict muscle activity and wing motion for a given latent vector. Each latent parameter is varied between -3σ and $+3\sigma$ (9 steps) while the other latent variables are kept at 0, and subsequently the muscle activity and wing motion are predicted from the latent input. The autoencoder learns a single variable for each sclerite and subsequently shows which aspects of muscle activity and wing motion are correlated with the latent parameter.

The first latent parameter describes the state of the basalare sclerite, the b-parameter.

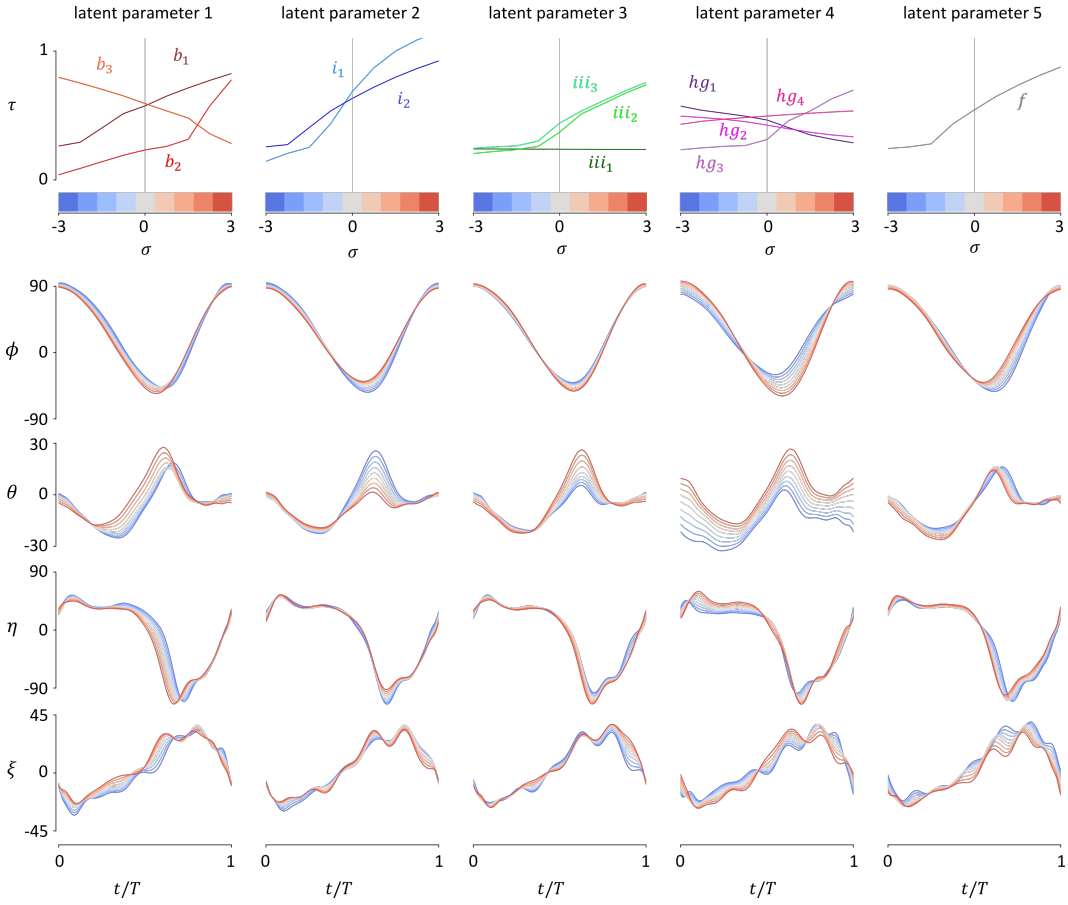

Figure 4.9: Predicted muscle activity (τ) and wing motion (ϕ, θ, η, ξ) for 5 latent parameters ($[-3\sigma, 3\sigma]$, blue-red colorbar).

A positive value for the b-parameter corresponds to an increase in b_1 and b_2 activity while the b_3 activity decreases. Negative values for the basalare latent parameter show an opposite trend with increasing b_3 activity and decreasing b_1 and b_2 activity. These relationships make sense, because anatomical and physiological evidence indicates that the b_1 and b_2 muscles both act to rotate the basalare apophysis anteriorly, whereas the b_3 muscles acts to rotate the basalare apophysis posteriorly. The corresponding wing motion shows a relatively small change in stroke angle, but a clear phase shift in ventral wing rotation, with advanced rotation for a positive latent value and delayed rotation for a negative latent value. During the downstroke, the deviation angle increases with an increase in the basalare latent parameter, followed by a rapid drop at the start of the upstroke. A decreasing value of the basalare latent parameter has the opposite effect on the deviation angle.

The second latent parameter corresponds to the state of the first axillary sclerite: the

i-parameter. For a positive value of the i-parameter, the i_1 and i_2 muscles increase activity while a negative value decreases muscle activity. The corresponding wing motion for a positive i-parameter shows a decrease in stroke amplitude and a decrease in the ventral deviation angle. During the upstroke, the wing pitch angle is higher, corresponding to a higher angle-of-attack. A negative value of the i parameter shows opposite trends in wing motion.

The third latent parameter describes the state of the third axillary sclerite or the iii-parameter. A positive value of the iii-parameter corresponds to increasing activity of the iii_2 and iii_3 muscles. For negative values of the iii-parameter, the iii muscles converge to a constant low activity level. The iii_1 muscle is not correlated to the iii-parameter, probably because iii_1 is rarely active during flight, and is primarily specialized for folding and unfolding the wing. Changes in wing motion are almost the exact opposite from the i-parameter: a positive iii-parameter shows an increase in stroke amplitude and ventral deviation angle, and a lower wing pitch angle (lower angle-of-attack) during the upstroke. The mirroring of the i-parameter and the iii-parameter is likely the result of the strong anti-correlation between i_1, i_2, and iii_2 muscle activity.

The fourth latent parameter gives the state of the fourth axillary sclerite or the hg-parameter. With an increasing hg-parameter, the hg_1 and hg_2 activity decreases, while the hg_3 activity rises strongly and the hg_4 activity increases more gradually. The similar trends of the hg_1 and hg_2 muscles correspond to the strong correlation between the two muscles, while the different slopes of the hg_3 and hg_4 muscles hint at their independent activity. A strong change in wing stroke amplitude and deviation angle throughout the wingbeat can be observed as a function of the hg-parameter. The wing pitch angle during the downstroke increases with the hg-parameter, resulting in a lower angle-of-attack. Similarly, the deformation angle changes throughout the wingbeat, creating a higher wing camber with an increasing hg-parameter.

Finally, the fifth latent parameter corresponds to the effect of wingbeat frequency on wing motion, the f-parameter. The trend between the f-parameter and the wingbeat frequency is roughly linear. With increasing wingbeat frequency, the stroke amplitude decreases and the downstroke-to-upstroke ratio decreases. The deviation angle and wing pitch angle show a similar shift in the downstroke-to-upstroke ratio. During the upstroke, the deformation angle decreases with increasing wingbeat frequency, resulting in lower wing camber. The decrease in stroke amplitude

with increasing wingbeat frequencies has been observed in Lehmann and Dickinson, 1997, where for higher wingbeat frequencies the stroke amplitude needs to be decreased as the maximum power output of the power muscles has been reached.

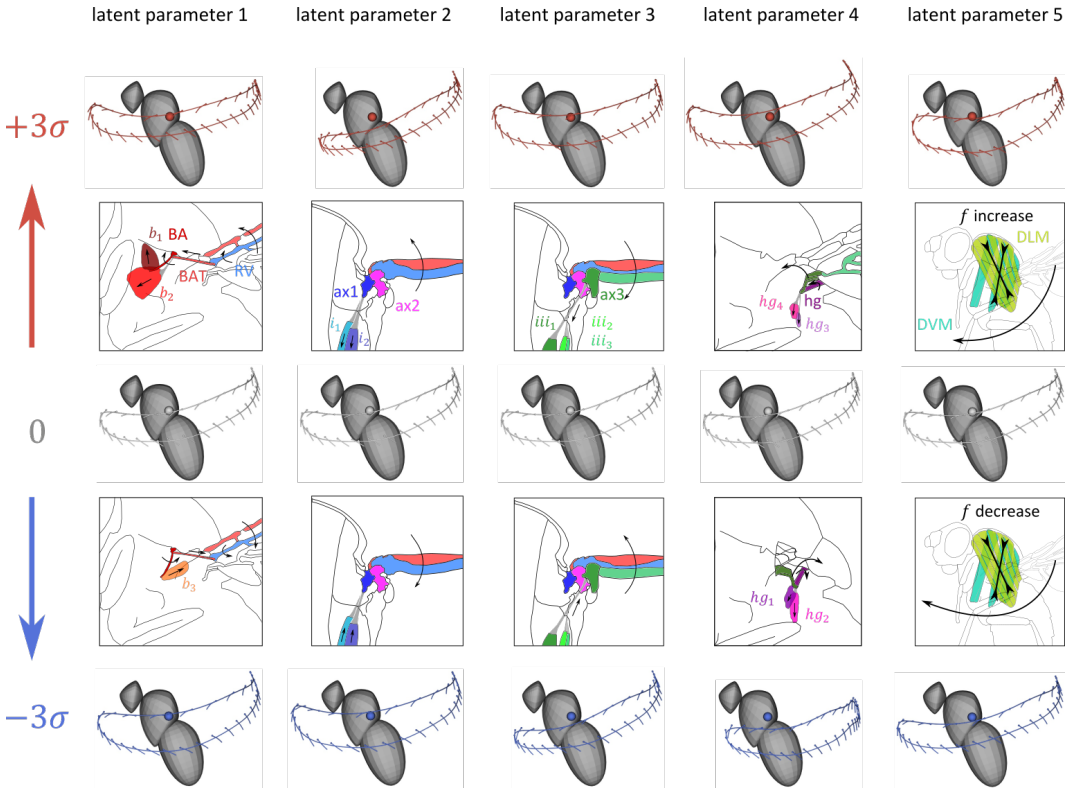

Figure 4.10: Hypothesized mechanisms of actuation by the steering muscles. The wing kinematics for -3σ, 0, and $+3\sigma$ for each latent parameter are displayed as lollipop figures: showing the wingtip trajectory and the wing pitch and deformation angles by portraying the cross-section of the wing at regular intervals. Abbreviations: basalare (BA), basalare tendon (BAT), radial vein (RV), first axillary sclerite (ax1), second axillary sclerite (ax2), third axillary sclerite (ax3), fourth axillary sclerite (hg), dorso-ventral muscles (DVM), and dorso-longitudinal muscles (DLM).

With the predicted wing motion of the sclerite function autoencoder, one can speculate about how the steering muscles work on the wing hinge. In Figure 4.10, I present my hypotheses of how steering muscles might act on the sclerites in the wing hinge to create the changes in wing motion. Although more research in the exact anatomy and mechanics of the sclerites in the wing hinge is required to verify the hypothesized actuation, the anatomy presented in Figure 4.10 follows the findings of (Miyan and Ewing, 1985) closely.

As discussed briefly above, the b-parameter encodes the known anatomy of the basalare sclerite and the attachments of the basalare muscles accurately, in particular,

the antagonistic relation among the b_1, b_2, and b_3 muscles. Activation of the b_1 and b_2 muscles (b-parameter = $+3\sigma$) rotates the apophysis of the basalare sclerite more anteriorly, thereby increasing the tension on the basalare tendon. The basalare tendon is attached to the radial vein and the increased tension on the tendon pulls the wing upwards during ventral stroke reversal. Additionally, the basalare tendon causes the radial vein to rotate earlier during ventral stroke reversal, causing a phase advance in wing rotation. When the b_3 muscle is activated (b-parameter = -3σ), the apophysis of the basalare is rotated posteriorly, decreasing the tension on the basalare tendon. The decreased tension on the tendon reduces the stroke and deviation angle during ventral stroke reversal, and delays wing rotation.

When the i_1 and i_2 muscles are activated (i-parameter = $+3\sigma$), both muscles exert a ventral pull on the first axillary sclerite, which is connected to the second axillary sclerite. The second axillary sclerite serves as a fulcrum for the back and forth motion of the wing, and the inboard location of the first axillary sclerite means that ventral stroke and deviation extend are dampened by the i_1 and i_2 activity. When the i_1 and i_2 muscles are relaxed (i-parameter = -3σ), the dampening effect on wing motion is not present and ventral stroke and deviation angle reach their full extent.

The activity of the iii_2 and iii_3 muscles (iii-parameter = $+3\sigma$), results in a ventral pull on the third axillary sclerite, which is coupled to the second axillary sclerite. As the third axillary sclerite is positioned outboard with respect to the second axillary sclerite, the ventral pull results in an increase in stroke and deviation angles during ventral stroke reversal. Relaxation of the iii muscles (iii-parameter = -3σ) has the opposite effect on wing motion and lowers the stroke and deviation angles during ventral stroke reversal.

Activation of the hg_3 and hg_4 muscles (hg-parameter = $+3\sigma$), changes the orientation of the helical tubes of the fourth axillary sclerite, such that the rotation amplitude of the sclerite increases. Rotation of the fourth axillary sclerite is transferred to the wing via the third axillary sclerite, and results in a larger wing stroke amplitude and deviation angle throughout the wingbeat. Because the third axillary sclerite is coupled to the more posterior veins of the wing, hg_3 and hg_4 activity also affects the wing pitch and deformation angles throughout the wingbeat (Figure 4.9). When the hg_1 and hg_2 muscles are active (hg-parameter = -3σ), the muscles affect the orientation of the helical tubes of the fourth axillary sclerite, such that the rotation amplitude is decreased. This decrease in fourth axillary rotation results in a reduction in stroke amplitude and deviation angle throughout the wingbeat.

Similarly, the absolute wing pitch and deformation angles are lower throughout the wingbeat.

When the wingbeat frequency is high (f-parameter = $+3\sigma$), the motor neurons of the power muscles fire more rapidly, but still an order of magnitude lower than the wingbeat frequency. The increased firing rate of the motor neurons, pumps more Ca^{2+} ions into the power muscles accelerating the asynchronous contraction cycle of the DLMs and DVMs. When the wingbeat frequency is high, the wing stroke amplitude decreases, because the power muscles can only provide a limited amount of mechanical power (Lehmann and Dickinson, 1997, Namiki et al., 2022). For low wingbeat frequencies (f-parameter = -3σ), the stroke amplitude can be higher as the wing kinematics are no longer limited by the power output of the power muscles.

Chapter 5

DYNAMICALLY-SCALED FLAPPING WING EXPERIMENTS

The detailed analysis of the relation between steering muscle activity and wing motion in the previous chapter does not provide any information on the aerodynamic forces that are created by changes in wing motion. Computing the aerodynamic forces via the quasi-steady aerodynamic model can be an easy way to investigate the aerodynamic effects of steering muscle activity. However, the quasi-steady approach does not include a model for wing deformation and might therefore be inaccurate. Although CFD simulations have become faster and easier to implement, it would still take hours to compute the aerodynamic forces for just one wing kinematic pattern. The fastest way to accurately measure aerodynamic forces of insect flight remains through the use of a dynamically-scaled flapping wing robot.

In the last three decades, there have been several versions of a dynamically-scaled flapping wing robot in the Dickinson lab, all nicknamed *RoboFly*, (Dickinson, 1994), (Lauder, 2001), (Birch and Michael H Dickinson, 2001), (Birch, William B Dickson, and Michael H Dickinson, 2004), (Poelma, W. Dickson, and Dickinson, 2006), (William B Dickson et al., 2010), (M. J. Elzinga, Van Breugel, and Michael H Dickinson, 2014).

In this chapter, I will give a short description of the hardware and software of the latest version of RoboFly that I developed. Subsequently, I will describe the formulas used to dynamically scale the experiments. To permit me to make force and moment measurements on a model wing that could deform as captured in my high-speed video analysis, I designed and built a wing that consists of four panels connected by three hinge lines, actuated by three micro servos. I will then present the aerodynamic results using the wing kinematics for maximum muscle activity patterns in Figure 4.7.

5.1 Design and control of RoboFly

The two wings of RoboFly were actuated by a stepper motor and two servo motors each (Figure 5.1). Stroke angle was controlled by a stepper motor (10 kHz clock) via two gears with a 1:3 gear ratio. The position of the stepper motor was controlled via micro-stepping, which permitted fine motor control ($7.8 \cdot 10^{-3}$ degrees per

microstep). Two magnets were positioned on the gear ($\phi = [-91°, 91°]$), such that a Hall-effect sensor was activated when the wing moves out of bounds. Besides protecting the wing, the Hall-effect sensor is also used to home the stepper motor. During the homing procedure, the stepper rotates the wing in one direction until the Hall-effect sensor is tripped. The wing rotation stops immediately and subsequently moves the wing back to the home position ($\phi = 0$). During an experiment, the Teensy 3.2 microcontroller kept track of the number of microsteps travelled, relative to the home position. Sometimes the stepper slipped due to large torques. To counter motor-slip, the steppers were homed after each experiment.

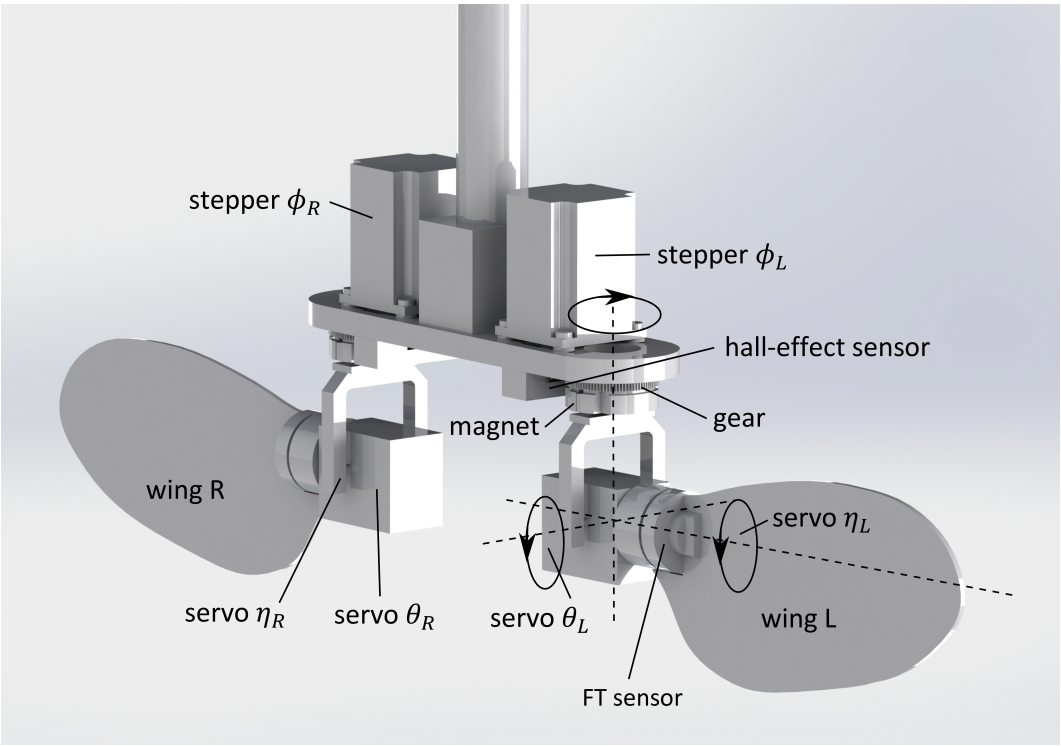

Figure 5.1: The two RoboFly wings are actuated by two stepper motors (ϕ_L, ϕ_R) and four servo motors (θ_L, η_L, θ_R, η_R). Both stepper motors actuate the wings via gears (1 : 3 gear ratio). Coupled to the gears are two magnets, that trip a Hall-effect sensor when the wing goes out of bound. At the base of the left wing is a FT sensor.

During an experiment, the stepper's position, ϕ, and velocity, $\dot{\phi}$, were controlled in a feed-forward process. The stepper made micro-steps on a 10 kHz clock, but the position and velocity set-points were updated at 200 Hz. After the position and velocity update, the control algorithm computed the required stepping frequency and direction to reach the specified position and velocity 5 ms ahead. When external torques on the stepper motor or velocity was too high, the stepper motor often start

slipping. Such slip was not automatically detected, as there was no feedback on the actual position of the stepper. To prevent motor slip, I set bounds on the maximum stepper velocity and acceleration.

The deviation and wing pitch angles were controlled by two servos (HiTec D951TW), the deviation angle servo moved the servo-pod along the deviation axis via two gears (1:1 gear ratio), and the wing was directly attached by the wing pitch servo. Position of the servos was specified by a pulse-width modulation (PWM) signal at 50 Hz. The deviation servo could move between $-45°$ and $+45°$ and the wing pitch servo could move between $-90°$ and $+90°$. During an experiment, the position of the servo was updated at 50 Hz in a feed-forward control loop.

The forces and torques on the wing were measured by a 6 degrees of freedom Force-Torque (FT) sensor (ATI Nano 17), mounted on the rotation axis of the wing pitch servo. Custom machined aluminum mounts coupled the base of the FT-sensor to the servo axis, and the head of the FT-sensor to the wing. During the experiment, the six 16-bit unsigned integers corresponding to the: F_x, F_y, F_z, T_x, T_y, T_z forces and torques were sampled at 200 Hz and directly written to a `.txt` file. As the FT-sensor was positioned at the wing base, vibrations due to stepper and servo motion were picked up by the FT-sensor. In an earlier version of RoboFly, the stepper motor actuated the wing directly, without gearing. This turned out to be problematic, as interference occurred at a stepper motor velocity that was within the required velocity range of some wing kinematic experiments. The wing vibration at the interference velocity was so strong that it would dominate the FT-measurements. By using two gears with a 1:3 gear ratio, the interference velocity was three times as high and the interference phenomenon no longer occurred during experiments.

RoboFly was submerged in an acrylic tank ($2.4 \times 1 \times 1.2\ m$) filled with mineral oil (Chevron SuperLa white oil), with a kinematic viscosity of $115 \cdot 10^{-6}\ m^2 \cdot s^{-1}$ and a density of 880 $kg \cdot m^{-3}$ at 22 $C°$, (M. J. Elzinga, William B Dickson, and Michael H Dickinson, 2012). Mineral oil is an electric isolator, and submerging the servos, stepper motors and FT-sensor was therefore not problematic. The oil level in the robofly tank was approximately 1 m. RoboFly was submerged at a depth of 50 cm and the wing's wingtip radius was 31 cm. A previous study carefully mapped the effects on lift and drag when the RoboFly wing moves close to the; top, side and bottom surfaces of the tank (Dickinson, Lehmann, and Sane, 1999). In this study, it was determined that as long as the wing remains further than $\sim$ 12 cm away from any surface, the surface does not have any noticeable effect on lift or drag generation, i.e.

there are no measurable wall- or surface effects, and the results approximate those of an infinite volume. The positioning of the RoboFly in the acrylic tank guarantees that the wing never gets too close to the walls or the top and bottom surfaces.

5.2 Actuating wing shape through micro servos

Besides the three Tait-Bryan angles describing wing orientation, FlyNet tracks a fourth angle describing wing shape, the deformation angle ξ. To implement this fourth angle on RoboFly, the wing was composed of four panels connected by three hinge lines (Figure 5.2). The four panels were cut out of an acrylic sheet (2.75 mm thickness) using a laser-cutter. Each hinge line as a 2 mm steel rod at the core, surrounded by an acrylic tube with inner and outer diameters of 2 mm and 4 mm, respectively. The acrylic tube was cut into sections of 20 mm and these sections were glued in an alternating pattern to two adjacent panels.

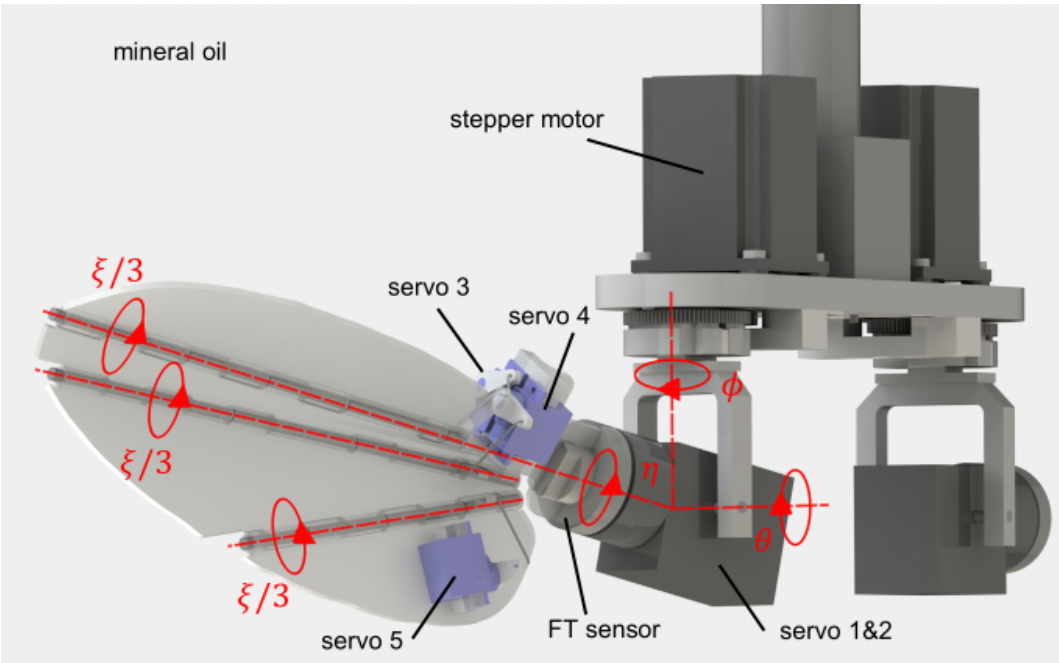

Figure 5.2: Deformation angle is controlled along three hinge lines ($\xi/3$) via three micro servos (servo 3-5).

The rotation angle between two subsequent panels was controlled by a micro servo (HiTec HS-7115TH), which was screwed onto one panel and connected to the next panel via a 1 mm metal rod that was coupled to the servo arm. A total of three micro servos were used to deform the wing. Following the assumption made in FlyNet, i.e. the wing bending angle is uniform over the three hinge lines, the deformation angle ξ was divided by 3 to obtain the rotation angle per hinge line. Because the

assembly of the wing was not perfect, the angle between two subsequent panels does not necessarily correspond to the rotation angle of the servo arm. A separate Teensy 3.2 microcontroller was used to set the servo arm position such that the orientation angle between two panels corresponds to $\xi/3$. The input to the microcontroller was a PWM signal encoding the angle ξ. This input signal is split into 3 PWM signals, where each PWM signal corresponded to the servo arm angle required to achieve a $\xi/3$ rotation angle along the hinge lines. The mapping function between servo arm angle and the rotation angle along the hinge line was measured for each micro servo and these values were used by the Teensy to compute the output PWM signals.

5.3 Dynamic scaling of fly flight

The density and viscosity of the mineral oil allows the RoboFly wing to be much larger than a fly's wing, and flap at a much lower frequency than 200 Hz. Without the benefits of dynamic scaling, it would be impossible to recreate the flow around a fly's wing in an experiment. Dynamic scaling was ensured by matching the Reynolds number of the robotic wing and a real fly, which requires:

$$\frac{n_{fly} \cdot R_{fly}^2}{\nu_{air}} = \frac{n_{robo} \cdot R_{robo}^2}{\nu_{robo}}, \tag{5.1}$$

where n_{fly} is the fly's wingbeat frequency, n_{robo} the wingbeat frequency of RoboFly, R_{fly} the fly's wing length, R_{robo} the wing length of RoboFly, ν_{air} the kinematic viscosity of air and ν_{robo} the kinematic viscosity of the mineral oil. Rewriting the Reynolds number equality yields:

$$n_{robo} = \frac{R_{fly}^2 \nu_{robo}}{R_{robo}^2 \nu_{air}} n_{fly}. \tag{5.2}$$

By entering the values for a typical fly ($R_{fly} = 2.7mm$, $\nu_{air} = 15.7 \cdot 10^{-6} m^2 s^{-1}$ at $25°C$, $n_{fly} = 200Hz$) and the RoboFly parameters ($R_{robo} = 310mm$, $\nu_{robo} = 115 \cdot 10^{-6} m^2 s^{-1}$ at $22°C$), the required flapping frequency for RoboFly is $0.11Hz$. The low flapping frequency required for RoboFly experiments can be realized easily generated using the stepper and servo motors.

Once RoboFly experiments have been performed, the equivalent aerodynamic forces and torques that a fly's wing would have experienced may be calculated. The scaling factor between aerodynamic forces measured on RoboFly and the equivalent forces in fly flight are found using the quasi-steady expressions for the different aerodynamic

mechanisms (equations: 1.14, 1.18, 1.22, 1.26, 1.30). Using dimensional analysis on the quasi-steady aerodynamic mechanisms shows that each expression satisfies the following:

$$F = \rho n^2 R^4 = \left[\frac{kg}{m^3}\frac{1}{s^2}m^4\right] = \left[\frac{kg \cdot m}{s^2}\right] = [N]. \tag{5.3}$$

The force scaling factor can, therefore, be written as:

$$\frac{F_{fly}}{F_{robo}} = \frac{\rho_{air} \cdot n_{fly}^2 \cdot R_{fly}^4}{\rho_{robo} \cdot n_{robo}^2 \cdot R_{robo}^4}, \tag{5.4}$$

and the density values are: $\rho_{air} = 1.18 kgm^{-3}$ at $25°C$ and $\rho_{robo} = 880 kgm^{-3}$ at $22°C$. With the force scaling factor known, it is easy to compute the torque scaling factor, as torque is the product of force and moment arm:

$$\frac{T_{fly}}{T_{robo}} = \frac{\rho_{air} \cdot n_{fly}^2 \cdot R_{fly}^5}{\rho_{robo} \cdot n_{robo}^2 \cdot R_{robo}^5}. \tag{5.5}$$

5.4 Experimental procedure for testing wing kinematics

Any experiment on RoboFly requires a `.txt` file that specifies the following angles: ϕ, $\dot{\phi}$, θ, η and ξ at intervals of 5 ms. A C++ program read the five angles and sent to the Teensy microcontroller via a USB-cable at 200 Hz (RawHID protocol). The Teensy microcontroller converted the angles to PWM commands for the servos and step-direction commands for the stepper motors. At the same time, the FT-measurements were sent from the Teensy micr-controller to the C++ program, which wrote the measurements to a `.mat` file.

The Legendre polynomials and coefficients describing a wingbeat, allow for easy interpolation at the 5 ms intervals required for the `.txt` file. At the start of an experiment, the wing was moved to the home position ($\phi = 0$, $\theta = 0$, $\eta = 0$, $\xi = 0$). In order to prevent rapid acceleration of the wing, the wing kinematic angles during the first wingbeat of an experiment were multiplied with the following function:

$$g_{start}(t) = \sin^2\left(\frac{\pi t}{4T}\right), \tag{5.6}$$

where T corresponds to the wingbeat period. Similarly, the last wingbeat of the experiment ends at the home position and the wing kinematic angles are multiplied by:

$$g_{end}(t) = \sin^2\left(\frac{\pi}{4} + \frac{\pi t}{4T}\right). \tag{5.7}$$

When the wings start moving in the oil, it typically takes two to three wingbeats before the wake is fully developed (Birch and Michael H Dickinson, 2001). In order to exclude these start-up effects, a wing kinematic pattern was repeated for 9 wingbeats and only wingbeats 4-8 were used for analysis (Figure 5.3).

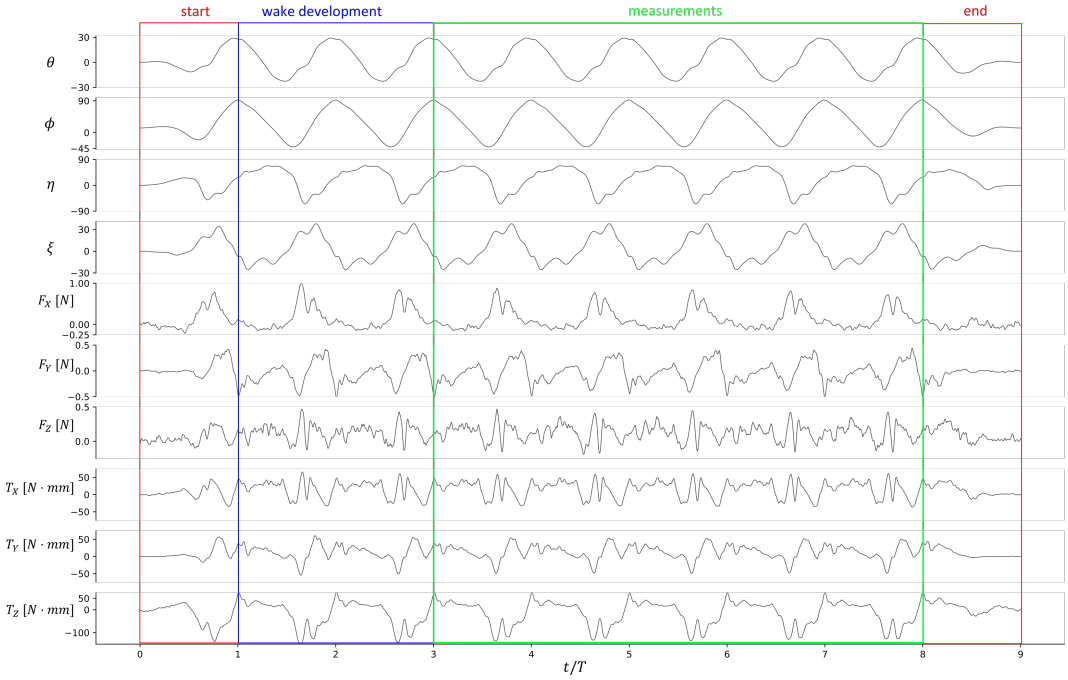

Figure 5.3: Typical RoboFly experiment, with the input wing motion and the measured forces and torques in the strokeplane reference frame. The wing kinematics were repeated for 9 wingbeats, with the first and last wingbeat being multiplied by a $\sin^2$ function, the 2^{nd} and 3^{rd} wingbeat eliminated due to wake development, and the remaining wingbeats for further analysis.

After each experiment, the FT-data needs to be converted from 16-bit unsigned integers into actual force and torque vectors. This was done by multiplying the 6 FT integer values with a calibration matrix, yielding a force vector in Newtons and a torque vector in Newton millimeter in the wing reference frame. The FT-data was also noisy and required smoothing by a linear Kalman filter.

Besides aerodynamic forces, the FT-sensor also measures inertial and gravitational forces. The low flapping frequency of RoboFly allows the inertial forces to be ignored. A fly's wing is a lightweight structure and gravitational force can be ignored. In case of the RoboFly wing, however, the gravitational and buoyancy

forces are substantial. To remove the gravitational force from the analysis, every wing kinematic pattern was replayed at a 5 times slower frequency. As aerodynamic forces scale with the frequency squared, the magnitude gets reduced by a factor of 25. The FT-measurements of the slow frequency experiment correspond to the gravity force, and was subtracted from the fast frequency experiment after interpolation. After gravity subtraction, the remaining FT-data were assumed to accurately represent the aerodynamic forces and torques.

Up to this point, the aerodynamic and gravitational forces have been measured by the FT-sensor in the wing reference frame. For subsequent analysis it is useful to transfer the forces and torques to the SRF. Transferring from the SRF to the wing reference frame is described by the following Tait-Bryan operations:

$$F^w = R_\eta \cdot R_\theta \cdot R_\phi \cdot F^{SRF}, \tag{5.8}$$

where R_η, R_θ, R_η, correspond to the rotation matrices. To transfer from the wing reference frame to the SRF, one only needs to transpose the rotation matrices:

$$F^{SRF} = R_\phi^T \cdot R_\theta^T \cdot R_\eta^T \cdot F^w. \tag{5.9}$$

5.5 Computing inertial forces via the Newton-Euler equations

Besides aerodynamic forces, wing inertia plays a significant role in fly flight. Although the wing mass is only $\sim 0.2\%$ of the body mass (Lehmann and Dickinson, 1997), the angular velocity and acceleration of wing motion is very high. Inertial forces and torques of a rotating rigid body are computed using the Newton-Euler equations:

$$\begin{bmatrix} F_I \\ T_I \end{bmatrix} = \begin{bmatrix} m_w & -m_w[c_w\times] \\ m_w[c_w\times] & I_w - m_w[c_w\times][c_w\times] \end{bmatrix} \begin{bmatrix} a \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} m_w[\omega\times][\omega\times]c_w \\ [\omega\times](I_w - m_w[c_w\times][c_w\times])\omega \end{bmatrix}, \tag{5.10}$$

where m_w corresponds to the wing mass, c_w the position of the center of gravity on the wing, I_w the inertia tensor of the wing, a the linear acceleration of the wing, $\dot{\omega}$ the angular acceleration and ω the angular velocity, all relative to the wing reference frame. In this thesis, the effects of linear acceleration of the wing will be ignored, as they are small compared to the angular acceleration. The first right-hand term in equation 5.10 corresponds to the inertial forces and torques due to wing acceleration,

$FT_{I\ acc}$. Inertial effects dependent on angular velocity, such as the Coriolis and the centrifugal forces, are captured by the second right-hand term equation 5.10, and are collectively referred to as FT_{Ivel}.

Although the RoboFly wing includes the deformation angle, ξ, the Newton-Euler equations for four linked, rigid bodies would become very complex. It is therefore assumed that the inertial effects of wing deformation are small. The calculated inertial forces and torques are, therefore, based on the assumption that the wing is a rigid, flat plate.

The angular velocity and acceleration have to be computed from the Legendre polynomials describing wing motion. It is relatively easy to compute temporal derivatives of the Legendre polynomials ($\dot{\phi}$, $\dot{\theta}$, $\dot{\eta}$, $\ddot{\phi}$, $\ddot{\theta}$, $\ddot{\eta}$) using equation 3.42. However, the temporal derivatives of the wing kinematic angles do not correspond to the angular velocity and acceleration. Angular velocity in the wing reference frame is given by:

$$\omega = R_\eta \cdot R_\theta \cdot \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_\eta \cdot \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\eta} \\ 0 \end{bmatrix}, \tag{5.11}$$

where R_θ and R_η correspond to the rotation matrices of the Tait-Bryan operations for θ and η, respectively. Angular acceleration is given by:

$$\dot{\omega} = \dot{R}_\eta \cdot R_\theta \cdot \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_\eta \cdot \dot{R}_\theta \cdot \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_\eta \cdot R_\theta \cdot \begin{bmatrix} \ddot{\phi} \\ 0 \\ 0 \end{bmatrix} + \dot{R}_\eta \cdot \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} + R_\eta \cdot \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \ddot{\eta} \\ 0 \end{bmatrix}, \tag{5.12}$$

with $\dot{R}$ being the temporal derivative of the rotation matrix.

To compute the inertial forces and torques, one needs to know the mass, center of gravity, and the inertia tensor of the wing. These parameters were computed using the scaled 3D model of FlyNet, with a wing length of $2.7mm$, an estimated cuticle density of $1200kg/m^3$, and a wing thickness of $5.4\mu m$, Charles Porter Ellington, 1984a. The inertial parameters of the (left) wing are given by:

$$
\begin{aligned}
m_w &= 1.61 \cdot 10^{-9}[kg], \\
c_w &= \begin{bmatrix} -0.16 \\ 1.31 \\ 0 \end{bmatrix}[mm], \\
I_w &= \begin{bmatrix} 6.32 & 0.55 & 0 \\ 0.55 & 0.36 & 0 \\ 0 & 0 & 6.67 \end{bmatrix} \cdot 10^{-9}[kg \cdot mm^2].
\end{aligned} \tag{5.13}
$$

With the wing inertia parameters, angular velocity, and acceleration defined, it is possible to compute the inertial forces and torques for a given wing motion pattern. Figure 5.4, shows the aerodynamic and inertial forces and torques, in the SRF, for the baseline wingbeat. The aerodynamic forces were measured on RoboFly and have been rescaled to the fly scale using equations 5.4 and 5.5. To make interpretation easier, the forces and torques have been non-dimensionalized by dividing the forces by the body weight (mg), and the torques by the body weight multiplied with the wing length (mgR). Using the estimated cuticle density and the scaled body components of the 3D fly model, the body mass was found to be $1.16 \cdot 10^{-6}$ $[kg]$.

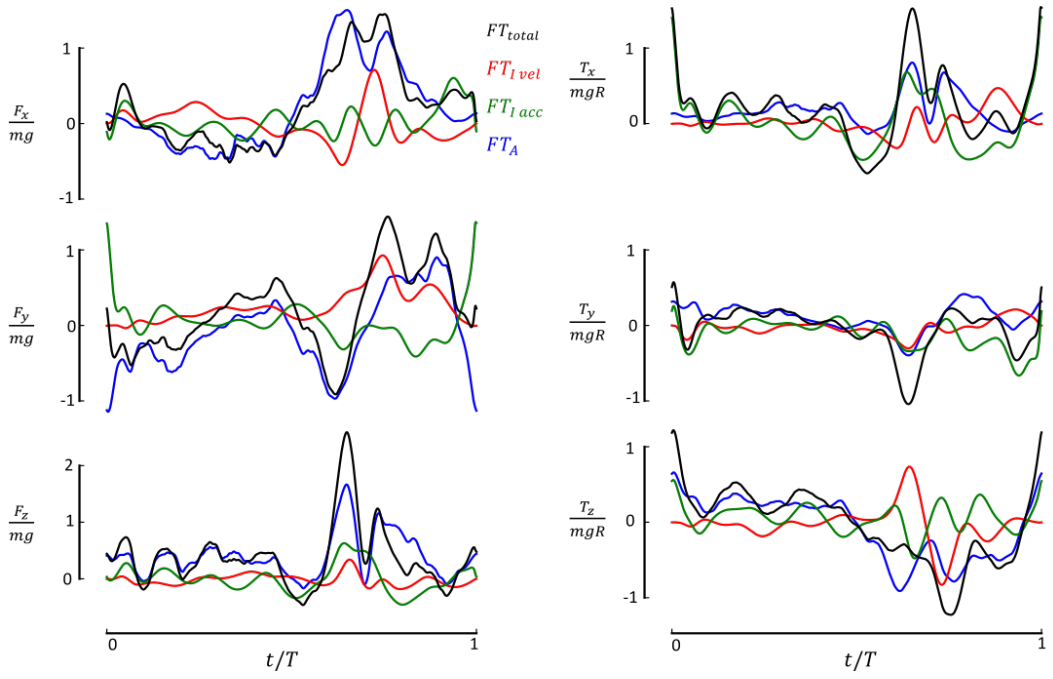

Figure 5.4: Aerodynamic (FT_A), inertial acceleration ($FT_{I\ acc}$), inertial velocity ($FT_{I\ vel}$) and total (FT_{total}) forces and torques for the baseline wingbeat.

The takeaway from Figure 5.4, is that the inertial forces and torques in wing motion are substantial, even though the wing mass is less than 0.2% of the body mass.

5.6 Aerodynamic and inertial force production of steering muscle activity

With the methods to measure aerodynamic forces and compute inertial forces established, the control forces generated by steering muscle activity can now be evaluated. For each steering muscle, seven wing kinematic patterns were tested on RoboFly. The seven wing kinematic patterns were found by sampling the muscle activity patterns on a line between the baseline muscle activity and the maximum muscle activity of a selected muscle, and subsequently predicting the corresponding wing motion using the trained CNN.

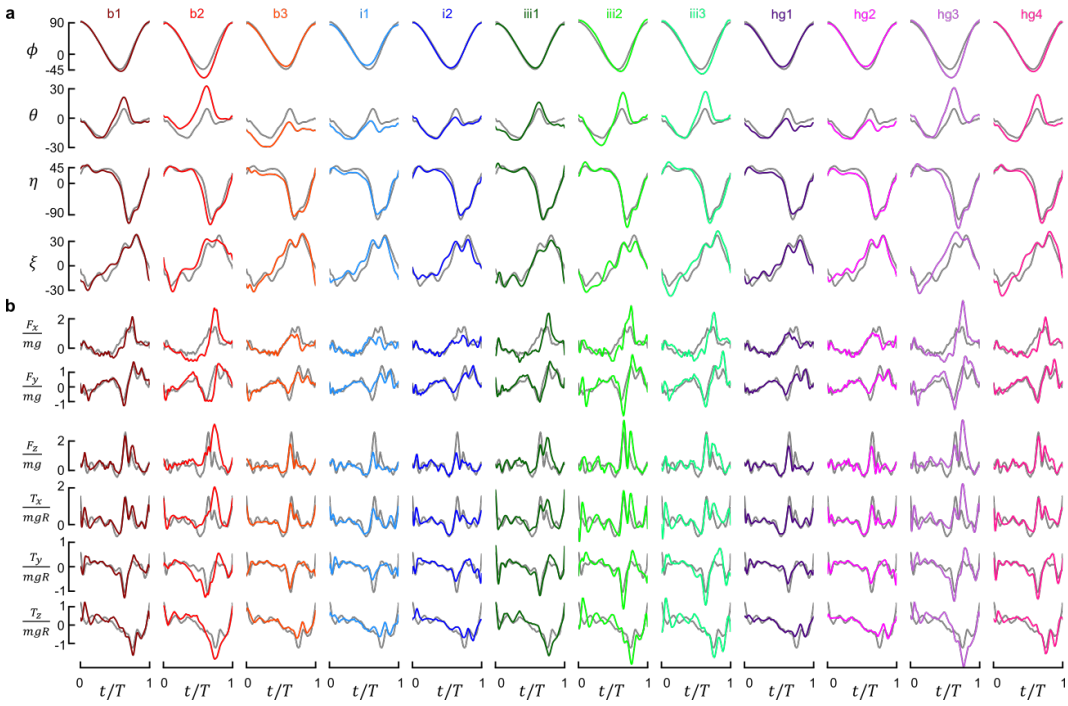

Figure 5.5: Wing motion and aerodynamic plus inertial forces and torques of maximum muscle activity wing kinematics. A: Baseline (grey) and maximum muscle activity (colored) wing kinematics. B: Aerodynamic plus inertial forces and torques for the wing kinematics in A.

The workflow to obtain the control forces and torques of muscle activity consists of: measuring the FT-traces of 84 wing kinematic patterns on RoboFly, performing the gravity subtraction, smoothing the traces with a Kalman filter, converting traces to the SRF, compute the median FT-traces from the measurement wingbeats, computing the inertial FT-traces, and finally, summing the inertial and aerodynamic forces

and torques during a wingbeat. Figure 5.5 shows the total forces and torques for the baseline wingbeat and the maximum muscle activity wing motion. Although there are some clear trends between wing motion and force and torque generation, discussing all details in Figure 5.5 would be a cumbersome exercise and does not necessarily provide substantial insight. Instead, the wingbeat-averaged forces and torques for the baseline wingbeat and maximum activity patterns are displayed in Figure 5.6.

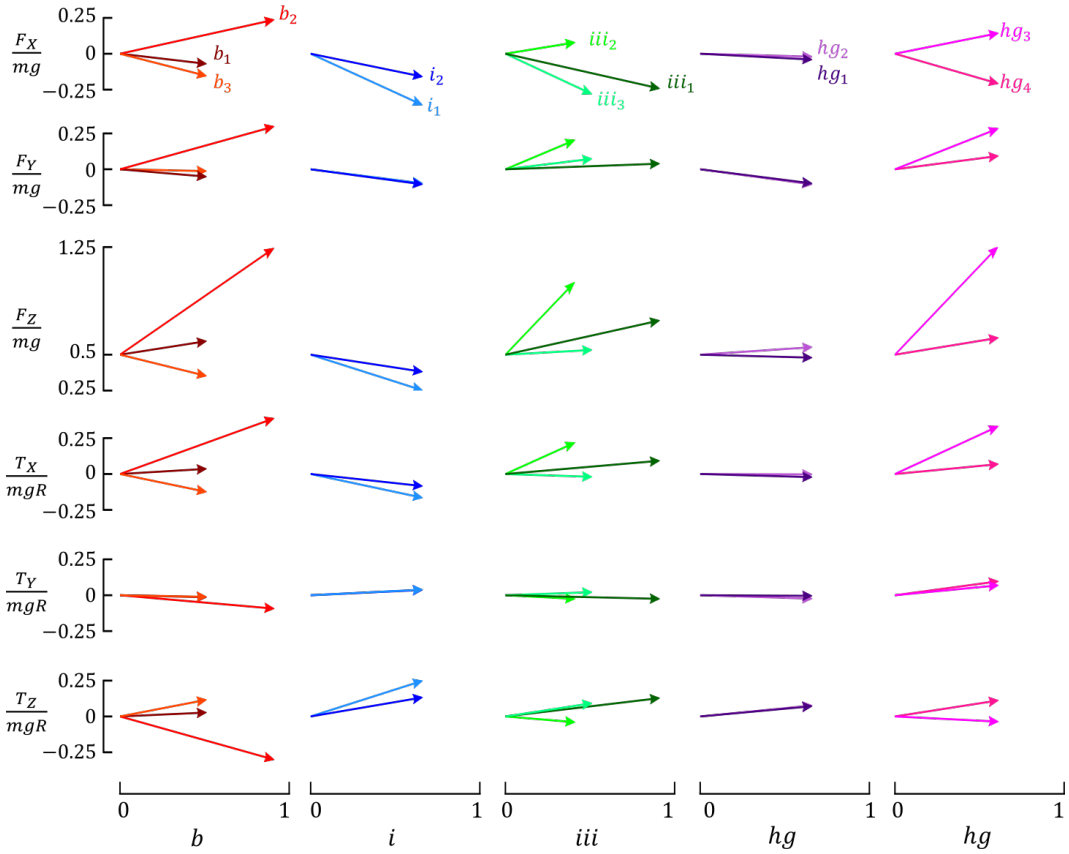

Figure 5.6: Wingbeat-averaged total FT for the baseline wingbeat and maximum muscle activity wing kinematic patterns of the left wing. The wingbeat-averaged FT of the baseline wingbeat are subtracted from the maximum muscle activity FT, except for F_Z, such that the baseline FT starts at 0 and $F_Z/mg = 0.5$. On the horizontal axis, the baseline muscle activity pattern is subtracted from the maximum muscle activity pattern.

The wingbeat-averaged total forces and torques in Figure 5.6 show that the left and right wing motion generate sufficient force for weight support ($F_Z/mg = 1$). For the b_2 and hg_3 muscle maximum activity patterns, the left and right wings produce a thrust force of $F_Z/mg = 2.5$. This level of thrust can accelerate the fly at $2.5g$,

when the strokeplane is oriented vertically. For free flight escape maneuvers, body accelerations between $2g$ and $3g$ have been observed (Muijres, Elzinga, Melis, et al., 2014). When the left and right wing are both actuated by the i_1 muscle, the maximum activity pattern, the vertical thrust force is reduced to $F_Z/mg = 0.5$. An interesting difference between the b_2 and hg_3 muscle maximum activity patterns can be seen for the pitch torque: b_2 muscle activity corresponds to a pitch-up torque ($T_Y/mgR < 0$) while the hg_3 muscle activity pattern corresponds to a pitch-down torque ($T_Y/mgR > 0$). The opposite effects on pitch torque by b_2 and hg_3 muscle activity shows that the fly can control pitch torque, while increasing thrust. When the left wing employs the maximum b_2 muscle activity pattern and the right wing the maximum i_1 muscle activity pattern, strong roll ($T_X/mgR > 0$) and yaw torques ($T_Z/mgR < 0$) to the right are created. Again, the strong coupling between roll and yaw torques has been observed during escape maneuvers (Muijres, Elzinga, Melis, et al., 2014).

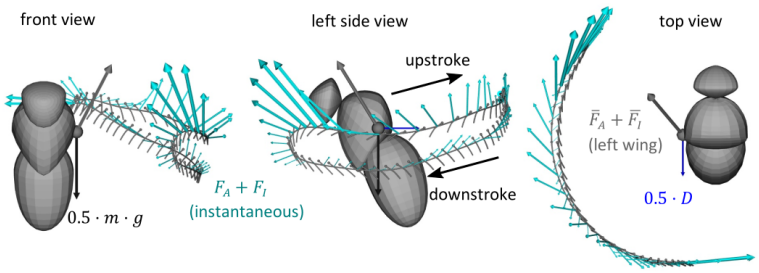

Figure 5.7: Lollipop figures of the baseline wingbeat and instantaneous forces (aerodynamic+inertial) from three views. The wingbeat-averaged aerodynamic force vector is displayed in grey (different scale than the instantaneous forces) and half of the body weight in black. Half the body drag force is displayed in blue, such that the fly is in force-equilibrium for the baseline wingbeat.

Figures 5.7 and 5.8 show the wing kinematics and instantaneous forces of the baseline wingbeat and maximum muscle activity patterns, respectively. A striking observation is the difference in force production between the downstroke and the upstroke. This difference might be an artefact of tethered flight, which increases the downstroke/upstroke ratio. In free flight, this ratio is around 0.55 but in tethered flight it is typically between 0.6 and 0.7. The slower wing motion during the downstroke affects the aerodynamic and inertial force production.

Another observation that can be made from the lollipop figures is the strong force production at the start of the upstroke, followed by a dip of zero or slightly negative lift force and a rapid increase in force during the second half of the upstroke. At

the start of the upstroke the wing has a high wing pitch rate, which indicates that rotational forces can at least partly explain the strong forces. The rapid recovery of (lift) force in the second half of the upstroke might be related to the rapid acceleration of the wing. Further research, either via CFD simulations or particle image velocimetry, is required to gain more insight in the aerodynamic mechanisms that are at play.

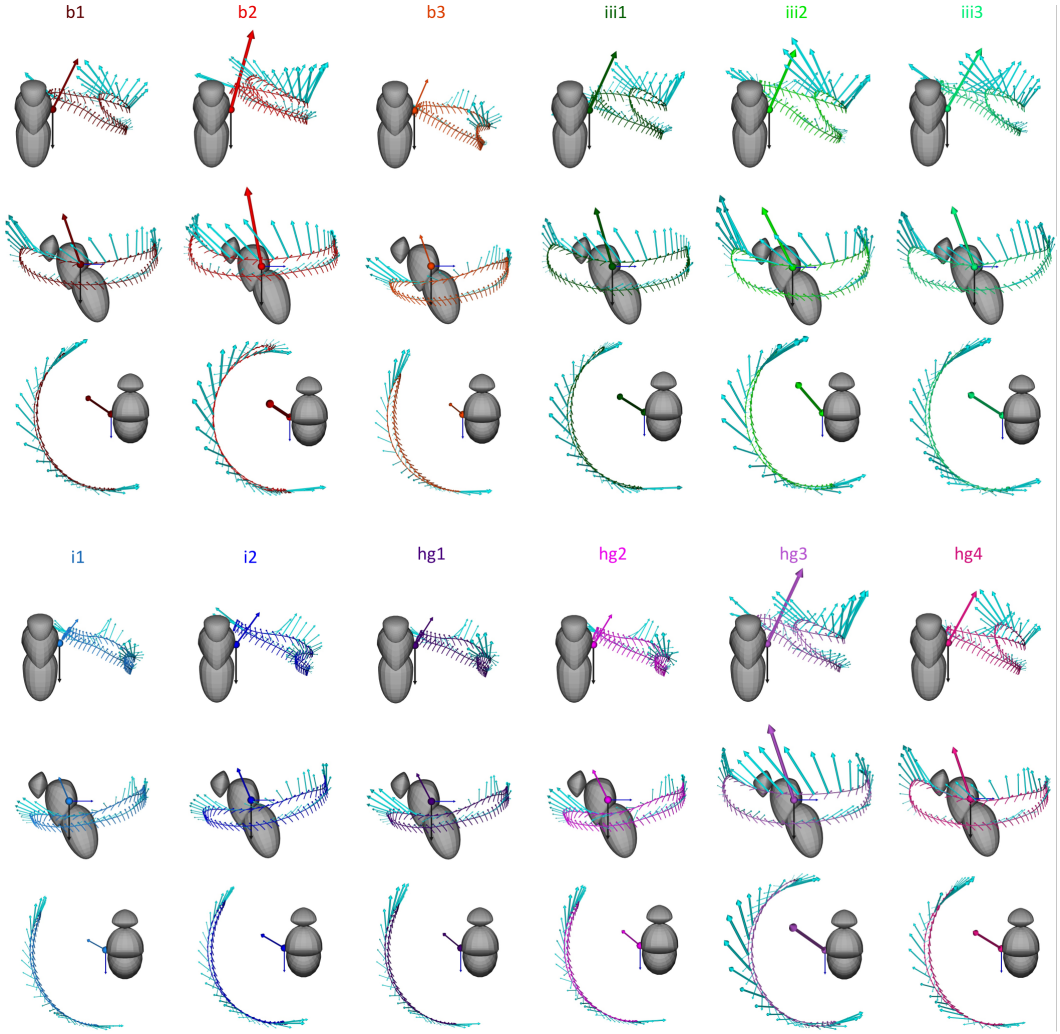

Figure 5.8: Lollipop figures for maximum muscle activity patterns of all muscles with the instantaneous aerodynamic plus inertial force in cyan. The wingbeat-averaged total force vector is displayed in the color of each muscle and the gravity (black) and body drag (blue) forces have the same magnitude as in Figure 5.7.

Chapter 6

RECONSTRUCTING FREE FLIGHT MANEUVERS WITH MODEL PREDICTIVE CONTROL

The non-linear mapping between steering muscle activity and wing motion that has been learned by a CNN, in combination with the RoboFly measurements and inertial force computations of the maximum muscle activity wing kinematics, form an accurate model of how the steering muscles can generate control forces and torques. To investigate flight control in insects, several studies have constructed state-space models of flapping flight (G. Taylor, R. Bomphrey, and 't Hoen, 2006, Dickson, Straw, and Dickinson, 2008, I. Faruque and Humbert, 2010, Taha, Hajj, and Beran, 2014, I. A. Faruque et al., 2018, Zahn et al., 2022). None of these studies include an accurate model of the wing kinematic control modes that are available to an insect, however. The results of Chapters 4 and 5 can be used to construct the control matrix of fly flight. This control matrix, in combination with a system matrix describing the physics of flight, forms the state-space system of fly flight.

In this chapter, I will discuss the different models that have been used to construct a state-space system of fly flight. A state-space system can be used to simulate free flight maneuvers using Model Predictive Control (MPC). By specifying an initial and goal state, and a time period to achieve the goal state, a MPC-controller will try to find the optimal trajectory given a cost function and constraints. I specified various goal states that mimic free flight maneuvers such as saccades, escape maneuvers, as well as forward, sideward and backward flight. The MPC-controller was able to predict the body trajectory, wing kinematics, and muscle activity for all specified goal states. Comparison with free flight studies shows that the body trajectory and wing kinematics of the simulated maneuvers are remarkably similar.

6.1 State-space representation of fly flight

State-space systems are ubiquitous in control theory and it is likely that the fly's nervous system incorporates some form of a state-space representation (Michael Dickinson, 2006, Michael H Dickinson and F. T. Muijres, 2016, Zahn et al., 2022). The state-space system used in this chapter constitutes as an explicit discrete time-variant state-space system. Although the nervous system of the fly and the underlying physics that are responsible for flight operate continuously in time, changes in

wing kinematics occur on a wingbeat-to-wingbeat basis. It is, therefore, logical to discretize the control problem on a wingbeat-to-wingbeat basis. Several studies show that mechano-sensory organs at the base of the haltere can detect Coriolis forces, but also serve as a *metronome* for the fly's nervous system (Dickerson et al., 2019, Dickerson, 2020). The haltere beats in anti-phase with respect to the wings, and afferent signals encoding haltere phase provide critical timing information to circuitry controlling the steering muscles of the wings.

The explicit discrete time-invariant state-space system is governed by the following equations:

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k), \\ y(k) &= C(k)x(k) + D(k)u(k), \end{aligned} \tag{6.1}$$

where $x(k)$ and $x(k+1)$ are the state vectors at times k and $k+1$ respectively, $u(k)$ is the control vector, $A(k)$ is the system matrix, $B(k)$ is the control matrix, $C(k)$ the output matrix, $D(k)$ the feed-forward matrix, and $y(k)$ the output vector (Figure 6.1). In the case of fly flight, there is no feed-forward process and the output of the state-space system is the state vector. This means that matrices $C(k)$ and $D(k)$ do not need to be defined.

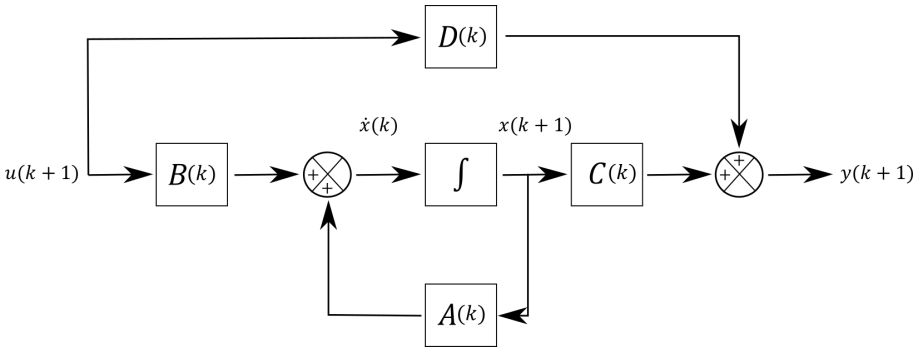

Figure 6.1: Overview of an explicit discrete time-variant state-space system, adapted from Ogata et al., 2010.

The state vector describes the orientation, position, velocity, and angular velocity of the fly's body:

$$x = \begin{bmatrix} v_x & v_y & v_z & \omega_x & \omega_y & \omega_z & q_0 & q_x & q_y & q_z & p_x & p_y & p_z \end{bmatrix}^T, \tag{6.2}$$

with v and ω being the linear and angular velocity of the body in the SRF, respectively, q is the quaternion describing SRF orientation relative to the inertial reference frame, and p is the position of the SRF in the inertial reference frame. Although the head, thorax and abdomen can move independently in FlyNet, I assume the head and abdomen to be stationary relative to the thorax in my simulations. This simplifies the inertia model of the body, although flies are known to move there head and abdomen during maneuvers (Zanker, Egelhaaf, and Warzecha, 1991, Hateren and Schilstra, 1999, G. J. Taylor et al., 2013).

The temporal derivative of the state vector, $\dot{x}$, is required for updating the state for each time step:

$$\dot{x} = \begin{bmatrix} a_x & a_y & a_z & \dot{\omega}_x & \dot{\omega}_y & \dot{\omega}_z & \omega_x & \omega_y & \omega_z & v_x & v_y & v_z \end{bmatrix}^T, \tag{6.3}$$

with a and $\dot{\omega}$ corresponding to the linear and angular acceleration of the body in the SRF, respectively. At each time step, the state vector is multiplied with the system matrix, $A(k)$, to compute the temporal derivative of the state vector. The time step, Δt, corresponds to the wingbeat period, which is assumed to be constant at $1/f = 1/200 = 0.005$ $[s]$ for simplicity. Although flies regulate wingbeat frequency during flight, it typically takes around 20 wingbeats to increase or decrease the wingbeat frequency to a new level (Muijres, Elzinga, Melis, et al., 2014). For simulating rapid maneuvers it is, therefore, not necessary to include variable wingbeat frequency.

The equations of motion of the system matrix include the following flight forces and torques: body weight, body inertia, body aerodynamics, and the aerodynamic and inertial damping of the wings. Inertial and aerodynamic damping effects will be discussed in the next section. The mass, center of gravity and inertia tensor of the body were determined from the scaled 3D body model (head, thorax, abdomen) for the average fly in the dataset. As for the wings, the cuticle density is assumed to be 1200 $[kg \cdot m^{-3}]$, and the body mass estimated as $m_b = 1.16 \cdot 10^{-6}$ $[kg]$. The center of gravity of the body in the SRF is estimated as $c_b = \begin{bmatrix} 0.04 & 0 & -0.24 \end{bmatrix}^T$ $[mm]$. Finally, the inertia tensor of the body is:

$$I_b = m_b \begin{bmatrix} 0.56 & 0 & -0.19 \\ 0 & 0.67 & 0 \\ -0.19 & 0 & 0.23 \end{bmatrix} [kg \cdot mm^2]. \tag{6.4}$$

The inertial forces and torques on the body can be computed with the Newton-Euler equations (equation 5.10). To simplify the expressions, the Newton-Euler equations are evaluated at the center of gravity of the body:

$$\begin{bmatrix} F_I \\ T_I \end{bmatrix}_b = \begin{bmatrix} m_b & 0 \\ 0 & I_b \end{bmatrix} \begin{bmatrix} a \\ \dot{\omega} \end{bmatrix}_b . \tag{6.5}$$

In order to obtain the (rotational) accelerations of the body, one only needs to solve the inverse problem:

$$\begin{bmatrix} a \\ \dot{\omega} \end{bmatrix}_b = \begin{bmatrix} m_b & 0 \\ 0 & I_b \end{bmatrix}^{-1} \begin{bmatrix} F_I \\ T_I \end{bmatrix}_b . \tag{6.6}$$

Because the body velocity and acceleration are expressed in the SRF, the gravitational force needs to be transformed from the inertial reference frame into the SRF:

$$[F_G]_b = m_b \cdot R_{SRF}(q) \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}, \tag{6.7}$$

with $R_{SRF}(q)$ as the rotation matrix, which can be found using equation 3.4, with the exclusion of the last column and bottom row. As the state is evaluated at the center of gravity, there are no torques due to the fly's weight.

Although the aerodynamics of the wings generate larger forces, the body of the fly experiences drag during flight. The combined 3D shape of the head, thorax, abdomen and legs is complex, and the body drag model is set to the worst-case scenario: a sphere with a radius of 1 $[mm]$. While the actual body drag is likely to be lower, the simplicity of the drag model makes it easier to implement in the state-space system. The drag on the sphere can be calculated as:

$$\begin{bmatrix} F_{Dx} \\ F_{Dy} \\ F_{Dz} \end{bmatrix} = \begin{bmatrix} -sgn(v_x) \cdot \frac{1}{2} C_D \pi \rho v_x^2 \\ -sgn(v_y) \cdot \frac{1}{2} C_D \pi \rho v_y^2 \\ -sgn(v_z) \cdot \frac{1}{2} C_D \pi \rho v_z^2 \end{bmatrix}, \tag{6.8}$$

with the drag coefficient as $C_D = 0.5$ and sgn as the sign operator. Because the center of pressure of the sphere is assumed to be at the center of gravity of the body, there are no torques due to body drag.

Although the specifics will be discussed in the next section, the two remaining FT-components (Force-Torque) come from the left and right wing motion. Because the aerodynamic and inertial damping terms are dependent on body (rotational) velocity, these FT-components are added to the system matrix. The equations of motion to compute linear and rotational accelerations can be written as:

$$\begin{bmatrix} a \\ \dot{\omega} \end{bmatrix}_{k+1} = \begin{bmatrix} m_b & 0 \\ 0 & I_b \end{bmatrix}^{-1} \left(\begin{bmatrix} F_G \\ 0 \end{bmatrix}_k + \begin{bmatrix} F_D \\ 0 \end{bmatrix}_k + \begin{bmatrix} F_{damp\ A} \\ T_{damp\ A} \end{bmatrix}_k + \begin{bmatrix} F_{damp\ I} \\ T_{damp\ I} \end{bmatrix}_k \right), \tag{6.9}$$

with F_{dampA} and T_{dampA} as the aerodynamic damping terms and F_{dampI} and T_{dampI} as the inertial damping terms.

The control inputs that are available to the fly are the muscle activity patterns of the left and right steering muscles: u^L and u^R. More details will be provided further on in this chapter, but the aerodynamic plus inertial control forces and torques can be computed as:

$$\begin{bmatrix} F_C \\ T_C \end{bmatrix}_{k+1} = \begin{bmatrix} \frac{dF}{du_L} u_L \\ \frac{dT}{du_L} u_L \end{bmatrix}_{k+1} + \begin{bmatrix} \frac{dF}{du_R} u_R \\ \frac{dT}{du_R} u_R \end{bmatrix}_{k+1}. \tag{6.10}$$

Adding the control forces and torques to equation 6.9, gives the complete equations of motion for body (rotational) acceleration:

$$\begin{bmatrix} a \\ \dot{\omega} \end{bmatrix}_{k+1} = \begin{bmatrix} m_b & 0 \\ 0 & I_b \end{bmatrix}^{-1} \left(\begin{bmatrix} F_G \\ 0 \end{bmatrix}_k + \begin{bmatrix} F_D \\ 0 \end{bmatrix}_k + \begin{bmatrix} F_{damp\ A} \\ T_{damp\ A} \end{bmatrix}_k + \begin{bmatrix} F_{damp\ I} \\ T_{damp\ I} \end{bmatrix}_k + \begin{bmatrix} F_C \\ T_C \end{bmatrix}_{k+1} \right), \tag{6.11}$$

The computed accelerations of equation 6.11 can subsequently be used to update the state vector. As the body accelerations and velocities are both in the SRF, the update of the body velocity is relatively simple:

$$\begin{bmatrix} v \\ \omega \end{bmatrix}_{k+1} = \begin{bmatrix} v \\ \omega \end{bmatrix}_k + \Delta t \begin{bmatrix} a \\ \dot{\omega} \end{bmatrix}_{k+1}. \tag{6.12}$$

The body quaternion and position are in the inertial reference frame and the (angular) velocity update needs to be transformed from the SRF to the inertial reference frame. Using quaternion multiplication, the quaternion update can be written as:

$$q_{k+1} = q_k + \frac{\Delta t}{2}\omega_{k+1} \otimes q_k = \begin{bmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{bmatrix}_k + \frac{\Delta t}{2} \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}_{k+1} \begin{bmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{bmatrix}_k . \quad (6.13)$$

After computing q_{k+1}, the quaternion needs to be normalized such that $\|q\| = 1$. The position update is given by:

$$p_{k+1} = p_k + \Delta t \cdot R_{SRF}^T v_{k+1}, \quad (6.14)$$

where R_{SRF}^T is computed using q_k.

It is impossible to write the state-update equations 6.11, 6.12, 6.13 and 6.14, in the form:

$$x(k+1) = A(k)x(k) + B(k)u(k), \quad (6.15)$$

because operations like quaternion normalization can not be embedded in a matrix multiplication. In practice, however, the matrix formulation is not required for using the state-space system for free flight simulations.

6.2 Aerodynamic and inertial damping

The aerodynamic and inertial forces for the maximum muscle activity patterns (Chapter 5) have been measured and computed in a stationary reference frame. During free flight, the body translates and rotates, which has an effect on both the inertial and aerodynamic forces on the wing (Cheng et al., 2010, Hedrick, Bo Cheng, and X. Deng, 2009, Bo Cheng and X. Deng, 2011). In this section, I will use the quasi-steady aerodynamic model and the Newton-Euler equations to compute the aerodynamic and inertial damping coefficients for fruit fly flight.

To compute the effects of inertial damping, one only needs to add one term to the angular velocity of the wing:

$$\omega = R_\eta \cdot R_\theta \cdot \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_\eta \cdot \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\eta} \\ 0 \end{bmatrix} + R_\eta \cdot R_\theta \cdot R_\phi \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_b , \quad (6.16)$$

where the last term adds the body angular velocity, converted to the wing reference frame. By inserting the redefined angular velocities of the left and right wings into the Newton-Euler equations, it is possible to compute the inertial forces and torques given a constant body angular velocity. Using the Newton-Euler equations, I computed the wingbeat-averaged inertial forces and torques for different body angular velocities, with baseline wing kinematics for both the left and right wing, Table 6.1. The values for body velocity are typical for fruit fly flight (Muijres, Elzinga, Melis, et al., 2014).

$v_x\ mm/s$	$v_y\ mm/s$	$v_z\ mm/s$	$\omega_x\ rad/s$	$\omega_y\ rad/s$	$\omega_z\ rad/s$
−1000	−1000	−1000	−100	−100	−100
−750	−750	−750	−75	−75	−75
−500	−500	−500	−50	−50	−50
−250	−250	−250	−25	−25	−25
0	0	0	0	0	0
250	250	250	25	25	25
500	500	500	50	50	50
750	750	750	75	75	75
1000	1000	1000	100	100	100

Table 6.1: Range of linear and angular velocities for damping experiments.

The wingbeat-averaged inertial forces and torques, that are generated by body rotation, show a linear trend with angular velocity. Translational body velocity does not cause any changes in inertial force or torque. The slopes of the linear trends form the *inertial damping matrix*:

$$\frac{dFT_I}{d\omega} = \begin{bmatrix} \frac{dF_x}{d\omega_x} & \frac{dF_x}{d\omega_y} & \frac{dF_x}{d\omega_z} \\ \frac{dF_y}{d\omega_x} & \frac{dF_y}{d\omega_y} & \frac{dF_y}{d\omega_z} \\ \frac{dF_z}{d\omega_x} & \frac{dF_z}{d\omega_y} & \frac{dF_z}{d\omega_z} \\ \frac{dT_x}{d\omega_x} & \frac{dT_x}{d\omega_y} & \frac{dT_x}{d\omega_z} \\ \frac{dT_y}{d\omega_x} & \frac{dT_y}{d\omega_y} & \frac{dT_y}{d\omega_z} \\ \frac{dT_z}{d\omega_x} & \frac{dT_z}{d\omega_y} & \frac{dT_z}{d\omega_z} \end{bmatrix} = \begin{bmatrix} 0 & -63.5 & 0 \\ -61.1 & 0 & -2.53 \\ 0 & 1.00 & 0 \\ -1.21 & 0 & 20.4 \\ 0 & 2.15 & 0 \\ -22.2 & 0 & -0.94 \end{bmatrix} \cdot 10^{-8}. \tag{6.17}$$

Rotational body velocity generates a torque in opposite direction for ω_x and ω_z, but a torque in the same direction for ω_y. The positive damping torque for pitch rotation means that flight is unstable around the pitch axis. There is a strong coupling between the roll rotation and yaw torque, and vice versa.

Aerodynamic damping can be computed using a quasi-steady model. As with the inertial forces, the quasi-steady simulations assume a rigid, flat, wing. The quasi-steady terms that were included in the model are the translational and rotational forces equations (1.14, 1.18, 1.30). Added mass and the Wagner effect have been ignored, as it adds complexity and both effects depend primarily on wing acceleration.

The angular velocity of the body can be added to the angular velocity of the wing using equation 6.16. Inclusion of the translational body velocity into the wing's angular velocity term is difficult, however. Therefore, I implemented the translational and rotational quasi-steady terms in blade-element formulation, where the wing is partitioned into spanwise sections and the air velocity vector is computed per section (Dickson, Straw, and Dickinson, 2008). Simulating the baseline wing kinematics on the left and right wings, with the body velocities of 6.1, yields the damping matrix for body translational velocity:

$$\frac{dFT_A}{dv} = \begin{bmatrix} \frac{dF_x}{dv_x} & \frac{dF_x}{dv_y} & \frac{dF_x}{dv_z} \\ \frac{dF_y}{dv_x} & \frac{dF_y}{dv_y} & \frac{dF_y}{dv_z} \\ \frac{dF_z}{dv_x} & \frac{dF_z}{dv_y} & \frac{dF_z}{dv_z} \\ \frac{dT_x}{dv_x} & \frac{dT_x}{dv_y} & \frac{dT_x}{dv_z} \\ \frac{dT_y}{dv_x} & \frac{dT_y}{dv_y} & \frac{dT_y}{dv_z} \\ \frac{dT_z}{dv_x} & \frac{dT_z}{dv_y} & \frac{dT_z}{dv_z} \end{bmatrix} = \begin{bmatrix} -19.3 & 0 & 53.4 \\ 0 & 1.82 & 0 \\ 19.7 & 0 & -21.5 \\ 0 & 4.91 & 0 \\ 22.5 & 0 & -46.5 \\ 0 & -9.71 & 0 \end{bmatrix} \cdot 10^{-7}, \tag{6.18}$$

and the damping matrix for body angular velocity:

$$\frac{dFT_A}{d\omega} = \begin{bmatrix} \frac{dF_x}{d\omega_x} & \frac{dF_x}{d\omega_y} & \frac{dF_x}{d\omega_z} \\ \frac{dF_y}{d\omega_x} & \frac{dF_y}{d\omega_y} & \frac{dF_y}{d\omega_z} \\ \frac{dF_z}{d\omega_x} & \frac{dF_z}{d\omega_y} & \frac{dF_z}{d\omega_z} \\ \frac{dT_x}{d\omega_x} & \frac{dT_x}{d\omega_y} & \frac{dT_x}{d\omega_z} \\ \frac{dT_y}{d\omega_x} & \frac{dT_y}{d\omega_y} & \frac{dT_y}{d\omega_z} \\ \frac{dT_z}{d\omega_x} & \frac{dT_z}{d\omega_y} & \frac{dT_z}{d\omega_z} \end{bmatrix} = \begin{bmatrix} 0 & 4.55 & 0 \\ -0.39 & 0 & -13.0 \\ 0 & -21.5 & 0 \\ -20.8 & 0 & -10.1 \\ 0 & -11.8 & 0 \\ -41.3 & 0 & -21.2 \end{bmatrix} \cdot 10^{-7}. \tag{6.19}$$

The aerodynamic damping coefficients are approximately an order of magnitude larger than the inertial damping coefficients. Both dF_{Ax}/dv_x and dF_{Az}/dv_z have a negative sign, which means that there is a force opposite to the direction of body motion. In case of dF_{Ay}/dv_y, dF_{Az}/dv_x, and dF_{Ax}/dv_z, the signs are positive however, which generates a force in the direction of body motion. Whether the

damping coefficient is negative or positive, depends on the effects of body velocity on parameters such as angle-of-attack, instantaneous air velocity, and orientation of the wing.

All the aerodynamic damping torques, $dT_A/d\omega$, have a negative sign. This means that for the baseline wing motion, flight is stable to perturbations in angular velocity. In Cheng et al., 2010 a similar result was found for a variety of wing kinematics.

Matrices $dFT_I/d\omega$, dFT_A/dv, and $dFT_A/d\omega$ can be converted into a single damping matrix. Multiplication of the damping matrix with the linear and angular velocity vectors of the body yields the damping forces and torques:

$$FT_{damp} = 10^{-7} \cdot \begin{bmatrix} -19.3 & 0 & 53.4 & 0 & -1.80 & 0 \\ 0 & 1.82 & 0 & -6.50 & 0 & -13.3 \\ 19.7 & 0 & -21.5 & 0 & -6.58 & 0 \\ 0 & 4.91 & 0 & -21.0 & 0 & -8.06 \\ 22.5 & 0 & -4.65 & 0 & -11.5 & 0 \\ 0 & -9.71 & 0 & -43.5 & 0 & -21.3 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \quad (6.20)$$

Although the expression for damping forces and torques is easy to use, the fact that it has been derived for baseline wing kinematics on both wings is a simplification of the underlying flight physics. A more accurate method would be, to compute the aerodynamic and inertial forces during the state-space simulations, with the actual left and right wing kinematic patterns, and the linear and angular velocity vectors of the body. It could be the case that the changes in wing kinematics during free flight maneuvers are such that the damping forces and torques are reduced. This would make the fly more maneuverable than the damping matrix for the baseline wing kinematics suggests.

6.3 Control matrix of fly flight

In my model, the control inputs that are available to the fly, are the muscle activity patterns of the left and right steering muscles: u_L, u_R. Using the RoboFly measurements of chapter 5, it is possible to compute the Jacobians (dFT/du_L, dFT/du_R) of the aerodynamic force and torque production for maximum muscle activity patterns, see Figure 5.6. Steering muscle activity is ordered as follows in the control vectors:

$$u_L = \begin{bmatrix} b_1^L & b_2^L & b_3^L & i_1^L & i_2^L & iii_1^L & iii_2^L & iii_3^L & hg_1^L & hg_2^L & hg_3^L & hg_4^L \end{bmatrix}^T, \quad (6.21)$$

and:

$$u_R = \begin{bmatrix} b_1^R & b_2^R & b_3^R & i_1^R & i_2^R & iii_1^R & iii_2^R & iii_3^R & hg_1^R & hg_2^R & hg_3^R & hg_4^R \end{bmatrix}^T. \quad (6.22)$$

The Jacobians for the aerodynamic force and torque production of the left and right steering muscles are given in Table 6.2. Control forces and torques can be computed by multiplying the Jacobian with the control vector:

$$FT_C^L = \frac{dFT}{du_L} u_L, \quad FT_C^R = \frac{dFT}{du_R} u_R. \quad (6.23)$$

The muscle activity correlation analysis in chapter 4 shows that not all combinations of muscle activity are possible. It is, therefore, required to impose constraints on the left and right control vectors. The constraint on muscle activity is that it has to lie on the 12D-surface given by Table 4.1. A mathematical formulation for the 12D-surface constraint is:

$$Cu_L = u_{base}, \quad Cu_R = u_{base}, \quad (6.24)$$

where C is the 12×12 correlation matrix described by Table 4.1, and u_{base} is the baseline muscle activity pattern (Table 4.2).

6.4 Model Predictive Control

The state-space system of fly flight can be used to simulate free flight maneuvers via the trajectory prediction by a controller. Although there are various types of controllers, such as proportional-integral-derivative (PID) controllers, the complex state update and the constraints on muscle activity, require a more sophisticated controller. In this section, I will describe how I combined the state-space system with a model predictive controller.

Model predictive control uses the state-space system to optimize trajectories in state-space, over a finite time window, for a given cost function (Kouvaritakis and Cannon, 2016). The state-space system is used to predict, one or multiple, state trajectories over a finite time window: the horizon. At each time step, the controller determines the optimal control input that generates a trajectory that minimizes the cost function, while being subject to system constraints. This process is repeated

	$\frac{F_X}{mg}$	$\frac{F_Y}{mg}$	$\frac{F_Z}{mg}$	$\frac{T_X}{mgR}$	$\frac{T_Y}{mgR}$	$\frac{T_Z}{mgR}$
b_1	−0.138	−0.101	0.185	0.070	−0.027	0.053
b_2	0.260	0.328	0.815	0.426	−0.103	−0.331
b_3	−0.303	−0.025	−0.290	−0.244	−0.022	0.227
i_1	−0.242	−0.159	−0.180	−0.127	−0.059	0.201
i_2	−0.540	−0.148	−0.376	−0.250	0.056	0.376
iii_1	−0.266	0.043	0.262	0.101	−0.028	0.140
iii_2	0.191	0.497	1.234	0.532	−0.061	−0.097
iii_3	−0.550	0.143	0.065	−0.039	0.041	0.180
hg_1	−0.062	−0.149	−0.031	−0.034	−0.006	0.108
hg_2	−0.031	−0.159	0.079	−0.004	−0.036	0.115
hg_3	0.234	0.466	1.236	0.545	0.113	−0.059
hg_4	−0.341	0.152	0.192	0.113	0.159	0.183

Table 6.2: Jacobian, dFT/du_L, of aerodynamic FT-production and left steering muscle activity. The Jacobian, dFT/du_R, for the right steering muscles can be obtained by multiplying columns F_y/mg, T_x/mgR and T_z/mgR by -1.

for every time step, shifting the horizon forward, and MPC is, therefore, also known as receding horizon control.

The software implementation of MPC is complex from a mathematical and coding perspective, and I, therefore, choose to use the Python-package *do-mpc* (Lucia et al., 2017). Do-mpc allows the user to setup MPC simulations for various control problems, including continuous time, discrete time, linear models, non-linear models, and robust control.

For the free flight simulations, the do-mpc controller is using discrete time and a non-linear model. The duration of all free flight simulations was 10 wingbeats, and the MPC horizon was set to 10 wingbeats as well. This means that the goal state is always "visible" to the controller. In robust MPC, it is possible to use multiple trajectories, or scenarios, with random variations in the control input at each time step. If a control problem contains uncertain parameters, the multiple scenarios make the MPC algorithm more robust to model prediction error. The state-space system of fly flight does not contain any uncertain parameters, and one trajectory is, therefore, sufficient.

At the center of MPC is the objective function, which is defined in do-mpc as:

$$J = \sum_{k=0}^{N-1} \left(l(x_k, u_k) + \Delta u_k^T R \Delta u_k \right) + m(x_N), \tag{6.25}$$

with N being the number of time steps of the problem, $l(x_k, u_k)$ is the Lagrange term, $\Delta u_k^T R \Delta u_k$ is the r-term, and $m(x_N)$ is the Meyer term. The Lagrange term provides the option to provide a reference state-trajectory, and add a penalty if the state deviates from this reference trajectory. Rapid changes in control input are penalized using the r-term: $\Delta u_k = u_k - u_{k-1}$, and R is a weighting matrix. Finally, the Meyer term evaluates how close the specified goal state is to the final state, x_N.

In case of the free flight simulations, the MPC objective function is defined as follows. First, the initial state, x_0, and the goal state, x_n, are specified. Subsequently, the Lagrange and Meyer terms are both defined as:

$$l(x_k, u_k) = m(x_N) = S_x \cdot (x_N - x_k)^2 , \tag{6.26}$$

with S_x as a scaling matrix. By setting the diagonal values of the S_x and R matrix, the user can tune the importance of certain aspects of the objective function. The Lagrange and Meyer terms are the same, as I did not want to specify any reference trajectory for the Lagrange term. In this way, the objective function allows the MPC controller to explore any trajectory that ends at the goal state. By setting a diagonal element to zero in the S_x matrix, the MPC controller will not optimize the trajectories for this parameter. If a diagonal element in S_x is set to a high value, the MPC controller will prioritize this element. For example, by setting the weight for the goal v_y to 1, the controller will punish any deviation from this goal heavily, and allows the user to enforce straight flight. In a similar way, the user can set the diagonal values of the R matrix, and determine the behavior of the steering muscles during the simulation. For the free flight simulations, I set all the diagonal values of R to 1.

Besides tuning the cost function, the user needs to specify the bounds of the state and control vectors. In case of the control vectors, all steering muscle activity is bounded between $[0, 1]$. The state vector is bounded as follows: linear velocity $[-10^4, 10^4]$ $[mm/s]$, angular velocity $[-1000, 1000]$ $[rad/s]$, quaternion bounds $[-1, 1]$, and position bounds $[-10^4, 10^4]$ $[mm]$. Both the linear and angular velocity bounds are much larger than what an actual fly can likely achieve in free flight.

After specifying the objective function and the state and control bounds, the state and control constraints need to be specified. The only constraint on the state is that the body quaternion needs to be a unit quaternion, $\|q\| = 1$. For the left and right steering muscles, the constraints are specified in equation 6.24. Do-mpc allows the

user to specify a constraint tolerance, i.e. how much the state or control vector can deviate from the constraint before the constraint will be enforced. For the free flight simulations, the constraint tolerance on the control vector constraints was set to $c_{tol} = 0.001$.

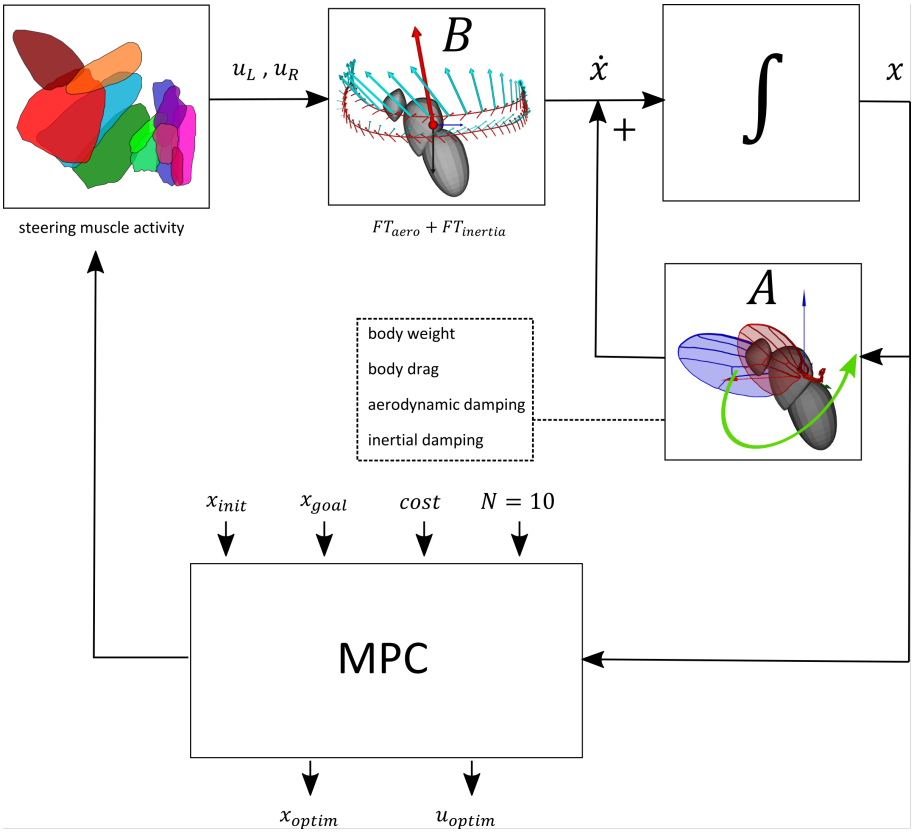

Figure 6.2: Schematic overview of the fly flight state-space system and the MPC controller.

With the objective function, state and control bounds, and the (non-)linear constraints defined, the MPC controller is now ready to use. Figure 6.2 shows the workflow of the MPC controller. The user only needs to specify the intial and goal states, time in which the fly should reach the goal state, and the weight matrix of the objective function, S_x. With these parameters, the MPC controller will try to find the optimal state trajectory and required control inputs, that minimizes the objective function while satisfying the constraints.

6.5 Simulating free flight maneuvers

Now that the state-space system of fly flight and the MPC problem are both defined, I will explore the MPC solutions for various virtual, free flight maneuvers. Table 6.3

shows the initial and goal states, and the goal weights for eight free flight scenarios. In Figures 6.3 and 6.4, the MPC solutions of the forward flight and a saccade to the left are displayed. The MPC results for the other maneuvers in Table 6.3 are shown in Appendix B.

The MPC simulation for forward flight in Figure 6.3 starts from hovering flight, and has a goal position ahead of the fly. As the goal position is too far to reach within 10 wingbeats, the MPC controller tries to maximize forward body acceleration. During the first wingbeats of the maneuver, the body pitches down rapidly, $\omega_y > 0$, thereby tilting the aerodynamic thrust vector forward. Pitch-down torque is generated by an increase in hg_3 and hg_4 activity and a simultaneous decrease in iii_3, i_2, hg_2 and b_3 activity. Although this muscle activity pattern decreases dorsal stroke amplitude, the pitch-down torque is primarily generated by the drop in deviation angle during the downstroke, which increases the moment arm with respect to the center of gravity of the body. The tilting of the aerodynamic thrust vector starts accelerating the fly forward. While the fly picks up speed, stronger iii_3 activity causes an increase in ventral stroke amplitude and higher deviation angles. These changes in wing kinematics stop the pitch down rotation of the fly and increase the overall aerodynamic thrust.

The body dynamics and wing kinematics computed by the forward flight MPC simulation correspond to the *helicopter model* of insect flight (David, 1978). According to the helicopter model, a fly will use roll and pitch rotations to direct the aerodynamic thrust vector into the direction it wants to move. The fly can subsequently use yaw rotations to align the body with the direction of motion. Several studies have corroborated the helicopter model for flies, and in particular the relationship between body pitch angle and forward flight speed: (Vogel, 1966, David, 1978, Zanker, 1988, Muijres, Elzinga, Melis, et al., 2014, M. J. Elzinga, Van Breugel, and Michael H Dickinson, 2014, Muijres, Elzinga, Iwasaki, et al., 2015). It is encouraging that the MPC results can replicate free flight fly behavior. Each simulated maneuver, in Figure 6.4 and Appendix B, makes use of the helicopter model to accelerate the fly in the desired direction.

A characteristic maneuver in fly flight is the saccade, a rapid turn of approximately 90° (Tammero and Michael H Dickinson, 2002, Fry, Sayaman, and Michael H Dickinson, 2003). At the start of the simulation, the fly has a forward velocity of 230 mm/s. The goal state requires a forward velocity of 230 mm/s as well, but at a yaw angle of 90° to the left with respect to the start orientation. During the

simulated maneuver, the fly first rolls slightly to the to the left while simultaneously yawing to the left and pitching down, Figure 6.4. The combination of roll and yaw tilts the aerodynamic thrust vector to the left. By tilting the aerodynamic thrust vector, the body velocity gets redirected to the left, and the yaw motion subsequently aligns the body orientation with the velocity vector. After the baseline wingbeat, the left i_1 and i_2 muscles show a slight uptick in activity, while the right i_2 muscle decreases in activity. One must keep in mind that the muscle activities were derived from GCaMP fluorescence, which means that changes in muscle activity that only last one or two wingbeats, have a relatively low impact on fluorescence. The effect on wing motion is very subtle, however; the wing pitch angle of the left wing is slightly lower than the right wing for the second wingbeat. This asymmetry in wing pitch angle generates a roll and yaw torque to the left. The hg_3 and hg_4 muscles of the left and right wings show an increase during the maneuver, which induces a pitch down torque similar to the forward flight maneuver. Comparing the wing kinematics of the simulated saccade to free flight saccades, shows that in both cases the yaw torque is generated by subtle asymmetries in the wing pitch angle (Bergou et al., 2010, Muijres, Elzinga, Iwasaki, et al., 2015).

The simulated maneuvers in Appendix B show similar body dynamics and wing kinematics, when being compared with free flight studies. For example, the sideward velocity maneuver in Figure B.2, shows a drop in deviation angle of the right wing and a rise in deviation angle of the left wing, similar to Ristroph, Berman, et al., 2009. Similarly, the simulated escape maneuvers in Figures B.3 and B.4 show similar trends as in Muijres, Elzinga, Melis, et al., 2014. The good correspondence between simulated wing kinematics and free flight data is a bit surprising, as the CNN was trained on tethered flight wing kinematics. Tethered and free flight wing motion patterns are different but it seems that the simulated flight conditions require wing kinematics that are closer to free flight (Fry, Sayaman, and Michael H Dickinson, 2003). Summarizing, the remarkably good agreement between simulated and free flight maneuvers suggests that my state-space system model of *Drosophila* captures the flight physics and steering muscle actuation with sufficient accuracy.

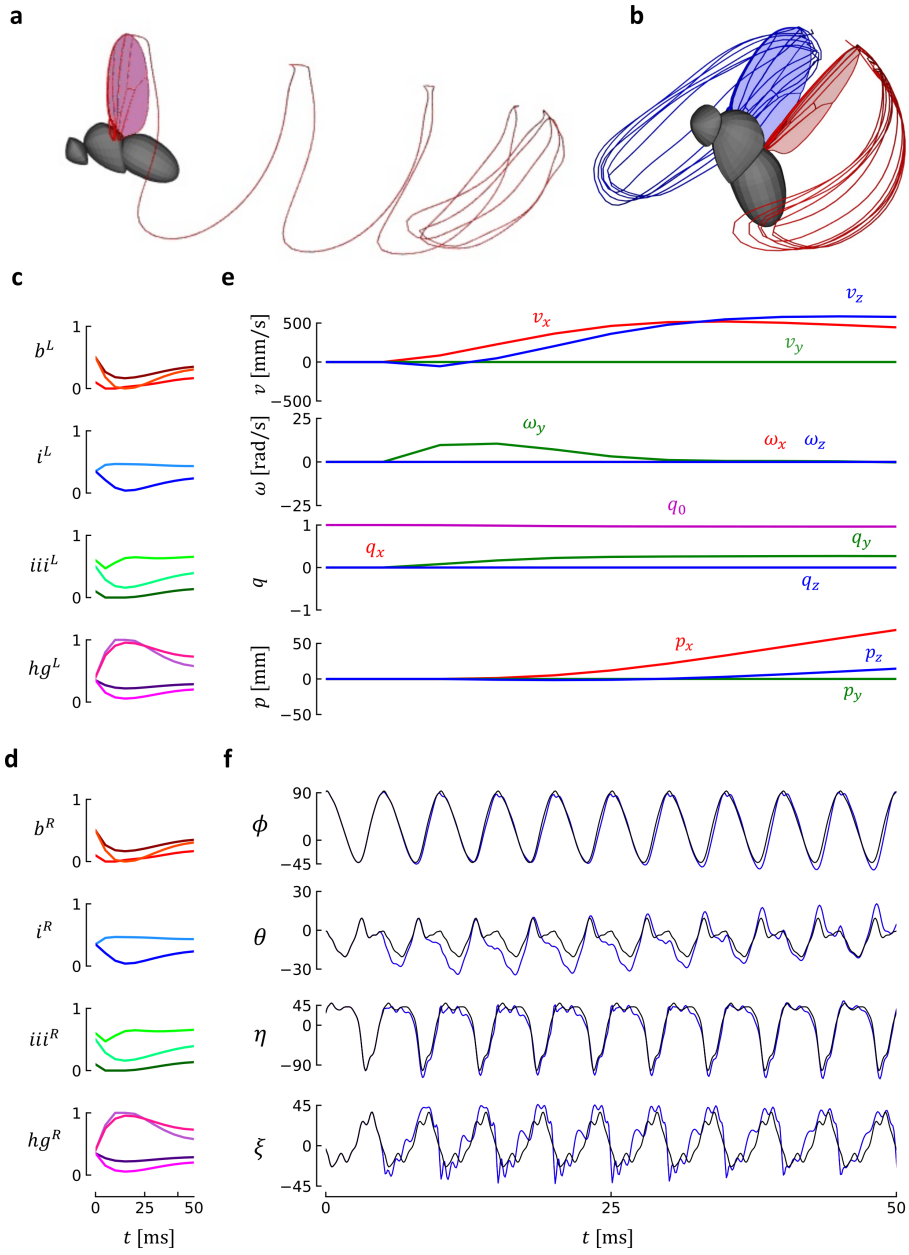

Figure 6.3: Forward flight maneuver. A: Side view at wingbeat 7 of the maneuver, wingtip traces are shown in red and blue. B: Wingtip traces at wingbeat 7 for stationary body state. C&D: Left and right steering muscle activity during the maneuver. E: Body state during the maneuver. F: Left (red) and right (blue) wing kinematics predicted from steering muscle activity in CD. The baseline wing kinematics are shown in black for comparison.

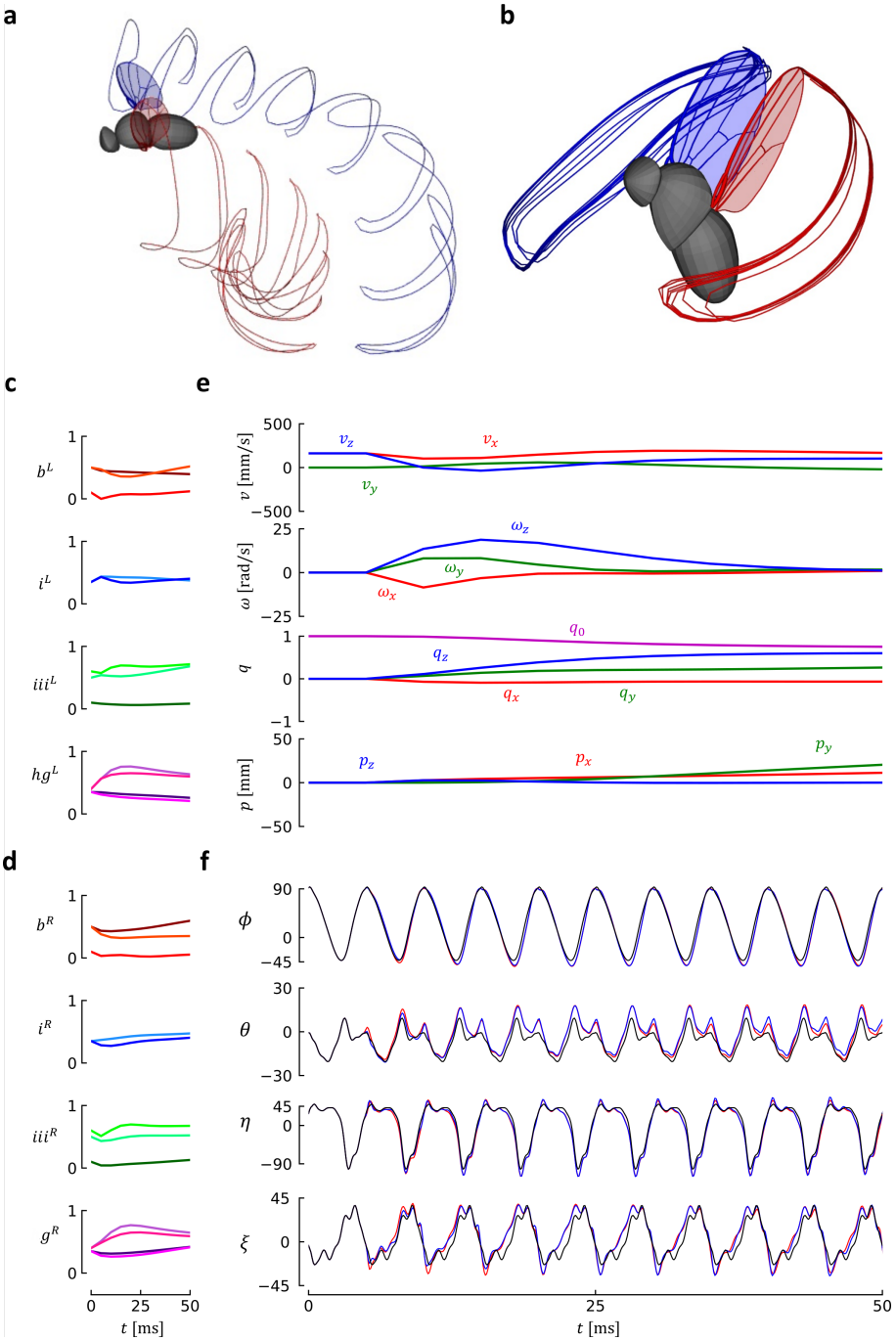

Figure 6.4: Saccade to the left. A: Top view at wingbeat 9 of the maneuver, wingtip traces are shown in red and blue. B: Wingtip traces at wingbeat 9 for stationary body state. C&D: Left and right steering muscle activity during the maneuver. E: Body state during the maneuver. F: Left (red) and right (blue) wing kinematics predicted from steering muscle activity in CD. The baseline wing kinematics are shown in black for comparison.

		forward velocity	backward velocity	sideward velocity	saccade left	saccade right	U-turn	loom left	loom front
x_{start}	v_x	0	0	0	163	163	163	0	0
	v_y	0	0	0	0	0	0	0	0
	v_z	0	0	0	163	163	163	0	0
	ω_y	0	0	0	0	0	0	0	0
	ω_y	0	0	0	0	0	0	0	0
	ω_y	0	0	0	0	0	0	0	0
	q_0	1	1	1	1	1	1	1	1
	q_x	0	0	0	0	0	0	0	0
	q_y	0	0	0	0	0	0	0	0
	q_z	0	0	0	0	0	0	0	0
	p_x	0	0	0	0	0	0	0	0
	p_y	0	0	0	0	0	0	0	0
	p_z	0	0	0	0	0	0	0	0
x_{goal}	v_x	325	-325	0	163	163	163	163	163
	v_y	0	0	-325	0	0	0	0	0
	v_z	488	488	0	163	163	163	163	163
	ω_y	0	0	0	0	0	0	0	0
	ω_y	0	0	0	0	0	0	0	0
	ω_y	0	0	0	0	0	0	0	0
	q_0	1	1	1	$\cos(\pi/4)$	$\cos(\pi/4)$	$\cos(\pi/2)$	$\cos(\pi/2)$	$\cos(\pi/4)$
	q_x	0	0	0	0	0	0	0	0
	q_y	0	0	0	0	0	0	0	0
	q_z	0	0	0	$\sin(\pi/4)$	$\sin(-\pi/4)$	$\sin(-\pi/2)$	$\sin(-\pi/2)$	$\sin(-\pi/4)$
	p_x	135	-135	0	27	27	-135	0	0
	p_y	0	0	-135	27	-27	0	-135	0
	p_z	0	0	0	0	0	0	0	0
S_x	v_x	0.1	0.1	1	0.1	0.1	1	0.1	0.1
	v_y	1	1	0.1	0.1	0.1	1	0.1	0.1
	v_z	0.1	0.1	1	0.1	0.1	1	0.1	0.1
	ω_y	1	1	1	1	1	1	1	1
	ω_y	1	1	1	1	1	1	1	1
	ω_y	1	1	1	1	1	1	1	1
	q_0	0	0	0	0	0	0	0	0
	q_x	1	1	1	0.5	0.5	1	1	1
	q_y	0	0	0	0	0	0	0	0
	q_z	1	1	1	0.5	0.5	1	1	1
	p_x	0.001	0.001	0.01	0	0	0	0.001	0.001
	p_y	0.01	0.01	0.001	0.001	0.001	0	0.001	0.001
	p_z	0	0	0.1	0.1	0.1	0.1	0.1	0.1

Table 6.3: MPC settings for virtual free flight maneuvers.

Chapter 7

CONCLUSIONS

After more than 400 million years of evolution, it is reasonable to assume that all aspects of the insect's flight system are highly optimized due to stringent and iterative natural selection. The question is, optimized for what? If an engineer could design the wings, wing hinge and musculature of a fly, he or she would probably focus on energy efficiency, maneuverability, and control. All these aspects are important in insect flight, but there are many other factors that need to be taken into consideration. For example, an insect cannot repair its wings and the flight system needs to be robust to damage or mechanical fatigue (Muijres, Iwasaki, et al., 2017). For some butterfly species, the shape of the wing is important for sexual selection (Le Roy, Debat, and Llaurens, 2019), while for other species the structure of the environment, such as forest versus open fields, determines the shape of the wing (Le Roy, Amadori, et al., 2021). Predation of moths by bats has led to the evolution of hairy scales on moth wings that can absorb ultrasound, however, these hairy scales deteriorate flight performance (Neil et al., 2020). The design of an insect's flight system is thus optimized for many factors, and trying to reverse-engineer this design is a complicated task.

A good example of biomechanical complexity is the wing hinge of a fly. Although the number of sclerites is low, the complex 3D shape of the sclerites, and the unknown mechanical properties of the tissue connecting them, makes it very challenging to construct a mechanical model of the wing hinge. As with most biological systems, the wing hinge is multi-functional, besides transforming thorax deformation into wing motion, the wing hinge also needs to facilitate folding of the wing, and male flies use the wing and wing hinge to sing during courtship (O'Sullivan et al., 2018). Besides multi-functionality, the shape of the wing hinge is subject to developmental constraints, i.e. the wing hinge and the wing have to grow from the same imaginal disc in the larvae. All these aspects of wing hinge evolution make it very difficult to infer mechanical function from anatomical observations alone. The complexity of the wing hinge, combined with the experimental challenges to image the sclerites during flight, has made the wing hinge an enigma, even after more than a century of research.

The complexity of the wing hinge was circumvented in this study, by simultaneously measuring the input, steering muscle activity, and output, wing kinematics, of the system. Combining real-time GCaMP imaging at 100 fps (Lindsay and Dickinson, 2016) and high-speed videography at 15,000 fps (Muijres, Elzinga, Melis, et al., 2014), required a highly automated experimental rig, because the high-speed video recording needed to be triggered on changes in muscle activity. The resulting experimental setup could record up to 8 seconds of high-speed video per fly in approximately one hour. In total, I collected 479 high-speed video sequences of 1.1 seconds from 82 flies, with synchronous muscle activity recording.

The vast high-speed video dataset created a challenge, as the wing pose needed to be extracted from the three orthogonal camera views. Although there were several automated wing pose reconstruction algorithms available, the clap-and-fling effect in tethered flight would require the user to annotate frames in each wingbeat, which was unfeasible given that the fly's wingbeat frequency is around 200 Hz. I, therefore, developed a novel tracking algorithm, named FlyNet, that combines pose prediction by a CNN with pose refinement via 3D model fitting by PSO. The main advantage of FlyNet is that it can track body and wing pose independent of time, i.e. if the tracked pose is inaccurate in one frame, it does not affect any other frames. This time independence helped FlyNet to automatically track wing pose, even during wingbeats when the clap-and-fling occurred. FlyNet can process 1 second of high-speed video data in 20 minutes on a high-end desktop computer. The combination of CNN prediction and PSO refinement proved to be accurate and robust for tracking tethered flight, and I am convinced that the methodology can be used successfully for any kind of 3D pose reconstruction from multiple camera views.

After tracking the dataset with FlyNet and converting the wing pose vector to four wing kinematic angles, Legendre polynomials were fitted to the wing kinematic traces of a wingbeat. By interpolating steering muscle activity at each wingbeat, a coupled dataset of steering muscle activity and associated Legendre coefficients was created. This coupled dataset was subsequently used to train a CNN to predict wing kinematics based on steering muscle activity. The CNN predictions are remarkably accurate, even when presented with unseen sequences of muscle activity. An correlation analysis of steering muscle fluorescence showed that steering muscle activity is highly correlated, and that, approximately, all muscle activity is constrained in a 12D plane. By moving to the point of maximum activity for a particular muscle on the 12D plane, one can find the maximum muscle activity patterns for any particular

muscle. The predicted wing motion for these maximum muscle activity patterns reveal the wing kinematic range that the muscles can modulate. Comparison with electrophysiology studies shows that, for the electrode-accessible steering muscles, similar trends in wing kinematic changes have been observed, providing support for my approach. By training an autoencoder on the coupled dataset, with an independent latent variable for each group of steering muscles, I could extract the effects on wing motion of each sclerite. This autoencoder analysis provided novel insights about the function of each sclerite in the wing hinge, especially for the fourth axillary sclerite, which has a strong effect on stroke amplitude, deviation and wing pitch.

The predicted wing motion for the maximum muscle activity patterns was replayed on a dynamically scaled robotic fly, RoboFly. Using the aerodynamic force-torque measurements of RoboFly and inertial force-torque computations via the Newton-Euler equations, I constructed a map between steering muscle activity and control forces and torques. This map was subsequently used in the control model of a state-space system of fly flight. Using the state-space system in combination with MPC, I could simulate various free flight maneuvers. The good correspondence between the body and wing kinematics of the simulated maneuvers and free flight studies provides confidence in the accuracy of the simulations.

The state-space system of fly flight does not only serve as a validation of how well the flight physics and role of the steering muscles are captured. By defining the state-space system, I have formulated the control problem that a fly needs to solve, every wingbeat, to stabilize flight and perform maneuvers. Because the steering muscles are innervated by only one motorneuron, muscle fluorescence serves as a proxy of the neural activity. Recently, several studies have published maps of neural connectivity of the *Drosophila* brain and upstream neurons of the motor-neurons (Scheffer et al., 2020, Phelps et al., 2021). With a map of neural connectivity, and a description of the control problem to be solved, it might be possible to understand how a fly performs the neural computations required for such a complex behavior as free flight.

BIBLIOGRAPHY

Agrawal, Sweta, Steve Safarik, and Michael Dickinson (2014). "The relative roles of vision and chemosensation in mate recognition of Drosophila melanogaster". In: *Journal of Experimental Biology* 217.15, pp. 2796–2805.

Akerboom et al. (2012). "Optimization of a GCaMP calcium indicator for neural activity imaging". In: *Journal of neuroscience* 32.40, pp. 13819–13840.

Almudi, Isabel et al. (2019). "Establishment of the mayfly Cloeon dipterum as a new model system to investigate insect evolution". In: *Evodevo* 10.1, pp. 1–10.

Altshuler, Douglas L et al. (2005). "Short-amplitude high-frequency wing strokes determine the aerodynamics of honeybee flight". In: *Proceedings of the National Academy of Sciences* 102.50, pp. 18213–18218.

Anderson, John D (2009). "Fundamentals of aerodynamics". In: *McGraw*.

Arshavsky, Vadim Y (2010). "Vision: the retinoid cycle in Drosophila". In: *Current Biology* 20.3, R96–R98.

Balint and Dickinson (2001). "The correlation between wing kinematics and steering muscle activity in the blowfly Calliphora vicina". In: *Journal of experimental biology* 204.24, pp. 4213–4226.

– (2004). "Neuromuscular control of aerodynamic forces and moments in the blowfly, Calliphora vicina". In: *Journal of experimental biology* 207.22, pp. 3813–3838.

Bastock, Margaret and Aubrey Manning (1955). "The courtship of Drosophila melanogaster". In: *Behaviour* 8.1, pp. 85–110.

Beatus, Guckenheimer, and Cohen (2015). "Controlling roll perturbations in fruit flies". In: *Journal of The Royal Society Interface* 12.105, p. 20150075.

Bergou et al. (2010). "Fruit flies modulate passive wing pitching to generate in-flight turns". In: *Physical review letters* 104.14, p. 148101.

Birch, James M and Michael H Dickinson (2001). "Spanwise flow and the attachment of the leading-edge vortex on insect wings". In: *Nature* 412.6848, pp. 729–733.

Birch, James M, William B Dickson, and Michael H Dickinson (2004). "Force production and flow structure of the leading edge vortex on flapping wings at high and low Reynolds numbers". In: *Journal of Experimental Biology* 207.7, pp. 1063–1072.

Boettiger and Furshpan (1952). "The mechanics of flight movements in Diptera". In: *The Biological Bulletin* 102.3, pp. 200–211.

Bomphrey et al. (2017). "Smart wing rotation and trailing-edge vortices enable high frequency mosquito flight". In: *Nature* 544.7648, pp. 92–95.

Cheng, B et al. (2010). "Aerodynamic damping during rapid flight maneuvers in the fruit fly Drosophila". In: *Journal of Experimental Biology* 213.4, pp. 602–612.

Cheng, Bo and Xinyan Deng (2011). "Translational and rotational damping of flapping flight and its dynamics and stability at hovering". In: *IEEE Transactions on Robotics* 27.5, pp. 849–864.

Clapham and Karr (2012). "Environmental and biotic controls on the evolutionary history of insect body size". In: *Proceedings of the National Academy of Sciences* 109.27, pp. 10927–10930.

Dana et al. (2019). "High-performance calcium sensors for imaging activity in neuronal populations and microcompartments". In: *Nature methods* 16.7, pp. 649–657.

David, Charles T (1978). "The relationship between body angle and flight speed in free-flying Drosophila". In: *Physiological entomology* 3.3, pp. 191–195.

Deng, J. et al. (2009). "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*.

Deora, Gundiah, and Sane (2017). "Mechanics of the thorax in flies". In: *Journal of Experimental Biology* 220.8, pp. 1382–1395.

Deora, Singh, and Sane (2015). "Biomechanical basis of wing and haltere coordination in flies". In: *Proceedings of the National Academy of Sciences* 112.5, pp. 1481–1486.

Dickerson, Bradley H (2020). "Timing precision in fly flight control: integrating mechanosensory input with muscle physiology". In: *Proceedings of the Royal Society B* 287.1941, p. 20201774.

Dickerson, Bradley H et al. (2019). "Flies regulate wing motion via active control of a dual-function gyroscope". In: *Current Biology* 29.20, pp. 3517–3524.

Dickinson and Gotz (1993). "Unsteady aerodynamic performance of model wings at low Reynolds numbers". In: *Journal of experimental biology* 174.1, pp. 45–64.

Dickinson, Lehmann, and Gotz (1993). "The active control of wing rotation by Drosophila". In: *Journal of experimental biology* 182.1, pp. 173–189.

Dickinson, Lehmann, and Sane (1999). "Wing rotation and the aerodynamic basis of insect flight". In: *Science* 284.5422, pp. 1954–1960.

Dickinson, M (1994). "The effects of wing rotation on unsteady aerodynamic performance at low Reynolds numbers". In: *The Journal of experimental biology* 192.1, pp. 179–206.

Dickinson, Michael (2006). "Insect flight". In: *Current Biology* 16.9, R309–R314.

Dickinson, Michael H and Florian T Muijres (2016). "The aerodynamics and control of free flight manoeuvres in Drosophila". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 371.1704, p. 20150388.

Dickinson and Tu (1997). “The function of dipteran flight muscle”. In: *Comparative Biochemistry and Physiology Part A: Physiology* 116.3, pp. 223–238.

Dickson, Straw, and Dickinson (2008). “Integrative model of Drosophila flight”. In: *AIAA journal* 46.9, pp. 2150–2164.

Dickson, William B et al. (2010). “A linear systems analysis of the yaw dynamics of a dynamically scaled insect model”. In: *Journal of Experimental Biology* 213.17, pp. 3047–3061.

Dudley, Robert (2002). *The biomechanics of insect flight: form, function, evolution*. Princeton University Press.

Dykstra, Richard L (1983). “An algorithm for restricted least squares regression”. In: *Journal of the American Statistical Association* 78.384, pp. 837–842.

Egelhaaf, Martin (2013). “Visual processing in free flight”. In: *Encyclopedia of Computational Neuroscience*, pp. 1–21.

Ellington, Charles Porter (1984a). “The aerodynamics of hovering insect flight. II. Morphological parameters”. In: *Philosophical Transactions of the Royal Society of London. B, Biological Sciences* 305.1122, pp. 17–40.

– (1984b). “The aerodynamics of hovering insect flight. IV. Aerodynamic mechanisms”. In: *Philosophical Transactions of the Royal Society of London. B, Biological Sciences* 305.1122, pp. 79–113.

– (1984c). “The aerodynamics of hovering insect flight. VI. Lift and power requirements”. In: *Philosophical Transactions of the Royal Society of London. B, Biological Sciences* 305.1122, pp. 145–181.

Ellington, CP (1995). “Unsteady aerodynamics of insect flight.” In: *Symposia of the society for experimental biology*. Vol. 49, pp. 109–129.

Ellington et al. (1996). “Leading-edge vortices in insect flight”. In: *Nature* 384.6610, pp. 626–630.

Elzinga, Michael J, William B Dickson, and Michael H Dickinson (2012). “The influence of sensory delay on the yaw dynamics of a flapping insect”. In: *Journal of The Royal Society Interface* 9.72, pp. 1685–1696.

Elzinga, Michael J, Floris Van Breugel, and Michael H Dickinson (2014). “Strategies for the stabilization of longitudinal forward flapping flight revealed using a dynamically-scaled robotic fly”. In: *Bioinspiration & biomimetics* 9.2, p. 025001.

Ennos, A Roland (1987). “A comparative study of the flight mechanism of Diptera”. In: *Journal of Experimental Biology* 127.1, pp. 355–372.

Ewing, Arthur W and HC Bennet-Clark (1968). “The courtship songs of Drosophila”. In: *Behaviour* 31.3-4, pp. 288–301.

Fabian, Schneeberg, and Beutel (2016). “Comparative thoracic anatomy of the wild type and wingless (wg1cn1) mutant of Drosophila melanogaster (Diptera)”. In: *Arthropod structure & development* 45.6, pp. 611–636.

Faruque, Imraan and J Sean Humbert (2010). "Dipteran insect flight dynamics. Part 1 Longitudinal motion about hover". In: *Journal of theoretical biology* 264.2, pp. 538–552.

Faruque, Imraan A et al. (2018). "Identification of optimal feedback control rules from micro-quadrotor and insect flight trajectories". In: *Biological cybernetics* 112.3, pp. 165–179.

Fontaine et al. (2009). "Wing and body motion during flight initiation in Drosophila revealed by automated visual tracking". In: *Journal of Experimental Biology* 212.9, pp. 1307–1323.

Fry, Steven N, Rosalyn Sayaman, and Michael H Dickinson (2003). "The aerodynamics of free-flight maneuvers in Drosophila". In: *Science* 300.5618, pp. 495–498.

Grewal, Mohinder S and Angus P Andrews (2014). *Kalman filtering: Theory and Practice with MATLAB*. John Wiley & Sons.

Hateren, JHV and C Schilstra (1999). "Blowfly flight and optic flow. II. Head movements during flight". In: *Journal of Experimental Biology* 202.11, pp. 1491–1500.

Hedenström, Anders (2014). "How insect flight steering muscles work". In: *PLoS biology* 12.3, e1001822.

Hedrick, Tyson L, Bo Cheng, and Xinyan Deng (2009). "Wingbeat time and the scaling of passive rotational damping in flapping flight". In: *Science* 324.5924, pp. 252–255.

Iwamoto, Hiroyuki (2011). "Structure, function and evolution of insect flight muscle". In: *Biophysics* 7, pp. 21–28.

Kalman, Rudolph Emil (1960). "A new approach to linear filtering and prediction problems". In.

Karashchuk, Pierre et al. (2021). "Anipose: a toolkit for robust markerless 3D pose estimation". In: *Cell reports* 36.13, p. 109730.

Kelly, Elchert, and Kahl (2017). "Dissection and immunofluorescent staining of mushroom body and photoreceptor neurons in adult Drosophila melanogaster brains". In: *JoVE (Journal of Visualized Experiments)* 129, e56174.

Kouvaritakis, Basil and Mark Cannon (2016). "Model predictive control". In: *Switzerland: Springer International Publishing* 38.

Kuipers, Jack B (1999). *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton university press.

Kwon, Young-Hoo (1998). *DLT method*. URL: `http://www.kwon3d.com/theory/dlt/dlt.html` (visited on 10/13/2022).

Lauder, George V (2001). "Flight of the robofly". In: *Nature* 412.6848, pp. 688–689.

Le Roy, Camille, Dario Amadori, et al. (2021). “Adaptive evolution of flight in Morpho butterflies”. In: *Science* 374.6571, pp. 1158–1162.

Le Roy, Camille, Vincent Debat, and Violaine Llaurens (2019). “Adaptive evolution of butterfly wing shape: from morphology to behaviour”. In: *Biological Reviews* 94.4, pp. 1261–1281.

Lehmann and Dickinson (1997). “The changes in power requirements and muscle efficiency during elevated force production in the fruit fly Drosophila melanogaster.” In: *The Journal of experimental biology* 200.7, pp. 1133–1143.

Leitch et al. (2021). “The long-distance flight behavior of Drosophila supports an agent-based model for wind-assisted dispersal in insects”. In: *Proceedings of the National Academy of Sciences* 118.17, e2013342118.

Lentink and Dickinson (2009). “Rotational accelerations stabilize leading edge vortices on revolving fly wings”. In: *Journal of experimental biology* 212.16, pp. 2705–2719.

Lentink, Dickson, et al. (2009). “Leading-edge vortices elevate lift of autorotating plant seeds”. In: *Science* 324.5933, pp. 1438–1440.

Lindsay and Dickinson (2016). “Functional imaging from the muscles of the fruit fly wing-hinge during tethered flight”. In: *Integrative and Comparative Biology* 56.S1, E128.

Lindsay, Sustar, and Dickinson (2017). “The function and organization of the motor system controlling flight maneuvers in flies”. In: *Current Biology* 27.3, pp. 345–358.

Liu and Aono (2009). “Size effects on insect hovering aerodynamics: an integrated computational study”. In: *Bioinspiration & Biomimetics* 4.1, p. 015002.

Lucia, Sergio et al. (2017). “Rapid development of modular and sustainable nonlinear model predictive control solutions”. In: *Control Engineering Practice* 60, pp. 51–62.

Mathis et al. (2018). “DeepLabCut: markerless pose estimation of user-defined body parts with deep learning”. In: *Nature neuroscience* 21.9, pp. 1281–1289.

MIYAN and EWING (1985). “Is the ‘click’mechanism of dipteran flight an artefact of CC14 anaesthesia?” In: *Journal of experimental biology* 116.1, pp. 313–322.

Miyan and Ewing (1985). “How Diptera move their wings: a re-examination of the wing base articulation and muscle systems concerned with flight”. In: *Philosophical Transactions of the Royal Society of London. B, Biological Sciences* 311.1150, pp. 271–302.

Muijres, Elzinga, Iwasaki, et al. (2015). “Body saccades of Drosophila consist of stereotyped banked turns”. In: *The Journal of experimental biology* 218.6, pp. 864–875.

Muijres, Elzinga, Melis, et al. (2014). “Flies evade looming targets by executing rapid visually directed banked turns”. In: *Science* 344.6180, pp. 172–177.

Muijres, Iwasaki, et al. (2017). “Flies compensate for unilateral wing damage through modular adjustments of wing and body kinematics”. In: *Interface Focus* 7.1, p. 20160103.

Muijres, Johansson, et al. (2008). “Leading-edge vortex improves lift in slow-flying bats”. In: *Science* 319.5867, pp. 1250–1253.

Nakai, Ohkura, and Imoto (2001). “A high signal-to-noise Ca2+ probe composed of a single green fluorescent protein”. In: *Nature biotechnology* 19.2, pp. 137–141.

Nalbach, Gerbera (1989). “The gear change mechanism of the blowfly (Calliphora erythrocephala) in tethered flight”. In: *Journal of Comparative Physiology A* 165.3, pp. 321–331.

Namiki, Shigehiro et al. (2022). “A population of descending neurons that regulates the flight motor of Drosophila”. In: *Current Biology* 32.5, pp. 1189–1196.

Neil, Thomas R et al. (2020). “Moth wings are acoustic metamaterials”. In: *Proceedings of the National Academy of Sciences* 117.49, pp. 31134–31141.

O’Sullivan, Angela et al. (2018). “Multifunctional wing motor control of song and flight”. In: *Current Biology* 28.17, pp. 2705–2717.

Ogata, Katsuhiko et al. (2010). *Modern control engineering*. Vol. 5. Prentice hall Upper Saddle River, NJ.

Page, Jonathan (2018). “The biomechanics of the dipteran wing hinge”. PhD thesis. University of Oxford.

PFAU, HANS K (1987). “Critical comments on a Novel mechanical model of dipteran flight (Miyan & Ewing, 1985)”. In.

Phelps, Jasper S et al. (2021). “Reconstruction of motor control circuits in adult Drosophila using automated transmission electron microscopy”. In: *Cell* 184.3, pp. 759–774.

Pishchulin, Leonid et al. (2016). “Deepcut: Joint subset partition and labeling for multi person pose estimation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4929–4937.

Poelma, C, WB Dickson, and MH Dickinson (2006). “Time-resolved reconstruction of the full velocity field around a dynamically-scaled flapping wing”. In: *Experiments in Fluids* 41.2, pp. 213–225.

Pringle, John William Sutton (1957). *Insect flight*. Vol. 9. Cambridge University Press.

Pringle, JWS (1949). “The excitation and contraction of the flight muscles of insects”. In: *The Journal of physiology* 108.2, p. 226.

Reiser and Dickinson (2008). "A modular display system for insect behavioral neuroscience". In: *Journal of neuroscience methods* 167.2, pp. 127–139.

Ristroph, Bergou, Guckenheimer, et al. (2011). "Paddling mode of forward flight in insects". In: *Physical review letters* 106.17, p. 178103.

Ristroph, Bergou, Ristroph, et al. (2010). "Discovering the flight autostabilizer of fruit flies by inducing aerial stumbles". In: *Proceedings of the National Academy of Sciences* 107.11, pp. 4820–4824.

Ristroph, Berman, et al. (2009). "Automated hull reconstruction motion tracking (HRMT) applied to sideways maneuvers of free-flying insects". In: *Journal of Experimental Biology* 212.9, pp. 1324–1335.

Ross, Andrew (2017). "Insect evolution: the origin of wings". In: *Current Biology* 27.3, R113–R115.

Sane, Sanjay P (2003). "The aerodynamics of insect flight". In: *Journal of experimental biology* 206.23, pp. 4191–4208.

– (2017). "Eppur si vola (and yet it flies)". In: *Journal of Experimental Biology* 220.4, pp. 514–516.

Scheffer, Louis K et al. (2020). "A connectome and analysis of the adult Drosophila central brain". In: *Elife* 9.

Schnell et al. (2010). "Processing of horizontal optic flow in three visual interneurons of the Drosophila brain". In: *Journal of neurophysiology* 103.3, pp. 1646–1657.

Seifert, Jost (2012). "A review of the Magnus effect in aeronautics". In: *Progress in Aerospace Sciences* 55, pp. 17–45.

Sotavalta, Olavi (1953). "Recordings of high wing-stroke and thoracic vibration frequency in some midges". In: *The Biological Bulletin* 104.3, pp. 439–444.

Stanford Artificial Intelligence Laboratory et al. (May 23, 2018). *Robotic Operating System*. Version ROS Melodic Morenia. URL: `https://www.ros.org`.

Stratonovich, Ruslan L (1959). "On the theory of optimal non-linear filtering of random functions". In: *Theory of Probability and its Applications* 4, pp. 223–225.

Suver, Marie P et al. (2016). "An array of descending visual interneurons encoding self-motion in Drosophila". In: *Journal of Neuroscience* 36.46, pp. 11768–11780.

Taha, Haithem E, Muhammad R Hajj, and Philip S Beran (2014). "State-space representation of the unsteady aerodynamics of flapping flight". In: *Aerospace Science and Technology* 34, pp. 1–11.

Tammero, Lance F and Michael H Dickinson (2002). "The influence of visual landscape on the free flight behavior of the fruit fly Drosophila melanogaster". In: *Journal of Experimental Biology* 205.3, pp. 327–343.

Tang and Fang (2019). "Elucidating Excited-State Hydrogen-Bonding Dynamics and Proton Transfer inside Fluorescent Protein Calcium Biosensors". In: *Hydrogen-Bonding Research in Photochemistry, Photobiology, and Optoelectronic Materials*. World Scientific, pp. 55–91.

Taylor, Gavin J et al. (2013). "Vision and air flow combine to streamline flying honeybees". In: *Scientific reports* 3.1, pp. 1–11.

Taylor, Graham, Richard Bomphrey, and Jochem 't Hoen (2006). "Insect flight dynamics and control". In: *44th AIAA Aerospace Sciences Meeting and Exhibit*, p. 32.

Tu and Dickinson (1994). "Modulation of negative work output from a steering muscle of the blowfly Calliphora vicina". In: *The Journal of experimental biology* 192.1, pp. 207–224.

– (1996). "The control of wing kinematics by two steering muscles of the blowfly (Calliphora vicina)". In: *Journal of Comparative Physiology A* 178.6, pp. 813–830.

V7 (2022). *Activation Functions in Neural Networks [12 Types Use Cases]*. URL: https://www.v7labs.com/blog/neural-networks-activation-functions.

Veen, van, van Leeuwen, and Muijres (2019). "A chordwise offset of the wing-pitch axis enhances rotational aerodynamic forces on insect wings: a numerical study". In: *Journal of The Royal Society Interface* 16.155, p. 20190118.

– (2020). "Malaria mosquitoes use leg push-off forces to control body pitch during take-off". In: *Journal of Experimental Zoology Part A: Ecological and Integrative Physiology* 333.1, pp. 38–49.

Veen, van, van Leeuwen, van Oudheusden, et al. (2022). "The unsteady aerodynamics of insect wings with rotational stroke accelerations, a systematic numerical study". In: *Journal of Fluid Mechanics* 936.

Videler, Stamhuis, and Povel (2004). "Leading-edge vortex lifts swifts". In: *Science* 306.5703, pp. 1960–1962.

Vogel, Steven (1966). "Flight in Drosophila: I. Flight performance of tethered flies". In: *Journal of Experimental Biology* 44.3, pp. 567–578.

Wagner, H (1925). "Dynamischer auftrieb von tragflugeln". In: *Zeitschrift fuer Angewandte Mathematik und Mechanik*.

Walker, Schwyn, et al. (2014). "In vivo time-resolved microtomography reveals the mechanics of the blowfly flight motor". In: *PLoS biology* 12.3, e1001823.

Walker, Thomas, and Taylor (2012). "Operation of the alula as an indicator of gear change in hoverflies". In: *Journal of The Royal Society Interface* 9.71, pp. 1194–1207.

Weis-Fogh, Torkel (1973). “Quick estimates of flight fitness in hovering animals, including novel mechanisms for lift production”. In: *Journal of experimental Biology* 59.1, pp. 169–230.

Werlé, Henri (1975). “On the Flow of Fluids Made Visible”. In: *Leonardo* 8.4, pp. 329–331.

Whitehead et al. (2015). “Pitch perfect: how fruit flies control their body pitch angle”. In: *Journal of Experimental Biology* 218.21, pp. 3508–3519.

Williams and Williams (1943). “The flight muscles of Drosophila repleta”. In: *Journal of Morphology* 72.3, pp. 589–599.

Wisser, Alfred (1988). “Wing beat of Calliphora erythrocephala: turning axis and gearbox of the wing base (Insecta, Diptera)”. In: *Zoomorphology* 107.6, pp. 359–369.

Wisser and Nachtigall (1984). “Functional-morphological investigations on the flight muscles and their insertion points in the blowfly Calliphora erythrocephala (Insecta, Diptera)”. In: *Zoomorphology* 104.3, pp. 188–195.

Zabala et al. (2009). “Flight dynamics and control of evasive maneuvers: the fruit fly’s takeoff”. In: *IEEE Transactions on Biomedical Engineering* 56.9, pp. 2295–2298.

Zahn, Olivia et al. (2022). “Pruning deep neural networks generates a sparse, bio-inspired nonlinear controller for insect flight”. In: *PLoS computational biology* 18.9, e1010512.

Zanker, Johannes M (1988). “On the mechanism of speed and altitude control in Drosophila melanogaster”. In: *Physiological entomology* 13.3, pp. 351–361.

Zanker, Johannes M, Martin Egelhaaf, and Anne-Kathrin Warzecha (1991). “On the coordination of motor output during visual flight control of flies”. In: *Journal of Comparative Physiology A* 169.2, pp. 127–134.

Appendix A

WING MOTION RECONSTRUCTION BY CNN REGRESSION

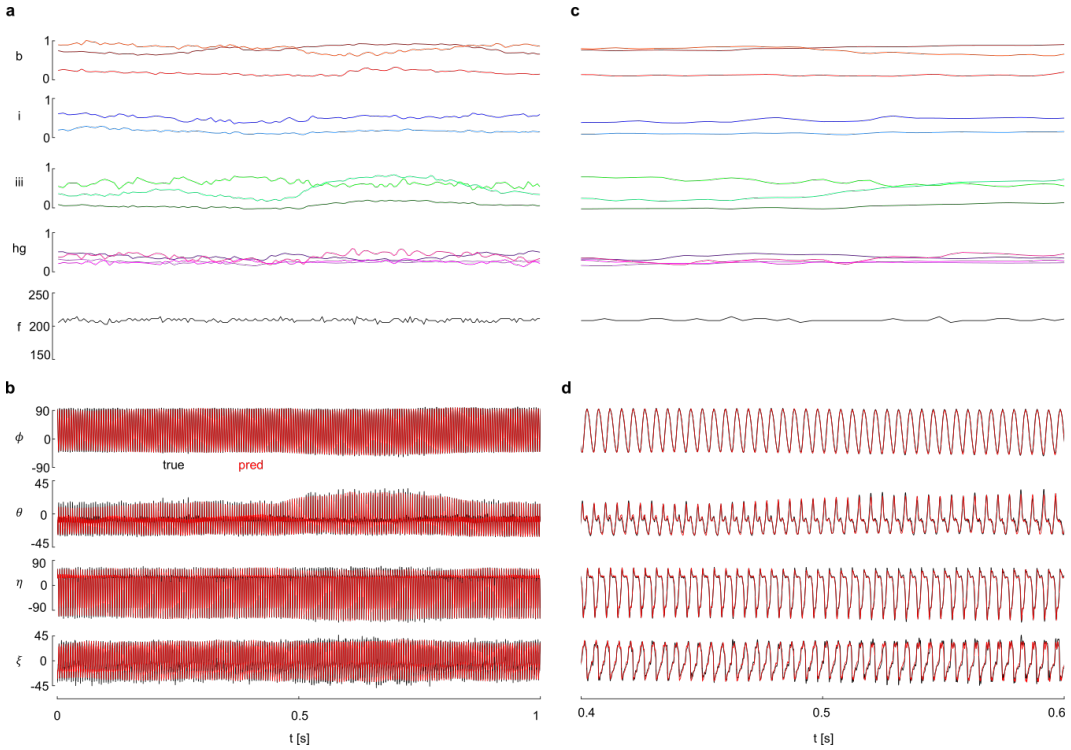

Figure A.1: Muscle activity and (predicted) wing motion during a high-speed video sequence. A: Muscle activity and wingbeat frequency. B: Tracked (black, true) and predicted (red, pred) wing motion. C: Close-up between 0.4 and 0.6 seconds of A. D: Close-up of B.

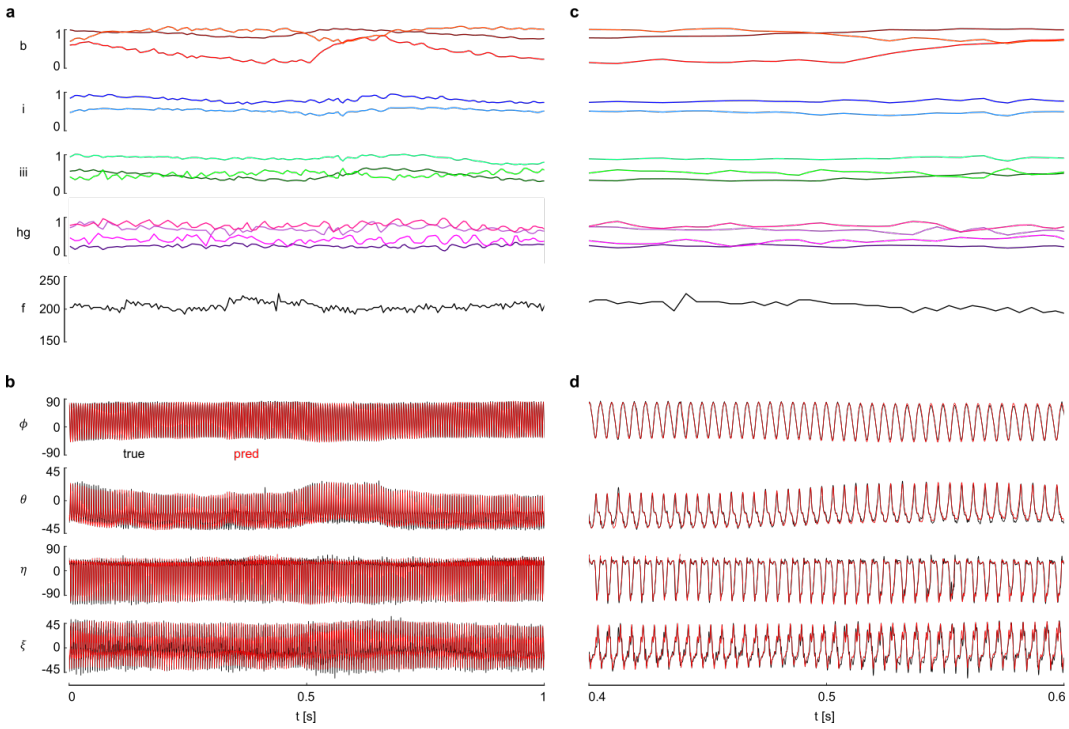

Figure A.2: Muscle activity and (predicted) wing motion during a high-speed video sequence. A: Muscle activity and wingbeat frequency. B: Tracked (black, true) and predicted (red, pred) wing motion. C: Close-up between 0.4 and 0.6 seconds of A. D: Close-up of B.

Appendix B

MPC FREE FLIGHT MANEUVERS

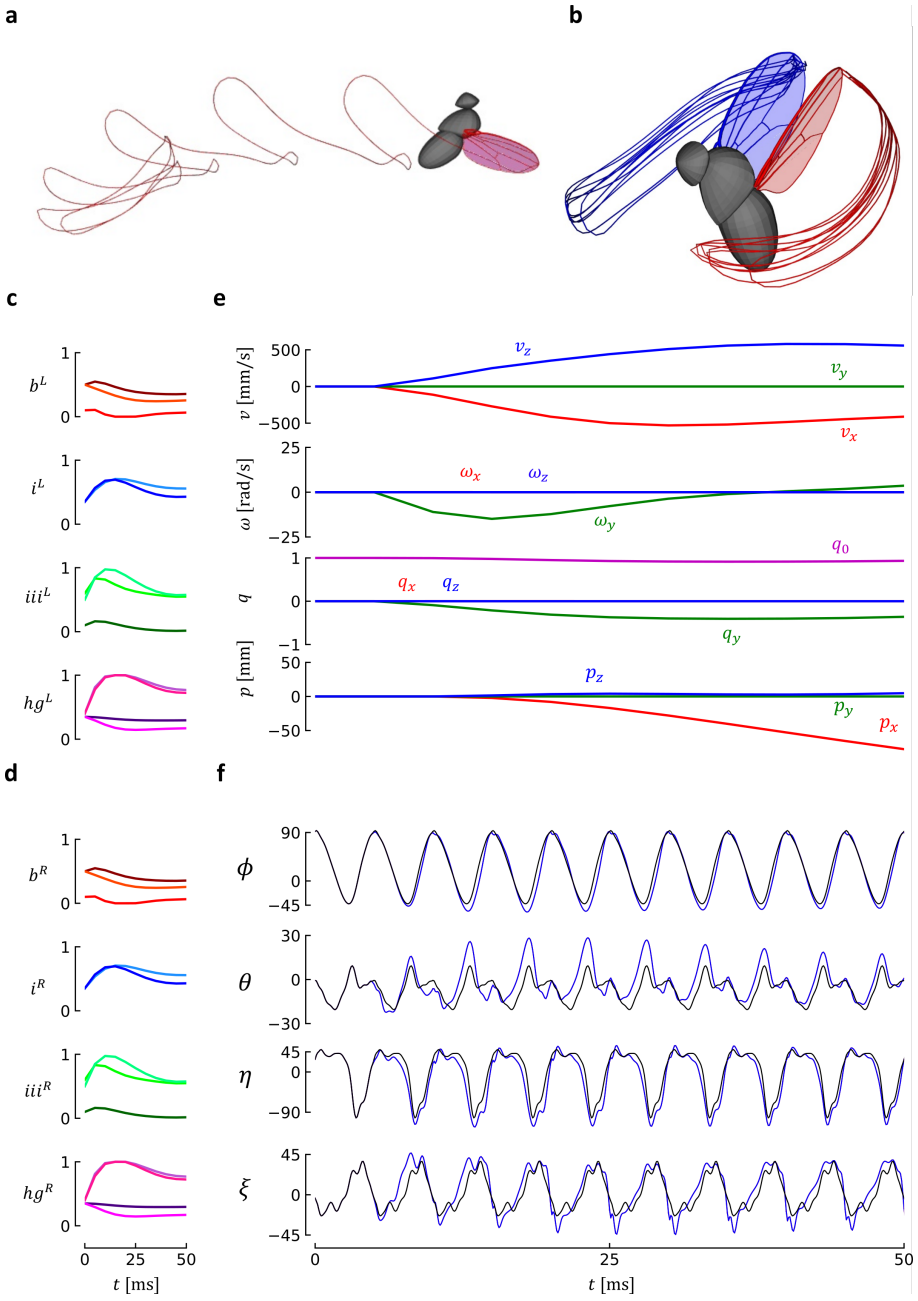

Figure B.1: Backward velocity. A: Side view at wingbeat 6 of the maneuver, wingtip traces are shown in red and blue. B: Wingtip traces at wingbeat 6 for stationary body state. C&D: Left and right steering muscle activity during the maneuver. E: Body state during the maneuver. F: Left (red) and right (blue) wing kinematics predicted from steering muscle activity in CD. The baseline wing kinematics are shown in black for comparison.

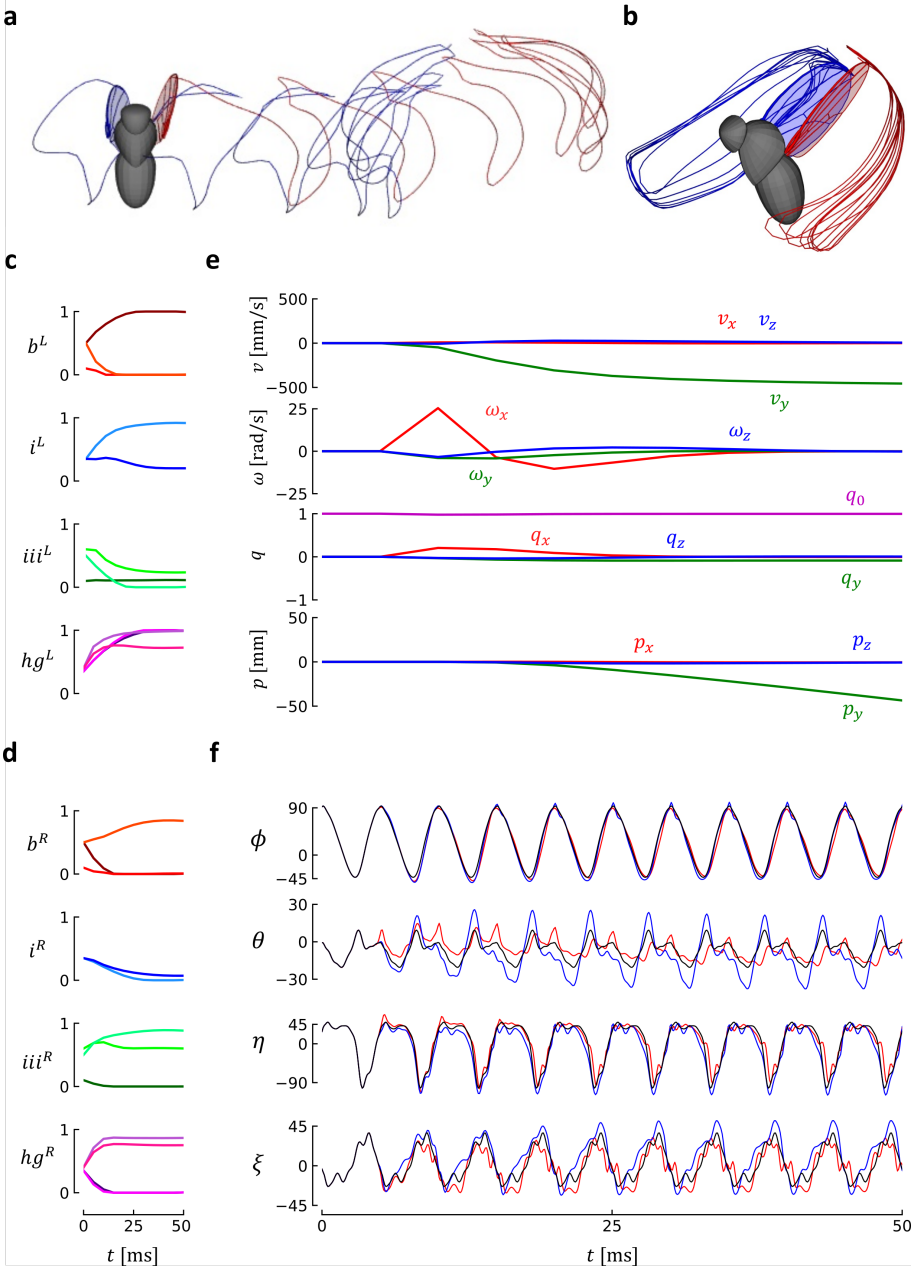

Figure B.2: Sideward velocity. A: Frontal view at wingbeat 7 of the maneuver, wingtip traces are shown in red and blue. B: Wingtip traces at wingbeat 7 for stationary body state. C&D: Left and right steering muscle activity during the maneuver. E: Body state during the maneuver. F: Left (red) and right (blue) wing kinematics predicted from steering muscle activity in CD. The baseline wing kinematics are shown in black for comparison.

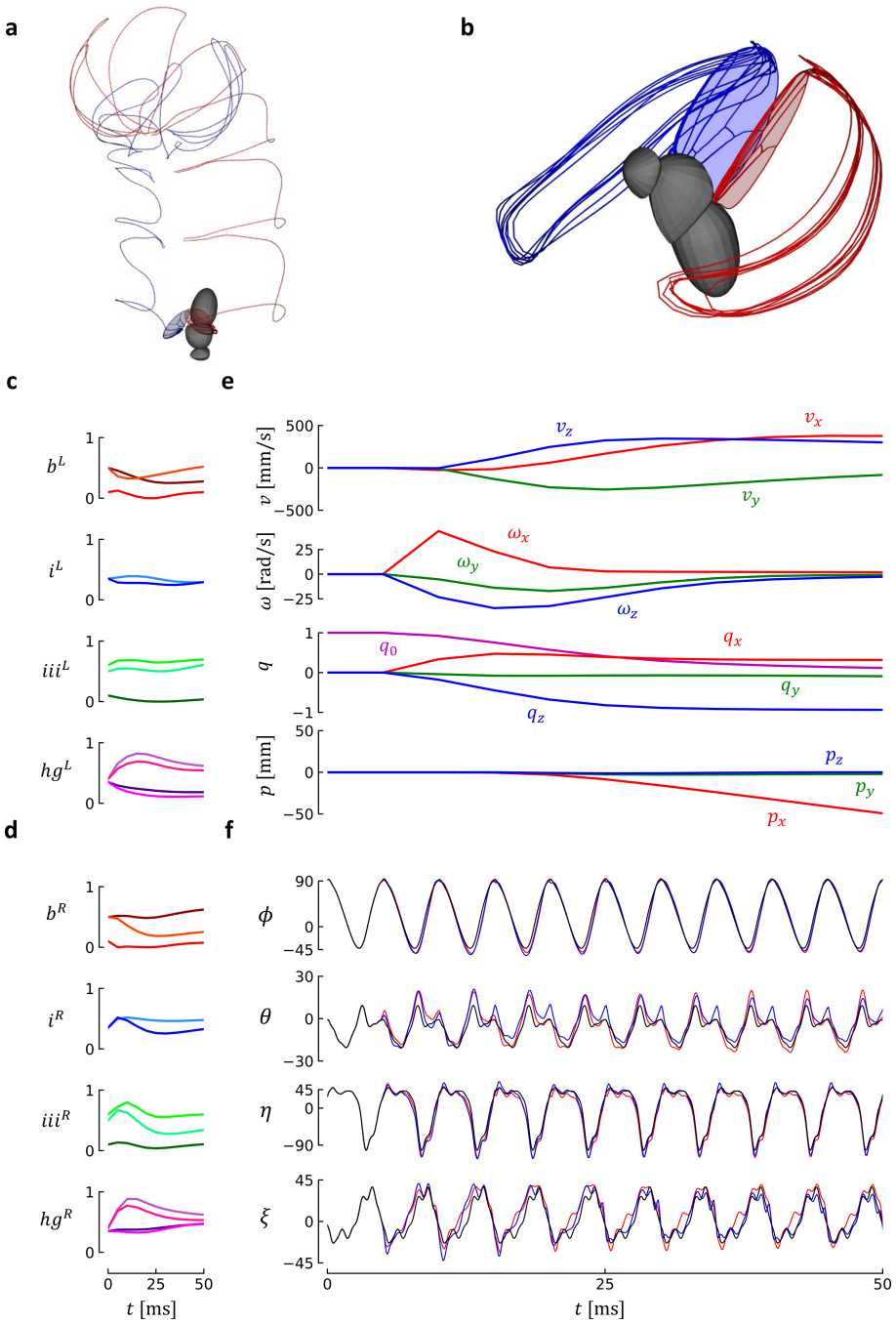

Figure B.3: Escape maneuver for frontal loom stimulus. A: Top view at wingbeat 7 of the maneuver, wingtip traces are shown in red and blue. B: Wingtip traces at wingbeat 7 for stationary body state. C&D: Left and right steering muscle activity during the maneuver. E: Body state during the maneuver. F: Left (red) and right (blue) wing kinematics predicted from steering muscle activity in CD. The baseline wing kinematics are shown in black for comparison.

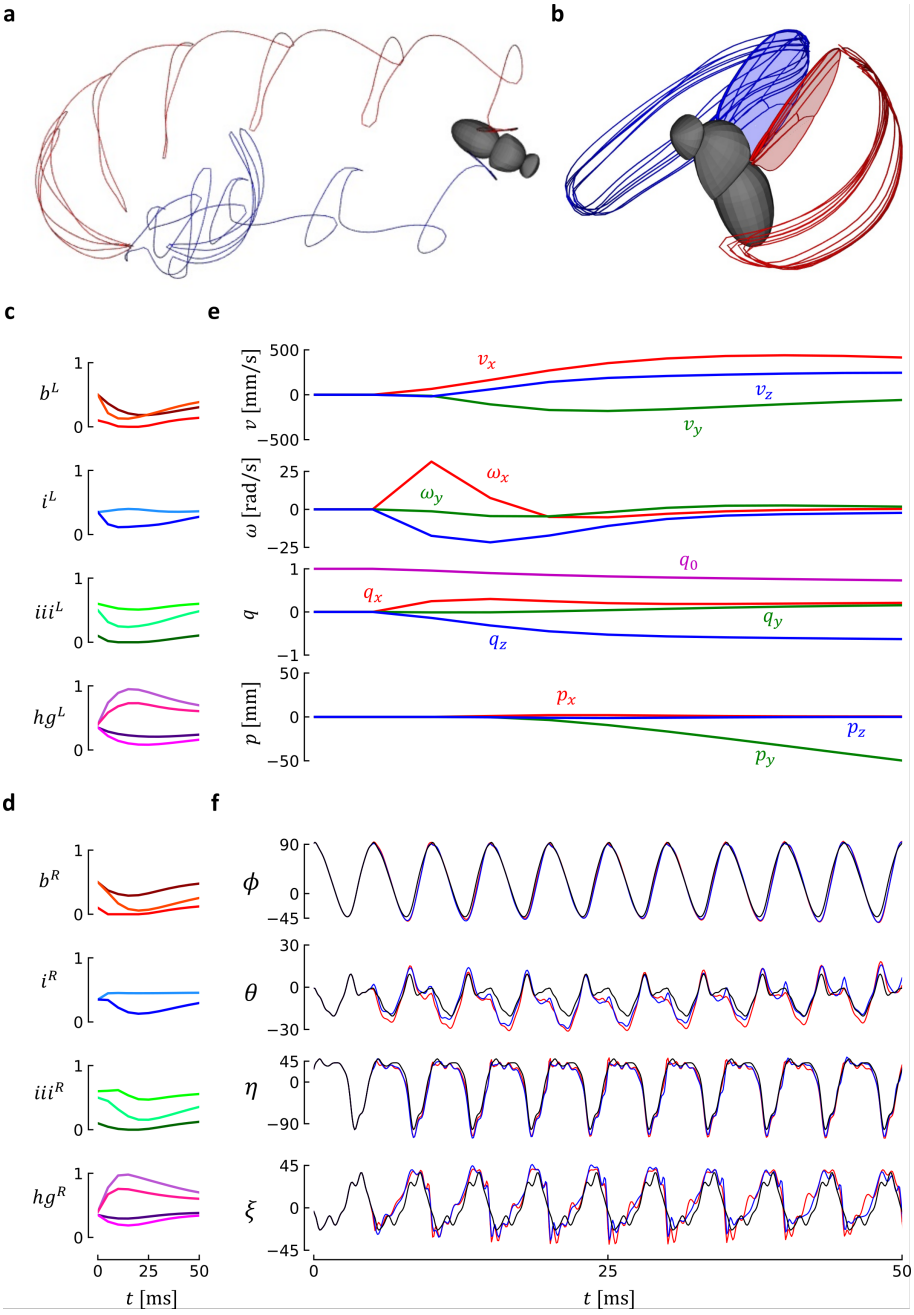

Figure B.4: Escape maneuver for left loom stimulus. A: Top view at wingbeat 7 of the maneuver, wingtip traces are shown in red and blue. B: Wingtip traces at wingbeat 7 for stationary body state. C&D: Left and right steering muscle activity during the maneuver. E: Body state during the maneuver. F: Left (red) and right (blue) wing kinematics predicted from steering muscle activity in CD. The baseline wing kinematics are shown in black for comparison.