

University of Alabama in Huntsville

LOUIS

Theses

UAH Electronic Theses and Dissertations

2015

Novel experimental method for studying trajectories and wing kinematics of freely flying butterflies

Jacob Taylor Cranford

Follow this and additional works at: <https://louis.uah.edu/uah-theses>

Recommended Citation

Cranford, Jacob Taylor, "Novel experimental method for studying trajectories and wing kinematics of freely flying butterflies" (2015). *Theses*. 698.
<https://louis.uah.edu/uah-theses/698>

This Thesis is brought to you for free and open access by the UAH Electronic Theses and Dissertations at LOUIS. It has been accepted for inclusion in Theses by an authorized administrator of LOUIS.

**NOVEL EXPERIMENTAL METHOD FOR STUDYING
TRAJECTORIES AND WING KINEMATICS OF
FREELY FLYING BUTTERFLIES**

by

JACOB CRANFORD

A THESIS

**Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
in
The Department of Mechanical and Aerospace Engineering
to
The School of Graduate Studies
of
The University of Alabama in Huntsville**

**HUNTSVILLE, ALABAMA
2015**

In presenting this thesis in partial fulfillment of the requirements for a master's degree from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.




Jacob Cranford

8/14/2015
(Date)

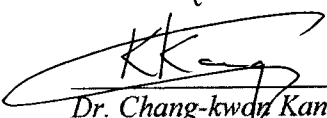
THESIS APPROVAL FORM

Submitted by Jacob Cranford in partial fulfillment of the requirements for the degree of Master of Science in Engineering in Mechanical Engineering and accepted on behalf of the Faculty of the School of Graduate Studies by the thesis committee.

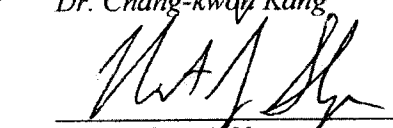
We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate of the work described in this thesis. We further certify that we have reviewed the thesis manuscript and approve it in partial fulfillment of the requirements for the degree of Master of Science in Engineering in Mechanical Engineering.



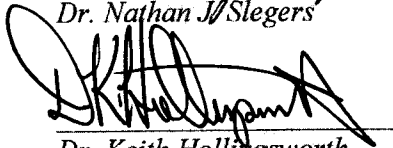
Dr. *Brian Landrum* (Date) 8/12/15 Committee Chair



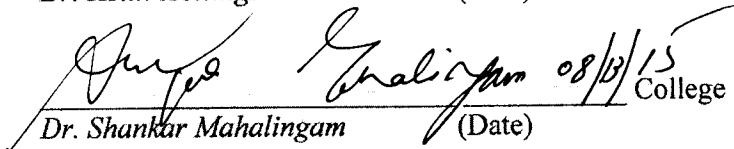
Dr. *Chang-kwon Kang* (Date) 7/29/2015



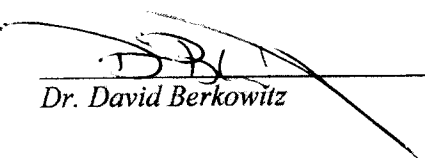
Dr. *Nathan J. Slegers* (Date) 7/29/2015



Dr. *Keith Hollingsworth* (Date) 8/12/15 Department Chair



Dr. *Shankar Mahalingam* (Date) 08/13/15 College Dean



Dr. *David Berkowitz* (Date) 10/1/15 Graduate Dean

ABSTRACT

School of Graduate Studies
The University of Alabama in Huntsville

Degree Masters of Science College/Dept. Engineering/Mechanical and
in Engineering Aerospace Engineering

Name of Candidate Jacob Cranford

Title A Novel Experimental Method for Studying Trajectories and Wing Kinematics of
Freely Flying Butterflies

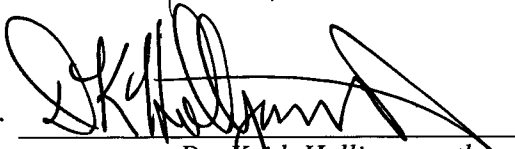
A novel experimental technique is presented for gathering data on the kinematics and trajectories of flapping insects. An optical tracking facility was used to record the free flight of the Monarch Butterfly (*Danaus plexippus*). The 5.7m×9.1m×3.0m capture volume allowed enough space to capture large numbers of sequential wing flaps. The system automatically tracked small reflective markers, which were modified to reduce the effects of additional mass on butterfly flight characteristics. This technique was used to record 2,000 flights of 86 different butterflies. A data analysis method is also presented that efficiently extracts information on flapping and body motion. A sample of data analyzed over 75 flights spanning 9 butterflies in a climbing trajectory is presented. Six flight characteristics - linear climbing rate, frequency and amplitude of flapping and body undulations, and the phase difference between them - and three dimensionless parameters - Reynolds number, Strouhal number, and reduced frequency - were derived from this data set. This experimental technique and data add to the understanding of biological flight and development of micro flapping robots.

Abstract Approval: Committee Chair

 8/12/15


Dr. D. Brian Landrum

Department Chair



Dr. Keith Hollingsworth

Graduate Dean



Dr. David Berkowitz

ACKNOWLEDGEMENTS

This thesis was only possible through the guidance, help and support from my advisors, friends and family. Through the process of this research over the past two years, I have learned more than just what is found in the contents of these pages. I have learned from many great mentors and friends throughout the process, and I would like to thank them all.

I would first like to acknowledge the National Science Foundation for providing funding for this project. It has been an honor to work with this prestigious group to develop technologies for the future. I am thankful for the opportunities that came from this collaboration.

Dr. Nathan Slegers, thank you for providing me with the opportunity to further my education while performing cutting edge research. I appreciate the guidance which was instrumental in helping me develop the initial experiments and analysis within this thesis.

I am Indebted to my advisor, Dr. Chang-Kwon Kang who was always active and interested in my research. This thesis would not have been possible without your guidance and support throughout the data analysis and writing portion of my time here. I am also grateful for the insightful comments and suggestions which really drove this research throughout even the hardest times.

My committee chair, Dr. David Brian Landrum, needs to be commended for seamlessly taking over his role in a difficult situation. I appreciated the long discussions we had which provided insight and motivation for research and life in

general. I am grateful for the extra opportunities that you helped me achieve, and all of the effort you put into ensuring that I would succeed.

My work at the Propulsion Research Center, and the friends that I made there, helped sustain me through even the most difficult times. Dr. Robert Frederick, thank you for the opportunity to work with you and your team, it was a priceless opportunity. Dr. David Lineberry, thank you for introducing me to the wonders of MATLAB, as well as always providing good advice and insight, no matter the problem. I appreciate the help and insight provided by Matthew Hitt whenever I had a question or needed correction. I would also like to thank you for your help in conducting some of the experiments.

To Dr. John Bennewitz who was a mentor and a friend from the very beginning of my research career, thank you for setting the bar high and always believing I would achieve it. I am grateful to have had the opportunity to work with you and learn all that I could from our work together. I appreciate the extra time you spent helping me learn to be a better researcher and person, and I will always remember all the great times we had.

To my colleagues who helped conduct my experiments, I appreciate your enthusiasm and companionship during the long hours of testing. Thank you Jasmine Conway, Dr. Shreyas Bidadi, James Bluman, Dan Jones, Raisa Chowdhury, and Amit Patel for the help that you provided.

Finally, I would like to thank my family for their support and willingness to listen throughout this entire process. Everything I have accomplished can be traced back to your love and support.

TABLE OF CONTENTS

	PAGE
1 Introduction	1
1.1 Motivation.....	1
1.2 Research Objectives.....	4
1.3 Thesis Outline	5
2 Literature Review.....	6
2.1 Lepidoptera	6
2.2 Monarch Butterfly (<i>Danaus plexipus</i>)	8
2.3 Unsteady Aerodynamic Mechanisms	9
2.3.1 Clap and Fling	10
2.3.2 Leading Edge Vortex	11
2.3.3 Wing Rotation.....	13
2.3.4 Wake Capture	14
2.4 Review of Experimental Work.....	16
2.4.1 Videography	16
2.4.2 Photogrammetry/Videogrammetry	17
2.5 Conclusions from Literature.....	18
3 Experimental Methods	19
3.1 Vicon Motion Capture System.....	19
3.2 Camera Calibration.....	23
3.3 Markers and Camera Placement	29
3.4 Post Processing in Nexus	34
3.5 Monarch Butterflies Handling Procedure	36
3.6 Flight Testing Procedure.....	41

4	Analysis	43
4.1	Data Export	43
4.2	Data Import	45
4.3	Data Analysis	45
4.3.1	Flapping Angle Calculation.....	45
4.3.2	Flapping and Trajectory of an Entire Flight	47
4.3.3	Graphical User Interface	48
4.3.4	Disappearing Markers	49
4.3.5	Cubic Spline Interpolation.....	51
4.3.6	Flapping Characteristics	52
4.3.7	Detrending of Vertical Trajectory.....	54
4.3.8	Vertical Body Oscillations	56
4.4	Results and Discussion	56
5	Conclusions	62
5.1	Summary	62
5.2	Limitations, Consequences, and Implications	64
5.3	Future Work	65
	APPENDIX A: MATLAB Code.....	68
8	References	119

LIST OF FIGURES

Figure	Page
Figure 1-1. Examples of MAVs. a) Rigid fixed wing design [4]; b) flexible fixed wing design [5]; c) Hornet 2-b (Prox Dynamics) [9]; d) Nano Hummingbird [6,10]; e) DeIFly flapping wing MAV [7,11]; f) Harvard Robobee [8,12].....	2
Figure 2-1. Diagram of the clap (a-c) and fling (d-f) mechanisms used by some insects to generate large lift coefficients. The black lines indicate fluid flow, dark blue arrows show induced velocity. Light blue arrows indicate net forces acting on the airfoil. Image reproduced with permission from [17]......	11
Figure 2-2. Schematic of the leading edge example adopted from [17]. a) Suction force produced by sharp diversion of flow about blunt airfoil which acts in the direction of the chord. b) The diversion of flow for the thin airfoil is caused by the leading edge vortex. The suction force augments the normal force, enhancing the lift and drag of the wing. Image reproduced with permission from [17]......	13
Figure 2-3. Wing wake interactions. a) The flow around the airfoil contains a leading edge vortex at somewhat modest angle of attack. b) As the wing approaches stroke reversal, the wing rotates about the spanwise axis, shedding a leading and trailing edge vortex into the wake. The shed vortices induce a jet between them c) The wing reverses direction and continues to rotate. d) The leading edge encounters the vortex from the upstroke and the jet of induced flow, increasing the lift. f) The wing continues the downstroke. Image reproduced with permission from [38].	15
Figure 3-1. ATOM lab capture volume [52]. a) Actual image of capture volume; b) Virtual capture volume environment in Nexus with three-dimensional positions of the motion tracking cameras; c) Top view of the ATOM Lab capture volume along with the camera positions marked as red boxes.	20
Figure 3-2. Vicon T40s camera with strobe on. Image source: svga.ru.	21
Figure 3-3. Demonstration of the onboard processing of each camera. a) The marker is identified as a grouping of pixels; b) The pixels are then represented as a best fit circle and the centroid is calculated; c) Only the radius and centroid of the circle are recorded during motion capture.	22
Figure 3-4. Images of the capture volume in different states of calibration. a) The cameras are randomly spaced and not representative of the physical capture volume. b) The location of each camera is known in relation to every other camera, but the coordinate system is not representative of the physical capture volume. c) The origin of the physical capture volume has been defined. The virtual capture volume is representative of the physical capture volume.	24

Figure 3-5. Calibration tool with the coordinate frame that is recognized by the cameras during the process of setting the volume origin.....25

Figure 3-6. Results of camera masking on an image from camera 12 (see Figure 3-1). a) The two markers recognized are actually a collection of markers which correspond to the strobes of cameras 1 and 2. b) The markers have been masked and no data will be collected from those pixels.26

Figure 3-7. Image from all 22 cameras during calibration. The green tabs located at the bottom right hand corner of some frames indicate that less than 850 wand counts have been recorded. The color of the tabs indicates how close to this limit each camera is, such as the tab for camera 3 is a darker green color indicating it is close to the 850 mark. The yellow-green color of the tabs seen for camera 21 indicates that the wand count is further away from the 850 limit.28

Figure 3-8. Demonstration of the difference between a) a good calibration with relatively even distribution of wand images and b) a poor calibration with a clustered distribution of wand images, leaving areas with no wand images.29

Figure 3-9. Markers that are used in the ATOM lab for motion tracking studies. Marker #5 is used for all butterfly testing and is a flat tape while the other markers are hemispherical.30

Figure 3-10. Depiction of resolution tests conducted to characterize effects of distance and angle on the quality of the marker image. An image of the collection of markers was taken from cameras 2 and 17 for each height increment, h.31

Figure 3-11. Comparison of the resolution for each size marker recorded from camera 17, (identified in Figure 3-1c) which was directly overhead of the markers so that the viewing angle does not change as height is increased. The images were taken from heights of a) 0h b) 2h c) 5h d) 8h and e) 12h.....32

Figure 3-12. Comparison of resolution from Camera 2, highlighted in Figure 3-1 c, which provided an angled view of the markers. As the height of the markers was increased, the angle between the camera and markers increased. The images were taken from heights of a) 0h b) 2h c) 4h d) 6h and e) 8h.32

Figure 3-13. Marker locations on the a) dorsal and b) ventral side of the butterfly.....34

Figure 3-14. Demonstration of the loss of markers during data capture and refitting the subject which consists of an orange head marker, blue left wing marker, a green right wing marker and a gold rear wing marker. a) First, all markers are represented, the subject is denoted by the colored markers. b) As the butterfly wings close, the yellow rear wing marker is lost. c) as the wings continue to close, the rear wing marker appears unfitted to the subject, while the blue right wing marker disappeared. d) The user is manually fitting the subject to the rear wing marker, e) then the right wing marker appears unfitted to the subject and is manually added. f) Once again all markers are present and subject is fully fitted.....36

Figure 3-15. Handling as well as placing markers on the butterfly. The images demonstrate accessing a) the ventral side of the forewing, b) the dorsal side of both wings as well as c) the head marker. Note that the butterflies were held as close to the body as possible, which was both due to an increase in robustness of the wings near the body as well as mitigating the effects of accidental scale removal.....38

Figure 3-16. Comparison of a) male and b) female monarch butterflies. The scent marks (indicated by white circles) which are present in a, but absent in b gives clear indication about each butterfly's gender.39

Figure 3-17. Butterflies sunning in incandescent lamp light in the terrarium. In the bottom of the terrarium a butterfly is feeding in the bowl containing Gatorade soaked cotton balls. The Citizen CX 120 scales used for weighing the butterflies is next to the terrarium.39

Figure 3-18. Comparison of butterfly a) with scales and b) without scales. Note the transparency of the wings of the butterfly without scales. This comparison is presented of two different butterflies.....40

Figure 4-1. Example data from flight 1139 of butterfly 46 exported from Nexus into a CSV file, viewed in Microsoft Excel.44

Figure 4-2. Visualization of parameters used in analysis. a) Markers and the vectors in Nexus. b) Butterfly in flight with important parameters highlighted.46

Figure 4-3. Raw flapping and altitude data from flight 1139 of Butterfly 46. a) Flapping angle of the two forewings where $\gamma=0^\circ$ at the end of the upstroke. b) Altitude of head marker which shows three different trajectories (1,2,3). Trajectories 1 and 2 show climbing flight while trajectory 3 is a descending trajectory corresponding to a glide in part a).48

Figure 4-4. Plunging and vertical trajectory data for the two distinct climbing trajectories seen in Figure 4-3. a) Plunging angle recorded during frames 340 - 440. b) Vertical trajectory recorded during frames 340 - 440. c) Plunging angle recorded during frames 480 - 560. d) Vertical trajectory recorded during frames 480 - 560.50

Figure 4-5. Plot of two interpolation methods which were investigated to replace data that was lost by disappearing markers.....52

Figure 4-6. Flapping angle with the peaks identified using the averages of the extrema. The red diamonds represent local maxima while the red circles represent local minima.54

Figure 4-7. Example of mean trend extraction. a) Mean trend compared to the actual data, b) differentiated mean trend against recorded velocity data.....55

Figure 4-8. Body undulations calculated by subtracting the mean trend from the vertical position data.....56

Figure 4-9. Time signal of the flapping and body oscillations normalized using the maximum value of each.....59

LIST OF TABLES

Table	Page
Table 3-1. Properties of each marker shown in Figure 3-9.....	30
Table 3-2. Distance between Marker 5 and both cameras and the angle between the marker and camera 2. These distances and angles correspond to the images seen in Figure 3-11 and Figure 3-12.	33
Table 4-1. Physical characteristics of the 9 butterflies compared in this thesis.....	57
Table 4-2. Mean and standard deviation of flight characteristics of 9 butterflies over 75 flights. The frequency of the body oscillations were found to be the same as that of the flapping frequency, and therefore they are listed as one entity.	58
Table 4-3. Mean and standard deviation of non-dimensional parameters of 9 butterflies over 75 flights.	60

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOL	DEFINITION	UNITS
f	Frequency of flapping motion	[Hz]
k	Reduced frequency	[1]
L_{ref}	Length from root to tip of forewing	[mm]
\vec{P}_{left}	Position vector from head to left wing	[mm]
\vec{P}_{right}	Position vector from head to right wing	[mm]
\vec{P}_{rear}	Position vector from head to rear wing	[mm]
Re	Reynolds number	[1]
s	(Span - 1)/2	[1]
$Span$	Moving average filter parameter	[1]
St	Strouhal number	[1]
U_{ref}	Magnitude of velocity of butterfly	[mm/s]
X	Data into moving average filter	[mm]
Y	Data out of moving average filter	[mm]
γ	Instantaneous flapping angle	[deg]
ϕ	Phase angle between flapping and body motions	[deg]
ν	Kinematic viscosity of air	[mm ² /s]
$\bar{\omega}$	Complex frequency	[Hz]

Chapter 1

Introduction

1.1 Motivation

Micro air vehicles (MAVs) and nano air vehicles (NAVs) are a rapidly developing field of flight research. These categories encompass aircraft with a maximum dimension of 15 cm and a mass no more than 100 g (MAV) or 20 g (NAV) [1]. The desired duration of flight for each of these vehicles is 20 minutes or longer. Applications range from military intelligence, surveillance and reconnaissance to disaster search and rescue missions [1,2]. These missions require the capability to maneuver in confined spaces such as indoors or urban areas, as well as deal with unsteady wind gusts during outdoor operation [3].

Figure 1-1 shows examples of MAVs that have been developed. The rigid fixed wing design built by Wu et al. [4] follows a traditional approach to designing MAVs. Ifju et al. [5] incorporated a novel flexible wing design. The Hornet 2-b developed by Prox Dynamics is a rotary design which resembles a full scale rotorcraft. The Nano Hummingbird developed by Keenon et al. [6] is a flapping wing design modeled after a

hummingbird. The DelFly [7] is an autonomous flapping wing vehicle. Finally the Robobee [8] is the first insect-scale flapping wing robot that weighs less than 100 mg.

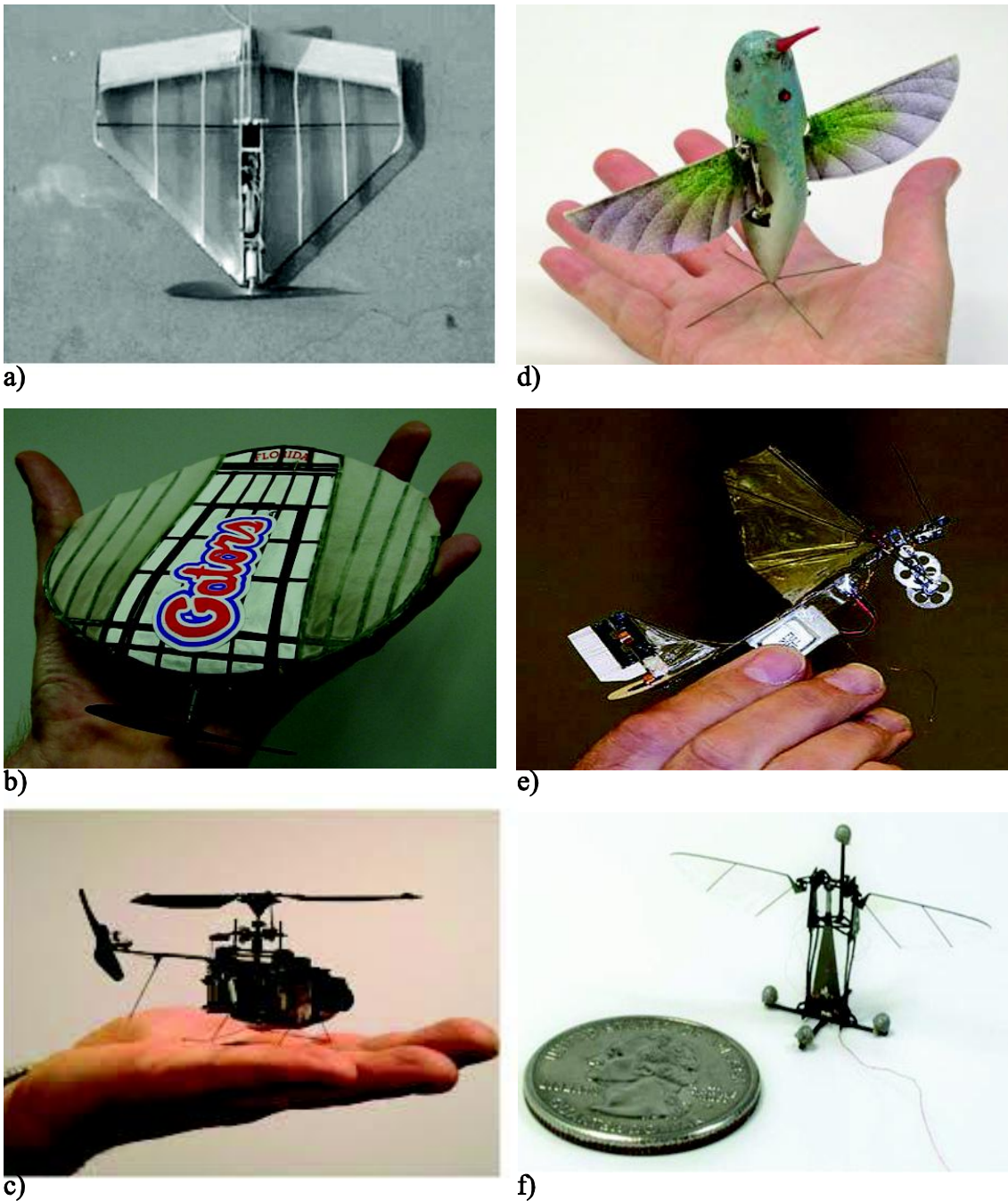


Figure 1-1. Examples of MAVs. a) Rigid fixed wing design [4]; b) flexible fixed wing design [5]; c) Hornet 2-b (Prox Dynamics) [9]; d) Nano Hummingbird [6,10]; e) DelFly flapping wing MAV [7,11]; f) Harvard Robobee [8,12].

Flight time is a significant issue that affects all of the MAV designs shown in Figure 1-1. The black hornet has the longest flight time of the MAVs shown, up to 25 minutes [13]. However this still falls in the bottom of the desired range. A large contributor to restricted flight time is the poor aerodynamic efficiency of conventional aircraft flying at low Reynolds numbers which produces a low lift to drag ratio [2,14]. This reduction in lift to drag ratio and inability to hover severely limit the capabilities of fixed wing MAV platforms. Rotary wing aircraft demonstrate high maneuverability which includes hovering and vertical takeoff and landing, but these also suffer from the reduction in lift to drag [2] and have relatively large power requirements for takeoff and hover [1]. A strong case for studying flapping flight can be made by looking at the flight performance of biological fliers, many of which operate at the same Reynolds number as MAVs. Insects in particular are noted by Petricca et al. [1] for their ability to produce high lift coefficients, execute vertical take offs or landings, as well as hover.

The flight of insects has inspired scientists and engineers for over 100 years [14,15]. In order to explain insect aerodynamics, numerous studies have explored insect flapping wing kinematics and unsteady aerodynamic mechanisms beyond the conventional stationary wing aerodynamics. Unsteady aerodynamic mechanisms such as clap and fling, leading edge vortices, rapid wing rotation and wing-wake interactions have been identified as lift enhancement techniques used by insects [3,14,16,17]. These techniques will be reviewed in Section 2.1. Methods for measuring kinematics and fluid flow, reviewed in Section 2.2, have advanced the understanding of aerodynamics of insects as well as providing strategies that can be applied to sensing and control of MAVs [14–16].

Butterflies present desirable flight characteristics such as maneuverability, efficiency and relatively simple flapping motions. With a large wing to body size ratio compared to other insects [18], butterflies have been noted for their rapid accelerations and evasive flight as well as the ability to maneuver in tight spaces [19,20]. It has been suggested that a hybrid of flapping and gliding flight could provide enhanced efficiency in birds [21] and butterflies [18]. The flapping motion of butterflies is also greatly simplified by restrictions of feathering created by closely coupled fore and hind wings [22]. This simplicity could be applied to development of similar scale MAVs using the same flight strategies.

The Monarch butterfly (*Danaus plexipus*) was the test subject chosen for this thesis. The migration of the Monarch is a spectacle in North America in the fall of every year. This migration extends up to 3,600 kilometers over the course of three months, which is the longest of any known insect [23]. Such a long migration suggests that Monarchs may have evolved into incredibly efficient fliers. Mimicry of the wing kinematics and trajectories of Monarchs could therefore prove invaluable to increasing the efficiency of MAV design.

1.2 Research Objectives

The objective of the study described in this thesis is to develop an effective method to gather statistically significant data on wing kinematics and trajectories of Monarch butterflies over large segments of free flight. A total of 22 high speed motion tracking cameras were used to measure and report the motion of specialized reflectors placed on the wings and body of the butterflies. The 5.7m × 9.1m × 3.0m capture volume provides enough space to let the butterflies fly unobstructed for a large sequence of flaps.

This research used a turnkey Vicon motion capture system. Data analysis was performed using Microsoft EXCEL and Mathwork's MATLAB. The large number of cameras and the semi-automated post processing Nexus software provide a unique opportunity to record and process large amounts of data relatively quickly. Over 2,000 untethered free flights of 86 different butterflies were recorded and some preliminary results are reported on the main characteristics of butterfly flight during climbing trajectories.

1.3 Thesis Outline

A review of the literature pertaining to unsteady aerodynamic mechanisms and experimental techniques for gathering wing kinematic data for Monarch butterflies can be found in Chapter 2. A description of the Vicon system, handling of the butterflies, and description of the experiments conducted are found in Chapter 3. A detailed look at the analysis procedure and a presentation of some initial results are found in Chapter 4. Finally, concluding remarks and recommendations of future work can be found in Chapter 5.

CHAPTER 2

Literature Review

This chapter provides a review of the unsteady aerodynamics and wing kinematic measurement state of the art. First, the morphology of insects of the order *Lepidoptera*, containing butterflies and moths, is reviewed, and then the Monarch butterfly, *Danaus plexippus*, is described in more detail. The unsteady aerodynamics of flapping insect flight are next discussed. Finally, methods for measuring the flight kinematics of insects using imaging techniques are presented.

2.1 Lepidoptera

Butterflies and moths make up the order of *Lepidoptera* in the class of insects. This is the second largest order of insects with more than 165,000 species identified to date [24]. *Lepidoptera* are identifiable by very small scales that cover the surface of two pairs of membranous wings [25]. The fore wing and hind wing are typically structurally coupled either by overlapping of the wings or a locking mechanism near the wing root [25]. The wings contain prominent venation which consists of tubular veins along which nerves are located and blood flows to the wing. This venation also provides additional

stiffness to the wing [26]. The wing scales give the Lepidoptera the bright coloring and complex patterning that are generally associated with them. There is preliminary data that suggest that the scales of Lepidoptera also increase the aerodynamic efficiency of flight [27]. Another defining trait of Lepidoptera is a "complete" metamorphosis in which the larval stage of the insect changes not only in appearance, but also in function. The larval stage exists to eat and store food so that the adult stage can focus almost exclusively on reproducing. In some cases, mature adults even lack the ability to feed [25]. Lepidoptera feed using a proboscis, which is a long hollow tube for the intake of fluids. Examples of foods which attract these insects include honey, blood, tears, decaying organic matter, sap and nectar from flowers. The nectar of flowers is generally visited by diurnal butterflies, which include the Monarch butterfly [25]. While butterflies nominally feed on natural substances, it has been demonstrated that they will also feed on concentrations of glucose, such as sugar water or Gatorade.

The wing beat frequency of butterflies is similar to the undulating motion of the body, suggesting that the flapping wing aerodynamics and flight dynamics are closely coupled to each other. Moreover, the butterfly wings significantly deform during flight. An accurate measurement of both wing kinematics and the body motion is crucial to understanding the dynamic flight stability and control of a flapping insect [15]. This information is typically neglected in simple flight dynamics models of most insects because wing flapping time is typically much faster than the motion of the entire flyer [14,28].

2.2 Monarch Butterfly (*Danaus plexipus*)

The Monarch is a commonly known butterfly in Northern America. The Monarch is of the family *Nymphalidae*, and subfamily *Danaini* [24]. The bright orange, black and white coloration on the wings make the Monarch a distinctly recognizable butterfly. Another feature of these insects is the extraordinary multi-generational migration pattern which is very prominent in populations that are found east of the Rocky Mountains. The small insects migrate during the colder winter months from their summer breeding grounds which reach up into Canada, down to the overwintering sites in the Neovolcanic belt in central Mexico [29]. The population west of the Rockies finds its overwintering spots in Southwest California. The migration of the eastern population spans a distance of up to 3,600 kilometers, the longest of any other known insect [23,30]. This large distance is covered over the span of days between late August and early December. Overwintering sights are located at approximately 10,000 ft in altitude. It is thought that a combination of the high altitude and tropical latitude results in a generally stable daily and seasonal thermal regime. Ambient temperature has been shown to affect the expenditure of butterfly lipid reserves which must last for approximately 90 days during overwintering [31]. It has been observed that body temperatures below -2° C can kill butterflies, while ambient temperatures above 20° C have been found to prematurely induce migration in some colonies [32]. The spring migration sees the butterflies return from Mexico to the Gulf Coast states to lay eggs and die [23]. The generation born in the spring continues the migration up to the summer breeding grounds. Depending on ambient temperatures, two or three generations remain in the north during the summer breeding months before

increasingly cold temperatures again drive the monarchs on their long migration south [23].

2.3 Unsteady Aerodynamic Mechanisms

The aerodynamic theories used to model the flow physics of fixed winged aircraft rely on steady-state analysis. Flow around flapping wings is unsteady due to the time-dependent wing motions and the nonlinear interactions with vortical structures in the flow field. Accurate closed-form analytic solutions that describe the unsteady aerodynamics of flapping wings have not been derived yet. Instead, many modern flapping wing aerodynamics analyses rely on numerical solutions of the Navier-Stokes equations or physical experiments of abstracted configurations [3].

Quasi-steady theory of flapping wings is an approximation enabling a quick analysis. This methodology discretizes the continuous system into a series of static conditions and uses standard steady analysis techniques without any attempt to model wing-wake interactions. A review of this method is provided by Ellington et al. [33] and Sane [17]. However, this method was called into question for small hovering flying insects [34]. Based on quasi-steady assumptions and measurements of wing kinematics, Dudley and Ellington [33] showed that steady state analysis was insufficient to explain even fast forward flight of a bumblebee. Further research into insect flight has resulted in the discovery of lift enhancing unsteady aerodynamic mechanisms that are used by a multitude of insects, such as clap and fling, leading-edge stall, and wake capture.

2.3.1 Clap and Fling

The first unsteady lift enhancing mechanism proposed was the "clap and fling" mechanism [3]. Weis-Fogh et al. [34] proposed this method to account for the large lift coefficients generated by the small Chalcid wasp *Encarsia formosa*. The mechanism consists of two distinct parts, first the clap and then the fling (see Figure 2-1). At the end of an upstroke, the leading edge of the wings of some insects come very close together and trailing edge vortices are shed into the wake. This is called the *clap*. Leading and trailing edge vortices developed during the upstroke are fully shed and flow is induced between them by their circulation to help the wings close quickly. As the trailing edges of the wings close, air flow induced by vortices as well as the trailing edges coming together provide a propulsive force. The *fling* mechanism is initiated by the separation of the wing leading edges. Air fills the low pressure gap left by the separating wings while the trailing edges are still kept together. Additional circulation resulting from the induced flow between the wings leads to higher lift. This circulation attaches to form leading edge vortices on each wing with equal magnitude, but opposite direction. In a slight variation of the clap and fling mechanism, larger and more deformable wings appear to "peel" as opposed to fling [35]. The clap and peel mechanism was visualized by Srygley and Thomas [36] in free flying Red-Admiral butterflies using smoke-wire and high speed digital images.

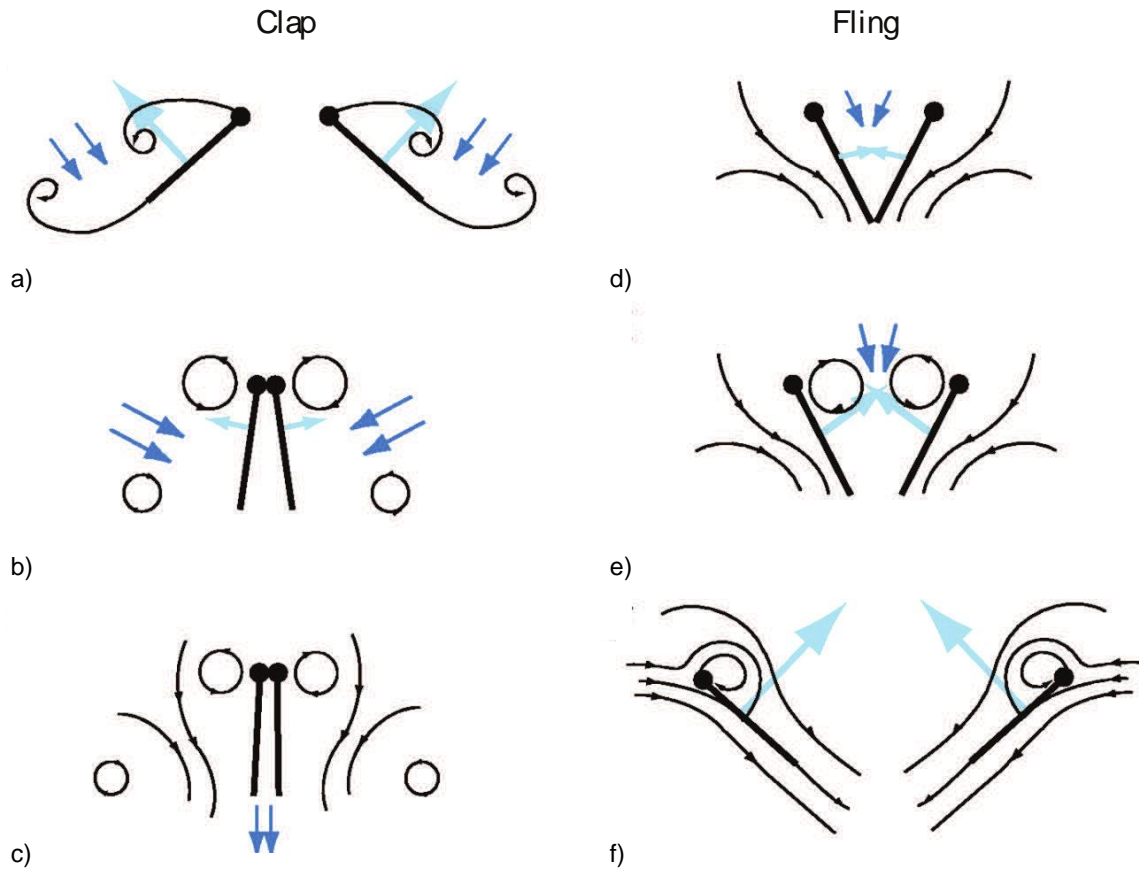


Figure 2-1. Diagram of the clap (a-c) and fling (d-f) mechanisms used by some insects to generate large lift coefficients. The black lines indicate fluid flow, dark blue arrows show induced velocity. Light blue arrows indicate net forces acting on the airfoil. Image reproduced with permission from [17].

2.3.2 Leading Edge Vortex

Ellington et al. [37] discovered a high lift mechanism of an attached leading edge vortex through flow visualizations around the wings of a Hawkmoth, as well as a mechanical flapper. A Leading edge vortex provides a mechanism to delay stall and augment the production of aerodynamic forces during translating of flapping wings. The lift enhancing effects of a leading edge vortex is analogous to the suction force that is present at the leading edge of a blunt airfoil. As shown in Figure 2-2, the flow at the

leading edge of a blunt airfoil bends sharply over the upper surface, creating a region of high flow velocity and low pressure. This region of low pressure causes the so called suction force that acts approximately parallel to the chord. In general, insect wings are better represented by thin airfoils. As air flows over thin airfoils at an angle of attack it separates immediately after the upper leading edge, creating a vortex. This vortex creates a diversion of flow and suction force similar to that seen in the blunt airfoil example, but with the resultant force acting perpendicular to the chord. This force acts to increase both the lift and drag on the wing. The main characteristics of leading edge vortices have been shown to vary with changes in Reynolds number, Strouhal number, wing flexibility and flapping kinematics [3,14,36]. Leading edge vortices have been suggested as the single most important feature of flows around insect wings, as well as the forces that they create [17].

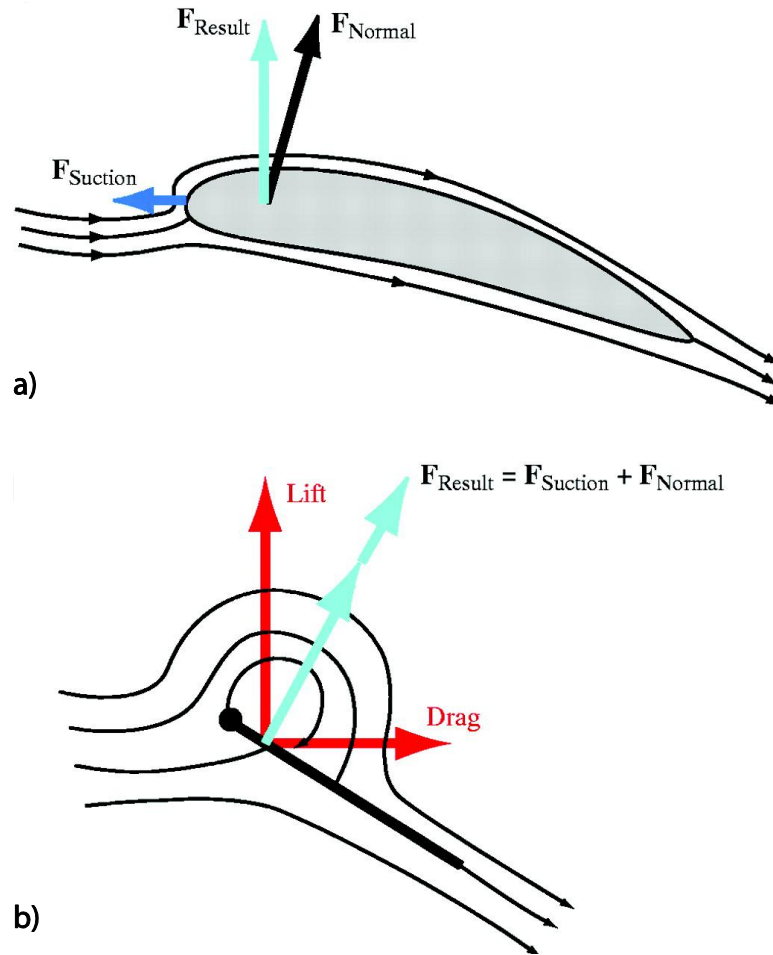


Figure 2-2. Schematic of the leading edge example adopted from [17]. a) Suction force produced by sharp diversion of flow about blunt airfoil which acts in the direction of the chord. b) The diversion of flow for the thin airfoil is caused by the leading edge vortex. The suction force augments the normal force, enhancing the lift and drag of the wing. Image reproduced with permission from [17].

2.3.3 Wing Rotation

Dickinson et al. [38] used an experimental study to show that rapid wing rotations that advance the wing strokes resulted in positive lift enhancement. In the study, a dynamically scaled mechanical flapping wing was fabricated and used to determine forces generated by varying wing kinematics. Two peaks in aerodynamic forces were recorded at the beginning and end of each stroke, corresponding to a pronation and

supination, respectively. The force associated with the first peak was explained as an effect from rotational circulation, which is also referred to as the *Kramer effect* [14]. The Kramer effect occurs when an airfoil is rotating from low angle of attack to high, causing lift coefficients above the steady flow stall value [14]. As the wing rotates rapidly, the flow over the trailing edge deviates from the Kutta condition due to the viscosity of the fluid which resists shear. As the flow attempts to re-establish the Kutta condition, circulation is induced, generating lift [17,38].

2.3.4 Wake Capture

Wake capture is another unsteady aerodynamic mechanism that was observed by Dickinson et al. [38]. The second peak in force occurred at the beginning of each stroke as the wings were reversing direction while rotating about the spanwise axis, shedding both the leading and trailing edge vortices (see Figure 2-3 (a-c)). The wing encounters the velocity field of the shed vortices at the beginning of the downstroke (see Figure 2-3 (d-f)), which can increase the effective fluid velocity at the start of the next stroke. The magnitude and direction of the augmentation of the forces on the wing due to the increase in velocity depends on the phase relationship between the translation and rotation. If the rotation precedes stroke reversal, the wing will encounter its own wake such that the lift is enhanced. This mechanism is recognized to affect hovering flight of many insects, but the nature of forward flight would cause the insect to advance past the wake of its wings and not interact with wakes from a previous wing stroke direction.

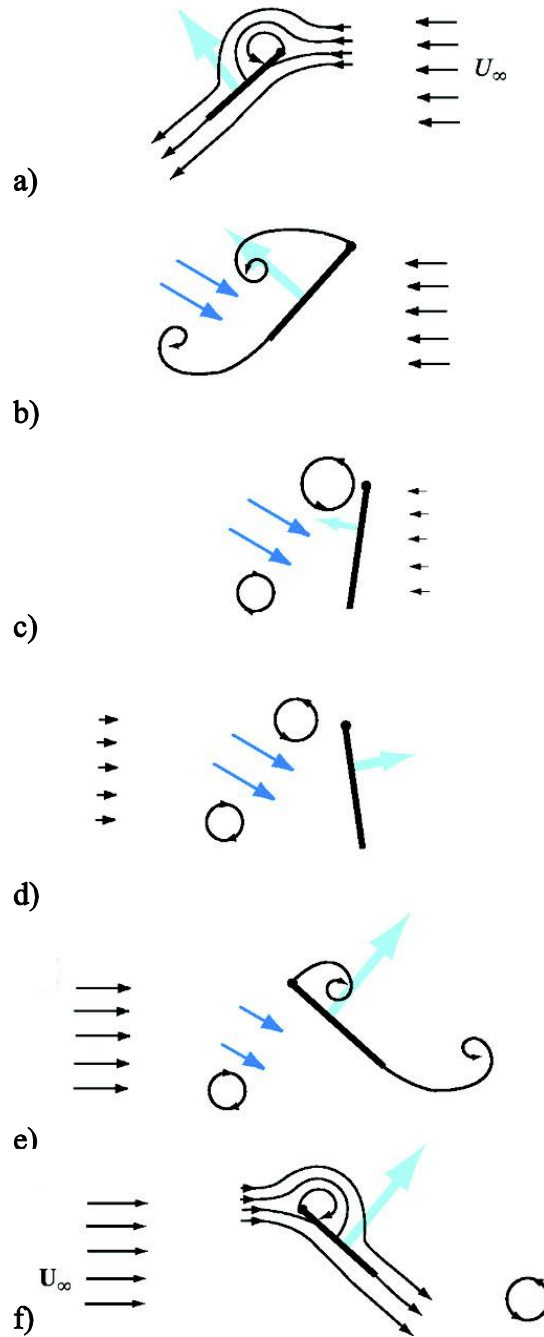


Figure 2-3. Wing wake interactions. a) The flow around the airfoil contains a leading edge vortex at somewhat modest angle of attack. b) As the wing approaches stroke reversal, the wing rotates about the spanwise axis, shedding a leading and trailing edge vortex into the wake. The shed vortices induce a jet between them c) The wing reverses direction and continues to rotate. d) The leading edge encounters the vortex from the upstroke and the jet of induced flow, increasing the lift. f) The wing continues the downstroke. Image reproduced with permission from [38].

2.4 Review of Experimental Work

Obtaining kinematic information of insects during free flight is crucial to studying the aerodynamics and dynamics of insect flight. Their small size makes direct measurements of the fluid flow and forces acting on flying insect difficult. Instead of direct measurements, some researchers have developed dynamically scaled mechanical flappers [37–39]. Another method is the use of unsteady Navier-stokes simulations [40,41]. These two methods for determining aerodynamic forces require a precise knowledge of the wing kinematics employed during insect flight. The high flapping frequency as well as small size of most insects also makes intrusive measurements difficult. Developments in photography, videography and photogrammetry offer researchers a non-intrusive measurement technique which can provide very good spatial and temporal resolution [17].

2.4.1 Videography

High speed videography is a valuable technique that has become prominent in the study of insect flight kinematics. High speed video was first utilized to film the tethered flight of a locust in order to determine the aerodynamic forces produced by the kinematics of the wings [16]. It was later demonstrated that tethering of locust could significantly reduce the wing beat frequency and, therefore, new methods for measuring kinematics and aerodynamics of insect flight were required [42]. New videography techniques were developed to capture kinematic data of specimens in hovering flight [16,34,43] and refine the models developed for tethered flight.

The main difficulty associated with using videography to record kinematic data of

a forward flying insect is that it must fly in a controlled trajectory through a relatively small volume that is in full camera view. An early method for controlling the trajectory of an insect utilized an ultraviolet light to take advantage of the optomotor response of a bumblebee to elicit forward flight within a wind tunnel. The flight of the bee was captured using a single high speed camera strategically placed so that the wing tip kinematics as well as angle of attack could be observed and extrapolated into three dimensions [43]. The use of a wind tunnel and an outside stimuli to produce forward flight was also used to investigate the flight of a Hawkmoth [42], as well as to visualize the various unsteady aerodynamic mechanisms that are used by a red admiral butterfly to generate lift at varying velocities [36]. Instead of using a wind tunnel, Tanaka and Shimoyana [44] captured the flight of a swallowtail butterfly as it navigated towards a light that was placed as a goal. One common aspect of each of these experiments is that the presented data was gathered using the two dimensional images from only one or two cameras.

2.4.2 Photogrammetry/Videogrammetry

Researchers have recently begun to use photogrammetric methods to track insect kinematics. These techniques utilize multiple synchronized cameras placed to record motion inside a three dimensional frame or 'capture volume' in order to estimate the coordinates of an object in three dimensional space. This technique has been used to record a variety of different phenomena. In typical studies of insect flight, multiple high speed cameras record three dimensional data inside a relatively small capture volume [45,46]. Data is recorded over relatively few wing beats, during either a free flight segment [19,47], a hovering flight segment [48], tethered flight [45], or using a subject

trained to fly in a specific capture volume [49,50]. The issue with the methods presented in these works is the limited size of the capture volume. Free flight like that presented by Lin et al. [19] and Ristroph et al. [47] is restricted to very short flapping sequences. The flapping sequences are not only kept short by the small capture volume, but also by the time required to process the data. Zheng et al. recorded data at 2000 fps with each frame requiring the user to identify 35 characteristic points. Walker et al. [48] automated the tracking of a tethered locust, but the free flying hover fly required the manual location of 15 wing locations in every image at 947 fps. The limited volume size and relatively short data captured limits the ability to capture large collections of consecutive flaps as well as the corresponding global trajectory

2.5 Conclusions from Literature

The state of the art identified by this literature review provided the basis for the development of the experiments and analysis outlined in the following chapters. It was determined that an experimental technique which could capture wing kinematics as well as trajectories for butterflies could prove invaluable to the design and development of MAVs. The study of unsteady aerodynamics of insect flight could also benefit from determining how and when the mechanisms reviewed in this chapter are employed. Finally, a method for quickly recording and calculating photogrammetric information in larger capture volumes and with more cameras can provide more statistically significant data.

CHAPTER 3

Experimental Methods

3.1 Vicon Motion Capture System

The experiments discussed in this thesis were conducted in the Autonomous Tracking and Optical Measurements (ATOM) laboratory at the University of Alabama in Huntsville (UAH). The ATOM lab is a motion capture facility which uses 33 VICON T40s cameras, an Mx Giganet, and Nexus software [51] to track reflective markers in three dimensional space. The maximum capture volume is $17.2\text{m} \times 9.1\text{m} \times 3.0\text{m}$. Twenty two cameras were placed inside a reduced capture volume of $5.7\text{m} \times 9.1\text{m} \times 3.0\text{m}$ shown in Figure 3-1. The relatively large size of even the reduced capture volume with respect to a butterfly enables data capture on unrestricted free flight trajectories.

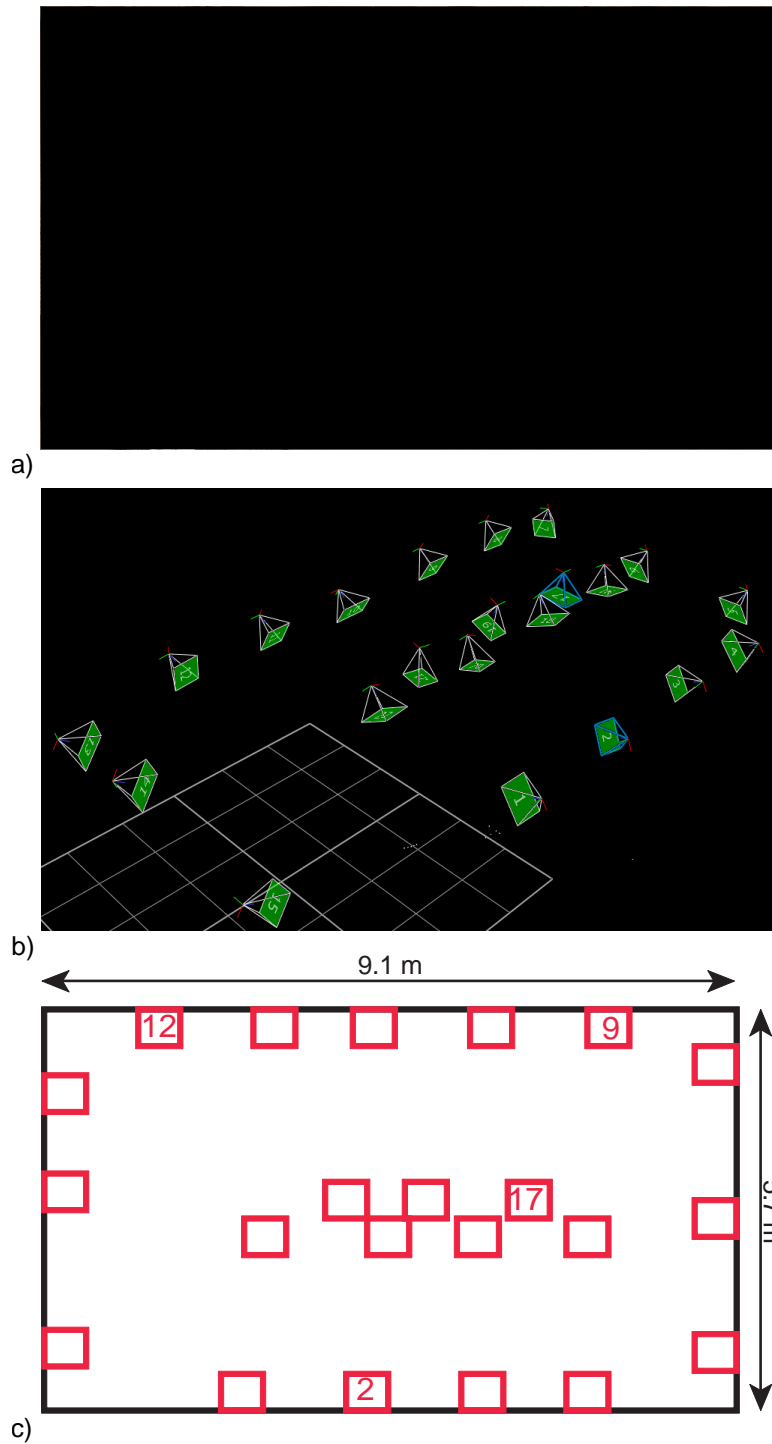


Figure 3-1. ATOM lab capture volume [52]. a) Actual image of capture volume; b) Virtual capture volume environment in Nexus with three-dimensional positions of the motion tracking cameras; c) Top view of the ATOM Lab capture volume along with the camera positions marked as red boxes.

The VICON T40s cameras (Figure 3-2) record at 515 fps at a full resolution of 4 megapixels. They are equipped with a near infrared (NIR) strobe, and visible light filter to allow for operation under a variety of lighting conditions. The cameras track reflective markers specifically designed for motion capture systems to efficiently reflect the NIR light.



Figure 3-2. Vicon T40s camera with strobe on. Image source: svga.ru.

Each camera contains an onboard processor which locates all markers recorded in a frame, calculates a circle fit and determines the centroid of the circle (see Figure 3-3). The location of the centroid and radius are the only data that are sent to the Mx Giganet during a motion capture test. The Mx Giganet calculates the three dimensional position of all markers seen simultaneously by two or more cameras and sends it to the Vicon workstation and into Nexus. Depending upon the user's requirements, the data can be transmitted to a monitor for real time monitoring, or recorded for post processing. The output environment is the virtual work space, which provides a three dimensional representation of the location of all cameras and markers detected (see Figure 3-1b).

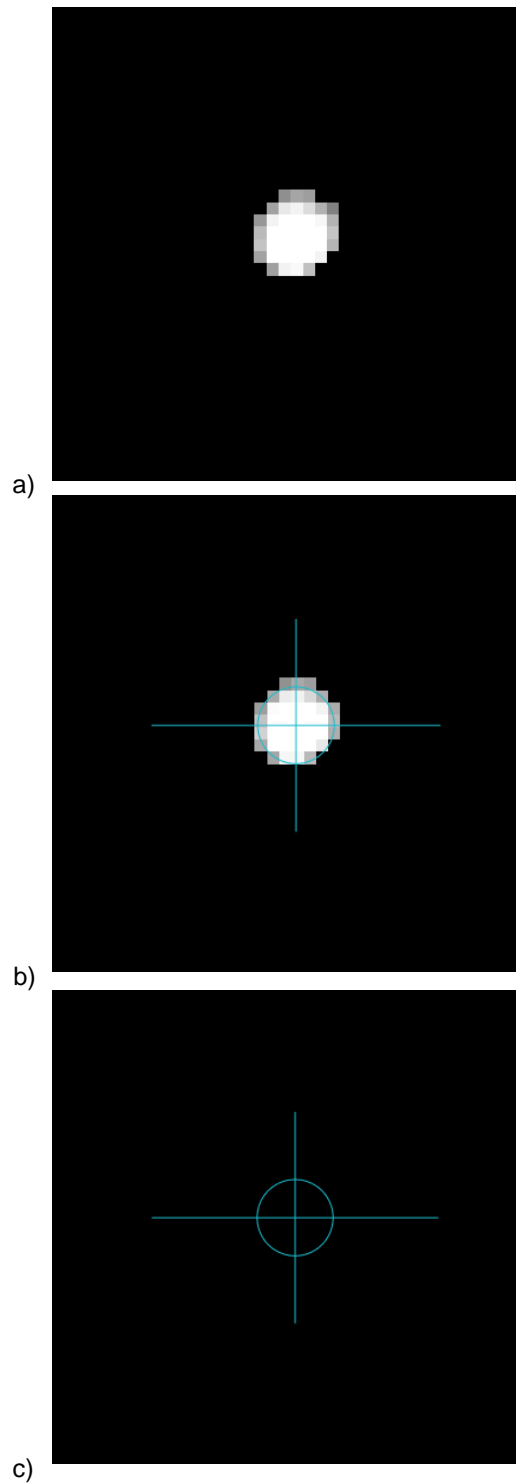


Figure 3-3. Demonstration of the onboard processing of each camera. a) The marker is identified as a grouping of pixels; b) The pixels are then represented as a best fit circle and the centroid is calculated; c) Only the radius and centroid of the circle are recorded during motion capture.

3.2 Camera Calibration

The system does not recognize camera locations in the capture volume when they are initially connected to Nexus. Each camera must be calibrated so that its location with respect to the other cameras is known, and location data can accurately be calculated from the images captured by two or more cameras. After the cameras are calibrated to each other, an origin and coordinate system must be defined so that the exported data has a global reference. Figure 3-4 demonstrates the state of the capture volume at the three stages of calibration.

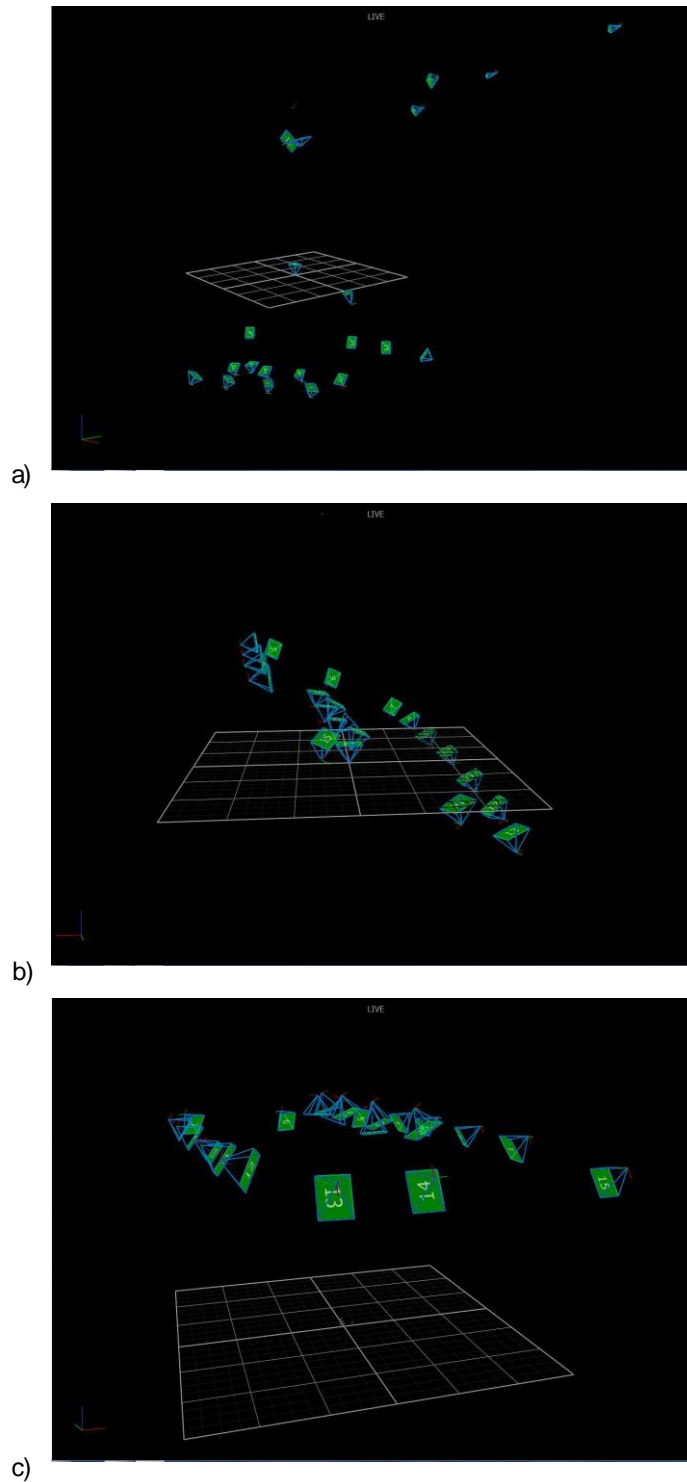


Figure 3-4. Images of the capture volume in different states of calibration. a) The cameras are randomly spaced and not representative of the physical capture volume. b) The location of each camera is known in relation to every other camera, but the coordinate system is not representative of the physical capture volume. c) The origin of the physical capture volume has been defined. The virtual capture volume is representative of the physical capture volume.

The motion capture system requires regular calibration before data is collected. In this dynamic calibration, an L-shaped tool (Figure 3-5) with five precisely spaced markers is used. The configuration of the five markers is recognized by the motion capture system. When two or more cameras record all five markers in a simultaneous frame, the system can begin to calculate information on the location of each camera. This process is repeated at least 850 times for each camera as the wand is moved throughout the capture volume. A complete dynamic calibration results in what is seen in part b) of Figure 3-4. Finally, the calibration tool is also used to set the volume origin and the corresponding camera frame of reference by placing it in the capture volume.

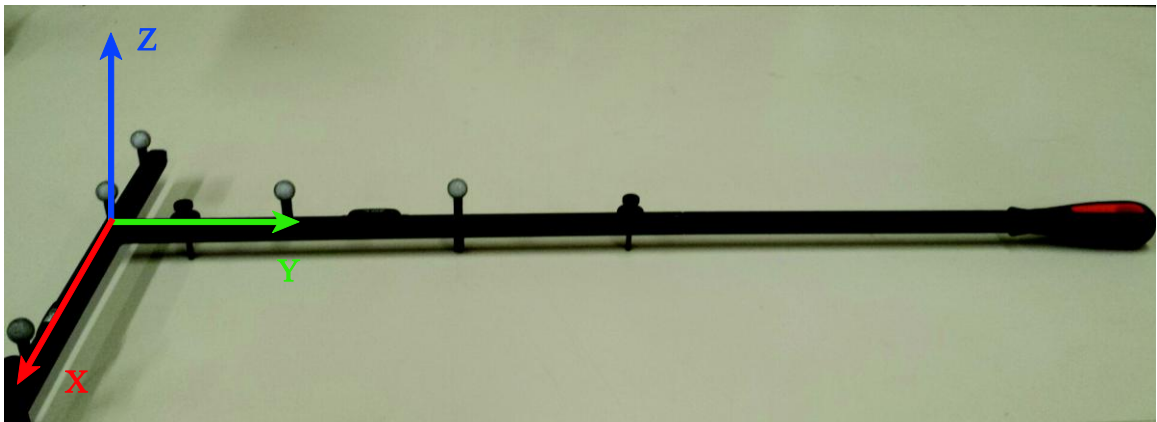


Figure 3-5. Calibration tool with the coordinate frame that is recognized by the cameras during the process of setting the volume origin.

To begin the calibration process, all cameras are checked using the camera view in Nexus to ensure that no unexpected markers or reflective surfaces are visible in the capture volume. Because the markers used for the butterflies are small and had a tendency to fall off during flight, this time was taken to remove these markers from the capture volume floor. Another cause of unexpected reflections is a high concentration of

camera strobes in small regions of the capture volume. These concentrations caused objects such as the carpet, walls, and the test operator to reflect the NIR light above the threshold of the cameras. Detecting a large number of reflections in small areas can cause some cameras to crash and stop recording data. This problem was solved by adjusting the strobe intensity of each camera. For all tests, the strobe intensity was kept at 80% for all cameras except camera 9 (see Figure 3-1) which had to be reduced to 60%. Camera 12's viewing angle caused it to see the strobes of cameras 1 and 2, which created false markers. This was solved using the masking feature. This feature creates dead pixels only where a false marker is seen at the time of masking. The masking process is automated and only takes a few seconds. The results of the masking process are shown in Figure 3-6.

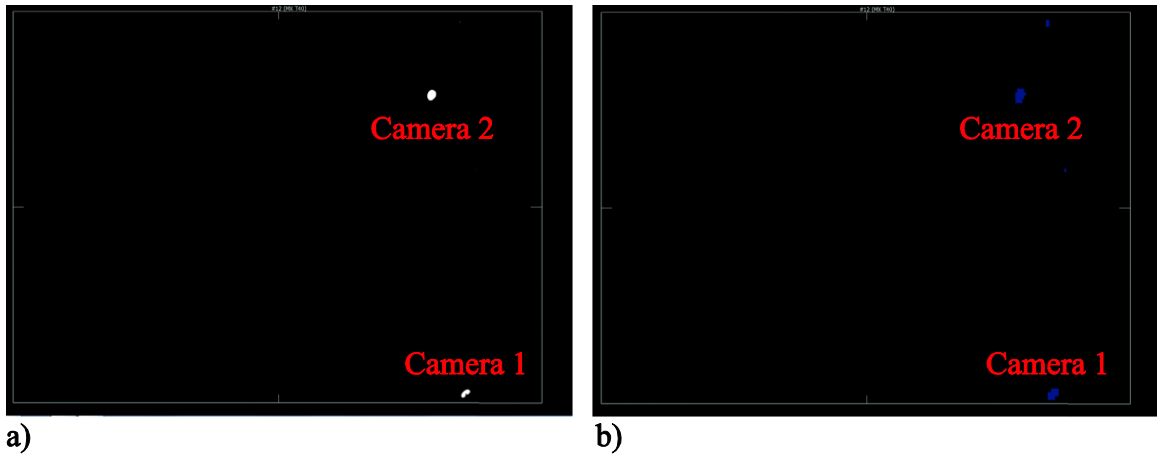


Figure 3-6. Results of camera masking on an image from camera 12 (see Figure 3-1). a) The two markers recognized are actually a collection of markers which correspond to the strobes of cameras 1 and 2. b) The markers have been masked and no data will be collected from those pixels.

Once the false markers and reflections have been removed, the calibration application is started. The operator takes the calibration tool into the capture volume and waves it around. As two or more cameras see all five markers in the same frame, an image is

recorded and each camera that records it receives a number, called a wand count. This wand count is used to stop the calibration once all 22 cameras have recorded at least 850 wand counts. When the wand count limit is reached, Nexus automatically begins calculating the calibration constants used to determine the location of each camera with respect to all other cameras. The result of this calculation is seen in Figure 3-4 (c).

After the cameras have been calibrated, an image error is provided for each camera. This error is a numerical value used to provide a standard to ensure the quality of each calibration. The testing standard for a quality calibration is all cameras recording an error less than 0.25. This number was selected after a series of calibration tests were performed in which flat tape markers, such as those used to test the butterflies, were attached to a sheet of paper. The paper was then placed at various locations in the capture volume. A poor calibration caused the markers to appear to vibrate slightly when viewed in Nexus, while a good calibration would not indicate any marker vibration. The image error was minimized by ensuring that the frames recorded during the calibration were thoroughly spread across each of the cameras views. Live feedback of the images recorded during calibration was provided so that the test operator could spread the images about the capture volume more efficiently. Each time a frame was recorded, a representation of each marker is shown as a colored marker path in the camera view (see Figure 3-7). A 42 inch Samsung TV was used to mirror the desktop PC window so that the test operator could see the camera images real time on a sufficiently large screen.

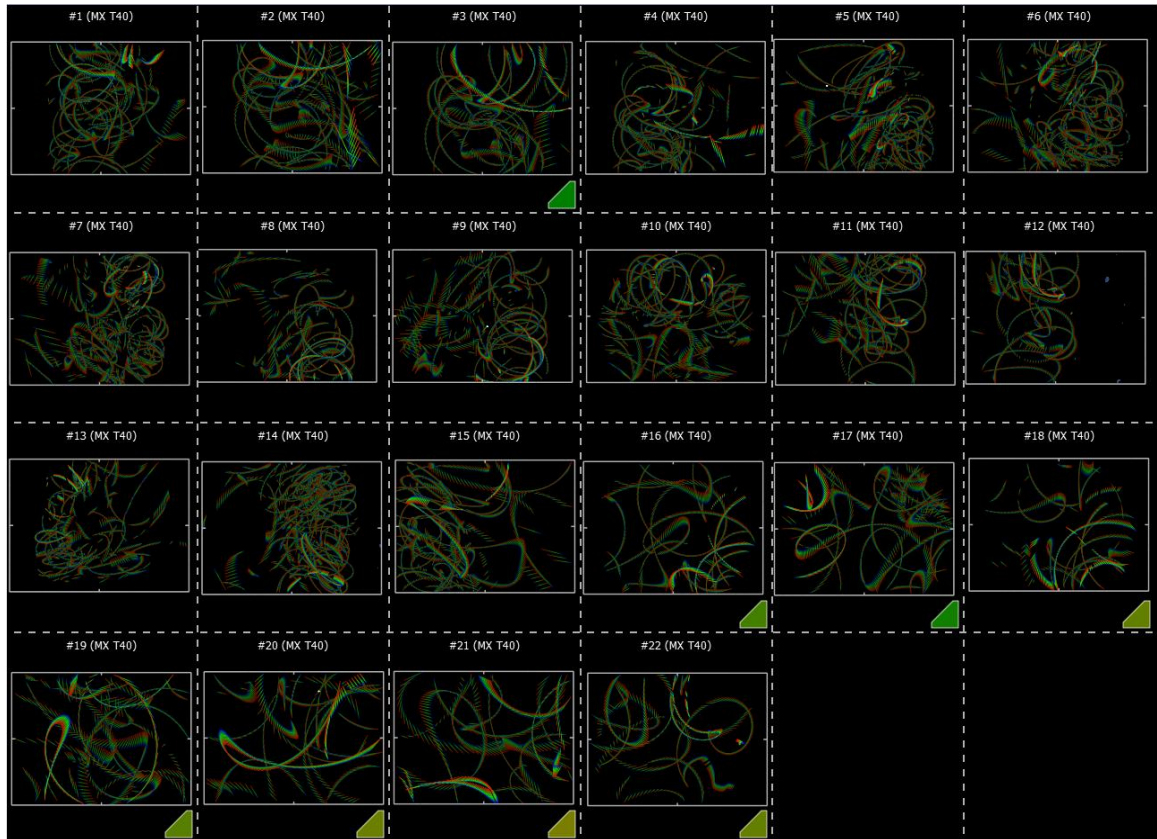


Figure 3-7. Image from all 22 cameras during calibration. The green tabs located at the bottom right hand corner of some frames indicate that less than 850 wand counts have been recorded. The color of the tabs indicates how close to this limit each camera is, such as the tab for camera 3 is a darker green color indicating it is close to the 850 mark. The yellow-green color of the tabs seen for camera 21 indicates that the wand count is further away from the 850 limit.

The distribution of the colored marker path gives an indication of the quality of the spread of images for each camera. Figure 3-8 demonstrates a camera with good coverage and one with poor coverage. The system requires calibration only when cameras move due to wall vibrations, slippage, or relocation. It was found that a calibration at the beginning of each day of testing was sufficient to retain uniform, accurate resolution.

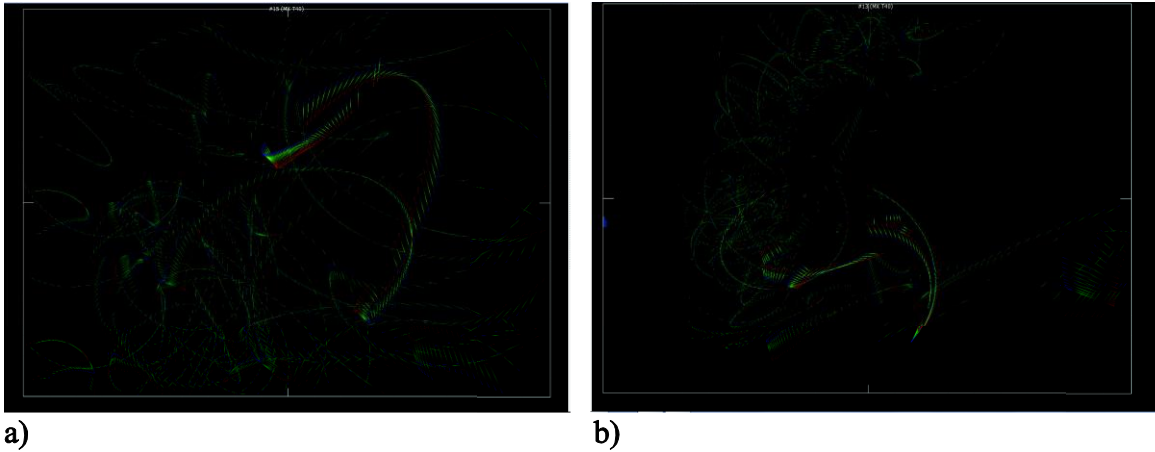


Figure 3-8. Demonstration of the difference between a) a good calibration with relatively even distribution of wand images and b) a poor calibration with a clustered distribution of wand images, leaving areas with no wand images.

3.3 Markers and Camera Placement

Four different size markers made by MoCap Solutions and a custom flat marker made from reflective tape were evaluated (see Figure 3-9). MoCap markers, labeled 1 to 4, are spherical in shape and are covered in a tape that is designed to efficiently reflect NIR light. The custom made marker is labeled as Marker 5. Table 3-1 summarizes the size of each marker and its relative mass to a typical Monarch butterfly. The reflective tape marker was chosen over the spherical MoCap solutions markers for its significantly reduced mass. This choice was critical to reduce the changes in flight behavior that could be caused by the addition of mass to the butterfly.

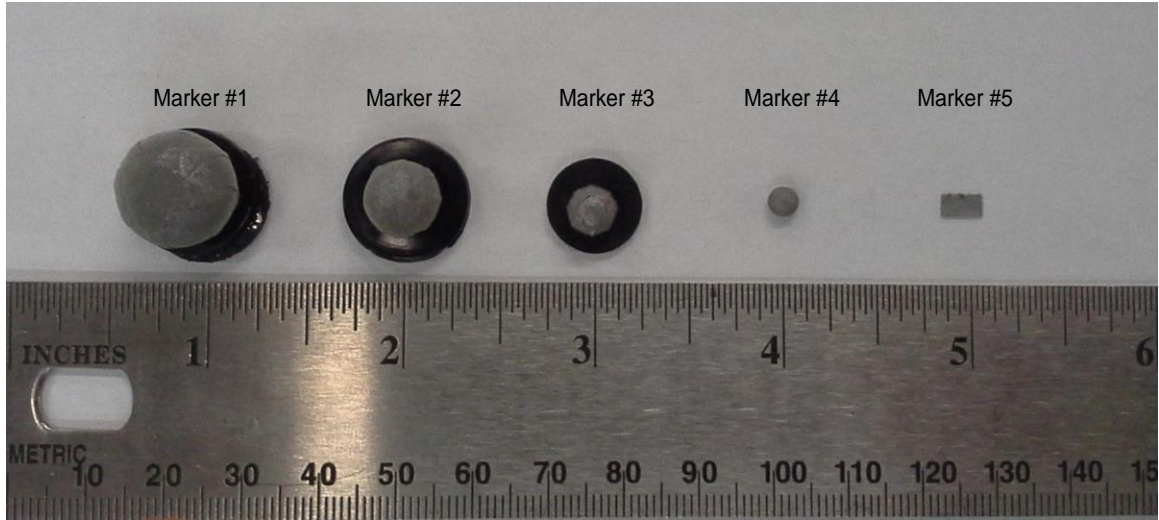


Figure 3-9. Markers that are used in the ATOM lab for motion tracking studies. Marker #5 is used for all butterfly testing and is a flat tape while the other markers are hemispherical.

Table 3-1. Properties of each marker shown in Figure 3-9.

Marker #	Diameter mm	Mass g	% Butterfly Mass (0.5g)
1	14	2.0377	407.5
2	9	0.5915	118.3
3	6	0.1871	37.4
4	3	0.0317	6.3
5	3x5	0.0058	1.2

Conversely, the size and shape of the custom made Marker 5 limited the ability of the cameras to effectively track the motion of the butterfly. One cause of this was the small size of the marker which reduced the resolution of the image the camera records, especially as distance between the two increased. To visualize this, a resolution test was conducted in which all five markers were placed directly under camera 17 (see Figure 3-1). The markers were tested at the same horizontal position, but at twelve different heights in the capture volume. The height was changed by placing the markers on T40s packaging boxes, which measured 7" in height, on top of each other (see Figure 3-10). At every height increment, h , an image was recorded from camera 17 and camera 2. Camera

17 provided a direct viewing angle to the marker, while camera 2 provided an image with an angle of incidence similar to the diagram seen in Figure 3-10.

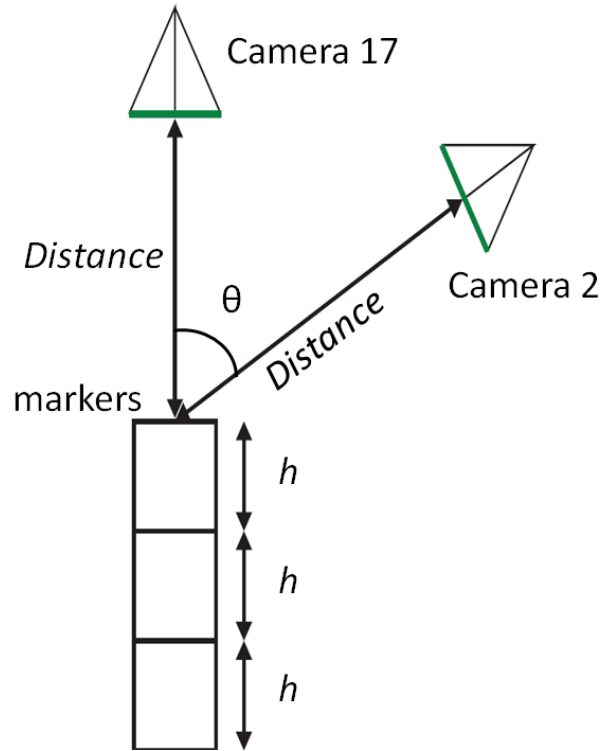


Figure 3-10. Depiction of resolution tests conducted to characterize effects of distance and angle on the quality of the marker image. An image of the collection of markers was taken from cameras 2 and 17 for each height increment, h .

Figure 3-11 and Figure 3-12 demonstrate the results of this test, while Table 3-2 reports the distance from Marker 5 to each camera, and the viewing angle between Marker 5 and camera 2 as boxes are added. Figure 3-11 demonstrates that the number of pixels representing the marker increases as the distance between the marker and the camera decreases. However, all markers were clearly visible even at the greatest distance. To increase the resolution of the smaller markers the width of the capture volume was reduced to 5.7m while the number of cameras was doubled.

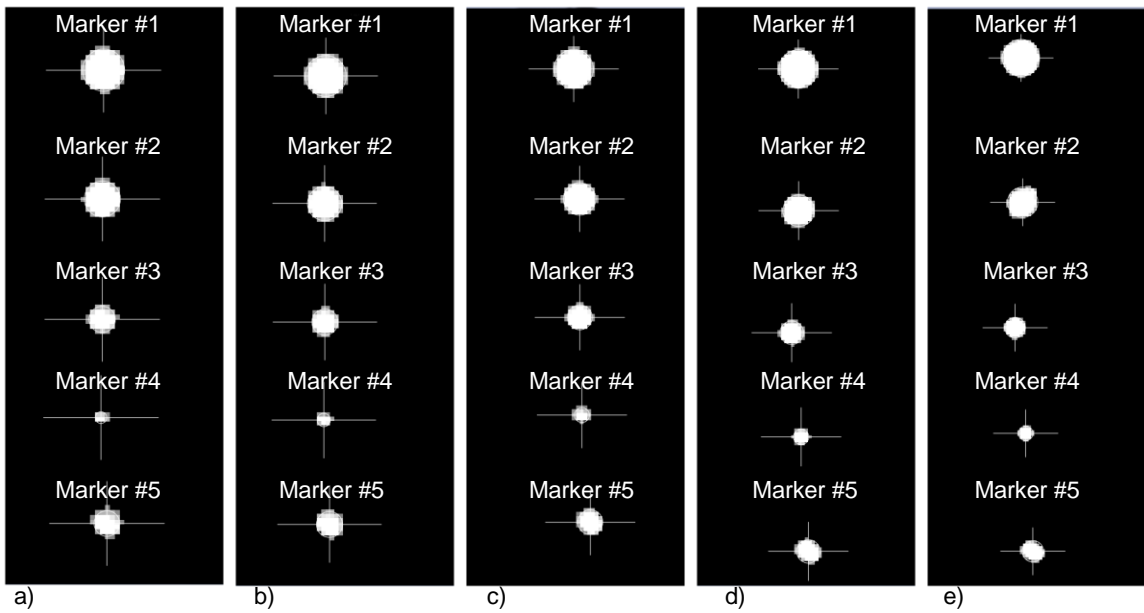


Figure 3-11. Comparison of the resolution for each size marker recorded from camera 17, (identified in Figure 3-1c) which was directly overhead of the markers so that the viewing angle does not change as height is increased. The images were taken from heights of a) 0h b) 2h c) 5h d) 8h and e) 12h.

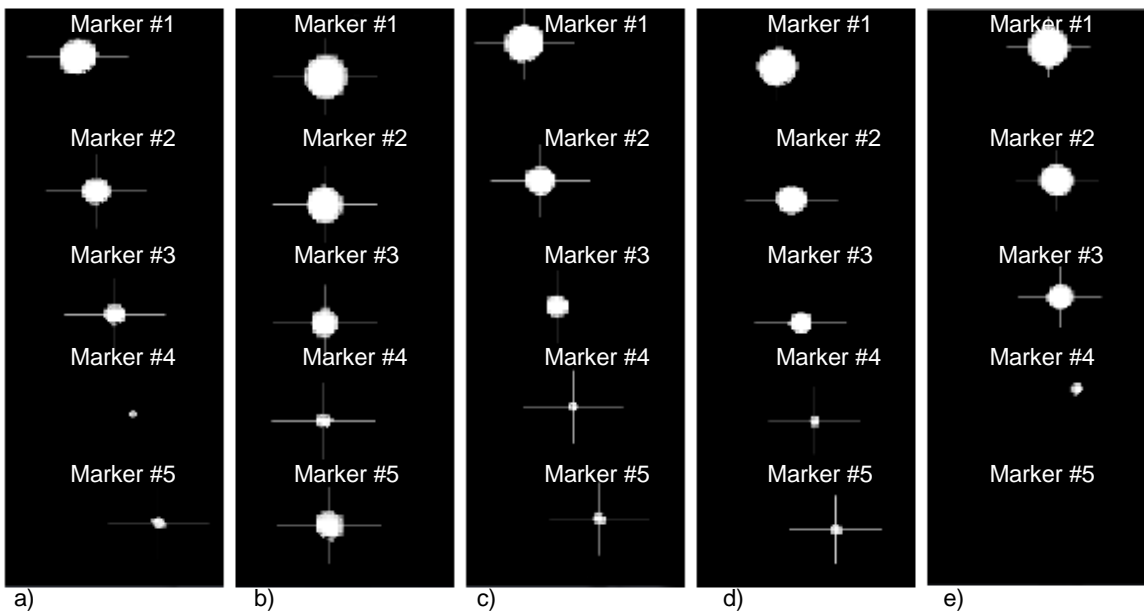


Figure 3-12. Comparison of resolution from Camera 2, highlighted in Figure 3-1 c, which provided an angled view of the markers. As the height of the markers was increased, the angle between the camera and markers increased. The images were taken from heights of a) 0h b) 2h c) 4h d) 6h and e) 8h.

Table 3-2. Distance between Marker 5 and both cameras and the angle between the marker and camera 2. These distances and angles correspond to the images seen in Figure 3-11 and Figure 3-12.

		0h	2h	4h	5h	6h	8h	11h
Camera 17	Distance [mm]	4152	3778	3405	3218	3030	2652	2093
Camera 2	Distance [mm]	3853	3504	3170	2998	2840	2512	2077
	Angle [degrees]	22.0	24.2	27.1	28.5	30.4	34.2	43.0

The second cause for a loss in resolution was the flatness of Marker 5. The spherical geometry of the MoCap markers makes the area of the marker seen by the camera less sensitive to incidence angle. A flat marker is significantly affected by the incidence angle as is evident in Figure 3-12. As the viewing angle increases, the marker reduces in resolution and eventually disappears, even though the distance from the camera is comparable to the images captured in Figure 3-11. The addition of cameras to the capture volume increased the range of detection angles inside the capture volume.

As shown in Figure 3-13, a total of seven markers were placed on each butterfly: six on the wings and one on the thorax (head). Wing markers were placed on the top and bottom of each forewing and one of the hindwings so that the flapping motion would not interfere with the camera's view of each marker throughout the stroke. The forewings provided data on flapping of the butterfly, the thorax provided information on the trajectory, and the hindwing aided in post processing and identification.

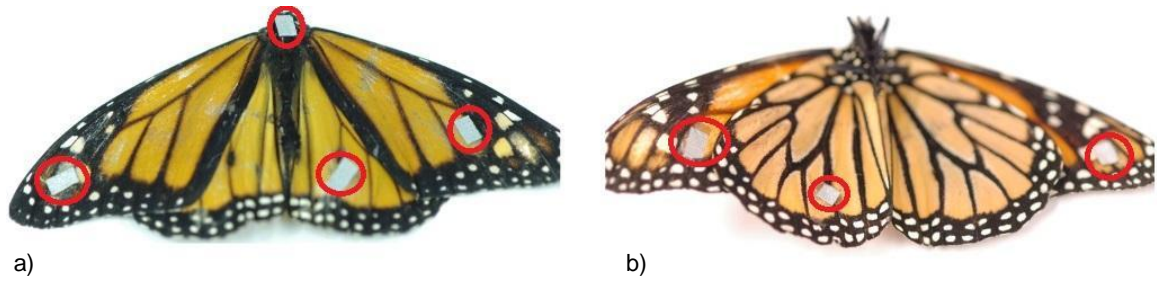


Figure 3-13. Marker locations on the a) dorsal and b) ventral side of the butterfly.

3.4 Post Processing in Nexus

The use of markers on the butterflies prevents the need for manual frame-by-frame identification of the butterfly motion and replaces complex shape recognition algorithms. Each unassigned marker is initially represented as a grey sphere in the three dimensional virtual work space of the Nexus post processing environment. Once a series of markers are identified as the butterfly, a *subject* is manually fitted to the markers. A *subject* consists of all markers of interest and the segments that connect the markers. The markers of each subject are set to a series of colors, such as in Figure 3-14 where the orange, blue, green, and gold represent the head, right wing, left wing and rear wing, respectively. Nexus uses this subject to identify the markers throughout the entire data capture sequence. The only user input required in the post processing is the creation of the subject, and inspection of the trajectory. This inspection requires the user to ensure that the trial contains usable data where all four markers are regularly present. The fitting provided by Nexus was susceptible to swapping the markers on each forewing at the end of a flapping wing stroke, when the wings move closely to each other. They could also be obstructed from the viewing area of a camera or mistaken for a single marker, in which case one marker would disappear. When this occurred, the Nexus software experienced

trouble identifying a marker upon reappearance, sometimes assigning markers to incorrect wings or not reassigning the marker at all. Manual fitting of a subject was then needed.

The manual fitting process is demonstrated for two disappearing markers in Figure 3-14. The first image demonstrates the representation of a full subject in the Nexus virtual workspace. In the second, third, and fourth images (b, c and d) both the rear wing and right wing disappear and the right wing marker is assigned to the physical left wing marker. The fourth and fifth images (e and f) demonstrate manual fitting of each of the unassigned markers. Finally the whole subject is presented once again. Once a complete trajectory was ensured, the position and velocity of each marker were exported to Microsoft Excel and MATLAB for further analysis.

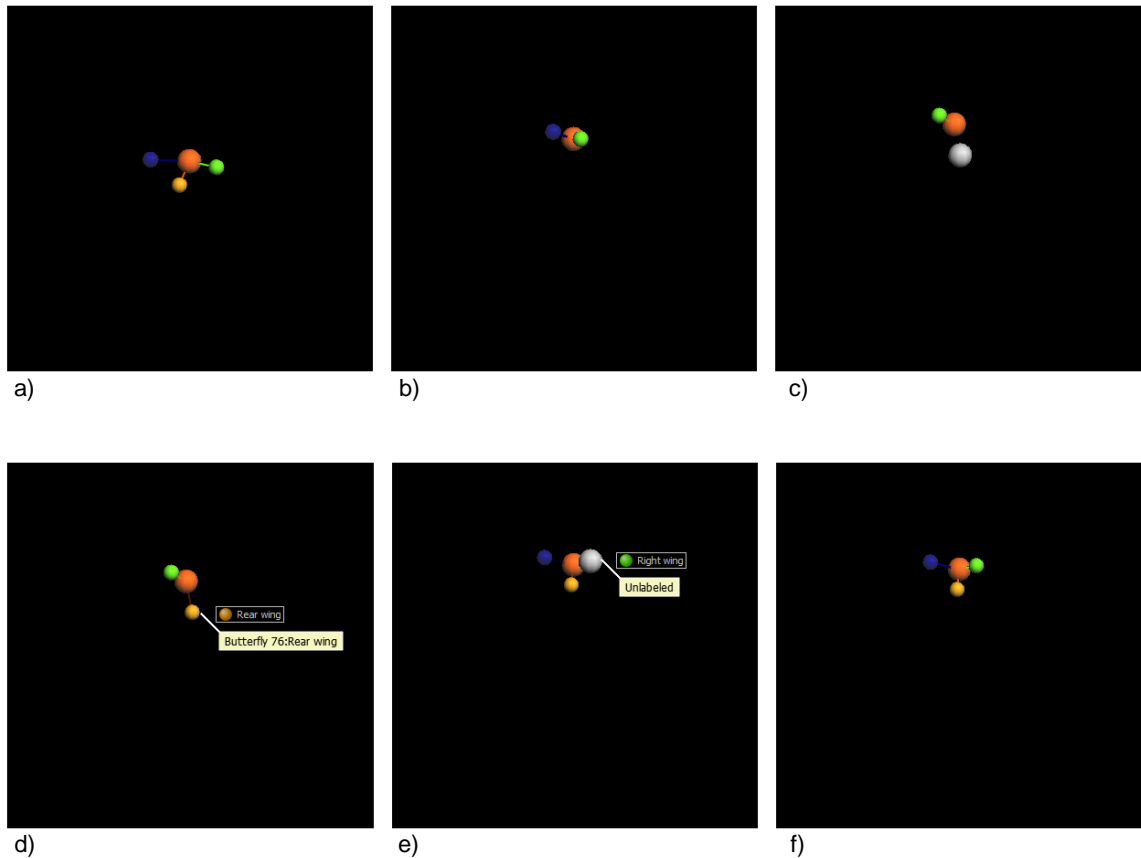


Figure 3-14. Demonstration of the loss of markers during data capture and refitting the subject which consists of an orange head marker, blue left wing marker, a green right wing marker and a gold rear wing marker. a) First, all markers are represented, the subject is denoted by the colored markers. b) As the butterfly wings close, the yellow rear wing marker is lost. c) as the wings continue to close, the rear wing marker appears unfitted to the subject, while the blue right wing marker disappeared. d) The user is manually fitting the subject to the rear wing marker, e) then the right wing marker appears unfitted to the subject and is manually added. f) Once again all markers are present and subject is fully fitted.

3.5 Monarch Butterflies Handling Procedure

The test matrix for the experiments described in this thesis was developed to gather data comparing the flight of butterflies with scales and with their scales removed. A total of 108 Monarch butterflies were purchased from Swallowtail farms [53] over five non-consecutive weeks between March 9 and June 24, 2014. Testing was restricted to these

dates due to the seasonal availability of the Monarch butterfly. The butterflies were shipped in individual boxes packed in a Styrofoam cooler with ice packs to ensure the butterflies were kept safe through the shipping process.

Cool temperatures during overwintering reduce the use of butterflies' lipid reserves [31] which consequently reduces body activity and even impairs their ability to fly [32]. This state of reduced activity was also used by the research team to more safely handle the specimens. Upon arrival, the butterflies were placed inside a refrigerator which was kept at approximately 2° Celsius. One at a time, the butterflies were removed from the refrigerator and weighed on a Citizon CX 120 scale which measures to 0.1 milligram precision. The butterfly was identified by a number which was assigned as consecutive numbers from 1 for the first butterfly of the first test week and extended to 86 for the last butterfly tested. In order to place the reflective markers, the butterflies were then transferred from the mass scale directly to a work station which consisted of an ice-pack covered with a paper towel. The wings of the Monarch butterfly are covered with microscopic scales which detach easily. To ensure that the markers were securely attached, the scales directly under the marker were removed before application. The butterfly was again placed on the ice-pack and immobilized by gently using a finger to hold the wing down as close to the body as possible. A cotton swab was then used to very gently brush the scales from the desired location, until the wing became transparent. Once the scales were removed, the marker was carefully applied using a finger. The markers were placed in such a manner that four butterflies could be kept in a single terrarium. This was done by recording which hind wing was marked as well as slightly offsetting one marker in the spanwise direction.

Figure 3-13 shows an example butterfly which has the right hind wing marked with a slight offset from the location of the right forewing marker. Another use for the offset in the forewing marker was to prevent the cameras from mistaking the markers on each forewing as a single marker when the wings were very close together. Slightly offsetting the one marker helped to reduce the frequency of this marker disappearance. The head marker was placed last, and did not require the removal of scales. Figure 3-15 shows images of butterflies being handled during marker application. The butterflies were lightly held as close to the root of the wing as possible.



Figure 3-15. Handling as well as placing markers on the butterfly. The images demonstrate accessing a) the ventral side of the forewing, b) the dorsal side of both wings as well as c) the head marker. Note that the butterflies were held as close to the body as possible, which was both due to an increase in robustness of the wings near the body as well as mitigating the effects of accidental scale removal.

After the addition of markers, the butterflies were placed once again into the refrigerator for a brief time. The butterflies were then weighed once more on the scale and the length of a single forewing was measured from root to tip. The gender of the specimen was identified using the scent markers which appear on the rear wing of male butterflies but not on females (see Figure 3-16). Finally, the butterflies were placed in a terrarium where they were kept until testing (Figure 3-17). The terrariums had directed incandescent light to provide warmth to the butterflies. Food was supplied in the form of

Gatorade soaked cotton balls which were present in the terrariums throughout the entire week of testing.

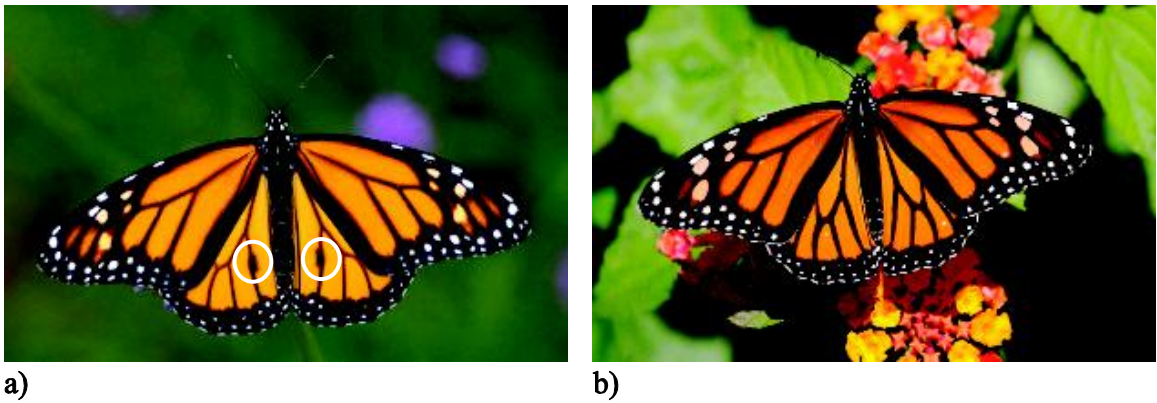


Figure 3-16. Comparison of a) male and b) female monarch butterflies. The scent marks (indicated by white circles) which are present in a, but absent in b gives clear indication about each butterfly's gender.

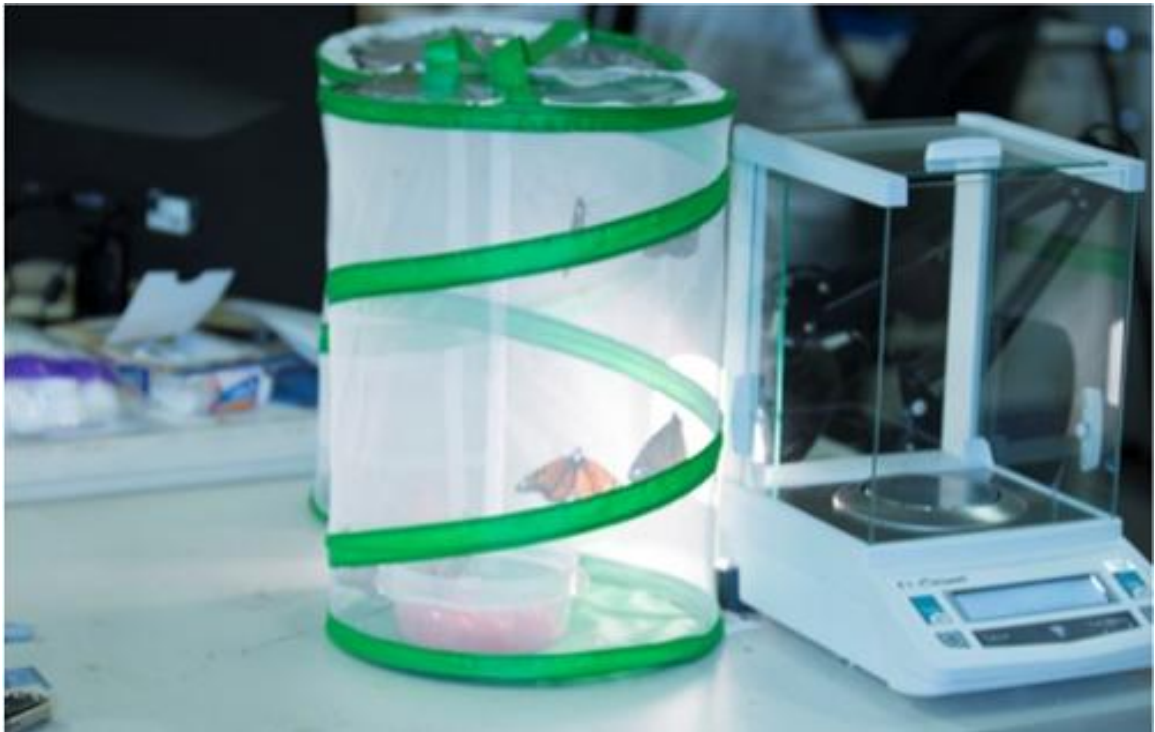


Figure 3-17. Butterflies sunning in incandescent lamp light in the terrarium. In the bottom of the terrarium a butterfly is feeding in the bowl containing Gatorade soaked cotton balls. The Citizen CX 120 scales used for weighing the butterflies is next to the terrarium.

After flight tests with scales were performed, the scales of the butterflies were removed. The process of scale removal consisted of placing the butterflies back into the refrigerator to reduce their activity. Once removed from the refrigerator, the butterflies were again placed on the ice pack covered with a paper towel. The scales were removed using a cotton swab and very delicate strokes. The tips of the wings and the trailing edge proved to be extremely fragile, and great care was needed so that the wing was not ripped. If the wing did rip during scale removal, the test operator made a judgment as to whether the effect of the rip was negligible. Due to the large number of butterflies tested at a time, the operator tended to err on the side of caution. Any butterfly whose wing was deemed too damaged for a comparison was released at the end of the day of scale removal. The scales provide the bright coloring of the wing so that the wing became transparent when the scales were removed. Figure 3-18 shows the difference between a butterfly with scales and without scales.

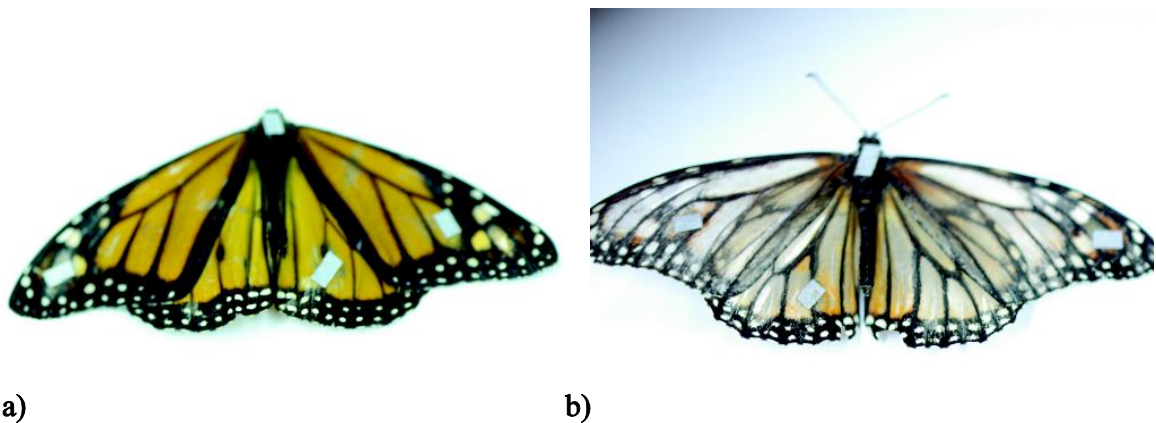


Figure 3-18. Comparison of butterfly a) with scales and b) without scales. Note the transparency of the wings of the butterfly without scales. This comparison is presented of two different butterflies.

3.6 Flight Testing Procedure

A flight test consisted of a 20 second motion capture trial, during which a butterfly was released and allowed to fly freely. The butterfly was released from the operator's hand, and the location of the release within the capture volume was subject to change. Early in the testing cycle it was observed that the camera coverage inside the capture volume was extremely heterogeneous. This resulted in areas where data could not be collected because one or more markers could not be seen regularly. The butterflies were observed to fly in consistently similar paths. Therefore, subsequent release points of a butterfly were refined after observing previous flight paths to ensure that usable data could be collected. Trials were also performed in which Gatorade soaked cotton balls and an incandescent light source were placed in the capture volume to attract a butterfly. These methods proved ineffective and were abandoned. It is speculated that the Monarchs would be attracted to milkweed, which provides food as well as a place to lay eggs. After Nexus had recorded data, the butterfly was captured in preparation for another test.

Testing was conducted the day after marker placement so that adverse effects of handling and additional mass of markers were reduced as much as possible. The first task completed on test day was calibrating the cameras. Next, the butterflies were removed from terrariums one at a time and flights recorded for the first round of 10 consecutive motion capture tests. At the end of the first round of 10 tests, the butterfly was placed back in the terrarium and the next butterfly was immediately removed for testing. Once all of the butterflies had been through the initial round of tests, a second round of similar

flight tests were performed on each butterfly. The second round of testing completed the first day of testing.

The day after testing consisted of removing scales which was described in Section 3-5. The scale removal process was very time intensive, and also took a toll on the butterflies. To mitigate the effects due to handling during scale removal, the butterflies were not tested during scale removal day. The next day consisted of testing the butterflies without scales, and the test procedure was identical to that of the first day of testing. At the end of the second flight testing day, all remaining butterflies were released.

This process of testing was refined during the first and second weeks and fully implemented in the third through fifth test weeks. The first two weeks represented the learning curve of the handling and testing of the butterflies. It was noted that room temperature had a noticeable effect on the butterfly flights. Temperatures higher than 75°F increased the length of time the butterfly flew without trying to land. Capturing the butterflies after each test proved time consuming. Therefore the temperature in the test volume was set at 75°F as this was the lowest temperature that could be maintained consistently.

CHAPTER 4

Analysis

The data collected from the Vicon motion capturing system included the frame and the position and velocity of each of the four independent marker locations. Sections 4.1-4.3 detail the analysis that was used to calculate the flapping wing motion and flight trajectory of the body. The data presented in this chapter was collected during Flight 1139, which was conducted using butterfly 46 recorded during the third week of testing.

4.1 Data Export

Nexus exports the data recorded during a motion capture trial in the form of an ASCII text file or a comma separated value (CSV) file. The CSV file was chosen as an output format as the MATLAB analysis software provides a simple method for data import from this file type. Details of the data import in MATLAB are explained in Section 4.2. As discussed in Section 3.4, the exported data consist of the three components of the position and velocity of each marker specified in the subject. All butterflies had four marker points that were tracked, and therefore all the data from this experiment comes in the

form of a frame index, twelve position components, and twelve velocity components. The position is recorded as the coordinates of each marker reported in the camera frame, as seen in Figure 4-1. The number under the *Trajectories* label represents the frame rate the data was collected at in frames per second (100 fps for this thesis). The *Frame* column provides the frame index for which the information on each row was captured. The value of the frame starts at 0 at the start of each motion capture trial. However only the frames in which a subject is identified is exported to the CSV file. The order that the four markers appear in the data is dependent on the order the markers are identified in the process for building subjects in Nexus. In order to create a consistent format across all data types, the order of marker identification was *Head*, *Left wing*, *Right wing*, and *Rear wing*. It can be seen in Figure 4-1 that in two frames (478, 479) position data of the *Rear wing* are missing This was caused by the disappearance of the marker on the rear wing from the cameras. The CSV file also contains the velocity of each marker below the position data, in the same format.

Trajectories														
100														
			Butterfly 46:Head			Butterfly 46:Left wing			Butterfly 46:Right wing			Butterfly 46:Rear wing		
Frame	Sub Frame	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	
		mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	
473	0	3529.99	1812.69	950.086	3518.7	1777.15	948.489	3524.45	1855.51	938.009	3546.83	1797.67	942.718	
474	0	3515.79	1813.59	958.095	3496.14	1785.56	942.716	3498.42	1847.31	932.695	3529.4	1797.39	945.296	
475	0	3501.56	1814.68	967.173	3479.49	1792.94	946.878	3479.37	1840.59	936.806	3512.59	1796.14	953.87	
476	0	3486.49	1815.67	975.418	3466.63	1792.05	955.526	3468.51	1846.48	948.078	3498.11	1798.48	960.524	
477	0	3470.63	1816.63	982.05	3460.25	1782.57	970.719	3470.63	1860.03	965.598	3484.01	1803.45	964.877	
478	0	3453.77	1817.61	986.752	3454.91	1781.22	993.557	3476.94	1855.56	993.075				
479	0	3436.83	1818.63	990.85	3458.31	1794.58	1008.04	3470.29	1840.17	1018.92				
480	0	3420.74	1820.37	993.785	3442.63	1806.32	1013.67	3450.73	1826.51	1022.93	3445.51	1807.95	981.131	

Figure 4-1. Example data from flight 1139 of butterfly 46 exported from Nexus into a CSV file, viewed in Microsoft Excel.

4.2 Data Import

The process for importing the data was greatly simplified by the consistent formatting of the CSV file. The data was imported into MATLAB using the `xlsread()` function. Blank cells were imported as having a value of *Not a Number* (NaN). The frame, and components of position and velocity of each marker were obtained from this matrix and collected into $n \times 1$ arrays, where n is the number of frames in data capture sequence.

4.3 Data Analysis

The four marker configuration was presented as a method to record information on the trajectory of the body, as well as the flapping motion of the wing. The trajectory information can be developed simply from the motion of the head marker through space. The flapping motion required additional analysis.

4.3.1 Flapping Angle Calculation

Three vectors, \vec{P}_{right} , \vec{P}_{left} , \vec{P}_{rear} , were defined as extending from the head marker to the marker located on each wing and were calculated in global frame (see Figure 4-2). These vectors were used to calculate the flapping information of the wing, and a visualization of their location and orientation.



Figure 4-2. Visualization of parameters used in analysis. a) Markers and the vectors in Nexus. b) Butterfly in flight with important parameters highlighted.

The flapping or *plunge* angle γ is defined as the angle between both wings, where $\gamma = 0^\circ$ is at the end of the upstroke. Using the position vectors, γ is calculated by

$$\gamma = \tan^{-1} \frac{|\vec{P}_{\text{left}} \times \vec{P}_{\text{right}}|}{\vec{P}_{\text{left}} \cdot \vec{P}_{\text{right}}}. \quad (1)$$

The function used to calculate the inverse tangent in MATLAB is *atan()*. This function provides quadrant data based off of the sign of the numerator and denominator of Equation (1). The sign of the cross product found in the numerator depends on the direction the butterfly is flying, affecting the order of the vectors in the cross product. If the order of the cross product is flipped, then the angle between the two vectors is equal to the original angle minus 2π . Therefore this provides an incomplete solution, as the direction the butterfly is flying dictates the location in the stroke where $\gamma = 0^\circ$. To find a unique value for γ , the rear wing vector was used to determine the butterfly flight direction by calculating

$$\text{Sign}_\gamma = (\vec{P}_{\text{left}} \times \vec{P}_{\text{right}}) \cdot \vec{P}_{\text{rear}}, \quad (2)$$

where the variable Sign_γ was only checked for its sign. If this parameter is negative, then the calculated flapping angle was correct; but if it is positive, then the flapping angle was subtracted from 180° .

4.3.2 Flapping and Trajectory of an Entire Flight

The results of the flapping angle calculation and the vertical position of the head marker are shown in Figure 4-3. It can be seen that the flapping data from frames 575 through the end of the trial do not represent the same periodic flapping seen earlier in the segment. There are extended periods in which $\gamma=0^\circ$ corresponds to the missing data in the vertical displacement of the head marker. This indicates that the head marker disappeared from the viewing angle of the cameras. The data during those frames do not represent the actual flapping of the butterfly.

The vertical displacement of the butterfly shown in Figure 4-3 has three distinct trends. In two sections, frames 340 - 440 and frames 480 - 560, the climbing rate is positive. However, the magnitude of the climbing rate in the first segment is visibly higher than the climbing rate in the second segment. The butterfly actually descends in what appears to be a short gliding segment in frames 590 - 640. The flapping behavior of the butterfly clearly changes between the climbing and descending trajectories. There are distinct flaps in the climbing flights and a lack of periodic oscillations in the gliding flight. However, there are not obvious differences in flapping frequency between the two climbing trajectories.

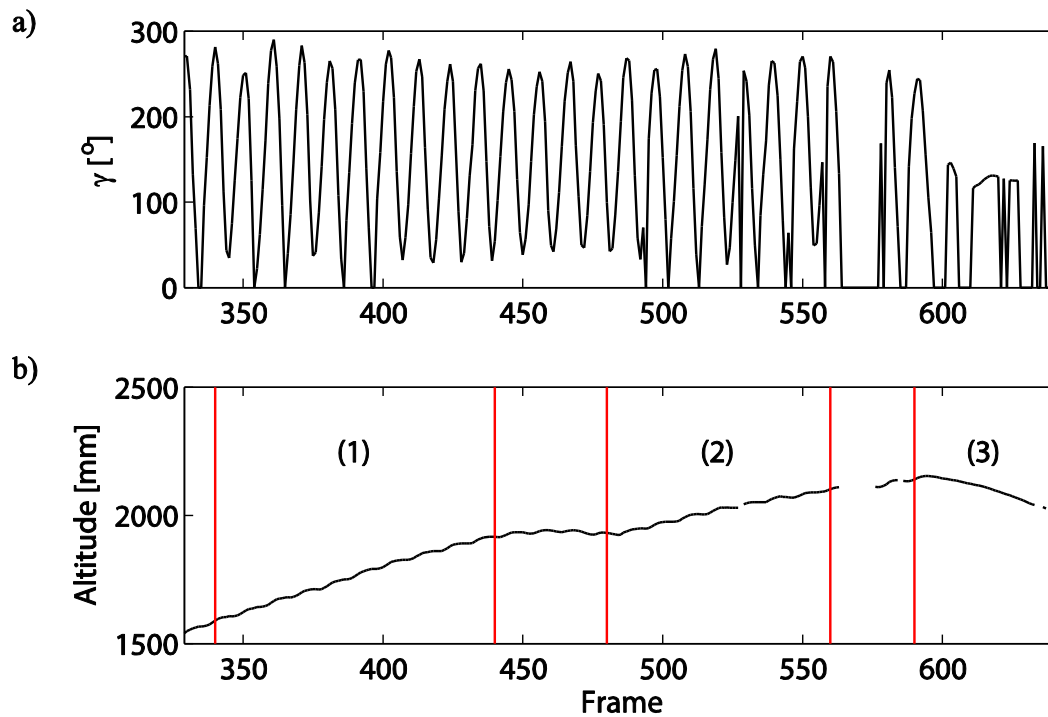


Figure 4-3. Raw flapping and altitude data from flight 1139 of Butterfly 46. a) Flapping angle of the two forewings where $\gamma=0^\circ$ at the end of the upstroke. b) Altitude of head marker which shows three different trajectories (1,2,3). Trajectories 1 and 2 show climbing flight while trajectory 3 is a descending trajectory corresponding to a glide in part a).

4.3.3 Graphical User Interface

A Graphical User Interface (GUI) was created, to collect the data into smaller segments for further analysis, and the code used to generate and run this GUI is presented in Appendix A. The GUI provided a means to display the raw data and interactively select the frames that bound the segment of interest for display. This method was used to create the plots shown in Figure 4-4, which demonstrates similar flap angle behavior, even for the two climbing rates. This indicates that closer analysis of the flapping signal, and potentially other parameters such as pitching angles, need to be investigated to draw conclusions on how the butterflies control these trajectories.

4.3.4 Disappearing Markers

The plunging data between frames 480 and 560 in Figure 4-3 also appear to have some anomalies that need to be accounted for. The angle at certain frames jumps to zero. This physically implies the butterfly's wings completely close in one frame, even though it appears that the flapping cycle is approximately ten frames. These discontinuous jumps to zero are caused by the disappearance of a head marker or a forewing marker. For example, at approximately frame 530, the head marker clearly disappears from view for a frame in Figure 4-4 (d), while the plunge angle jumps to zero for one frame and jumps back to the sinusoidal behavior in Figure 4-4 (c). The head marker disappearing was relatively infrequent, but as demonstrated in Figure 4-4, the wing markers disappear with more regularity. The wing markers also disappear more frequently at the transition between strokes when the wings are close together. The research team investigated and determined that as the flat markers used in this experiment came close together, the cameras would only see one marker instead of two distinct markers. The offset marker positioning mentioned in Section 3.3 attempted to mitigate this, but was not able to eliminate the problem completely.

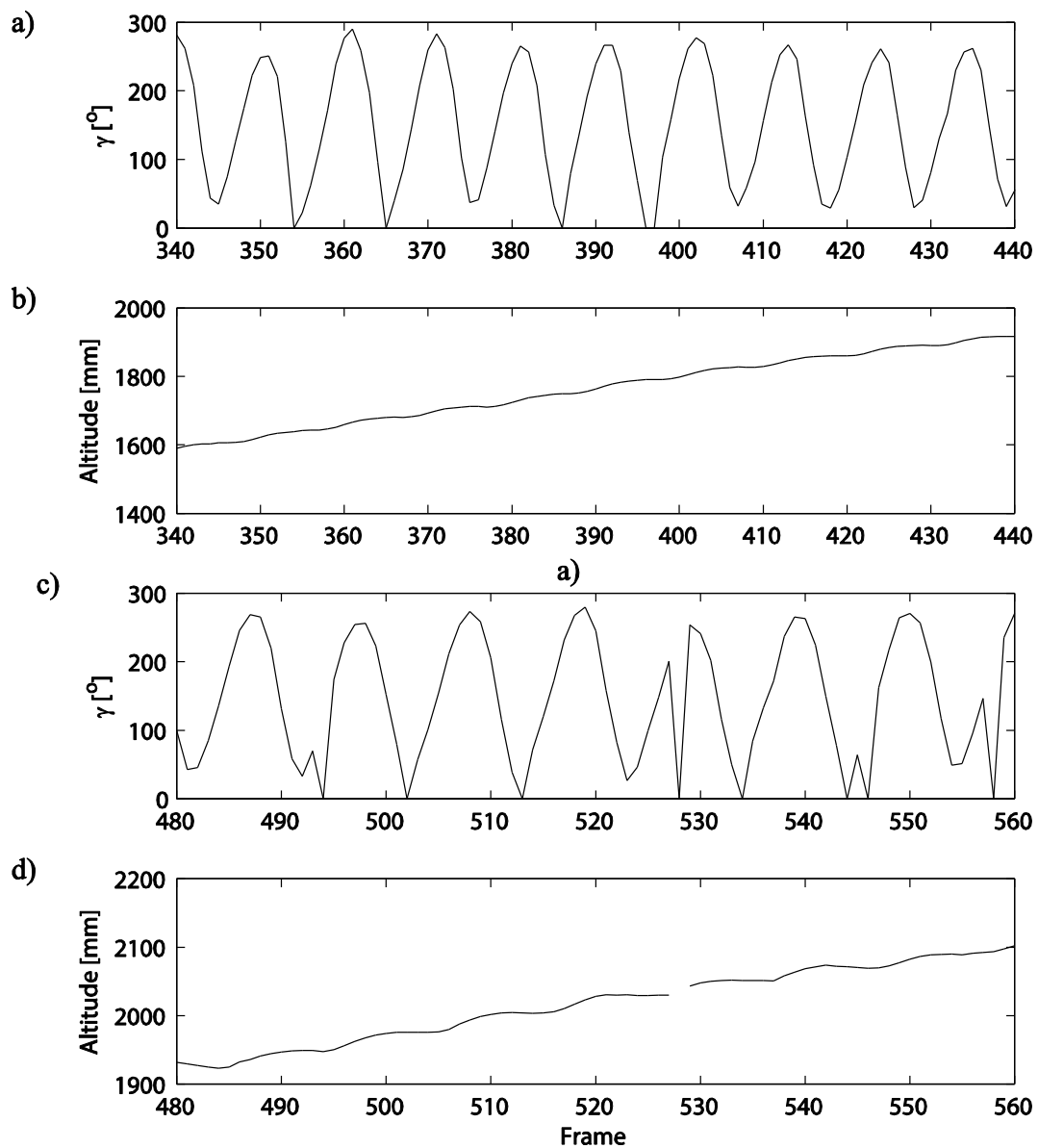


Figure 4-4. Plunging and vertical trajectory data for the two distinct climbing trajectories seen in Figure 4-3. a) Plunging angle recorded during frames 340 - 440. b) Vertical trajectory recorded during frames 340 - 440. c) Plunging angle recorded during frames 480 - 560. d) Vertical trajectory recorded during frames 480 - 560.

4.3.5 Cubic Spline Interpolation

Multiple methods to eliminate the gaps in data introduced by disappearing markers were investigated. A linear interpolation was first applied, but this method was abandoned due to the highly nonlinear behavior of the plunging angle. Because the signal appears somewhat sinusoidal, a Fourier interpolation was next applied. As seen in Figure 4-5, the Fourier interpolation appeared to poorly represent the frequency of flapping. One explanation for this is that there are flap-to-flap changes in flapping frequency, which reduces the effectiveness of interpolating while assuming a periodic function. The final interpolation method attempted was the cubic spline interpolation, also shown in Figure 4-5. The cubic spline interpolation provided a good fit to the already collected data as well as a better approximation of missing data points than a linear interpolation.

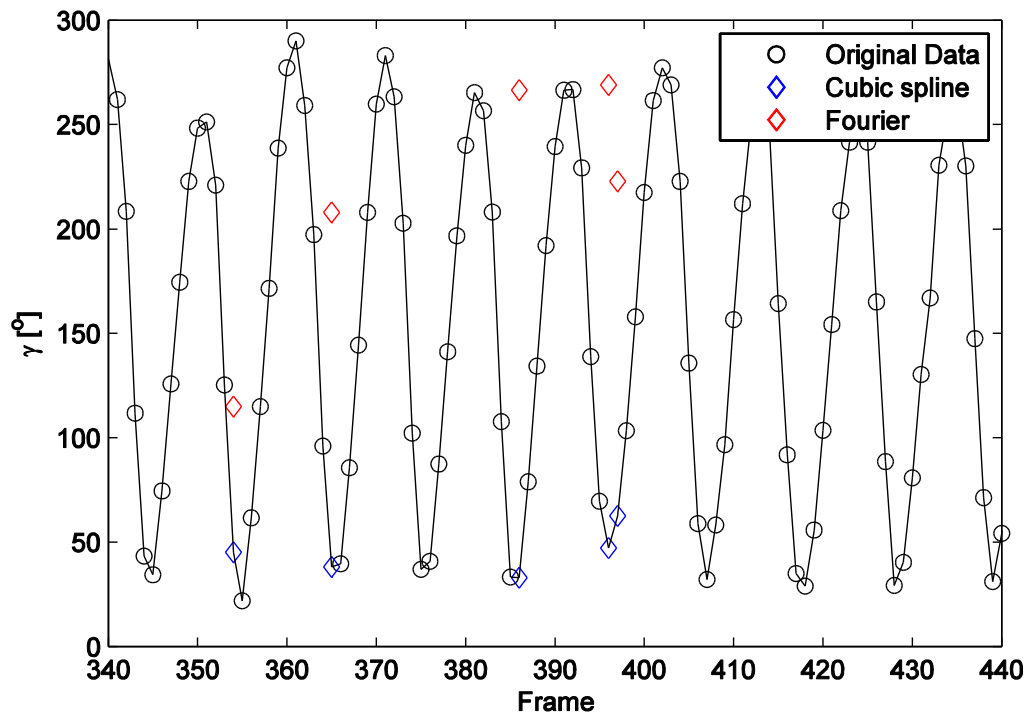


Figure 4-5. Plot of two interpolation methods which were investigated to replace data that was lost by disappearing markers.

4.3.6 Flapping Characteristics

The frequency of the plunging angle was calculated with the Fast Fourier Transform (FFT), which provides an average frequency over the entire signal. As previously mentioned, the frequency appears to slightly shift during each flap. This, is expected as the butterfly makes minor adjustments to control flight or change its trajectory. However, the variation in the flapping frequency cannot currently be described due to the low number of sampled points per flap. Sampling at a higher frequency would shed more light on this phenomenon, and can be pursued in future flight tests. The phase ϕ was determined using the complex flapping frequency $\bar{\omega}$ obtained from the FFT by

$$\phi = \tan^{-1} \left(\frac{\text{imag}(\bar{\omega})}{\text{real}(\bar{\omega})} \right). \quad (3)$$

The peak to peak amplitude of the plunge angle was calculated by checking the derivative at every time step for a change in sign, which for a periodic function should indicate a peak. When the sign changed from negative to positive, then a local minimum was expected, while a change from positive to negative should indicate a local maximum. To reduce the likelihood of a false peak being detected, five data points on either side of the current frame were checked to ensure that the frame did in fact correspond to a local extrema. The results of this algorithm are shown in Figure 4-6. The average of the minima subtracted from the average of the maxima provided average flapping amplitude. The results of this algorithm can also be used to segment the data for flap-by-flap analysis. Comparing individual flaps to changes in trajectory could provide more detailed information on what effects changes in frequency or flapping amplitude have on climbing flight. To complete this analysis, the data needs to be sampled at a higher frame rate to capture the true location of the maxima in question. Figure 4-5 and Figure 4-6 demonstrate the problem of low sampling rate, where it is clear that some peaks are not captured.

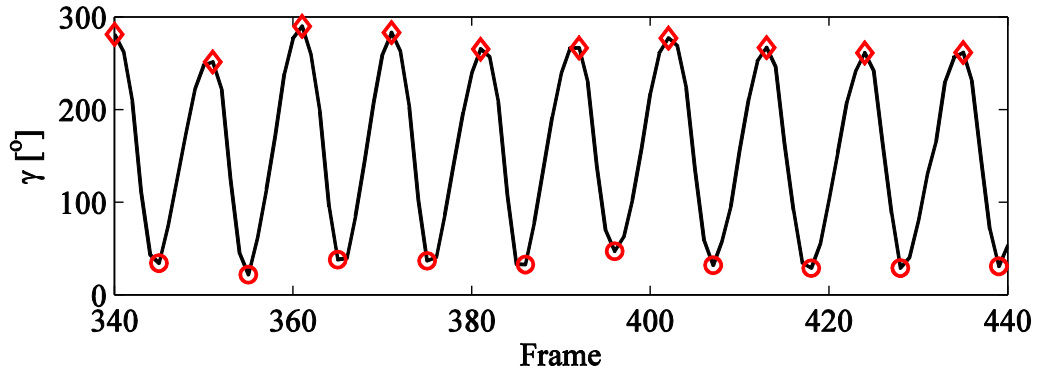


Figure 4-6. Flapping angle with the peaks identified using the averages of the extrema. The red diamonds represent local maxima while the red circles represent local minima.

4.3.7 Detrending of Vertical Trajectory

The trajectory of the butterfly, especially in climbing flight, was observed to consist of a mean flight path and an oscillation. This is clearly seen in Figure 4-7, where the trajectory is approximately linear, but the body oscillates about the mean trend. In order to better characterize trajectories, as well as collect data on the body oscillations of a butterfly in flight, a method for detrending the trajectory was devised. This detrending process utilized a moving average filter which calculated the mean trend of the signal by

$$Y_i = \frac{1}{2s + 1} \sum_{k=i-s}^{i+s} X_k, \quad (4)$$

where X is the original signal, Y is the mean trend, and $s = (\text{Span} - 1)/2$ is a parameter that was used to tune the filter based on the user defined span length (Span). An example of the instantaneous vertical position compared to the mean trend can be seen in Figure 4-7.

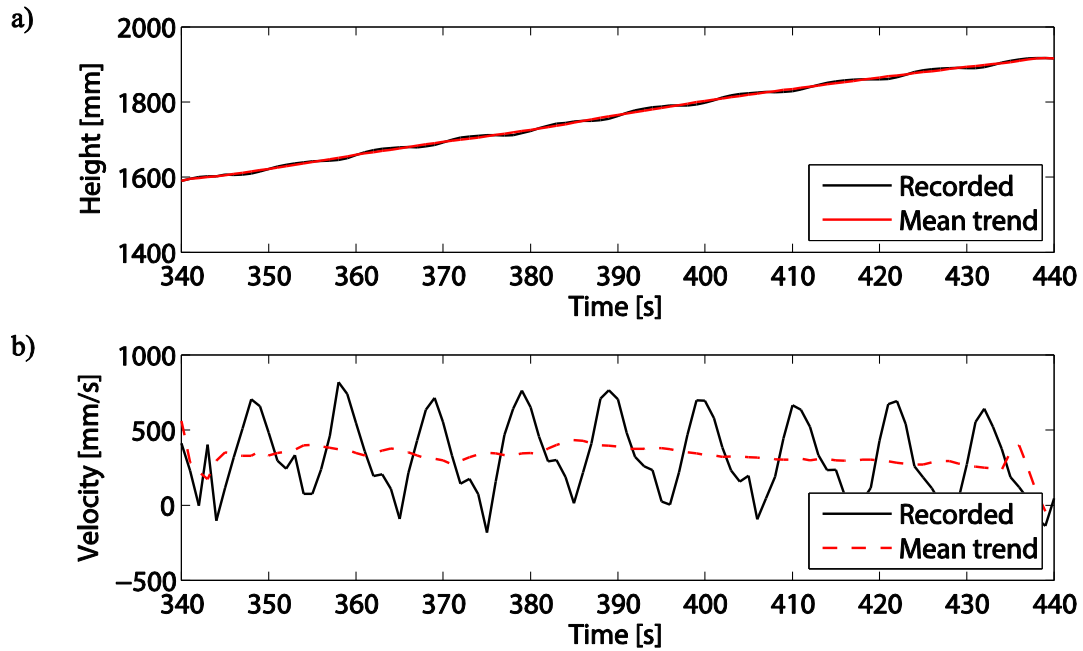


Figure 4-7. Example of mean trend extraction. a) Mean trend compared to the actual data, b) differentiated mean trend against recorded velocity data.

The value of *Span* was determined for each data segment by calculating the mean trend and differentiating it to get a velocity mean trend (Figure 4-7 (b)). When the oscillations reached a minimum average value in the velocity mean trend, the span was optimized for a particular flight. Numerical differentiation magnifies trends in data, and therefore the new velocity mean was used as the metric for the fit of the moving average filter. The optimal *Span* for calculating data was generally found to be on the order of the body oscillation frequency. One limitation of the moving average filter was that the *Span* was constrained to odd integers. Data was collected at approximately ten data points per cycle. This did not provide sufficient resolution to settle the exact period of the signal. This resulted in small residual oscillations in the velocity mean trend which could not be eliminated.

4.3.8 Vertical Body Oscillations

It has been shown that the forward flight of insects with large wings compared to the body and overlapping fore and hind wings is realized through the passive body motion [44]. To better characterize the body motion of the butterfly in flight, the mean trend was subtracted from the vertical position data (Figure 4-7 (a)) resulting in Figure 4-8. The body motion is relatively sinusoidal in nature, and therefore the frequency, phase and amplitude of body oscillations were calculated by the same method presented for the flapping data in Section 4.3.6. This methodology and data can be used in the future to study the effects of both flapping and body motion upon the general flight of butterflies.

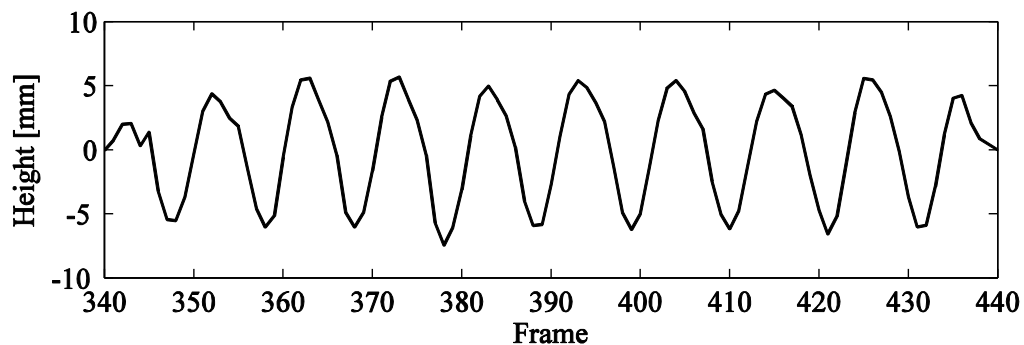


Figure 4-8. Body undulations calculated by subtracting the mean trend from the vertical position data.

4.4 Results and Discussion

The analysis process described in the previous sections was implemented into a GUI. The user can quickly view data from a given flight, select data based on quality, as well as differing trends in trajectory. When these data are selected, the amplitude and

frequency of the flapping angle, the vertical position, and the vertical and horizontal velocities of the butterfly are presented. Next, the user defines the *Span* of the moving average filter that will be used to detrend the vertical position data. The Span is refined until an optimally smooth velocity profile is obtained. At this point, another GUI was constructed to calculate, and tabulate average characteristics for the data segment currently selected. The tabulated data contains information specifying the flight the segment was obtained from, the bounding frames, and the span used for the moving average filter. The butterfly is identified by its number, mass, wingspan and gender. Six flight characteristics are calculated: flapping frequency, flapping amplitude, frequency of body oscillations, amplitude of body oscillations, phase difference between flapping and body oscillation, and average vertical velocity. In addition, three aerodynamic parameters were calculated: Reynolds number, Strouhal number and reduced frequency.

The data recorded over 75 flight segments and 9 different butterflies are tabulated with physical characteristics in Table 4-1. The specimens were chosen for a diverse representation of the butterflies tested, as well as the quality of the recorded data. The rest of the butterfly flight data will be post processed and analyzed in the future.

Table 4-1. Physical characteristics of the 9 butterflies compared in this thesis.

Butterfly #	Gender	Span wise wing length [mm]	Mass [g]
46	Male	53	0.55
54	Male	45	0.39
55	Male	47	0.47
56	Male	52	0.54
66	Female	49	0.49
67	Female	51	0.51
71	Male	52	0.52
72	Male	54	0.46
75	Female	50	0.46

Table 4-2 summarizes the six flight characteristics of interest. The data is presented as a mean and standard deviation for each butterfly. This provides information on the variation in flight characteristics that exist between different flight trajectories as well as different specimens.

Table 4-2. Mean and standard deviation of flight characteristics of 9 butterflies over 75 flights. The frequency of the body oscillations were found to be the same as that of the flapping frequency, and therefore they are listed as one entity.

Butterfly #	Vertical Velocity [mm/s]	Flapping Amplitude [o]	Flapping/Body Frequency [Hz]	Body Amplitude [mm]	Phase Difference [o]
46	476 ± 131	247 ± 8.72	9.57 ± 0.214	10.58 ± 0.578	92.6 ± 1.53
54	579 ± 201	276 ± 13.71	9.53 ± 0.349	9.77 ± 1.061	92.8 ± 2.67
55	367 ± 216	251 ± 13.35	9.99 ± 0.307	9.23 ± 0.688	93.0 ± 2.86
56	660 ± 175	264 ± 9.83	10.24 ± 0.326	9.69 ± 1.783	99.2 ± 6.18
66	387 ± 90	242 ± 15.77	9.38 ± 0.000	9.77 ± 0.936	81.7 ± 4.34
67	377 ± 98	239 ± 12.30	9.62 ± 0.358	11.31 ± 1.073	81.0 ± 3.72
71	418 ± 145	215 ± 24.34	9.99 ± 0.633	10.31 ± 1.963	87.8 ± 5.16
72	249 ± 82	226 ± 7.64	9.63 ± 0.320	9.81 ± 0.579	86.2 ± 2.25
75	635 ± 292	258 ± 14.91	10.79 ± 0.358	8.25 ± 0.971	90.6 ± 4.88

Flapping frequency remained fairly uniform between 9 Hz and 11 Hz. Also, the frequency of the undulating motion of the body was consistent with the flapping frequency, suggesting that the wing motion and the body motion are closely coupled to each other. The body of the butterfly is pulled up during the wing downstroke and undulates with a phase lag with the wing motion, as is clearly seen in Figure 4-9. Most flight dynamics models of flapping wing insects neglect the influence of wing mass and inertia on the body motion by assuming that the flapping frequency is much higher than that of the body oscillation and that wing mass is much smaller than the body mass [15,54]. However, for the considered Monarch butterflies, both flapping and body

frequencies are similar to each other and a simplified flight dynamics models may be inadequate to analyze their dynamics and stability.

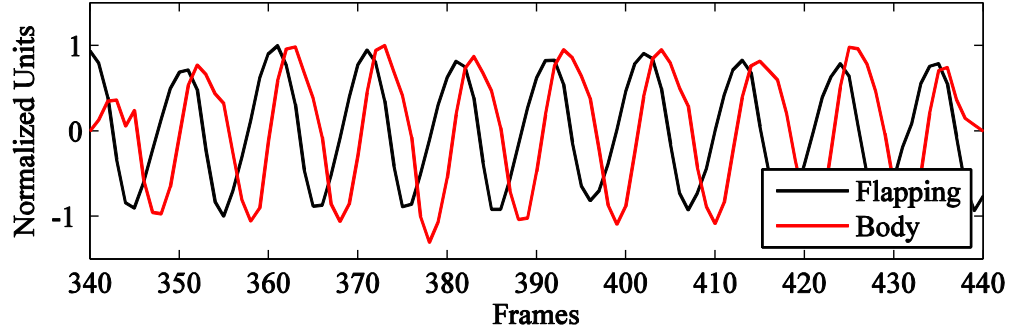


Figure 4-9. Time signal of the flapping and body oscillations normalized using the maximum value of each.

Three non-dimensional parameters were calculated to characterize the flapping wing aerodynamics of butterflies: the Reynolds number Re , Strouhal number St and reduced frequency k . The Reynolds number was calculated as

$$Re = \frac{U_{ref} L_{ref}}{\nu}, \quad (5)$$

where the free stream reference velocity U_{ref} is the average velocity of the flight segment, the reference length L_{ref} is measured as half the wing span, which is the length of one forewing root to tip, and the kinematic viscosity ν of air was found for 75°F at 1 atm. Reynolds number is a critical parameter in aerodynamics, giving the relative importance of fluid inertia and viscous effects.

The Strouhal number was calculated by

$$St = \frac{\gamma f L_{ref}}{2 U_{ref}}, \quad (6)$$

where f is the flapping frequency of the butterfly. The Strouhal number compares the velocity of the wing tip in flapping motion to the forward flight of the butterfly, providing a propulsive efficiency.

Finally, the reduced frequency was calculated as

$$k = \frac{\pi f L_{\text{ref}}}{U_{\text{ref}}} \quad (7)$$

Reduced frequency provides a characterization of the measure of unsteadiness by comparing spatial wavelength of flow disturbance to the wing span [14]. All dimensionless parameters were calculated from average values over the length of a particular data segment. The mean and standard deviation of these non-dimensional parameters were calculated for each of the nine butterflies and are shown in Table 4-3.

Table 4-3. Mean and standard deviation of non-dimensional parameters of 9 butterflies over 75 flights.

Butterfly #	Re	St	k
46	5999 ± 375	0.631±0.055	0.920±0.054
54	5156±215	0.587±0.049	0.767±0.036
55	5471±419	0.578±0.057	0.831±0.076
56	5737±724	0.742±0.131	1.010±0.162
66	5820±672	0.542±0.091	0.803±0.092
67	5173±511	0.664±0.092	0.996±0.090
71	5653±790	0.596±0.096	0.999±0.111
72	6842±632	0.531±0.055	0.847±0.075
75	6025±838	0.682±0.130	0.951±0.174

These non-dimensional parameters have been compared to values found in the literature and they are in reasonable agreement. This experimental method can be used in

the future to measure not only wing kinematics but also commonly used non-dimensional parameters to aid in creating more accurate numerical and analytic models.

The analysis presented in this chapter will provide a framework to retrieve and analyze data for future insect motion capture studies. Already, this method has been used in collaboration with Dr. Amy Lang at the University of Alabama to complement her experiments on the aerodynamic effects of butterfly scales. However, the data from that work falls outside of the scope of this thesis and therefore is not presented. It is the belief of the author that this analysis framework can provide valuable information about the design of MAVs.

CHAPTER 5

Conclusions

5.1 Summary

A novel technique utilizing turnkey motion capture software was developed to measure trajectories and wing kinematics of freely flying butterflies. The procurement, handling, and measurement of some physical characteristics of the specimens were presented. Reflective markers were modified in order to reduce their weight and effect on the flight of the butterfly. One marker placed on the thorax was dedicated to tracking the trajectory of the butterfly. Four markers placed on the top and bottom of the forewings provided the flapping data. Two markers placed on the top and bottom of one hind wing provided asymmetry for the processing software and a reference for calculation of the flapping signal. Over 86 butterflies were tested in more than 2,000 flight tests. A general tendency for the butterflies to climb was noted during testing, leading to an interest in the vertical position and velocity of the trajectory. The butterflies also demonstrated a prominent body oscillation in all flights. Methods for analyzing the data were developed to extract the flapping angle, body oscillation, and vertical trajectory from flight data. Information on frequency, amplitude, and coupling between the flapping and body oscillations are presented along with altitude and climbing rate.

Some experimental observations based on over 2,000 flight tests and 86 test specimens are as follows:

- The butterflies' natural tendency was to climb after being released, requiring continual flapping. Trajectories which included gliding flight were captured infrequently and mostly for butterflies with scales removed.
- A pronounced oscillation in body oscillation was noticed during all flights, but it was more pronounced in vertical trajectories with lower horizontal velocities.
- After approximately 10 flights, the butterflies began to noticeably change flight trajectories, exhibiting a lower vertical velocity. This was attributed to either tiring butterflies, or getting used to being handled. These effects were mitigated by only testing each butterfly 10 consecutive flight tests.
- Additional markers added to the forewings were poorly recorded. This may be due to the small size of the markers, and distance from the cameras compounded with effects from a change in incidence angle.

Based on a literature survey performed by the author, this study represents the first time that butterfly body undulation was quantified. The mean trend which is used to extract the body undulation information can also be used to compare flight trajectories to changes to wing kinematics. Based on a detailed analysis of 75 flight segments observed over 9 different butterflies, the following observations were made:

- The flapping angle γ and body undulations were approximately sinusoidal with respect to time.
- The flapping frequency was measured between 9 Hz and 11 Hz. For all flights analyzed, the frequency of the flapping angle was equal to that of the body undulations indicating a coupling between the two characteristics.

- A nonzero phase offset between wing and body was recorded for every butterfly between 80° and 100° . The variation of this phase offset was more prominent between specimens than between flight trials of the same specimens. This indicates that this parameter represents physical characteristics as opposed to flight characteristics.
- The flapping frequency was observed to change slightly, even during apparently similar climbing rates.
- The flapping amplitude was found to vary significantly between flights and butterflies. A loose correlation between flapping amplitude and climb rate was noted which is consistent with what is found in the literature.
- Body undulation amplitude was found to vary between 5 mm and 15 mm from peak to peak for all butterflies. This oscillation represents approximately 10% of the wingspan.
- The values of dimensionless variables presented in this thesis compare well to what are reported in a survey of the literature conducted by the author.

5.2 Limitations, Consequences, and Implications

The wing kinematics of a butterfly is too complicated to be fully characterized with one marker on each forewing. The pitching angle of the wings, which have been shown to have significant effects on lift generation, cannot be determined. Information on the deformation of wings is also not available with the current marker configuration. The large size of the capture volume requires the use of relatively large markers and sufficient spacing between them. If the markers are too small and close together, then the cameras will struggle to distinguish separate markers. Insects with large wings must be used to ensure sufficient marker size and spacing. The frame rate achievable by the system also limits the wing beat frequency of the specimen to be

studied. Large Lepidopteran with relatively slow wing beat frequencies are well suited to this experimental method. These characteristics along with a simplified flapping motion also make these good candidates for mimicry in MAV design.

The analysis techniques presented in this thesis can be used to calculate average frequency, amplitude and phase information of flapping angle and body undulations. This information can also be collected for multiple different trajectory types such as steady forward flight, turning flight, descending flight or accelerating flight. The resolution of these calculations as well as the accuracy are dependent upon the frame rate as well as the tendency for the butterfly to change frequency during data segments. An increase in frame rate is accompanied by challenges including increasing post processing time, and potentially negatively affecting the quality of data (more disappearing markers). A flap-by-flap analysis is easily implemented in the analysis framework as the peak definition algorithm provides a reliable method to segment data per flap.

This experimental method was able to calculate three flight parameters - body undulation frequency, amplitude and phase - which have not been studied up to this point in Lepidopteran. . The information provided from these parameters can be used to create more realistic dynamic models which take body oscillation into consideration.

5.3 Future Work

Observing the wing kinematics of relatively large Lepidopteran in the context of their undirected trajectories has not been possible to this point. This thesis provides an indepth description of experiments which can be further refined to study any number of interesting flight phenomena. The method for analyzing the data from these experiments also provides a framework that can be built upon in future experiments. Some recommendations for future work are:

1. Increase frame rate to at least 200 Hz to increase the resolution of the time averaged analysis. An analysis which calculates the six flight characteristics during every flap can be developed and compared with the time averaged characteristics. This investigation could be used to determine whether changes in flapping frequency and amplitude during similar trajectories are relevant to the flight trajectory, or if they occur randomly due to unpredictable animal behavior.
2. Reduce capture volume size and add additional markers to forewings. The size of the capture volume affects the resolution of the images the cameras collect of each marker. As this resolution increases, multiple markers can be placed closer together without being mistaken for a single marker. These additional markers could be utilized to calculate more kinematic parameters such as changes in pitch angle, and potentially measure the deflection of the wing during flight. The effects of these parameters have been widely studied and linked to lift generation in some insects, including butterflies.
3. This experimental set up can also be used to explore the effects of and response to wind gusts by insects. This information can be invaluable to developing control algorithms that deal with wind gusts, which pose a major challenge to MAV development.
4. Comparing Monarch flight to other Lepidopteran members can also provide useful information. The differing flight strategies of migratory and non-migratory species can provide insight into flight adaptations and their implications for flight behavior. A migratory species such as the Monarch relies on efficiency to successfully complete its migration. This may significantly differ from a species that does not migrate but has significantly more predators where maneuverability is critical.
5. Develop a method to compare relatively nonlinear flight trajectories in three dimensions. Three dimensional effects of forward flight velocity which are not accounted for in this thesis could significantly affect the climbing rate. A change in horizontal direction

realized in a turn would also require different wing kinematics than those used in solely forward climbing and steady flight.

APPENDIX A

MATLAB Code

```
function varargout = Butterfly_GUI(varargin)
% BUTTERFLY_GUI M-file for Butterfly_GUI.fig
%     BUTTERFLY_GUI, by itself, creates a new
BUTTERFLY_GUI or raises the existing
%     singleton*.
%
%     H = BUTTERFLY_GUI returns the handle to a new
BUTTERFLY_GUI or the handle to
%     the existing singleton*.
%
%
% BUTTERFLY_GUI('CALLBACK',hObject,eventData,handles,...)
calls the local
%     function named CALLBACK in BUTTERFLY_GUI.M with the
given input arguments.
%
%     BUTTERFLY_GUI('Property','Value',...) creates a new
BUTTERFLY_GUI or raises the
%     existing singleton*. Starting from the left,
property value pairs are
%     applied to the GUI before Butterfly_GUI_OpeningFcn
gets called. An
%     unrecognized property name or invalid value makes
property application
%     stoData.Pos. All inputs are passed to
Butterfly_GUI_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI
allows only one
%     instance to run (singleton)".
```

```

%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Butterfly_GUI

% Last Modified by GUIDE v2.5 28-Apr-2015 14:27:03

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @Butterfly_GUI_OpeningFcn, ...
                  'gui_OutputFcn',  @Butterfly_GUI_OutputFcn, ...
                  'gui_LayoutFcn',  [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Butterfly_GUI is made visible.
function Butterfly_GUI_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to Butterfly_GUI (see
VARARGIN)

% Choose default command line output for Butterfly_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

```

```

% UIWAIT makes Butterfly_GUI wait for user response (see
UIRESUME)
% uiwait(handles.figure1);
cla(handles.axes1)
cla(handles.axes2)
cla(handles.axes3)
cla(handles.axes4)
set(handles.axes1, 'visible', 'off')
set(handles.axes2, 'visible', 'off')
set(handles.axes3, 'visible', 'off')
set(handles.axes4, 'visible', 'off')
set(handles.popup, 'string', {'None'; 'Cubic-spline'; 'Auto
Smooth'; 'Manual Smooth'; 'Dimensionless'})
set(handles.text7, 'Visible', 'off')
set(handles.text94, 'Visible', 'off')
set(handles.span, 'Visible', 'off')
set(handles.mass, 'Visible', 'off')
legend(handles.axes1, 'hide')
legend(handles.axes2, 'hide')
legend(handles.axes3, 'hide')
legend(handles.axes4, 'hide')

% --- Outputs from this function are returned to the
command line.
function varargout = Butterfly_GUI_OutputFcn(hObject,
eventdata, handles)
% varargout    cell array for returning output args (see
VARARGOUT);
% hObject      handle to figure
% eventdata    reserved - to be defined in a future version
of MATLAB
% handles      structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in import.
function import_Callback(hObject, eventdata, handles)
% hObject      handle to import (see GCBO)
% eventdata    reserved - to be defined in a future version
of MATLAB
% handles      structure with handles and user data (see
GUIDATA)
global GUI Data

```

```

Funs = ButterflyGUIFuns;
[Data ,GUI] = Funs.import(GUI,handles);
set(handles.axes1,'visible','on')
set(handles.axes2,'visible','on')
set(handles.axes3,'visible','on')
set(handles.axes4,'visible','on')

% --- Executes on button press in browse.
function browse_Callback(hObject, eventdata, handles)
% hObject    handle to browse (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global GUI
[GUI.file GUI.path] = uigetfile('C:\Users\Jake\Google
Drive\ATOM\Butterfly\Data/*.csv');
[GUI.pathname GUI.filename] = fileparts([GUI.path
GUI.file]);
set(handles.filename,'String',GUI.filename)

function filename_Callback(hObject, eventdata, handles)
% hObject    handle to filename (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of filename
as text
%         str2double(get(hObject,'String')) returns contents
of filename as a double

% --- Executes during object creation, after setting all
properties.
function filename_CreateFcn(hObject, eventdata, handles)
% hObject    handle to filename (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%         See ISPC and COMPUTER.

```

```

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in linterData.Pos.
function popup_Callback(hObject, eventdata, handles)
% hObject      handle to popup (see GCBO)
% eventdata    reserved - to be defined in a future version
of MATLAB
% handles      structure with handles and user data (see
GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popup contents as cell array
%           contents{get(hObject,'Value')} returns selected
item from popup

switch get(handles.popup,'value')
    case 1
        set(handles.text7,'Visible','off')
        set(handles.text94,'Visible','off')
        set(handles.span,'Visible','off')
        set(handles.mass,'Visible','off')
    case 2
        set(handles.text7,'Visible','off')
        set(handles.text94,'Visible','off')
        set(handles.text7,'Visible','off')
        set(handles.text94,'Visible','off')
        set(handles.span,'Visible','off')
        set(handles.mass,'Visible','off')
    case 3
        set(handles.text7,'Visible','off')
        set(handles.text94,'Visible','off')
        set(handles.text7,'Visible','off')
        set(handles.text94,'Visible','off')
        set(handles.span,'Visible','off')
        set(handles.mass,'Visible','off')
    case 4
        set(handles.text7,'Visible','off')
        set(handles.text94,'Visible','on')
        set(handles.text7,'String','Vel Span')
        set(handles.text94,'String','Pos Span')
        set(handles.span,'Visible','on')
        set(handles.mass,'Visible','off')
    case 5
        set(handles.text7,'Visible','on')

```



```

        set(handles.text94, 'Visible', 'on')
        set(handles.text7, 'String', 'Mass (g)')
        set(handles.text94, 'String', 'Wing Span (mm)')
        set(handles.span, 'Visible', 'on')
        set(handles.mass, 'Visible', 'on')
end

% --- Executes during object creation, after setting all
properties.
function popup_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popup (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: popupmenu controls usually have a white background
on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function start_Callback(hObject, eventdata, handles)
% hObject    handle to start (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject, 'String') returns contents of start as
text
%         str2double(get(hObject, 'String')) returns contents
of start as a double

% --- Executes during object creation, after setting all
properties.
function start_CreateFcn(hObject, eventdata, handles)
% hObject    handle to start (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

```

```

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function finish_Callback(hObject, eventdata, handles)
% hObject    handle to finish (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of finish
as text
%       str2double(get(hObject,'String')) returns contents
of finish as a double

% --- Executes during object creation, after setting all
properties.
function finish_CreateFcn(hObject, eventdata, handles)
% hObject    handle to finish (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in apply.
function apply_Callback(hObject, eventdata, handles)
% hObject    handle to apply (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

```

```

global GUI Data
Funs = ButterflyGUIFuns;
switch get(handles.popup, 'Value')
    case 1
        [ Data, GUI ] = Funs.limits(GUI, Data, handles);
    case 2
        [ Data, GUI ] = Funs.filter(GUI, Data, handles);
    case 3
        [ Data, GUI ] = Funs.autosmooth(GUI, Data,
handles);
    case 4
        [ Data, GUI ] = Funs.mansmooth(GUI, Data, handles);
    case 5
        [ Data, GUI ] = Funs.dimensionless(GUI, Data,
handles);
end

```

```

% --- Executes on button press in images.
function images_Callback(hObject, eventdata, handles)
% hObject    handle to images (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

```

```

global GUI Data
fun = ButterflyGUIFuns;
switch get(handles.popup, 'Value');
    case 1
        Butterfly_analysis_GUIcode_export_limits
    case 2
        fun.filterplot(GUI, Data)
    case 3
        fun.smoothplot(GUI, Data)
end

```

```

% --- Executes on button press in ENERGYcalc.
function ENERGYcalc_Callback(hObject, eventdata, handles)
% hObject    handle to ENERGYcalc (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

```

```

global GUI

```

```

GUI.number =
str2double(get(handles.butterflyNumber, 'string'));
Butterfly_Test

function butterflyNumber_Callback(hObject, eventdata,
handles)
% hObject    handle to butterflyNumber (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject, 'String') returns contents of
butterflyNumber as text
%          str2double(get(hObject, 'String')) returns contents
of butterflyNumber as a double

% --- Executes during object creation, after setting all
properties.
function butterflyNumber_CreateFcn(hObject, eventdata,
handles)
% hObject    handle to butterflyNumber (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%          See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

% --- Executes during object deletion, before destroying
properties.
function axes1_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

```

```

% --- Executes during object creation, after setting all
properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

function span_Callback(hObject, eventdata, handles)
% hObject    handle to span (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of span as
text
%          str2double(get(hObject,'String')) returns contents
of span as a double

% --- Executes during object creation, after setting all
properties.
function span_CreateFcn(hObject, eventdata, handles)
% hObject    handle to span (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%          See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function mass_Callback(hObject, eventdata, handles)
% hObject    handle to mass (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

```

```

% Hints: get(hObject,'String') returns contents of mass as
text
%         str2double(get(hObject,'String')) returns contents
of mass as a double

% --- Executes during object creation, after setting all
properties.
function mass_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mass (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function out = ButterflyGUIFuncs
out.energy          = @energy;
out.thesis          = @thesis;
out.phase           = @PhaseDiff;
out.perflap         = @thesis_perflap;
out.table           = @Maketable;
out.import          = @import;
out.limits          = @limits;
out.filter          = @filter;
out.autosmooth      = @AutoSmooth;
out.mansmooth       = @ManualSmooth;
out.amplitude       = @PEAKtoPEAK;
out.filterplot      = @filterplot;
out.smoothplot      = @smoothplot;
out.autospan        = @AutoSpan;
out.dimensionless   = @dimensionless;
out.importbatch     = @importBatch;
out.mansmoothbatch  = @ManualSmoothBatch;
out.filterbatch     = @filterBatch;
out.perflapbatch    = @thesisperflapbatch;
out.batch           = @Batch;
out.fft             = @FFT;

function [ Data, GUI ] = import(GUI,handles)

```

```

fun = ButterflyGUIFun;

% read data using path and filename specified during browse
GUI.data = xlsread([GUI.path GUI.filename '.CSV']);

% clear all axes
cla(handles.axes1)
cla(handles.axes2)
cla(handles.axes3)
cla(handles.axes4)

% use while loop to separate data into usable variables
% while loop is conditioned so that it breaks when there is
not a number
%   in the frame column. This only occurs at the end of a
data collection
%   segment
i = 5;
j = 1;
while isnan(GUI.data(i,1)) == 0
    Data.Pos.frame(j,1) = GUI.data(i,1);

    Data.Pos.Headx(j,1) = GUI.data(i,3);
    Data.Pos.Heady(j,1) = GUI.data(i,4);
    Data.Pos.Headz(j,1) = GUI.data(i,5);

    Data.Pos.LeftWingx(j,1) = GUI.data(i,6);
    Data.Pos.LeftWingy(j,1) = GUI.data(i,7);
    Data.Pos.LeftWingz(j,1) = GUI.data(i,8);

    Data.Pos.RightWingx(j,1) = GUI.data(i,9);
    Data.Pos.RightWingy(j,1) = GUI.data(i,10);
    Data.Pos.RightWingz(j,1) = GUI.data(i,11);

    Data.Pos.LowerWingx(j,1) = GUI.data(i,12);
    Data.Pos.LowerWingy(j,1) = GUI.data(i,13);
    Data.Pos.LowerWingz(j,1) = GUI.data(i,14);

    i = i+1;
    j = j+1;
end

clear i j

% remove velocity headers from import and only extract head
information
i = length(Data.Pos.frame)+12;

```

```

j = 1;

for k = 1:length(GUI.data)
    Data.Vel.frame(j,1) = GUI.data(k,1);

    Data.Vel.Headxdot(j,1) = GUI.data(k,3);
    Data.Vel.Headydot(j,1) = GUI.data(k,4);
    Data.Vel.Headzdot(j,1) = GUI.data(k,5);

    j = j+1;
end

clear i j k

%forming 3-D vectors for position and velocity
% distance from left wing to head
Data.Pos.v1x = Data.Pos.LeftWingx - Data.Pos.Headx;
Data.Pos.v1y = Data.Pos.LeftWingy - Data.Pos.Heady;
Data.Pos.v1z = Data.Pos.LeftWingz - Data.Pos.Headz;
Data.Pos.v1 = [Data.Pos.v1x Data.Pos.v1y Data.Pos.v1z]';

% distance from right wing to head
Data.Pos.v2x = Data.Pos.RightWingx - Data.Pos.Headx;
Data.Pos.v2y = Data.Pos.RightWingy - Data.Pos.Heady;
Data.Pos.v2z = Data.Pos.RightWingz - Data.Pos.Headz;
Data.Pos.v2 = [Data.Pos.v2x Data.Pos.v2y Data.Pos.v2z]';

% distance from lower left wing to head
Data.Pos.v3x = Data.Pos.LowerWingx - Data.Pos.Headx;
Data.Pos.v3y = Data.Pos.LowerWingy - Data.Pos.Heady;
Data.Pos.v3z = Data.Pos.LowerWingz - Data.Pos.Headz;
Data.Pos.v3 = [Data.Pos.v3x Data.Pos.v3y Data.Pos.v3z]';

GUI.dt = 1/GUI.data(1,1);

for i = 1:length(Data.Pos.frame)

    % flapping angle
    % calculate angle between two vectors
    Data.Pos.angle(i) =
atan2(norm(cross(Data.Pos.v1(:,i),Data.Pos.v2(:,i))),dot(Data
ta.Pos.v1(:,i),Data.Pos.v2(:,i)));

    % establish sign
    Data.Pos.crossproduct(i) =
dot(cross(Data.Pos.v1(:,i),Data.Pos.v2(:,i)),Data.Pos.v3(:,
i));
    if Data.Pos.crossproduct(i) > 0

```



```

        Data.Pos.angleindegree(i) = 360-
Data.Pos.angle(i)*180/pi;
        elseif Data.Pos.crossproduct(i) == 0
            Data.Pos.angleindegree(i) =
Data.Pos.angle(i)*180/pi;
        elseif Data.Pos.crossproduct(i) < 0
            Data.Pos.angleindegree(i) =
Data.Pos.angle(i)*180/pi;
        elseif isnan(Data.Pos.crossproduct(i))==1
            Data.Pos.angleindegree(i) = 0;
        end
        % establish time vector
        if i == 1
            Data.Pos.t(i) = 0;
        else
            Data.Pos.t(i) = Data.Pos.t(i-1)+ GUI.dt;
        end
    end
end

for j = 1:length(Data.Vel.frame)

    % calculate horizontal velocity
    Data.Vel.horizontalvel(:,j) =
[Data.Vel.Headxdot(j);Data.Vel.Headydot(j)];
    Data.Vel.horizontalvel(j) =
norm(Data.Vel.horizontalvel(:,j),2);

    % calculate flight direction
    Data.Vel.direction(j) =
atan2(Data.Vel.Headxdot(j),Data.Vel.Headydot(j));

    % establish time vector
    if j == 1
        Data.Vel.t(j) = 0;
    else
        Data.Vel.t(j) = Data.Vel.t(j-1)+ GUI.dt;
    end
end

end

% on import, the entire data segment is displayed for
segmenting later
GUI.start = 1;
GUI.finish = length(Data.Pos.frame);
GUI.intv = GUI.start:GUI.finish;

% remove 180 deg jumps in velocity data
Data.Vel.direction = unwrap(Data.Vel.direction);
Data.Vel.directionindegree = Data.Vel.direction*180/pi;

```

```

% [ maxFFTfreq, maxFFTindex , f, Y ]
[ ~,~,Data.Pos.f, Data.Pos.Y ] = fun.fft(
Data.Pos.angleindegree );

% average velocity
Data.Vel.Headzdotavg = Data.Vel.Headzdot;
Data.Vel.horizontalsspeedavg = Data.Vel.horizontalsspeed;

% remove instances where head data does not exist so an
average can
% be taken
Data.Vel.horizontalsspeedavg(isnan(Data.Vel.horizontalsspeeda
vg)==1) = [];
Data.Vel.Headzdotavg(isnan(Data.Vel.Headzdotavg)==1) = [];

% ensure that velocity data is not shorter than desired
interval
if length(GUI.intv)>length(Data.Vel.Headzdotavg)
    GUI.intvavg = 1:length(Data.Vel.Headzdotavg);
else
    GUI.intvavg = GUI.intv;
end
Data.Vel.AVGhorizontal =
sum(Data.Vel.horizontalsspeedavg(GUI.intvavg)) ...
    /length(Data.Vel.horizontalsspeed(GUI.intvavg));
Data.Vel.AVGvertical =
sum(Data.Vel.Headzdotavg(GUI.intvavg)) ...
    /length(Data.Vel.Headzdot(GUI.intvavg));

% Plot angle between wing markers
hold(handles.axes1,'off')
plot(handles.axes1,Data.Pos.frame(GUI.intv),Data.Pos.anglei
ndegree(GUI.intv),'k')
xlabel(handles.axes1,'Frame (s)')
ylabel(handles.axes1,'Flapping Angle (deg)')
grid(handles.axes1,'on')
hold(handles.axes1,'off')
% Plot FFT of angle between wings
plot(handles.axes2,Data.Pos.f,Data.Pos.Y)
xlabel(handles.axes2,'Frequency (Hz)')
ylabel(handles.axes2,'|Y(f)|')
grid(handles.axes2,'on')

% Plot average horizontal and vertical speeds
plot(handles.axes3,Data.Vel.frame(GUI.intvavg),Data.Vel.hor
izontalspeed(GUI.intvavg),'r',Data.Vel.frame,Data.Vel.Headz
dot)
xlabel(handles.axes3,'Frame')

```

```

ylabel(handles.axes3, 'Magnitude of Velocity (mm/s)')
legend(handles.axes3, 'Horizontal', 'Vertical', 'Location', 'SW
')
grid(handles.axes3, 'on')

plot(handles.axes4, Data.Pos.frame(GUI.intv), Data.Pos.Headz (
GUI.intv))
ylabel(handles.axes4, 'z-axis position (mm)')
xlabel(handles.axes4, 'Frame')
grid(handles.axes4, 'on')

function [ Data, GUI ] = limits(GUI, Data, handles)

% initialize the use of previously defined functions
fun = ButterflyGUIFuncs;

% get bounding frames for the interval of interest
GUI.start = str2double(get(handles.start, 'string'));
GUI.finish = str2double(get(handles.finish, 'string'));

% find index that correspond to the desired frame interval
GUI.startindex = find(Data.Pos.frame == GUI.start, 1);
GUI.finishindex = find(Data.Pos.frame == GUI.finish, 1);
GUI.intv = GUI.startindex:GUI.finishindex;

% remove 180 deg jumps in data
Data.Vel.direction = unwrap(Data.Vel.direction);
Data.Vel.directionindegree = Data.Vel.direction*180/pi;

% calculate the fast fourier transform using the function
FFT
[ Data, GUI ] = fun.fft( GUI, Data );

% average velocity
% define new variables
Data.Vel.Headzdotavg = Data.Vel.Headzdot;
Data.Vel.horizontalsspeedavg = Data.Vel.horizontalsspeed;

% remove data where head marker is not present
Data.Vel.horizontalsspeedavg(isnan(Data.Vel.horizontalsspeeda
vg)==1) = [];
Data.Vel.Headzdotavg(isnan(Data.Vel.Headzdotavg)==1) = [];
GUI.intvavg = GUI.startindex:GUI.finishindex;

% calculate averages
Data.Vel.AVGhorizontal =
mean(Data.Vel.horizontalsspeedavg(GUI.intvavg));

```

```

Data.Vel.AVGvertical =
mean(Data.Vel.Headzdotavg(GUI.intvavg));

% generate plots to gui axes
hold(handles.axes1,'off')
plot(handles.axes1,Data.Pos.frame(GUI.intv),Data.Pos.anglei
ndegree(GUI.intv),'k')
xlabel(handles.axes1,'Frame')
ylabel(handles.axes1,'Flapping Angle (deg)')
set(handles.axes1,'xlim',[GUI.start GUI.finish])
grid(handles.axes1,'on')
hold(handles.axes1,'off')

% Plot FFT of angle between wings
hold(handles.axes2,'off')
plot(handles.axes2,Data.Pos.f(GUI.LPF:GUI.NFFT/2+1),2*abs(D
ata.Pos.Y(GUI.LPF:GUI.NFFT/2+1)))
xlabel(handles.axes2,'Frequency (Hz)')
ylabel(handles.axes2,'|Y(f)|')
set(handles.axes2,'ylim',[0 200])
grid(handles.axes2,'on')
hold(handles.axes2,'off')

% Plot average horizontal and vertical speeds
hold(handles.axes3,'off')
plot(handles.axes3,Data.Pos.frame(GUI.intv),Data.Vel.horizo
ntalspeed(GUI.intv),'r',Data.Pos.frame(GUI.intv),Data.Vel.H
eadzdot(GUI.intv))
xlabel(handles.axes3,'Frame')
ylabel(handles.axes3,'Magnitude of Velocity (mm/s)')
legend(handles.axes3,'Horizontal','Vertical','Location','SW
')
grid(handles.axes3,'on')
set(handles.axes3,'xlim',[GUI.start GUI.finish])
hold(handles.axes3,'off')

hold(handles.axes4,'off')
plot(handles.axes4,Data.Pos.frame(GUI.intv),Data.Pos.Headz(
GUI.intv))
xlabel(handles.axes4,'Frame')
ylabel(handles.axes4,'Vertical displacement(mm)')
grid(handles.axes4,'on')
set(handles.axes4,'Xlim',[GUI.start GUI.finish])
hold(handles.axes4,'off')

```

```

function [ Data, GUI ] = filter(GUI, Data, handles)
% initialize the use of other functions
fun = ButterflyGUIFuncs;

% gather data from gui text boxes
GUI.start = str2double(get(handles.start,'string'));
GUI.finish = str2double(get(handles.finish,'string'));
Data.Pos.Span = str2double(get(handles.span,'string'));
Data.Vel.Span = str2double(get(handles.mass,'string'));

% define the index interval for the frames of interest
GUI.startindex = find(Data.Pos.frame == GUI.start,1);
GUI.finishindex = find(Data.Pos.frame == GUI.finish,1);
GUI.intv = GUI.startindex:GUI.finishindex;

% define new variables for wing angle (x) and frame (t)
x = Data.Pos.angleindegree(GUI.intv);
t = Data.Pos.frame(GUI.intv);

% remove all frames where wing angle is equal to 0
t(x == 0) = [];
x(x == 0) = [];

% create a time increment for the spline interpolation
tinc = GUI.data(1,1)/100;
Data.Pos.frame_spl = GUI.start:tinc:GUI.finish;

% calculate interpolation
Data.Pos.anglefilt = spline(t,x,Data.Pos.frame_spl);

% amplitude of wing angle using PEAKtoPEAK
[ Data.Pos.Wing.Max, Data.Pos.Wing.Min ] =
fun.amplitude(Data.Pos.frame_spl, Data.Pos.anglefilt);

% remove 180 deg jumps in data
Data.Vel.direction = unwrap(Data.Vel.direction);
Data.Vel.directionindegree = Data.Vel.direction*180/pi;

% calculate fast fourier transform using function FFT
[ Data, GUI ] = fun.fft( GUI ,Data );

% average velocity
% define new variables
Data.Vel.Headzdotavg = Data.Vel.Headzdot;
Data.Vel.horizontalavg = Data.Vel.horizontal;

% remove data where head marker is not present

```

```

Data.Vel.horizontalaverage(isnan(Data.Vel.horizontalaverage)
==1) = [];
Data.Vel.Headzdotaverage(isnan(Data.Vel.Headzdotaverage)
==1) = [];
GUI.intvavg = GUI.startindex:GUI.finishindex;

% calculate averages
Data.Vel.AVGhorizontal =
mean(Data.Vel.horizontalaverage(GUI.intvavg));
Data.Vel.AVGvertical =
mean(Data.Vel.Headzdotaverage(GUI.intvavg));

% clear axes
cla(handles.axes1)
cla(handles.axes2)
cla(handles.axes3)
cla(handles.axes4)

% Plot angle between wing markers
hold(handles.axes1,'off')
plot(handles.axes1,Data.Pos.frame_spl,Data.Pos.anglefilt,'k'
')
hold(handles.axes1,'on')
plot(handles.axes1,Data.Pos.Wing.Max(:,1),Data.Pos.Wing.Max
(:,2),'xr')
plot(handles.axes1,Data.Pos.Wing.Min(:,1),Data.Pos.Wing.Min
(:,2),'ro')
text(GUI.start,10,num2str(mean(Data.Pos.Wing.Max(:,2))- ...
    mean(Data.Pos.Wing.Min(:,2))), 'Parent',handles.axes1)
xlabel(handles.axes1,'Frame')
ylabel(handles.axes1,'Flapping Angle (deg)')
set(handles.axes1,'xlim',[GUI.start GUI.finish])
grid(handles.axes1,'on')
hold(handles.axes1,'on')

% Plot FFT of angle between wings
hold(handles.axes2,'off')
plot(handles.axes2,Data.Pos.f(GUI.LPF:GUI.NFFT/2+1),2*abs(D
ata.Pos.Y(GUI.LPF:GUI.NFFT/2+1)))
hold(handles.axes2,'on')
plot(handles.axes2,Data.Pos.maxFFTfreq,Data.Pos.maxFFT,'xr'
)
text(floor(Data.Pos.maxFFTfreq)+2,Data.Pos.maxFFT, ...
    num2str(Data.Pos.maxFFTfreq), 'Parent',handles.axes2)
xlabel(handles.axes2,'Frequency (Hz)')
ylabel(handles.axes2,'|Y(f)|')
set(handles.axes2,'ylim',[0 200])
set(handles.axes2,'xlim',[0 50])
grid(handles.axes2,'on')

```

```

hold(handles.axes2, 'off')

% Plot average horizontal and vertical speeds
hold(handles.axes3, 'off')
plot(handles.axes3, Data.Pos.frame(GUI.intv), Data.Vel.horizontalalspeed(GUI.intv), 'r', Data.Pos.frame(GUI.intv), Data.Vel.Headzdot(GUI.intv))
xlabel(handles.axes3, 'Frame')
ylabel(handles.axes3, 'Magnitude of Velocity (mm/s)')
legend(handles.axes3, 'Horizontal', 'Vertical', 'Location', 'SW')
grid(handles.axes3, 'on')
set(handles.axes3, 'xlim', [GUI.start GUI.finish])
hold(handles.axes3, 'off')

hold(handles.axes4, 'off')
plot(handles.axes4, Data.Pos.frame(GUI.intv), Data.Pos.Headz(GUI.intv))
xlabel(handles.axes4, 'Frame')
ylabel(handles.axes4, 'Vertical displacement(mm)')
set(handles.axes4, 'xlim', [GUI.start GUI.finish])
grid(handles.axes4, 'on')
hold(handles.axes4, 'off')

function [ Data, GUI ] = ManualSmooth(GUI, Data, handles)

% initialize the use of other functions
fun = ButterflyGUIFuncs;

% gather data from gui text boxes
GUI.start = str2double(get(handles.start, 'string'));
GUI.finish = str2double(get(handles.finish, 'string'));
Data.Pos.Span = str2double(get(handles.span, 'string'));
Data.Vel.Span = str2double(get(handles.mass, 'string'));

% define the index interval for the frames of interest
GUI.startindex = find(Data.Pos.frame == GUI.start, 1);
GUI.finishindex = find(Data.Pos.frame == GUI.finish, 1);
GUI.intv = GUI.startindex:GUI.finishindex;

% create extra variables for the frame, position and velocity of head
% marker
f = Data.Vel.frame(GUI.intv);
fP = Data.Pos.frame(GUI.intv);
zdotspl = Data.Vel.Headzdot(GUI.intv);
zspl = Data.Pos.Headz(GUI.intv);

```

```

% remove data where the head marker position and velocity
are not available
f(isnan(zdotspl)==1)=[];
zdotspl(isnan(zdotspl)==1)=[];
fP(isnan(zspl) == 1)=[];
zspl(isnan(zspl)==1)=[];

% create a spline interpolation for every frame on the
interval
Data.Vel.Headzdotspline =
spline(f,zdotspl,GUI.start:GUI.finish);
Data.Pos.Headzspline =
spline(fP,zspl,GUI.start:GUI.finish);

% moving average filter over position data using input from
GUI textbox
Data.Pos.smooth = smooth(Data.Pos.Headzspline,
Data.Pos.Span);

% differentiate smoothed data with respect to time for
comparison with
% measured velocity
Data.Vel.smooth = diff(Data.Pos.smooth)/0.01;
Data.Vel.smooth = [Data.Vel.smooth; Data.Vel.smooth(end)];

% subtract smoothed data from the initial data
Data.Pos.Smooth.undulation = Data.Pos.Headzspline'-
Data.Pos.smooth;

% calculate peaks using the function PEAKtoPEAK
[ Data.Pos.Smooth.Max Data.Pos.Smooth.Min ] =
fun.amplitude( ...
    Data.Pos.frame(GUI.intv), Data.Pos.Smooth.undulation);

% calculate fast fourier transform using function FFT
[ Data, GUI ] = fun.fft( GUI ,Data );

% generating plots on each of the axes
hold(handles.axes1,'off')
plot(handles.axes1,Data.Pos.frame(GUI.intv),
Data.Pos.Smooth.undulation)
hold(handles.axes1,'on')
plot(handles.axes1,Data.Pos.Smooth.Max(:,1),Data.Pos.Smooth
.Max(:,2),'xr')
plot(handles.axes1,Data.Pos.Smooth.Min(:,1),Data.Pos.Smooth
.Min(:,2),'ro')
line(get(handles.axes1,'Xlim'),[mean(Data.Pos.Smooth.undula
tion) ...

```



```

mean(Data.Pos.Smooth.undulation)], 'Parent', handles.axes1)
set(handles.axes1, 'xlim', [GUI.start GUI.finish])
xlabel(handles.axes1, 'Frame')
ylabel(handles.axes1, 'Vertical displacement (mm)')
grid(handles.axes1, 'on')
hold(handles.axes1, 'off')

hold(handles.axes2, 'off')
plot(handles.axes2, Data.Pos.Smooth.f(GUI.LPF:GUI.NFFT/2+1),
2*abs(Data.Pos.Smooth.Y(GUI.LPF:GUI.NFFT/2+1)))
hold(handles.axes2, 'on')
plot(handles.axes2, Data.Pos.Smooth.maxFFTFreq, Data.Pos.Smooth.ma
xFFT, 'xr')
text(floor(Data.Pos.Smooth.maxFFTFreq)+2, Data.Pos.Smooth.ma
xFFT, ...

num2str(Data.Pos.Smooth.maxFFTFreq), 'Parent', handles.axes2)
xlabel(handles.axes2, 'Frequency (Hz)')
ylabel(handles.axes2, '|Y(f)|')
set(handles.axes2, 'ylim', [0 10])
set(handles.axes2, 'xlim', [0 50])
grid(handles.axes2, 'on')
hold(handles.axes2, 'off')

hold(handles.axes3, 'off')
plot(handles.axes3,
Data.Vel.frame(GUI.intv), Data.Vel.Headzdot(GUI.intv), ...
Data.Vel.frame(GUI.intv), Data.Vel.smooth)
xlabel(handles.axes3, 'Frame')
ylabel(handles.axes3, 'Vertical velocity (mm/s)')
set(handles.axes3, 'xlim', [GUI.start GUI.finish])
grid(handles.axes3, 'on')
legend(handles.axes3, 'Data', ['Span '
num2str(Data.Vel.Span)], 'location', 'SouthEast')
hold(handles.axes3, 'off')

hold(handles.axes4, 'off')
plot(handles.axes4, Data.Pos.frame(GUI.intv), Data.Pos.Headz(
GUI.intv), ...
Data.Pos.frame(GUI.intv), Data.Pos.smooth);
set(handles.axes4, 'xlim', [GUI.start GUI.finish])
grid(handles.axes4, 'on')
xlabel(handles.axes4, 'Frame')
ylabel(handles.axes4, 'Vertical displacement (mm)')
legend(handles.axes4, 'Data', ['Span '
num2str(Data.Pos.Span)], 'location', 'SouthEast')
hold(handles.axes4, 'off')

```

```

function [ Data, GUI ] = AutoSmooth(GUI, Data, handles)

% initialize the use of other functions
fun = ButterflyGUIFuncs;

% gather data from gui text boxes
GUI.start = str2double(get(handles.start, 'string'));
GUI.finish = str2double(get(handles.finish, 'string'));
Data.Pos.Span = str2double(get(handles.span, 'string'));
Data.Vel.Span = str2double(get(handles.mass, 'string'));

% define the index interval for the frames of interest
GUI.startindex = find(Data.Pos.frame == GUI.start, 1);
GUI.finishindex = find(Data.Pos.frame == GUI.finish, 1);
GUI.intv = GUI.startindex:GUI.finishindex;

% spline interpolate the head velocity and position
Data.Vel.Headzdotspline = spline(Data.Vel.frame(GUI.intv),
...
    Data.Vel.Headzdot(GUI.intv), GUI.start:GUI.finish);
Data.Pos.Headzspline = spline(Data.Pos.frame(GUI.intv), ...
    Data.Pos.Headz(GUI.intv), GUI.start:GUI.finish);

% call function AutoSpan from list to calculate optimal
span
[ Data.Pos.Span, Data.Vel.Span ] = fun.autospan( ...
    Data.Pos.frame(GUI.intv), Data.Pos.Headzspline ...
    , Data.Vel.frame(GUI.intv), Data.Vel.Headzdotspline);

% apply smoothing average filter to the data with the
corresponding
% values of span calculated in the lines above
Data.Pos.smooth = smooth(Data.Pos.Headzspline,
Data.Pos.Span);
Data.Vel.smooth = smooth(Data.Vel.Headzdot(GUI.intv),
Data.Vel.Span);

% subtract the smoothed data from the captured data, result
is sine like
% wave
Data.Pos.Smooth.undulation =
spline(Data.Pos.frame(GUI.intv), ...
    Data.Pos.Headz(GUI.intv) -
Data.Pos.smooth, GUI.start:GUI.finish);

% calculate amplitude of body undulations calculated in the
previous step

```

```

[ Data.Pos.Smooth.Max Data.Pos.Smooth.Min ] =
fun.amplitude( ...
    Data.Pos.frame(GUI.intv), Data.Pos.Smooth.undulation);

% calculate the fast fourier transform of the undulations
[ Data, GUI ] = fun.fft( GUI, Data );

% generate plots for the GUI
hold(handles.axes1, 'off')
plot(handles.axes1, Data.Pos.frame(GUI.intv),
Data.Pos.Smooth.undulation)
hold(handles.axes1, 'on')
plot(handles.axes1, Data.Pos.Smooth.Max(:,1), Data.Pos.Smooth
.Max(:,2), 'xr')
plot(handles.axes1, Data.Pos.Smooth.Min(:,1), Data.Pos.Smooth
.Min(:,2), 'ro')
line(get(handles.axes1, 'Xlim'), [mean(Data.Pos.Smooth.undula
tion) ...

mean(Data.Pos.Smooth.undulation)], 'Parent', handles.axes1)
set(handles.axes1, 'xlim', [GUI.start GUI.finish])
xlabel(handles.axes1, 'Frame')
ylabel(handles.axes1, 'Vertical displacement(mm)')
legend(['Span ' num2str(Data.Pos.Span)], ['Average ' ...
    num2str(mean(Data.Pos.Smooth.Max(:,2)) - ...
    mean(Data.Pos.Smooth.Min(:,2))])])
grid(handles.axes1, 'on')
hold(handles.axes1, 'off')

hold(handles.axes2, 'off')
plot(handles.axes2, Data.Pos.Smooth.f(GUI.LPF:GUI.NFFT/2+1),
2*abs(Data.Pos.Smooth.Y(GUI.LPF:GUI.NFFT/2+1)))
hold(handles.axes2, 'on')
plot(handles.axes2, Data.Pos.Smooth.maxFFTfreq, Data.Pos.Smoo
th.maxFFT, 'xr')
text(floor(Data.Pos.Smooth.maxFFTfreq)+2, Data.Pos.Smooth.ma
xFFT, ...

num2str(Data.Pos.Smooth.maxFFTfreq), 'Parent', handles.axes2)
xlabel(handles.axes2, 'Frequency (Hz)')
ylabel(handles.axes2, '|Y(f)|')
set(handles.axes2, 'ylim', [0 10])
set(handles.axes2, 'xlim', [0 50])
grid(handles.axes2, 'on')
hold(handles.axes2, 'off')

hold(handles.axes3, 'off')

```

```

plot(handles.axes3,
Data.Vel.frame(GUI.intv),Data.Vel.Headzdot(GUI.intv),...
    Data.Vel.frame(GUI.intv),Data.Vel.smooth)
xlabel(handles.axes3,'Frame')
ylabel(handles.axes3,'Vertical velocity (mm/s)')
set(handles.axes3,'xlim',[GUI.start GUI.finish])
grid(handles.axes3,'on')
legend(handles.axes3,'Data',['Span '
num2str(Data.Vel.Span)],'location','SouthEast')
hold(handles.axes3,'off')

hold(handles.axes4,'off')
plot(handles.axes4,Data.Pos.frame(GUI.intv),Data.Pos.Headz(
GUI.intv), ...
    Data.Pos.frame(GUI.intv),Data.Pos.smooth);
set(handles.axes4,'xlim',[GUI.start GUI.finish])
grid(handles.axes4,'on')
xlabel(handles.axes4,'Frame')
ylabel(handles.axes4,'Vertical displacement(mm)')
legend(handles.axes4,'Data',['Span '
num2str(Data.Pos.Span)],'location','SouthEast')
hold(handles.axes4,'off')

function [ Max, Min ] = PEAKtoPEAK(frame, data)
% set up a new variable
data1 = data;
% set the desired length of the vector data
data = ones(length(data)+20,1);
% put the data input into the vector of ones, leaving the
first 7 and last
% 6 values equal to 1
data(7:length(data1)+6) = data1;
j = 1;
k = 1;
for i = 7:length(frame)+6
    % approximating the sign of the derivative (time is
ALWAYS positive)
    der(i+1) = (data(i+1)-data(i));
    % check if derivative is positive or negative
    if isreal(sqrt(der(i+1))) ~= isreal(sqrt(der(i)))
        % determine if the derivative went from + to - or
vice versa
        switch isreal(sqrt(der(i)))
            case 1
                % if + to -, check if local maximum
                if data(i) == max(data(i-5:i+5))
                    % save frame and value of maxima

```

```

        Max(k,:) = [frame(i-6) data(i)];
        k = k+1;
    end
    case 0
        % if - to +, check if local minima
        if data(i) == min(data(i-5:i+5)) ||
min(data(i:i+5)) == 1
            % save frame and value of minima
            Min(j,:) = [frame(i-6) data(i)];
            j = j+1;
        end
    end
end
end
end

function [ Data, GUI] = dimensionless(GUI, Data, handles)
%http://www.engineeringtoolbox.com/dry-air-properties-
d\_973.html
% Calculating Reynolds Number
kinvisc = [1.343 1.568];
Temp = [275 300];
Data.dim.kinvisc = linterp(Temp,kinvisc,297.15)*10^-5; %
m^2 / s
Data.dim.Uref = sqrt(Data.Vel.Headxdot(GUI.intv).^2 + ...
    Data.Vel.Headydot(GUI.intv).^2 ...
    + Data.Vel.Headzdot(GUI.intv).^2)/1000; % m/s
Data.dim.Uref(isnan(Data.dim.Uref)==1)=[];
Data.dim.Lref =
str2double(get(handles.span,'string'))/1000; % m
Data.dim.Re = Data.dim.Uref .*Data.dim.Lref ./
Data.dim.kinvisc;

% Calculating Strouhal number
Data.dim.angle = pi/(2*180) *(mean(Data.Pos.Wing.Max(:,2))
- ...
    mean(Data.Pos.Wing.Min(:,2)));
Data.dim.freq = Data.Pos.maxFFTFreq;
Data.dim.ha = Data.dim.Lref*Data.dim.angle;
Data.dim.St = Data.dim.freq * Data.dim.ha ./ Data.dim.Uref;

% Calculating reduced frequency
Data.dim.k = pi*Data.dim.freq*Data.dim.Lref
./Data.dim.Uref;

% Calculating energy at each point of the flight
Data.dim.mass =
str2double(get(handles.mass,'string'))/1000;
Data.dim.KE = (1/2) * Data.dim.mass * Data.dim.Uref.^2;

```

```

Data.dim.PE = Data.dim.mass * Data.Pos.Headz (GUI.intv) *
9.81;
Data.dim.E = Data.dim.KE+Data.dim.PE;

% Plots
hold(handles.axes1, 'off')
plot(handles.axes1, Data.dim.E, Data.dim.Re)
xlabel(handles.axes1, 'Frame')
ylabel(handles.axes1, 'Re')
xtext = get(handles.axes1, 'xlim');
ytext = get(handles.axes1, 'ylim');
onefourth = 0.25*(ytext(2)-ytext(1));
text(xtext(1), ytext(1)+onefourth, ['Avg Re: '
num2str(mean(Data.dim.Re)) ] ...
    , 'Parent', handles.axes1)
% xlim(handles.axes1, [GUI.start GUI.finish])
grid(handles.axes1, 'on')

hold(handles.axes2, 'off')
plot(handles.axes2, Data.dim.E, Data.dim.St)
xtext = get(handles.axes2, 'xlim');
ytext = get(handles.axes2, 'ylim');
onefourth = 0.25*(ytext(2)-ytext(1));
text(xtext(1), ytext(1)+onefourth, ['Avg St: '
num2str(mean(Data.dim.St)) ] ...
    , 'Parent', handles.axes2)
xlabel(handles.axes2, 'Frame')
ylabel(handles.axes2, 'St')
% xlim(handles.axes2, [GUI.start GUI.finish])
grid(handles.axes2, 'on')

hold(handles.axes3, 'off')
plot(handles.axes3, Data.dim.E, Data.dim.k)
xtext = get(handles.axes3, 'xlim');
ytext = get(handles.axes3, 'ylim');
onefourth = 0.25*(ytext(2)-ytext(1));
text(xtext(1), ytext(1)+onefourth, ['Avg k: '
num2str(mean(Data.dim.k)) ] ...
    , 'Parent', handles.axes3)
xlabel(handles.axes3, 'Frame')
ylabel(handles.axes3, 'k')
% xlim(handles.axes3, [GUI.start GUI.finish])
grid(handles.axes3, 'on')

function [ maxFFTFreq, maxFFTindex , f, Y ] = FFT( signal )

Tincrements = 100;           % 100 Hz for data collected
Fs = Tincrements;           % Sampling frequency

```

```

L = length(signal);           % Length of signal
NFFT = 2^nextpow2(L);        % Next power of 2 from length of
angleindegree
Y = fft(signal,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1);

LPF = 3;                       % Low Pass Filter
Value
[~, maxFFTindex] = max(2*abs(Y(LPF:NFFT/2+1)));
maxFFTFreq = f(maxFFTindex+LPF-1);
Y = 2*abs(Y(1:NFFT/2+1));

function [SpanPos SpanVel] = AutoSpan(t,X,vt,V)

% initialize the use of other functions
fun = ButterflyGUIFuncs;

% ensure that length of vectors are odd
if isinteger(length(t)/2) == 0
    len = length(t)-1;
else
    len = length(t);
end
it = 1;

% test span from 3 to 31 for position
for i = 3:2:31
    XX = smooth(X,i);
    % determine amplitude between original signal and
    filtered
    [ PosMax, PosMin ] = fun.amplitude(t,X'-XX);
    % determine peak to peak amplitude and save to vector
    PosAVG(it,:) = [i mean(PosMax(:,2))-mean(PosMin(:,2))];
    it = it+1;
end

it = 1;
% test span from 3 to 31 for velocity
for i = 3:2:len
    VV = smooth(V,i);
    % determine amplitude of the signal calculated as the
    difference
    % between original signal and filtered
    [ VelMax, VelMin ] = fun.amplitude(vt,VV);
    % determine peak to peak amplitude and save to vector
    VelAVG(it,:) = [i mean(VelMax(:,2))-mean(VelMin(:,2))];
    it = it+1;
end

```

```

% determine minimum peak to peak amplitude for velocity
[~, VelINDEX] = min(VelAVG(:,2));

% determine maximum peak to peak amplitude for position
[~, PosINDEX] = max(PosAVG(:,2));

% most appropriate span maximizes the peak to peak
amplitude of oscillatory
% signal
SpanPos = PosAVG(PosINDEX,1);

% most appropriate span for velocity minimizes peaks in data
SpanVel = VelAVG(VelINDEX,1);

%Table GUI
function [ Data, GUI ] = thesis_perflap(GUI, Data, handles)

% initialize the functions for further use
fun = ButterflyGUIFUNS;

% specify new variables for convenience
f = Data.Pos.frame(GUI.intv);
t = f/100;
z = Data.Pos.Headz(GUI.intv);
zdot = Data.Vel.Headzdot(GUI.intv);
zsmooth = Data.Pos.smooth;
zdotsmooth = Data.Vel.smooth;
anglefilt = Data.Pos.anglefilt;
u = Data.Pos.Smooth.undulation;

% read mass from the text input box on GUI and convert to
kg
Data.perflap.mass =
str2double(get(handles.mass, 'string'))/1000;

% read half wing span from text input box on GUI and
convert to m
Data.perflap.span =
str2double(get(handles.thesis_span, 'string'))/1000;

% read popup menu to record gender of specimen
switch get(handles.gender, 'Value')
    case 1
        Data.perflap.gender = 'M';
    case 2
        Data.perflap.gender = 'F';
end

```



```

% calculate location of every local extrema for flapping
and undulation
[ Max_angle, Min_angle ] = fun.amplitude(f, anglefilt);
[ Max_u, Min_u ] = fun.amplitude(f, u);

% locate index which relates to each local maxima for
flapping angle
[ ~, index ] = ismember(Max_angle(:,1)/100,t);
Data.perflap.index = index;

% calculate parameters of interest for each flap, defined
as data between
% local maxima
for i = 2:length(index)-1
    Data.perflap.anglefilt{i} =
anglefilt(index(i):index(i+1));
    Data.perflap.frame{i} = f(index(i):index(i+1));
    Data.perflap.z{i} = z(index(i):index(i+1));
    Data.perflap.zsmooth{i} = zsmooth(index(i):index(i+1));
    Data.perflap.climb{i} = mean(Data.perflap.zsmooth{i});
    Data.perflap.u{i} = u(index(i):index(i+1));
    Data.perflap.t{i} = t(index(i):index(i+1));
    Data.perflap.zdot{i} = zdot(index(i):index(i+1));
    Data.perflap.zdotsmooth{i} =
zdotsmooth(index(i):index(i+1));
    Data.perflap.freq(i) = (Data.perflap.t{i}(end) -
Data.perflap.t{i}(1))^-1;
    Data.perflap.phase_min(i) = ((Min_angle(i,1) -
Min_u(i,1)) * ...
    3.60*Data.perflap.freq(i));
    Data.perflap.phase_max(i) = ((Max_angle(i,1) -
Max_u(i,1)) * ...
    3.60*Data.perflap.freq(i));
    Data.perflap.phase(i) = mean([Data.perflap.phase_min(i)
...
    Data.perflap.phase_max(i)]);
    Data.perflap.newTable(i,:) = {GUI.filename GUI.number i
...
    [num2str(Data.perflap.frame{i}(1)) '-' ...
num2str(Data.perflap.frame{i}(end))]...
    Data.perflap.mass Data.perflap.span ...
    Data.perflap.gender Data.perflap.freq(i)
Max_angle(i,2) ...
    Min_angle(i,2) ...
    Data.Pos.Span Data.perflap.freq(i) Max_u(i,2)
Min_u(i,2) ...
    Data.perflap.phase(i) Data.perflap.climb{i}};

```

```

end

% check if data calculated every flap exists
if exist('ButterflyAnalysisDataThesis_Perflap.mat') == 2

    % if data exists: load data
    D = load('ButterflyAnalysisDataThesis_Perflap.mat');
    Data.perflap.table = D.data;

    % check size of data
    [ Data.perflap.length Data.perflap.width ] = size( ...
        Data.perflap.table.data );
    Data.perflap.table.colheaders = {'Flight #' 'Butterfly
#' 'flap #' ...
    'Frame' 'Mass' 'Wing Span' 'Gender' 'Flap Freq'
'Max Amp' ...
    'Min Amp' 'Avg Span' 'Body Freq' 'Max Body' 'Min
Body' ...
    'Phase Diff' 'Climb Rate'};

    % add data calculated for current flight to previously
calculated data
    Data.perflap.table.data = [Data.perflap.table.data
        Data.perflap.newTable];

    % write full data to table along with column names
    set(handles.uitable1, 'Data', Data.perflap.table.data)

set(handles.uitable1, 'ColumnName', Data.perflap.table.colhea
ders)

    % save data
    data = Data.perflap.table;
    save 'ButterflyAnalysisDataThesis_Perflap.mat' data
else
    % if data does not exist

    % create column headers
    Data.perflap.table.colheaders = {'Flight #' 'Butterfly
#' 'flap #' ...
    'Frame' 'Mass' 'Wing Span' 'Gender' 'Flap Freq'
'Max Amp' ...
    'Min Amp' 'Avg Span' 'Body Freq' 'Max Body' 'Min
Body' ...
    'Phase Diff' 'Climb Rate'};

    % new data will be the only data set to table
    set(handles.uitable1, 'Data', Data.perflap.newTable)

```

```

set(handles.uitable1, 'ColumnName', Data.perflap.table.colheaders)

    % save data
    Data.perflap.table.data = Data.perflap.newTable;
    data = Data.perflap.table;
    save 'ButterflyAnalysisDataThesis_Perflap.mat' data

end
% save data to the corresponding butterfly
Data.perflap.Butterfly.(genvarname(['Butterfly'
num2str(GUI.number)])) ...
    = Data.perflap.table.data;
DATA = Data.perflap.Butterfly;
save ButterflyPerflap.mat DATA

function [ phase ] = PhaseDiff( GUI, Data )
% define variables for convenience
angle = Data.Pos.anglefilt;
z = Data.Pos.Headzspline';
zz = Data.Pos.smooth;

% define variables for calculating fft
t = Data.Pos.frame(GUI.intv);
Fs = 100;                % Sampling frequency
L = length(t);          % Length of signal

NFFT = 2^nextpow2(L);    % Next power of 2 from length of y
f(:,1) = Fs/2*linspace(0,1,NFFT/2+1);

% fft of flapping angle
Y(:,1) = fft(angle,NFFT)/L;

% find index of maximum frequency
[~, index(1)] = max(abs(Y(3:NFFT/2+1,1)));

% determine phase angle at the maximum flapping frequency
Phase(1) =
atan2(imag(Y(index(1)+2,1)),real(Y(index(1)+2,1)));

% fft of body undulations
Y(:,2) = fft(z-zz,NFFT)/L;

% find index of maximum frequency
[~, index(2)] = max(abs(Y(3:NFFT/2+1,2)));

% calculate phase angle at maximum undulation frequency

```

```

Phase(2) =
atan2(imag(Y(index(2)+2,2)),real(Y(index(2)+2,2)));

% calculate phase difference defined by undulations -
flapping
phase = abs(Phase(2)-Phase(1));

function [ Data, GUI ] = energy(GUI, Data, handles)
% initialize the functions for further use
fun = ButterflyGUIFuncs;

% Finding the magnitude of the velocity vector at the
beginning of the data
% segment
Data.energy.totalspeedinitial =
norm([Data.Vel.Headxdot(GUI.intv(1)),Data.Vel.Headydot(GUI.
intv(1)),Data.Vel.Headzdot(GUI.intv(1))])/1000;

% Finding the magnitude of velocity vector at the end of
the data segment
Data.energy.totalspeedfinal =
norm([Data.Vel.Headxdot(GUI.intv(length(GUI.intv))),Data.Ve
l.Headydot(GUI.intv(length(GUI.intv))),Data.Vel.Headzdot(GU
I.intv(length(GUI.intv)))])/1000;

% For indicating change in velocity
Data.energy.deltaV = Data.energy.totalspeedfinal-
Data.energy.totalspeedinitial;

% Find the dominant frequency of the fft to find the time
average flapping
% frequency.
Data.energy.maxfreq =
Data.Pos.f(find(2*abs(Data.Pos.Y(GUI.LPF:GUI.NFFT/2+1)) ==
max(2*abs(Data.Pos.Y(GUI.LPF:GUI.NFFT/2+1))),1)+4);

[ Data.energy.Max, Data.energy.Min ] =
fun.amplitude(Data.Pos.frame_spl, Data.Pos.anglefilt);
% assignin('base','Max',Max)
% assignin('base','Min',Min)

Data.energy.maxamp = (mean(Data.energy.Max(:,2))-
mean(Data.energy.Min(:,2)));
% Change in altitude, for potential energy calculations
Data.energy.deltaH = (Data.Pos.Headz(GUI.finishindex)-
Data.Pos.Headz(GUI.startindex))/1000;

```

```

% Mass of butterfly, the input is grams, this converts to
kg
Data.energy.mass =
str2double(get(handles.mass, 'string'))/1000;

% Kinetic Energy + Potential Energy. Use speed*abs(speed)
for cases of
% negative velocity.
Data.energy.TotalEnergy =
Data.energy.mass*(0.5*(Data.energy.totalspeedfinal* ...
    abs(Data.energy.totalspeedfinal)-
(Data.energy.totalspeedinitial* ...

abs(Data.energy.totalspeedfinal)))+9.81*Data.energy.deltaH)
;

Data.energy.Efficiency =
Data.energy.TotalEnergy/(Data.energy.maxfreq*((GUI.finish-
GUI.start)*GUI.dt));

Data.energy.span = get(handles.thesis_span, 'string');

switch get(handles.gender, 'Value')
    case 1
        Data.energy.gender = 'M';
    case 2
        Data.energy.gender = 'F';
end

switch get(handles.scales, 'Value')
    case 1
        D = load('ButterflyAnalysisDataScales.mat');
    case 2
        D = load('ButterflyAnalysisDataNoScales.mat');
end

Data.energy.table = D.data;
[Data.energy.table.length Data.energy.table.width] =
size(Data.energy.table.data);
Data.energy.table.colheaders = {'Flight #' 'Butterfly #'
'Frame' 'Mass' 'Span' 'Gender' 'Delta V' 'Delta H' 'Delta
E' ...
'Max Freq' 'Amplitude' 'Time' '# of Flaps'
'Efficiency'};

```

```

Data.energy.table.data(Data.energy.table.length+1,:) = {
GUI.filename ...
    GUI.number [num2str(GUI.start) '-' ...
    num2str(GUI.finish)] Data.energy.mass Data.energy.span
Data.energy.gender Data.energy.deltaV ...
    Data.energy.deltaH Data.energy.TotalEnergy
Data.energy.maxfreq Data.energy.maxamp ...
    (GUI.finish-GUI.start)*GUI.dt ...
    Data.energy.maxfreq*((GUI.finish-GUI.start)*GUI.dt)
Data.energy.Efficiency};

set(handles.uitable1, 'Data', Data.energy.table.data)
set(handles.uitable1, 'ColumnName', Data.energy.table.colhead
ers)
data = Data.energy.table;
switch get(handles.scales, 'Value')
    case 1
        save ButterflyAnalysisDataScales.mat data
    case 2
        save ButterflyAnalysisDataNoScales.mat data
end

function[ Data, DATA, GUI ] = thesis(GUI, Data, handles)
% initialize the use of other functions
fun = ButterflyGUIFuncs;

% read mass from the text input box on GUI and convert to
kg
Data.perflap.mass =
str2double(get(handles.mass, 'string'))/1000;

% read half wing span from text input box on GUI and
convert to m
Data.perflap.span =
str2double(get(handles.thesis_span, 'string'))/1000;

% calculate peak to peak amplitude of flapping angle
Data.thesis.flapamp = mean(Data.Pos.Wing.Max(:,2))- ...
    mean(Data.Pos.Wing.Min(:,2));

% calculate peak to peak amplitude of boddy undulations
Data.thesis.bodyamp = mean(Data.Pos.Smooth.Max(:,2))- ...
    mean(Data.Pos.Smooth.Min(:,2));

% average climb rate for segment determined using the mean
trend of the
% butterfly trajectory

```

```

Data.thesis.climbrate = mean(Data.Vel.smooth);

% calculate phase angle between flapping and body
undulations
[Data.thesis.phase] = fun.phase(GUI, Data);

% values for kinematic viscosity and temperature found at
link:
%      http://www.engineeringtoolbox.com/dry-air-
properties-d_973.html
kinvisc = [1.343 1.568];
Temp = [275 300];

% interpolate kinematic viscosity
Data.thesis.kinvisc = linterp(Temp,kinvisc,297.15)*10^-5; %
m^2 / s

% reference velocity for butterfly is defined as total
velocity of the head
% marker including body oscillations
Data.thesis.Uref = sqrt(Data.Vel.Headxdot(GUI.intv).^2 +
...
    Data.Vel.Headydot(GUI.intv).^2 ...
    + Data.Vel.Headzdot(GUI.intv).^2)/1000; % m/s

% remove instances where head marker disappears
Data.thesis.Uref(isnan(Data.thesis.Uref)==1)=[];

% reference length defined as the half span of the
butterfly wing
Data.thesis.Lref = Data.thesis.span; % m

% calculate Reynolds number
Data.thesis.Re = mean(Data.thesis.Uref *Data.thesis.Lref ./
...
    Data.thesis.kinvisc);

% convert flapping angle to radians
Data.thesis.angle = pi/(2*180)
*(mean(Data.Pos.Wing.Max(:,2)) - ...
    mean(Data.Pos.Wing.Min(:,2)));

% flapping frequency
Data.thesis.freq = Data.Pos.maxFFTFreq;

% length of the path of the wingtip
Data.thesis.ha = Data.thesis.Lref.*Data.thesis.angle;

```

```

% calculate Strouhl number
Data.thesis.St = mean(Data.thesis.freq .* Data.thesis.ha ./
Data.thesis.Uref);

% Calculating reduced frequency
Data.thesis.k = mean(pi*Data.thesis.freq.*Data.thesis.Lref
./Data.thesis.Uref);

% Calculating energy at each point of the flight
Data.thesis.KE = (1/2) * Data.thesis.mass *
Data.thesis.Uref.^2;
Data.thesis.PE = Data.thesis.mass *
Data.Pos.Headz(GUI.intv) * 9.81;

% read popup menu to record gender of specimen
switch get(handles.gender, 'Value')
    case 1
        Data.thesis.gender = 'M';
    case 2
        Data.thesis.gender = 'F';
end

% Perflap
fun = ButterflyGUIFuncs;
f = Data.Pos.frame(GUI.intv);
t = f/100;
z = Data.Pos.Headz(GUI.intv);
zdot = Data.Vel.Headzdot(GUI.intv);
zsmooth = Data.Pos.smooth;
zdotsmooth = Data.Vel.smooth;
anglefilt = Data.Pos.anglefilt;
u = Data.Pos.Smooth.undulation;

Data.perflap.mass =
str2double(get(handles.mass, 'string'))/1000;

Data.perflap.span =
str2double(get(handles.thesis_span, 'string'))/1000;

switch get(handles.gender, 'Value')
    case 1
        Data.perflap.gender = 'M';
    case 2
        Data.perflap.gender = 'F';
end

[ Max_angle, Min_angle ] = fun.amplitude(f, anglefilt);

```



```

[ Max_u, Min_u ] = fun.amplitude(f, u);
[ ~, index ] = ismember(Max_angle(:,1)/100,t);
Data.perflap.index = index;

for i = 2:length(index)-1
    Data.perflap.anglefilt{i} =
anglefilt(index(i):index(i+1));
    Data.perflap.frame{i} = f(index(i):index(i+1));
    Data.perflap.z{i} = z(index(i):index(i+1));
    Data.perflap.zsmooth{i} = zsmooth(index(i):index(i+1));
    Data.perflap.climb{i} =
mean(Data.Vel.Smooth(index(i):index(i+1)));
    Data.perflap.u{i} = u(index(i):index(i+1));
    Data.perflap.t{i} = t(index(i):index(i+1));
    Data.perflap.zdot{i} = zdot(index(i):index(i+1));
    Data.perflap.zdotsmooth{i} =
zdotsmooth(index(i):index(i+1));
    Data.perflap.freq(i) = (Data.perflap.t{i}(end)-
Data.perflap.t{i}(1))^-1;
    Data.perflap.phase_min(i) = ((Min_angle(i,1) -
Min_u(i,1))*3.60*Data.perflap.freq(i));
    Data.perflap.phase_max(i) = ((Max_angle(i,1) -
Max_u(i,1))*3.60*Data.perflap.freq(i));
    Data.perflap.phase(i) = mean([Data.perflap.phase_min(i)
Data.perflap.phase_max(i)]);
    Data.perflap.newTable(i-1,:) = {GUI.filename GUI.number
i ...
    [num2str(GUI.start) '-' num2str(GUI.finish)] ...
    [num2str(Data.perflap.frame{i}(1)) '-'
num2str(Data.perflap.frame{i}(end))]} ...
    Data.perflap.mass Data.perflap.span ...
    Data.perflap.gender Data.perflap.freq(i)
Max_angle(i,2) Min_angle(i,2) ...
    Data.Pos.Span Data.perflap.freq(i) Max_u(i,2)
Min_u(i,2) ...
    Data.perflap.phase(i) Data.perflap.climb{i}};
end

% Saving Data
if exist('ButterflyAnalysisDataThesis.mat','file') == 2
    D = load('ButterflyAnalysisDataThesis.mat');

    Data.thesis.table = D.data;
    [Data.thesis.table.length Data.thesis.table.width] =
size(Data.thesis.table.data);

```

```

        Data.thesis.table.colheaders = {'Flight #' 'Butterfly
#' 'Frame' 'Mass' 'Wing Span' 'Gender' ...
        'Flap Freq' 'Flap Amp' 'Avg Span' 'Body Freq' 'Body
Amp' 'Phase Diff' 'Climb Rate' 'Re' 'St' 'k'};

        Data.thesis.table.data(Data.thesis.table.length+1,:) =
{ GUI.filename ...
        GUI.number [num2str(GUI.start) '-'
num2str(GUI.finish)] ...
        Data.thesis.mass Data.thesis.span
Data.thesis.gender ...
        Data.Pos.maxFFTfreq Data.thesis.flapamp
Data.Pos.Span Data.Pos.Smooth.maxFFTfreq ...
        Data.thesis.bodyamp Data.thesis.phase
Data.thesis.climbrate Data.thesis.Re Data.thesis.St ...
        Data.thesis.k};

        set(handles.uitable1, 'Data', Data.thesis.table.data)

set(handles.uitable1, 'ColumnName', Data.thesis.table.colhead
ers)
        data = Data.thesis.table;
        save ButterflyAnalysisDataThesis.mat data

else

        Data.thesis.table.colheaders = {'Flight #' 'Butterfly
#' 'Frame' 'Mass' 'Wing Span' 'Gender' ...
        'Flap Freq' 'Flap Amp' 'Avg Span' 'Body Freq' 'Body
Amp' 'Phase Diff' 'Climb Rate' 'Re' 'St' 'k'};
        Data.thesis.table.data(1,:) = { GUI.filename ...
        GUI.number [num2str(GUI.start) '-'
num2str(GUI.finish)] ...
        Data.thesis.mass Data.thesis.span
Data.thesis.gender ...
        Data.Pos.maxFFTfreq Data.thesis.flapamp
Data.Pos.Span Data.Pos.Smooth.maxFFTfreq ...
        Data.thesis.bodyamp Data.thesis.phase
Data.thesis.climbrate Data.thesis.Re Data.thesis.St ...
        Data.thesis.k};

        set(handles.uitable1, 'Data', Data.thesis.table.data)

set(handles.uitable1, 'ColumnName', Data.thesis.table.colhead
ers)
        data = Data.thesis.table;
        save ButterflyAnalysisDataThesis.mat data

```

```

end

if exist('ButterflyPerflap.mat','file') == 2
    load('ButterflyPerflap.mat');
    if isfield(DATA,['Butterfly' num2str(GUI.number)]) == 1
        DATA.(genvarname(['Butterfly'
num2str(GUI.number)])) = ...
            [DATA.(genvarname(['Butterfly'
num2str(GUI.number)]));Data.perflap.newTable];
    else
        DATA.(genvarname(['Butterfly'
num2str(GUI.number)])) =...
            Data.perflap.newTable;
    end
else
    DATA.(genvarname(['Butterfly' num2str(GUI.number)]))
=...
    Data.perflap.newTable;
end

save ButterflyPerflap.mat DATA

% Batch
function [ Data, GUI ] = importBatch(batch)

% read data from the list of files generated using
GUI.data = xlsread(batch.FILE);

% use while loop to separate data into usable variables
% while loop is conditioned so that it breaks when there is
not a number
%   in the frame column. This only occurs at the end of a
data collection
%   segment
i = 5;
j = 1;
while isnan(GUI.data(i,1)) == 0
    Data.Pos.frame(j,1) = GUI.data(i,1);

    Data.Pos.Headx(j,1) = GUI.data(i,3);
    Data.Pos.Heady(j,1) = GUI.data(i,4);
    Data.Pos.Headz(j,1) = GUI.data(i,5);

    Data.Pos.LeftWingx(j,1) = GUI.data(i,6);
    Data.Pos.LeftWingy(j,1) = GUI.data(i,7);
    Data.Pos.LeftWingz(j,1) = GUI.data(i,8);
end

```

```

    Data.Pos.RightWingx(j,1) = GUI.data(i,9);
    Data.Pos.RightWingy(j,1) = GUI.data(i,10);
    Data.Pos.RightWingz(j,1) = GUI.data(i,11);

    Data.Pos.LowerWingx(j,1) = GUI.data(i,12);
    Data.Pos.LowerWingy(j,1) = GUI.data(i,13);
    Data.Pos.LowerWingz(j,1) = GUI.data(i,14);

    i = i+1;
    j = j+1;
end

clear i j

% remove velocity headers from import and only extract head
information
i = length(Data.Pos.frame)+12;
j = 1;

for k = i:length(GUI.data)
    Data.Vel.frame(j,1) = GUI.data(k,1);

    Data.Vel.Headxdot(j,1) = GUI.data(k,3);
    Data.Vel.Headydot(j,1) = GUI.data(k,4);
    Data.Vel.Headzdot(j,1) = GUI.data(k,5);

    j = j+1;
end

clear i j k

%forming 3-D vectors for position and velocity
% distance from left wing to head
Data.Pos.v1x = Data.Pos.LeftWingx - Data.Pos.Headx;
Data.Pos.v1y = Data.Pos.LeftWingy - Data.Pos.Heady;
Data.Pos.v1z = Data.Pos.LeftWingz - Data.Pos.Headz;
Data.Pos.v1 = [Data.Pos.v1x Data.Pos.v1y Data.Pos.v1z]';

% distance from right wing to head
Data.Pos.v2x = Data.Pos.RightWingx - Data.Pos.Headx;
Data.Pos.v2y = Data.Pos.RightWingy - Data.Pos.Heady;
Data.Pos.v2z = Data.Pos.RightWingz - Data.Pos.Headz;
Data.Pos.v2 = [Data.Pos.v2x Data.Pos.v2y Data.Pos.v2z]';

% distance from lower left wing to head
Data.Pos.v3x = Data.Pos.LowerWingx - Data.Pos.Headx;
Data.Pos.v3y = Data.Pos.LowerWingy - Data.Pos.Heady;
Data.Pos.v3z = Data.Pos.LowerWingz - Data.Pos.Headz;

```

```

Data.Pos.v3 = [Data.Pos.v3x Data.Pos.v3y Data.Pos.v3z]';

GUI.dt = 1/GUI.data(1,1);

for i = 1:length(Data.Pos.frame)

    % flapping angle
    % calculate angle between two vectors
    Data.Pos.angle(i) =
atan2(norm(cross(Data.Pos.v1(:,i),Data.Pos.v2(:,i))),dot(Data
ta.Pos.v1(:,i),Data.Pos.v2(:,i))));

    % establish sign
    Data.Pos.crossproduct(i) =
dot(cross(Data.Pos.v1(:,i),Data.Pos.v2(:,i)),Data.Pos.v3(:,
i));
    if Data.Pos.crossproduct(i) > 0
        Data.Pos.angleindegree(i) = 360-
Data.Pos.angle(i)*180/pi;
    elseif Data.Pos.crossproduct(i) == 0
        Data.Pos.angleindegree(i) =
Data.Pos.angle(i)*180/pi;
    elseif Data.Pos.crossproduct(i) < 0
        Data.Pos.angleindegree(i) =
Data.Pos.angle(i)*180/pi;
    elseif isnan(Data.Pos.crossproduct(i))==1
        Data.Pos.angleindegree(i) = 0;
    end
    % establish time vector
    if i == 1
        Data.Pos.t(i) = 0;
    else
        Data.Pos.t(i) = Data.Pos.t(i-1)+ GUI.dt;
    end
end

for j = 1:length(Data.Vel.frame)

    % calculate horizontal velocity
    Data.Vel.horizontalvel(:,j) =
[Data.Vel.Headxdot(j);Data.Vel.Headydot(j)];
    Data.Vel.horizontalvel(:,j) =
norm(Data.Vel.horizontalvel(:,j),2);

    % calculate flight direction
    Data.Vel.direction(j) =
atan2(Data.Vel.Headxdot(j),Data.Vel.Headydot(j));

```

```

    % establish time vector
    if j == 1
        Data.Vel.t(j) = 0;
    else
        Data.Vel.t(j) = Data.Vel.t(j-1)+ GUI.dt;
    end
end

function [ Data, GUI ] = ManualSmoothBatch(GUI, Data,
batch)

% initialize the functions for further use
fun = ButterflyGUIFuncs;

for i = 1:length(batch.frames)
    a(i) = str2double(batch.frames(i));
end
index = find(isnan(a)==1);
GUI.start = str2double(batch.frames(1:index-1));
GUI.finish = str2double(batch.frames(index+1:end));
GUI.startindex = find(Data.Pos.frame == GUI.start,1);
GUI.finishindex = find(Data.Pos.frame == GUI.finish,1);
GUI.intv = GUI.startindex:GUI.finishindex;
Data.Pos.Span = batch.span;

f = Data.Vel.frame(GUI.intv);
fP = Data.Pos.frame(GUI.intv);
zdotspl = Data.Vel.Headzdot(GUI.intv);
zspl = Data.Pos.Headz(GUI.intv);

f(isnan(zdotspl)==1)=[];
zdotspl(isnan(zdotspl)==1)=[];

fP(isnan(zspl) == 1)=[];
zspl(isnan(zspl)==1)=[];

Data.Vel.Headzdotspline =
spline(f,zdotspl,GUI.start:GUI.finish);
Data.Pos.Headzspline =
spline(fP,zspl,GUI.start:GUI.finish);

Data.Pos.smooth = smooth(Data.Pos.Headzspline,
Data.Pos.Span);
% Data.Vel.smooth = smooth(Data.Vel.Headzdot(GUI.intv),
Data.Vel.Span);
Data.Vel.smooth = diff(Data.Pos.smooth)/0.01;
Data.Vel.smooth = [Data.Vel.smooth; Data.Vel.smooth(end)];

```

```

Data.Pos.Smooth.undulation = Data.Pos.Headzspline'-
Data.Pos.smooth;
[ Data.Pos.Smooth.Max Data.Pos.Smooth.Min ] =
fun.amplitude(Data.Pos.frame(GUI.intv), ...
    Data.Pos.Smooth.undulation);

%FFT
% Taking fft
GUI.sysvector = Data.Pos.Smooth.undulation;
GUI.Tincrements = 100;
% flapping frequency
GUI.Fs = GUI.Tincrements; % Sampling
frequency
GUI.Ts = 1/GUI.Fs; % Sample time
GUI.L = length(Data.Pos.Smooth.undulation); % Length of
signal
GUI.NFFT = 2^nextpow2(GUI.L); % Next power of 2 from length
of angleindegree
Data.Pos.Smooth.Y = fft(GUI.sysvector,GUI.NFFT)/GUI.L;
Data.Pos.Smooth.f = GUI.Fs/2*linspace(0,1,GUI.NFFT/2+1);

GUI.LPF = 3; % Low Pass
Filter Value
[Data.Pos.Smooth.maxFFT, Data.Pos.Smooth.maxFFTindex] =
max(2*abs(Data.Pos.Smooth.Y(GUI.LPF:GUI.NFFT/2+1)));
Data.Pos.Smooth.maxFFTFreq =
Data.Pos.Smooth.f(Data.Pos.Smooth.maxFFTindex+GUI.LPF-1);

GUI.startindex = find(Data.Pos.frame == GUI.start,1);
GUI.finishindex = find(Data.Pos.frame == GUI.finish,1);
GUI.intv = GUI.startindex:GUI.finishindex;
Data.Pos.Span = batch.span;

f = Data.Vel.frame(GUI.intv);
fP = Data.Pos.frame(GUI.intv);
zdotspl = Data.Vel.Headzdot(GUI.intv);
zspl = Data.Pos.Headz(GUI.intv);

f(isnan(zdotspl)==1)=[];
zdotspl(isnan(zdotspl)==1)=[];

fP(isnan(zspl) == 1)=[];
zspl(isnan(zspl)==1)=[];

```

```

Data.Vel.Headzdotspline =
spline(f,zdotspl,GUI.start:GUI.finish);
Data.Pos.Headzspline =
spline(fP,zspl,GUI.start:GUI.finish);

Data.Pos.smooth = smooth(Data.Pos.Headzspline,
Data.Pos.Span);
% Data.Vel.smooth = smooth(Data.Vel.Headzdot(GUI.intv),
Data.Vel.Span);
Data.Vel.smooth = diff(Data.Pos.smooth)/0.01;
Data.Vel.smooth = [Data.Vel.smooth; Data.Vel.smooth(end)];

Data.Pos.Smooth.undulation = Data.Pos.Headzspline'-
Data.Pos.smooth;
[ Data.Pos.Smooth.Max Data.Pos.Smooth.Min ] =
fun.amplitude(Data.Pos.frame(GUI.intv), ...
Data.Pos.Smooth.undulation);

%FFT
% Taking fft
GUI.sysvector = Data.Pos.Smooth.undulation;
GUI.Tincrements = 100;
% flapping frequency
GUI.Fs = GUI.Tincrements; % Sampling
frequency
GUI.Ts = 1/GUI.Fs; % Sample time
GUI.L = length(Data.Pos.Smooth.undulation); % Length of
signal
GUI.NFFT = 2^nextpow2(GUI.L); % Next power of 2 from length
of angleindegree
Data.Pos.Smooth.Y = fft(GUI.sysvector,GUI.NFFT)/GUI.L;
Data.Pos.Smooth.f = GUI.Fs/2*linspace(0,1,GUI.NFFT/2+1);

GUI.LPF = 3; % Low Pass
Filter Value
[Data.Pos.Smooth.maxFFT, Data.Pos.Smooth.maxFFTindex] =
max(2*abs(Data.Pos.Smooth.Y(GUI.LPF:GUI.NFFT/2+1)));
Data.Pos.Smooth.maxFFTfreq =
Data.Pos.Smooth.f(Data.Pos.Smooth.maxFFTindex+GUI.LPF-1);

function [ Data, GUI ] = filterBatch(GUI, Data, batch)

% initialize the functions for further use
fun = ButterflyGUIFuns;

for i = 1:length(batch.frames)
a(i) = str2double(batch.frames(i));
end

```



```

index = find(isnan(a)==1);
GUI.start = str2double(batch.frames(1:index-1));
GUI.finish = str2double(batch.frames(index+1:end));
GUI.startindex = find(Data.Pos.frame == GUI.start,1);
GUI.finishindex = find(Data.Pos.frame == GUI.finish,1);
GUI.intv = GUI.startindex:GUI.finishindex;

% define new variables for wing angle (x) and frame (t)
x = Data.Pos.angleindegree(GUI.intv);
t = Data.Pos.frame(GUI.intv);

% remove all frames where wing angle is equal to 0
t(x == 0) = [];
x(x ==0 ) = [];

% create a time increment for the spline interpolation
tinc = GUI.data(1,1)/100;
Data.Pos.frame_spl = GUI.start:tinc:GUI.finish;

% calculate interpolation
Data.Pos.anglefilt = spline(t,x,Data.Pos.frame_spl);

% amplitude of wing angle using PEAKtoPEAK
[ Data.Pos.Wing.Max, Data.Pos.Wing.Min ] =
fun.amplitude(Data.Pos.frame_spl, Data.Pos.anglefilt);

% remove 180 deg jumps in data
Data.Vel.direction = unwrap(Data.Vel.direction);
Data.Vel.directionindegree = Data.Vel.direction*180/pi;

% calculate fast fourier transform using function FFT
[ Data, GUI ] = fun.fft( GUI ,Data );

% average velocity
%   define new variables
Data.Vel.Headzdotavg = Data.Vel.Headzdot;
Data.Vel.horizontalsspeedavg = Data.Vel.horizontalsspeed;

% remove data where head marker is not present
Data.Vel.horizontalsspeedavg(isnan(Data.Vel.horizontalsspeedavg)==1) = [];
Data.Vel.Headzdotavg(isnan(Data.Vel.Headzdotavg)==1) = [];
GUI.intvavg = GUI.startindex:GUI.finishindex;

% calculate averages
Data.Vel.AVGhorizontal =
mean(Data.Vel.horizontalsspeedavg(GUI.intvavg));

```

```

Data.Vel.AVGvertical =
mean(Data.Vel.Headzdotavg(GUI.intvavg));

function [ Data, GUI ] = thesisperflapbatch(GUI, Data,
batch)

% initialize the functions for further use
fun = ButterflyGUIFuncs;

% Mass of butterfly, the input is grams, this converts to
kg
Data.thesis.mass = batch.mass;

Data.thesis.span = batch.WingSpan;

Data.thesis.maxamp = mean(Data.Pos.Smooth.Max(:,2))- ...
    mean(Data.Pos.Smooth.Min(:,2));

Data.thesis.flapamp = mean(Data.Pos.Wing.Max(:,2))- ...
    mean(Data.Pos.Wing.Min(:,2));

Data.thesis.bodyamp = mean(Data.Pos.Smooth.Max(:,2))- ...
    mean(Data.Pos.Smooth.Min(:,2));

Data.thesis.climbrate = mean(Data.Vel.smooth);

[Data.thesis.phase] = fun.phase(GUI, Data)*180/pi;
Data.thesis.phase(Data.thesis.phase>180) = 360- ...
    Data.thesis.phase(Data.thesis.phase>pi);

kinvisc = [1.343 1.568];
%http://www.engineeringtoolbox.com/dry-air-properties-
d\_973.html
Temp = [275 300];
Data.thesis.kinvisc = linterp(Temp,kinvisc,297.15)*10^-5; %
m^2 / s
Data.thesis.Uref = sqrt(Data.Vel.Headxdot(GUI.intv).^2 +
Data.Vel.Headydot(GUI.intv).^2 ...
    + Data.Vel.Headzdot(GUI.intv).^2)/1000; % m/s
Data.thesis.Uref(isnan(Data.thesis.Uref)==1)=[];
Data.thesis.Lref = Data.thesis.span; % m
Data.thesis.Re = mean(Data.thesis.Uref *Data.thesis.Lref ./
Data.thesis.kinvisc);

% Calculating Strouhal number
Data.thesis.angle = pi/(2*180)
*(mean(Data.Pos.Wing.Max(:,2)) - ...

```

```

    mean(Data.Pos.Wing.Min(:,2)));
Data.thesis.freq = Data.Pos.maxFFTFreq;
Data.thesis.ha = Data.thesis.Lref.*Data.thesis.angle;
Data.thesis.St = mean(Data.thesis.freq .* Data.thesis.ha ./
Data.thesis.Uref);

% Calculating reduced frequency
Data.thesis.k = mean(pi*Data.thesis.freq.*Data.thesis.Lref
./Data.thesis.Uref);

% Calculating energy at each point of the flight
Data.thesis.KE = (1/2) * Data.thesis.mass *
Data.thesis.Uref.^2;
Data.thesis.PE = Data.thesis.mass *
Data.Pos.Headz(GUI.intv) * 9.81;
% Data.thesis.E = Data.thesis.KE+Data.thesis.PE;

Data.thesis.gender = batch.gender;

% Perflap
fun = ButterflyGUIFuncs;
f = Data.Pos.frame(GUI.intv);
t = f/100;
z = Data.Pos.Headz(GUI.intv);
zdot = Data.Vel.Headzdot(GUI.intv);
zsmooth = Data.Pos.smooth;
zdotsmooth = Data.Vel.smooth;
anglefilt = Data.Pos.anglefilt;
u = Data.Pos.Smooth.undulation;

Data.perflap.mass = batch.mass;

Data.perflap.span = batch.WingSpan;

Data.perflap.gender = batch.gender;

[ Max_angle, Min_angle ] = fun.amplitude(f, anglefilt);
[ Max_u, Min_u ] = fun.amplitude(f, u);
[ ~,index ] = ismember(Max_angle(:,1)/100,t);
Data.perflap.index = index;

for i = 2:length(index)-1
    Data.perflap.anglefilt{i} =
anglefilt(index(i):index(i+1));

```

```

    Data.perflap.frame{i} = f(index(i):index(i+1));
    Data.perflap.z{i} = z(index(i):index(i+1));
    Data.perflap.zsmooth{i} = zsmooth(index(i):index(i+1));
    Data.perflap.climb{i} =
mean(Data.Vel.smooth(index(i):index(i+1)));
    Data.perflap.u{i} = u(index(i):index(i+1));
    Data.perflap.t{i} = t(index(i):index(i+1));
    Data.perflap.zdot{i} = zdot(index(i):index(i+1));
    Data.perflap.zdotsmooth{i} =
zdotsmooth(index(i):index(i+1));
    Data.perflap.freq(i) = (Data.perflap.t{i}(end) -
Data.perflap.t{i}(1))^-1;
    Data.perflap.phase_min(i) = ((Min_angle(i,1) -
Min_u(i,1))*3.60*Data.perflap.freq(i));
    Data.perflap.phase_max(i) = ((Max_angle(i,1) -
Max_u(i,1))*3.60*Data.perflap.freq(i));
    Data.perflap.phase(i) =
abs(mean([Data.perflap.phase_min(i)
Data.perflap.phase_max(i)]));
    if Data.perflap.phase(i) > 180
        Data.perflap.phase(i) = 360-Data.perflap.phase(i);
    end
    Data.perflap.newTable(i-1,:) = {batch.filename
batch.number i ...
[num2str(GUI.start) '-' num2str(GUI.finish)] ...
[num2str(Data.perflap.frame{i}(1)) '-'
num2str(Data.perflap.frame{i}(end))]} ...
    Data.perflap.mass Data.perflap.span ...
    Data.perflap.gender Data.perflap.freq(i)
Max_angle(i,2) Min_angle(i,2) ...
    Data.Pos.Span Data.perflap.freq(i) Max_u(i,2)
Min_u(i,2) ...
    abs(Data.perflap.phase(i)) Data.perflap.climb{i}};
end

% Saving Data
if exist('ButterflyAnalysisDataThesisBatch.mat','file') ==
2
    D = load('ButterflyAnalysisDataThesisBatch.mat');

    Data.thesis.table = D.data;
    [Data.thesis.table.length Data.thesis.table.width] =
size(Data.thesis.table.data);
    Data.thesis.table.colheaders = {'Flight #' 'Butterfly
#' 'Frame' 'Mass' 'Wing Span' 'Gender' ...

```

```

        'Flap Freq' 'Flap Amp' 'Avg Span' 'Body Freq' 'Body
Amp' 'Phase Diff' 'Climb Rate' 'Re' 'St' 'k'};

```

```

        Data.thesis.table.data(Data.thesis.table.length+1,:) =
{ batch.filename ...
    batch.number [num2str(GUI.start) '-'
num2str(GUI.finish)] ...
    Data.thesis.mass Data.thesis.span
Data.thesis.gender ...
    Data.Pos.maxFFTFreq Data.thesis.flapamp
Data.Pos.Span Data.Pos.Smooth.maxFFTFreq ...
    Data.thesis.bodyamp Data.thesis.phase
Data.thesis.climbrate Data.thesis.Re Data.thesis.St ...
    Data.thesis.k};

```

```

        % set(handles.uitable1,'Data',Data.thesis.table.data)
        %
set(handles.uitable1,'ColumnName',Data.thesis.table.colhead
ers)
        data = Data.thesis.table;
        save ButterflyAnalysisDataThesisBatch.mat data

```

```

else

```

```

        Data.thesis.table.colheaders = {'Flight #' 'Butterfly
#' 'Frame' 'Mass' 'Wing Span' 'Gender' ...
    'Flap Freq' 'Flap Amp' 'Avg Span' 'Body Freq' 'Body
Amp' 'Phase Diff' 'Climb Rate' 'Re' 'St' 'k'};
        Data.thesis.table.data(1,:) = { batch.filename ...
    batch.number [num2str(GUI.start) '-'
num2str(GUI.finish)] ...
    Data.thesis.mass Data.thesis.span
Data.thesis.gender ...
    Data.Pos.maxFFTFreq Data.thesis.flapamp
Data.Pos.Span Data.Pos.Smooth.maxFFTFreq ...
    Data.thesis.bodyamp Data.thesis.phase
Data.thesis.climbrate Data.thesis.Re Data.thesis.St ...
    Data.thesis.k};

```

```

        %
set(handles.uitable1,'Data',Data.thesis.table.data)
        %
set(handles.uitable1,'ColumnName',Data.thesis.table.colhead
ers)
        data = Data.thesis.table;
        save ButterflyAnalysisDataThesisBatch.mat data

```

```

end

if exist('ButterflyPerflapBatch.mat','file') == 2
    load('ButterflyPerflapBatch.mat');
    if isfield(DATA,['Butterfly' num2str(batch.number)]) ==
1
        DATA.(genvarname(['Butterfly'
num2str(batch.number)])) = ...
            [DATA.(genvarname(['Butterfly'
num2str(batch.number)]));Data.perflap.newTable];
    else
        DATA.(genvarname(['Butterfly'
num2str(batch.number)])) =...
            Data.perflap.newTable;
    end
else
    DATA.(genvarname(['Butterfly' num2str(batch.number)]))
=...
    Data.perflap.newTable;
end

save ButterflyPerflapBatch.mat DATA

function [ Data, GUI ] = Batch(batch)

% initialize the functions for further use
fun = ButterflyGUIFuncs;

[ Data, GUI ] = fun.importbatch(batch);

[ Data, GUI ] = fun.filterbatch(GUI, Data, batch);

[ Data, GUI ] = fun.mansmoothbatch(GUI, Data, batch);

[ Data, GUI ] = fun.perflapbatch(GUI, Data, batch);

```

References

1. Petricca L., Ohlckers P., Grinde C., Micro- and nano-air vehicles: State of the art. *International Journal of Aerospace Engineering*. 2011;2011.
2. Pines D., Bohorquez F., Challenges Facing Future Micro-Air-Vehicle Development. *Journal of Aircraft*. 2006;43(2):290–305.
3. Shyy W., Aono H., Chimakurthi S.K., Trizila P., Kang C.K., Cesnik C.E.S., et al., Recent progress in flapping wing aerodynamics and aeroelasticity. *Progress in Aerospace Sciences* [Internet]. Elsevier; 2010;46(7):284–327. Available from: <http://dx.doi.org/10.1016/j.paerosci.2010.01.001>.
4. Wu H., Sun D., Zhou Z., Micro air vehicle: Configuration, analysis, fabrication, and test. *IEEE/ASME Transactions on Mechatronics*. 2004;9(1):108–17.
5. Ifju P.G., Jenkins D.A., Waszak M.R., Ettinger S., Lian Y., Shyy W., Flexible-Wing-Based Micro Air Vehicles. *AIAA*. 2002;1–13.
6. Keenon M., Klingebiel K., Won H., Andriukov A., Development of the Nano Hummingbird: A Tailless Flapping Wing Micro Air Vehicle. *AIAA Aerospace Science Meeting*. 2012. p. 1–24.
7. De Croon G.C.H.E., Groen M. a, De Wagter C., Remes A., Ruijsink R., van Oudheusden B.W., Design, aerodynamics and autonomy of the DelFly. *Bioinspiration & Biomimetics*. 2012;7(2):025003.
8. Teoh Z.E., Fuller S.B., Chirarattananon P., Prez-Arancibia N.O., Greenberg J.D., Wood R.J., A hovering flapping-wing microrobot with altitude control and passive upright stability. *IEEE International Conference on Intelligent Robots and Systems*. 2012;3209–16.
9. <http://www.hindawi.com/journals/ijae/2011/214549/fig26/>.
10. <http://www.avinc.com/nano>.
11. <http://www.delfly.nl/history.html>.
12. <http://spectrum.ieee.org/automaton/robotics/robotics-hardware/harvard-robobees-learn-to-steer-mostly>.
13. <http://www.proxdynamics.com/products/pd-100-black-hornet-prs>.
14. Shyy W., Aono H., Kang C.-K., Liu H., *An Introduction to Flapping Wing Aerodynamics*. New York; 2013.
15. Sun M., Insect flight dynamics: Stability and control. *Reviews of Modern Physics* [Internet]. 2014 May 16 [cited 2014 Nov 20];86(2):615–46. Available from: <http://link.aps.org/doi/10.1103/RevModPhys.86.615>.

16. Wang Z.J., Dissecting Insect Flight. *Annual Review of Fluid Mechanics*. 2005;37:183–210.
17. Sane S.P., The aerodynamics of insect flight. *The Journal of experimental biology*. 2003;206:4191–208.
18. Kovac M., Vogt D., Ithier D., Smith M., Wood R., Aerodynamic evaluation of four butterfly species for the design of flapping-gliding robotic insects. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems [Internet]. Ieee; 2012 Oct;1102–9. Available from: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6385453>.
19. Lin T., Zheng L., Hedrick T., Mittal R., The Significance of Moment-of-Inertia Variation in Flight Manoeuvres of Butterflies. *Bioinspiration & biomimetics* [Internet]. 2012 Dec;7(4):044002. Available from: <http://www.ncbi.nlm.nih.gov/pubmed/23092976>.
20. Jantzen B., Eisner T., Hindwings are unnecessary for flight but essential for execution of normal evasive flight in Lepidoptera. *Proceedings of the National Academy of Sciences of the United States of America*. 2008;105(43):16636–40.
21. Rayner J.M. V., Viscardi P.W., Ward S., Speakman J.R., Aerodynamics and Energetics of Intermittent Flight in Birds¹. *American Zoologist*. 2001;41(2):188–204.
22. Dudley R., *The Biomechanics of Insect Flight: Form, Function, Evolution*. 2000.
23. Brower L., Monarch butterfly orientation: missing pieces of a magnificent puzzle. *The Journal of experimental biology* [Internet]. 1996;199:93–103. Available from: <http://www.ncbi.nlm.nih.gov/pubmed/9317405>.
24. Daccordi M., Triberti P., Zanetti A., *Simon & Schuster's Guide to Butterflies & Moths*. New York; 1988.
25. Scoble M.J., *The Lepidoptera: form, function, and diversity*. New York: Oxford University Press; 1992.
26. Steppan S., Flexural stiffness patterns of butterfly wings (Papilionoidea). *Journal of Research on the Lepidoptera* [Internet]. 1996;1996(May 1998):61–77. Available from: <http://bio.fsu.edu/steppan/butterfly.pdf>.
27. Nachtigal W., Wing movements and generation of aerodynamic forces by some medium sized insects. *Insect Flight Symposia of the Royal Entomological Society of London*. 1976;7:31–47.
28. Kim E.J., Wolf M., Ortega-Jimenez V.M., Cheng S.H., Dudley R., Hovering Performance of Anna's Hummingbirds (*Calypte anna*) in Ground Effect. *Journal of The Royal Society Interface* [Internet]. 2014 Jul 2;11(98). Available from: <http://rsif.royalsocietypublishing.org/content/11/98/20140505.abstract>.

29. Urquhart F.A., Urquhart N.R., The overwintering site of the eastern population of the Monarch butterfly (*Danaus plexippus*; *Danaidae*) in southern Mexico. *Journal Of The Lepidopterists' Society*. 1976;30(3):153–8.
30. Howard E., Davis A.K., The fall migration flyways of monarch butterflies in eastern North America revealed by citizen scientists. *Journal of Insect Conservation*. 2009;13(3):279–86.
31. Alonso-Mejía A., Rendon-Salinas E., Montesinos-Patiño E., Brower L.P., Use of Lipid Reserves by Monarch Butterflies Overwintering in Mexico: Implications for Conservation. *Ecological Applications*. 1997;7(3):934–47.
32. Masters a. R., Malcolm S.B., Brower L.P., Monarch butterfly (*Danaus plexippus*) Thermoregulatory Behavior and Adaptations for Overwintering in Mexico. *Ecology*. 1988;69(2):458–67.
33. Ellington C.P., The Aerodynamics of Hovering Insect Flight. I. The Quasi-Steady Analysis. *Philosophical Transactions of the Royal Society*. 1984;305:1–15.
34. Weis-Fogh T., Quick Estimates of Flight Fitness in Hovering Animals, Including Novel Mechanisms for Lift Production. *Journal of Experimental Biology* [Internet]. 1973;59:169–230. Available from: <http://jeb.biologists.org/content/59/1/169.short>.
35. Brackenbury J., Wing movements in the bush-cricket *Tettigonia viridissima* and the mantis *Ameles spallanziana* during natural leaping. *Journal of Zoology* [Internet]. 1990;220:593–602. Available from: <Go to ISI>://WOS:A1990DC94400007.
36. Srygley R.B., Thomas A.L.R., Unconventional Lift-generating Mechanisms in Free-Flying Butterflies. 2002;420(December):487–9.
37. Ellington C.P., van den Berg C., Willmott A.P., Thomas A.L.R., Leading-edge vortices in insect flight. *Nature*. 1996;384:356–8.
38. Dickinson M.H., Lehmann F.O., Sane S.P., Wing rotation and the aerodynamic basis of insect flight. *Science (New York, NY)*. 1999;284(5422):1954–60.
39. Altshuler D.L., Dickson W.B., Vance J.T., Roberts S.P., Dickinson M.H., Short-amplitude high-frequency wing strokes determine the aerodynamics of honeybee flight. *Proceedings of the National Academy of Sciences of the United States of America*. 2005;102(50):18213–8.
40. Hao L., Kawachi K., A numerical study of insect flight. *Journal of Computational Physics*. 1998;146(1):124–56.
41. Ramamurti R., Sandberg W.C., A three-dimensional computational study of the aerodynamic mechanisms of insect flight. *The Journal of Experimental Biology*. 2002;205(Pt 10):1507–18.
42. Willmott A.P., Ellington C.P., The Mechanics of Flight in the Hawkmoth *Manduca sexta*. *Journal of Experimental Biology*. 1997;2722:2705–22.

43. Dudley R., Ellington C.P., Mechanics of Forward Flight in Bumblebees: I. Kinematics and Morphology. *Journal of Experimental Biology* [Internet]. 1990;148:19–52. Available from: <http://jeb.biologists.org/content/148/1/19>.
44. Tanaka H., Shimoyama I., Forward Flight of Swallowtail Butterfly with Simple Flapping Motion. *Bioinspiration & biomimetics* [Internet]. 2010 Jun [cited 2014 Dec 11];5(2):026003. Available from: <http://www.ncbi.nlm.nih.gov/pubmed/20484782>.
45. Walker S.M., Thomas A.L.R., Taylor G.K., Photogrammetric reconstruction of high-resolution surface topographies and deformable wing kinematics of tethered locusts and free-flying hoverflies. *Journal of the Royal Society*. 2009;6(33):351–66.
46. Zheng L., Hedrick T.L., Mittal R., Time-Varying Wing-Twist Improves Aerodynamic Efficiency of Forward Flight in Butterflies. *PLoS ONE*. 2013;8(1):1–10.
47. Ristroph L., Berman G.J., Bergou A.J., Wang Z.J., Cohen I., Automated hull reconstruction motion tracking (HRMT) applied to sideways maneuvers of free-flying insects. *The Journal of Experimental Biology*. 2009;212(Pt 9):1324–35.
48. Walker S.M., Thomas A.L.R., Taylor G.K., Deformable wing kinematics in free-flying hoverflies. *Journal of the Royal Society, Interface / the Royal Society*. 2010;7(May 2009):131–42.
49. Hedrick T.L., Usherwood J.R., Biewener A., Low speed maneuvering flight of the rose-breasted cockatoo (*Eolophus roseicapillus*). II. Inertial and aerodynamic reorientation. *The Journal of experimental biology*. 2007;210(Pt 11):1912–24.
50. Hedrick T.L., Usherwood J.R., Biewener A., Wing inertia and whole-body acceleration: an analysis of instantaneous aerodynamic force production in cockatiels (*Nymphicus hollandicus*) flying across a range of speeds. *The Journal of Experimental Biology*. 2004;207(Pt 10):1689–702.
51. Vicon Nexus 1.8.5. Vicon motion systems Ltd.; 2013.
52. Cranford J., Kang C.K., Landrum D.B., Slegers N., Experimental Characterization of Butterfly in Climbing Flight. *AIAA Aviation Conference - AIAA-2015-2328*. Dallas; 2015.
53. www.swallowtailfarms.com.
54. Orłowski C.T., Girard A.R., Modeling and Simulation of Nonlinear Dynamics of Flapping Wing Micro Air Vehicles. *AIAA Journal* [Internet]. 2011 May [cited 2014 Dec 10];49(5):969–81. Available from: <http://arc.aiaa.org/doi/abs/10.2514/1.J050649>.