University of Alabama in Huntsville
LOUIS

Theses | UAH Electronic Theses and Dissertations

2024

A novel method for time-based synchronization of acquired force and motion data

Darnisha Detraniece Crane

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

Recommended Citation

Crane, Darnisha Detraniece, "A novel method for time-based synchronization of acquired force and motion data" (2024). *Theses*. 664.
https://louis.uah.edu/uah-theses/664

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.

A NOVEL METHOD FOR TIME-BASED SYNCHRONIZATION OF ACQUIRED FORCE AND MOTION DATA

Darnisha Detraniece Crane

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in
The Department of Mechanical and Aerospace Engineering
to
The Graduate School
of
The University of Alabama in Huntsville
May 2024

Approved by:

Dr. Chang-kwon Kang, Research Advisor, Committee Chair
Dr. Farbod Fahimi, Committee Member
Dr. Ryan Conners, Committee Member
Dr. George Nelson, Department Chair
Dr. Shankar Mahalingham, College Dean
Dr. Jon Hakkila, Graduate Dean

Abstract

A NOVEL METHOD FOR TIME-BASED SYNCHRONIZATION OF ACQUIRED FORCE AND MOTION DATA

Darnisha Detraniece Crane

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science

Mechanical and Aerospace Engineering

The University of Alabama in Huntsville
May 2024

Enhancing the performance of flapping wing micro air vehicles hinges upon a thorough comprehension of the intricate interplay between wing kinematics and resulting forces. This understanding necessitates a method of synchronization capable of establishing a precise correlation of wing motion and force generation. The objective of this project is to develop a time-based synchronization method using existing hardware in the ATOM lab. Algorithms were developed to control the Vicon motion-tracking system and the ATI force sensor to achieve the synchronization of acquired data. Experimental testing was conducted to validate the algorithms and system architecture. Results from testing revealed a time lag delay, equivalent to the acquisition time of the force transducer between the two measurements. This delay indicates that synchronization was not achieved. Further investigations suggest that the operating system and the data acquisition board were limited in executing resource intensive input/output bound tasks to support synchronization.

Acknowledgements

I would like to thank my advisor Dr. Chang-kwon Kang for his assistance, advice, and support throughout my time as an undergraduate and graduate researcher. I would like to thank Madhu Sridhar for helping me learn how to use the ATOM lab equipment and Thomas Clark for allowing me to use his experimental wing for testing. Finally, I would like to thank my family, friends, and colleagues who have offered support and advice throughout this process.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

The concept of the micro air vehicle (MAV) was proposed by the Defense Advanced Research Projects Agency (DARPA) in 1996. Micro air vehicles are small, unmanned aircraft that can be autonomous or remotely controlled. Commercial and military use are some of the driving factors for the development of MAVs. Currently, there are three core designs for MAVs: fixed-wing, flapping-wing, and multi-rotor systems. The area of interest for this research is the development of a flapping-wing micro air vehicle with the capability of desirable flight characteristics. A flapping wing MAV with desirable flight characteristics would improve the understanding of biological flyers that use flapping wing mechanisms and could potentially be used for military and space applications.

1.1 Background

Small natural flyers use unsteady aerodynamic mechanisms to produce lift and thrust, which is qualitatively different than large aircraft aerodynamics (Shyy *et al.*2013). Scaling laws indicate that a reduction in the size of a flyer leads to an increase in environmental influence for small flyers – natural flyers overcome this by improving flight performance (force generation, flapping wings, and wing tail coordination, etc.) (Shyy *et al.*2013). While flapping wing flight for natural flyers has improved over time through evolution, manmade flapping wing designs cannot fully emulate these characteristics. As a result, flapping wing designs are continually evolving as researchers continue to study the kinematic and kinetic relationship of flapping wing designs.

Figure 1.1 shows a monarch butterfly – the inspiration behind the design of the artificial wing used for validation of this research project. The flapping motion of the artificial wing – Figure 1.2 – was achieved using a gear box and a motor.

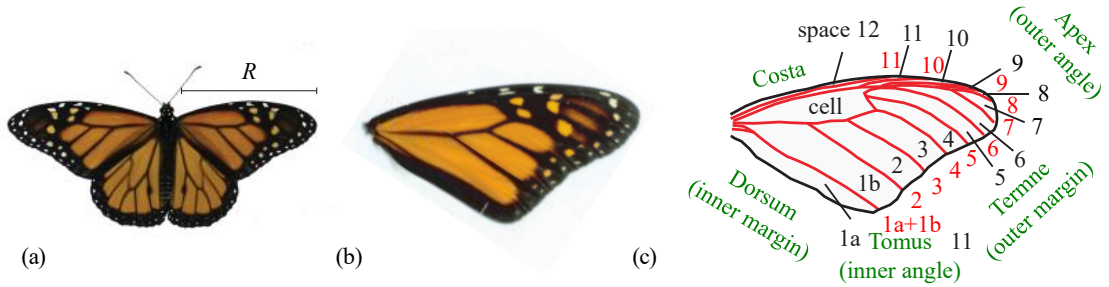


Figure 1.1 Monarch Butterfly and its right forewing structure. (a) A monarch butterfly with a wing length of R. (b) A right forewing. (c) Morphological labels of a monarch butterfly wing, as defined by Emmet and heath: red - veins; black - membrane spaces; green anatomical region names (Twigg, 2020). From Twigg with permission.

Figure 1.2 Micron wing flapper, with the artificial wing mounted on the left side.

Results of a study performed by Sane and Dickinson concluded that subtle alterations in stroke kinematics have large effects on force production (Sane & Dickinson, 2001). An understanding of the relationship between the wing kinematics and resulting forces on a flapping wing is essential for the development and improvement of flapping wings. This understanding requires a method of synchronizing the wing kinematic data and force data which is critical in directly correlating the increases or decrease in force production to the flapping wing kinematics (*i.e.,* stroke amplitude, etc.). Previous synchronization methods have used aerodynamic

characteristics such as angle of attack, upstroke/downstroke, and in some cases the horizonal suction column as a method for synchronizing acquired kinematic and kinetic. Other methods have achieved the synchronization of kinematic and kinetic data by deriving the forces acting on the object using the captured motion data. While these methods do achieve the desired results, acquiring the data based on the same timing trigger would result in an increase in the accuracy of the data correlation.

The primary challenge that exists with achieving synchronization is technological capability. First, computers are limited in their ability to execute multiple tasks simultaneously. Specifically, Input/Output (I/O) bound processes cannot be executed simultaneously due to them being resource intensive. That means that the computer used directly affects the ability to achieve synchronization and matters greatly when attempting data synchronization. Multicore, multiprocessor, and supercomputers are some of the computer types available. Multicore computers have an increased number of cores allowing for a task to be split amongst the cores for more efficient execution. Multiprocessor computers have multiple central processing units (CPUs) which allow for parallel execution of tasks – supercomputers fall into this category as well. However, what separates a supercomputer from what most users have – traditional computers – is that supercomputers function at the highest operational rate or peak performance for computers. The use of multiprocessor computers would achieve the synchronization task. However, the use of these computers would not be feasible for researchers to use due to cost. Another challenge associated with technological capabilities is the use of specific operating systems. For a task as resource intensive as simultaneous acquisition, the Linux operating system would be most suitable due to the speed in which it executes tasks and its reliability. However, Linux is limited in terms of hardware driver support meaning that there may be challenges in

finding hardware equipment that is compatible with the Linux operating system. The Windows operating system is compatible with most hardware indicating that there would be less challenges associated with finding compatible hardware. However, this operating system has reliability issues and the speed in which it executes tasks can degrade over time.

1.2 Objective

The objective of this project is to method t that allows for acquiring kinetic (force) and kinematic (motion) data points simultaneously by developing a software toolset. While this toolset is intended for application with flapping wing MAV development, it could potentially be used for other areas of research such as biomechanics and kinesiology. The developed software tools need to work with the existing lab equipment and should be able to work with any test setup. Additionally, the method should be developed using a system engineering approach.

1.3 Requirements

Requirement definition was critical in identifying the constraints for the project and defining the expectations for the developed method. The requirements for the developed software suite are as follows:

- Ability to integrate with the existing hardware.
- Acquire force data from the ATI Nano 17 force/torque transducer and motion data from the Vicon Motion Capture Camera System simultaneously.
- The software tools need to be developed using software programs compatible with the existing hardware).
- Execution of the toolset should occur without software failures.

- Develop method using a system engineering approach (*i.e.*, requirements definition, system architecture design, system integration, etc.).

1.4 Outline

Chapter 2 of this thesis describes a literature study in which three topics are discussed. The first topic is a discussion of force generation and wing defamation characteristics. Additionally, established methods of synchronizing data and acquiring data from multiple devices are discussed. Chapter 3 covers the system design and algorithm development section which includes a summary of the system architecture, the algorithm development process for the selected software tools, and the system implementation process. In Chapter 4, results from the testing and validation process are discussed. Furthermore, alternative system designs and solutions are proposed. In Chapter 5, the conclusion is stated, future work is defined, and novel contributions are presented.

Chapter 2. Literature Study

The literature review discusses three primary topics. First, a summary of studies related to force generation and wing deformation of flapping wings is discussed. This section discusses findings made through research on the forces acting upon and deformation characteristics observed from a flapping wing in motion. The section also discusses the importance of understanding how force generation is influenced by wing deformation respective to improving existing flapping wing designs. Next, a discussion of methods used historically to acquire kinematic and kinetic data will take place. The literature study will conclude with a discussion of established methods used to synchronize acquired data. This review influenced the algorithm development process and provided a baseline for the process used to synchronize the data acquisition.

2.1 Measurements of Force Production

The relationship between wing deformation and how that deformation affects aerodynamic force generation has been studied extensively (Shyy *et al.*2013). An understanding of the relationship between force production and wing motion is essential for the continual development of flapping wing MAVs and improvement of knowledge regarding biological flyers capable of flapping wing flight.

Bahlman *et al.* (Bahlman *et al.*2013) constructed a robotic bat wing to circumvent the problem of trying to measure power input and force from a flying vertebrate. A range of kinematic parameters were used to observe the force output and power input for the designed bat

wing. This study resulted in the discovery that the wing would drift forward during the downstroke from thrust generation and due to inertia. However, despite the drifting issues, researchers concluded that the creation of the wing would allow for a better understanding of aerodynamic performance of flapping wings without having to use a live specimen (Bahlman *et al.*2013).

The wing deformation characteristics and aerodynamic force generation of a DelFly II MAV in hovering flight was observed for the purpose of posing alternative wing designs (Percin *et al.*2016). A force transducer and particle image velocimetry (PIV) system were used to observe force generation, power consumption, and defamation characteristics for trials in which flapping frequency and wing configuration was altered – specifically wing thickness, wing layout, and material were varied (Figure 2.1) .

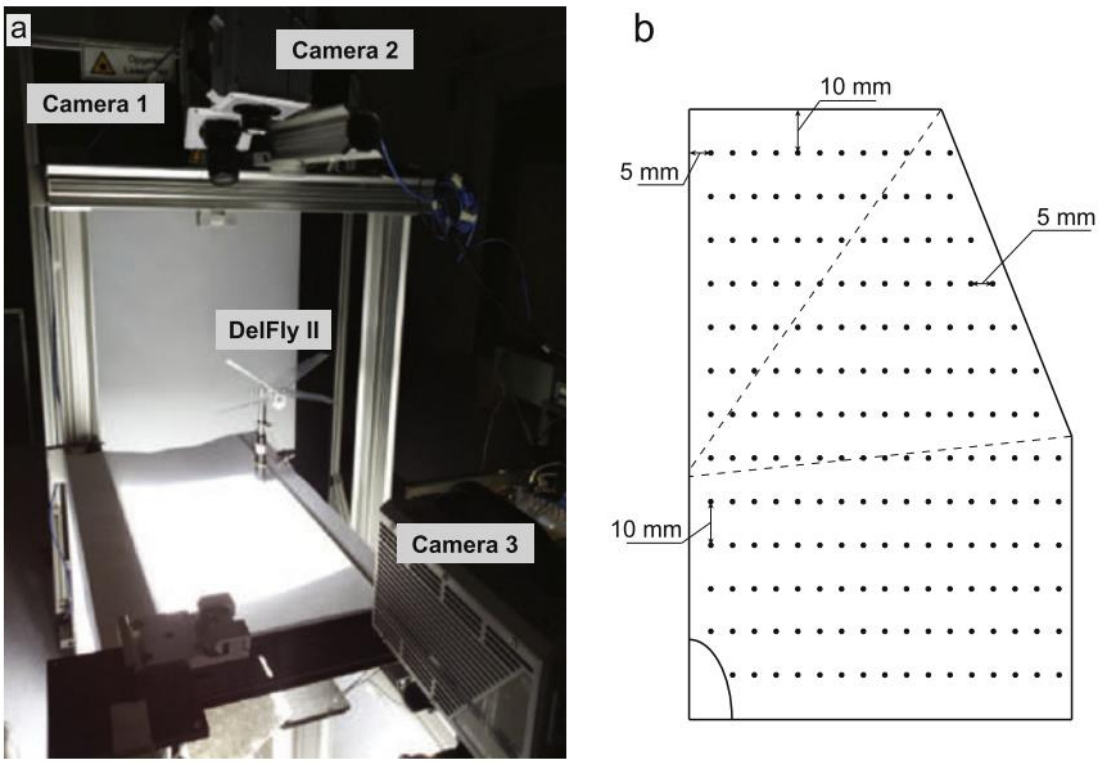


Figure 2.1 (a) Experimental setup for the wing structural deformation measurements. (b) Sketch of the wing layout with circular markers (Percin *et al.*2016). From Percin et al with permission (Percin *et al.*2016)**.**

They concluded that an increase in flapping frequency resulted in an increase in force generation with respect to the x component of the three-dimensional force vector and power consumption, but also introduced a phase lag in the temporal evolution of forces. Another conclusion stated was that an increase in the thickness of the wing material (Mylar) resulted in an increase in both X-force and power consumption which in turn decreased the ratio of force to power (Percin *et al.*2016). The study presents more conclusions derived from the testing variations conducted with the DelFly II MAV related to testing with thicker or thinner membranes, different span lengths, and aspect ratios. However, the significance presented with this study is that an understanding of force generation as it relates to wing deformation and in turn wing design is essential for the optimization of the wing itself.

In a study conducted by Thomson *et al.* (Thomson *et al.*2009), a hawkmoth inspired wing was fabricated and tested to determine the optimal flight trajectory that would maximize average vertical force. The wing was connected to a flapping mechanism comprised of gears, an encoder, and motors. The flapping mechanism was mounted above a load cell and a LabVIEW developed controller was used to send desired trajectories to the encoder for testing force (Thomson *et al.*2009). This test was conducted iteratively with the output of one trial (*i.e.*, angle and force data) serving as the inputs of the next until the optimal trajectories were reached. This study showed that understanding the force production as a result of flight trajectories was essential in discovering the trajectory that would maximize the force production (Figure 2.2).

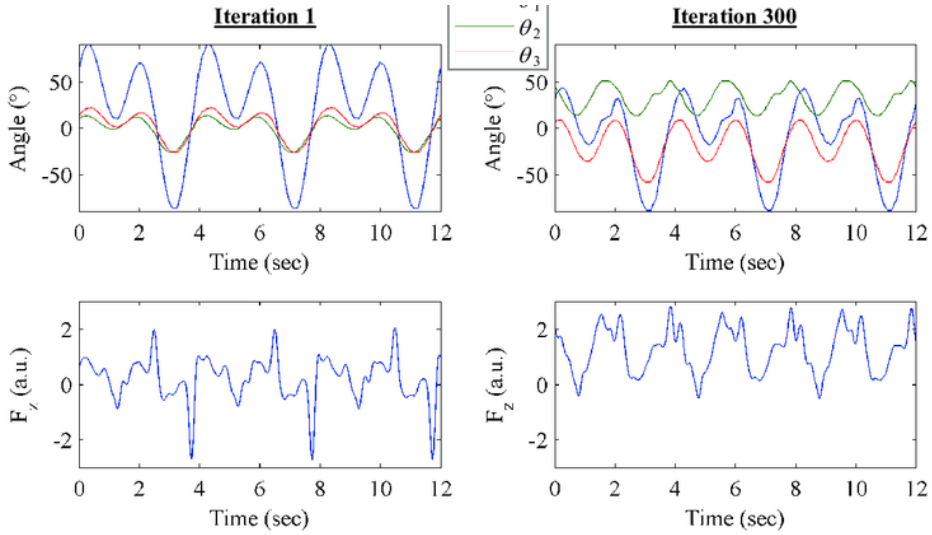


Figure 2.2 Rotation angles vs time for iteration 1 (top left) and iteration 300 (top right). Instantaneous vertical force (arbitrary units) vs. time for iteration 1 (bottom left) and iteration 300 (bottom right) (Thomson *et al.*2009). From Thomson *et al.* with permission (Thomson *et al.*2009).

2.2 Established Methods for Acquiring Kinematic and Kinetic Data

Section 2.1 focused on the key trends observed during studies conducted related to unsteady aerodynamics and their correlation to wing deformation characteristics. While this

section includes information related to the topic, it primarily focuses on the methods used to measure wing deformation and unsteady aerodynamics. Several methods have been used historically to acquire kinematic and kinetic data. An established method used for acquiring both kinematic and kinetic data has been using a load cell in combination with a camera system. While investigating the aerodynamic characteristics of an experimental flapping wing, Wu and his co-workers (Wu *et al.*2011) used a four camera Digital Image Correlation (DIC) system to measure the kinematics and deformation of a flapping wing while measuring the aerodynamic forces acting on the wing using an ATI Nano 17 Titanium force transducer (Figure 2.3). The acquisition of the data informed key decisions on wing design.

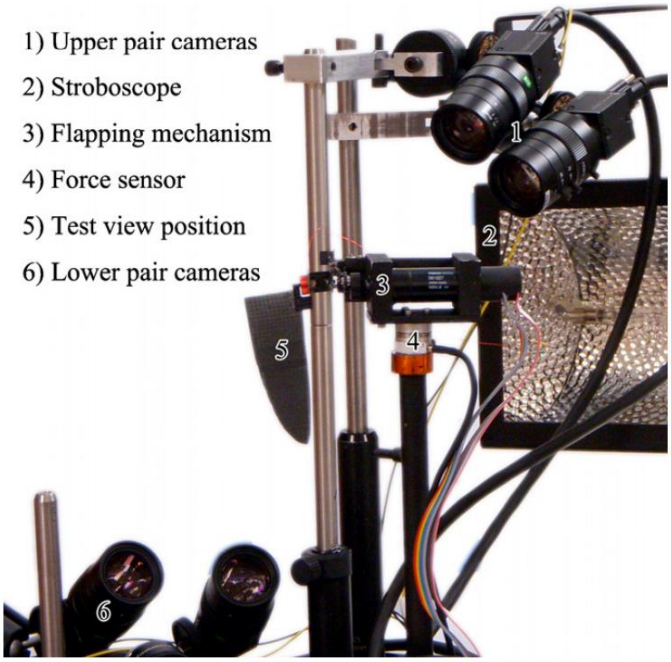


Figure 2.3: The test stand used by Wu *et al.* includes four Point Grey Research Flea2 cameras, a stroboscope, and an ATI Nano 17 force/torque transducer mounted underneath the wing (Wu *et al.*2011). From Wu *et al.* (Wu *et al.*2011) with permission.

In another study, a two camera DIC system in conjunction with an ATI Nano 17 force transducer (Figure 2.4) was used to capture kinematic and kinetic data using a dove inspired

flapping wing MAV. Additionally, a hall angle sensor was incorporated to capture the flapping angle (Yang *et al.*2022). In both studies, the use of the DIC camera system and ATI Nano 17 force transducer were critical in capturing force data and wing kinematics which allowed for researchers to understand the data in relation to the performance of flapping wings.

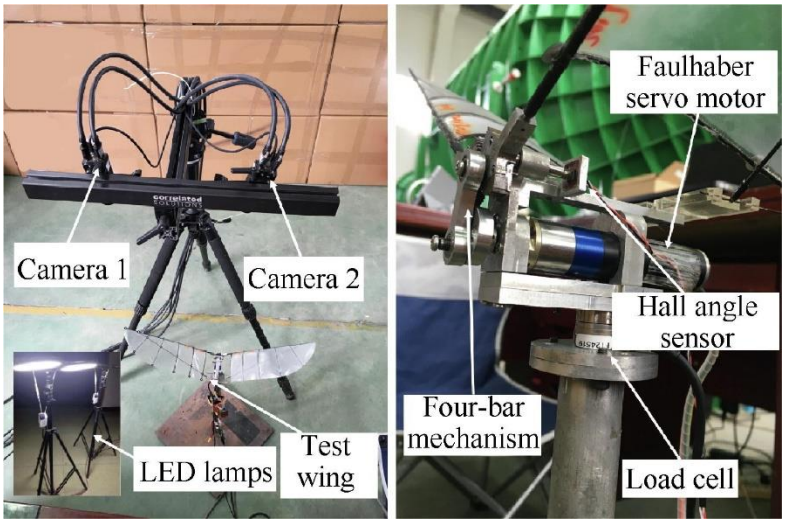


Figure 2.4 Measurement of wing deformation and forces (YANG *et al.*2022).

While the use of a force transducer and camera system is an established method for the acquisition of force kinematic and kinetic data, other methods are used to acquire the data. For rigid wings, Sane and Dickinson (Sane & Dickinson, 2001) used stroke position (ϕ), angle of attack (α), and stroke deviation (θ) to describe the kinematics of the wings and move a dynamically scaled wing in oil. The stroke position was described by a triangular waveform, a trapezoidal waveform was used to describe the angle of attack, and two functions were used to describe the stroke deviation. An 'oval' pattern was used to describe the wing tip deviation from the stroke plane according to a half-sine wave (Sane & Dickinson, 2001). A 'figure-of-eight' pattern was used to describe stroke deviation which was varied as a full sine wave (Sane &

Dickinson, 2001). Force measurements were acquired using a two-dimensional sensor acquired using a National Instruments BNC 2090 data acquisition board (Sane & Dickinson, 2001). Similar to the previously discussed research studies, kinetic data were measured and observed. However, in this study kinematics were described using waveform functions – specifically, custom MATLAB programs were used to convert the angular trajectories into stepper motor commands (Sane & Dickinson, 2001).

Kinematic and kinetic data can also be measured using a magnetic position sensor and load cell, respectively. In a study conducted by Leys *et al.* (Leys *et al.*2016), the kinematic and kinetic performance of a hummingbird inspired flapping wing mechanism was observed. The kinetic performance, specifically thrust, of the flapping wing mechanism was measured using a double beam load cell. The kinematic performance, specifically stroke angle (θ), was measured using a AS5055A contactless magnetic position sensor. A key aspect of this study was to prove that a magnetic position sensor can acquire accurate kinematic which provides a less expensive alternative when compared to high-speed cameras.

Deriving forces using acquired motion data is another proven method used to acquire both kinematic and kinetic data. Meng and Sun (Meng & Sun, 2016) acquired motion data using three high speed cameras and then used stereovision-based triangulation to extract the three-dimensional body and wing kinematics. For reference, stereovision-based triangulation identifies a point in three-dimensional space from the left and right pixel in a pair of stereo images. The inputs of using this method are the homogeneous coordinates of the detected image points and the camera matrices for the left and right cameras. The output of the triangulation method is a three-dimensional point in the homogenous representation (*Computer Vision: Stereo 3D Vision*, 2022). Following the collection of images and extraction of kinematic data, a Computation Fluid

Dynamics (CFD) method was used to derive thrust and vertical force coefficients by solving the Navier Stokes equation. In another study, a motion capture system comprised of an Optitrack Flex V100R2 hardware and Arena Motion Capture Software was used to capture the motion of subjects performing body weight squats while a Rice Lake Weighting Solution's force plate was used to measure ground reaction forces (Fry *et al.*2016). Force data was sampled using a Biopac data acquisition system and LabVIEW was used to control sample rate. Following data acquisition, ground reaction forces were derived from the data captured by the motion capture system using proprietary methods (Moodie, 2013) and derived forces were then compared to data collected from the force plate (Fry *et al.*2016).

2.3 Established Methods for Synchronizing Kinematic and Kinetic Data

Synchronizing acquired kinematic and kinetic data is essential for understating the relationship between the two. Synchronization methods are necessary to ensure that the kinematic and kinetic data correlation is accurate. Established methods for data synchronization include using software, microcontrollers, and aerodynamic characteristics.

Software based methods have been used previously to synchronize kinetic and kinematic acquired data. A C++ based software algorithm was developed to trigger image acquisition at specific wing positions for a DelFly II MAV in simulated hovering flight. Force measurements were captured using a strain gauge with Wheatstone bridges. In this study, two Lavision HighspeedStar CMOS cameras were used to capture motion data – this acquisition process was triggered by the PC used to control the motion of the MAV's wings. The vertical force of the object was measured using a strain-gauge balance and two Wheatstone sensors. The tracking of motor pulses during testing allowed for the acquired force data to be synchronized with the wing location (De Clercq *et al.*2009).

Microcontroller boards have also been used to synchronize acquired kinetic and kinematic data. A microcontroller is a small computer on an integrated circuit and is used to execute specific functions within electronic systems. Microcontrollers typically contain a processor, memory, and input/output (I/O) peripherals. These boards can also be used to trigger and control external systems. A microcontroller board was used to synchronize the stereo-PIV system and force measurements while also controlling the flapping frequency during a study conducted on the Del Fly Micro MAV (Deng *et al.*2016). In this study, force measurements were recorded using an ATI Nano 17 Titanium force transducer and three Photron Fastcam SA1.1 high speed cameras to record the flapping motion – the microcontroller was used to synchronize both systems based on the flapping frequency (Deng *et al.*2016). A similar experimental setup was also used to examine the kinematic and kinetic performance for the DelFly II MAV. Similar to the previously discussed study, a microcontroller was used to control the flapping frequency, synchronize the force, and image acquisition to flapping frequency. In addition to the microcontroller, a National Instruments field-programmable gate array (FPGA) was used for the data acquisition (Percin *et al.*2017).

A method used previously to correlate acquired kinematic and kinetic data has been to use wing deformation characteristics and non-dimensional time. The stroke angle for the DelFly II MAV in hovering motion was calculated using the mechanical model of the driving system. Concurrently, the forces acting on the DelFly II were measured using an ATI Nano 17 Titanium force transducer. Stroke angle and acquired force were then plotted against non-dimensional – time in seconds divided by the period of a flapping cycle – to show the relationship between stroke angle and the generated forces (Percin *et al.*2017).

For the purpose of distinguishing between the synchronization methods mentioned previously and the synchronization method that is the focus of the project, synchronization will be separated into two categories: time-based synchronization and characteristic based synchronization. Time-based synchronization is data synchronization in which the external devices start acquiring data at the same time based on a shared internal clock. Characteristic based synchronization is achieved by triggering acquisition based on the characteristics of the test specimen or parameters of the trial.

To put this idea into perspective, acquisition is triggered based on the flapping angle set for a flapping-wing MAV during a trial. Characteristic based synchronization usually occurs during the post-processing of data. Based on typical characteristics of the wing motion, *e.g.,* the flapping angle, the force data are shifted to match the kinematic motion. The research studies mentioned previously are all examples of characteristic based synchronization. One exception may be the study by Deng *et al.* (Deng *et al.*2016). This appears to be an example of the time-based synchronization because the microcontroller board acts as the control for two external acquisition systems and synchronizes the measurements for both systems. A key observation to make in regard to this study is that the microcontroller's primary purpose was to control the flapping frequency of the Del Fly II and to synchronize measurements from each system with the flapping angle of the MAV. The actual correlation of force and motion data occurred during the post-processing phase of the data using the flapping angle as the basis for correlation.

The characteristic based synchronization is useful in the correlation of wing motion and force production, the accuracy of the results can be called into question. However, the emphasis on observing force production at specific flapping angles such as the upstroke or downstroke of a flapping wing could result in trends in force production at other flapping strokes being

overlooked. On the contrary, time-based synchronization would allow for a correlation force production and motion throughout the flight of the test specimen allowing for all potential trends in force production to be observed and increase the accuracy of the results captured. The focus of this study is the development of a time-based synchronization method which will allow for an accurate and direct correlation of force production and wing motion.

Chapter 3. System Design and Algorithm Development

The goal of this project was to develop a method for acquiring measurements simultaneously from two external systems using a systems engineering approach. In order to achieve this objective, a system architecture had to be defined with the necessary hardware components used. Subsequently algorithms had to be developed to integrate these components together. This chapter details the design of the system architecture, algorithm development process, and system implementation.

3.1 System Architecture Design I

The first step in approaching this task was defining an initial system architecture showing the hardware components with the system along with the connections indicating integration points (*i.e.*, algorithms). Figure 3.1 is a representation of this initial system architecture. Hardware components are represented by blue boxes. The gray boxes indicated areas where algorithm development was necessary for hardware integration.

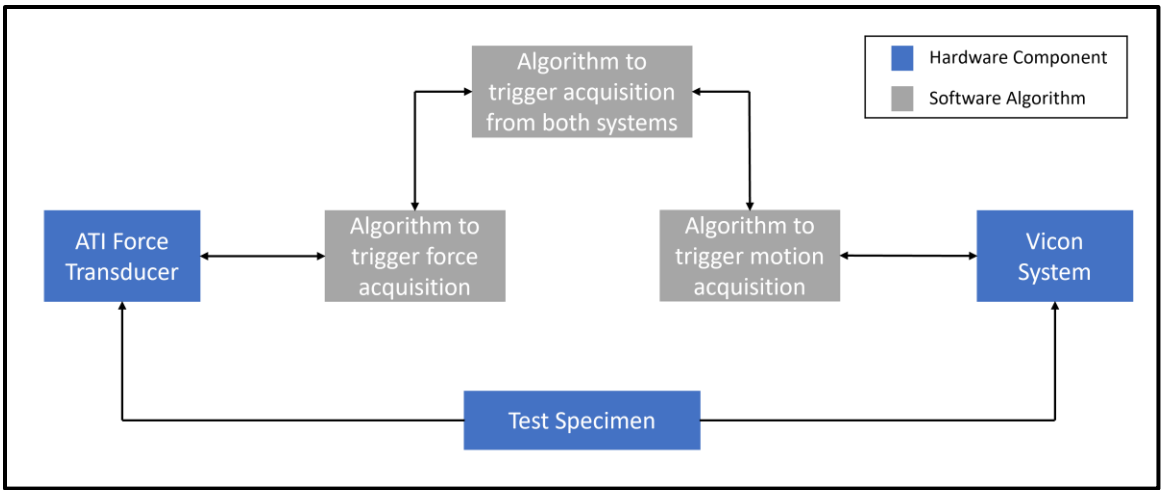


Figure 3.1 Block diagram shows the initial system architecture. The diagram includes the connection of the hardware components to the algorithms that are under development.

3.2 Algorithm Development

Algorithm development was critical in integration of the hardware components. However, before algorithm development could occur, software tools had to be selected that would serve as the development environment for the algorithms. Since the hardware components presented in the system architecture strongly influenced the selection of software tools, a non-trivial process had to be used for the selection of software. Specifically, the following sub-sections discuss the software tools selected for integration with the National Instruments Data Acquisition Device (NI DAQ) connected to the ATI force transducer and the Vicon camera system, respectively. The process involved selecting the software used for algorithm development by investigating from a set of software tools that are compatible with the existing hardware components. Although the processes discussed pertain to the components specified, they can be applied to other hardware products.

3.2.1 Software Tool Selection for the NI DAQ

As mentioned in the introductory section, the selection of the software tool used to develop the force acquisition algorithm was dependent on the hardware itself. For force acquisition, the NI DAQ has the task of reading, storing, and transmitting data from the ATI force transducer. This meant that the software tool selected had to be capable of triggering force acquisition from the DAQ while also supporting the tasks that the DAQ is responsible for. The NI DAQ is compatible with the following software programs: LabVIEW, LabWindows / CVI, Measurement Studio, and Visual Basic. Table 3.1 provides a brief overview of the programs – this overview includes a description of the tool, primary purpose, and capabilities.

Table 3.1 The table shows an overview of the software tools that are compatible with the NI DAQ USB 6110.

Software Program	Development Environment Type	Programming Language	Description	Key Capabilities
LabVIEW	Graphical	G	Program developed by National Instruments. Graphical programming environment that allows for signal generation, measurement analysis, and automation of data acquisition from hardware.	• Works with third-party hardware • Acquire data and control hardware • Monitor and control tests with UI development • Integrate with Python, C/C++, .NET, MATLAB
LabWindows / CVI	Textual	X	Program developed by National Instruments. ANSI C development environment that allows for the creation of test and measurement applications.	• Works with third-party hardware • Develop applications for real-time measurements • Control hardware virtually

Software Program	Development Environment Type	Programming Language	Description	Key Capabilities
				• Provide intuitive debugging capabilities
Measurement Studio	Graphical	C, VBA, etc. (.NET framework)	Developed by National Instruments as an extension of Microsoft Visual Studio. .NET tools used to design applications for acquiring, analyzing, and displaying measurement data.	• Works with third-party hardware • Remove complexities of hardware communication through object-oriented hardware class • Perform real-time analysis on acquired signals
Visual Basic	Textual	VBA (.NET framework)	Developed by Microsoft. Programming language that allows for development of web applications and GUI development.	• Works with third-party hardware • Communicate with hardware through ActiveX components • Provide object-oriented programming • Access to basic data acquisition function in Visual Basic

The following paragraphs discuss the differences between these software programs and the decision process used to down select to the program used to develop the force acquisition algorithm.

LabVIEW is a graphical development environment that allows for the development of applications to acquire from and control third-party hardware. Key advantages of this software tool are that it allows for automation of acquisition and can integrate with other software tools like Python and MATLAB. A disadvantage of LabVIEW is that the cost of the tool can increase

depending on the functionality required. Also, debugging in the LabVIEW environment can be more complex when compared to other development environments.

LabWindows / CVI is a textual development environment that uses the C programming language. Similar to LabVIEW, the tool is capable of connecting to third-part hardware and acquiring data using developed applications. A disadvantage of using this programming environment is that the dependency on the C programming language can create a steep learning curve for users who are not experienced with the programming language.

Measurement Studio was developed by National Instruments to serve as an extension of the Microsoft Visual Studio program. This program relies on .NET tools and the .NET framework to allow for the designing of applications for data acquisition and for connecting to hardware. An advantage of this program is that the graphical environment allows users to develop interfaces capable of being integrated with hardware. However, there is a disadvantage with using this program. Since the tool is an extension of Visual Studio, it does require that users have some experience with using programming languages like C/C++ and VBA which can create a steep learning curve for users.

Visual Basic can connect to and acquire from the DAQ by using Active X components. This poses a disadvantage to the program because the components have limitations in terms of security and compatibility with other operating systems (OS), such as macOS and Linux, and programming environments. Also, data acquisition from DAQ is limited to what functions are available in the Visual Basic programming environment.

Given the advantages and disadvantages of each development environment, LabVIEW was selected as the tool for the development of the force acquisition algorithm. The primary reason LabVIEW was selected was because of its ability to integrate with other programming

tools. Referring back to Figure 3.1 which shows the first design of the system architecture, one of the key algorithms needed for the task was an algorithm that was capable of triggering acquisition from both systems. LabVIEW's ability to integrate with other programs meant that it could be connected to and triggered from another program which would support the automation of data acquisition required to satisfy the objective of the project.

3.2.2 Software Selection for the Vicon Camera System

Nexus is a software tool developed by Vicon Motion Systems to allow users to model and process measurement data. It was specifically designed for the life sciences community. A key feature of Nexus is that it "delivers precise, repeatable data, and clinically validated model outputs" (NEXUS, 2023). Nexus is compatible with both MATLAB and Python –each software suite included the Nexus API. This API allows users to connect to Vicon Nexus for offline data access and model development.

Tracker is a software tool developed by Vicon Motion Systems that allows users to capture high quality real-time data. The tool suite was designed for engineering applications such as drone tracking and human factors engineering. The Vicon Tracker software is compatible with MATLAB, specifically MATLAB Simulink, as well as Python. For MATLAB, the Vicon Data Stream SDK provides the connection between Vicon and MATLAB Simulink. For Python, the Tracker API allows Python to access Vicon camera system through Tracker functions.

While Nexus can be used to acquire motion data, the primary function of the NEXUS API is to allow for streamlined data processing and model development. It was not designed to automate the acquisition process. As a result, Nexus is not suitable for supporting the motion acquisition algorithm development.

Tracker is also capable of acquiring motion capture data and like Nexus, it works with both MATLAB and Python. Unlike Nexus, the connection between Tracker and MATLAB is different when compared to the connection between Tracker and Python. The connection to MATLAB from Tracker is achieved via the Vicon DataStream SDK (TCP/IP connection) or UDP stream. In each case, data from Vicon can be streamed to MATLAB via Simulink using one of the previously mentioned connections. However, one requirement of using these connections is that Vicon must be already streaming data before Simulink can access it. Therefore, Simulink cannot trigger acquisition and is instead dependent on Tracker. So, using Tracker with MATLAB would not support algorithm development for acquiring motion data because automation is not possible using Simulink.

Python is able to connect to Tracker through the Tracker API – this API was introduced with Tracker version 3.9 in 2020. It contains functionality that allows Python to trigger data acquisition through its capture processes. The API also contains functionality that allows Python to trigger camera calibration and data exporting. Given the capabilities contained within the Tracker API, developing an algorithm to automate motion capture acquisition is possible. Therefore, the use of Tracker would support the development of an algorithm that satisfies the objective of this project..

3.3 Final System Design

For the development of the force acquisition algorithm, LabVIEW was selected for the algorithm development. Vicon Tracker in conjunction with Python was selected for the development of the motion capture algorithm. Since Python is capable of being integrated with both LabVIEW and Vicon Tracker, it was selected for the development of the algorithm that triggered acquisition from both systems simultaneously. The first design of the system

architecture shown in Figure 3.1 was updated with the selected software tools and the final system architecture is shown in Figure 3.2. This updated architecture describes the interaction between the major components of the system and displays how the developed algorithms interact with the software and hardware components of the system.

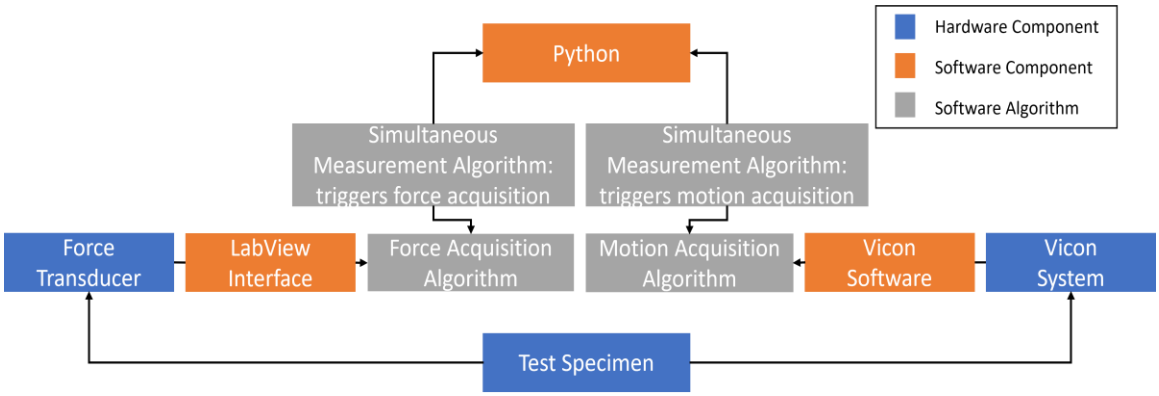


Figure 3.2 Block diagram shows the final system architecture. The diagram includes the connection of the hardware and software components as well as the developed software algorithms.

3.3.1 Components in System Architecture

The components shown in the system architecture (**Figure 3.2**) are critical to the execution of the data acquisition process and contain a mixture of software and hardware components. The key components in the system architecture are as follows:

- Test Specimen
- ATI Nano 17 Force/Torque Transducer
- Vicon Motion Capture Camera System
- Data Acquisition Device (NI USB 6210 DAQ)
- Python
- LabVIEW

- Vicon Tracker Software

3.3.2 Component Limitations

This section addresses the limitations encountered when using the components in the full architecture. The limitations of specific hardware components directly affect the configuration of the system. Additionally, limitations in both hardware and software components were a driving factor behind the selection of specific software builds that were used during the algorithm development process. Table 3.2 identifies the limitations associated with specific system components and how the component may affect other components within the system architecture.

Table 3.2 The table shows the limitations of the hardware and software components used the in the final system architecture.

Component Limitations		
Component	**Limitation**	**Affected Components**
Test Specimen	-	-
ATI Nano 17 Force/Torque Transducer	Constrained to Windows OS due to DAQ	Python and LabVIEW components are constrained to Windows OS
Vicon Motion Capture Camera System	Constrained to Windows OS	Python version is constrained to Windows
Data Acquisition Device (NI USB 6210 DAQ)	Constrained to Windows OS due to lack of support on other operating systems	Python version is constrained to Windows
Python	-	-

LabVIEW	Constrained to Python Version 3.6 and later releases	Python version selection is dependent on LabVIEW compatibility
Vicon Tracker Software	Constrained to Tracker version 3.9	Python version selection is constrained to 3.x and later releases

3.4 LabVIEW Edition Selection

This section discusses the process used to identify the LabVIEW edition that would satisfy the objective of the project. The following paragraphs provide a brief overview of the LabVIEW software and a description of the LabVIEW editions available. The section concludes with a discussion of the pros and cons associated with each edition.

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a graphical programming environment that is used by engineers to develop automated research, validation, and production test systems (*Select Your LabVIEW Edition,* 2023). The software suite was developed by National Instruments (NI).

LabVIEW has three editions available for purchase: LabVIEW Base, LabVIEW Full, and LabVIEW Professional. Each edition includes the following capabilities:

- Acquire data from NI and third-party hardware and communicate using industry protocols (*Select Your LabVIEW Edition*, 2023).
- Create interactive UIs for test monitoring and control (*Select Your LabVIEW Edition,* 2023).
- Utilize standard math, probability, and statistical functions (*Select Your LabVIEW Edition,* 2023).
- Integrate code written in Python, C/C++, .Net, and MathWorks MATLAB® software (*Select Your LabVIEW Edition,* 2023).

- Save data to .csv, .tdms, or any custom-defined binary file (*Select Your LabVIEW Edition*, 2023).

In addition to the capabilities presented above, the LabVIEW Full edition also includes advanced analysis algorithms, signal processing functions, and signal generation functionality. The LabVIEW Professional edition includes the capabilities introduced with the LabVIEW Full edition as well as an Application Builder, Report Generation Toolkit, Advanced Signal Processing Toolkit, code comparison tools, and a Database Connectivity Toolkit. Each edition is supported on the Windows operating system. However, only the Full and Professional editions are supported on the Mac and Linux operating systems.

The selection of the LabVIEW edition that was most suited to supporting this project was determined using the following criteria:

- Ability to support meeting the objectives and requirements defined in Sections 1.3
- Ability to function with the full system architecture given the limitations of the physical system identified in Section 3.3.2

While each edition met the second criteria listed above, assessing whether the software builds satisfied the first criteria required more investigation. Due to limitations with the device used to capture force data, the build selected had to also support meeting the objectives of the project. As mentioned previously, the device's compatibility with existing operating systems is limited to Windows. However, with respect to the LabVIEW edition selection process, this limitation was negligible as all editions are compatible with the device. When comparing the three editions in regard to the first criteria, the selection process was not straightforward and required further discernment. While each edition satisfied the requirement of acquiring data from external hardware, further scrutiny had to be applied when determining whether the editions supported

the objectives of the project and whether the added features of each edition would contribute to the success of the developed method. Initially, the LabVIEW Full edition was ruled out as a viable candidate for the selection process. Although the signal processing toolbox included with the Full edition posed an advantage over the base version, the toolbox was not deemed as critical to meeting the objective of this project. Signal processing does play a role in the final product; however, LabVIEW was not used to oversee this task. This logic was also applied to the Professional edition which includes an advanced signal processing toolbox as well as the signal processing toolbox of the Full edition. This added capability was weighted lowly when the final determination of the edition was made. When comparing the Base and Professional editions, the Professional edition had a significant advantage in terms of the Application Builder functionality that was incorporated. The builder allows users to convert LabVIEW virtual instruments into applications that can be executed outside of the LabVIEW environment – the only prerequisite to this being that the LabVIEW Runtime Engine (a free application provided by National Instruments) had to be installed for execution to occur. The potential implications of using this edition would be that the need to continuously renew a LabVIEW subscription would decrease due to the capability of running developed code outside of its development environment. However, the upfront costs of the edition as well as the risk of potentially needing to repurchase the software should the existing application require changes puts the Professional edition at a disadvantage. Although, utilizing the Base version does require that the user execute tasks within the LabVIEW environment which in turn requires a yearly subscription renewal, it is far more cost effective than the Professional version. It was decided that the Base version of LabVIEW was best suited to support this project. The edition is more cost-effective and contains the necessary tools needed to interact with the existing external devices and acquire data from the

system. Additionally, it was determined that the need to use LabVIEW can potentially be eliminated by replacing critical LabVIEW instruments with Python developed modules, making the need to incorporate the Application Builder inconsequential.

3.5 Force Acquisition Algorithm Development

During the algorithm development process for force acquisition, the LabVIEW virtual instrument (VI) development was key in supporting the data acquisition requirement defined in Section 1.3. Specifically, the LabVIEW VIs are utilized for acquiring data from the force/torque transducer. This section covers the virtual instruments that are critical to this acquisition task. These LabVIEW VIs will be referred to by its process and not by the name of the virtual instrument itself. The main algorithm for force acquisition is tasked with issuing commands to the sub-algorithms that contain most of the processes completing the acquisition tasks. The sub-algorithms that perform the acquisition tasks are Acquire, Save Data, Process Data, and Load Data.

3.5.1 Force Acquisition Algorithm

The force acquisition algorithm commands the sub-algorithms to execute acquisition tasks. This algorithm is organized using case structures and list variables. The front panel of the algorithm contains buttons that link to the case structures for each sub-algorithm. Selecting each button prompts LabVIEW to execute the code contained within the defined case structures and provides feedback to the user regarding the status of the code execution. In addition to the buttons, the front panel contains a graph this is automatically updated with force data once the algorithm recognizes that data has been stored in the variable tied to the graph – data is

populated if the user opts to acquire data from the force transducer or load previously acquired data.

3.5.2 Acquire Data Algorithm

The acquire data algorithm (Figure 3.3) is responsible for acquiring data from the force/torque transducer. Inputs such as sample size and trial name are passed to this algorithm in the form of control objects (a static object used to pass reference values from the force acquisition algorithm to sub-algorithms). Inputs are then used with the National Instruments data acquisition toolbox (NI-DAQmx) to extract raw voltage data from the force transducer through the DAQs analog output channels. Voltages are converted to force and torque data using the force transducer's calibration matrix and the newly converted data is stored in a new control object – information stored in this control object is sent to the force acquisition algorithm and distributed to other sub-algorithms.

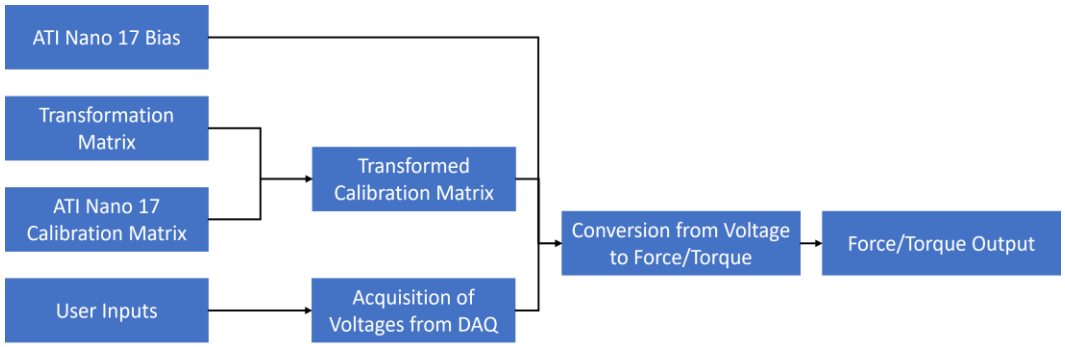


Figure 3.3: Process diagram for the acquire data algorithm. The diagram includes the primary inputs and sequential steps that the algorithm executes to acquire force data.

3.5.3 Process Data Algorithm

The process data algorithm performs the post-acquisition processing of the acquired data. This algorithm (Figure 3.4) establishes a connection to Python using the 'Python Open Session' node in LabVIEW. Following successful connection, LabVIEW sends inputs from the user and the stored data to Python through the LabVIEW Python node. Data is then processed within LabVIEW using a Python developed processing algorithm. Following successful execution pf the Python processing algorithm, LabVIEW ends the connection to Python and concludes its execution. The post processing script is discussed in further detail during the discussion of the Python algorithm development process in Section 3.6.2.

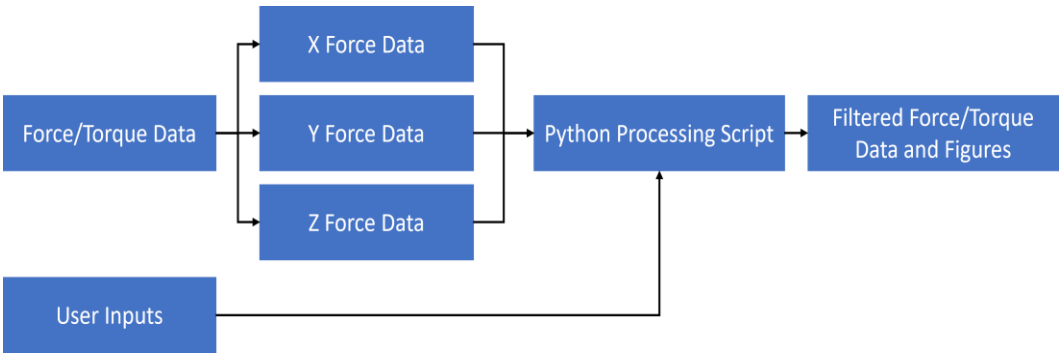


Figure 3.4 Process diagram for the save data algorithm. The diagram includes the primary inputs and sequential steps that the algorithm executes to process the force data using Python.

3.5.4 Save Data Algorithm

The save data algorithm (Figure 3.5) is used to save the acquired or loaded data. LabVIEW saves the data to the path specified by the user and in the format specified. Users have the option to save data in the following formats: .csv, .tdms, and .bin.

Figure 3.5 Process diagram for the save data algorithm. The diagram includes the primary inputs and sequential steps that the algorithm executes to save acquired data.

3.5.5 Load Data Algorithm

The load data algorithm (Figure 3.6) is used to load previously acquired data. The loaded data is stored in the force acquisition algorithm and distributed to other sub-algorithms as needed.

Figure 3.6 Process diagram for the acquire data algorithm. The diagram includes the primary inputs and sequential steps that the algorithm executes to load data previously acquired.

3.5.6 Unit Testing

Unit testing was a critical aspect of the algorithm development process. By unit testing each sub-algorithm developed within LabVIEW, it ensured that the execution of the force acquisition algorithm could occur without failures from the sub-algorithms. Unit Testing for the LabVIEW VIs occurred by assessing each algorithm on its own before testing to ensure that the main force acquisition algorithm was able to command the sub-algorithms. The following bullet

points summarize the process used to unit test each algorithm and include graphic examples when applicable.

- **Acquire Data Algorithm**: The acquire data algorithm was exclusively assessed during live trial testing. Validation of this procedure was defined as a successful update of the force acquisition algorithm with data following the execution of the acquire data process.
- **Process Data Algorithm:** The process data algorithm was assessed using previously acquired data. The testing process provided validation that the algorithm was properly connecting to Python and saving data in the specified folder. An example (Figure 3.7) of the output of this algorithm is shown below.

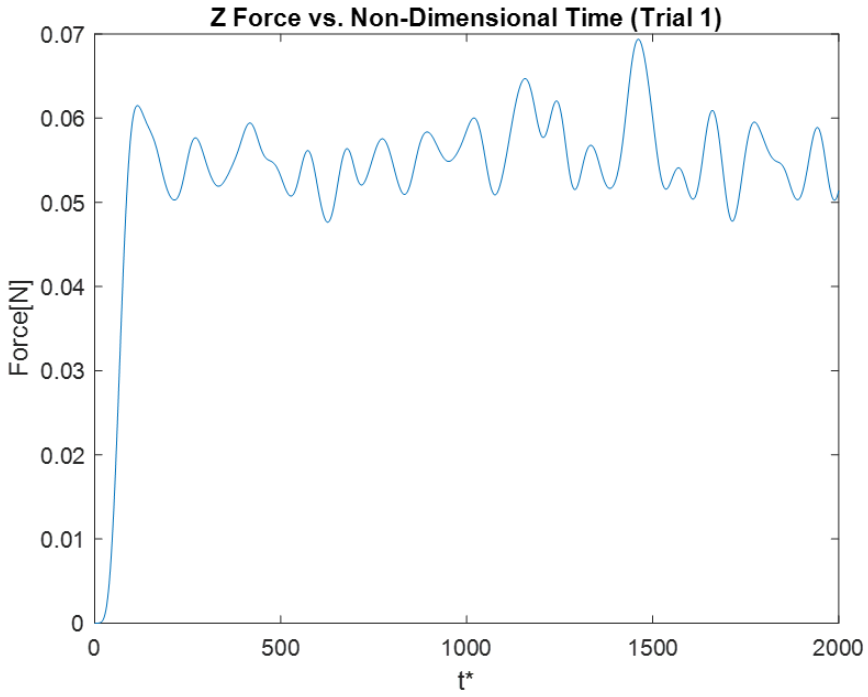


Figure 3.7 Example of Output from Process Data Algorithm. Image is an example of the Python output that is saved in the user defined location.
NOTE: The data presented above is from an uncalibrated force transducer.

- **Save Data Algorithm:** The save data algorithm was evaluated using previously acquired data and data acquired during live testing. Data loaded into LabVIEW was resaved in the original format to verify that the save method preserved the original data. For data acquired during live testing, the data was saved in all available formats to verify that LabVIEW was saving the files in the specified location. The formats were checked using a tool developed within LabVIEW to verify that the save data algorithm preserved the data.
- **Load Data Algorithm:** The load data procedure was evaluated using previously acquired force data. For full compatibility testing, data stored in different formats were loaded to verify that LabVIEW was reading and storing the files properly.

3.6 Python

Python is a high-level general purpose programming language, first released in 1991. Due to Python's ability to automatically compile code to byte code and execute it, the software is suitable for use as a scripting language (Dave Kuhlman, 2012). Additionally, Python can manage extensive tasks due to the ability to extend Python to C and C++ (Dave Kuhlman, 2012). This capability alone made Python an ideal choice for developing code to execute simultaneous acquisition which can be computationally intensive.

3.6.1 Software Build Selection

Python version selection was crucial to project development due to limitations identified previously. As mentioned in Section 3.3.2, the Python version used for code development was dependent on compatibility with LabVIEW and Vicon Tracker software. LabVIEW is compatible with Python versions 2.7 and 3.6-3.10. However, that compatibility is not uniform

across all LabVIEW releases that support Python. Specifically, Python 2.7 is only compatible with LabVIEW releases from 2018 – 2019 whereas Python 3.6 is compatible with all releases of LabVIEW that support Python. Compatibility between the two software tools decreases as newer versions of each are released. Regarding this project, LabVIEW version 2022 Q3 was used to develop acquisition code for the force transducer. As a result, the Python version was initially constrained to Python 3.6. Vicon Tracker 3.9 was selected as the software used to acquire motion data and it is compatible with Python 2.7 and all releases of Python 3; per developer guidance it is recommended to use releases of Python 3. Given the requirements and recommendations set forth by each software developer Python the Python 3.6. release was selected for algorithm development. While Vicon Tracker is not constrained to specific releases of Python, the LabVIEW software build dictates the use of Python 3.6 since the base version is useable. Additionally, release 3.6 is supported across all LabVIEW versions with Python support, making it the ideal choice for the project.

3.6.2 Python Module / Library Selection Process

Python library selection was critical in the algorithm development process. Specifically, this section will discuss the Python libraries that were considered during the algorithm development process. The libraries that were evaluated for the project were the Threading module, Process-based Parallelism library (multiprocessing), and Asynchronous I/O library (asyncio).

Threading is a module within the base Python library that allows users to create multiple threads in order to execute blocks of code or functions concurrently. A key advantage of using this module is that it is not constrained to using specific Python objects meaning that it can handle simple Python objects like list variables or more complex Python objects such as Python

objects spawned from a Component Object Model (COM). A disadvantage of this module is that its ability is restricted by that of the Global Interpreter Lock (GIL) which forces only one thread to run at a time. In order to implement this module to achieve the task of executing multiple threads simultaneously within Python, the use of a different coding language is required to overcome GIL restrictions.

Process-based Parallelism (referred to as multiprocessing) is a library that spawns processes using an API similar to the threading module. A key advantage of this library over the threading module is that it circumvents the Global Interpreter Lock (GIL) allowing for multiple functions to be executed at the same time. The Process class within multiprocessing was evaluated to assess whether it could support executing force and motion acquisition at the same time. However, the use of this library and class presented limitations. Specifically, the LabVIEW objects defined initially were incompatible with the process class. In order to execute functions, the Process class has to 'pickle' (the process of converting a Python object into a byte stream) the Python object before the function can run to completion. Due to the size of the LabVIEW objects, Python was not able to 'pickle' the object posing a significant obstacle with integrating this library into the existing code architecture.

The asynchronous I/O library was designed to allow for the writing of concurrent code using the async/await syntax. This library is used by Python as the basis for asynchronous frameworks that provide high-performance network and webservers, database connection libraries, and distributed tasks. This library was chosen for evaluation because of its compatibility with I/O bound network code. Similar to the multiprocessing library, the asynchronous I/O library is also limited in terms of what data formats are compatible with the library's functionality.

Given the advantages and disadvantages of the module and libraries presented, the asynchronous I/O library was chosen for integrating the two systems. The threading module is capable of working with the existing systems. However, because acquisition tasks are computationally intensive, the threads created with the threading module would not execute simultaneously. The multiprocessing library and asynchronous I/O library would execute in an equivalent manner and achieve the desired results. However, due to the asynchronous I/O library's compatibility with I/O bound operations, it was selected as the best option for integration with the developed algorithms and communicating with the external hardware. Additionally, the Python objects limitation presented by using the asynchronous I/O library can be overcome by replacing complex Python objects with simple commands. The use of the asynchronous I/O does impose a Python build change from 3.6 to 3.7 since the library was introduced in version 3.7. This software build change does not affect integration with LabVIEW.

3.7 Python Algorithm Development

Several algorithms were developed during the project to support the developed method. The algorithms that will be discussed in this section are the control algorithm, process data algorithm, and motion capture algorithm.

3.7.1 Control Algorithm

The control algorithm was developed to automate the force acquisition process from LabVIEW. The algorithm has one main process which has the following inputs:

- Save folder: folder path that processed data will be saved in (String Input).

- Load data format: data type associated with the data file that will be loaded as (*i.e.,* csv, bin, tdms, etc.).
- Save data format: data type associated with the data file that will be saved as (*i.e.*, csv, bin, tdms, etc.).

The process works by first initializing the connection from Python to LabVIEW using the win32com library in Python. Then the inputs are defined and stored in a structure allowing for Python to use them in the force acquisition algorithm developed for LabVIEW. Next, the acquire data sub-algorithm is defined and called which then executes and stores off acquired data. Following this, acquired data is processed using the post process data sub-algorithm – generated figures and raw data are stored in the user specified save folder. Lastly, once all processes have concluded their execution, the connection between Python and LabVIEW is terminated.

3.7.2 Post Process Data Algorithm

The post process data algorithm was developed to automate the method used to process acquired data. The algorithm has one main function, process force data, which has the following inputs:

- X force: array of acquired force data with respect to the x component of the three-dimensional force vector.
- Y force: array of acquired force data with respect to the y component of the three-dimensional force vector.
- Z force: array of acquired force data with respect to the z component of the three-dimensional force vector.

- X motion: array of acquired motion data with respect to the x component of the three-dimensional position vector.
- Y motion: array of acquired motion data with respect to the y component of the three-dimensional position vector.
- Z motion: array of acquired motion data with respect to the z component of the three-dimensional position vector.
- File path: full path to the location where the figures are saved off.

The function works by filtering the acquired data using a low pass Butterworth filter with a cutoff frequency of 10 Hz and a sampling frequency of 1000 Hz. Next, the filtered x, y, and z force data are plotted and saved off in user defined folder. Following this, the x, y, and z motion data are plotted and saved off. Lastly, a three-dimensional plot of motion data is plotted against time and saved off.

3.7.3 Motion Capture Algorithm

The motion capture algorithm was developed to automate the process of acquiring motion data from the Vicon motion capture system through Vicon Tracker 3.9. The module contains one main function, tracker data acquisition, which has the following inputs:

- Save path: file path where the trial data is saved (.csv).
- Trial name: name of trial for acquired data.
- Export data: flag set by the user to determine if the script will export the captured motion data to a csv file (the default for the system is false).
- Acquisition time: time (in seconds) that the system will capture data (the default acquisition time is 30 seconds).

- Host: Tracker host IP address (the default IP address is ‘localhost’)

The process works by connecting Python to Vicon Tracker using the Tracker API Python library. The script then sets the trial name in Vicon Tracker 3.9 using the user defined trial name and the capture path function in the Tracker API. Next, the start and stop capture functions are used along with the defined acquisition time to acquire motion data from the Vicon camera system. Once acquisition is complete, Python disconnects from the Vicon system using the Tracker API disconnect function.

3.7.4 Unit Testing

The process of unit testing the Python based algorithms by testing critical blocks of the code that are critical for the execution of the overall module. For the control algorithm, first the connection to LabVIEW was tested and verified. Next the connection to the sub-algorithms developed in LabVIEW was tested to verify that Python could send inputs and trigger successful execution of these algorithms. For the post process data algorithm, existing force data was used to verify that the algorithm was appropriately applying the designated filter to the raw data, generating the necessary figures, and saving the generated figures in the appropriate folder location. Lastly, successful execution of the motion capture algorithm was verified using the following method:

- Testing the connection between Python and Vicon Tracker 3.9 software.
- Visual confirmation that the capture name is set appropriately in Vicon Tracker by checking the trial name field in the Tracker software.
- Visual confirmation that Python triggered the acquisition process by inspecting the status of the ‘Recording’ field in the Vicon Tracker software.

3.8 Implementation

Implementation of both systems primarily focused on the efforts undertaken to develop a robust Python algorithm capable of executing acquisition without failure. This section discusses the system integration undertaken during the project's life cycle (*i.e.,* integrating the code with the physical hardware) and the testing done to verify that the code could be executed without a software failure.

3.8.1 System Integration

While the development of the standalone algorithms for force and motion acquisition were necessary for achieving the objective of this project, the critical aspect of the algorithm development process was integrating the force and motion acquisition algorithms to achieve the simultaneous measurements. Successful integration of these algorithms was required to satisfy the objective. This section primarily discusses the primary challenge encountered with integrating the two systems and the development process used to create the simultaneous measurement algorithm.

The integration of the two systems was not as simple as merging the acquired force data and acquire motion data algorithms. Each algorithm was developed specifically to automate the acquisition process from their respective system and in turn contain capabilities that pose challenges when integrating the two systems.

The primary challenge posed during the integration process was related to the Python library used to execute the simultaneous measurements. As mentioned in Section 3.5.2, Python library selection was critical for algorithm development in regard to acquiring simultaneous measurements. The library selected for algorithm integration was the asynchronous I/O library

due to its ability to work with I/O bound processes. However, the use of this library posed challenges when integrating the acquire force data and acquire motion data algorithms. Specifically, integrating the LabVIEW components, which are the main components of the acquire force data algorithm, with the asynchronous I/O library is not possible. Due to the size of the components, Python is not able to 'pickle' (the process of converting a Python object into a byte stream) for use with the asynchronous I/O library resulting in a failure when conducting trial testing. As a result, the process used for the acquire force data algorithm had to be reevaluated. Overcoming this issue required that the process used for acquisition be modified to remove the LabVIEW components that were assigned specifically with the acquisition task and preserve the primary functionality. The solution created to overcome the challenge was to remove the LabVIEW based subprocess responsible for reading measurements from the DAQ to a Python based force acquisition algorithm. By using Python to execute the reading of measurements from the DAQ, the issue of being able to 'pickle' is overcome by the fact that Python can 'pickle' the acquisition function in the algorithm . Altering the algorithm using this method was achievable because like LabVIEW, Python also contains the DAQmx library (a library developed by National Instruments to allow for integration of Python and NI DAQ devices) which allows for Python to directly read measurements from NI DAQs. However, this alteration to the algorithm's primary functionality required that the LabVIEW algorithm also be modified to accept the acquired data obtained using Python as an input. Further explanation on the new algorithm structure is discussed subsequently.

The structure of the algorithm used to trigger the simultaneous acquisition from both systems was a matter of ensuring that the developed method met the objective of the project. While the acquire force data and acquire motion data algorithms can be integrated into a singular

algorithm, it was decided that having a main algorithm that triggers sub-algorithms would allow for a clean separation of defined acquisition tasks. Additionally, a failure of one algorithm does not impact the execution of the other. The main algorithm is the simultaneous data acquisition algorithm and the sub-algorithms called by the main algorithm are the acquire force data, acquire motion data, convert force data, process data, and save data.

The primary purpose of the developed simultaneous data acquisition algorithm is to control the sub-algorithms and execute the simultaneous measurements task. Specifically, the algorithm uses the asynchronous I/O library in Python to execute the acquire force data and acquire motion data algorithms simultaneously. Following completion of acquisition from both systems, the data acquired from the DAQ is converted for force and torque data using the convert force data algorithm. The simultaneous data acquisition algorithm then uses the acquired force data and defined user inputs (*i.e.*, save folder, trial name, etc.) to execute the post process data algorithm and save data algorithm, respectively.

The acquire data algorithm used for the system integration uses the process discussed in Section 3.4.2. However, modifications were made to accommodate integration with Python. While the logic used has remained the same, the acquire data algorithm used for system integration does not contain any LabVIEW components and is primarily dependent on the DAQmx Python library. This modification improved the efficiency of the force acquisition process by eliminating the need to connect to external programs.

The acquire motion data algorithm contains the logic required to acquire motion data from the Vicon camera system. This algorithm contains the necessary steps required to initialize the connection between Python and the Vicon Tracker software. It also contains the logic that starts and stops the capturing of motion data from the Vicon system.

The convert force data sub-algorithm was developed for system integration. Since the acquire data algorithm reads measurements directly from the DAQ outside of the LabVIEW environment, the raw voltage data acquired had to be converted to force and torque data. The algorithm uses the acquired data in addition to a conversion matrix and the calibration matrix provided with the ATI force transducer to convert the voltages to force and torque. The converted data is then transferred to the simultaneous measurements algorithm and used in the process data and save data sub-algorithms.

The remaining two sub-algorithms, process data and save data, are called, and executed by the simultaneous data acquisition algorithms were discussed prior in Sections 3.4.3 and 3.4.4, respectively. There were no modifications made to these algorithms for the integration process.

3.8.2 Integration Testing

Integration testing occurred using the Python modules listed above, physical system architecture, and a test specimen. This section details the process of the integration testing. The testing process occurred as shown below:

- Obtain and store bias data from the force transducer – bias data are measurements from the force transducer while the test specimen is at rest.
- Define required inputs:
    - Number of samples to acquire from force transducer.
    - Time in seconds to capture data from the Vicon system.
    - Name of the trial.
    - Location to store acquire data and figures generate during the post processing.
- Power up test specimen.

- Execute main function in the simultaneous data acquisition algorithm.
- Note acquisition time from each system and determine the time delta between the systems.
- Power down test specimen.

This testing process was repeated multiple times with varying sample sizes, acquisition times, and sampling frequencies. Results from the testing will not be presented in this section, but instead in Chapter 4.

Chapter 4. Experimental Results

This chapter discusses the experimental testing results observed while conducting trials to validate the simultaneous data acquisition algorithm and the sub-algorithms developed to accompany it. Section 4.1 primarily focuses on validating the algorithm execution and assessing whether significant timing phase offsets were observed between the acquisition times of both systems. Additionally, hypotheses are presented to explain why a timing phase offset in the acquisition start times might be observed using the tested system. Additionally, Section 4.2 discusses alternative system designs that could be implemented to achieve simultaneous data acquisition from two external systems.

4.1 Testing and Validation

This section presents the results from trial testing in the form of processed force and motion data as well as timing results. The trial testing procedure is described in Section 4.1.1. It also discusses observations made from experimental testing and presents hypotheses as to why these observations may have occurred.

4.1.1 Experimental Testing

The section primarily focuses on the experimental setup used to conduct testing and the parameters used to constrain data collection. The experimental setup (Figure 4.1) used to capture the force and motion data consisted of an ATI Nano 17 Titanium force transducer and VICON T40s motion capture cameras. A gearbox was mounted to the ATI Nano Force Transducer and a

single wing was mounted to the top of the gear box. The test stand was surrounded by 11 VICON T40s cameras and black material was used to cover the reflective surfaces that interfered with the motion capture system's ability to capture the wing's motion.

Three small reflective markers were placed on the wing (Figure 4.2) and the three-dimensional positions of the markers were acquired at a sampling rate of 300 Hz using the VICON system for a duration of 10 seconds. The force generated by the wing motion was acquired by the force transducer at a sampling of 1000 Hz for 10 seconds. For reference, measurements from the force transducer were not accurate due to the force transducer malfunctioning during testing. However, the accuracy of the force transducer measurements does not impact satisfying the objective of the project since the primary objective is to acquire data simultaneously. The primary purpose of the force transducer as it relates to this project is to read measurements which it is still capable of doing.

Prior to each trial, the load due to the wing's weight at rest was recorded and used by the force algorithm to calculate the force data. The simultaneous acquisition algorithm was used to trigger the simultaneous acquisition of the motion and force data.

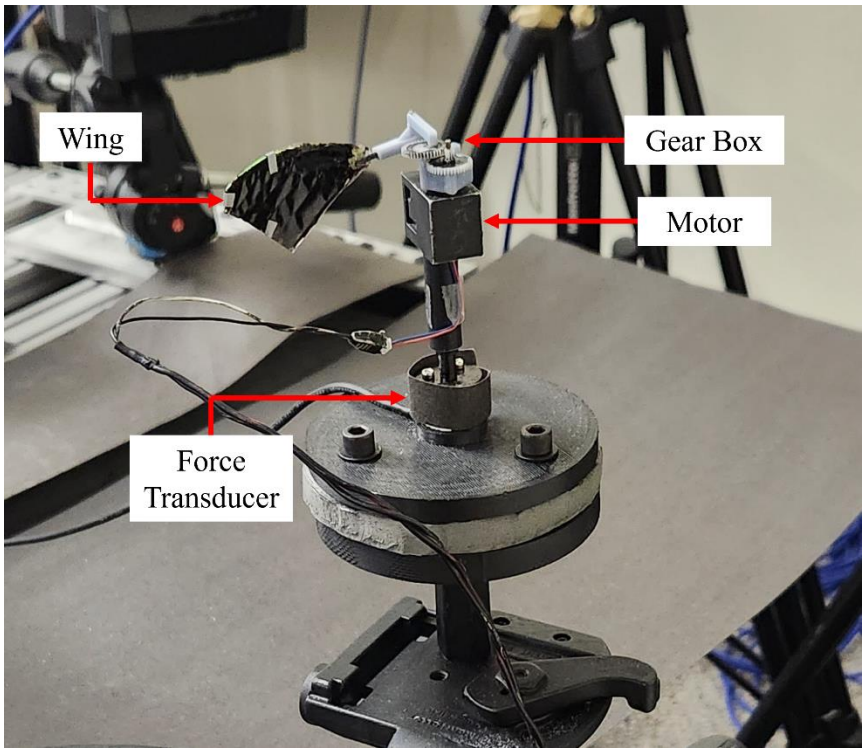


Figure 4.1 An ATI Nano 17 force transducer mounted on a custom 3D printed base, with a foam layer inserted between the upper and lower halves of the mount. The gearbox is attached at the top of a 3D printed mounting pedestal containing the motor used to generate wing motion.

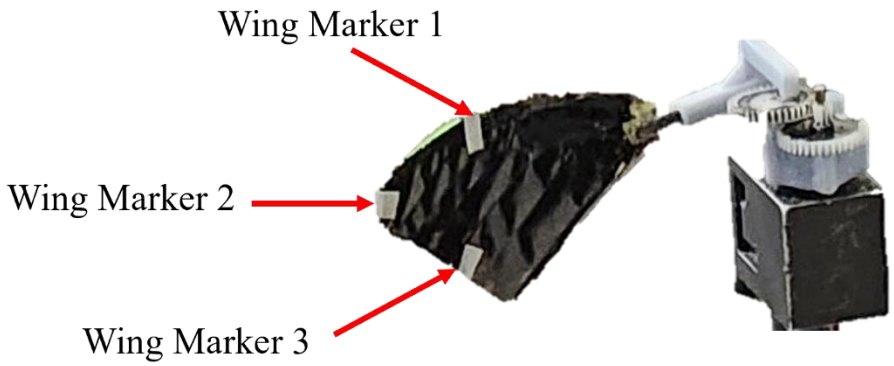
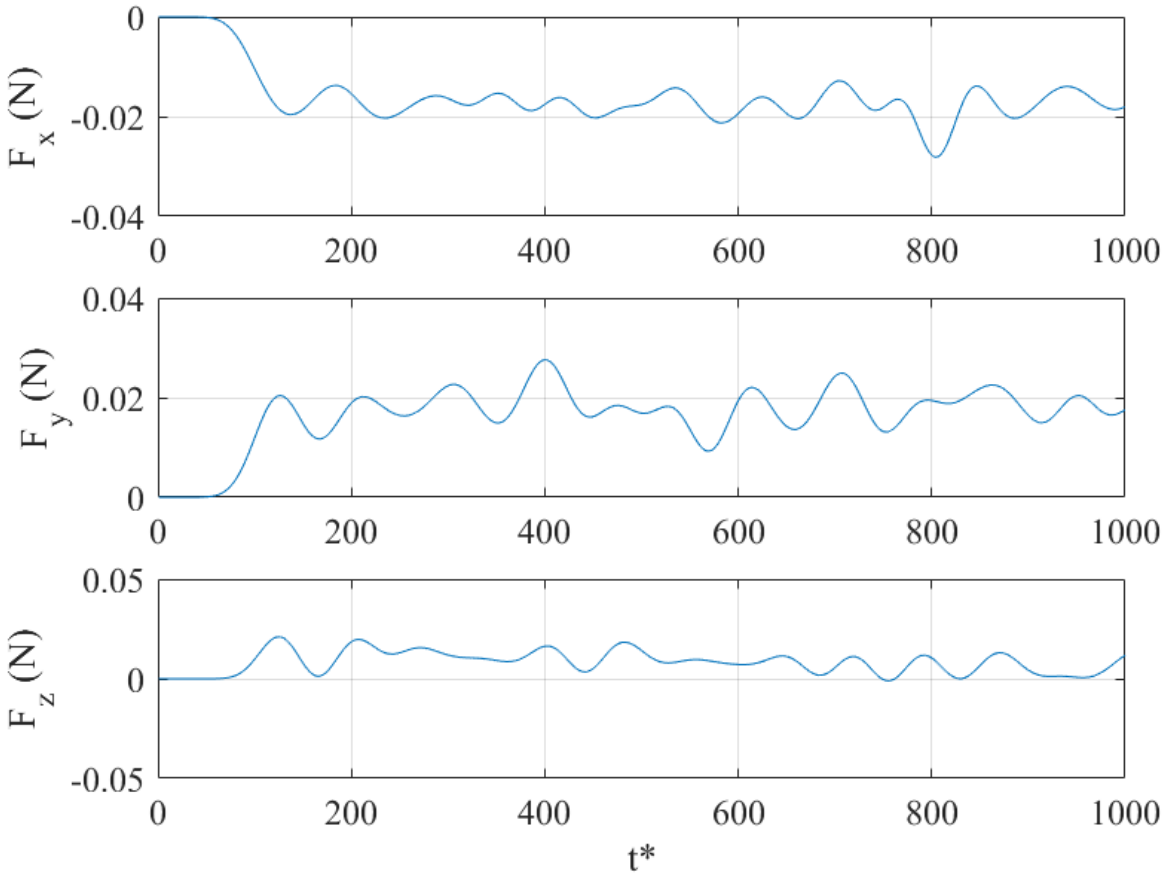


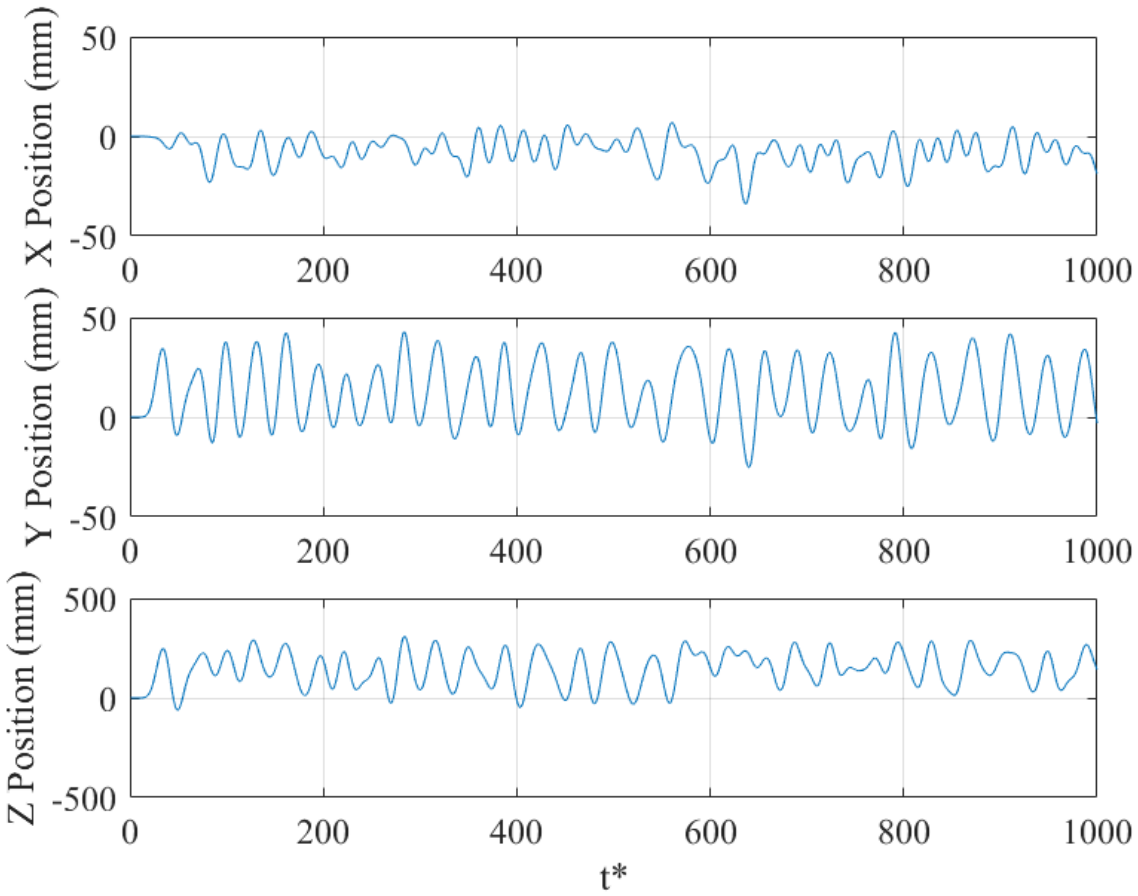
Figure 4.2 Flapper gearbox and wing mounted to the left side. The three wing markers form a triangle to calculate the Euler angles.

4.1.2 Experimental Results

An example of data collected during the trial testing serves as validation that the developed code can acquire force data, acquire motion data, and process acquired data. Figure 4.3 shows force and motion data processed using a 12th order lowpass Butterworth filter with a cutoff frequency of 0.03. Additionally, unprocessed position data for each directional component was plotted on a three-dimensional figure to show the full motion of the wing (Figure 4.4).



(a.)



(b.)

Figure 4.3 Force and motion measured with the experimental flapping wing. (a) Force measurements observed during trial testing with respect to the x-axis, y-axis, and z-axis. (b) Motion deformation observed during trial testing with respect to the x-axis, y-axis, and z-axis.

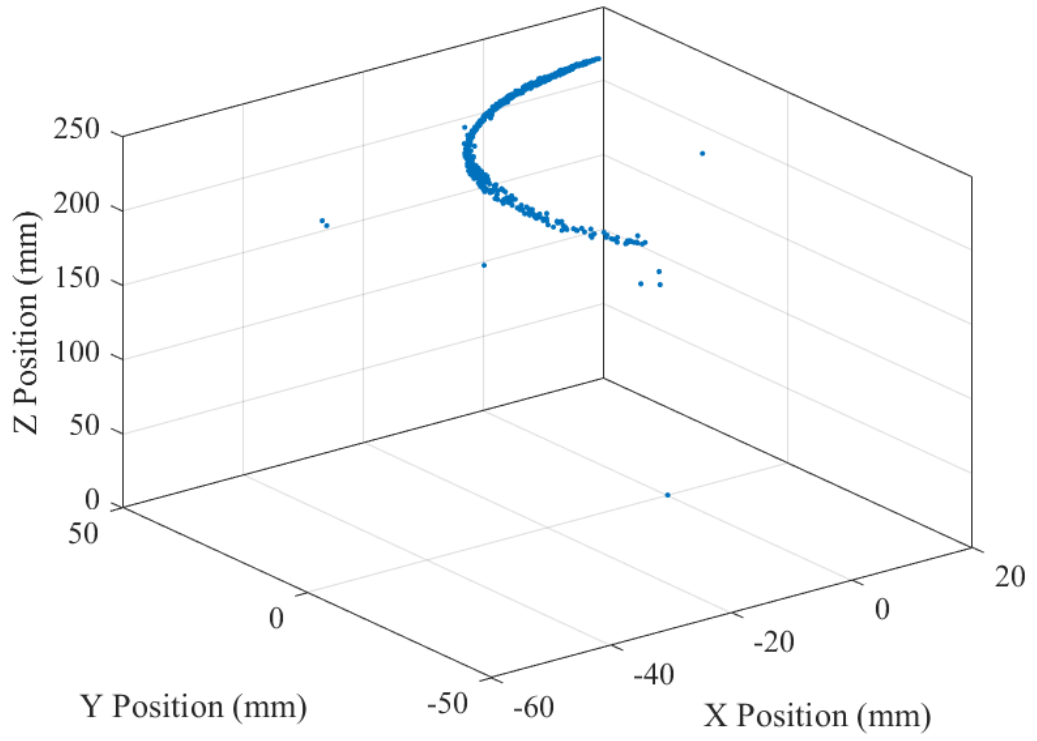


Figure 4.4 Un-filtered three-dimensional deformation data captured during trial testing for the experimental wing. The data represents the motion of the wing during testing.

4.1.3 Timing Results

In order to assess whether the objective of the project was met, an examination of timing results was key. In this case, timing results indicate the timing phase offset (Δ) between the ATI Nano 17 force transducer and the Vicon motion capture system. Timing results from several trials are shown in Table 4.1. Table 4.1 shows the LabVIEW timestamp recorded by Python, and it also denotes the time acquisition started from Vicon Tracker. Lastly, the Δ between the two systems is shown – here Δ is the difference between the LabVIEW start time and the Tracker start time in Central Standard Time. For each trial, 10 seconds of data was acquired from the

force transducer with a sampling rate of 1000 Hz and 10 seconds were acquired from the motion capture camera system with a sampling rate of 300 Hz.

Table 4.1 Timing Results for 10 Experimental Trials conducted. Force data was recorded with a sampling rate of 1000 Hz for 10 seconds and motion data was sampled at 300 Hz for a duration of 10 seconds. Each timestamp represents the time denoted by the computer in which the system started acquisition execution. The Δ is the difference between the timestamp of each system.

Trial	LabVIEW Timestamp	Tracker Timestamp	Δ (seconds)
1	16:16:39.835	16:16:49.893	10.058
2	17:44:19.54	17:44:29.613	10.073
3	17:46:27.787	17:56:37.905	10.118
4	17:48:16.030	17:48:26.096	10.066
5	17:49:24.692	17:49:34.744	10.052
6	17:50:20.80	17:59:30.886	10.086
7	18:03:04.246	18:03:14.302	10.056
8	18:05:17.388	18:05:27.505	10.117
9	18:08:51.452	18:09:01.541	10.089
10	18:10:03.028	18:10:13.232	10.204
		Average Time Δ:	10.0919

As shown in Table 4.1, there was an average time Δ of 10 seconds between the start of the force acquisition and motion acquisition. For each trial, force acquisition would be executed prior to motion acquisition starting. Acquisition for one system would not start until the other system had completed. Other trial configurations were tested in which the number of samples collected, and acquisition times were varied. These results suggest that the timing delta depended heavily on the chosen sample size and acquisition time. For force acquisition, a sample size of 10000 at a sampling rate of 1000 Hz is 10 s of data. Given that the motion acquisition started on average 10 s after the force acquisition began, it can be inferred that the start time of the motion acquisition is dependent on the force acquisition sample size set by the user. The order of

acquisition (*i.e.*, executing acquisition from LabVIEW or Vicon first) also affects the observed start time Δ between the two systems.

Table 4.2 shows an additional example of the timing Δ behavior observed during trial testing. Table 4.2 shows the LabVIEW and Vicon start times were recorded for five trials in which 3000 samples were collected from the force transducer at 1000 Hz and 30 seconds of motion data sampled at 300 Hz was captured from the Vicon camera system. For these trials, LabVIEW was configured to execute first during these trials. An average timing delta of three seconds was observed across the five trials.

Table 4.2 Timing Results for 5 Experimental Trials conducted. Force data was recorded with a sampling rate of 1000 Hz for 3 seconds and motion data was sampled at 300 Hz for a duration of 30 seconds. Each timestamp represents the time denoted by the computer in which the system started acquisition execution. The Δ is the difference between the timestamp of each system.

Trial	LabVIEW Timestamp	Tracker Timestamp	Δ (seconds)
1	17:35:36.004	17:35:39.041	3.037
2	17:36:58.816	17:37:01.853	3.037
3	17:38:25.688	17:38:28.728	3.04
4	17:40:00.349	17:40:3.432	3.083
5	17:41:39.839	17:41:42.881	3.042
		Average Time Δ:	3.0478

While the timing phase offset seems insignificant in these trials, it actually results in neither system having any synchronized data. The three second phase offset originates from the system only requiring approximately three seconds to collect the samples from the force transducer. So, the three seconds of force data cannot be correlated to the first three seconds of the motion capture data. As stated previously, the delta is also dependent on the order of acquisition from the two systems. Since the force acquisition took place first and the sample size

was small, the Δ was also small. If the order of acquisition was switched for these trials to have Vicon acquisition occur first, then the delta would have been much larger – approximately 30 seconds. Given the timing results presented, simultaneous measurements were not achieved with the system architecture.

4.2 Discussion of Results

The discussion of the results mostly focuses on hypothesizing why simultaneous data acquisition from both systems was not achieved. Two hypotheses are discussed in this section. The first hypothesis focuses on the potential limitations of the computer and its ability to execute certain tasks at the same time. The next hypothesis relates to the physical connections between the two external systems and the computer; it explores how these connections may have impacted the ability to start both systems at the same time.

4.2.1 Hypothesis 1: Limitations of the Computer Affect Its Ability to Execute Certain Tasks Concurrently

This hypothesis discusses a potential limitation of the computer that may affect its ability to schedule concurrent tasks. A common misconception about most traditional computers is that they are executing multiple tasks at the same time. However, this is not the case. A computer can appear to be executing multiple tasks concurrently, but in actuality, it is rapidly changing tasks or interrupting tasks already in execution. There are situations in which a computer can execute multiple processes simultaneously and this is called parallel computing. Parallel computing is capable in multi-core (depending on the applications used) or multi-processor computers and systems of computers assigned with working on the same task. If a computer is not capable of parallel computing, it can only execute one task at a time. CPUs use a short-term scheduler to

manage the given tasks and determine which tasks in the queue to handle given the current state. The CPU scheduler makes its decision based on changing states (*i.e.,* a process was interrupted, running, or completed). Given the state at that moment in time, the schedule decides whether to execute a new process or continue with the existing one. In the case of this project, the CPU was tasked with managing I/O bound operations. These operations can result in processes that are idling and cannot be interrupted, meaning that the CPU schedule must wait for the process to terminate before attempting to execute a new process. So, a potential reason for why the simultaneous acquisition could not occur is due to the I/O bound tasks from both external systems. Since acquisition from both systems cannot be interrupted and the computer can only handle one process at a time, whichever system was not queued for acquisition first would have to wait for the computer to finish processing the task from the first system.

Assessing this hypothesis was primarily conducted by executing acquisition independently from the developer provided interfaces of both systems. Each system was started independently of the other with an understanding that there would be a phase offset due to human interaction with the system – in this case the phase offset was the time between the start of the first system and the start of the second system. Three trials were conducted to assess this hypothesis. A description of the trials and the observed outcomes are listed below.

Trial 1: Force acquisition started prior to motion acquisition. The outcome of this trial was that motion acquisition did not start until force acquisition was completed. This was verified by visually inspecting the Vicon Tracker interface and denoting the start times.

Trial 2: Motion acquisition started prior to force acquisition. The outcome of this trial was that force acquisition did not start until force acquisition was completed. This was verified by visually inspecting the Vicon Tracker interface and denoting the start times.

Trial 3: Motion acquisition started prior to force acquisition. The outcome of this trial was that the motion acquisition process was interrupted by the computer scheduling and executing the force acquisition resulting in incomplete motion data.

4.2.2 Hypothesis 2: Physical Connections Have an Impact on the Computer's Ability to Start Both Systems Concurrently

This hypothesis discusses the possibility that the physical connections between the two external systems and the computer affect the ability to receive data simultaneously. The force transducer was connected to the computer via a USB connection and was managed by the universal serial bus communication protocol. The Vicon camera system was connected to the computer using an ethernet connection. Despite the two different physical connections, during trial testing, the two systems were trying to transfer data to the computer using a serial connection. This was considered as a potential issue in regard to acquiring data simultaneously because data transferred over serial connections can only be executed one at a time

The assessment of this hypothesis was primarily done through visual inspection of the ports assigned to each system by the computer. First, the device manager was used to examine the physical ports that the systems were connected to. The purpose of this first check was to verify that the two systems were not connected to ports of the same connection type (i.e., both systems connected via USB, ethernet, etc.). Connections made with the same connection type can limit the ability of a computer to transmit data simultaneously over the same channel. The next check was to use the device monitor and verify that the two systems were not using the same port for the TCP connections. This check was verified by examining the port connections while both systems were acquiring data. This check was critical because data cannot be transmitted simultaneously through the same TCP port.

Both systems were subjected to the defined checks, and it was determined through examination of the assigned ports that the computer was assigning unique ports to each system. This hypothesis does not hold as much weight as the hypothesis presented in Section 4.2.1 because it was verified using the Windows device manager and resource monitor that the two systems were assigned unique port identifiers by the operating system and were not transmitting data using the same TCP ports.

4.3 Solutions and Alternative System Designs

Large timing phase offsets were observed during the experimental testing phase of this project, meaning that the current system design is not capable of acquiring simultaneous measurements from multiple devices. This section presents alternative system designs that could potentially be implemented in order to achieve this goal. The two alternative designs are integrating a peripheral device with the camera system and adding a secondary computer.

4.3.1 Adding a Peripheral Device to the Existing Camera System

An alternative system design that would be capable of supporting simultaneous measurements from a motion capture camera system and another external device would be to integrate the external device with the camera system. Using this system design, the camera system would function as the timing trigger for data acquisition from both devices. Once motion capture acquisition is started, the camera system would also simultaneously stream data from the other device and transfer that data in a packet with the motion data. The data needs to be parsed into its respective categories, which would require an understanding of which channels correspond to which systems.

In relation to this specific project, Vicon has a device called the Vicon Lock Sync Box

which would allow for third-party devices to be connected to the Vicon camera system. The device allows for eight peripheral devices to be integrated with an existing Vicon system using the General-Purpose Output (GPO) sync output sockets located on the back of the device. The integration of the external device and the camera system does require that the external device possesses a GENLOCK / trigger input socket (a connector that would allow for the output of one source to be used as to synchronize other sources together) and an RCA (an electrical connector used to carry audio and video signals) to RCA sync cable is used. Since the NI USB 6210 DAQ does not possess the necessary socket to connect to the Vicon Lock Sync Box, this design would not work with the force sensor used for this project. However, this design would work with sensors manufactured by companies such as AMTI Force & Motion and ProtoKinetics. Additionally, were this design to be used, modifications to the software would be required. Specifically, the Tracker software settings would have to be adjusted to include the analog device connected to the Lock Sync Box and the acquire force data algorithm would have to be modified to work with the analog device. Figure 4.5 shows a diagram of this proposed alternative system design. Note that with this system design, LabVIEW is not required because measurements from the force transducer are read and transmitted through the Vicon Lockbox. This system design would require significant algorithm redevelopment for the acquired force data because the LabVIEW based algorithm was responsible for converting the voltage data to force and torque data. Since this system design would not require the addition of the LabVIEW software, another algorithm would have to be developed using Python to convert the raw measurements from the DAQ to force and torque data.

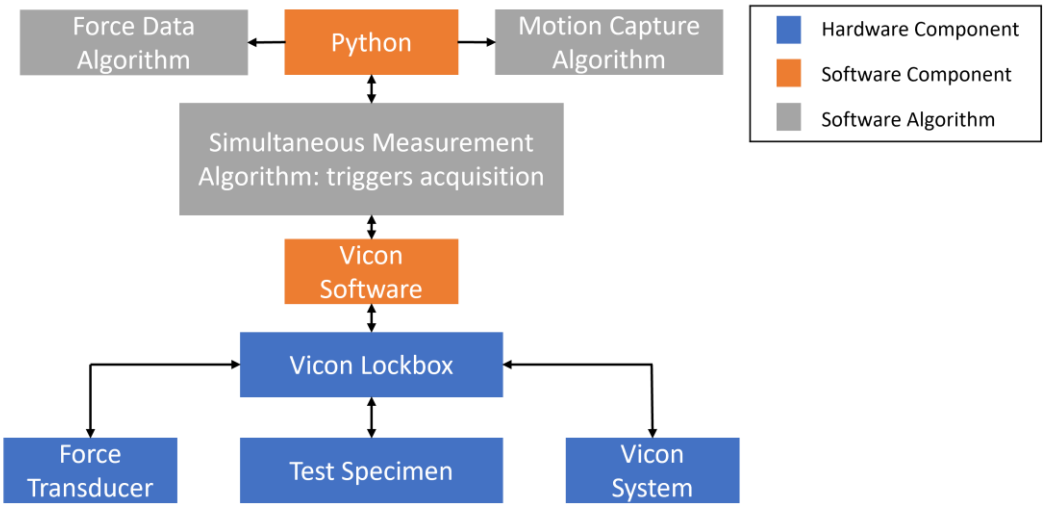


Figure 4.5 Alternative System Design that includes the addition of the Vicon Lockbox. Instead of the force transducer and Vicon system being directly connected to the computer, the systems are instead connected to the Vicon Lockbox. The Vicon Lockbox would record data from both systems using its internal clock as the triggering mechanism.

4.3.2 Adding a Secondary Computer to the Existing Architecture

Another alternative system design that would be capable of supporting simultaneous measurements from a motion capture camera system and another external device would be to integrate another computer into the existing system architecture. Using this system design, the other external device would be connected to the secondary computer which would be tasked with executing the functions associated with acquiring data from the external device. In order for successful execution to occur with this system design, the primary computer would have to connect to the secondary computer and send the task to the secondary computer to start acquisition. Additionally, the main Python algorithm, simultaneous data acquisition, would have to account for a potential time delay in connecting the two systems which can be done using the 'await' function in the asynchronous I/O library.

Regarding the existing system architecture, the ATI Nano 17 force transducer would be connected to the secondary computer and the acquire force data algorithm would be executed from this computer. An additional Python algorithm would have to be developed to serve as an intermediate control between the simultaneous data acquisition algorithm and the acquire force data algorithm. This algorithm would have to initialize the connection between the primary (the computer controlling the Vicon camera system) and secondary computers and command the secondary computer to start acquiring data from the force transducer. Results gathered during the acquisition process would then be transferred to the primary computer for post-processing.

An attempt to integrate this idea into the existing architecture was made during the project's duration using a Raspberry PI computer. However, this attempt was not successful due to limitations identified in the Methodology section. The Raspberry Pi does not support the Windows operating system which is a requirement for using the NI DAQ. Successful execution with this modified system design and the existing equipment requires that another computer with the Windows operating system is integrated into the architecture.

Figure 4.6 shows a diagram of the proposed alternative system design that incorporates a second computer. Incorporating a second computer into the architecture would allow for parallel computing and potentially satisfy the simultaneous measurements objective. However, prior to executing the algorithms, the connection between the computers would have to be verified as this design depends on the primary computer being able to command the secondary computer to start acquiring data.

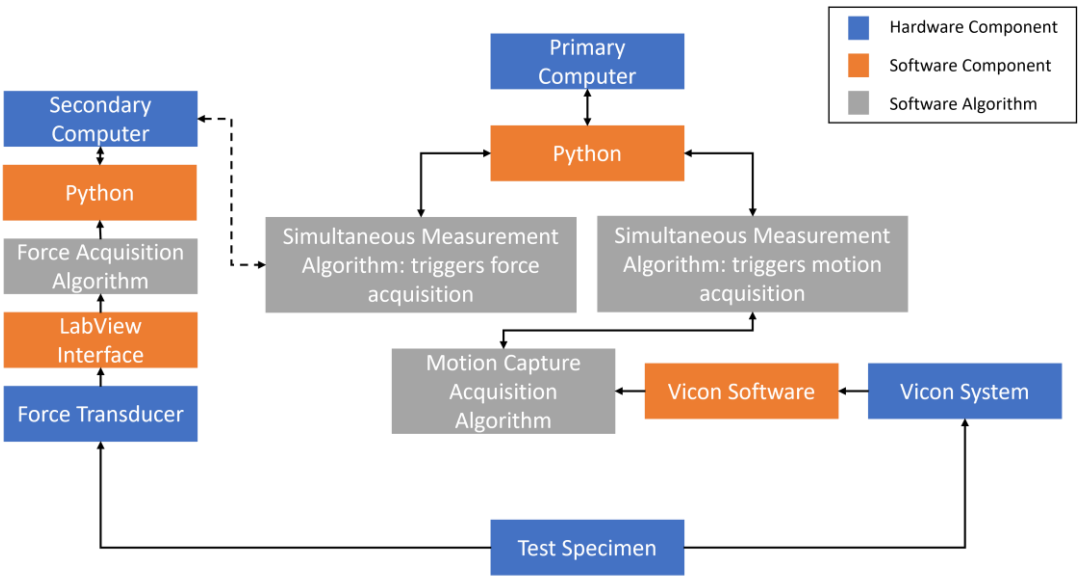


Figure 4.6 Alternative System Design that includes the addition of an additional computer. The force transducer is connected to the secondary computer in this architecture to reduce resource strain on the primary computer. A network connection allows the two computers to communicate with each other.

The addition of the second computer may not be able to precisely synchronize the data because each computer would trigger acquisition based on their respective internal clocks. If these two clocks are off by seconds, it could have a significant impact on the phase offset observed resulting in a misinterpretation of the data. Additionally, since a network connection is required for the primary computer to send the start acquisition command to the second computer, a slight delay in network traffic will result in a phase offset and in turn a misrepresentation of the acquired data.

Chapter 5. Conclusion and Future Work

5.1 Summary and Concluding Remarks

The objective of this project was to develop a method to synchronize data acquisition from two external systems – specifically, acquisition of force and motion data synchronously from a force transducer and motion capture camera system, respectively. The project was constrained to several requirements ensuring that the developed product would satisfy the stated objected. The requirements defined for the project were to acquire force and motion data using existing hardware and to integrate commercial software programs with the existing hardware.

In response to the objective, a rudimentary system architecture was created to establish the connection between the existing software and hardware components and identify areas within the system in which processes needed to be developed to facilitate the synchronization process. Software based algorithms were selected as the method to achieve the synchronization of data from the two systems.

In addition, before the algorithms were developed, a non-trivial selection process based on cost, software capabilities, and hardware compatibility was conducted to identify the software that was used to create the algorithms. LabVIEW was selected as the software tool used to develop the force acquisition algorithm and Vicon Tracker was selected to develop the motion acquisition algorithm. Python was also selected to develop an algorithm that would simultaneously trigger the force and motion acquisition algorithms. Following algorithm development and integration into the system architecture, verification testing was conducted to

determine whether the developed algorithms and overall system would meet the objective of the project. The acquisition start times were recorded for each system during trial testing and a Δ was calculated – this Δ represented the difference between the start times of both systems. Based on the Δ values observed during the verification testing, the objective of the process was not met. Instead of both acquisition algorithms running in parallel, they ran in series. Investigations conducted in relation to the software, developed algorithms, and hardware components used in the tested system revealed that the hardware components used in their current states could not support the parallel acquisition synchronization. Due to each system requiring an extensive number of resources to conduct acquisition, the computer was not able to schedule each acquisition task simultaneously.

5.2 Novel Contributions

The novel contributions made during this project are as follows:

- Developed an algorithm to automate the force acquisition process.
- Developed an algorithm to automate the motion acquisition process.
- Developed an algorithm to trigger the simultaneous acquisition of data from two external systems.
- Defined parameters for experimental testing to validate the force and motion acquisition algorithms.
- Defined the criteria to establish whether simultaneous measurements were achieved based on timing phase offsets observed during experimental testing.

5.3 Future Work

Despite this system configuration not being able to support the goal of simultaneous acquisition, there are other system configurations which could potentially satisfy the objective.

The first alternative system configuration proposed would be adding a secondary computer into the architecture. The secondary computer would reduce the resource strain on the first computer by managing the acquisition task from one of the systems. In this configuration, the main computer would connect to the second computer via a network connection to trigger acquisition from device connected to the secondary computer.

Another system configuration proposed was adding another peripheral device to the architecture. The peripheral device would allow for acquisition from the two systems to be triggered based on the devices internal device. In this configuration, the two devices would be connected to the added peripheral device. Although the developed algorithms and hardware did not result in the synchronization of the data acquisition, the system as whole can be modified to achieve the objective.

In addition to the implementation of other system configurations, altering task manager settings of the primary computer could also potentially satisfy the simultaneous acquisition task. The Windows operating system allows for priority setting using the computer's task manager. The priorities associated with executing the acquisition tasks from the force transducer and camera system would be set to 'high' or 'real-time' in the task manager ensuring that the CPU prioritizes these tasks over other tasks that are scheduled or in the queue to be scheduled.

Lastly, LabVIEW is used for not only acquisition of the raw voltage data from the force transducer, but it also converts the raw data to force/torque data. The development of an

algorithm to convert the data would eliminate the need to continue using LabVIEW since Python would oversee the acquisition and conversion of force data.

References

Bahlman, J. W., Swartz, S. M., & Breuer, K. S. (2013). Design and characterization of a multi-articulated robotic bat wing. *Bioinspiration and Biomimetics*, *8*(1). https://doi.org/10.1088/1748-3182/8/1/016009

Computer Vision: Stereo 3D Vision. (2022, November 4). Baeldung.

Dave Kuhlman. (2012). *A Python Book: Beginning Python, Advanced Python, and Python Exercises* (1.1a).

De Clercq, K. M. E., De Kat, R., Remes, B., Van Oudheusden, B. W., & Bijl, H. (2009). Flow visualization and force measurements on a hovering flapping-wing MAV "DelFly II." *39th AIAA Fluid Dynamics Conference*. https://doi.org/10.2514/6.2009-4035

Deng, S., Percin, M., & Van Oudheusden, B. (2016). Experimental investigation of aerodynamics of flapping-wing micro-air-vehicle by force and flow-field measurements. *AIAA Journal*, *54*(2), 588–602. https://doi.org/10.2514/1.J054403

Fry, A. C., Herda, T. J., Sterczala, A. J., Cooper, M. A., & Andre, M. J. (2016). Validation of a motion capture system for deriving accurate ground reaction forces without a force plate. *Big Data Analytics*, *1*(1). https://doi.org/10.1186/s41044-016-0008-y

Leys, F., Vandepitte, D., & Reynaerts, D. (2016). *A novel measurement setup to measure the fast harmonic motion and the power consumption of a hummingbird sized flapping wing driven by a resonant flapping mechanism*.

Meng, X., & Sun, M. (2016). Wing Kinematics, Aerodynamic Forces and Vortex-wake Structures in Fruit-flies in Forward Flight. *Journal of Bionic Engineering*, *13*(3), 478–490. https://doi.org/10.1016/S1672-6529(16)60321-9

Moodie, P. (2013). *Apparatus and method for physical evaluation* (Patent 8527217).

Percin, M., Oudheusden, B. W. V., Croon, G. C. H. E. D., & Remes, B. (2016). Force generation and wing deformation characteristics of a flapping-wing micro air vehicle “DelFly II” in hovering flight. *Bioinspiration and Biomimetics*, *11*(3). https://doi.org/10.1088/1748-3190/11/3/036014

Percin, M., Van Oudheusden, B., & Remes, B. (2017). Flow structures around a flapping-wing micro air vehicle performing a clap-&-peel motion. *AIAA Journal*, *55*(4), 1251–1264. https://doi.org/10.2514/1.J055146

Sane, S. P., & Dickinson, M. H. (2001). *The Control of Flight Force by a Flapping Wing: Lift and Drag Production*.

Select Your LabVIEW Edition. (2023). National Instruments. https://www.ni.com/en/shop/labview/select-edition.html

Shyy, W., Aono, H., Kang, C., & Liu, H. (2013). *An Introduction to Flapping Wing Aerodynamics*. Cambridge University Press.

Thomson, S. L., Mattson, C. A., Colton, M. B., Harston, S. P., Carlson, D. C., & Cutler, M. (2009). Experiment-based optimization of flapping wing kinematics. *47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. https://doi.org/10.2514/6.2009-874

Twigg, R. S. (2020). *AEROELASTIC CHARACTERIZATION OF REAL AND BIOINSPIRED ARTIFICIAL BUTTERFLY WINGS*. https://doi.org/10.2514/6.2020-2002

Wu, P., Stanford, B. K., Sällström, E., Ukeiley, L., & Ifju, P. G. (2011). Structural dynamics and aerodynamics measurements of biologically inspired flexible flapping wings. *Bioinspiration and Biomimetics*, *6*(1). https://doi.org/10.1088/1748-3182/6/1/016009

YANG, X., SONG, B., YANG, W., XUE, D., PEI, Y., & LANG, X. (2022). Study of aerodynamic and inertial forces of a dovelike flapping-wing MAV by combining experimental and numerical methods. *Chinese Journal of Aeronautics*, *35*(6), 63–76. https://doi.org/10.1016/j.cja.2021.09.020