ABSTRACT

Response of Passive Surface Hairs
in Steady and Unsteady Falkner-Skan Boundary Layers

Lance C. Case, M.S.

Committee Chairperson: Stephen T. McClain, Ph.D.

Arrays of biologically inspired artificial hair sensors for flow detection are being considered to provide small unmanned aerial vehicles greater platform stability through gust mitigation. Analytical models of hair sensor response to flow conditions have been previously developed, but fundamental assumptions of those models have remained essentially unvalidated. A model adaptation for non-wall-orthogonal fiber deflection was developed due to the geometric nature of the attached fibers. The current work seeks to validate this hair sensor model with wind tunnel testing results of hair sensor response to flows. Because the hair sensor arrays are not yet active, an optical fiber displacement measurement scheme and image analysis algorithms were developed to compute fiber deflection response to steady and unsteady flow conditions. Results indicate agreement between model predictions and experimental results sufficient for future sensor design employing the adapted model.

Response of Passive Surface Hairs
in Steady and Unsteady Falkner-Skan Boundary Layers

by

Lance C. Case, B.S.

A Thesis

Approved by the Department of Mechanical Engineering

William Jordan, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Master of Science in Mechanical Engineering

Approved by the Thesis Committee

Stephen T. McClain, Ph.D., Chairperson

Lesley M. Wright, Ph.D.

Lance Littlejohn, Ph.D.

Accepted by the Graduate School
August 2012

J. Larry Lyon,Ph.D, Dean

Page bearing signatures is kept on file in the Graduate School.

Copyright © 2012 Lance C. Case

All rights reserved

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

A	Cross-sectional area, Equation 2.11
A	Image area (pixels), Equation 3.4
AR	Aspect ratio
C	Power-law constant of proportionality
CCR	Cross-correlation array
C_D	Drag coefficient
E	Young's Modulus
I	Moment of inertia
L	Length
N	Number of samples
PV	Pixel value (in image)
PW	Pixel width (calibration coefficient)
R	Damping constant
Re	Reynolds number
Re_d	Reynolds number based on diameter
S	Sample standard deviation, Equation 3.7
S	Spring constant, Figure 2.4
T	Threshold value
U	Mainstream flow velocity in the x-direction, Equations 2.2, 2.4, 2.5, 2.10
U	Uncertainty, Equations 3.7 and 3.8
V	Mainstream flow velocity in the y-direction
$\mathbf{L}$	Fiber axial vector

$\mathbf{U}$	Flow velocity vector at boundary layer edge
$\mathbf{V}$	Flow velocity vector inside boundary layer
$\mathbf{r}$	Fiber deflection projected on x-z plane
d	Diameter
f	Falkner-Skan flow parameter
f_D	Drag force
g	Non-uniform load
h	Height
m	Falkner-Skan power-law parameter, Equation 2.2
m	Shape index, Equation 2.1
r	Deflection of fiber, Equations 2.11, 2.12, 2.13, 2.18, 2.19, 2.21
r	Fiber radial direction, Figure 2.14
t	Student's t, Equation 3.7
t	Time, Equations 2.11, 2.12, 2.13
u	Flow velocity in the x direction
v	Flow velocity in the y direction
x	Streamwise direction
x_{max}	x-location of max value (pixels)
y	Plate-normal direction
z	Spanwise direction
z_{max}	z-location of max value (pixels)
Δ	Fiber deflection measured experimentally
α	Fiber attachment angle in the y-z plane
β	Fiber attachment angle in the x-y plane

δ	Boundary layer thickness
ℓ	Fiber axial direction
η	Falkner-Skan similarity variable
γ	Kelvin-Voigt coefficient of material damping, Equation 2.11
γ	Wedge half-angle, Equation 2.2
μ	Dynamic viscosity of the fluid
ν	Kinematic viscosity of the fluid
ϕ	Fiber attachment angle in the x-z plane
ρ	Density
θ	Angular position

Acronyms

ACF	Aerodynamic Characterization Facility
AHC	Artificial Hair Cell
AR	Aspect Ratio
AoA	Angle of Attack
$C2$	Command and Control
CAS	Close Air Support
$CLAHE$	Contrast-Limited Adaptive Histogram Equalization
CR	Close Range
EN	Endurance
FSR	Force Sensitive Resistor
$HALE$	High Altitude Long Endurance
$HARM$	High Aspect Ratio Microstructure
HL	Hand-Launched

IFF	Identify Friend or Foe
ISR	Intelligence, Surveillance and Reconnaissance
L	Launcher
$LADP$	Low Altitude, Deep Penetration
$LPCVD$	Low-Pressure Chemical Vapor Deposition
LR	Long Range
$MALE$	Medium Altitude Long Endurance
MAV	Micro Air Vehicle
$MOOTW$	Military Operations Other Than Warfare
MR	Medium Range
NBC	Nuclear, Biological and Chemical (weapons)
OCA	Offensive Counter-Air
OCI	Offensive Counter-Information
$PDMS$	Polydimethylsiloxane
$RATO$	Rocket Assisted Take-Off
$REEF$	Research and Engineering Education Facility
REM	Raster Electron Microscope
RLG	Retractable Landing Gear
$RSTA$	Reconnaissance, Surveillance, and Target Acquisition
SAR	Search And Rescue
SAV	Small Air Vehicle
SEM	Scanning Electron Microscope
SR	Short Range
UAS	Unmanned Air System

UAV	Unmanned Air Vehicle
$VTOL$	Vertical Take-Off and Landing

Subscripts

A	Area-based
AR	Aspect ratio-based
a	Air
$binarized$	Binarized image
$brightspot$	Location of bright spot (pixels)
e	Edge (of the boundary layer)
f	Fiber
i	Image pixel counter (column)
$initial$	Initial
j	Image pixel counter (row)
m	Mound (fiber base)
$major$	Major axis
$minor$	Minor axis
n	Normal
rej	Rejection
s	Solid (fiber or hair)
$shift$	Image shift length (pixels)
tip	Tip
$total$	Total
x	In the x-direction
y	In the y-direction

z	In the z-direction
Δx	x-deflection measurements
Δz	z-deflection measurements
0	Base

Superscripts

$\hat{(\cdot)}$	Unit vector
$\overline{(\cdot)}$	Average

ACKNOWLEDGMENTS

My greatest gratitude goes first to my Lord and Savior Jesus Christ. Thank you, Lord, for your amazing gifts of life, grace, and purpose. Everything that I am, and all I have done or have yet to do - my God, it is all because of you!

I next want to express my appreciation to my advisor, Dr. Stephen McClain. This work would not have been possible without his wise direction, abounding patience, and timely encouragement. Dr. McClain, you are a great mentor and friend. I would also like to thank Dr. Lesley Wright and Dr. Lance Littlejohn for setting aside their time. I immensely enjoyed each of the classes taught by you all - now, I am honored by the presence of each of you on my committee.

My gratitude must also be extended to Dr. Benjamin Dickinson for his support. His vision, resources, and advice were an indispensable contribution to this thesis. I also want to thank Dr. Jennifer Talley for her advice on hair sensor construction, Dr. David Jack for his generously shared knowledge of class and style, Mr. Ashley Orr for his expertise in machining, and Haden Duke for his help with image analysis. Also, I thank Evan Martin, Steven Mart, Jason Gregg, Neil Jordan, Tim Burdett, Jimmy Becker, Theresa Vo, Logan Tecson, Cash Elston, Russell Mailen, and Charlie Brown for their help and general graduate student camaraderie.

I deeply appreciate Diana Milam, Courtney Burge, Joey Rodriguez, Zak Paroff, and my dear Aimie Cox for their support and encouragement during my studies. Finally, I want to thank my parents for their ceaseless love and support. Ink and paper alone will never tell of the gratitude and love I have for each of them.

Special thanks also goes again to Dr. Benjamin Dickinson, as well as Dr. Friedrich Barth, Dr. George Schmitz, Professor Jérôme Casas, as well as many other authors and contributors for their inspiring work. Additional thanks to IOP Publishing Ltd., The Royal Society, Springer and Springer-Verlag, CSIRO Publishing, The Company of Biologists, The Institute of Electrical and Electronics Engineers, The American Society of Mechanical Engineers, The Air Force Research Lab Munitions Directorate, The American Society for Engineering Education, The University of Florida, and The School of Engineering and Computer Science at Baylor University for the contributions that made this document possible.

To my family,

and to my Vine

"*...to the only God our Savior be glory, majesty, power and authority,*

through Jesus Christ our Lord, before all ages, now and forevermore! Amen."

Jude 1:25

CHAPTER ONE

Introduction

Over the past decade, the Small Air Vehicle (SAV), also referred to as the Micro Air Vehicle (MAV), has gained recognition as a potential platform from which tasks such as reconnaissance, surveillance, and target acquisition (RSTA) may be performed. The nominal 6-inch dimension of the SAV makes it a highly portable tool with the potential for a small logistics footprint. However, its light weight and characteristically small Reynolds number flight envelope makes the SAV particularly susceptible to wind gusts. The detection and mitigation of the adverse effects of these gusts is therefore important to the stability of the SAV platform.

1.1 The Basis of Interest in the SAV

The growing demand for SAV technology development is due to the growing necessity of on-the-spot military aerial capabilities. These capabilities may be divided into two categories - those directly enabling or supporting warfare operations, and civilian and military operations other than warfare (MOOTW). According to Huber [1], the SAV has been viewed as a potential tool for the fulfillment of several capabilities directly supporting warfare. In Huber's text the author lists potential capabilities of the SAV to include offensive counter-air (OCA), interdiction, close air support (CAS), strategic attack, offensive counter-information (OCI), and command and control (C2). Other capabilities that Huber suggests include special operations employment, and intelligence, surveillance and reconnaissance (ISR) operations including targeting, tagging, identification of friend or foe (IFF) and sensing nuclear, biological and chemical (NBC) agents. Potential non-warfare capabilities of

the SAV include: search and rescue (SAR) operations, border patrol, and humanitarian, peace, and anti-terrorism operations.

1.2 Limitations of the SAV

This on-the-spot nature of the SAV - referred to sometimes as "over-the-next-hill" or "around-the-corner" - drives the vehicles' design requirements towards portability and ease-of-use. These requirements define the SAV in terms of mass, performance characteristics, size, and expected flight envelope. Table 1.1 presents the SAV sub-group requirements overview within the broader class of Unmanned Aerial Vehicles (UAVs), adapted from Van Blyenburgh [2]. The SAV lies at the bottom of the currently-realizable spectrum of UAVs with respect to range, endurance, altitude, and other characteristics that increase the overall weight of the vechile.

Further, Table 1.2 presents an adapted estimate from the Lincoln Laboratory by Davis et al. [3] of the baseline mass distribution of a SAV with respect to the component level. As with most other aircraft, mass-efficiency of component systems is of great importance in the SAV. This is especially true in the flight control system with approximately four percent of the total mass allocation, a necessary system due to the reasons described below.

Figure 1.1 provides a basic visual order-of-magnitude comparison by McMichael and Francis [4] of the flight regime of the SAV to several other flying objects. The small Reynolds number flight regimes ($Re \sim 10^5$) under which SAVs operate, present the SAV platform with stability problems due to inherent flow unsteadiness and the larger propensity for flow separation over the aerodynamic surfaces of the SAV. Due to the small size, mass, and Reynolds number characteristic of the SAV, atmospheric flow phenomena such as wind gusts can have a considerable detrimental effect on the stability of the SAV platform as well, as mentioned by Shyy et

Table 1.1: UAV Classifications, Characteristics, and Examples, as adapted from Van Blyenburgh [2]

Categories (Abbrv.)	Data link Range (km)	Endurance (hours)	Maximum Flight Altitude (m)	Launch Method	Recovery Method
Nano	Unknown	Unknown	Unknown	Unknown	Unknown
Meso	Unknown	Unknown	Unknown	VTOL,	belly, expendable
Micro (μ)	<10	1	250	HL, VTOL	belly, skids, expendable
Mini	<10	<2	250	HL, VTOL, wheels	belly, skids, wheels, parachute
Close Range (CR)	10 - 30	2 - 4	3,000	HL, VTOL, wheels	belly, skids, wheels, parachute
Short Range (SR)	30 - 70	3 - 6	3,000	VTOL, RATO	belly, skids, parachute / airbag
Med. Range (MR)	70 - 200	1	3,000 - 5,000	VTOL, wheels, RATO	skids, wheels, parachute/airbag
LADP	>250	1	0.12 - 9,000	RATO	parachute/airbag
Long Range (LR)	>500	6 - 13	5,000	wheels, RATO	wheels
Endurance (EN)	>500	12 - 24	5,000 - 8,000	wheels, L	wheels, parachute/airbag
MALE	>500	24 - 48	5,000 - 8,000	RLG	RLG
HALE	>1,000	12 - 40	15,000 - 20,000	RLG	RLG

Table 1.2: Baseline SAV weight distribution [3]

Component	Mass (g)
Airframe	6
Propulsion	36
Flight Control	2
Communications	3
Payload	2
Total Mass	49

al. [5]. Because of limitations on the SAV in terms of both payload and power, novel flow sensors are sought for feedback into a platform stability control process to ensure the safe and effective flight of the SAV.

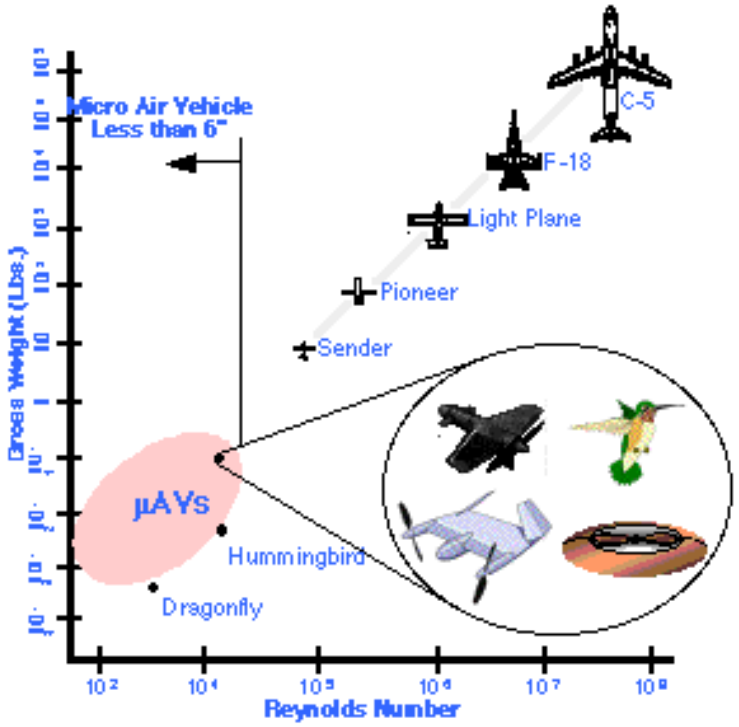


Figure 1.1: Comparison of the SAV flight regime with other flying objects [4]

1.3 Bio-inspired Hair Sensor Arrays

The flight regimes experienced by SAVs are consistently encountered in nature by many animals, including winged fliers such as bats, which possess distributed hair receptor arrays on their wing surfaces. Crowley and Hall [6] postulate that the purpose of these hair receptor arrays is to provide flow field feedback to the animal, allowing it to adjust its path according to its intention and current flow environment. Thus, the creation and use of artificial hair sensor arrays is a concept that draws inspiration from the hair sensor arrays observed in nature. Flow field feedback to the SAV controller may be provided by a suite of surface-mounted artificial hair sensor

arrays. However, an understanding of the relationship between flow field and hair sensor response is required.

1.4 Objectives and Significance of this Work

Because bio-inspired hair sensor arrays are being considered for flow sensing on SAV and other unmanned aerial system (UAS) platforms, predictive hair sensor deflection models will be consulted during the sensor design process. Currently, these models are based on large-scale fluid-structural interactions that are expected to scale to the boundary-layer and fiber scales for SAVs and other UAVs. While the models are employed for fluid/hair element interaction predictions, the underlying assumptions of the models are essentially unvalidated. The aim of this study is to measure fiber deflection on the boundary-layer and fiber scales to be used in the validation of these models. The validated models will be useful to UAS and SAV designers because more accuracy or confidence can be placed in the flow information obtained by these sensors, allowing for better control and platform stability.

1.5 Presentation Outline

This thesis presents the results of carbon fiber hair sensor deflection in response to Falkner-Skan boundary layers, discussed in detail later, for application in SAV platform stability sensors. Chapter Two discusses relevant information about biological and artificial hair sensor arrays and their use in boundary layer sensing, Falkner-Skan flows, and a model of non-orthogonal fiber deflection. Chapter Three details both the methodology used during the testing performed for this study, as well as the image analysis algorithms developed to distill pertinent information from the raw image data collected during experimentation, and the calculation of uncertainty. Chapter Four presents an analysis of data collected and reduced using methods from

the previous chapter. Chapter Five concludes the thesis with closing remarks and recommendations for future work. The appendices include the MATLAB® codes developed to analyze the image sets for both steady and unsteady flow conditions, and the non-orthogonal fiber deflection model as developed in MathCAD.

CHAPTER TWO

Technical Background

Before an explanation of the methods used in this study is given, several relevant topics must first be discussed. A review of hair sensor research is provided, regarding both biological and artificial hair sensors. A brief overview of boundary layers and Falkner-Skan flows is also provided. The development of the fiber deflection model under validation is then presented. Additional model considerations for fiber non-wall-orthogonality are also presented.

2.1 Bio-Inspired Hair Sensors

Bio-inspired hair sensors are being considered as a novel type of flow sensor for SAV applications. A review of both biological and artificial hair sensor research is given, with an emphasis on physical hair sensor properties and sensitivity. As shown by Brücker et al. [7, 8], the sensitivity, response, and corresponding resolution of hair sensors depend on the Young's modulus, E, of the hair sensor material, and also on the length-to-diameter aspect ratio, AR, of the hairs themselves. A comparison of biological and artificial hair sensor aspect ratios is provided at the end of the section.

2.1.1 Biological Hair Sensors

There are a multitude of examples in nature that demonstrate the use of distributed arrays of hair sensors to detect fluid flow. Among these examples are bats, which have distributed networks of hair sensors over their winged surfaces, crickets, whose cerci each contain hundreds of individual fluid perturbation-sensing filiform hairs, and spiders, whose many appendages each contain several hair structures of

different types. Research pertaining to the hair sensors found on these animals, as well as other arthropods and mammals, is ongoing.

Bats possess several arrays of hair sensors throughout their wing surfaces, and it is thought that the extreme maneuverability of these creatures in low Reynolds number flight is due in part to flow information gathered from these hair sensors. Histological studies of the grey-headed flying fox, a type of bat, performed by Crowley and Hall [6] have observed hairs up to 4mm long and $25\mu m$ in diameter on the wing surface, shown in Figure 2.1.

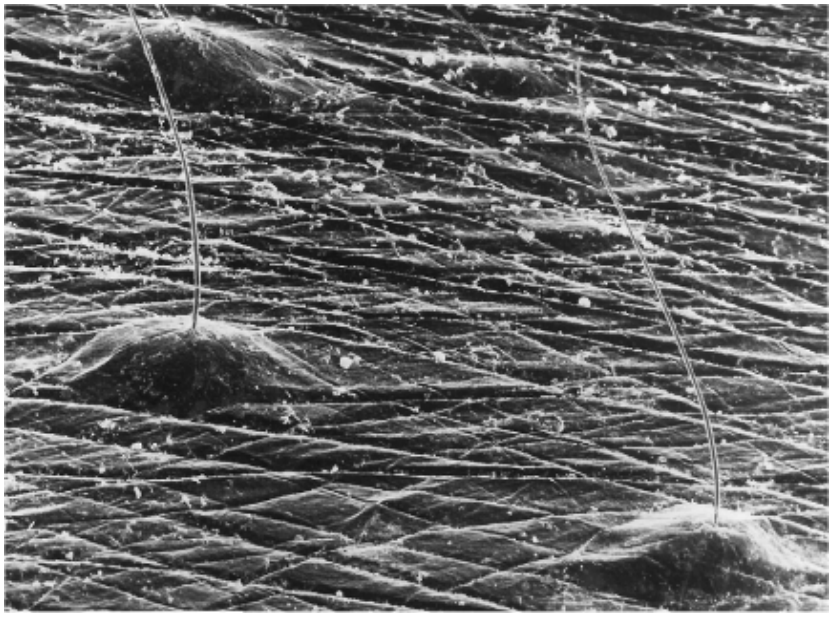


Figure 2.1: Hair-dome complexes on bat wing surface [6]. With permission from CSIRO PUBLISHING.

These hairs were found to grow from an array of domed structures up to 1 mm in diameter, later determined by Zook [9] to contain Merkel cell-neurite complexes (touch-sensing mechanoreceptors). While the hair-dome complexes were observed to grow at a multitude of angles from the surface, a trend relating surface hair location and incident angle was noted (but not defined). An additional trend between hair structure population density and location on the wing was observed, and it is

thought that the areas of higher hair population density coincide with areas along the wing surface where more turbulence is expected. Later electrophysiological studies performed by Sterbing-D'Angelo et al. [10] and Chadha et al. [11] using big brown bats confirmed the hypothesized purpose of hair-dome complexes in providing flow feedback information to the animal, noting the degree of over-representation of the winged surfaces in the primary somatosensory cortex. It was observed that the hair sensors exhibited highly preferential neural response based on flow direction and wing location. This was most notably observed along the trailing edge of the wing, where the sensors proved more sensitive to flow from the rear, typical of wake vortices. It is therefore thought that these hair sensors detect reverse, turbulent flow, and may have purpose in providing flight stability.

Cartilaginous and bony fishes, and other aquatic amphibians, possess similar arrays of hair sensors in structures called lateral lines as noted by Montgomery et al. [12]. The lateral line provides pertinent information such as location (azimuthal and polar) and distance of moving objects and other stimuli to the fish, as evidenced in behavioral studies by Coombs et al. [13, 14].

The cerci of cockroaches and crickets have also been thought instrumental in flow detection used in predator avoidance. Behavioral studies performed by Camhi et al. [15] support the idea that the flow-sensitive cerci of cockroaches aid in predator stimulus detection. Crickets have been observed to have arrays of small hair sensors distributed over their cerci, as shown by Dangles et al. [16] in Figure 2.2.

Various lengths of cricket cerci filiform hairs, from 30 to $1,500\mu m$, have been observed, and the function of these hairs in sensing fluid flow has been established through observation by Shimozawa and Kanou [17] of afferent response to oscillating flows of various frequencies. Scanning electron microscope (SEM) analysis of cricket

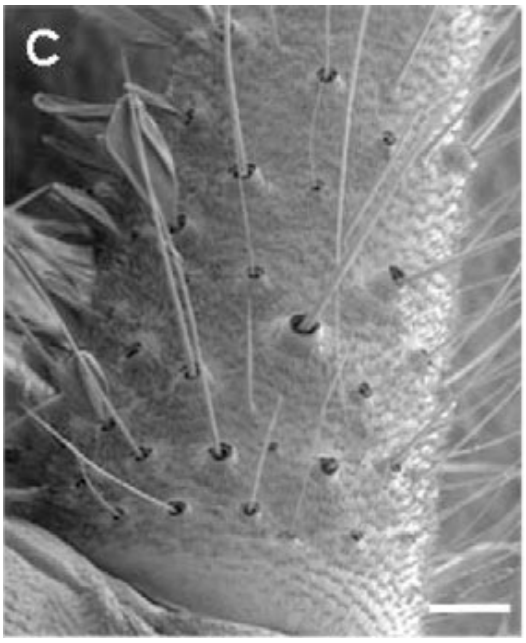


Figure 2.2: Scanning electron microscopy image of adult cricket cercus [16]. Reproduced/adapted with permission.

cerci hair receptors by Kumagai et al. [18] and Shimozawa et al. [19] have revealed a non-uniform hair profile. The hair profile was neither cylindrical, nor linearly tapered; instead, the hair was shown to follow a profile closer to a power function, shown in Equation 2.1, in which the variable m is referred to as the *shape index*. A schematic of the effect of the shape index m is shown in Figure 2.3.

$$d = d_0 \cdot \left(1 - \frac{L}{L_0}\right)^{\frac{1}{m}} \tag{2.1}$$

This profile was found to occur independent of observed hair length, which varied from $280\mu m$ to $1200\mu m$. A regression plot of the observed data is also presented in Figure 2.3, which was used in determining a shape index of $m = 1.91$ for the observed cricket cerci hair sensors, and $m = 2.22$ for cockroach cerci. It has been postulated (and proven) by Dickinson [20] that non-uniform fiber cross-section profiles, such as those observed in crickets and cockroaches, have favorable effects on hair sensitivity. Additional research by Steinman et al. [21] suggests that the shape of the

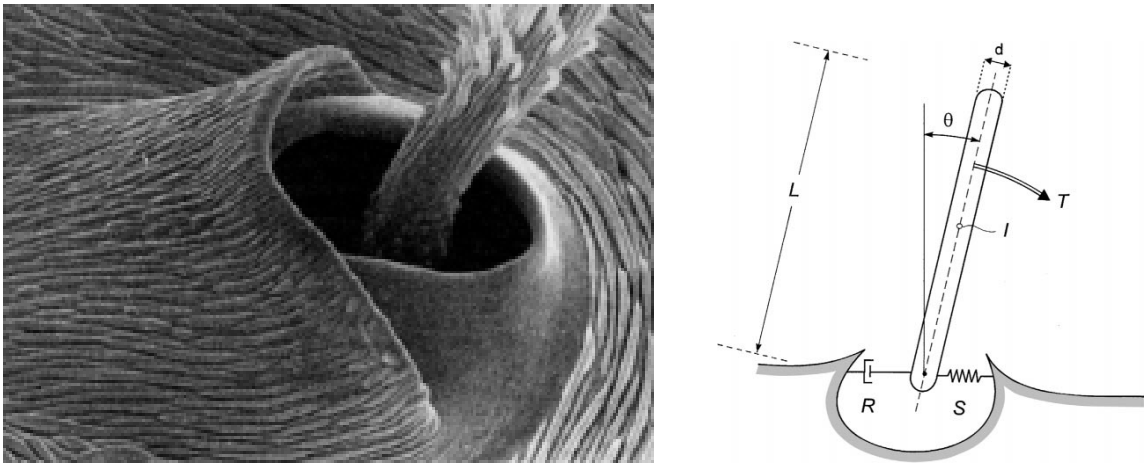
cricket cercus itself provides a sensitivity advantage to the cricket by introducing an element of spatial heterogeneity into the response of the flow-sensitive hairs attached to the round cercus.

(a) (b)

Figure 2.3: (a) Schematic drawing of hair shape, and (b) Relationship between diameter of hair sensor shaft and distance from hair tip [18]

The dynamics of spider trichobothria have been studied at length [22–28]. Figure 2.4(a) shows a scanning electron microscopy image of the base of one of the larger types of spider trichobothria, where a "cup" structure may be observed. A mechanical abstraction of this hair-and-cup structure is presented in Figure 2.4(b), where L and d are the length and diameter of the hair, respectively, I is the moment of inertia of the hair, θ is the angular position of the hair, and R and S are the damping and spring constants of the cup material, respectively. Information about the flow from the dynamics of the hair is passed via the deflection of the hair itself and into neuronal receptors in the cup structure. Despite observations of consistent

trichobothria length and other hair structure parameters between several samples, differences in trichobothria response to various oscillating flows were observed. It has been postulated that variances in the material composition of the cup structure alter the values of R and S, allowing for the tuning of individual hairs to be more sensitive to various airflow oscillation frequencies.



(a) (b)

Figure 2.4: (a) SEM view of spider trichobothria cup structure [23] Barth et al., Dynamics of Arthropod Filiform Hairs. II. Mechanical Properties of Spider Trichobothria (Cupiennius salei Keys.), Philosophical Transactions of the Royal Society B: Biological Sciences, 1993, 340, 1294, 445-461, by permission of the Royal Society., and (b) Mechanical trichobothrium and cup structure abstraction [27], with kind permission from Springer Science and Business Media

This frequency sensitivity is also observed in a different manner for trichobothria of a smaller type. Figure 2.5 presents a micrograph depicting the responses of smaller spider trichobothria of many lengths to an oscillating airflow. Instead of the variations in "cup" base material composition theorized above, obvious variations in trichobothrium length control the hair sensor sensitivity to various airflow oscillation frequencies.

Several other studies on biological and artificial hair sensors have also been presented. Fletcher [29] provides a first-order linear model of the acoustical response

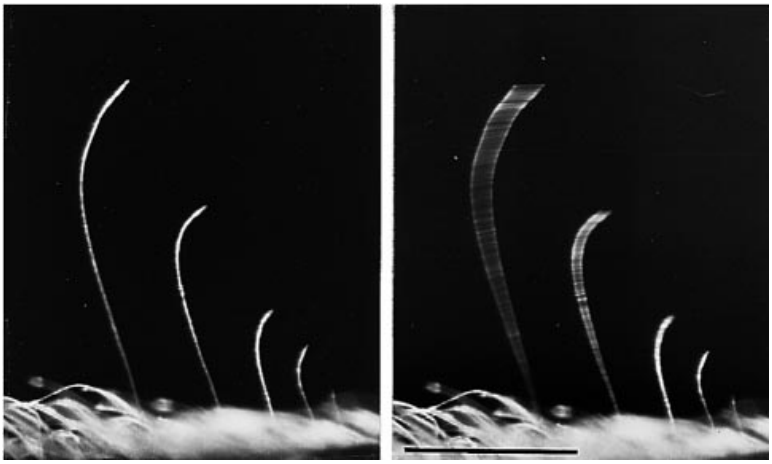


Figure 2.5: Micrograph showing the differential response of trichobothria of different length to an airflow oscillating at 50 Hz [27], with kind permission from Springer Science and Business Media

of an artificial hair receptor. Taylor and Krapp [30] offer a summary of recent studies on the wind-sensitive hairs of locusts and other animals.

2.1.2 Artificial Hair Sensors

A process of manufacturing microstructured surfaces with "flexible micropillars" or "microposts" was presented by Brücker et al. [7] and Schmitz et al. [31] to develop a spatially detailed, temporally resolved wall shear stress field measurement technique. The technique involved the creation of micropillar arrays using an excimer laser to drill small bores into a thin planar wax film, into which the transparent elastomer PDMS (the micropillar material) is cast before being melted away with hot de-ionized water. Then an imaging technique developed earlier by Brücker et al. [32] is used to measure the deflection response of these micropillar arrays. This imaging technique took advantage of the wall-orthogonality, low aspect ratio, and transparency of the manufactured micropillars (which allowed them to act as "optical microfibres") to measure micropillar tip deflection through the use of median and canny filters, followed by a circular Hough transform. While excellent at tracking tip

deflections of low aspect ratio micropillars, this imaging technique would not function when presented with non-wall-orthogonal fibers, or fibers of a different material.

Figure 2.6 shows a raster electron microscopy (REM) image of the first prototype of this so-called "lost mould" manufacturing technique. This manufacturing process produced surfaces with semi-hyperboloid-shaped fibers with a maximum length-to-diameter AR, of approximately 25.

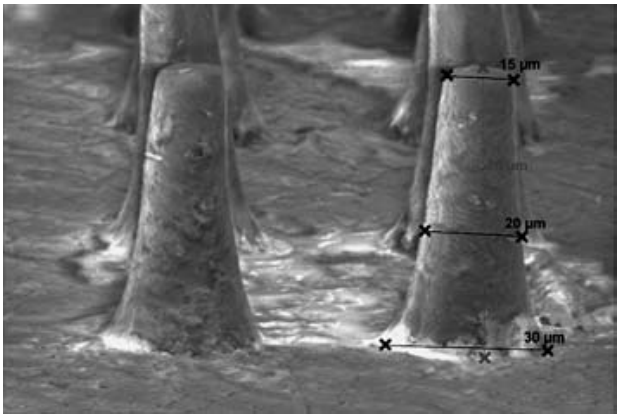


Figure 2.6: Raster Electron Microscopy (REM) image of the sensor film with microfabricated "type B" micropillars for wall shear stress imaging [7] with kind permission from Springer Science and Business Media

Figure 2.7 shows a scanning electron microscope image of an array of "high aspect ratio microstructures" (HARMs) created using the "lost mould" manufacturing process described previously. A 75% success rate of well-aligned microposts was observed for the "type B" micropost geometry, which presents structures with an average diameter of approximately 30 μm and an average aspect ratio of approximately 19. The remaining 25% of microposts in the array exhibited a deformed, curved spiral geometry which the authors will investigate further. Further investigations by Schaefer et al. [33] using this manufacturing process only observed straight microposts with diameters between 100 μm and 120 μm and lengths of approximately 1500 μm, yielding aspect ratios of approximately 15.

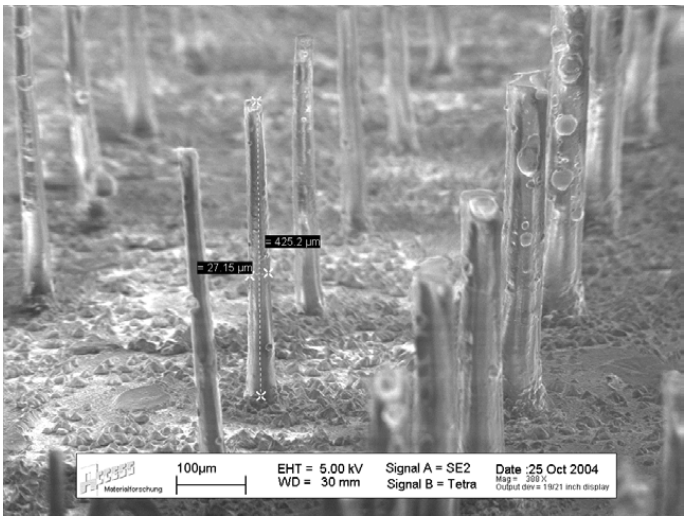


Figure 2.7: SEM sizing of "type B" micropost geometry [31] ©IOP Publishing Ltd.

Figure 2.8 shows an array of SU-8 photoresist hairs mounted normal to an array of capacitive read-out structures, manufactured by Dijkstra et al. [34] in an effort to transform artificial hair sensor arrays from simple passive deflection responders to active arrays of hair sensors that supply electrical information based on fiber deflection. The arrays were created through a combination of spin-coating and low-pressure chemical vapor deposition (LPCVD), by applying various materials to a highly conductive silicon wafer. These materials act as either sacrificial construction material, base spring or membrane material, or the hair material itself. Electrodes attached to the spring-membrane structure at the base of the hair aid in transmitting capacitive information related to the deflection of the hair caused by drag forces on the hair shaft. The creation of hairs up to 470 μm in length with an aspect ratio of approximately 8 was achieved using this technique.

The authors noted both that the aspect ratio of the hairs was variable by varying the diameter of the hairs, and that future work would involve doubling the spin-coating process to produce fibers twice as long. Additional research by this group aimed at transforming these sensor arrays into active arrays may be found in [35–38].

Figure 2.8: Array of spiral-suspended sensory hairs with SU-8 hairs of $470\mu m$ [34]

Another construction method concerned with the generation of active artificial hair cell (AHC) sensor arrays was developed by Engel et al. [39]. The method involves the use of carbon-impregnated polyurethane force-sensitive resistors (FSRs) on which the hair sensor is mounted. A micrograph of a single FSR is shown in Figure 2.9. In the figure, the FSR is shown without an attached hair sensor, which mounts directly over the exposed FSR surface shown. The FSRs were manufactured at the micro scale by combining a commercial polyurethane rubber, PMC121/30, with conductive carbon black material in various mixing ratios and using elastomer patterning techniques. At low mixing ratios of approximately 15 $wt\%$ carbon black, the FSRs exhibited poor conductivity, while at high mixing ratios of approximately 40 $wt\%$ carbon black, conductivity is enhanced at the expense of FSR sensitivity. While recommending that an appropriate mixing ratio balance was between $25 - 30$ $wt\%$, the authors also noted that there was still a large variance in initial FSR resistances due to imperfections in the manufacturing process; more research is required to suppress the adverse effects of the current manufacturing process before consistent FSR sensitivity is achieved.

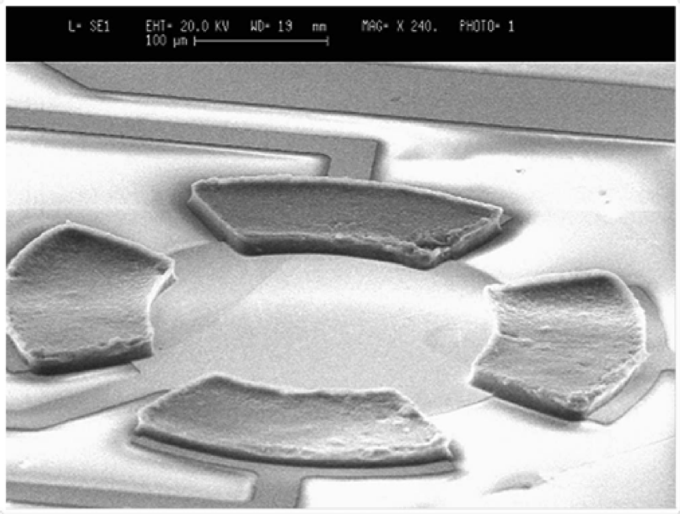
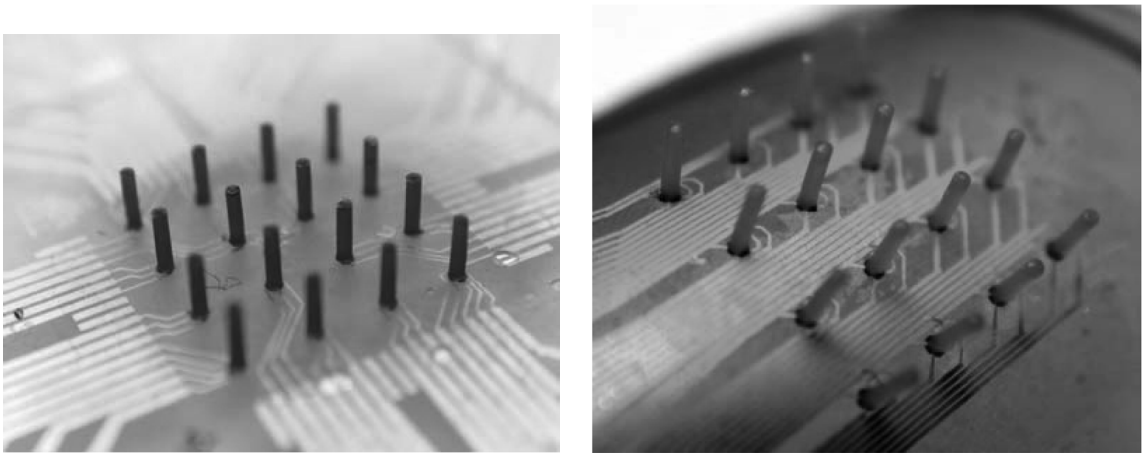


Figure 2.9: SEM micrograph of patterned FSRs [39] ©2006 IEEE.

Additional efforts by the authors were undertaken to create artificial hairs mounted to the FSRs described above in order to create a complete artificial hair cell sensor array, as shown in Figure 2.10 [39]. Due to cost limitations of manufacturing smaller FSRs, the authors created polyurethane AHC hairs using a lost mold process similar to those already described that were 500 μm in diameter and 3000 μm in height, yielding an AR of 6. The authors noted that the AHC sensors produced did not provide a sufficient "off-axis insensitivity" - that is, the sensors produced false information indicating some component of the true load was acting in an orthogonal direction. This type of off-axis insensitivity is necessary for accurate directional load sensing; more research would be required to correct this issue before use of this type of sensor is feasible.

A revised approach to the creation of AHC sensor arrays as presented above by Engel et al. [39] is given by Liu [40]. The revised approach, which is capable of producing single-sensor directional sensing, uses the same lost-mold fabrication technique for AHC creation as described above. While dimensions and material composition of both the FSRs and hair sensor material were not disclosed by the author, Liu



(a) (b)

Figure 2.10: (a) An AHC tactile sensor array and associated wiring on glass substrate.(b) AHC array on flexible polyimide substrate [39] ©2006 IEEE.

demonstrated the ability of these AHC sensor arrays to be integrated with hot-wire anemometry instrumentation on the same silicon wafer.

As previously mentioned, Brücker et al. [7, 8] has shown that the sensitivity, response, and corresponding resolution of AHC sensors depend on the Young's modulus, E, of the material, and also on the aspect ratio of the hairs themselves. When a comparison is made between the best aspect ratio exhibited by these artificially manufactured hair sensors, approximately 25, to aspect ratios observed on bat wing surfaces of approximately 160, it is apparent that these manufactured hair sensor arrays have yet to achieve the same level of sensitivity as those found in nature. Research in this area is ongoing however, and an understanding of the relationship between fluid flow and hair sensor response is still necessary even prior to the ability to manufacture surfaces with active hair sensors of sufficient sensitivity.

2.2 Boundary Layers

First systematically described by Ludwig Prandtl in 1904 [41], boundary layers are regions of a fluid flow near an immersed body where viscous effects dominate

the behavior of the flow. Figure 2.11 describes a boundary layer profile in terms of its distance from the leading edge of a flat plate [42], where $u(x, y)$ describes the velocity of the flow at a point (x, y) from the origin, U_∞ (referred to in this thesis as U_e) is the mainstream flow velocity, and $\delta(x)$ is the boundary layer thickness as a function of x. As can be seen in the figure, the magnitude of the velocity inside the boundary layer degrades rapidly when approaching the plate surface. While the velocity profile is dependant on the geometric characteristics of the surface in contact with the flow such as on a wedge or over an airfoil, the basic concept remains the same - fluid viscosity dominates the flow behavior close to a surface, where the hair sensors described previously are located.

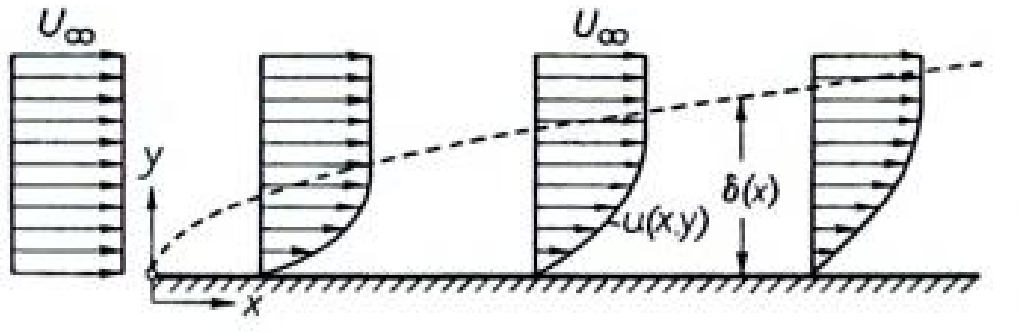
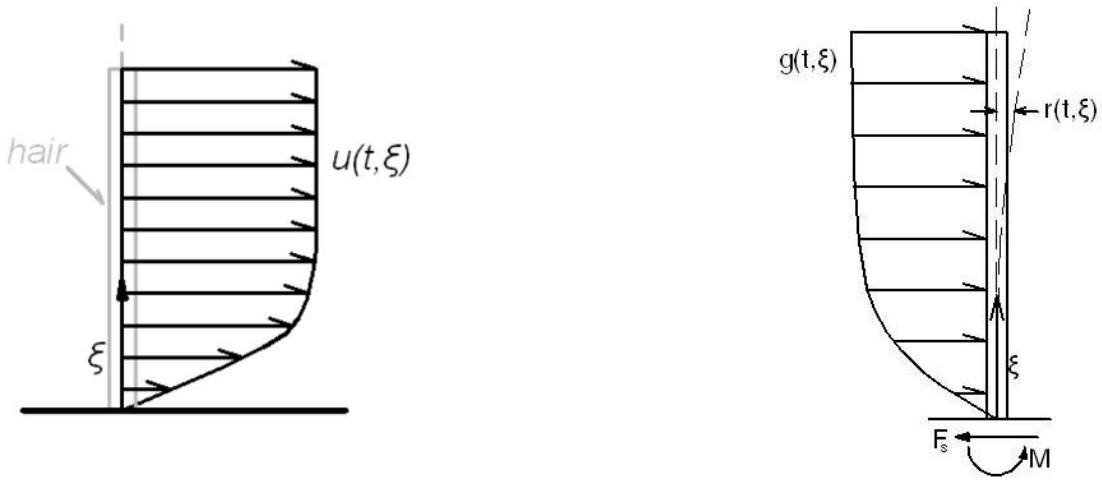


Figure 2.11: Boundary layer on a flat plate at 0° plate angle of attack [42]

Changes in the boundary layer are observable by these hair sensors. Figure 2.12(a) [43] describes a boundary layer experienced by a hair sensor element, and Figure 2.12(b) [43] shows the corresponding free-body diagram of the hair sensor under the nonuniform, distributed load $g(t, \zeta)$ applied to the hair from the fluid flow of velocity $u(t, \zeta)$. Deflection of the hair sensor, $r(t, \zeta)$ caused by this load is also shown in Figure 2.12(b). If an accurate understanding of hair sensor deflection response to fluid flow is to be obtained, proper characterization of the boundary layer is essential.



(a) (b)

Figure 2.12: (a) Nonuniform flow velocity profile incident on hair receptor, and (b) corresponding free body diagram of hair [43]

While under steady flow conditions, the load on the hair sensor, and its corresponding deflection, remains reasonably constant; however, the additional velocity component due to lateral gusting conditions serves to modify not only the size of the boundary layer and corresponding load applied to the hair sensor, but also the direction relative to the hair sensor along which the load is applied. This in turn modifies the magnitude and direction of the hair sensor deflection response, and the resultant moment and shear force at the sensor base. Previous studies by Aranyosi and Freeman [44] and Nam et al. [45] also note the relationship between hair motion and surface forces.

2.3 Falkner-Skan Flows

Falkner-Skan flows are canonical boundary-layer flow situations where the velocity at the edge of the boundary layer is described by:

$$U_e(x) = Cx^m \tag{2.2a}$$

$$m = \frac{\gamma}{2-\gamma} \tag{2.2b}$$

This edge velocity profile describes the flow over a wedge where γ is the half-angle of the wedge, as shown in Figure 2.13, and m is called the *power-law parameter* [46]. Similarity solutions to the two-dimensional, incompressible boundary layer equations

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.3}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = U_e\frac{dU_e}{dx} + v\frac{\partial^2 u}{\partial y^2} \tag{2.4}$$

may be found for Falkner-Skan flows using the similarity variable η, defined in Equation 2.5.

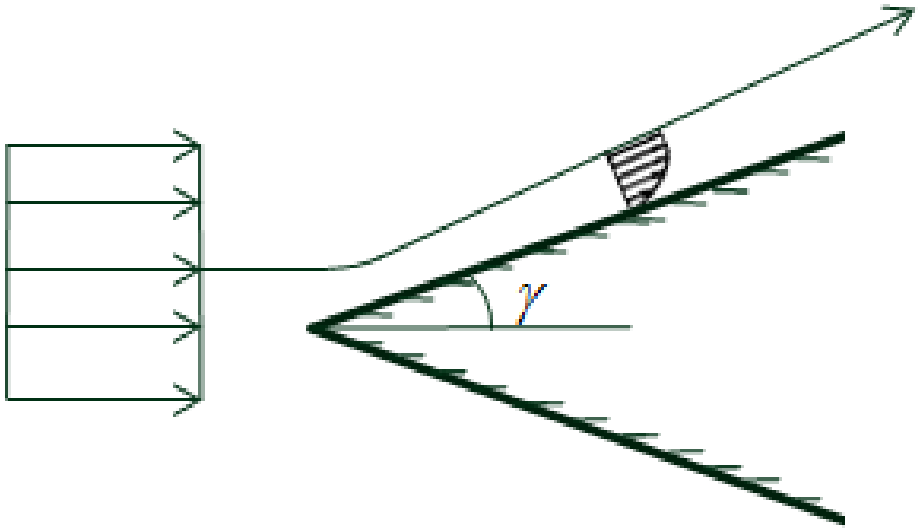


Figure 2.13: Falkner-Skan flow over a wedge with half-angle γ

$$\eta = y\left(\frac{m+1}{2}\frac{U_e}{vx}\right)^{\frac{1}{2}} \tag{2.5}$$

With this similarity variable, the boundary-layer equations may be transformed to the ODE of Equation 2.6.

$$f''' + \frac{m+1}{2}ff'' + m\left[1 - (f')^2\right] = 0 \tag{2.6}$$

With the boundary conditions:

$$\eta = 0 : f' = 0 \tag{2.7a}$$

$$\eta = 0 : f = 0 \tag{2.7b}$$

$$\eta \to \infty : f' \to 1 \tag{2.7c}$$

The solution of Equation 2.6 involves a numerical "shooting" method to determine the initial condition of f'' at the wall. Once the solution is found that matches the boundary values, the velocity components as noted in [42] at each elevation are:

$$u = U_e f' \tag{2.8a}$$

$$v = -\left(\frac{m+1}{2}\frac{\nu U_e}{x}\right)^{1/2}\left[\frac{m-1}{m+1}\eta f' + f\right] \tag{2.8b}$$

Thus, the velocity vector at each height along the length of the fiber is

$$\mathbf{V} = \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} \tag{2.9}$$

With the velocity vector at the edge of the boundary layer represented as

$$\mathbf{U} = \begin{bmatrix} U_e \\ V_e \\ 0 \end{bmatrix} \tag{2.10}$$

2.4 Non-Orthogonal Fiber Deflection Model

Dickinson [43] developed a laminar, unsteady hair sensor deflection model as presented in Equation 2.11. An Euler-Bernoulli beam deflection model, coupled with considerations for flow and material damping, is outlined. Drag equations for a circular cylinder in crossflow and the Kelvin-Voigt model comprise the hair sensor deflection model in Equation 2.11:

$$\rho_s A r_{tt}(t,\ell) + \gamma I r_{tt\ell\ell}(t,\ell) + EI r_{\ell\ell\ell\ell}(t,\ell) = f_D(t,\ell) \tag{2.11}$$
$$0 < \ell < L_f, \qquad t > 0$$

with the boundary conditions listed in Equation 2.12:

$$r(t,0) = 0 \tag{2.12a}$$

$$r_\ell(t,0) = 0 \tag{2.12b}$$

$$EI r_{\ell\ell}(t,L_f) + \gamma I r_{t\ell\ell}(t,L_f) = 0 \tag{2.12c}$$

$$EI r_{\ell\ell\ell}(t,L_f) + \gamma I r_{t\ell\ell\ell}(t,L_f) = 0 \tag{2.12d}$$

and the initial condition listed in Equation 2.13:

$$r(0,\ell) = r_0(\ell) \tag{2.13}$$

where ρ_s, r, L_f, describe hair density, deflection, and length, respectively; E, I, and A describe the hair Young's Modulus, moment of inertia, and cross-sectional area, respectively; γ is the Kelvin-Voigt coefficient of material damping; and f_D is the distributed load due to the drag forces acting along the length of the fiber. The subscripts ℓ and t denote partial derivatives.

Because of imperfections in the construction process outlined in the next chapter, it was impossible to ensure wall-orthogonality of every fiber constructed or studied. Instead, according to the coordinate system presented in Figures 2.14 and 2.15, the fiber deflection model described above must be expanded to allow for a non-orthogonal axial fiber vector. The fiber vector is described in Equation 2.14:

$$\mathbf{L} = L_f \hat{\mathbf{L}} \tag{2.14a}$$

$$\hat{\mathbf{L}} = \begin{bmatrix} sin(\alpha)cos(\beta) \\ sin(\alpha)sin(\beta) \\ cos(\alpha)sin(\beta) \end{bmatrix} \tag{2.14b}$$

where the fiber angles relative to the wall and flow are presented in Figure 2.15, and h_f and h_m denote heights of the fiber and the mound on which the fiber is mounted.

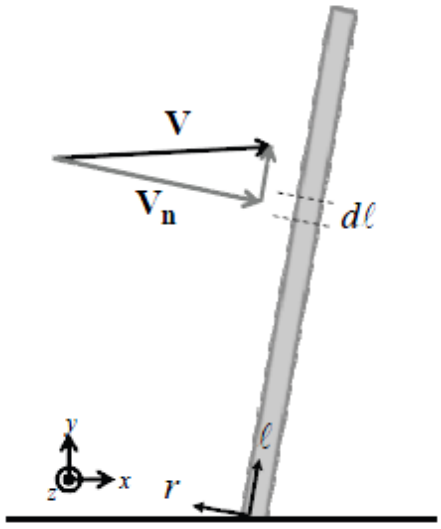


Figure 2.14: Fiber representation and relative local velocity

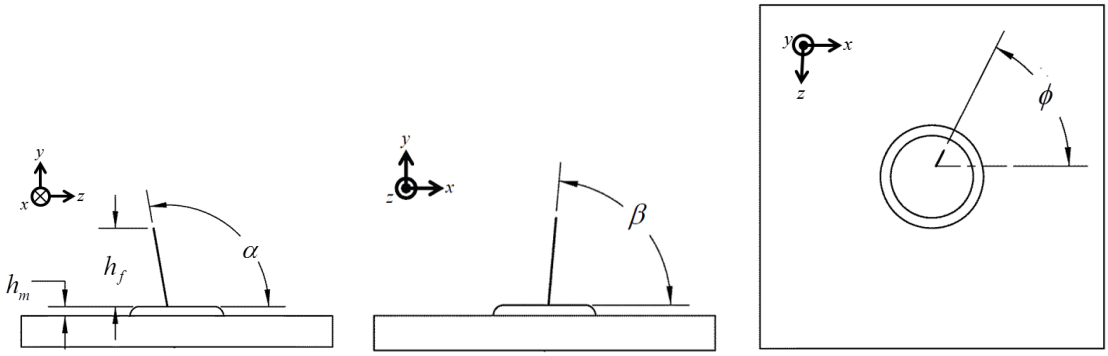


Figure 2.15: Angles and dimensions used to describe fiber geometry

Then for an incremental section of fiber, $d\ell$, the normal force on the section is:

$$f_D = \frac{1}{2}\rho_a C_D \left(\mathbf{V}_n \cdot \mathbf{V}_n\right) d \tag{2.15}$$

where:

$$\mathbf{V}_n = \mathbf{V} - \left(\mathbf{V} \cdot \mathbf{L}\right) \hat{\mathbf{L}} \tag{2.16}$$

and:

$$C_D = exp\left[-0.67ln\left(Re_d\right) + 2.51\right] \tag{2.17a}$$

$$Re_d = \frac{\rho d|\mathbf{V}_n|}{\mu} \tag{2.17b}$$

For the steady flows studied in this work, Equation 2.11 reduces to Equation 2.18:

$$\frac{d^4 r}{d\ell^4} = \frac{f_D}{EI} \tag{2.18}$$

With boundary conditions of zero deflection at the rigid fiber base, and zero shear and moment at the fiber tip. These boundary conditions are listed in Equation 2.19:

$$\ell = 0 : r = 0 \tag{2.19a}$$

$$\ell = 0 : \frac{dr}{d\ell} = 0 \tag{2.19b}$$

$$\ell = L_f : \frac{d^2 r}{d\ell^2} = 0 \tag{2.19c}$$

$$\ell = L_f : \frac{d^3 r}{d\ell^3} = 0 \tag{2.19d}$$

The fourth-order ODE in Equation 2.18 may be integrated to determine the deflection at the tip. One major assumption is that while the direction of the flow component normal to the fiber axis changes slightly along the length of the fiber, the deflection found by integrating Equation 2.18 occurs in the direction of $\mathbf{U}_n$ where $\mathbf{U}_n = \mathbf{U} - (\mathbf{U} \cdot \mathbf{L})\,\hat{\mathbf{L}}$. Thus, once the deflection is determined, the deflection projected on the flow and optical coordinate system is found using

$$\mathbf{r} = r\hat{\mathbf{U}}_n \tag{2.20}$$

The imaging system described in the subsequent chapter only measured the fiber deflection in the x-z plane. Thus, to compare the analytical fiber deflection to the experimental measurements, Equation 2.21 is used.

$$r_{total} = \left(r_x^2 + r_y^2\right)^{\frac{1}{2}} \tag{2.21}$$

CHAPTER THREE

Methodology

For this study, an array of nine carbon fiber hairs was mounted to a flat plate constructed in a thermal printer. The flat plate studied was constructed to have boundary layer characteristics similar to those of flow over the wing of a medium-sized bat [10]. To measure the passive mechanical response of these carbon fibers, an optical imaging technique was used whereby a camera of known spatial resolution recorded images of a given carbon fiber while exposed to several mainstream flow velocities. Data reduction was then performed using several image processing functions in MATLAB®.

3.1 Plate and Carbon Fiber Array Construction

3.1.1 Plate Construction

The flat plate had a finite length of 121.16 mm (4.77 in.), a span of 177.80 mm (7.00 in.), a thickness of 4.76 mm (3/16 in.), and a 10 degree leading edge angle. This plate length was chosen to provide a nominal plate Reynolds number of 25,000 at 3 m/s. The flat plates were made of ABS plastic, created using a Dimension 3D thermal printer housed at Baylor University. Because of the finite printer height resolution, a 0.51 mm (0.02 in.) radius of curvature was employed for the leading edge. This radius of curvature ensured that a minimum of two layers of plastic were used along the span of the leading edge. Figure 3.1 shows the isometric and plan-form views of the test plate.

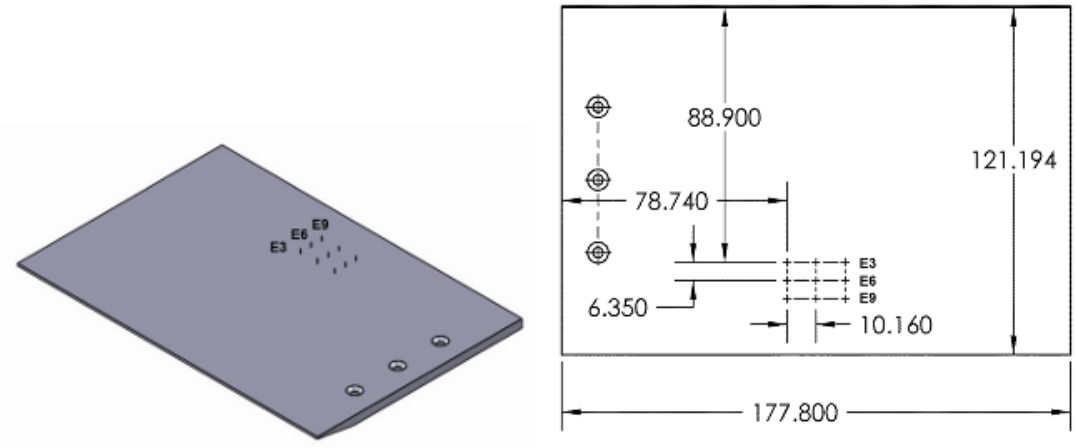


Figure 3.1: Test plate (dimensions in mm)

3.1.2 Plate Surface Preparation

Several methods were examined to determine the most feasible surface upon which a bead of gel cyanoacrylate and a carbon fiber could be attached. Different surface textures were explored to improve glue adhesion. Additionally, the visibility of the carbon fiber against a contrasting surface background was also an important factor. Several 50.8 mm x 50.8 mm (2 inch x 2 inch) ABS plastic test coupons were made in a Dimension SST extrusion-deposition 3D printer to study different preparation methods with these parameters in mind. Application of wood putty was attempted on one of the test coupons, but ruled out due to poor glue adhesion. Sanding the surface with a 180-grit sanding block proved to be the most beneficial preparation method, providing a smoother surface than the raw thermal-print model structure while retaining enough roughness for effective application of the glue. To create enough contrast between the surface and the carbon fiber, use of a flat black spray paint on a test coupon was initially attempted in conjunction with the application of a small amount of white paint to the tip of a carbon fiber. However, the added mass of the paint at the tip of the carbon fiber significantly altered its behavior.

Instead, a coat of flat white spray paint was applied to the carbon fiber array surfaces (first to the test coupons, then to the flat plate) to contrast the naturally-dark carbon fiber enough for feasible motion capture of the carbon fibers.

3.1.3 Carbon Fiber Array Construction

Figure 3.2 shows the carbon fiber array construction facility used in the investigation. A stand supporting the carbon fiber array surface was placed under a Zeiss OPMI-1 microscope with an attached LFS 200 Fiberoptic Light Source. A second stand, supporting a Narishiye Micromanipulator, was also placed under the microscope. The fibers used in this study were cut from a tow of Thornel T-650 PAN-based fibers. The nominal tensile modulus of the fibers is 255 GPa, and the nominal filament diameter is 6.8 μm.

Using a pair of Jewlers #5 Forceps from World Precision Instruments (WPI) and a soldering iron, a strand of carbon fiber was embedded into a piece of paraffin wax material located on the arm of the micromanipulator. This step, more than any other, introduced a large variance in fiber attachment angle because of the difficulty associated with embedding the carbon fiber in the wax. A small amount of gel cyanoacrylate was then applied to the array surface in a predetermined location using a toothpick. The free end of the carbon fiber was positioned into the bead of gel cyanoacrylate to form a butt joint. Then, the bead was allowed to harden while the carbon fiber and micromanipulator remained in place. The end of the carbon fiber attached to the bead of gel cyanoacrylate was cut to the desired length from the end embedded in the paraffin wax using a pair of WPI titanium micro-scissors. This process was repeated for each location, to create a three by three array of carbon fiber hair sensors. The carbon fibers were named according to their location on the plate, from fiber E1 to fiber E9 as depicted in Figure 3.1.

3.1.4 Carbon Fiber Geometry Determination

After the fibers were clipped, the video inspection system shown in Figure 3.3 was used to measure each fiber's dimensions and orientation. The video inspection system consisted of Scienscope WSXGA VGA CMOS camera, a Scienscope micro zoom lens, and an annular ring light mounted on a single-arm boom stand via a standard Scienscope focus mount. The imaging system was output to a standard LCD monitor via VGA, and the system is illuminated with a 24W Scienscope LED fiber optic illumination unit. Bitmap-format images are saved via a removable SD card using the Scienscope CMOS camera.

The motion of four of the fibers, (E3, E3c, E6, and E9, as shown in Figure 3.1) was characterized in the study. For each fiber, top, side and frontal images were acquired and used to determine the fiber dimensions and orientation. Figure 3.4 presents example images from the ScienScope video inspection system used to determine the geometry of fiber E3.

Calibration images of a machinists rule with 1/32-inch increments, shown in Figures 3.5 and 3.6, were taken to determine the correlation between physical length and image pixel width and height from the cameras for the imaging systems used in both the steady and unsteady flow cases, respectively. This was accomplished by counting the number of pixels between one edge of a hash mark on the machinists rule and the corresponding edge on the adjacent hash mark. Due to the small depth of focus of the camera, variations in this correlation are considered negligible.

Table 3.1 presents the geometry of each of the fibers studied. Note that E3c is a clipped version of E3. After all of the motion of E3 was characterized, the fiber was clipped to approximately 50% of its original length and the motion characterization was performed again.

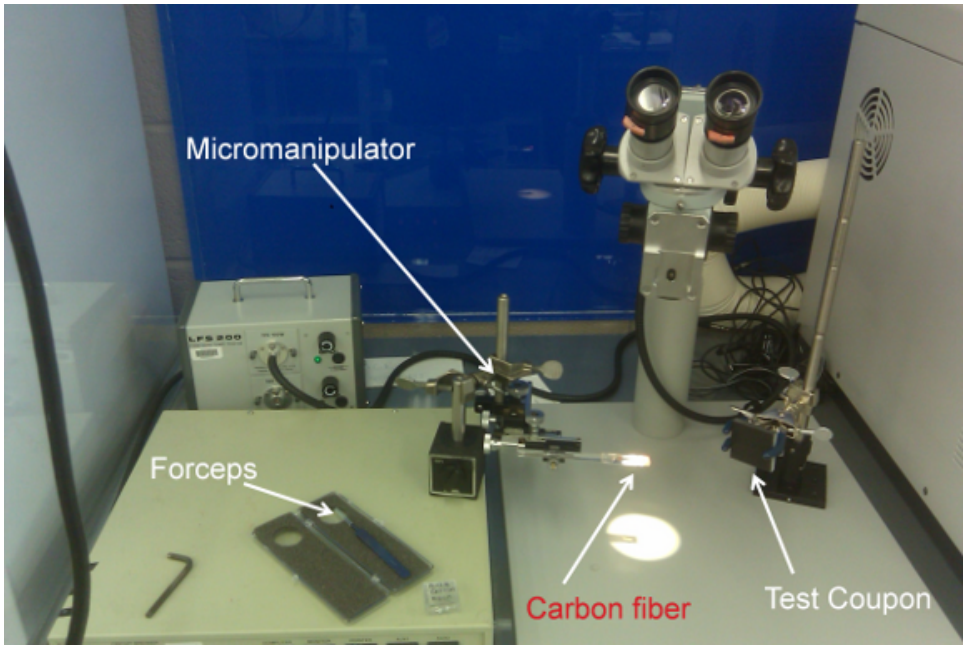


Figure 3.2: Microscope and micromanipulator system for fiber attachment

Figure 3.3: Camera inspection system constructed for fiber measurement

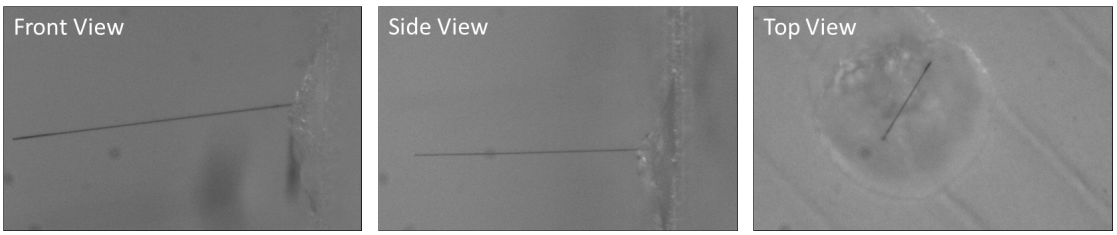


Figure 3.4: Example images used for fiber geometry determination (E3 is shown)

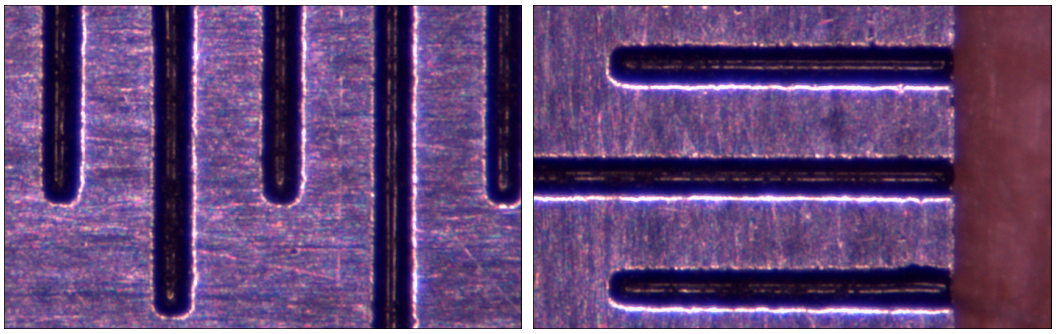


Figure 3.5: Steady flow calibration images

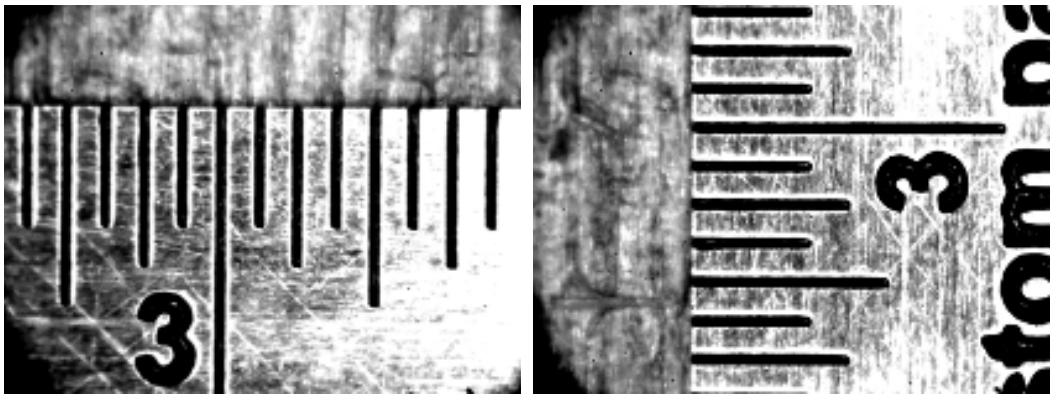


Figure 3.6: Unsteady flow calibration images

Table 3.1: Fiber dimensions and geometry relative to plate

$Fiber$	$L_f(mm)$	α	β	ϕ	$h_m(mm)$	$h_f(mm)$	$x_f(mm)$
$E3$	3.10	101.0°	83.1°	52.7°	0.18	3.01	88.9
$E3c$	1.63	101.0°	83.1°	52.7°	0.18	1.58	88.9
$E6$	3.25	97.7°	86.8°	56.8°	0.12	3.22	95.3
$E9$	2.80	95.4°	108.6°	115.9°	0.22	2.64	101.6

3.2 Wind Tunnel Fiber Motion Measurement

The fiber response measurements were performed in the Aerodynamic Characterization Facility (ACF) at the University of Florida's Research and Engineering Education Facility (REEF). The ACF, shown in Figure 3.7, is an open-loop, open jet wind tunnel with a 1.07-m by 1.07-m (42-in. by 42-in.) cross-section. The ACF is an open jet wind tunnel in that sidewalls do not bound the flow as it passes through the test section. A 40-hp motor and axial fan enable freestream velocities of 0.25 m/s to 22 m/s [47].

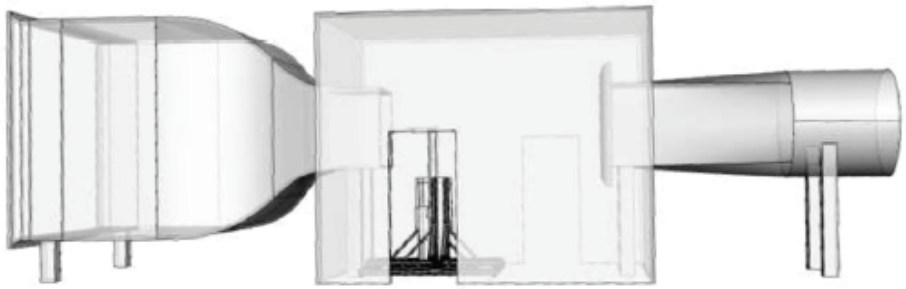


Figure 3.7: The ACF at the University of Florida REEF [47]

A wind tunnel test stand was constructed so that the test plate was placed within the wind-tunnel jet. Since the fibers were not active sensors, components from the ScienScope video inspection system were used to determine the fiber deflection when exposed to various flows. The test stand was constructed so that the plate surface with the carbon fiber array pointed towards the floor while the optics and

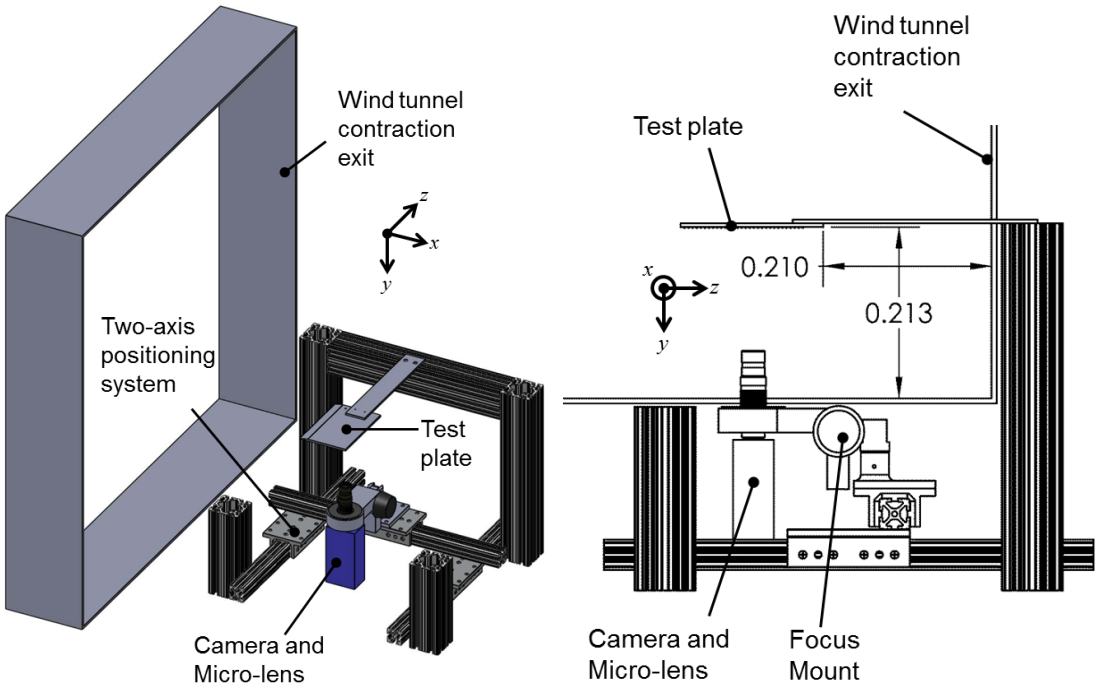


Figure 3.8: Wind tunnel experimental apparatus (dimensions in m)

the camera were placed below and outside the jet to reduce aerodynamic loading and vibration during flow measurements. A two-dimensional traversing system was used to position the camera and optics on each individual fiber of the array. A focusing mount enabled motion of the camera and optics in the direction normal to the test plate surface. Diametric and rear-view drawings of the apparatus and its position relative to the wind-tunnel contraction exit are presented in Figure 3.8. Figure 3.9 presents two images of the constructed apparatus.

A Pitot-static probe was used to measure the speed of the air flowing through the test section during each of the steady-flow and unsteady-flow tests. As can be noted in Figure 3.9, the Pitot-Static probe was placed such that the stagnation port of the probe was just behind the trailing edge of the test plate. A Heise ST-2H pressure transducer with a range of 0-2 inH_2O was used to measure the Pitotstatic pressure differential, a Druck DPI 142 barometer was used to measure the absolute pressure in

the wind tunnel test section, and a thermistor was used to measure the temperature of the air in the test section. No time-resolved flow measurements were taken during the unsteady-flow tests.

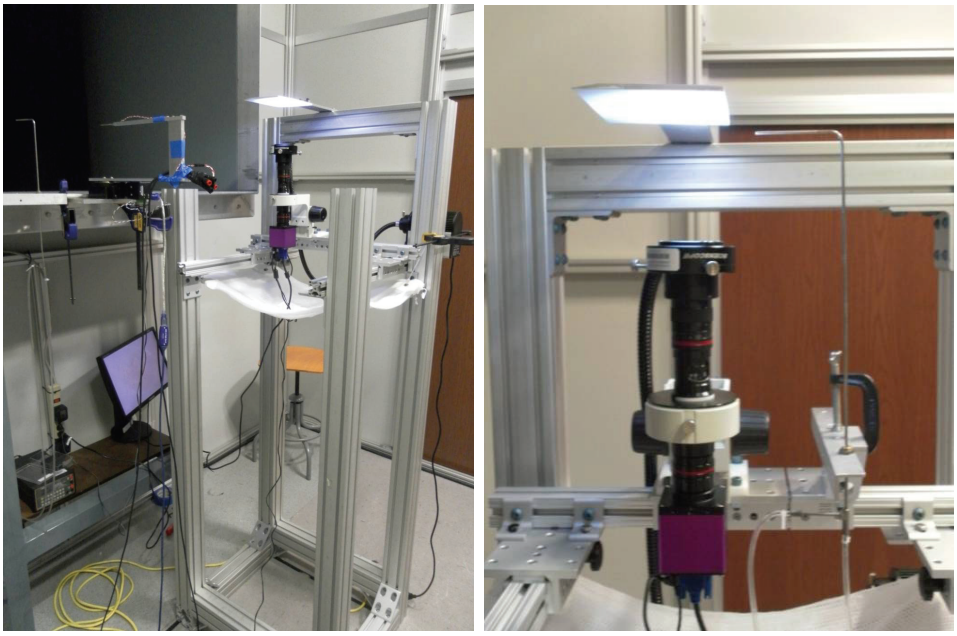


Figure 3.9: Wind tunnel apparatus for fiber motion detection under steady flow

The imaging system used to capture steady fiber motion consisted of 2-Megapixel ScienScope CMOS camera, a ScienScope micro zoom lens, an annular ring light mounted on the micro zoom lens, and a standard ScienScope focus mount. The imaging system was output to a standard LCD monitor via VGA, and the system was illuminated with a 24W ScienScope LED fiber optic illumination unit. Bitmap-format images were saved via a removable SD card using the ScienScope CMOS camera. For each steady-flow condition, ten images of the fiber position were acquired. The imaging system used to capture unsteady fiber motion consisted of a Phantom v7.1 camera recording 256x256-pixel images at 250 fps, the ScienScope micro zoom lens, annular ring light, and 24W ScienScope LED fiber optic illumination unit used in the steady flow imaging system. An extra 250W flood light focused on the region

of interest was necessary for recording images at 250 fps. A flap was added to the upstream side of the open-jet test section in order to introduce manually-controlled perturbations (gusts) into the flow during the unsteady tests.

3.3 Test Matrix

The effects of several parameters on hair sensor response to Falkner-Skan flows were observed. The angle of attack (AoA) of the test plate was set at $-2.5°$ and $+2.5°$ in order to observe the effects of favorable and adverse pressure gradients on the hair sensors. The effect of hair sensor penetration into the boundary layer was also observed by varying the mainstream flow velocity inside the wind tunnel test section from 1-10 m/s. This let to a fiber tip Reynolds number based on their nominal diameters ranging from 0.2 to 4.3. Possible changes in hair response due to both hair sensor attachment angle and the presence of an upstream fiber were also considered (note the x_f, α, β, and ϕ properties listed in Table 3.1). The effects on fiber deflection response due to both steady and unsteady flows were measured in this study. Table 3.2 outlines the test matrix considered in this study.

Table 3.2: Test matrix

Flow Regime	Plate Angle of Attack	Mainstream Velocity (m/s)	Fibers Characterized
Steady	$+2.5°$, $-2.5°$	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10	E3, E3c, E6, E9
Unsteady	$+2.5°$, $-2.5°$	3, 6, 9	E6

3.4 Image Analysis

Two separate MATLAB® image analysis scripts were developed to analyze the experimental images obtained during both the steady and unsteady flow cases. While the same basic MATLAB® image analysis functions are used for both the steady and unsteady experiments, differences in the experimental implementation and data collection require separate scripts to effectively process the images.

3.4.1 Steady Flow Image Analysis

Each steady carbon fiber image set is processed in the following manner. A boundary box in the image set is first determined around a specific interrogation region. An average reference image is then constructed by combining every image in the "zero-flow" image set. Then, every image in an image set corresponding to a different mainstream velocity is analyzed using several MATLAB® functions to distinguish the carbon fiber from its background, determine the pixel location of the carbon fiber tip, and correct for any shift in the images relative to the average reference image. Criterion for data rejection is then used to reject individual spurious data points. Finally, fiber deflection in both the lateral and longitudinal directions, as well as the corresponding uncertainties, for each flow case image set are computed from the remaining data points.

3.4.1.1 Initial Analysis Steps. To minimize computation time, a box bounding a specific image interrogation region is first determined, as shown by the green "+" markers in Figure 3.10. This box is defined so that the tip of the carbon fiber always remains inside the interrogation region for every flow velocity studied. However, care is also taken as much as possible to exclude any anomalies present in the background of the image that may interfere with the proper determination of the carbon fiber tip location, such as the diagonal thermal printer striations found in the background of Figure 3.10.

A reference image (not shown) is then computed from the 8-bit grayscale average of every image present in the zero-flow image set. This average reference image is used later in a cross correlation to determine and correct for the magnitude of the image shift present in each image analyzed.

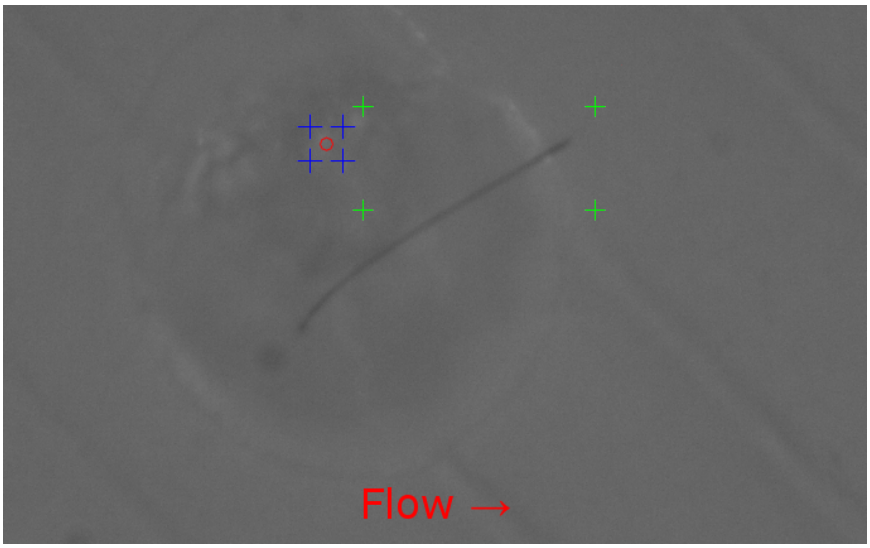


Figure 3.10: Steady flow raw image

3.4.1.2 Fiber Image Reduction. Each image in the data set is then sequentially read using the `imread()` function in MATLAB®. This image is cropped down to the interrogation region, and converted to grayscale using `rgb2gray()`, as shown in Figure 3.11.

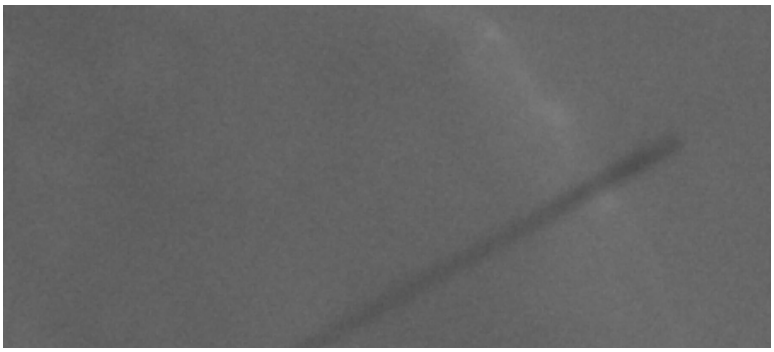


Figure 3.11: Steady flow cropped image

After the image is cropped down, and converted to grayscale using `rgb2gray()`, the `adapthisteq()` function is used to perform a Contrast-Limited Adaptive Histogram Equalization (CLAHE), shown in Figure 3.12. CLAHE enhances the contrast of small interest regions, called *tiles*, such that the histogram of each tile

approximates a specified distribution. After this histogram equalization is completed for each tile, bilinear interpolation is used to combine neighboring tiles while eliminating artificial tile boundaries. For this static image analysis, each image is split into a 3x3 array of tiles, and a uniform distribution is approximated.

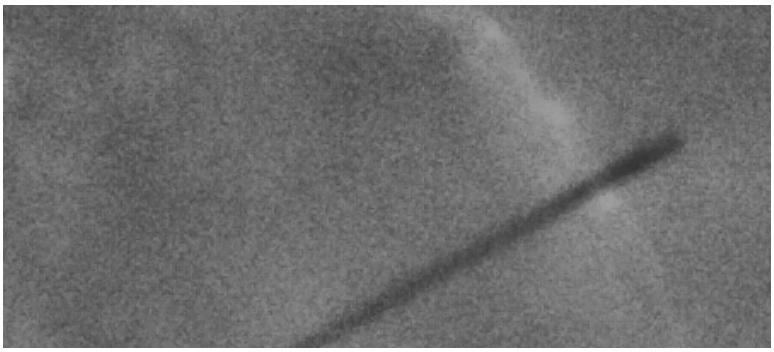


Figure 3.12: Steady flow CLAHE image

To isolate the carbon fiber from the background in the subsequent steps of this image analysis using MATLAB®, the image must be converted from a grayscale image to binary, as shown in Figure 3.13. This binarization process is completed using the `im2bw()` function in MATLAB®, specifying an 8-bit binarization threshhold integer for each image set corresponding to a different mainstream velocity.

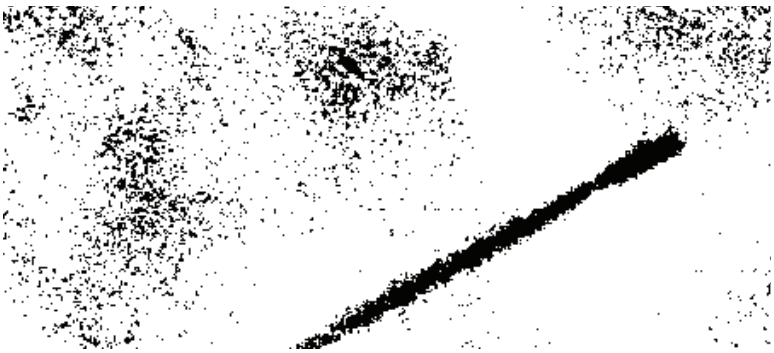


Figure 3.13: Steady flow binarization image

To remove much of the inherent noise present in the binarized image represented by Figure 3.13, the image is then filtered using the `bwmorph()` function in

MATLAB®, shown in Figure 3.14. For this static image analysis, a *majority* filtering operation, whereby a pixel value is set to 0 unless at least 5 of its connecting neighbors have a value of 1, is used to remove the salt-and-pepper-style noise present in the image. Then, a *fill* operation is used to prevent hollow regions and a *thin* operation is used to prevent unwanted breaks in the carbon fiber due to bright spots present in the image background that 'bleed' into the foreground, as can be seen near the bottom of the image in the image progression between Figures 3.12 and 3.13.

Figure 3.14: Steady flow filtered image

The image must then be inverted before the MATLAB® fiber detection function discussed below can be used. Since the image is already in binary format, this inversion process is accomplished with the simple equation: `image = 1 - image`. This inversion is shown in Figure 3.15.

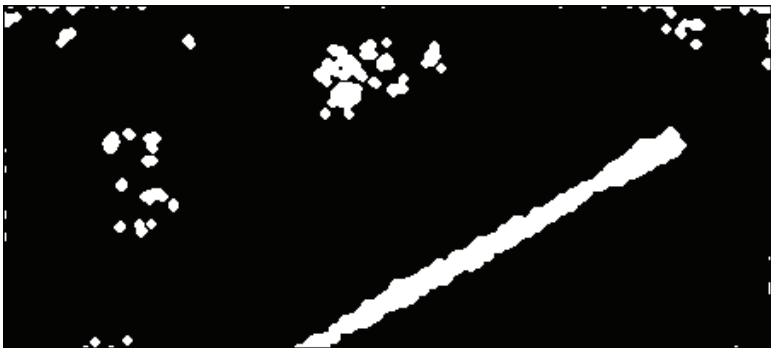


Figure 3.15: Steady flow inverted image

Properties of the binary image are then measured using the `bwconncomp()` and `regionprops()` functions in MATLAB®. The `bwconncomp()` function finds every *connected component*, or region defined by any number of neighboring pixels of value 1, in the image. In this study, an 8-connected neighborhood is used instead of a 4-connected neighborhood. The data structure array, containing information about each connected component, returned by `bwconncomp()` is input directly into the `regionprops()` function. The `regionprops()` function then analyzes each structure in this array to determine a number of properties of each connected component region. In this study, *Area* (pixels), *BoundingBox*, *Image*, *Orientation* (degrees), *MajorAxisLength* (l_{major}, pixels), and *MinorAxisLength* (l_{minor}, pixels) information about each connected component is determined. Then, only the information about the connected component with the largest *Area* is retained, as it is assumed to be the connected component associated with the carbon fiber under observation. The result of the connected component analysis is shown in Figure 3.16.

Figure 3.16: Steady flow output image

To correctly calculate fiber deflection information, a correction must be made for any shift in the current image relative to the reference image created at the beginning of the analysis. This is done by capturing the motion of a pre-determined bright spot present in the image. An interrogation region bounding the full locus of

possible travel of this bright spot for every flow case in the image set is determined, as shown by the blue "+" markers in Figure 3.10. This interrogation region is then binarized for both the reference image created at the beginning of the analysis, and for the current image, according to Equation 3.1. This bright spot image binarization is shown in Figure 3.17.

$$PV_{binarized,i,j} = \frac{PV_{i,j} - \overline{PV}}{PV_{max} - \overline{PV}} \tag{3.1}$$

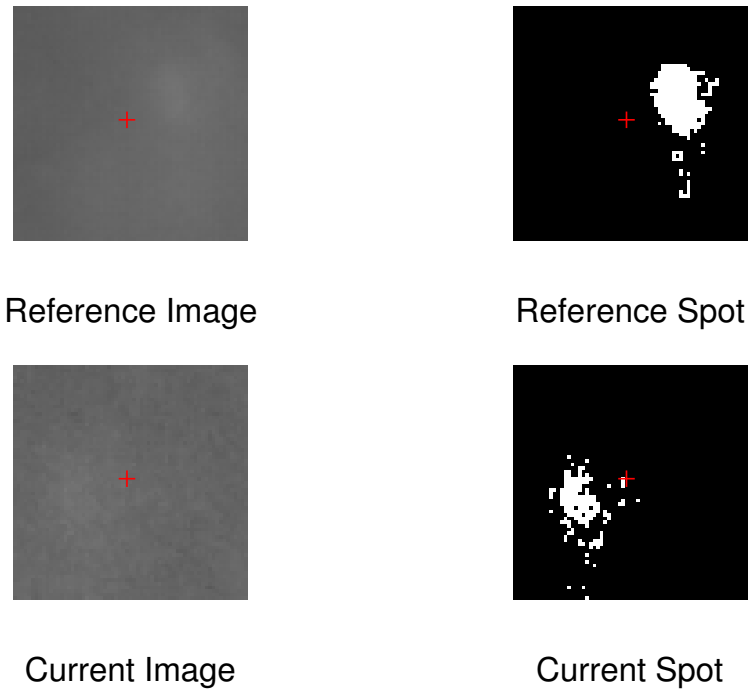


Figure 3.17: Steady flow bright spot image

These binarized images are then processed with the `xcorr2()` function in MATLAB®, which performs a cross-correlation on the image pair. A two-dimensional contour plot of this cross-correlation is shown in Figure 3.18. A sub-pixel resolution location of the maximum of this cross-correlation is determined using a parabolic sub-pixel estimator, described in Equations 3.2 and 3.3, and the difference

between the pixel location of the center of the cross-correlation (corresponding to zero image shift) and the pixel location of the maximum is computed for both the horizontal and vertical dimensions of each image. By knowing the correlation between pixel count and physical length from the calibration process explained earlier on page 29, the physical shift in the current image relative to the reference image is calculated and corrected for in the data reduction.

$$x_{shift} = -x_{brightspot} + x_{max} + \frac{CCR_{x_{max}-1,z_{max}} - CCR_{x_{max}+1,z_{max}}}{2 \cdot CCR_{x_{max}-1,z_{max}} + CCR_{x_{max}+1,z_{max}} - 2 \cdot CCR_{x_{max},z_{max}}} \tag{3.2}$$

$$z_{shift} = z_{brightspot} - z_{max} - \frac{CCR_{x_{max},z_{max}-1} - CCR_{x_{max},z_{max}+1}}{2 \cdot CCR_{x_{max},z_{max}-1} + CCR_{x_{max},z_{max}+1} - 2 \cdot CCR_{x_{max},z_{max}}} \tag{3.3}$$

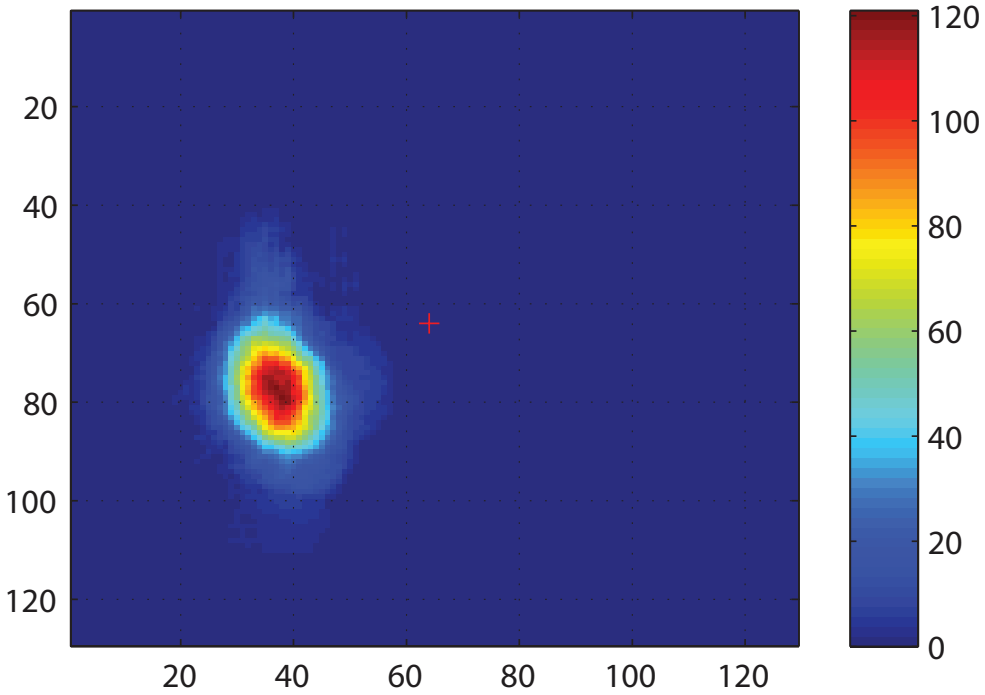


Figure 3.18: Steady flow cross-correlation image

3.4.1.3 Image Data Reduction. Once the raw experimental images are reduced to binary images of the carbon fiber object itself, and an image shift correction is calculated, the total deflection of the tip of the carbon fiber may be computed.

The first item that must be accomplished in the data reduction is the validation of the reduced carbon fiber image set through the rejection of any individual outlier images in the set. During the image analysis process, occasionally the fiber will be split into two separate connected component regions during binarization, which no longer accurately represents the carbon fiber. In this study, this image set validation is accomplished through the comparison of the *Area* of each carbon fiber in the reduced image set that was calculated by the `regionprops()` function in MATLAB®. The algorithm for fiber rejection is composed of two simple comparison operations - unless both conditions in Equation 3.4 are true, the fiber is rejected:

$$\left| \frac{A_f}{\overline{A_f}} - 1 \right| \leq T_{rej,A} \tag{3.4a}$$

$$\frac{l_{major}}{l_{minor}} = AR_f \geq \overline{AR_f} \cdot T_{rej,AR} \tag{3.4b}$$

where A_f and AR_f denote the area and aspect ratio of the analyzed fiber, in pixels; $T_{rej,A}$ and $T_{rej,AR}$ are predetermined rejection threshold values. The values of $T_{rej,A}$ and $T_{rej,AR}$ used in this analysis were 0.25 and 0.85, respectively.

The remaining fiber images are then analyzed to determine the fiber tip location in each image. This is accomplished by recalling the *BoundingBox* information computed by the `regionprops()` function in MATLAB®. The the pixel location from *BoundingBox* that most closely approximates the carbon fiber tip location is recorded for each image. Then for each flow case, an average of the fiber tip pixel

locations is computed. The difference between the average fiber tip pixel location for each flow case and the average fiber tip pixel location for the corresponding zero-flow case is then computed to determine the deflection response (in pixels) of the carbon fiber to each Falkner-Skan boundary layer characterized by each flow case. The lateral and transverse deflection response is then calculated in Equation 3.5, and the total deflection is calculated in Equation 3.6. These deflections are converted to units of length by multiplying by the *Pixel Width* (PW) conversion coefficient resulting from the calibration process discussed on page 29.

$$\Delta_x = ((x_{tip} - x_{shift}) - x_{tip,initial}) \cdot PW \tag{3.5a}$$

$$\Delta_z = ((z_{tip} - z_{shift}) - z_{tip,initial}) \cdot PW \tag{3.5b}$$

$$\Delta_{total} = \sqrt{\Delta_x^2 + \Delta_z^2} \tag{3.6}$$

Only random uncertainty analysis was performed on the reduced data according to Equation 3.7:

$$U_{\Delta x} = \frac{S_{\Delta x} \cdot t_{95,N-1}}{\sqrt{N}} \tag{3.7}$$

where N is the number of valid tip deflection measurements, $S_{\Delta x}$ is the standard deviation of the tip deflection measurements, and $t_{95,N-1}$ is the Student's-t value based on the degrees of freedom, (N-1), and a level of confidence of 95%. The $U_{\Delta z}$ was evaluated similarly. Then the total uncertainty was evaluated using Equation 3.8:

$$U_{total} = \sqrt{U_{\Delta x}^2 + U_{\Delta z}^2} \tag{3.8}$$

Systematic uncertainty, due to the calibration process, focus of the camera, or variance

of fiber location within the depth-of-focus of the optics, are considered negligible with respect to the random uncertainty in this analysis.

3.4.2 Unsteady Image Analysis

Each unsteady carbon fiber image set is processed in a manner similar to the steady image analysis process. A boundary box in the image set is first determined around a specific interrogation region. An average reference image is then constructed from every image during the period of time in the test in which there was a zero mainstream velocity in the wind tunnel. Then, every image in the image set is analyzed using several MATLAB® functions to distinguish the carbon fiber from its background, determine the pixel location of the carbon fiber tip, and correct for any shift in the images relative to the average reference image. Finally, fiber deflection in both the lateral and longitudinal directions for each image are computed from the reduced image data.

3.4.2.1 Initial Analysis Steps. To minimize computation time, a box bounding a specific image interrogation region is first determined, as shown by the blue "+" markers in Figure 3.19. This box is chosen to include the full locus of carbon fiber tip locations over the entire span of mainstream flow velocity images studied for a given carbon fiber. However, care is also taken as much as possible to exclude any anomalies present in the background of the image that may interfere with the proper determination of the carbon fiber tip location, such as the diagonal thermal printer striations found in the background of Figure 3.19.

A reference image is then computed from the average of every image during the period of time in the test in which there was a zero mainstream velocity in the wind

tunnel. This average reference image is used later in a cross correlation to determine and correct for the magnitude of the image shift present in each image analyzed.

To provide a baseline "zero-flow" case similar to the "zero-flow" case from the steady image analysis section, the entire unsteady image reduction discussed below is performed on the average reference image computed in the previous step. This provides the basis image from which fiber tip deflection information is calculated.

Due to large variances in the image brightness of each image set, it became necessary to equalize the image brightness for every image in a given image set. This minimizes the brightness-induced variations introduced during the binarization process. Before the image analysis is performed on any image, the mean intensity of each image in the image set is computed using the `rgb2hsv()` function in MATLAB®.

3.4.2.2 Fiber Image Reduction. The first process performed for each image in the reduction process is the equalization of the image brightness in each image. Using a pair of `while` loops in MATLAB®, image intensity values are incremented or decremented until the average image brightness level computed above is reached. After this equalization is complete, a raw grayscale image, as shown in Figure 3.19, is obtained through the use of the `hsv2rgb()` and `rgb2gray()` functions in MATLAB®.

The remainder of the unsteady image analysis is very similar to that of the steady image analysis presented earlier. Notable differences in the image analysis procedure include changes in the executions of the contrast-limited adaptive histogram equalization and filtering steps, and the overlay of the *regionprops()* output onto the brightness-equalized raw images. An 8x8 array of tiles was used during CLAHE, and a Rayleigh distribution was approximated instead of a uniform distribution. The *thin* filter present in the steady image analysis was omitted in the unsteady image analysis. No rejection criteria was used in the unsteady image analysis.

Figures 3.19 - 3.26 show the entire unsteady image analysis progression. Figure 3.20 presents the result of the CLAHE performed on the unsteady image set. Figures 3.21, 3.22, and 3.23 show the binarized, filtered, and inverted unsteady images, respectively. Figure 3.24 shows the largest connected component in the interrogation region described earlier, as shown in Figure 3.19. Figure 3.25 presents an overlay of this connected component on the raw image from Figure 3.19. Figure 3.26 shows the result of the cross-correlation step performed in this unsteady image analysis.

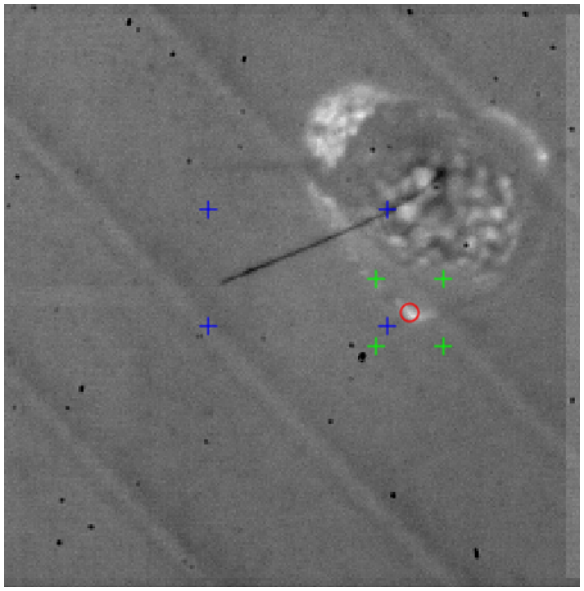


Figure 3.19: Unsteady raw image

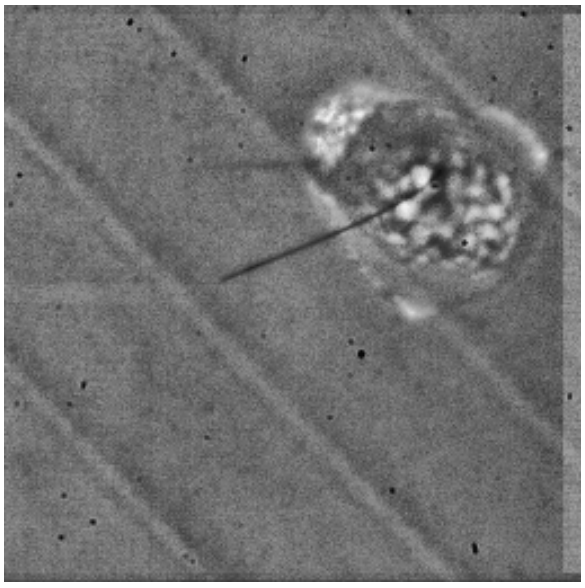


Figure 3.20: Unsteady CLAHE image

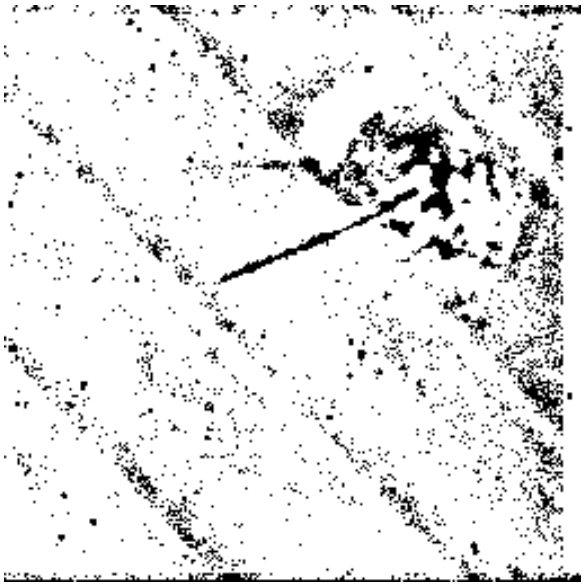


Figure 3.21: Unsteady binarized image

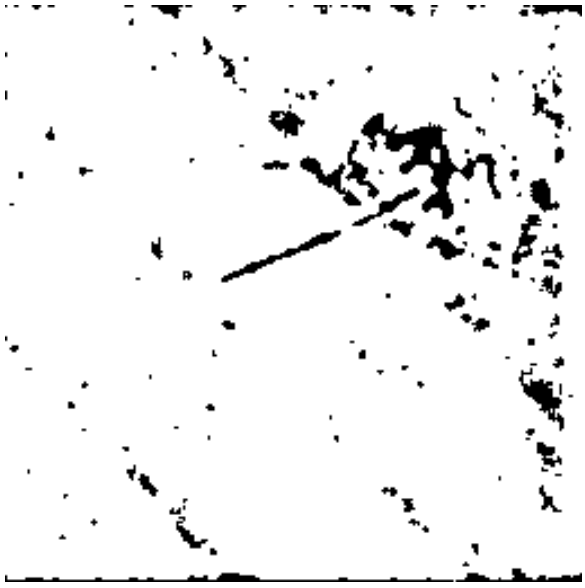


Figure 3.22: Unsteady filtered image

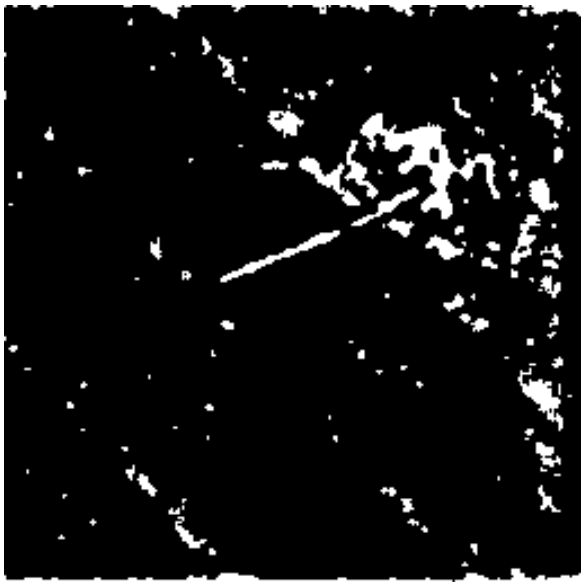


Figure 3.23: Unsteady inverted image

Figure 3.24: Unsteady region properties image

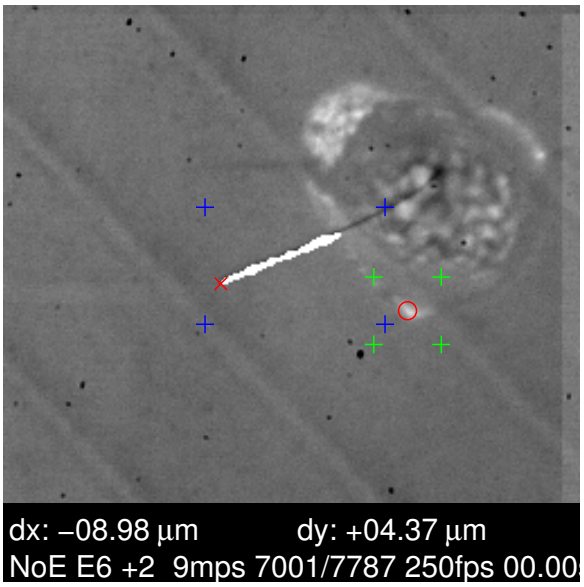


Figure 3.25: Unsteady overlay image

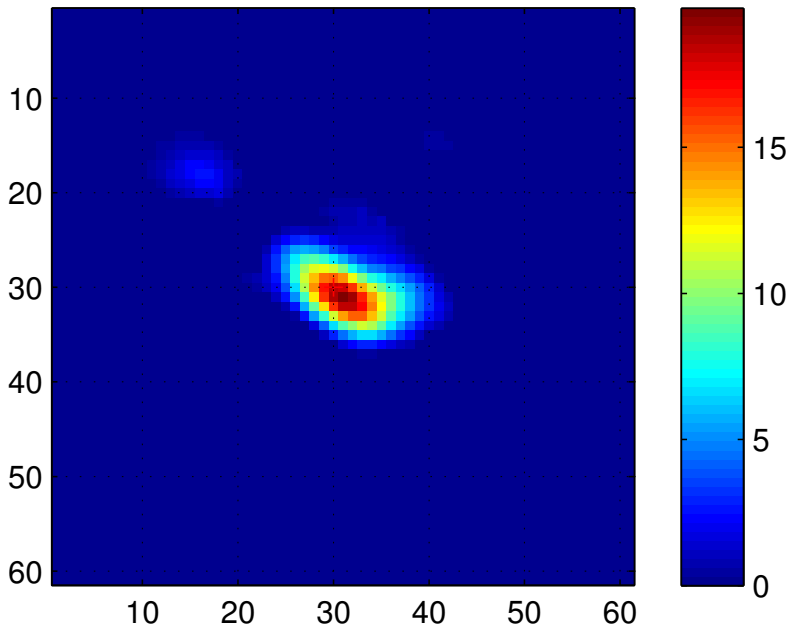


Figure 3.26: Unsteady cross-correlation image

CHAPTER FOUR

Results and Discussion

Using the MATLAB® functions developed earlier, the fiber behavior was investigated. Steady carbon fiber tip deflection response to wind tunnel velocities ranging from 1-10 m/s was calculated and graphed for several carbon fiber hair sensors at both positive and negative angles of attack. Unsteady carbon fiber tip deflection response was also measured at positive and negative plate angles of attack for wind tunnel velocities of 3, 6, and 9 m/s. The nominal values of the angle of attack for both steady and unsteady measurements were +2.5 and -2.5-degrees.

4.1 Steady Hair Sensor Deflection Results

Figure 4.1 presents the total fiber tip deflection in standard length dimensions verses the nominal freestream velocity. The results shown in Figure 4.1 are important for two reasons 1) they demonstrate the low random uncertainties of the measurements, which are on the order of $\pm 10 \mu m$ for all of the measurements and (2) the measurements of E6 (the longest fiber) and E3c (the shortest fiber) exhibit minimal differences relative to the two angles of attack studied. The angle of attack investigation was performed to investigate whether wing-mounted hair sensors could be used to determine aircraft attitude. The measurements presented here indicate that hairs attached aft of the maximum chord on the wing of a small air vehicle will have limited abilities to sense aircraft attitude.

For these hairs, the primary sensing of aircraft attitude is exhibited through unsteady fiber motion. Figure 4.1 shows that measurements of E6 were made for all speeds for negative plate angle of attack; however, measurements of fiber deflection

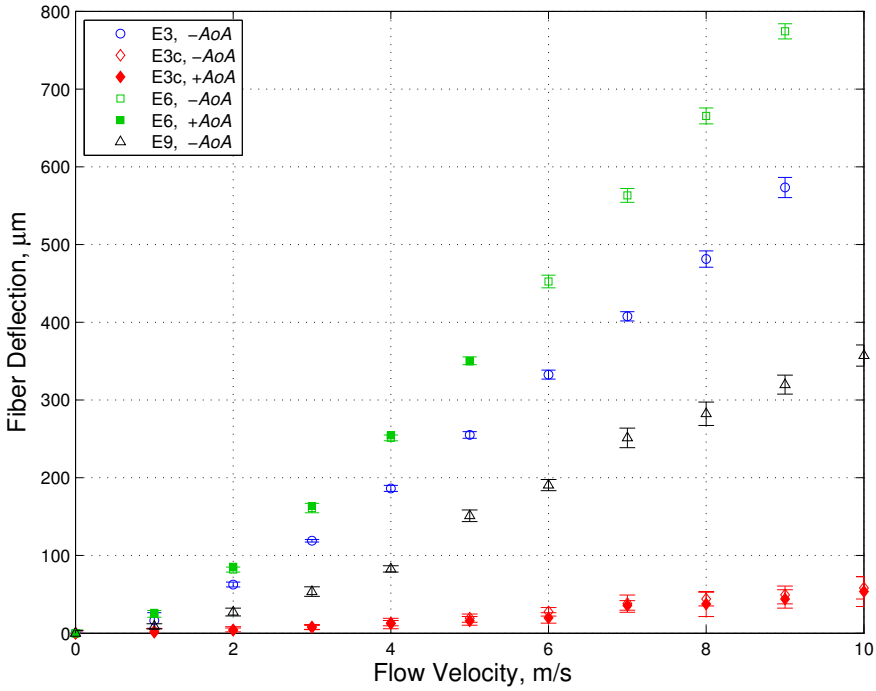


Figure 4.1: Steady deflection of every fiber

are presented only up to 5 m/s for the positive angle of attack. Above 5 m/s, the fiber motion was resonant and unmeasurable with the present optical systems for positive plate angle of attack. While laminar flows should be able to sustain a 2.5° wedge expansion, flow separation caused by the leading edge geometry of the plate is suspected of causing the resonant fiber motion of fiber E6. However, this suspicion cannot be verified without further flow visualization. The fact that the shorter fiber E3c exhibits non-resonant motion for all velocities studied casts doubt on that suspicion.

Table 4.1 presents steady total fiber deflection results for both experimental and model-predicted values. The Δ_{total} columns present the experimental total fiber deflection measurements, $U_{\Delta_{total}}$ columns present the experimental uncertainty of the total fiber deflection values, r_{total} columns present the predictions from the Euler-Bernoulli model of the total fiber deflection discussed earlier on page 25, and *% Error*

columns present the percent error between the experimental total deflection results and the model predictions, as described by Equation 4.1.

$$\%Error = \frac{(\Delta_{total} - r_{total})}{\Delta_{total}} \cdot 100 \tag{4.1}$$

For the longest fiber, E6, and the shortest fiber, E3c, the model predictions fall within the experimental uncertainty bands up to relative deflections of 10%. This limit on the agreement would be expected for an Euler-Bernoulli beam approach which considers forces to be applied to the "non-deflected" shape of the fiber. The model overpredicts the deflection for fibers E3 and E9, but the trends are captured. For fiber E3, an almost constant offset of 20% in the model predictions is observed. For fiber E9, the average percent error is 32%.

Figure 4.2 is a graphical representation of the experimental measurements and model predictions as presented in Table 4.1, normalized by the lengths of each carbon fiber. As stated above, the maximum percent difference between the model prediction and the measurements is 32% for fiber E9, which is the only fiber that points "upstream" in its unloaded condition. While the beam approach used should be able to handle the configuration of E9, this fiber has the largest out of focal-plane fiber movement over the range when the prediction difference is greatest for relative fiber deflections less than 10% at the 4 m/s case. The agreement of the model predictions with experimental measurements demonstrated in Figure 4.2 demonstrates that the Euler-Bernoulli beam approach, while simplistic, is sufficient for preliminary sensor design calculations predicting either fiber tip deflections or fiber base forces and moments. Because the trends are captured and because the focus of this study is capturing relative deflections and relative force changes related to unsteady events, the agreement is considered very good.

Table 4.1: Steady Fiber Deflection Percent Error

Flow Velocity (m/s)	Δ_{total} (μm)	$U_{\Delta_{total}}$ (μm)	r_{total} (μm)	% Error	Δ_{total} (μm)	$U_{\Delta_{total}}$ (μm)	r_{total} (μm)	% Error
	Fiber E3, $-2.5°$ AoA				Fiber E9, $-2.5°$ AoA			
1	16	10.1	21	28.5	9	3.3	11	22.0
2	63	3.1	72	15.0	27	5.1	39	43.1
3	119	1.7	139	17.2	54	6.1	78	46.4
4	186	3.9	217	16.4	83	4.1	125	51.3
5	255	4.2	301	18.0	151	7.4	177	17.2
6	333	5.7	391	17.6	191	7.2	233	22.1
7	408	6.0	486	19.3	251	12.6	292	16.1
8	481	10.5	586	21.7	282	15.0	353	25.2
9	573	12.9	689	20.2	320	12.2	418	30.6
10	648	17.6	797	22.9	357	13.7	484	35.5
	Fiber E3c, $-2.5°$ AoA				Fiber E3c, $+2.5°$ AoA			
1	3	3.5	1	69.7	1	4.7	1	36.6
2	4	2.4	3	27.5	4	4.6	3	22.8
3	8	2.6	7	11.9	8	3.3	6	19.0
4	13	3.4	11	12.8	13	6.7	11	12.6
5	19	5.2	17	12.9	16	5.5	16	3.1
6	28	5.5	23	15.9	20	6.8	23	14.8
7	38	11.0	30	21.1	36	6.2	29	17.6
8	44	8.7	37	14.7	38	16.1	37	2.0
9	49	11.5	45	8.1	44	11.8	45	1.4
10	58	14.2	53	8.4	54	19.3	53	1.5
	Fiber E6, $-2.5°$ AoA				Fiber E6, $+2.5°$ AoA			
1	25	4.4	26	2.8	26	2.8	25	6.7
2	82	3.2	87	6.5	85	1.9	85	0.3
3	161	6.1	169	4.7	164	2.0	166	1.7
4	251	3.7	262	4.2	255	3.0	260	2.1
5	350	5.0	363	3.7	350	3.6	362	3.5
6	452	8.2	472	4.3	—	—	—	—
7	563	8.9	587	4.2	—	—	—	—
8	666	10.2	707	6.2	—	—	—	—
9	774	9.8	832	7.5	—	—	—	—
10	872	30.2	962	10.2	—	—	—	—

* Deflection measurements are rounded to the nearest whole number.

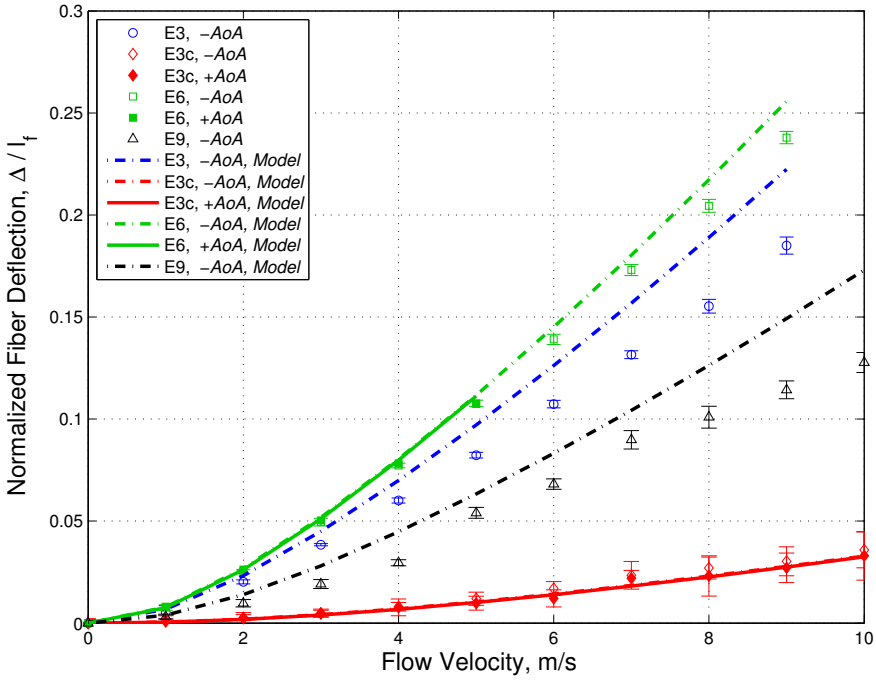


Figure 4.2: Comparison of model prediction to experimental results of steady normalized deflection of every fiber

Additionally, Figure 4.2 demonstrates that the model predictions indicate very little difference between the predictions of fiber deflection between the positive and negative plate-angle-of-attack cases for the E3c and E6 fibers. Thus, just as the experimental measurements indicated, the Euler-Bernoulli model indicates that the fiber and plate geometries studied here are not sensitive to the changes in plate angle of attack.

To further explore the fiber behavior, the relative fiber deflections were compared to the changes in the fiber height relative to the local boundary layer thickness, δ, where δ is the value of y as calculated from the Falkner-Skan profile from Equations 2.5 and 2.6 when f' equals 0.99. Figure 4.3 presents the fiber-height to boundary-layer thickness ratios (h_f/δ) for each case presented.

Figure 4.4 presents the relative fiber deflections versus the h_f/δ. Figure 4.4 demonstrates that the data and the model predictions essentially collapse. For a

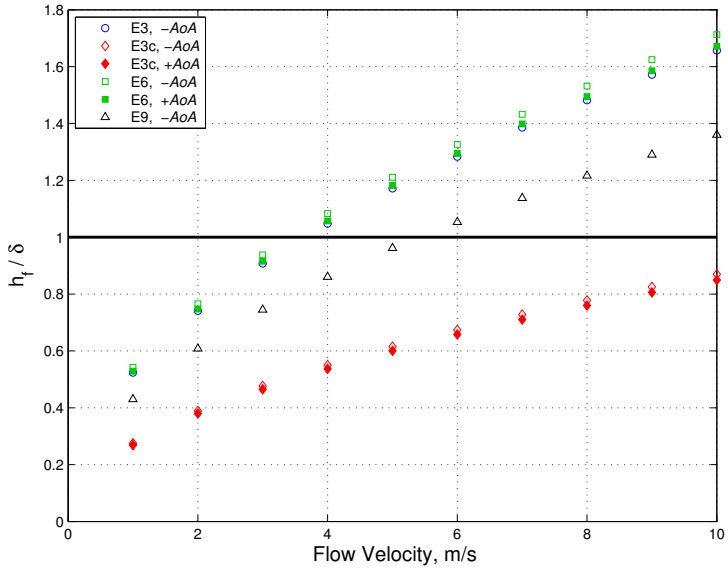


Figure 4.3: Fiber height relative to boundary layer thickness

given (h_f/δ), differences exist between the different fibers accounting for changes in fiber length, fiber orientation, distance downstream of the plate leading edge, and placement behind other fibers. However, these differences are subtle, and the dominant factor affecting the relative tip deflection is the relative penetration depth of the fiber into the boundary layer.

Finally, Fibers E6 and E9 were placed directly downstream of E3. Figure 4.1 demonstrates similar levels of agreement with the beam model, if not better, for E6 and E9 than for E3 and E3c, and as noted previously, Figure 4.4 shows that all of the relative fiber deflection measurements essentially collapse. Thus, for the given fiber array and given the present optical measurement system, fiber placement downstream of other fiber elements did not significantly affect the fiber response to steady flows.

4.2 Unsteady Hair Sensor Deflection Results

Because no simultaneous, time-resolved flow measurements were taken during the unsteady-flow tests, no quantitative correlation between unsteady flow and unsteady fiber response was developed. A qualitative assessment of fiber response to

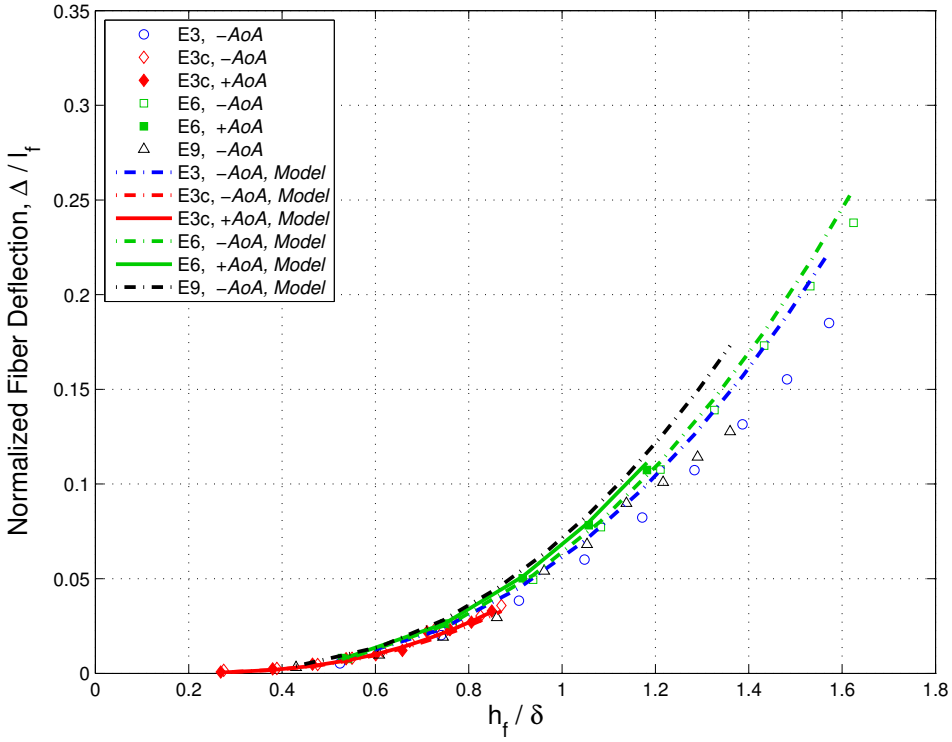


Figure 4.4: Comparison of model prediction to experimental results of fiber deflection versus fiber height relative to boundary layer thickness

unsteady flow is presented instead. Figure 4.5 shows a screenshot view of the unsteady fiber motion analysis video generated using the analysis presented in Chapter Three and the code found in Appendices B and C. Unsteady carbon fiber tip deflection response to lateral gusting perturbations in the fluid flow is apparent in Figure 4.5. The time tracking chart of tip displacement (in μm) versus time at the top of Figure 4.5 shows fiber tip deflection as measured in both dimensions of the image. This chart clearly demonstrates four significant events during the unsteady flow tests:

1. The wind-tunnel flow speed ramp-up to 9 m/s (which occurs essentially linearly at 1 m/s per second), starting approximately at $time = 4s$ and ending approximately at $time = 13s$,

2. A series of three discrete lateral gusts at approximately $17s \leq time \leq 21s$,

3. Two sinusoidal gusting conditions with the amplitude of the second set being larger than the first at approximately $21s \leq time \leq 27s$, and

4. A second series of three discrete gusts where the first gust was very large, beginning approximately at $time = 27s$.

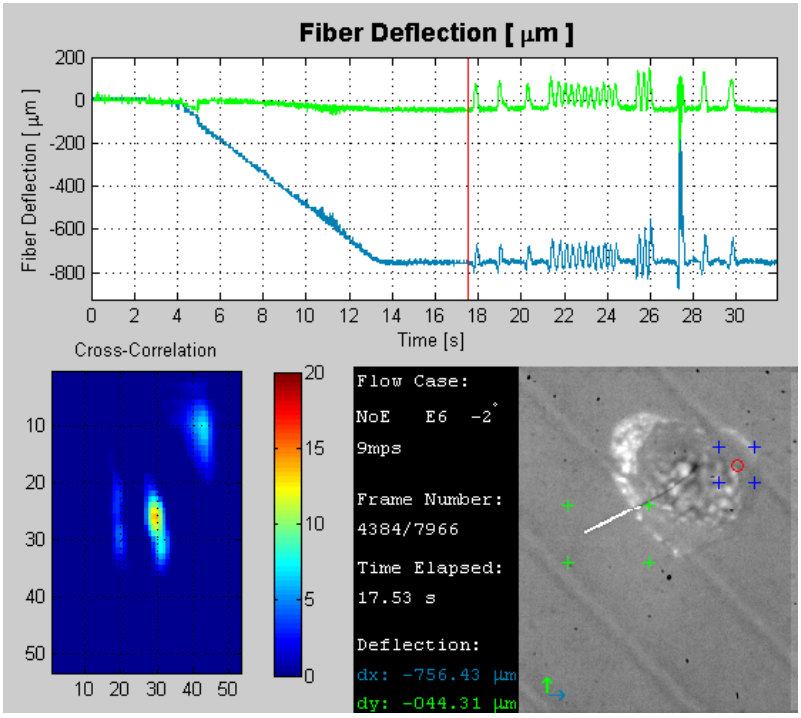


Figure 4.5: Screenshot of unsteady fiber motion analysis video

Several of the events listed above were repeated for each test performed on fiber E6. Each test consisted of a flow speed ramp-up, at least one series of discrete lateral gusts, and at least one sinusoidal gusting condition. The manual control of the flap-induced flow perturbations did not allow for consistent, repeatable gusting conditions during the unsteady tests. [Note: data sets from each test in Figures 4.6 - 4.9 are shifted in time to provide a collapsed set of data referenced to a single important event in the figure]. Figure 4.6 shows a time chart of total carbon fiber deflection response, in μm, to unsteady lateral gusting perturbations introduced to flows at 3,

6, and 9 m/s at a plate angle of attack of $-2.5°$. Figure 4.7 shows the same set of carbon fiber deflection responses for a plate angle of attack of $+2.5°$. Immediately apparent in Figure 4.7 is the resonant nature of the fiber during the 9 m/s test case. During the steady part of the tests, the fiber is resonant; however, the fiber ceases to be resonant during lateral gusting conditions. Figure 4.8 shows a comparison of the E6 fiber response to these gusting perturbations for both $+2.5°$ and $-2.5°$ plate angles of attack. It is clear from the figure that the fiber resonance ceases during times of fiber response to lateral gusts. Figure 4.9(a) and Figure 4.9(b) show more detailed graphs of this behavior during multiple discrete gusts and the sinusoidal gust. While the "mean deflection" values shown in the graphs of Figure 4.9 are slightly different for the two test cases shown, similar unsteady fiber deflection response amplitudes is observed.

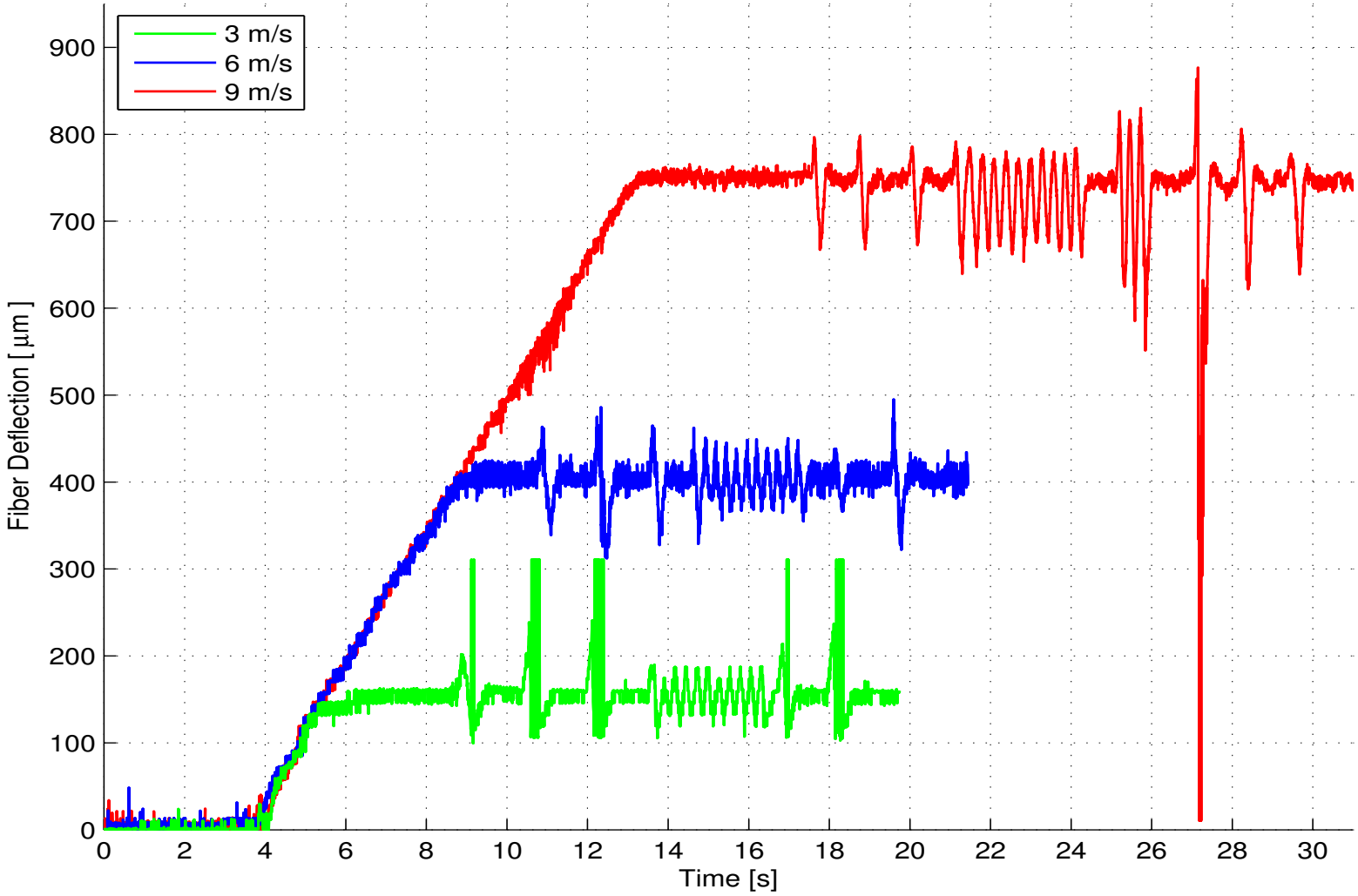


Figure 4.6: Total deflection response of fiber E6 to unsteady flows (-2.5 degree plate angle of attack)

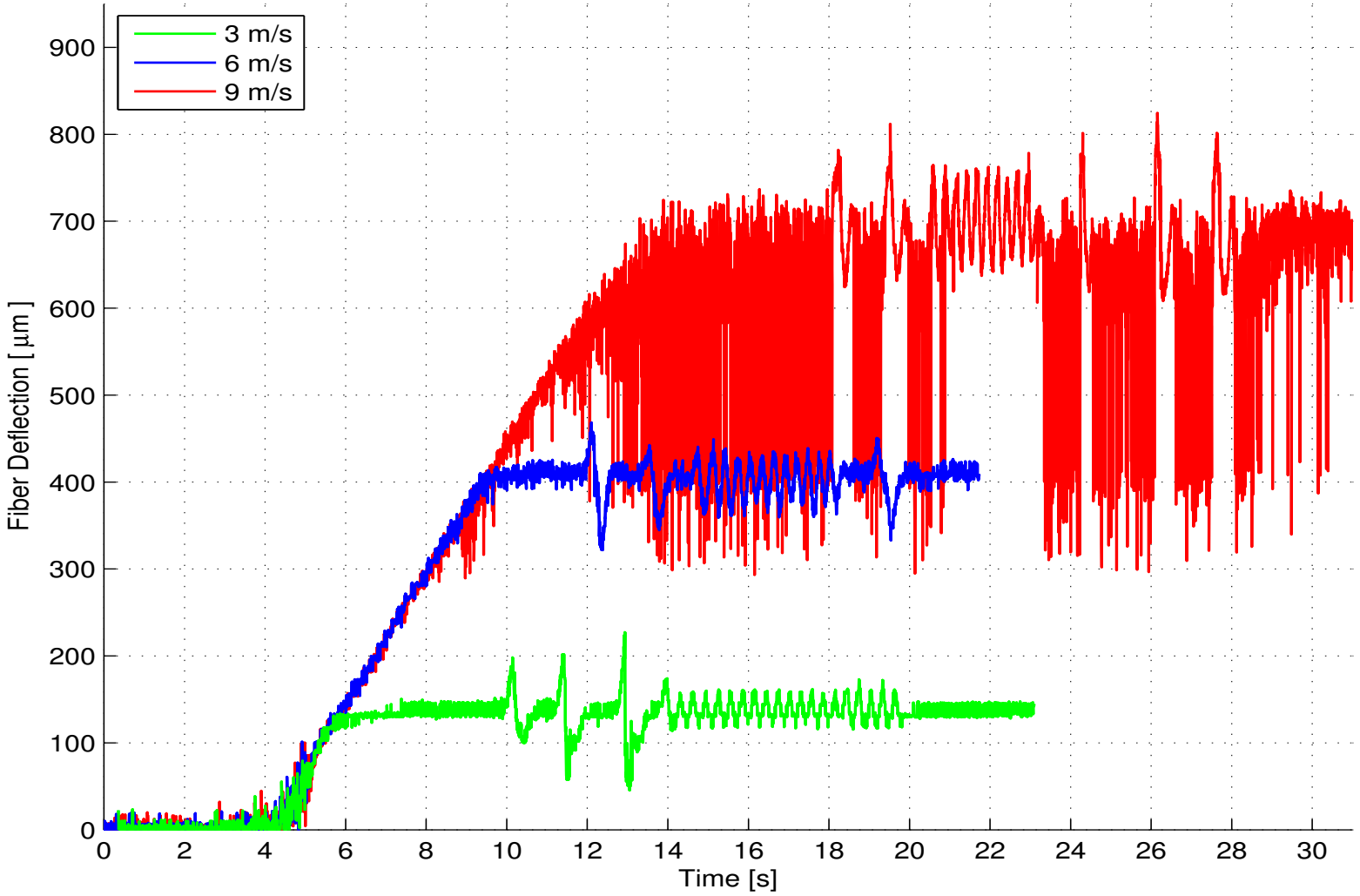


Figure 4.7: Total deflection response of fiber E6 to unsteady flows (+2.5 degree plate angle of attack)

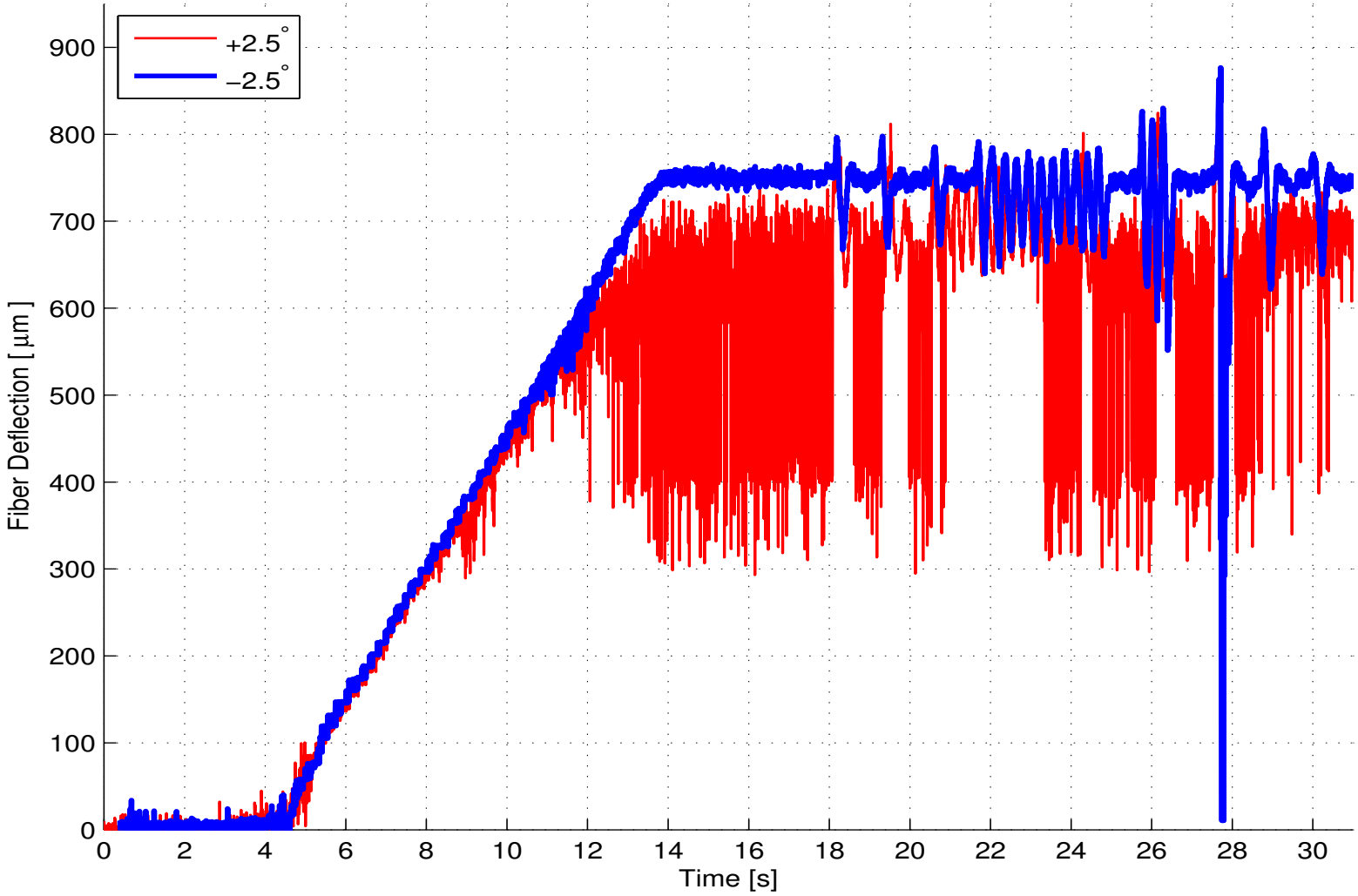
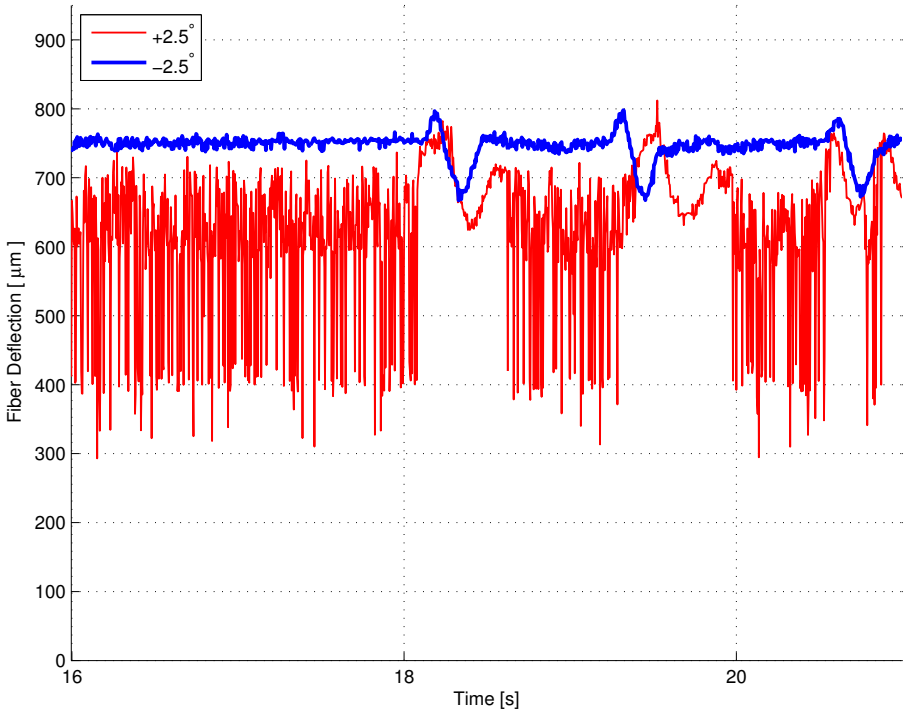
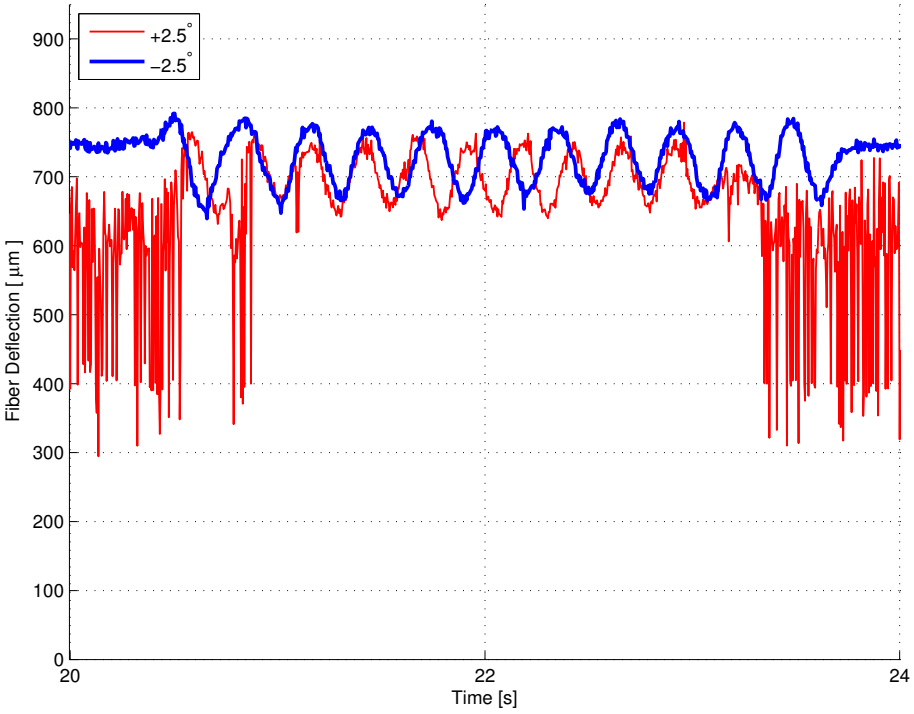


Figure 4.8: Comparison of fiber deflection response to 9 m/s unsteady flow with ±2.5° plate angle of attack



(a)



(b)

Figure 4.9: Comparison of fiber deflection response with ±2.5° plate angle of attack to (a) a series of 3 lateral gusts, and (b) an oscillating gust

CHAPTER FIVE

Conclusions and Future Work

In this chapter, the most pertinent results regarding artificial hair sensor response in Falkner-Skan boundary layers are summarized. Suggestions for the improvement of the current experimental procedure and data reduction are provided. Finally, recommendations for future work are also presented.

5.1 Summary of Current Work

Passive, long-aspect-ratio carbon fiber hairs were glued to a small flat plate and exposed to steady flows ranging from 1-10 m/s. The plates were oriented to produce Falkner-Skan boundary layers, or wedge flows, for two plate angles of attack. The deflection of the carbon-fiber hairs was measured using inspection microscope optics and a digital camera. An Euler-Bernoulli beam model was also developed to predict the deflection of non-wall orthogonal hairs or micro-pillars. The primary observations of the study were:

1. The experimental measurements of the relative fiber deflection agreed well with the Euler-Bernoulli model presented for relative deflections less than 10%, and agreement was still reasonable for relative deflections up to 25%.

2. Relative fiber displacement was shown to be sensitive to the local velocity and consequently to the height of the fiber relative to the local boundary-layer height.

3. The experimental measurements and the model predictions show insensitivity to the change in the plate angle of attack except when the increased angle of attack produces resonant fiber motion.

4. Fiber displacement response to unsteady gusts was measured, but because no simultaneous time-resolved flow measurements were taken, no correlation between unsteady flow condition and fiber behavior was developed.

5. Resonant behavior was observed in fiber E6 during the 9 m/s, $+2.5^\circ$ plate angle of attack unsteady flow case. When compared to the deflection response of the corresponding -2.5° plate angle of attack flow case, it is clear that the resonant behavior ceases only in response to a gusting condition. Fiber deflection response amplitudes of similar magnitude between these two cases were measured.

5.2 Recommendations for Improvement of Current Methodology

While useful information was gathered using the procedure outlined in this thesis, there are several areas in which improvements may be made. If hair geometries closer to wall-orthogonality are desired, a two-step carbon fiber wax embedding process may be developed to reduce the variance in fiber geometry and attachment angle. A carbon fiber may first be embedded in a secondary piece of paraffin wax to remove the difficulty of positioning the fiber using forceps, and then positioned into the primary piece of paraffin wax on the tip of the micromanipulator positioning system at an angle more conducive to wall-orthogonality once glued to the testing surface. Investigations into improving the gluing process itself may also be warranted to reduce image background interference and allow for more distinguishable fibers.

Additionally, time-resolved local fluid velocity measurements using hot-wire anemometry or a similar measurement technique, synchronous with imaging of fiber response to unsteady boundary-layer perturbations, will be needed to fully characterize the perturbations and corresponding hair sensor response to unsteady flows.

Automated introduction of flow perturbations will also be required to create repeatable gusting conditions in the wind tunnel. With this automation, fiber sensitivity to unsteady gusts can be properly quantified through the perturbation of a single variable at a time, without the inadvertent variations in other variables that come with manual control of the gusting conditions.

5.3 Recommendations for Future Work

A recent study by Narvaez and McClain [48] has examined the downstream effects of arrays of obliquely aligned surface elements and their usefulness for flow tailoring. Flow at the trailing edge of these obliquely aligned elements is essentially forced into a local Kutta condition, imparting some lateral momentum on the flow based on the angle of alignment of the elements. In this way, flow over an aerodynamic surface may be "tailored" by arrays of elements with a known angle of alignment to provide some downstream benefit. Relevant to the continuation of the current effort, additional studies could examine combinations of these obliquely aligned elements and hair sensor arrays. An example test plate with these oblique elements is shown in Figure 5.1. As flow velocities increase, the longitudinal forces acting on the hair sensors increase. This increased longitudinal force implies that the hair sensors will become less sensitive to lateral gusts. Through the addition of upstream oblique elements, the longitudinal flow experienced by the hair sensors will be impeded, allowing for greater sensitivity to lateral gusting conditions.

The experimental methodology presented in Chapter 3 has already been used to create steady-flow image sets of fiber deflection for two plates with combinations of oblique element arrays and hair sensors. Each plate contained elements with either 0° or 10° alignment angles, and each plate was exposed to steady flows ranging from 1-10 m/s at plate angles of attack of $+2.5^\circ$ or -2.5°. The image analysis presented

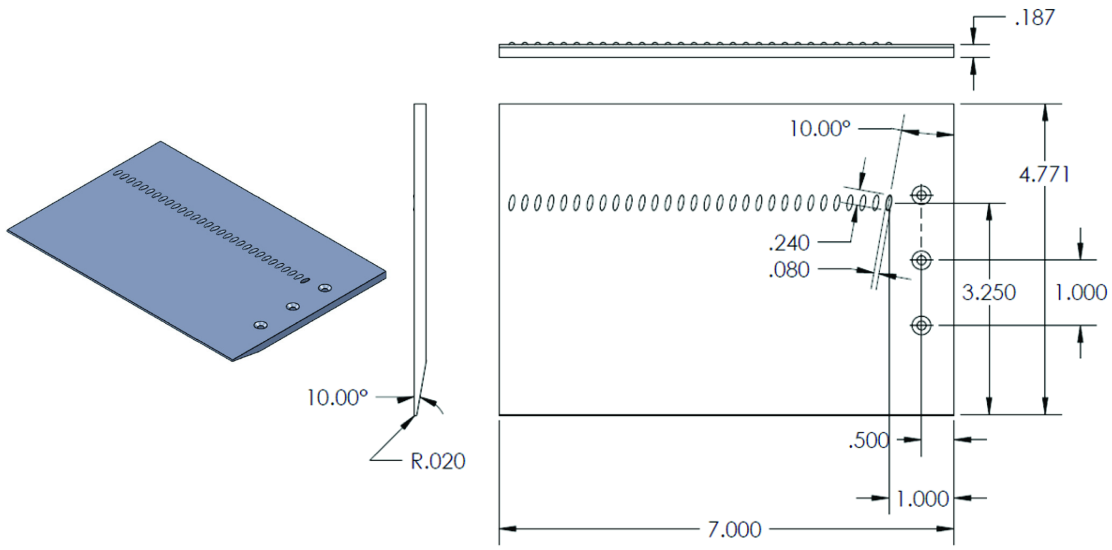


Figure 5.1: Test plate containing a distribution of obliquely-aligned elements (dimensions in inches)

in this thesis to reduce fiber deflection information from the steady flow image sets for these plates is ongoing. The results will be useful in determining the effectiveness of oblique elements at increasing hair sensor sensitivity to lateral gusts.

APPENDICES

APPENDIX A

Steady Fiber Analysis MATLAB® Code

```
%% Function Name:   steadyAnalysis    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Author:          Lance Case
%  Date Created:    02.06.2012
%  Last Modified:   03.14.2012
%
%  Purpose:     This file serves as the main executable for the
%               MATLAB steady fiber image analysis program.
%
%  Inputs:      varargin -  Traditionally-optional MATLAB function
%                           arguments
%
%               varargin{1} -   The text file containing all of the
%                               function variables to be used
%                               during analysis.  If not included,
%                               the function will instruct the UI
%                               to prompt the user for the variable
%                               file.
%
%               Variables included in this text file:
%                   ImgBaseDir      - The base directory containing
%                                     the image set
%                   ImgInBase       - Base string of input image
%                                     filename
%                   ImgInType       - Input image file type
%                   AnlysImgOut     - Analyzed, output image file
%                                     type
%                   ImgOutBase      - Base string of output image
%                                     filename
%                   ImgOutType      - Filetype of output image
%                   FlowPlate       - Name of the test plate used
%                   FlowAtkAng      - Angle of Attack of test plate
%                   DynViscPas      - Dynamic viscosity of the
%                                     fluid in the test section
%                   DensKgpms       - Density of the fluid in the
%                                     test section
%                   BLayerHgt       - Boundary layer height
%                                     metric convention (0.99)
%                   FlowFiber       - Name of the test element
%                                     under observation
%                   FiberXposmm     - Streamwise location of test
%                                     element under observation(mm)
%                   FiberYposmm     - Deprecated.
%                   FiberHgtmm      - Height of the test fiber (mm)
%                   BoundBoxXS      - X-coordinate for left side
%                                     of fiber tip interrogation
%                                     region boundary box
%                   BoundBoxXF      - X-coordinate for right side
%                                     of fiber tip interrogation
%                                     region boundary box
%                   BoundBoxYS      - Y-coordinate for top side
%                                     of fiber tip interrogation
%                                     region boundary box
%                   BoundBoxYF      - Y-coordinate for bottom side
%                                     of fiber tip interrogation
```

```
%                                region boundary box
%                 BSpotPosX      - X-coordinate for bright spot
%                                  interrogation region boundary
%                                  box
%                 BSpotPosY      - Y-coordinate for bright spot
%                                  interrogation region boundary
%                                  box
%                 BSpotPosSz     - Pixel size of bright spot
%                                  interrogation region boundary
%                                  box
%                 CCRclipMax     - Bright spot maximum value for
%                                  use during Cross-Correlation
%                 CCRclipMin     - Bright spot minimum value for
%                                  use during Cross-Correlation
%                 CCRGreyFig     - Boolean dictating whether the
%                                  cross-correlation figure
%                                  should be grayscale
%                 CLAHEtiles     - Number of tiles to use during
%                                  CLAHE
%                 RunCLAHE       - Boolean controlling the CLAHE
%                 CLAHEdistr     - Type of image histogram
%                                  distribution to approximate
%                                  during CLAHE
%                 Filtering      - String determining type of
%                                  image filtering used during
%                                  filtering step.
%                                  (OPTIONAL, MULTIPLE ALLOWED)
%                 BinThresh      - Threshhold integer used
%                                  during image binarization
%                                  (MULTIPLE ALLOWED, SEPARATE
%                                   VALUES FOR EACH MAINSTREAM
%                                   VELOCITY IMAGE SET)
%                 SkiptoVeloc    - Optional variable used during
%                                  analysis setup or
%                                  troubleshooting.  Allows the
%                                  analysis algorithm to skip to
%                                  the necessary velocity image
%                                  set
%                 BSpotFilter    - Allows the cross-correlation
%                                  to discard incorrect/invalid
%                                  areas in the bright spot
%                                  image
%
%                                  EXAMPLE 1:
%                                       BSpotFilter<tab>06,X>110
%                                          -discards any image
%                                           information for the
%                                           6mps test case where
%                                           the X position in
%                                           the image is greater
%                                           than 110px
%                                  EXAMPLE 2:
%                                       BSpotFilter<tab>01,Y<30
%                                          -discards any image
```

```
%                                       information for the
%                                       1mps test case where
%                                       the Y position in
%                                       the image is less
%                                       than 30px
%
%                   ShowBounds      - Boolean used during setup or
%                                     troubleshooting.  Creates a
%                                     figure showing the location
%                                     of the image bounding boxes
%                                     specified in the variables
%                                     file.
%                   ShowEachBnd     - Boolean used during setup or
%                                     troubleshooting. Performs the
%                                     same function as 'ShowBounds'
%                                     for each velocity present
%                   SaveEachBnd     - Boolean controlling the save
%                                     of the figure created if
%                                     the 'ShowEachBnd' variable is
%                                     set to '1'
%                   OnlyBounds      - Boolean used to determine
%                                     whether the image analysis
%                                     program should stop execution
%                                     after displaying a figure
%                                     showing the boundary box
%                                     location
%                   ShowFigRaw      - Boolean controlling display
%                                     of raw image figure
%                   ShowFigAHE      - Boolean controlling display
%                                     of CLAHE figure
%                   ShowFigBin      - Boolean controlling display
%                                     of binarization figure
%                   ShowFigFil      - Boolean controlling display
%                                     of filtered image figure
%                   ShowFigInv      - Boolean controlling display
%                                     of inverted image figure
%                   ShowFigRP       - Boolean controlling display
%                                     of region props figure
%                   ShowFigRej      - Boolean controlling display
%                                     of rejected/passed fiber
%                                     image figure
%                   ShowFigOvr      - Boolean controlling display
%                                     of overlay figure
%                   ShowFigCCR      - Boolean controlling display
%                                     of cross-correlation figure
%                   SaveFigRaw      - Boolean controlling saving
%                                     of raw image figure
%                   SaveFigAHE      - Boolean controlling saving
%                                     of CLAHE figure
%                   SaveFigBin      - Boolean controlling saving
%                                     of binarization figure
%                   SaveFigFil      - Boolean controlling saving
%                                     of filtered image figure
%                   SaveFigInv      - Boolean controlling saving
```

```
%                                   of inverted image figure
%                   SaveFigRP       - Boolean controlling saving
%                                   of region props figure
%                   SaveFigRej      - Boolean controlling saving
%                                   of rejected/passed fiber
%                                   image figure
%                   SaveFigOvr      - Boolean controlling saving
%                                   of overlay figure
%                   SaveFigCCR      - Boolean controlling saving
%                                   of cross-correlation figure
%                   ShowGphFSBL     - Boolean controlling display
%                                   of the Falkner-Skan boundary
%                                   layer profile based on the
%                                   plate angle of attack
%                   ShowGphDefl     - Boolean controlling display
%                                   of the fiber deflection graph
%                   ShowGphTDEr     - Boolean controlling display
%                                   of the total fiber deflection
%                                   graph (with errorbars)
%                   ShowGphNDef     - Boolean controlling display
%                                   of the normalized fiber
%                                   deflection graph
%                   ShowGphRAng     - Boolean controlling display
%                                   of the fiber relative
%                                   deflection angle graph
%                   RejThresh       - Rejection threshold value
%                                   used to reject fibers based
%                                   on their deviation from the
%                                   mean fiber area (in image
%                                   pixel count)
%                                   (This is T.rej,A in the
%                                    thesis)
%                   ARrejThresh     - Rejection threshold value
%                                   used to reject fibers based
%                                   on their deviation from the
%                                   mean fiber aspect ratio
%                                   (This is T.rej,AR in the
%                                    thesis)
%
%                   Variables in this text file should be formatted
%                   as follows ('<tab>' denotes tab character):
%
%                   #BOF
%                   VariableNameOne<tab>VariableValueOne
%                   VariableNameTwo<tab>VariableValueTwo
%                   ========================================
%                   Comments can be included on separate lines
%                   ========================================
%                   VariableNameThree<tab>VariableValueThree
%                   ...
%                   #EOF
%
%  Outputs:         The variable A, a structure containing all of
%                   the pertinent information about the current
```

```
%                   image analysis
%
%  Ignore MATLAB Errors:
%#ok<*ST2NM,*ASGLU,*AGROW,*ISMT>
%#ok<*NOPRT>
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A = steadyAnalysis(varargin)
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        MATLAB Program Inititalization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Accept variables file as an optional input
if length(varargin)>0
    varFile = varargin{1};
else
    varFile = '';
end

% Clear and format the screen
% clearvars -except imageParentDirectory;
% clc;
format compact;

% Remove excess grey borders from MATLAB figures
iptsetpref('ImshowBorder','tight');

% Supress MATLAB Warnings:
warning off all;

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            Constant Initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('Constant Initialization\n');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Track the current directory
A.CurrentDirectory = pwd;

% Pixel Width
PW = (2/32)/(973-256)*25400;    % um per pixel  (Scienscope)

% Falkner-Skan Variables
etamax   = 6.5;
etasteps = 1000;


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            Variable Initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('Variable Initialization\n');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialize variable-value arrays before reading input file
```

```
A.ImageDirectory = A.CurrentDirectory;
A.BinThresh = [];
A.Filtering = {};
A.IAfType = {};
A.Output = {};
A.CCR.grey = 0;
A.CCR.BSFilter = [];
zipFileList = {};

% Initialize other variables
A.Display.Fig.OnlyBnd = 0;
A.Display.Fig.Ebd = 0;
A.Display.Fig.Save.Ebd = 0;
A.Display.Gph.TDEr = 0;
aRejectionThreshhold = [];
ARrejectionThreshhold = 0.85;

A.CLAHE.Distr = 'uniform';
skipToVelocity = 0;
areaRejectMin = 0.900;  % Default Fiber Rejection Criteria
areaRejectMax = 1.125;  % Default Fiber Rejection Criteria

% Variables file containing image reduction information
if isempty(varFile)
    [A.varFile,dummy,nextdummy] = uigetfile({'*.txt','*.dat'});
    A.varFile = sprintf('%s\\%s',dummy,A.varFile);
else
    A.varFile = varFile;
end

% Start the timer
start = clock;

% Open the variables input file
variablesFile = fopen(A.varFile,'r');

% Read the variables from the input file
while ~feof(variablesFile)
    newVar = fgets(variablesFile);
    a = textscan(newVar,'%s%s','delimiter','\t'); b=a{1}; c=a{2}; d ...
       = c{1};

    % Set Variables Related to Image File Input and Output
    if strcmp(b,'ImgBaseDir'),  A.ImageDirectory = d; end;
    if strcmp(b,'ImgInBase'),   A.Input{1}       = d; end;
    if strcmp(b,'ImgInType'),   A.Input{2}       = d; end;
    if strcmp(b,'AnlysImgOut'), A.IAfType{length(A.IAfType)+1}= d; end;
    if strcmp(b,'ImgOutBase'),  A.Output{1}      = d; end;
    if strcmp(b,'ImgOutType'),  A.Output{length(A.Output)+1} = d; end;

    % Set Variables Related to the Current Flow Case
    if strcmp(b,'FlowPlate'),   A.FlowCase{1}   = d; end;
    if strcmp(b,'FlowAtkAng'),  A.FlowCase{3}   = d; end;
    if strcmp(b,'DynViscPas'),  A.Air.DynVisc   = str2double(d); end;
```

```
if strcmp(b,'DensKgpms');   A.Air.Density   = str2double(d); end;
if strcmp(b,'BLayerHgt'),   u_delta         = str2double(d); end;

% Set Variables Related to the Current Fiber
if strcmp(b,'FlowFiber'),   A.FlowCase{2}   = d; end;
if strcmp(b,'FiberXposmm'), A.Fiber.Pos(1)  = str2double(d); end;
if strcmp(b,'FiberYposmm'), A.Fiber.Pos(2)  = str2double(d); end;
if strcmp(b,'FiberHgtmm'),  A.Fiber.Height  = str2double(d); end;

% Set Variables Related to Image Analysis
if strcmp(b,'BoundBoxXS'),  A.BoundBox(1)   = str2num(d); end;
if strcmp(b,'BoundBoxXF'),  A.BoundBox(2)   = str2num(d); end;
if strcmp(b,'BoundBoxYS'),  A.BoundBox(3)   = str2num(d); end;
if strcmp(b,'BoundBoxYF'),  A.BoundBox(4)   = str2num(d); end;
if strcmp(b,'BSpotPosX'),   A.CCR.BSLoc(1)  = str2num(d); end;
if strcmp(b,'BSpotPosY'),   A.CCR.BSLoc(2)  = str2num(d); end;
if strcmp(b,'BSpotPosSz'),  A.CCR.BSLoc(3)  = str2num(d); end;
if strcmp(b,'CCRclipMax'),  A.CCR.clip(1)   = str2num(d); end;
if strcmp(b,'CCRclipMin'),  A.CCR.clip(2)   = str2num(d); end;
if strcmp(b,'CCRGreyFig'),  A.CCR.grey      = str2num(d); end;
if strcmp(b,'CLAHEtiles'),  A.CLAHE.NTiles  = str2num(d); end;
if strcmp(b,'RunCLAHE'),    A.CLAHE.Run     = str2num(d); end;
if strcmp(b,'CLAHEdistr'),  A.CLAHE.Distr   = d; end;
if strcmp(b,'Filtering'),   A.Filtering{length(A.Filtering)+1} ...
                                    = d; end;
if strcmp(b,'BinThresh'),   A.BinThresh(length(A.BinThresh)+1) ...
                                    = str2num(d); end;
if strcmp(b,'SkiptoVeloc'), skipToVelocity  = str2num(d); end;
if strcmp(b,'BSpotFilter'), A.CCR.BSFilter{...
                             length(A.CCR.BSFilter)+1} = ...
                             d; end;

% Set Variables Related to Displaying Figures
if strcmp(b,'ShowBounds'),  A.Display.Fig.Bnd = str2num(d); end;
if strcmp(b,'ShowEachBnd'), A.Display.Fig.Ebd = str2num(d); end;
if strcmp(b,'ShowFigRaw'),  A.Display.Fig.Raw = str2num(d); end;
if strcmp(b,'ShowFigAHE'),  A.Display.Fig.AHE = str2num(d); end;
if strcmp(b,'ShowFigBin'),  A.Display.Fig.Bin = str2num(d); end;
if strcmp(b,'ShowFigFil'),  A.Display.Fig.Fil = str2num(d); end;
if strcmp(b,'ShowFigInv'),  A.Display.Fig.Inv = str2num(d); end;
if strcmp(b,'ShowFigRP'),   A.Display.Fig.RP  = str2num(d); end;
if strcmp(b,'ShowFigRej'),  A.Display.Fig.Rej = str2num(d); end;
if strcmp(b,'ShowFigOvr'),  A.Display.Fig.Ovr = str2num(d); end;
if strcmp(b,'ShowFigCCR'),  A.Display.Fig.CCR = str2num(d); end;
if strcmp(b,'OnlyBounds'),  A.Display.Fig.OnlyBnd  = str2num(d); ...
   end;
if strcmp(b,'SaveFigRaw'),  A.Display.Fig.Save.Raw = str2num(d); ...
   end;
if strcmp(b,'SaveFigAHE'),  A.Display.Fig.Save.AHE = str2num(d); ...
   end;
if strcmp(b,'SaveFigBin'),  A.Display.Fig.Save.Bin = str2num(d); ...
   end;
if strcmp(b,'SaveFigFil'),  A.Display.Fig.Save.Fil = str2num(d); ...
   end;
```

```
    if strcmp(b,'SaveFigInv'),  A.Display.Fig.Save.Inv = str2num(d); ...
       end;
    if strcmp(b,'SaveFigRP'),   A.Display.Fig.Save.RP  = str2num(d); ...
       end;
    if strcmp(b,'SaveFigRej'),  A.Display.Fig.Save.Rej = str2num(d); ...
       end;
    if strcmp(b,'SaveFigOvr'),  A.Display.Fig.Save.Ovr = str2num(d); ...
       end;
    if strcmp(b,'SaveFigCCR'),  A.Display.Fig.Save.CCR = str2num(d); ...
       end;
    if strcmp(b,'SaveEachBnd'), A.Display.Fig.Save.Ebd = str2num(d); ...
       end;

    % Set Variables Related to Displaying Graphs / Plots
    if strcmp(b,'ShowGphFSBL'), A.Display.Gph.FSBL = str2num(d); end;
    if strcmp(b,'ShowGphDefl'), A.Display.Gph.Defl = str2num(d); end;
    if strcmp(b,'ShowGphTDEr'), A.Display.Gph.TDEr = str2num(d); end;
    if strcmp(b,'ShowGphNDef'), A.Display.Gph.NDef = str2num(d); end;
    if strcmp(b,'ShowGphRAng'), A.Display.Gph.RAng = str2num(d); end;

    % Set Fiber Rejection Variables
    if strcmp(b,'RejThresh'),  aRejectionThreshhold = str2double(d); ...
       end;
    if strcmp(b,'ARrejThresh'), ARrejectionThreshhold = ...
       str2double(d); end;
end

% Create Analysis Results subdirectory
if ~exist(sprintf('%s\\Analysis_Results',A.ImageDirectory),'dir');
    mkdir(sprintf('%s\\Analysis_Results',A.ImageDirectory));
end
AnalysisResultDirectory = ...
   sprintf('%s\\Analysis_Results',A.ImageDirectory);

% Reset Fiber Rejection Criteria if in variables file
if ~isempty(aRejectionThreshhold)
    areaRejectMin = 1 - aRejectionThreshhold;
    areaRejectMax = 1 + aRejectionThreshhold;
end

% if isempty(ARrejectionThreshhold)
%     ARrejectionThreshhold = aRejectionThreshhold;
% end

% Close the input file
fclose(variablesFile);
disp(A);

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Generate Boundary Box Location Image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if A.Display.Fig.Bnd
    fprintf('Boundary Box Location Image Generation\n');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
n = 1;

figure(n); set(gcf,'NumberTitle','off');
set(gcf,'Name','Boundary Box Location');
set(gcf,'Color',[1 1 1]);

leftSubplot = subplot(1,2,1);
set(leftSubplot,'Position',[0 0 .5 1]);

imshow(imread(sprintf('%s\\%s_%s_Static_NoFlow\\%s%05.0f.%s',...
                A.ImageDirectory,A.FlowCase{1},A.FlowCase{2},...
                A.Input{1},1,A.Input{2})));
xlabel('NoFlow');
hold on;
% Plot the fiber tip bounding box with green '+' markers
plot([A.BoundBox(1),A.BoundBox(2),A.BoundBox(1),A.BoundBox(2)],...
     [A.BoundBox(3),A.BoundBox(3),A.BoundBox(4),A.BoundBox(4)],...
     'g+','MarkerSize',8);

% Plot the bright spot location with a red 'o' marker
plot(A.CCR.BSLoc(1),A.CCR.BSLoc(2),...
    'ro','MarkerSize',5);

% Plot the bright spot search region bounding box with blue '+' ...
   markers
plot([A.CCR.BSLoc(1)-floor(A.CCR.BSLoc(3)/2),...
      A.CCR.BSLoc(1)+floor(A.CCR.BSLoc(3)/2),...
      A.CCR.BSLoc(1)-floor(A.CCR.BSLoc(3)/2),...
      A.CCR.BSLoc(1)+floor(A.CCR.BSLoc(3)/2)],...
     [A.CCR.BSLoc(2)-floor(A.CCR.BSLoc(3)/2),...
      A.CCR.BSLoc(2)-floor(A.CCR.BSLoc(3)/2),...
      A.CCR.BSLoc(2)+floor(A.CCR.BSLoc(3)/2),...
      A.CCR.BSLoc(2)+floor(A.CCR.BSLoc(3)/2)],...
     'b+','MarkerSize',10);
hold off;

rightSubplot = subplot(1,2,2);
set(rightSubplot,'Position',[.5 0 .5 1]);

imshow(imread(sprintf('%s\\%s_%s_Static_%imps\\%s%05.0f.%s',...
                A.ImageDirectory,A.FlowCase{1},A.FlowCase{2},...
                length(A.BinThresh)-1,A.Input{1},1,A.Input{2})));
xlabel('Largest Velocity Magnitude');
hold on;
% Plot the fiber tip bounding box with green '+' markers
plot([A.BoundBox(1),A.BoundBox(2),A.BoundBox(1),A.BoundBox(2)],...
     [A.BoundBox(3),A.BoundBox(3),A.BoundBox(4),A.BoundBox(4)],...
     'g+','MarkerSize',8);

% Plot the bright spot location with a red 'o' marker
plot(A.CCR.BSLoc(1),A.CCR.BSLoc(2),...
    'ro','MarkerSize',5);
```

```
    % Plot the bright spot search region bounding box with blue '+' ...
       markers
    plot([A.CCR.BSLoc(1)-floor(A.CCR.BSLoc(3)/2),...
          A.CCR.BSLoc(1)+floor(A.CCR.BSLoc(3)/2),...
          A.CCR.BSLoc(1)-floor(A.CCR.BSLoc(3)/2),...
          A.CCR.BSLoc(1)+floor(A.CCR.BSLoc(3)/2)],...
         [A.CCR.BSLoc(2)-floor(A.CCR.BSLoc(3)/2),...
          A.CCR.BSLoc(2)-floor(A.CCR.BSLoc(3)/2),...
          A.CCR.BSLoc(2)+floor(A.CCR.BSLoc(3)/2),...
          A.CCR.BSLoc(2)+floor(A.CCR.BSLoc(3)/2)],...
         'b+','MarkerSize',10);
    hold off;

end
if ~A.Display.Fig.OnlyBnd
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Build Cross-Correlation Reference Image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('Cross-Correlation Reference Image Construction\n');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Create the NoFlow Directory string
NoFlowDir = sprintf('%s\\%s_%s_Static_NoFlow',...
                    A.ImageDirectory,A.FlowCase{1},A.FlowCase{2});

% Count how many NoFlow images are present in the folder
NumNoFlowImgs = length(dir(sprintf('%s\\%s*.%s',...
                                   NoFlowDir,A.Input{1},A.Input{2})));

% Initialize the refImg matrix
refImg = ...
    zeros(ceil((2*A.CCR.BSLoc(3)+1)/2),ceil((2*A.CCR.BSLoc(3)+1)/2));

for i=1:NumNoFlowImgs
    filename = sprintf('%s\\%s%05.0f.%s',...
                       NoFlowDir,A.Input{1},i,A.Input{2});
    refImgRaw = imread(filename);

    % Sum the images together
    refImg = refImg + double(rgb2gray(refImgRaw(...
                             A.CCR.BSLoc(2)-floor(A.CCR.BSLoc(3)/2):...
                             A.CCR.BSLoc(2)+floor(A.CCR.BSLoc(3)/2),...
                             A.CCR.BSLoc(1)-floor(A.CCR.BSLoc(3)/2):...
                             A.CCR.BSLoc(1)+floor(A.CCR.BSLoc(3)/2),...
                        :)));
end

% Divide by the number of image in the set to obtain the average image
A.CCR.refImg = uint8(floor(refImg/NumNoFlowImgs));

clipMax = A.CCR.clip(1);
clipMin = A.CCR.clip(2);

meanRefImg = mean(mean(A.CCR.refImg));
```

```
maxRefImg  = max(max(A.CCR.refImg));

imageSize = size(refImg);
A.CCR.refImgSBC = double(zeros(imageSize));

A.CCR.refImgSBC = (A.CCR.refImg-meanRefImg)/(maxRefImg-meanRefImg);

for k=1:imageSize(1)
for L=1:imageSize(2)

if A.CCR.refImgSBC(k,L) > clipMax, A.CCR.refImgSBC(k,L) = clipMax; end;
if A.CCR.refImgSBC(k,L) < clipMin, A.CCR.refImgSBC(k,L) = clipMin; end;

end
end

% A.CCR.refImgSBC = A.CCR.refImgSBC*255;

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Generate Falkner-Skan Boundary Layer Profile
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('Falkner-Skan Boundary Layer Profile Generation\n');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=2;

u = 0:0.01:1;
Beta = 2*str2num(regexprep(regexprep(A.FlowCase{3},'+',''),'-',''));
Beta = str2double(A.FlowCase{3});

z = calc_FS_Profile(Beta,etamax,etasteps,u);

eta = z(:,3);
eta_delta = interp1(eta,u,u_delta,'spline');

% Display the figure, if user prefers
if A.Display.Gph.FSBL
    figure(n); set(gcf,'NumberTitle','off');
    set(gcf,'Color',[1 1 1]);
    set(gcf,'Name','Falkner-Skan Boundary Layer Profile');
    plot(u,eta,'b-','LineWidth',2);
    grid on;
    titlename = ['\bf Falkner-Skan Boundary Layer Profile ( \beta = ...
        ',...
                 sprintf('%2.1f',Beta),'^\circ )'];
    title(titlename,'FontSize',14);
    xlabel('Dimensionless Velocity');
    ylabel('Scaled Height');
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     Loop Over Every Flow Case Present
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('=============== Main Program Loop ===============\n');
fprintf('================ %s_%s_%sAOA ================\n',...
```

```
        A.FlowCase{1},A.FlowCase{2},...
        regexprep(regexprep(A.FlowCase{3},...
                                    '-', 'n'),'+', 'p'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:length(A.BinThresh)
    if i < skipToVelocity+1 && i~= 1, continue; end;
    FlowCaseImageDirectory = sprintf('%s\\%s_%s_Static_%1.0imps',...
                                     A.ImageDirectory,...
                                     A.FlowCase{1},A.FlowCase{2},i-1);

    if i==1 %#ok<*ALIGN>
        FlowCaseImageDirectory = ...
            sprintf('%s\\%s_%s_Static_NoFlow',A.ImageDirectory,...
                                              A.FlowCase{1},...
                                              A.FlowCase{2});
    end

    % Find the number of images in current flow case
    NumFlowCaseImages = length(dir(sprintf('%s\\%s*.%s',...
                           FlowCaseImageDirectory,...
                           A.Input{1},A.Input{2})));

   fprintf('-------------------------------------------------------------\n');
    if i==1, fprintf('      %s: NoFlow\n',...
                     'Calculating Data for Flow case');
    else     fprintf('      %s: %2.0i m/s\n',...
                     'Calculating Data for Flow case',i-1);
    end
   fprintf('-------------------------------------------------------------\n');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               Loop over every image in the current flow case
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j=1:NumFlowCaseImages
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                          Read in the raw image
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    n = 3;  % Set Image Number

    % Create
    ImageFilename = sprintf('%s\\%s%05.0f.%s',...
                            FlowCaseImageDirectory,...
                            A.Input{1},j,A.Input{2});
    refImgRaw = imread(ImageFilename);
    refImg = rgb2gray(refImgRaw(...
                                A.BoundBox(3):A.BoundBox(4),...
                                A.BoundBox(1):A.BoundBox(2),:));

    % Display the figure, if user prefers
    if A.Display.Fig.Raw
        figure(n); set(gcf,'NumberTitle','off');
        set(gcf,'Color',[1 1 1]);
        set(gcf,'Name','Raw'); imshow(refImg);
    end
```

```
image = refImg;

n = n + 1;
if A.Display.Fig.Ebd
    figure(n); set(gcf,'NumberTitle','off');
    set(gcf,'Color',[1 1 1]);

    set(gcf,'Name','Raw Bounds'); imshow(refImgRaw);

    hold on;
    % Plot the fiber tip bounding box with green '+' markers
    plot([A.BoundBox(1),A.BoundBox(2),A.BoundBox(1),A.BoundBox(2)],...
         [A.BoundBox(3),A.BoundBox(3),A.BoundBox(4),A.BoundBox(4)],...
         'g+','MarkerSize',8);

    % Plot the bright spot location with a red 'o' marker
    plot(A.CCR.BSLoc(1),A.CCR.BSLoc(2),...
        'ro','MarkerSize',5);

    % Plot the bright spot search region bounding box with blue ...
       '+' markers
    plot([A.CCR.BSLoc(1)-floor(A.CCR.BSLoc(3)/2),...
          A.CCR.BSLoc(1)+floor(A.CCR.BSLoc(3)/2),...
          A.CCR.BSLoc(1)-floor(A.CCR.BSLoc(3)/2),...
          A.CCR.BSLoc(1)+floor(A.CCR.BSLoc(3)/2)],...
         [A.CCR.BSLoc(2)-floor(A.CCR.BSLoc(3)/2),...
          A.CCR.BSLoc(2)-floor(A.CCR.BSLoc(3)/2),...
          A.CCR.BSLoc(2)+floor(A.CCR.BSLoc(3)/2),...
          A.CCR.BSLoc(2)+floor(A.CCR.BSLoc(3)/2)],...
         'b+','MarkerSize',10);
    hold off;

    if A.Display.Fig.Save.Ebd
        for k=1:length(A.IAfType)
              saveas(gcf,sprintf('%s\\%s_%s_%imps_%05i',...
              AnalysisResultDirectory,...
              A.Output{1},get(gcf,'Name'),...
              i-1,j),A.IAfType{k});
            end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      Contrast-Limited Adaptive Histogram Equalization (CLAHE)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = n + 1;  % Set Image Number
if A.CLAHE.Run
    image = adapthisteq(refImg,...
                'NumTiles',[A.CLAHE.NTiles,A.CLAHE.NTiles],...
                'Distribution',A.CLAHE.Distr);

    % Display / Save the figure, if user prefers
    if A.Display.Fig.AHE
```

```
        figure(n); set(gcf,'NumberTitle','off');
        set(gcf,'Color',[1 1 1]);
        set(gcf,'Name','CLAHE'); imshow(image);
        if A.Display.Fig.Save.AHE
            for k=1:length(A.IAfType)
              saveas(gcf,sprintf('%s\\%s_%s_%imps_%05i',...
              AnalysisResultDirectory,...
              A.Output{1},get(gcf,'Name'),...
              i-1,j),A.IAfType{k});
            end
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Image Binarization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = n + 1;  % Set Image Number
image = im2bw(image,A.BinThresh(i)/255);

% Display / Save the figure, if user prefers
if A.Display.Fig.Bin
    figure(n); set(gcf,'NumberTitle','off');
    set(gcf,'Color',[1 1 1]);
    set(gcf,'Name','Binarized'); imshow(image);
    if A.Display.Fig.Save.Bin
        for k=1:length(A.IAfType)
              saveas(gcf,sprintf('%s\\%s_%s_%imps_%05i',...
              AnalysisResultDirectory,A.Output{1},get(gcf,'Name'),...
              i-1,j),A.IAfType{k});
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            Image Filtering
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = n + 1;  % Set Image Number
for k=1:length(A.Filtering), image = ...
   bwmorph(image,A.Filtering{k}); end

% Display the figure, if user prefers
if A.Display.Fig.Fil
    figure(n); set(gcf,'NumberTitle','off');
    set(gcf,'Color',[1 1 1]);
    set(gcf,'Name','Filtered'); imshow(image);
    if A.Display.Fig.Save.Fil
        for k=1:length(A.IAfType)
          saveas(gcf,sprintf('%s\\%s_%s_%imps_%05i',...
          AnalysisResultDirectory,A.Output{1},get(gcf,'Name'),...
          i-1,j),A.IAfType{k});
        end
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            Image Inversion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = n + 1;  % Set Image Number
image = 1 - image;

% Display the figure, if user prefers
if A.Display.Fig.Inv
    figure(n); set(gcf,'NumberTitle','off');
    set(gcf,'Color',[1 1 1]);
    set(gcf,'Name','Inverted'); imshow(image);
    if A.Display.Fig.Save.Inv
        for k=1:length(A.IAfType)
          saveas(gcf,sprintf('%s\\%s_%s_%imps_%05i',...
          AnalysisResultDirectory,A.Output{1},get(gcf,'Name'),...
          i-1,j),A.IAfType{k});
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     Get Regionprops Information
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = n + 1;  % Set Image Number
CC = bwconncomp(image,8);
RP = regionprops(CC,'Area','BoundingBox','Image',...
                    'Orientation','MajorAxisLength',...
                    'MinorAxisLength');

% Index fiber area information to search for max
fiber_area =        zeros(length(RP),1);            % Initialization
for k=1:length(RP), fiber_area(k) = RP(k).Area; end % Indexing
[fiber_maxarea,fiber_index] = max(fiber_area);
A.Fiber.Area(i,j)           = RP(fiber_index).Area;
A.Fiber.BoundingBox(i,j,:)  = RP(fiber_index).BoundingBox;
A.Fiber.Orientation(i,j)    = RP(fiber_index).Orientation;
A.Fiber.MajAxLen(i,j)       = RP(fiber_index).MajorAxisLength;
A.Fiber.MinAxLen(i,j)       = RP(fiber_index).MinorAxisLength;
A.Fiber.AR(i,j)             = A.Fiber.MajAxLen(i,j) / ...
                              A.Fiber.MinAxLen(i,j);

% Save fiber image
base_image                  = zeros(size(refImg));
fiber_image                 = RP(fiber_index).Image;

% Place the fiber image inside of the base image
% in order to conserve original image size
for k=1:size(fiber_image,1)
    for L=1:size(fiber_image,2)
        base_image(...
          k+floor(RP(fiber_index).BoundingBox(1,2)),...
          L+floor(RP(fiber_index).BoundingBox(1,1))) ...
          = fiber_image(k,L);
```

```
    end
end

fiber_image             = base_image;           clear base_image;
A.Fiber.Images(i,j).RP  = repmat(uint8(255*fiber_image),[1 1 3]);

% Display the figure, if user prefers
if A.Display.Fig.RP
    figure(n); set(gcf,'NumberTitle','off');
    set(gcf,'Color',[1 1 1]);
    set(gcf,'Name','Regionprops'); imshow(fiber_image);
    if A.Display.Fig.Save.RP
        for k=1:length(A.IAfType)
          saveas(gcf,sprintf('%s\\%s_%s_%imps_%05i',...
          AnalysisResultDirectory,A.Output{1},get(gcf,'Name'),...
          i-1,j),A.IAfType{k});
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Perform a cross-correlation using the current and reference ...
   images
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = n + 1;  % Set Image Number
curImg = double(rgb2gray(refImgRaw(...
                A.CCR.BSLoc(2)-floor(A.CCR.BSLoc(3)/2):...
                A.CCR.BSLoc(2)+floor(A.CCR.BSLoc(3)/2),...
                A.CCR.BSLoc(1)-floor(A.CCR.BSLoc(3)/2):...
                A.CCR.BSLoc(1)+floor(A.CCR.BSLoc(3)/2),...
                :))));
% Current image statistics
meanCurImg = mean(mean(curImg));
maxCurImg  = max(max(curImg));

imageSize = size(curImg);
A.CCR.curImgSBC = double(zeros(imageSize));


% Create the current clipped image
for k=1:imageSize(1)
for L=1:imageSize(2)

A.CCR.curImgSBC(k,L) = ...
    (curImg(k,L)-meanCurImg)/(maxCurImg-meanCurImg);
if A.CCR.curImgSBC(k,L) > clipMax, A.CCR.curImgSBC(k,L) = ...
    clipMax; end;
if A.CCR.curImgSBC(k,L) < clipMin, A.CCR.curImgSBC(k,L) = ...
    clipMin; end;

end
end

% Apply any filtering to curImgSBC, defined in the input file
```

```
    for k=1:length(A.CCR.BSFilter)
        dummy = A.CCR.BSFilter{k};
        if str2num(dummy(1:2)) ~= i-1, continue; end;

        usingX = strcmpi(dummy(4),'X');
        usingY = strcmpi(dummy(4),'Y');
        gt = strcmp(dummy(5),'>');
        lt = strcmp(dummy(5),'<');

        parameter = str2num(dummy(6:length(dummy)));

        if usingX
            if gt
                A.CCR.curImgSBC(:,...
                    parameter:size(A.CCR.curImgSBC,2)) = 0;
            end

            if lt
                A.CCR.curImgSBC(:,1:parameter) = 0;
            end

        end

        if usingY
            if gt
                A.CCR.curImgSBC(...
                    parameter:size(A.CCR.curImgSBC,1),:) = 0;
            end

            if lt
                A.CCR.curImgSBC(1:parameter,:) = 0;
            end

        end

    end

%     A.CCR.curImgSBC = A.CCR.curImgSBC*255;  % (unnecessary)

    % Perform the cross-correlation
    CCR = xcorr2(A.CCR.curImgSBC,A.CCR.refImgSBC);

    % Pick off the maximum spot
    [yMax,xMax] = find(CCR==max(CCR(:)),1);

    % X-shift for the current image
    A.CCR.Shift(i,j,1) = -A.CCR.BSLoc(3) + ...
        (xMax +          (CCR(yMax,xMax-1) - CCR(yMax,xMax+1)) / ...
             (2*(CCR(yMax,xMax-1) + CCR(yMax,xMax+1) - ...
                2*CCR(yMax,xMax))));

    % Y-shift for the current image
    A.CCR.Shift(i,j,2) = A.CCR.BSLoc(3) - ...
        (yMax +          (CCR(yMax-1,xMax) - CCR(yMax+1,xMax)) / ...
```

```
            (2*(CCR(yMax-1,xMax) + CCR(yMax+1,xMax) - ...
               2*CCR(yMax,xMax))));

    % Display the figure, if user prefers
    if A.Display.Fig.CCR
        figure(n); set(gcf,'NumberTitle','off');
        set(gcf,'Color',[1 1 1]);
        set(gcf,'Name','Cross-Correlation');
        imagesc(CCR,...                                % Graph ...
           cross-correlation
            [0,max(max(CCR(:)),160)]);                 % Include a color ...
               range
        hold on;
        plot(A.CCR.BSLoc(3),A.CCR.BSLoc(3),'r+'); % Red '+' in the ...
           center
        hold off;
        if A.CCR.grey, colormap('gray');           % Set colormap color
        else           colormap('jet'); end

        colorbar;                                  % Add a colorbar
        grid on;
        if A.Display.Fig.Save.CCR
            for k=1:length(A.IAfType)
              saveas(gcf,sprintf('%s\\%s_%s_%imps_%05i',...
              AnalysisResultDirectory,A.Output{1},get(gcf,'Name'),...
              i-1,j),A.IAfType{k});
            end
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              Fiber Rejection
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Once the entire set of image data has been populated for the ...
   current flow
% case, examine the fiber area from Regionprops to determine if ...
   there are
% any outliers.  Discard these outliers if they exist.  Otherwise, ...
   conserve
% the analysis data for the current fiber image. Count the number of
% outliers discarded.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = n + 1;
mean_area = mean(A.Fiber.Area(i,:));
mean_AR   = mean(A.Fiber.AR(i,:));

% Reset loop variables
x_tipLocation = [];
y_tipLocation = [];
x_shift = [];
y_shift = [];
angle = [];
rejConsOut = '';
```

```
for j=1:NumFlowCaseImages

    A.Fiber.Rejection(i,j) = 0;

%     fprintf('A.Fiber.AR(i,j): %5.4f\n',A.Fiber.AR(i,j));
%     fprintf('mean_AR: %5.4f\n',mean_AR);
%     fprintf('ARRejectionThreshhold: %5.4f\n',ARrejectionThreshhold);

    if A.Fiber.Area(i,j) < mean_area*areaRejectMin || ...
       A.Fiber.Area(i,j) > mean_area*areaRejectMax || ...
       A.Fiber.AR(i,j) < mean_AR*ARrejectionThreshhold

        A.Fiber.Rejection(i,j) = 1;

        rejConsOut = sprintf('%sFiber rejected: %imps %i ',...
            rejConsOut,i-1,j);

        if A.Fiber.AR(i,j) < mean_AR*ARrejectionThreshhold
            rejConsOut = sprintf('%s(AR)',rejConsOut);
        end
        if A.Fiber.Area(i,j) < mean_area*areaRejectMin
            rejConsOut =sprintf('%s(Area - Too small)',rejConsOut);
        end
        if A.Fiber.Area(i,j) > mean_area*areaRejectMax
            rejConsOut = sprintf('%s(Area - Too big)',rejConsOut);
        end
        rejConsOut = sprintf('%s\n',rejConsOut);

        A.Fiber.Images(i,j).RP(:,:,2) = 0;
        A.Fiber.Images(i,j).RP(:,:,3) = 0;
    else
        x_tipLocation(length(x_tipLocation)+1) ...
           =A.Fiber.BoundingBox(i,j,1);
        y_tipLocation(length(y_tipLocation)+1) ...
           =A.Fiber.BoundingBox(i,j,2);
        x_shift(length(x_shift)+1) = A.CCR.Shift(i,j,1);
        y_shift(length(y_shift)+1) = A.CCR.Shift(i,j,2);
        angle(length(angle)+1) = A.Fiber.Orientation(i,j);

        if angle(length(angle)) > 0
            x_tipLocation(length(x_tipLocation)) = ...
                            x_tipLocation(length(x_tipLocation)) + ...
                            A.Fiber.BoundingBox(i,j,3);
        end

%         disp([x_tipLocation(length(x_tipLocation)),...
%               y_tipLocation(length(y_tipLocation))]);


        A.Fiber.Images(i,j).RP(:,:,1) = 0;
        A.Fiber.Images(i,j).RP(:,:,3) = 0;
    end
```

```
    % Display the figure, if user prefers
    if A.Display.Fig.Rej
        figure(n); set(gcf,'NumberTitle','off');
        set(gcf,'Color',[1 1 1]);
        set(gcf,'Name','RP-Rejection');
        imshow(A.Fiber.Images(i,j).RP);
        if length(x_tipLocation) && length(y_tipLocation)
            hold on;
            plot(x_tipLocation(length(x_tipLocation)),...
                 y_tipLocation(length(y_tipLocation)),'rx');
            hold off;
        end
        pause(0.1);
        if A.Display.Fig.Save.Rej
            for k=1:length(A.IAfType)
              saveas(gcf,sprintf('%s\\%s_%s_%imps_%05i',...
              AnalysisResultDirectory,A.Output{1},get(gcf,'Name'),...
              i-1,j),A.IAfType{k});
            end
        end

    end

end

n = length(angle);
disp(['X Tip Loc  ','Y Tip Loc  ',' X Shift ',' Y Shift  ','  Angle']);
disp([x_tipLocation',y_tipLocation',x_shift',y_shift',angle']);
fprintf('%s',rejConsOut);

if i==1
    initial_x_tipLocation = mean(x_tipLocation);
    initial_y_tipLocation = mean(y_tipLocation);
    initial_angle         = mean(angle);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Fiber Deflection Calculations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Fiber X Deflection
A.Fiber.Deflection(i,1) = ...
    (mean(x_tipLocation - x_shift) - initial_x_tipLocation)*PW;

% Fiber Y Deflection
A.Fiber.Deflection(i,2) = ...
    (mean(y_tipLocation + y_shift) - initial_y_tipLocation)*PW;

% Fiber Total Deflection
% A.Fiber.Deflection(i,3) = ...
%     sqrt((A.Fiber.Deflection(i,1))^2 + (A.Fiber.Deflection(i,2))^2);

% Fiber Absolute Relative Angle
A.Fiber.RelativeAngle(i) = abs(mean(angle) - initial_angle);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      Deflection Uncertainty Calculations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Find the Student's t value for two-sided 95% confidence using new ...
    # DoF
student_t    = tinv(0.975,n-1);

% X-Deflection Uncertainty
A.Fiber.Udeflection(i,1) = ...
     (std(((x_tipLocation(:) - x_shift(:)) - ...
        initial_x_tipLocation)*PW)/...
                        sqrt(n))*student_t;

% Y-Deflection Uncertainty
A.Fiber.Udeflection(i,2) = ...
     (std(((y_tipLocation(:) - y_shift(:)) - ...
        initial_y_tipLocation)*PW)/...
                        sqrt(n))*student_t;

% Total Deflection Uncertainty
A.Fiber.Udeflection(i,3) = ...
     sqrt((A.Fiber.Udeflection(i,1))^2 + (A.Fiber.Udeflection(i,2))^2);

% Relative Angle Uncertainty
A.Fiber.Urelativeangle(i) = ...
     (std(angle-initial_angle)/sqrt(n))*student_t;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Boundary Layer Penetration Calculation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A.Fiber.HOverDelta(i) = 0;
m = abs(Beta/(2-Beta));

if i > 1
    A.Fiber.HOverDelta(i) = (A.Fiber.Height/(1000*eta_delta))*...
        (((m+1)/2)*(i-1)*A.Air.Density/...
       (A.Air.DynVisc*(A.Fiber.Pos(1)/1000)))^(1/2);  % Dim'n-less
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end % of main program loop

% Remove any bias in the data
A.Fiber.Deflection(:,1) = A.Fiber.Deflection(:,1) ...
   -A.Fiber.Deflection(1,1);
A.Fiber.Deflection(:,2) = A.Fiber.Deflection(:,2) ...
   -A.Fiber.Deflection(1,2);

for i=1:length(A.BinThresh)
```

```
% A.Fiber.Deflection(:,3) = A.Fiber.Deflection(:,3) ...
   -A.Fiber.Deflection(1,3);
A.Fiber.Deflection(i,3) = ...
    sqrt((A.Fiber.Deflection(i,1))^2 + (A.Fiber.Deflection(i,2))^2);
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                Figure Plotting
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate the Fiber Deflections Figure
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if A.Display.Gph.Defl
figure;
set(gcf,'Color',[1 1 1]);
set(gcf,'Name','Fiber Deflections','NumberTitle','off');
plot(0:length(A.BinThresh)-1,...
     abs(A.Fiber.Deflection(:,1)),'ko-'); % Longitudinal
hold on;
plot(0:length(A.BinThresh)-1,...
     abs(A.Fiber.Deflection(:,2)),'ko:'); % Transverse
plot(0:length(A.BinThresh)-1,...
     abs(A.Fiber.Deflection(:,3)),'ks-',...
     'MarkerFaceColor',[0 0 0]); % Total ([x^2+y^2]^.5)
hold off;
legend('Longitudinal','Transverse','Total','Location','NorthWest');
Yaxismaxnumber = ceil(max(abs(A.Fiber.Deflection(:,3)))/50)*50;
axis([0 10 0 Yaxismaxnumber]);
set(gca,'XTick',[0 2 4 6 8 10]);
set(gca,'YTick',0:Yaxismaxnumber/5:Yaxismaxnumber);
grid on;
xlabel('Flow Velocity (m/s)');
ylabel('Fiber Displacement (\mum)');
for i=2:length(A.Output)
saveAsFileName = ...
         sprintf(...
         '%s\\Analysis_Results\\%s_%s_Static_Fiber_Deflections',...
                 A.ImageDirectory,A.FlowCase{1},A.FlowCase{2});
saveas(gcf,saveAsFileName,A.Output{i});

if strcmp(A.Output{i},'jpg') || strcmp(A.Output{i},'fig')
    saveAsFileName = sprintf('%s.%s',saveAsFileName,A.Output{i});
    zipFileList{length(zipFileList)+1} = saveAsFileName;
end

end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate the Total Fiber Deflection + Errorbar Figure
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if A.Display.Gph.TDEr
figure;
```

```
set(gcf,'Color',[1 1 1]);
set(gcf,'Name','Total Fiber Deflection (Errorbar)',...
        'NumberTitle','off');
errorbar(0:length(A.BinThresh)-1,A.Fiber.Deflection(:,3),...
         A.Fiber.Udeflection(:,3),'ko-');
current_Ylim = get(gca,'YLim');
Yaxismaxnumber = ceil(max(abs(A.Fiber.Deflection(:,3)))/50)*50;
axis([0 10 0 current_Ylim(2)]);
set(gca,'XTick',[0 2 4 6 8 10]);
% set(gca,'YTick',0:Yaxismaxnumber/5:Yaxismaxnumber);
grid on;
     xlabel('Flow Velocity (m/s)');
ylabel('Fiber Displacement (\mum)');
for i=2:length(A.Output)
saveAsFileName = ...
         sprintf('%s\\Analysis_Results\\%s_%s_%s',...
                 A.ImageDirectory,A.FlowCase{1},A.FlowCase{2},...
                 'Static_Total_Fiber_Deflections_Errorbar');
saveas(gcf,saveAsFileName,A.Output{i});

if strcmp(A.Output{i},'jpg') || strcmp(A.Output{i},'fig')
    saveAsFileName = sprintf('%s.%s',saveAsFileName,A.Output{i});
    zipFileList{length(zipFileList)+1} = saveAsFileName;
end

end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate the Normalized Fiber Deflections Figure
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if A.Display.Gph.NDef
figure;
set(gcf,'Color',[1 1 1]);
set(gcf,'Name','Normalized Fiber Deflections','NumberTitle','off');
plot(0:length(A.BinThresh)-1,...                % Longitudinal
     abs(A.Fiber.Deflection(:,1)/(1000*A.Fiber.Height)),'ko-');
hold on;
plot(0:length(A.BinThresh)-1,...                % Transverse
     abs(A.Fiber.Deflection(:,2)/(1000*A.Fiber.Height)),'ko:');
plot(0:length(A.BinThresh)-1,...                % Total ([x^2+y^2]^.5)
     abs(A.Fiber.Deflection(:,3)/(1000*A.Fiber.Height)),'ks-',...
     'MarkerFaceColor',[0 0 0]);
hold off;
legend('Longitudinal','Transverse','Total','Location','NorthWest');
% axis([0 10 0 1]);
set(gca,'XTick',[0 2 4 6 8 10]);
grid on;
xlabel('Flow Velocity (m/s)');
ylabel('Normalized Fiber Displacement');
for i=2:length(A.Output)
    saveAsFileName = sprintf(...
  '%s\\Analysis_Results\\%s_%s_Normalized_Static_Fiber_Deflections',...
                A.ImageDirectory,A.FlowCase{1},A.FlowCase{2});
```

```
saveas(gcf,saveAsFileName,A.Output{i});

if strcmp(A.Output{i},'jpg') || strcmp(A.Output{i},'fig')
    saveAsFileName = sprintf('%s.%s',saveAsFileName,A.Output{i});
    zipFileList{length(zipFileList)+1} = saveAsFileName;
end

end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate the Relative Fiber Angle Figure
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if A.Display.Gph.RAng
figure;
set(gcf,'Color',[1 1 1]);
set(gcf,'Name','Relative Fiber Angle (deg)','NumberTitle','off');
hold off;
plot(0:length(A.BinThresh)-1,abs(A.Fiber.RelativeAngle),'ro');
set(gca,'XTick',[0 2 4 6 8 10]);
ytick_array = get(gca,'YTick');
% set(gca,'YTick',0:5:ceil(ytick_array(length(ytick_array))/5)*5);
axis 'auto y';
grid on;
xlabel('Flow Velocity (m/s)');
ylabel('Relative Fiber Angle (deg)');
for i=2:length(A.Output)
saveAsFileName = ...
   sprintf('%s\\Analysis_Results\\%s_%s_Rel_Fiber_Angles',...
               A.ImageDirectory,A.FlowCase{1},A.FlowCase{2});
saveas(gcf,saveAsFileName,A.Output{i});

if strcmp(A.Output{i},'jpg') || strcmp(A.Output{i},'fig')
    saveAsFileName = sprintf('%s.%s',saveAsFileName,A.Output{i});
    zipFileList{length(zipFileList)+1} = saveAsFileName;
end

end
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Output Information to a File
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Create Output Filenames
outFilename_Fiber = sprintf('%s\\Analysis_Results\\%s_%s_%sAOA_%s',...
                    A.ImageDirectory,A.FlowCase{1},A.FlowCase{2},...
                    regexprep(regexprep(...
                       A.FlowCase{3},'-','n'),'+','p'),...
                    'Static_Fiber_Image_Properties.txt');

outFilename_FiberHdr = ...
   sprintf('%s\\Analysis_Results\\%s_%s_%sAOA_%s',...
```

```
                    A.ImageDirectory,A.FlowCase{1},A.FlowCase{2},...
                    regexprep(regexprep(...
                       A.FlowCase{3},'-','n'),'+','p'),...
                    'Static_Fiber_Image_Properties_Header.txt');

outFilename_Defl  = sprintf('%s\\Analysis_Results\\%s_%s_%sAOA_%s',...
                    A.ImageDirectory,A.FlowCase{1},A.FlowCase{2},...
                    regexprep(regexprep(...
                       A.FlowCase{3},'-','n'),'+','p'),...
                    'Static_Fiber_Deflections.txt');

outFilename_DeflHdr  = ...
   sprintf('%s\\Analysis_Results\\%s_%s_%sAOA_%s',...
                    A.ImageDirectory,A.FlowCase{1},A.FlowCase{2},...
                    regexprep(regexprep(...
                       A.FlowCase{3},'-','n'),'+','p'),...
                    'Static_Fiber_Deflections_Header.txt');

% Add output files to zip file list
zipFileList{length(zipFileList)+1} = outFilename_Fiber;
zipFileList{length(zipFileList)+1} = outFilename_FiberHdr;
zipFileList{length(zipFileList)+1} = outFilename_Defl;
zipFileList{length(zipFileList)+1} = outFilename_DeflHdr;

out_Fiber    = fopen(outFilename_Fiber,'w');
out_FiberHdr = fopen(outFilename_FiberHdr,'w');
out_Defl     = fopen(outFilename_Defl,'w');
out_DeflHdr  = fopen(outFilename_DeflHdr,'w');

for i=1:length(A.BinThresh)
    FlowCaseImageDirectory = sprintf('%s\\%s_%s_Static_%1.0imps',...
                                     A.ImageDirectory,...
                                     A.FlowCase{1},A.FlowCase{2},i-1);

    if i==1 %#ok<*ALIGN>
        FlowCaseImageDirectory = ...
            sprintf('%s\\%s_%s_Static_NoFlow',A.ImageDirectory,...
                                              A.FlowCase{1},...
                                              A.FlowCase{2});
    end

    % Find the number of images in current flow case
    NumFlowCaseImages = length(dir(sprintf('%s\\%s*.%s',...
                           FlowCaseImageDirectory,...
                           A.Input{1},A.Input{2})));

    for j=1:NumFlowCaseImages
        % Output the fiber information to out_Fiber
        if (i-1) == 0, fprintf(out_Fiber,'0\t');
        else           fprintf(out_Fiber,'%i\t',i-1);
        end
        fprintf(out_Fiber,'%i\t',j);
        fprintf(out_Fiber,'%i\t',A.Fiber.Rejection(i,j));
        if A.Fiber.Rejection(i,j)
```

```
            fprintf('\t\t\t\t\t\t\t\t\t\t');
        else
            fprintf(out_Fiber,'%i\t',A.Fiber.Area(i,j));
            fprintf(out_Fiber,'%f\t',A.Fiber.BoundingBox(i,j,1));
            fprintf(out_Fiber,'%f\t',A.Fiber.BoundingBox(i,j,2));
            fprintf(out_Fiber,'%f\t',A.Fiber.BoundingBox(i,j,3));
            fprintf(out_Fiber,'%f\t',A.Fiber.BoundingBox(i,j,4));
            fprintf(out_Fiber,'%f\t',A.Fiber.Orientation(i,j));
            fprintf(out_Fiber,'%f\t',A.CCR.Shift(i,j,1));
            fprintf(out_Fiber,'%f\t',A.CCR.Shift(i,j,2));
        end
        fprintf(out_Fiber,'\n');
    end

    % Output the deflection information to out_Defl
    fprintf(out_Defl,'%2.0i\t',i-1);
    fprintf(out_Defl,'%8.4f\t',A.Fiber.Deflection(i,1));
    fprintf(out_Defl,'%8.4f\t',A.Fiber.Deflection(i,2));
    fprintf(out_Defl,'%8.4f\t',A.Fiber.Deflection(i,3));
    fprintf(out_Defl,'%6.3f\t',A.Fiber.RelativeAngle(i));
    fprintf(out_Defl,'%8.4f\t',A.Fiber.Udeflection(i,1));
    fprintf(out_Defl,'%8.4f\t',A.Fiber.Udeflection(i,2));
    fprintf(out_Defl,'%8.4f\t',A.Fiber.Udeflection(i,3));
    fprintf(out_Defl,'%8.4f\t',A.Fiber.Urelativeangle(i));
    fprintf(out_Defl,'%8.4f\t',A.Fiber.HOverDelta(i));
    fprintf(out_Defl,'\n');
end

% Output the header information to out_FiberHdr
fprintf(out_FiberHdr,'Flow Velocity (m/s)\n');
fprintf(out_FiberHdr,'Fiber Image\n');
fprintf(out_FiberHdr,'Rejection Bool\n');
fprintf(out_FiberHdr,'Fiber Area (pixels)\n');
fprintf(out_FiberHdr,'Fiber Bounding Box - XS (pixel location)');
fprintf(out_FiberHdr,'Fiber Bounding Box - XF (pixel location)');
fprintf(out_FiberHdr,'Fiber Bounding Box - YS (pixel location)');
fprintf(out_FiberHdr,'Fiber Bounding Box - YF (pixel location)');
fprintf(out_FiberHdr,'Fiber Orientation (deg)\n');
fprintf(out_FiberHdr,'Cross-Correlation X-Shift (pixels)\n');
fprintf(out_FiberHdr,'Cross-Correlation Y-Shift (pixels)\n');

% Output the header information to out_DeflHdr
fprintf(out_DeflHdr,'Flow Velocity (m/s)');
fprintf(out_DeflHdr,'Longitudinal Fiber Deflection (micrometers)\n');
fprintf(out_DeflHdr,'Transverse Fiber Deflection (micrometers)\n');
fprintf(out_DeflHdr,'Total Fiber Deflection (micrometers)\n');
fprintf(out_DeflHdr,'Relative Fiber Angle (degrees)\n');
fprintf(out_DeflHdr,'Lontitudinal Fiber Deflection Uncertainty %s\n',...
                    '(micrometers)');
fprintf(out_DeflHdr,'Transverse Fiber Deflection Uncertainty %s\n',...
                    '(micrometers)');
fprintf(out_DeflHdr,'Total Fiber Deflection Uncertainty ...
    (micrometers)\n');
fprintf(out_DeflHdr,'Relative Fiber Angle Uncertainty (degrees)\n');
```

```
fprintf(out_DeflHdr,'H/delta\n');

% Close the output files
fclose(out_Fiber);
fclose(out_FiberHdr);
fclose(out_Defl);
fclose(out_DeflHdr);

% Create Zip file with required information in the base image directory
zipFileName = sprintf('%s\\%s_%s_%sAOA_Analysis_Results.zip',...
                      A.ImageDirectory,A.FlowCase{1},A.FlowCase{2},...
                      regexprep(regexprep(regexprep(...
                         A.FlowCase{3},'.','_'),'-','n'),'+','p'));
zip(zipFileName,zipFileList);

fprintf('\nProgram Complete.\nTotal Runtime: %6.4f seconds\n\n',...
        etime(clock,start));

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end % of function:  steadyAnalysis
%% Function Name:   calc_FS_Profile   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Author:          Lance Case
%  Date Created:    02.06.2012
%  Last Modified:   02.07.2012
%
%  Purpose:         This function obtains the similarity solution ...
   for flow
%                   over a flat plate, or the "Blassius" solution, by
%                   solving the Falkner-Skan ODE using order-reduction
%                   techniques to obtain a vector of first-order ODEs,
%                   which are then input together into MATLABs ode45()
%                   function.  Cubic spline interpolation is then ...
   performed
%                   on the output from ode45() to obtain results ...
   conforming
%                   to the required solution vector length specified ...
   by the
%                   user in the input variable 'usteps'.
%
%                   The Falkner-Skan ODE:
%                         f''' + f f''( (M+1)/2 ) - M( 1-[f']^2 ) = 0
%
%  Inputs:          Beta     - The full wedge angle, in degrees.
%                   etamax   - The maximum size for the similarity
%                              variable, eta.
%                   etasteps - The number of steps in eta to use in ...
   ode45()
%                   uspan    - The vector of dimensionless velocity ...
   values,
%                              u, at which the user desires to calculate
%                              the corresponding eta
%
%  Outputs:         z        - A matrix of four columns:
```

```
%                   z(:,1)   - The dimensionless velocity, u.
%                   z(:,2)   - The first derivative of f, w.r.t. eta.
%                   z(:,3)   - The similaritiy variable, eta.
%                   z(:,4)   - The third derivative of f, w.r.t. eta.
%
%  Ignore MATLAB Errors:
%#ok<*INUSD,*STOUT,*INUSL>
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
function z = calc_FS_Profile(Beta,etamax,etasteps,uspan)

% Variable Initialization
global M;
Beta      = Beta*pi/180;              % Convert Full Wedge Angle to ...
   Radians
M         = Beta/(2*pi-Beta);         % Compute    the value  M
f_initial = [0;0;0.3595];             % Initialize the vector f

etaspan   = linspace(0,etamax,etasteps);  % Create the eta vector
% uspan     = linspace(0,1,usteps);         % Create the  u  vector

FS_Profile = ode45(@calc_fprime,etaspan,f_initial);

z(:,1) = uspan;     % Dimensionless velocity vector
z(:,2) = interp1(FS_Profile.y(1,:),FS_Profile.x,uspan','spline');
z(:,3) = interp1(FS_Profile.y(2,:),FS_Profile.x,uspan','spline');
z(:,4) = interp1(FS_Profile.y(3,:),FS_Profile.x,uspan','spline');
end % of function: calc_FS_Profile

%% Function Name:   calc_fprime    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Author:          Lance Case
%  Date Created:    02.06.2012
%  Last Modified:   02.06.2012
%
%  Purpose:         This function computes a vector of increasingly
%                   higher-order derivatives of the vector, f, ...
   described in
%                   the Falkner-Skan ODE.
%
%  Inputs:          n         - The similarity variable, eta.
%                   f         - The similarity vector, f, described in
%                               the Falkner-Skan ODE.
%
%                   The Falkner-Skan ODE:
%                         f''' + f f''( (M+1)/2 ) - M( 1-[f']^2 ) = 0
%
%  Outputs:         fprime    - A vector of three rows:
%                   fprime(1) - The first  derivative of f, w.r.t. eta.
%                   fprime(2) - The second derivative of f, w.r.t. eta.
%                   fprime(3) - The third  derivative of f, w.r.t. eta.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
```

```
function fprime = calc_fprime(n,f)

% Variable Initialization
global M;
fprime = zeros(3,1);

fprime(1) = f(2);
fprime(2) = f(3);
fprime(3) = -f(3)*f(1)*((M+1)/2)-M*(1-f(2)^2);
end % of function: calc_fprime
```

APPENDIX B

Unsteady Fiber Analysis MATLAB® Code

```
%% Filename:            transientAnalysis.m
%  Author:              Lance Case
%  Date Last Modified:  03.14.2012
%
%  Purpose:  This file serves as the main executable for the MATLAB
%            transient fiber image analysis program.
%
%  Inputs:   varargin -  Traditionally-optional MATLAB function
%                        arguments
%
%               varargin{1} -   The text file containing all of the
%                               function variables to be used
%                               during analysis.  If not included,
%                               the function will instruct the UI
%                               to prompt the user for the variable
%                               file.
%
%               Variables included in this text file:
%                   PlateName       - Name of the test plate used
%                   ElementName     - Name of the test element
%                                     under observation
%                   FlowAtkAng      - Angle of Attack of test plate
%                   BaseFileName    - Base image file name
%                   CameraFPS       - Camera recording framerate
%                                     for image set
%                   PlaybackFPS     - Created movie playback
%                                     framerate
%                   CamBitDepth     - Camera bit resolution used
%                   NumNoFlowImgs   - Number of 'NoFlow' control
%                                     images in the image set
%                   NumCLAHExTiles  - Number of tiles to use in the
%                                     x-dimension during CLAHE
%                   NumCLAHEyTiles  - Number of tiles to use in the
%                                     y-dimension during CLAHE
%                   NumFirstGoodImg - Number of the first valid
%                                     image in the image set
%                   NumLastGoodImg  - Number of the last valid
%                                     image in the image set
%                   BinarizeThresh  - Threshhold integer used
%                                     during image binarization
%                   BoundBoxXS      - X-coordinate for left side
%                                     of fiber tip interrogation
%                                     region boundary box
%                   BoundBoxXF      - X-coordinate for right side
%                                     of fiber tip interrogation
%                                     region boundary box
%                   BoundBoxYS      - Y-coordinate for top side
%                                     of fiber tip interrogation
%                                     region boundary box
%                   BoundBoxYF      - Y-coordinate for bottom side
%                                     of fiber tip interrogation
%                                     region boundary box
%                   changeBoundsXS  - Boundary Box changer-integer.
%                   changeBoundsXF  - Specify both frame number and
```

```
%                   changeBoundsYS  - pixel number, tab delimited.
%                   changeBoundsYF  - Same XS-YF format as above.
%                   BSLocX          - X-coordinate for bright spot
%                                     interrogation region boundary
%                                     box
%                   BSLocY          - Y-coordinate for bright spot
%                                     interrogation region boundary
%                                     box
%                   BSLocSz         - Pixel size of bright spot
%                                     interrogation region boundary
%                                     box
%                   OvrDispBBoxLoc  - Boolean in Overlay figure
%                                     controlling display of
%                                     fiber tip boundary box
%                   OvrDispBSLoc    - Boolean in Overlay figure
%                                     controlling display of
%                                     bright spot boundary box
%                   OvrDispFooter   - Boolean in Overlay figure
%                                     controlling display of
%                                     footer info
%                   OvrDispDeflInfo - Boolean in Overlay figure
%                                     controlling display of
%                                     fiber tip deflection info
%                   OvrDispFiberTip - Boolean in Overlay figure
%                                     controlling display of
%                                     fiber tip location marker
%                   RunCLAHE        - Boolean controlling the CLAHE
%                   meanIntens      - Mean intensity value in image
%                                     set, determined from previous
%                                     function run on same image
%                                     set
%                                     (OPTIONAL)
%                   CCRClipMax      - Bright spot maximum value for
%                                     use during Cross-Correlation
%                   CCRClipMin      - Bright spot minimum value for
%                                     use during Cross-Correlation
%                   Filtering       - String determining type of
%                                     image filtering used during
%                                     filtering step.
%                                     (OPTIONAL, MULTIPLE ALLOWED)
%                   DispImgRaw      - Boolean controlling display
%                                     of raw image figure
%                   DispImgAHE      - Boolean controlling display
%                                     of CLAHE figure
%                   DispImgBin      - Boolean controlling display
%                                     of binarization figure
%                   DispImgFil      - Boolean controlling display
%                                     of filtered image figure
%                   DispImgInv      - Boolean controlling display
%                                     of inverted image figure
%                   DispImgRP       - Boolean controlling display
%                                     of region props figure
%                   DispImgOvr      - Boolean controlling display
%                                     of overlay figure
```

```
%                   DispImgCCR      - Boolean controlling display
%                                     of cross-correlation figure
%                   DispImgDef      - Boolean controlling display
%                                     of fiber tip deflection graph
%                   DispImgAng      - Boolean controlling display
%                                     of fiber relative angle graph
%                   DispImgLac      - Boolean controlling display
%                                     of 'fiber video / deflection
%                                     graph / cross-correlation'
%                                     interlaced figure
%                   CreatMovRaw     - Boolean controlling creation
%                                     of raw movie file
%                   CreatMovAHE     - Boolean controlling creation
%                                     of CLAHE movie file
%                   CreatMovBin     - Boolean controlling creation
%                                     of binarization movie file
%                   CreatMovFil     - Boolean controlling creation
%                                     of filtering movie file
%                   CreatMovInv     - Boolean controlling creation
%                                     of inverted image movie file
%                   CreatMovRP      - Boolean controlling creation
%                                     of region props movie file
%                   CreatMovOvr     - Boolean controlling creation
%                                     of Overlay movie file
%                   CreatMovCCR     - Boolean controlling creation
%                                     of cross-correlation movie
%                   CreatMovDef     - Boolean controlling creation
%                                     of fiber tip deflection graph
%                                     movie file
%                   CreatMovAng     - Boolean controlling creation
%                                     of relative fiber angle graph
%                                     movie file
%                   CreatMovLac     - Boolean controlling creation
%                                     of raw movie file
%                   MovSkipFrm      - Frame count to skip during
%                                     playback, to reduce file size
%
%                   Variables in this text file should be formatted
%                   as follows ('<tab>' denotes tab character):
%
%                   #BOF
%                   VariableNameOne<tab>VariableValueOne
%                   VariableNameTwo<tab>VariableValueTwo
%                   ========================================
%                   Comments can be included on separate lines
%                   ========================================
%                   VariableNameThree<tab>VariableValueThree
%                   ...
%                   #EOF
%
%  Outputs:         The variable B, a structure containing all of
%                   the pertinent information about the current
%                   image analysis
%
```

```
% Ignore MATLAB Errors:
%#ok<*SAGROW,*NASGU,*TNMLP,*FNDSB,*ISMT,*ST2NM,*ASGLU>
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function B = transientAnalysis(varargin)
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     MATLAB Program Initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Clear everything from memory
format compact;
% clear all; clc;

% Remove excess grey borders from MATLAB figures
iptsetpref('ImshowBorder','tight');

% Supress MATLAB Warnings:
% warning off Images:imshow:magnificationMustBeFitForDockedFigure;
warning off all;

% Open up a pool of parallel labs for efficiency
% if ~matlabpool('size'), matlabpool(3); end


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                             Initialize variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Display variable initialization to command window
clc; fprintf('Initializing variables...\n');

% Directory Information
% (Where the matlab image analysis m-file is stored)
B.MATLABdir = pwd;

% Pixel Width
% PW = (2/32)/(973-256)*25400;          % um per pixel  (Scienscope)
PW = 13.344;                            % um per pixel    (Phantom)

% Initialize Variables
B.Filtering = {};

% Accept variables file as an optional input
if length(varargin)>0
    varFile = varargin{1};
else
    varFile = '';
end

% Variables file containing image reduction information
if isempty(varFile)
    [A.varFile,dummy,nextdummy] = uigetfile({'*.txt','*.dat'});
    B.varFile = sprintf('%s\\%s',dummy,A.varFile);
```

```
    clear dummy; clear nexdummy;
else
    B.varFile = varFile;
end

% Handle 'Variable File Does Not Exist' Exception
if ~exist(B.varFile,'file'), error('No such file exists.'); end;

% Start the timer
start = clock;

% Open the variables input file to get directory information
variablesFile = fopen(B.varFile,'r');
while ~feof(variablesFile)
    newVar = fgets(variablesFile);
    a = textscan(newVar,'%s%s','delimiter','\t');
    b=a{1}; c=a{2}; d = c{1};

    % Set Variables Related to Image File Input
    if strcmp(b,'BaseImgDir')
        B.ImageDirectory = d;
        e = textscan(B.ImageDirectory,'%s','Delimiter','\\');
        f = e{length(e)}; B.InputFilename{1} = f{length(f)};
        clear e; clear f;
    end
    if strcmp(b,'BaseFileType'),    B.InputFilename{2} = d; end;
end; fclose(variablesFile);

% Get total number of images in the set
B.NumCurImg = 0;
B.N = length(dir(sprintf('%s\\*.%s',...
                B.ImageDirectory,B.InputFilename{2})));

% Initialize boundary box change matrix
B.changeBounds = zeros(4,B.N);

% Initialize Mean Intensity for current flow case
B.meanIntens = [];

% Open the variables input file to get most of the other variables
variablesFile = fopen(B.varFile,'r');
while ~feof(variablesFile)
    newVar = fgets(variablesFile);
    a = textscan(newVar,'%s%s','delimiter','\t');
    b=a{1}; c=a{2}; d = c{1};

    % Set Variables Related to Flow Case
    if strcmp(b,'PlateName'),       B.FlowCase{1} = d; end;
    if strcmp(b,'ElementName'),     B.FlowCase{2} = d; end;
    if strcmp(b,'FlowAtkAng'),      B.FlowCase{3} = d; end; % AoA
    if strcmp(b,'BaseFileName'),    B.FlowCase{4} = d; end; % Veloc

    % Set Variables Related to Video Capture Information
    if strcmp(b,'CameraFPS'),       B.Video(1) = str2num(d); end;
```

```
if strcmp(b,'PlaybackFPS'),      B.Video(2) = str2num(d); end;
if strcmp(b,'CamBitDepth'),      B.Video(3) = str2num(d); end;

% Set Important Integer Variables
if strcmp(b,'NumNoFlowImgs'),    B.NumNoFlowImgs     = ...
   str2num(d); end;
if strcmp(b,'NumCLAHExTiles'),   B.NumCLAHETiles(1)  = ...
   str2num(d); end;
if strcmp(b,'NumCLAHEyTiles'),   B.NumCLAHETiles(2)  = ...
   str2num(d); end;
if strcmp(b,'NumFirstGoodImg'), B.NumFirstGoodImg   = ...
   str2num(d); end;
if strcmp(b,'NumLastGoodImg'),   B.NumLastGoodImg    = ...
   str2num(d); end;
if strcmp(b,'BinarizeThresh'),   B.BinarizeThresh    = ...
   str2num(d); end;

% Set Variables Related to Interrogation Region Locations
if strcmp(b,'BoundBoxXS'),    B.BoundBox(1)    = str2num(d); end;
if strcmp(b,'BoundBoxXF'),    B.BoundBox(2)    = str2num(d); end;
if strcmp(b,'BoundBoxYS'),    B.BoundBox(3)    = str2num(d); end;
if strcmp(b,'BoundBoxYF'),    B.BoundBox(4)    = str2num(d); end;
if strcmp(b,'BSLocX'),        B.BSLoc(1)       = str2num(d); end;
if strcmp(b,'BSLocY'),        B.BSLoc(2)       = str2num(d); end;
if strcmp(b,'BSLocSz'),       B.BSLoc(3)       = str2num(d); end;

% Set Overlay Display Options
if strcmp(b,'OvrDispBBoxLoc'),  B.Overlay(1) = str2num(d); end;
if strcmp(b,'OvrDispBSLoc'),    B.Overlay(2) = str2num(d); end;
if strcmp(b,'OvrDispFooter'),   B.Overlay(3) = str2num(d); end;
if strcmp(b,'OvrDispDeflInfo'), B.Overlay(4) = str2num(d); end;
if strcmp(b,'OvrDispFiberTip'), B.Overlay(5) = str2num(d); end;

% Set Miscellaneous Variables
if strcmp(b,'RunCLAHE'),     B.RunCLAHE      = str2num(d); end;
if strcmp(b,'meanIntens'),   B.meanIntens    = ...
   double(str2num(d)); end;

% Set Cross-Correlation Clipping Values
if strcmp(b,'CCRClipMax'),       B.CCRclip(1) = str2num(d); end;
if strcmp(b,'CCRClipMin'),       B.CCRclip(2) = str2num(d); end;

% Set Filtering Variables
if strcmp(b,'Filtering'),  B.Filtering{length(B.Filtering)+1} = ...
   d; end;

% Set Variables About Figure Display and Movie Creation
if strcmp(b,'DispImgRaw'),   B.DisplayImage(1)  = str2num(d); end;
if strcmp(b,'DispImgAHE'),   B.DisplayImage(2)  = str2num(d); end;
if strcmp(b,'DispImgBin'),   B.DisplayImage(3)  = str2num(d); end;
if strcmp(b,'DispImgFil'),   B.DisplayImage(4)  = str2num(d); end;
if strcmp(b,'DispImgInv'),   B.DisplayImage(5)  = str2num(d); end;
if strcmp(b,'DispImgRP'),    B.DisplayImage(6)  = str2num(d); end;
if strcmp(b,'DispImgOvr'),   B.DisplayImage(7)  = str2num(d); end;
```

```
    if strcmp(b,'DispImgCCR'),  B.DisplayImage(8)  = str2num(d); end;
    if strcmp(b,'DispImgDef'),  B.DisplayImage(9)  = str2num(d); end;
    if strcmp(b,'DispImgAng'),  B.DisplayImage(10) = str2num(d); end;
    if strcmp(b,'DispImgLac'),  B.DisplayImage(11) = str2num(d); end;
    if strcmp(b,'CreatMovRaw'), B.CreateMovie(1)   = str2num(d); end;
    if strcmp(b,'CreatMovAHE'), B.CreateMovie(2)   = str2num(d); end;
    if strcmp(b,'CreatMovBin'), B.CreateMovie(3)   = str2num(d); end;
    if strcmp(b,'CreatMovFil'), B.CreateMovie(4)   = str2num(d); end;
    if strcmp(b,'CreatMovInv'), B.CreateMovie(5)   = str2num(d); end;
    if strcmp(b,'CreatMovRP'),  B.CreateMovie(6)   = str2num(d); end;
    if strcmp(b,'CreatMovOvr'), B.CreateMovie(7)   = str2num(d); end;
    if strcmp(b,'CreatMovCCR'), B.CreateMovie(8)   = str2num(d); end;
    if strcmp(b,'CreatMovDef'), B.CreateMovie(9)   = str2num(d); end;
    if strcmp(b,'CreatMovAng'), B.CreateMovie(10)  = str2num(d); end;
    if strcmp(b,'CreatMovLac'), B.CreateMovie(11)  = str2num(d); end;
    if strcmp(b,'MovSkipFrm'),  B.MovieFrameSkip   = str2num(d); end;

    % Set Variables Related to changing the boundary box during analysis
    if length(c) > 1
        e = b{2};
        if      strcmp(b{1},'changeBoundsXS')
            B.changeBounds(1,str2num(d)) = str2num(e);
        elseif  strcmp(b{1},'changeBoundsXF')
            B.changeBounds(2,str2num(d)) = str2num(e);
        elseif  strcmp(b{1},'changeBoundsYS')
            B.changeBounds(3,str2num(d)) = str2num(e);
        elseif  strcmp(b{1},'changeBoundsYF')
            B.changeBounds(4,str2num(d)) = str2num(e);
        end
    end

end; fclose(variablesFile);

% Preserve these original boundary values
B.origBounds = B.BoundBox;

% Initialize Figure and Movie Stuff
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Set the figure names
B.FigName{1}  = 'Raw';
B.FigName{2}  = 'CLAHE';
B.FigName{3}  = 'Binarized';
B.FigName{4}  = 'Filtered';
B.FigName{5}  = 'Inverted';
B.FigName{6}  = 'Region Props';
B.FigName{7}  = 'Overlay';
B.FigName{8}  = 'Cross-Correlation';
B.FigName{9}  = 'Fiber Deflection';
B.FigName{10} = 'Fiber Orientation';
B.FigName{11} = 'Fiber-Graph-Interlaced';

fighandle = zeros(1,length(B.DisplayImage));  % Figure handle array
imgsz     = zeros(length(B.DisplayImage),2);  % Image size matrix
```

```
buf       = [10 0]; % px ct of buffer between figures [horz vert]
bot       = 70; % pixel location from bottom of screen
lft       = 3;  % pixel location from left of screen

if B.NumLastGoodImg, B.EndImg = B.NumLastGoodImg;
else                 B.EndImg = B.N;
end

% Initialize all necessary figures
for i=1:length(B.DisplayImage)

    % Close each figure
    close(figure(i));

    if B.DisplayImage(i) || B.CreateMovie(i)
        % Create raw figure
        fighandle(i) = figure(i);
        set(gcf,'WindowStyle','Normal');
        set(gcf,'Toolbar','None');
        set(gcf,'DockControls','off');
        set(gcf,'MenuBar','None');
        set(gcf,'Resize','off');
        set(gcf,'NumberTitle','off');
        if i ~= 11, set(gcf,'Color',[1 1 1]); end;
    end
end

% Find an open figure
openFig = find(fighandle>0);
if ~isempty(openFig), openFig = openFig(1); end;

% Initialize movie files
if B.CreateMovie(1)
mov_raw = avifile(sprintf('%s\\%s_%s.avi',...
                  B.ImageDirectory,B.InputFilename{1},...
                  B.FigName{1}),...
                  'fps',B.Video(2),'compression','None');
end

% Still haven't added all of the movie file initializations for
% less-common movie types (filtering, binarization, CLAHE, etc.)
% but they go here

if B.CreateMovie(7)
mov_overlay = avifile(sprintf('%s\\%s_%s.avi',...
              B.ImageDirectory,B.InputFilename{1},B.FigName{7}),...
              'fps',B.Video(2),'compression','None');
end

% Initialize the elapsed time
timeElapsed = -1/B.Video(1);

testCase = [B.FlowCase{1} ' ' B.FlowCase{2} ' ' B.FlowCase{3},...
            '^\circ ' B.InputFilename{1}];
```

```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Build the cross-correlation reference image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

CCRrefImg = double(zeros(B.BSLoc(3)+1,B.BSLoc(3)+1));
TIPrefImg = double(zeros(length(B.BoundBox(3):B.BoundBox(4)),...
                         length(B.BoundBox(1):B.BoundBox(2))));

% Loop over all images to be used in creating the CCR average image
for i=B.NumFirstGoodImg:(B.NumFirstGoodImg + B.NumNoFlowImgs)

    B.NumCurImg = i;
    B.Filename = sprintf('%s\\%s_%04i.%s',...
                    B.ImageDirectory,  B.InputFilename{1},...
                    B.NumCurImg,       B.InputFilename{2});
    clc;
    fprintf(...
        'Reading cross-correlation image number: %04i/%04i\n',...
            B.NumCurImg,B.NumNoFlowImgs+B.NumFirstGoodImg);
    fprintf('Runtime: % 8.3f s\n',etime(clock,start));

    % Read in the current image based on the current image filename
    refImgRaw = imread(B.Filename);

    % Sum the images together
    if size(refImgRaw,3) > 1  % If theres a 3rd dim'n to the image
                              % matrix, it is RGB and not greyscale
    CCRrefImg = CCRrefImg + double(rgb2gray(refImgRaw(...
          B.BSLoc(2)-floor(B.BSLoc(3)/2):...
          B.BSLoc(2)+floor(B.BSLoc(3)/2),...
          B.BSLoc(1)-floor(B.BSLoc(3)/2):...
          B.BSLoc(1)+floor(B.BSLoc(3)/2),...
                                    :)));

    TIPrefImg = TIPrefImg + double(rgb2gray(refImgRaw(...
          B.BoundBox(3):B.BoundBox(4),...
          B.BoundBox(1):B.BoundBox(2),:)));
    else
    CCRrefImg = CCRrefImg + double(refImgRaw(...
          B.BSLoc(2)-floor(B.BSLoc(3)/2):...
          B.BSLoc(2)+floor(B.BSLoc(3)/2),...
          B.BSLoc(1)-floor(B.BSLoc(3)/2):...
          B.BSLoc(1)+floor(B.BSLoc(3)/2),...
                                    :));

    TIPrefImg = TIPrefImg + double(refImgRaw(...
          B.BoundBox(3):B.BoundBox(4),...
          B.BoundBox(1):B.BoundBox(2),:));
    end
end

% Create the cross-correlation reference image
```

```
B.CCRrefImg = uint8(floor(CCRrefImg/B.NumNoFlowImgs));
B.TIPrefImg = uint8(floor(TIPrefImg/B.NumNoFlowImgs));
clear refImgRaw CCRrefImg TIPrefImg;

% Get mean and max values of the cross-correlation reference image
meanRefImg = mean(mean(B.CCRrefImg));
maxRefImg  = max(max(B.CCRrefImg));

% Initialize the matrix that will contain the clipped reference img
refImgSBC = double(zeros(size(B.CCRrefImg)));

% Loop over the image
for i=1:size(B.CCRrefImg,1)
for j=1:size(B.CCRrefImg,2)

    % Create normalized image
    refImgSBC(i,j) = (B.CCRrefImg(i,j)-meanRefImg)/...
                     (maxRefImg-meanRefImg);

    % Clip normalized image
    if refImgSBC(i,j) > B.CCRclip(1)
        refImgSBC(i,j) = B.CCRclip(1);
    end
    if refImgSBC(i,j) < B.CCRclip(2)
        refImgSBC(i,j) = B.CCRclip(2);
    end

end
end

B.refImgSBC = refImgSBC;

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Do entire fiber analysis outlined below for composite TIPrefImg
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Get the raw image
image = B.TIPrefImg;

% Perform a CLAHE on the image
if B.RunCLAHE
image = adapthisteq(image,'Distribution','rayleigh',...
                          'Range','full',...
                          'NBins',256,'NumTiles',B.NumCLAHETiles);
end

% Binarize the image using the threshhold specified
image = im2bw(image,B.BinarizeThresh/255);

% Use filters to remove random particles
for j=1:length(B.Filtering)
    image = bwmorph(image,B.Filtering{j});
end
```

```
% Invert image to use with regionprops
image = 1 - image;

% Get connected component information
CC = bwconncomp(image,8);

% Use Regionprops on the connected component structure
RP = regionprops(CC,'Area','BoundingBox','Image','Orientation');

% Index fiber area information to search for max
fiber_area =        zeros(length(RP),1);           % Initialization
for j=1:length(RP), fiber_area(j) = RP(j).Area; end   % Indexing

% Pull off largest fiber (the actual carbon fiber, using max area)
[fiber_maxarea,fiber_index] = max(fiber_area);
fiber_boundbox              = floor(RP(fiber_index).BoundingBox);
fiber_orientation           = RP(fiber_index).Orientation;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Here is the modification:
%   The pixels lying along the BoundBox line (both dimensions) are
%   counted twice - once in fiber_boundbox and once in B.BoundBox -
%   so subtracting 1 off of this sum should correct the bias.
%   MOD: 09.52.01.20.2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Consider the tip as the *top*-left point from Regionprops
initial_x_tipLocation = fiber_boundbox(1) + B.BoundBox(1) - 1;
initial_y_tipLocation = fiber_boundbox(2) + B.BoundBox(3) - 1;

% If the fiber angle is >0 (bot-left to top-right orientation)
if fiber_orientation > 0
    % Consider the tip as the *bottom*-left point from Regionprops
    % by adding on the Regionprops height of the fiber to
    % y_topLocation
    initial_y_tipLocation = ...
        initial_y_tipLocation + fiber_boundbox(4);
end

clear fiber_area fiber_boundbox fiber_orientation;

% Initialize fiber property vectors and matrices
% for the main program loop
fiber_area          = zeros(B.N,1);
fiber_centroid      = zeros(B.N,2);
fiber_boundbox      = zeros(B.N,4);
fiber_majaxlen      = zeros(B.N,1);
fiber_minaxlen      = zeros(B.N,1);
fiber_orientation   = zeros(B.N,1);
x_tipLocation       = zeros(B.N,1);
y_tipLocation       = zeros(B.N,1);
dx                  = zeros(B.N,1);
dy                  = zeros(B.N,1);
dtot                = zeros(B.N,1);
```

```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Find the mean intensity over the entire image set
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if isempty(B.meanIntens)
    meanImgIntens = zeros(B.N,1);

    for i=B.NumFirstGoodImg:B.EndImg

        B.NumCurImg = i;
        B.Filename = sprintf('%s\\%s_%04i.%s',...
                        B.ImageDirectory,  B.InputFilename{1},...
                        B.NumCurImg,       B.InputFilename{2});

        image = imread(B.Filename);

        if size(image,3)>1, hsvImage = rgb2hsv(image);
        else                hsvImage = ...
                                rgb2hsv(repmat(image,[1 1 3]));
        end

        meanImgIntens(i) = mean(mean(hsvImage(:,:,3)));
        clc;
        fprintf(...
            'Calculating mean intensity in image %04i/%04i\n',...
                i,B.N);
        fprintf('Runtime: % 8.3f s\n',etime(clock,start));

    end

    % Calculate the mean of the nonzero 'meanImgIntens' terms
    B.meanIntens = mean(meanImgIntens(find(meanImgIntens)));
end

%% Set the proper boundary box

fprintf('\nNow setting the proper boundary box conditions...\n');

for i=1:B.NumFirstGoodImg
for j=1:4
if B.changeBounds(j,i),B.BoundBox(j)=B.changeBounds(j,i);end;
end
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Loop over each image in the directory
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=B.NumFirstGoodImg:B.EndImg
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Update information for the current loop iteration
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

B.NumCurImg = i;
B.Filename = sprintf('%s\\%s_%04i.%s',...
```

```
                    B.ImageDirectory,  B.InputFilename{1},...
                    B.NumCurImg,       B.InputFilename{2});

% Change boundary box
for j=1:4
    if B.changeBounds(j,i)
        B.BoundBox(j)=B.changeBounds(j,i);
    end
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               Read in new iteration's image for analysis
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Read image
image = imread(B.Filename);

% Convert image to HSI
if size(image,3) > 1,  hsvImage = rgb2hsv(image);
else                   hsvImage = rgb2hsv(repmat(image,[1 1 3]));
end

% Perform a brighness equalization (brighter)
while mean(mean(hsvImage(:,:,3))) < B.meanIntens
    hsvImage(:,:,3) = hsvImage(:,:,3) + 1/255;
end

% Perform a brighness equalization (dimmer)
while mean(mean(hsvImage(:,:,3))) > B.meanIntens
    hsvImage(:,:,3) = hsvImage(:,:,3) - 1/255;
end

% Convert image back to grayscale
image = rgb2gray(hsv2rgb(hsvImage));

rawimg = image;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Output Image and/or Movie

n = 1;  % Figure number

% Display new image
if B.DisplayImage(n) || B.CreateMovie(n)
    imgsz(n,:) = size(image) + buf;
    figure(n); imshow(image);

    set(fighandle(n),'Position',...
                [lft bot imgsz(n,1)-buf(1) imgsz(n,2)-buf(2)],...
            'Name',sprintf('%s (%04i/%04i)',...
                           B.FigName{n},B.NumCurImg,B.N));

end
```

```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Contrast-Limited Adaptive Histogram Equalization (CLAHE)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Perform a CLAHE
if B.RunCLAHE
image = adapthisteq(image,'Distribution','rayleigh',...
                          'Range','full',...
                          'NBins',256,'NumTiles',B.NumCLAHETiles);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Output Image and/or Movie

n = 2;  % Figure number

% Display new image
if B.DisplayImage(n) || B.CreateMovie(n)
    imgsz(n,:) = size(image) + buf;

    figure(n); imshow(image);
    set(fighandle(n),'Position',...
      [lft+sum(imgsz(1:n-1,1)) bot,...
       imgsz(n,1)-buf(1) imgsz(n,2)-buf(2)],...
          'Name',sprintf('%s (%04i/%04i)',...
                         B.FigName{n},B.NumCurImg,B.N));
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              Image Binarization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Binarize the image using the binarization threshhold
image = im2bw(image,B.BinarizeThresh/255);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Output Image and/or Movie

n = 3;  % Figure number

% Display new image
if B.DisplayImage(n) || B.CreateMovie(n)
    imgsz(n,:) = size(image) + buf;

    figure(n); imshow(image);
    set(fighandle(n),'Position',...
      [lft+sum(imgsz(1:n-1,1)) bot,...
       imgsz(n,1)-buf(1) imgsz(n,2)-buf(2)],...
          'Name',sprintf('%s (%04i/%04i)',...
                         B.FigName{n},B.NumCurImg,B.N));
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              Image Filtering
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Use filters to remove random particles
for j=1:length(B.Filtering)
    image = bwmorph(image,B.Filtering{j});
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Output Image and/or Movie

n = 4;  % Figure number

% Display new image
if B.DisplayImage(n) || B.CreateMovie(n)
    imgsz(n,:) = size(image) + buf;

    figure(n); imshow(image);
    set(fighandle(n),'Position',...
      [lft+sum(imgsz(1:n-1,1)) bot,...
       imgsz(n,1)-buf(1) imgsz(n,2)-buf(2)],...
          'Name',sprintf('%s (%04i/%04i)',...
                         B.FigName{n},B.NumCurImg,B.N));
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              Image Inversion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Invert image to use with regionprops
image = 1 - image;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Output Image and/or Movie

n = 5;  % Figure number

% Display new image
if B.DisplayImage(n) || B.CreateMovie(n)
    imgsz(n,:) = size(image) + buf;

    figure(n); imshow(image);
    set(fighandle(n),'Position',...
      [lft+sum(imgsz(1:n-1,1)) bot,...
       imgsz(n,1)-buf(1) imgsz(n,2)-buf(2)],...
          'Name',sprintf('%s (%04i/%04i)',...
                         B.FigName{n},B.NumCurImg,B.N));
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Get Regionprops Information for Fiber Analysis
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Pull off the region of interest from the main image
refImg = image(B.BoundBox(3):B.BoundBox(4),B.BoundBox(1):B.BoundBox(2));
```

```
% figure(length(B.DisplayImage)+1); imshow(refImg);

% Get connected component information
CC = bwconncomp(refImg,8);

% Use Regionprops on the connected component structure
RP = regionprops(CC,'Area','Centroid','BoundingBox','Image',...
                    'MajorAxisLength','MinorAxisLength',...
                    'Orientation');
if isempty(RP), continue; end;

% Index fiber area information to search for max
fiber_area =        zeros(length(RP),1);          % Initialization
for j=1:length(RP), fiber_area(j) = RP(j).Area; end    % Indexing

% Pull off largest fiber (the actual carbon fiber, using max area)
[fiber_maxarea,fiber_index] = max(fiber_area);
fiber_area                  = fiber_area(fiber_index);
fiber_centroid(i,:)         = RP(fiber_index).Centroid;
fiber_boundbox(i,:)         = floor(RP(fiber_index).BoundingBox);
fiber_majaxlen(i)           = RP(fiber_index).MajorAxisLength;
fiber_minaxlen(i)           = RP(fiber_index).MinorAxisLength;
fiber_orientation(i)        = RP(fiber_index).Orientation;

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                         Display Regionprops Image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Save fiber image
base_image                  = zeros(size(refImg));
fiber_image                 = RP(fiber_index).Image;

new_image                   = zeros(size(image));

% Place the fiber image inside of the base image
% in order to conserve original image size
for k=1:size(fiber_image,1)
    for j=1:size(fiber_image,2)

        base_image(k+floor(fiber_boundbox(i,2)),...
                   j+floor(fiber_boundbox(i,1)))...
        = fiber_image(k,j);
    end
end

% Place the fiber image inside of 'new_image'
% in order to conserve original image size
for k=2:size(base_image,1)
    for j=2:size(base_image,2)
        if base_image(k,j)
            new_image(k+B.BoundBox(3)-1,j+B.BoundBox(1)-1) ...
            = base_image(k,j);
        end
    end
```

```
end

image = 1-new_image;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Output Image and/or Movie

n = 6;  % Figure number

% Display new image
if B.DisplayImage(n) || B.CreateMovie(n)
    imgsz(n,:) = size(image) + buf;

    figure(n); imshow(image);
    set(fighandle(n),'Position',...
      [lft+sum(imgsz(1:n-1,1)) bot,...
       imgsz(n,1)-buf(1) imgsz(n,2)-buf(2)],...
          'Name',sprintf('%s (%04i/%04i)',...
                         B.FigName{n},B.NumCurImg,B.N));
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      Overlay Region Props on Raw Image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Sum the raw image and fiber pixel values to create the overlay
image = (rawimg) + (new_image);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Output Image and/or Movie

n = 7;  % Figure number

% Display new image
if B.DisplayImage(n) ||  B.CreateMovie(n)
if B.Overlay(1) || B.Overlay(2)
    imgsz(n,:) = size(image) + buf;

    figure(n); imshow(image);

    % If displaying the bright spot location,
    % place markers on the overlay image
    if B.Overlay(2)
        hold on;
        plot([B.BSLoc(1)-floor(B.BSLoc(3)/2),...
              B.BSLoc(1)-floor(B.BSLoc(3)/2),...
              B.BSLoc(1)+floor(B.BSLoc(3)/2),...
              B.BSLoc(1)+floor(B.BSLoc(3)/2)],...
             [B.BSLoc(2)-floor(B.BSLoc(3)/2),...
              B.BSLoc(2)+floor(B.BSLoc(3)/2),...
              B.BSLoc(2)-floor(B.BSLoc(3)/2),...
              B.BSLoc(2)+floor(B.BSLoc(3)/2)],'b+');
        plot(B.BSLoc(1),B.BSLoc(2),'ro');
        hold off;
```

```
    end

    % If displaying the bounding box,
    % place markers on the overlay image
    if B.Overlay(1)
        hold on;
        plot([B.BoundBox(1),B.BoundBox(1),...
              B.BoundBox(2),B.BoundBox(2)],...
             [B.BoundBox(3),B.BoundBox(4),...
              B.BoundBox(3),B.BoundBox(4)],...
             'g+');
        hold off;
    end

    if B.Overlay(3)
        % Update time elapsed
        timeElapsed = timeElapsed + 1/B.Video(1);

        % Create footer text
        textLabel = sprintf(...
                        ' %04.0i/%04.0i %05.2fs          .',...
                        i,B.N,timeElapsed);

        % Append footer text to current image
      text(3,imgsz(n,2)-9,[testCase,textLabel],'Color','white',...
           'FontName','FixedWidth','BackgroundColor','black');
    end

    set(fighandle(n),'Position',...
      [lft+sum(imgsz(1:n-1,1)) bot,...
       imgsz(n,1)-buf(1) imgsz(n,2)-buf(2)],...
          'Name',sprintf('%s (%04i/%04i)',...
                         B.FigName{n},B.NumCurImg,B.N));
end
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Perform a cross-correlation on the current image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Read in the curImg, making sure it is in grayscale
if size(rawimg,3) > 1
    curImg = rgb2gray(rawimg(...
          B.BSLoc(2)-floor(B.BSLoc(3)/2):...
          B.BSLoc(2)+floor(B.BSLoc(3)/2),...
          B.BSLoc(1)-floor(B.BSLoc(3)/2):...
          B.BSLoc(1)+floor(B.BSLoc(3)/2),...
          :));
else
    curImg = rawimg(...
          B.BSLoc(2)-floor(B.BSLoc(3)/2):...
          B.BSLoc(2)+floor(B.BSLoc(3)/2),...
          B.BSLoc(1)-floor(B.BSLoc(3)/2):...
          B.BSLoc(1)+floor(B.BSLoc(3)/2),...
```

```
          :);
end

% Take the mean and max values of the current image
meanCurImg = mean(mean(curImg));
maxCurImg  = max(max(curImg));

% Initialize the normalized curImg variable
curImgSBC = double(zeros(size(curImg)));

% Loop over the image
for k=1:size(curImg,1)
for j=1:size(curImg,2)

    % Create normalized image
    curImgSBC(k,j) = (curImg(k,j)-meanCurImg)/...
                     (maxCurImg-meanCurImg);

    % Clip normalized image
    if curImgSBC(k,j) > B.CCRclip(1)
        curImgSBC(k,j) = B.CCRclip(1);
    end
    if curImgSBC(k,j) < B.CCRclip(2)
        curImgSBC(k,j) = B.CCRclip(2);
    end

end
end

% Cross correlation function
CCR = xcorr2(curImgSBC,refImgSBC);

% Match the location of the maximum using MATLABs find()
[yMax,xMax] = find(CCR==max(CCR(:)),1);

% Compute x_shift and y_shift
% Positive x is to the right, positive y is upward.
% This means that positive x is in the ascending col direction
% in the image and also that positive y is in the descending
% row direction in the image
x_shift = -(B.BSLoc(3) + 1) + ...
(xMax +      (CCR(yMax,xMax-1) - CCR(yMax,xMax+1)) / ...
     (2*(CCR(yMax,xMax-1) + CCR(yMax,xMax+1) - 2*CCR(yMax,xMax))));

y_shift = (B.BSLoc(3)  ) - ...
(yMax +      (CCR(yMax-1,xMax) - CCR(yMax+1,xMax)) / ...
     (2*(CCR(yMax-1,xMax) + CCR(yMax+1,xMax) - 2*CCR(yMax,xMax))));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Output Image and/or Movie

n = 8;  % Figure number
```

```
% Display new image
if B.DisplayImage(n) || B.CreateMovie(n)
    imgsz(n,:) = size(CCR) + buf;

    figure(n); imagesc(CCR,[0,20]);%max(CCR(:))]);
    hold on;
    plot(B.BSLoc(3),B.BSLoc(3),'r+'); % Put a red '+' in the center
    hold off;
    colorbar;                         % Add a colorbar
    grid on;

    set(fighandle(n),'Position',...
            [sum(imgsz(1:n-1,1)),bot,...
             max(imgsz(n,1)-buf(1),imgsz(openFig,1)),...
             max(imgsz(n,2)-buf(2),imgsz(openFig,2))],...
            'Name',sprintf('%s (%04i/%04i)',...
                           B.FigName{n},B.NumCurImg,B.N));
%       axis equal;
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Use Regionprops and CCR Information for Fiber Analysis
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Consider the tip as the top-left point from Regionprops
x_tipLocation(i) = fiber_boundbox(i,1) + B.BoundBox(1);
y_tipLocation(i) = fiber_boundbox(i,2) + B.BoundBox(3);

% If the fiber angle is > 0 (bot-left to top-right orientation)
if fiber_orientation(i) > 0
    % Consider the tip as the bottom-left point from Regionprops
    % by adding on the Regionprops height of the fiber to
    % y_tipLocation
    y_tipLocation(i) = y_tipLocation(i) + fiber_boundbox(i,4);
end

% Compute the deflections
dx(i)    = ((x_tipLocation(i)-x_shift) - initial_x_tipLocation)...
           * PW;
dy(i)    = ((y_tipLocation(i)+y_shift) - initial_y_tipLocation)...
           * PW;
dtot(i) = ( dx(i)^2 + dy(i)^2 )^(1/2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Update Overlay Image with dx and dy information

n = 7;
if B.DisplayImage(n) || B.CreateMovie(n)
if B.Overlay(4) || B.Overlay(5)

    figure(n);

    if B.Overlay(5)
        hold on; plot(x_tipLocation(i),y_tipLocation(i),'rx');
```

```
        hold off;
    end

    if B.Overlay(4)

        % Append footer text to current image
        text(3,imgsz(n,2)-(9+14),...
            ['dx: ',sprintf('%+07.2f',dx(i)),...
             ' \mum                 .'],...
            'Interpreter','tex','Color','White',...
            'Background','Black',...
            'FontName','FixedWidth');
        text(floor(imgsz(n,1)/2),imgsz(n,2)-(9+14),...
            ['dy: ',sprintf('%+07.2f',dy(i)),' \mum ',...
'                                                    .'],...
            'Interpreter','tex','Color','White',...
            'Background','Black',...
            'FontName','FixedWidth');

        if B.CreateMovie(n)
            figurepos = get(fighandle(n),'Position');
            F = getframe(fighandle(n),...
                         [0,2,figurepos(3)-1,figurepos(4)-1]);
            mov_overlay = addframe(mov_overlay,F);
        end

    end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Output Image and/or Movie

n = 9;  % Figure number

% Display new image
if B.DisplayImage(n) || B.CreateMovie(n)
    imgsz(n,:) = size(image) + buf;

    figure(n);

    xaxisvector = B.NumFirstGoodImg:i;

    plot(xaxisvector,dx(xaxisvector),'-',...
         xaxisvector,dy(xaxisvector),'-');
    grid on;
    xlabel('Frame');
    ylabel('Deflection [\mum]');
    title('\bf Fiber Deflection [\mum]','FontSize',14);
    legend('dx [\mum]','dy [\mum]');

    axis([B.NumFirstGoodImg,B.N,...
          min(min(dx(xaxisvector)),...
              min(min(dy(xaxisvector)),-100)),...
```

```
            max(max(dx(xaxisvector)),...
                max(max(dy(xaxisvector)), 100))]);

    set(fighandle(n),'Position',...
           [lft bot+max(imgsz(1:n-1,2))+30 500 500],...
           'Name',sprintf('%s (%04i/%04i)',...
                          B.FigName{n},B.NumCurImg,B.N));
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      Output information about the current iteration to the screen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc;
disp(B);
fprintf('\n%s\nRuntime: % 8.3f s\n',...
        B.Filename,etime(clock,start));
fprintf('x_shift: %05.3f\ny_shift: %05.3f\n',x_shift,y_shift);
fprintf('dx:      %05.3f\ndy:      %05.3f\n',dx(i),dy(i));

end % of the main loop over the entire image set
%%

% Get rid of the bias
B.dx = dx - mean(dx(1:B.NumNoFlowImgs));
B.dy = dy - mean(dy(1:B.NumNoFlowImgs));

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create Fiber / Graph Interlaced movie during Post-processing

n = 11;

if B.CreateMovie(n)
    B.BoundBox = B.origBounds;
    mov_interlace = avifile(sprintf('%s\\%s_%s_%sAOA_%s.avi',...
                    B.ImageDirectory,B.FlowCase{1},...
                    B.FlowCase{2},...
                    strrep(strrep(B.FlowCase{3},'+','p'),...
                                                '-','n'),...
                    B.FlowCase{4}),...
                    'fps',B.Video(2)/B.MovieFrameSkip,...
                    'compression','None');
    create_interlaced_figure(B,mov_interlace);
    mov_interlace = close(mov_interlace);
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Wrap up and close out movie files

% Close movie files
if B.CreateMovie(1), mov_raw = close(mov_raw); end;
if B.CreateMovie(7), mov_overlay = close(mov_overlay); end;

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%                 Save the workspace to the image directory
save(sprintf('%s\\MATLAB_analysis_workspace',B.ImageDirectory));
end
```

APPENDIX C

Create Fiber Motion Video MATLAB® Code

```
%% Filename:              create_interlaced_figure.m
%  Author:                Lance Case
%  Date Created:          Unknown
%  Date Last Modified:    03.20.2012
%
%  Purpose:          This file serves to create the interlaced fiber
%                    deflection video and graph movie in AVI format.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function create_interlaced_figure(B,mov_interlace)
%#ok<*NASGU,*ASGLU>
frameSkip = B.MovieFrameSkip;

n = length(B.FigName);
interlacedFigure = figure(n);
set(gcf,'Name','Fiber Deflection Interlaced Video');
set(gcf,'Position',[308,190,560,500]);
set(gcf,'WindowStyle','Normal');
set(gcf,'Toolbar','None');
set(gcf,'DockControls','off');
set(gcf,'MenuBar','None');
set(gcf,'Resize','off');
set(gcf,'NumberTitle','off');

newstart = clock;

light_blue = [0 .5 .7];
colormap(jet(128));
mov_interlace.quality = 10;

%% Set the proper boundary box

fprintf('\nNow setting the proper boundary box conditions...\n');

for i=1:B.NumFirstGoodImg
for j=1:4
if B.changeBounds(j,i),B.BoundBox(j)=B.changeBounds(j,i);end;
end
end


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Perform entire analysis for the image set again
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=B.NumFirstGoodImg:B.EndImg

% Change boundary box
for j=1:4,if ...
    B.changeBounds(j,i),B.BoundBox(j)=B.changeBounds(j,i);end;end

if ~mod(i,frameSkip)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B.NumCurImg = i;
```

```
% Set properties for the top graph
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
bigfig = subplot(2,2,1);
set(bigfig,'Position',[.11,.5838,.86,.3412]);

y_axis_min = min(min(B.dx),min(min(B.dy),-100))-50;
y_axis_max = max(max(B.dx),max(max(B.dy),100))+50;

plot(1:B.N,B.dx,'-','Color',light_blue);        hold on;
plot(1:B.N,B.dy,'g-');
plot(B.NumCurImg,y_axis_min:y_axis_max,'r-');   hold off;
axis([1,B.N,y_axis_min,y_axis_max]);
axis on; grid on;
title('\bf Fiber Deflection [ \mum ]','FontSize',14);
ylabel('Fiber Deflection [ \mum ]');
xlabel('Time [s]');
set(gca,'XTick',1:2*B.Video(1):4*floor(B.N/B.Video(1))*B.Video(1));
set(gca,'XTickLabel',0:2:floor(B.N/(B.Video(1))));
% legend('dx [\mum]','dy [\mum]','Location','SouthWest');

% Raw Image Formation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B.Filename = sprintf('%s\\%s_%04i.%s',...
                     B.ImageDirectory,  B.InputFilename{1},...
                     B.NumCurImg,       B.InputFilename{2});
image = imread(B.Filename);

% Brightness Equalization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if size(image,3) > 1,     hsvImage = rgb2hsv(image);
else                      hsvImage = rgb2hsv(repmat(image,[1 1 3]));
end
while mean(mean(hsvImage(:,:,3))) < B.meanIntens
    hsvImage(:,:,3) = hsvImage(:,:,3) + 1/255;
end
while mean(mean(hsvImage(:,:,3))) > B.meanIntens
    hsvImage(:,:,3) = hsvImage(:,:,3) - 1/255;
end
image = rgb2gray(hsv2rgb(hsvImage));
rawimg = image;     % Save Raw Image

% CLAHE Image Formation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if B.RunCLAHE
image = adapthisteq(image,'Distribution','rayleigh','Range','full',...
                          'NBins',256,'NumTiles',B.NumCLAHETiles);
end

% Image Binarization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
image = im2bw(image,B.BinarizeThresh/255);

% Image Filtering
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j=1:length(B.Filtering), image = bwmorph(image,B.Filtering{j}); end;

% Image Inversion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
image = 1 - image;

% Region Props Calculation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
refImg = image(B.BoundBox(3):B.BoundBox(4),B.BoundBox(1):B.BoundBox(2));
CC = bwconncomp(refImg,8);
RP = regionprops(CC,'Area','BoundingBox','Image','Orientation');
if isempty(RP), continue; end;

% Largest Fiber in RP
fiber_area =        zeros(length(RP),1);
for j=1:length(RP), fiber_area(j) = RP(j).Area; end
[fiber_maxarea,fiber_index] = max(fiber_area);
fiber_boundbox = floor(RP(fiber_index).BoundingBox);

% Region Props Image Formation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
base_image                  = zeros(size(refImg));
fiber_image                 = RP(fiber_index).Image;
new_image                   = zeros(size(image));

for k=1:size(fiber_image,1)
    for j=1:size(fiber_image,2)

        base_image(k+floor(fiber_boundbox(2)),...
                   j+floor(fiber_boundbox(1)))...
        = fiber_image(k,j);
    end
end

for k=2:size(base_image,1)
    for j=2:size(base_image,2)
        if base_image(k,j)
            new_image(k+B.BoundBox(3)-1,j+B.BoundBox(1)-1)...
            = base_image(k,j);
        end
    end
end

% Overlay Image Formation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
image = (rawimg) + (new_image);

% Cross-Correlation Image Formation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if size(rawimg,3) > 1
    curImg = rgb2gray(rawimg(...
       B.BSLoc(2)-floor(B.BSLoc(3)/2):B.BSLoc(2)+floor(B.BSLoc(3)/2),...
       B.BSLoc(1)-floor(B.BSLoc(3)/2):B.BSLoc(1)+floor(B.BSLoc(3)/2),...
```

```
        :));
else
    curImg = rawimg(...
       B.BSLoc(2)-floor(B.BSLoc(3)/2):B.BSLoc(2)+floor(B.BSLoc(3)/2),...
       B.BSLoc(1)-floor(B.BSLoc(3)/2):B.BSLoc(1)+floor(B.BSLoc(3)/2),...
       :);
end
meanCurImg = mean(mean(curImg));
maxCurImg  = max(max(curImg));
curImgSBC  = double(zeros(size(curImg)));
% Create and clip normalized image
for k=1:size(curImg,1)
for j=1:size(curImg,2)
    curImgSBC(k,j) = (curImg(k,j)-meanCurImg)/(maxCurImg-meanCurImg);
    if curImgSBC(k,j) > B.CCRclip(1), curImgSBC(k,j) = B.CCRclip(1); ...
       end;
    if curImgSBC(k,j) < B.CCRclip(2), curImgSBC(k,j) = B.CCRclip(2); ...
       end;
end
end

% Cross-Correlation Computation
CCR = xcorr2(curImgSBC,B.refImgSBC);

% Calculate Time Elapsed
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
timeElapsed = (B.NumCurImg-B.NumFirstGoodImg)/B.Video(1);

% Set properties for the bottom left graph
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ccrfig = subplot(2,2,3);
imagesc(CCR,[0,20]);
bl_cb_handle = colorbar;
title('Cross-Correlation');
% ('Location','NorthOutside');
grid on;
%axis equal;
colormap(jet(64));
% set(gca,'XColor','White','YColor','White');
set(ccrfig,'Position',[.06,.06,.238,.425]);

freezeColors;
cbfreeze(bl_cb_handle);

% Set properties for the bottom right graph
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

overlayfig = subplot(2,2,4);

% Image
imshow(image);
colormap(gray(128));
set(overlayfig,'Position',[.512,0,.539,.49])
```

```
% Bright Spot Markers
hold on;
plot([B.BSLoc(1)-floor(B.BSLoc(3)/2),...
      B.BSLoc(1)-floor(B.BSLoc(3)/2),...
      B.BSLoc(1)+floor(B.BSLoc(3)/2),...
      B.BSLoc(1)+floor(B.BSLoc(3)/2)],...
     [B.BSLoc(2)-floor(B.BSLoc(3)/2),...
      B.BSLoc(2)+floor(B.BSLoc(3)/2),...
      B.BSLoc(2)-floor(B.BSLoc(3)/2),...
      B.BSLoc(2)+floor(B.BSLoc(3)/2)],'b+');
plot(B.BSLoc(1),B.BSLoc(2),'ro');

% Bounding Box Markers
plot([B.BoundBox(1),B.BoundBox(1),B.BoundBox(2),B.BoundBox(2)],...
     [B.BoundBox(3),B.BoundBox(4),B.BoundBox(3),B.BoundBox(4)],...
     'g+');
hold off;

% Set Text Information
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

left = -70;

% Blank
text(left,153,sprintf('%-14s',' '),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');
text(left,168,sprintf('%-14s',' '),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');
text(left,185,sprintf('%-14s',' '),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');
text(left,130,sprintf('%-14s',' '),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');
text(left,113,sprintf('%-14s',' '),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');
text(left,78,sprintf('%-14s',' '),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');
text(left,50,sprintf('%-14s',' '),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');
text(left,25,sprintf('%-14s',' '),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');

% Fiber Deflection Information
text(left,247,['dy: ',sprintf('%+07.2f',B.dy(B.NumCurImg)),' \mum'],...
    'Color','Green','BackgroundColor','Black','FontName','FixedWidth');
text(left,226,['dx: ',sprintf('%+07.2f',B.dx(B.NumCurImg)),' \mum'],...
    'Color',light_blue,'BackgroundColor','Black',...
    'FontName','FixedWidth');
text(left,205,sprintf('%-14s','Deflection:'),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');

% Time Elapsed Information
text(left,170,sprintf('%05.2f s%-7s',timeElapsed,' '),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');
```

```
text(left,148,sprintf('%-14s','Time Elapsed:'),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');

% Frame Number Information
text(left,120,sprintf('%04.0i/%04.0i%-5s',B.NumCurImg,B.N,' '),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');
text(left,98,sprintf('%-14s','Frame Number:'),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');

% Flow Case Information
text(left,11,sprintf('%-14s','Flow Case:'),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');
text(left,38,sprintf('%-14s','. '),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');
text(left,34,[sprintf('%-5s %-3s %-2s',...
                      B.FlowCase{1},B.FlowCase{2},B.FlowCase{3}),...
              '^\circ'],...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');
text(left,61,sprintf('%-14s',B.FlowCase{4}),...
    'Color','White','BackgroundColor','Black','FontName','FixedWidth');

text(70,240,'\rightarrow','Color',light_blue,'FontWeight','bold');
text(66,235,'\uparrow','Color','Green','FontWeight','bold');

figure(n);

if B.CreateMovie(n)
    figurepos = get(gcf,'Position');
    F = getframe(gcf,[0,2,figurepos(3)-1,figurepos(4)-1]);
    mov_interlace = addframe(mov_interlace,F);
end

clc;
fprintf('Now performing Interlaced Image Movie Generation\n');
disp(B);
fprintf('Program Runtime: %06.2f s',etime(clock,newstart));
end
end
fprintf('\n\n');
end
```

APPENDIX D

Fiber Deflection Prediction MathCAD Code

The similarity solution for flow over a flat Plate, or "Blassius" solution, may be developed by integrating the ODE numerically using MathCAD's Runge-Kutta integrator.

To begin the integration, the initial conditions vector and the derivative vector are entered. Note: beta is the wedge half-angle; not the wedge half-angle as is alternatively used.

$$\beta := \frac{2.5}{180} \qquad M := \frac{\beta}{2-\beta} = 6.993 \times 10^{-3}$$

$$f := \begin{pmatrix} 0 \\ 0 \\ 0.48735 \end{pmatrix} \qquad D(\eta, f) := \begin{bmatrix} f_1 \\ f_2 \\ -f_2 \cdot f_0 - \beta \cdot \left[1 - (f_1)^2\right] \end{bmatrix}$$

0.3595	4.752	0.48735	3.44	(-)
0.3035	5.04	0.45125	3.506	(+)

p

$\eta_\delta := 3.44$ $\quad x := 88.9mm$ $\quad h_f := 3.01 \cdot mm$ $\quad h_m := 0.18 \cdot mm$ $\quad \alpha_f := 101.95 \cdot deg$ $\quad \beta_f := 83.06deg$

FileName := "NoE_E3_nAOA_alt_revised.txt"

The domain (largest value of eta) and the number of steps in the domain are entered.

$\eta_{max} := 10$ $\qquad N_r := 1000$

$z := rkfixed(f, 0, \eta_{max}, N_r, D)$

The result of the integration is shown below. To the right, the velocity vector and the eta vector are extracted from the integration result.

$z =$

	0	1	2	3
0	0	0	0	0.4873
1	0.01	$2.4365 \cdot 10^{-5}$	$4.8728 \cdot 10^{-3}$	0.4872
2	0.02	$9.7451 \cdot 10^{-5}$	$9.7442 \cdot 10^{-3}$	0.4871
3	0.03	$2.1924 \cdot 10^{-4}$	0.0146	...

$uoU := z^{\langle 2 \rangle}$

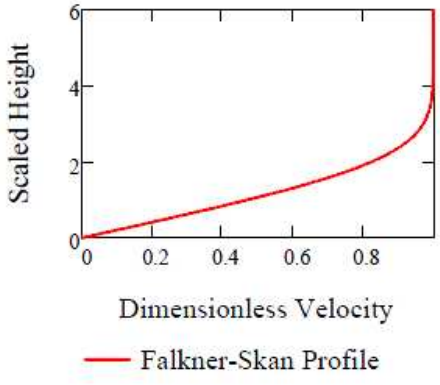
$\eta := z^{\langle 0 \rangle}$

$i := 0 .. N_r$

$intu := z^{\langle 1 \rangle}$

$df := z^{\langle 3 \rangle}$

The resulting profile is plotted below.



$x := 101.6 \cdot mm$ $\rho := 1.1934 \frac{kg}{m^3}$ $\mu := 1.8807 \times 10^{-5} \frac{kg}{m \cdot s}$ $ch := 121.94 \cdot mm$ $U_{ch} := 2 \cdot \frac{m}{s}$

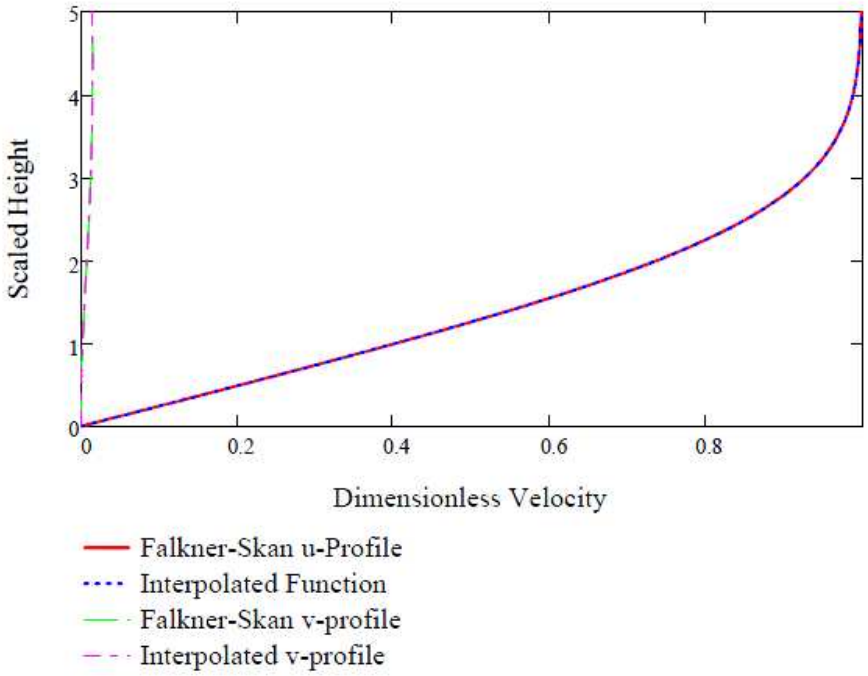
$U = Cx^M$ $Cm := \frac{U_{ch}}{\left(\frac{ch}{mm}\right)^M} = 1.934 \frac{m}{s}$ $\nu := \frac{\mu}{\rho} = 1.576 \times 10^{-5} \frac{m^2}{s}$

$U_x := Cm \cdot \left(\frac{x}{mm}\right)^M = 1.996 \frac{m}{s}$ $Rex := \frac{\rho \cdot U_x \cdot x}{\mu} = 1.126 \times 10^4$

$y := \eta \cdot \left(\frac{M+1}{2} \cdot \frac{U_x}{\nu \cdot x}\right)^{-\frac{1}{2}}$ $u := U_x \cdot uoU$ $\delta_{bl} := \eta_\delta \cdot \left(\frac{M+1}{2} \cdot \frac{U_x}{\nu \cdot x}\right)^{-\frac{1}{2}} = 4.062 \cdot mm$

$v_i := -\left(\frac{2}{M+1} \cdot \frac{\nu \cdot U_x}{x}\right)^{0.5} \cdot \left[intu_i + \frac{(M-1)}{(M+1)} \cdot \eta_i \cdot uoU_i\right]$

$ui(y2) := interp(cspline(y, u), y, u, y2)$ $vi(y2) := interp(cspline(y, v), y, v, y2)$



Fiber Geometric Properties

$d_f := 6.8 \cdot \mu m = 6.8 \times 10^{-6}\, m$ $\quad I_f := \frac{\pi \cdot d_f^{\,4}}{64} = 104.956 \cdot \mu m^4$ $\quad Re_d := \frac{\rho \cdot U_x \cdot d_f}{\mu} = 0.861$

$h_f := 3.424 \cdot mm$ $\quad h_m := 0.15 \cdot mm$ $\quad \frac{h_f}{d_f} = 442.647$ $\quad \alpha_f := 91.8 \cdot deg$ $\quad \beta_f := 90.0 \cdot deg$

Fiber unit direction vector and flow vector are created.The function for determining the the normal component of the flow that causes drag force is also created.

$$P := \begin{pmatrix} \sin(\alpha_f)\cdot\cos(\beta_f) \\ \sin(\alpha_f)\cdot\sin(\beta_f) \\ \cos(\alpha_f)\cdot\sin(\beta_f) \end{pmatrix} \qquad P = \begin{pmatrix} 0.118 \\ 0.971 \\ -0.206 \end{pmatrix} \qquad uv(y) := \begin{pmatrix} ui(y) \\ vi(y) \\ 0\cdot\frac{m}{s} \end{pmatrix} \qquad l_f := \frac{h_f}{(\sin(\alpha_f)\cdot\sin(\beta_f))} = 3.099 \cdot mm$$

$$un(y) := \left[uv(y)\cdot uv(y) - (P\cdot uv(y))^2\right]^{0.5}$$

The freestream flow vector is entered below. The deflection of the fiber is assumed to be in the normal component direction of the freestream flow relative to the fiber direction. The freestream flow direction, the normal component vector, and the unit normal component vector are evaluated below.

$$Uv := \begin{pmatrix} u_{800} \\ v_{800} \\ 0 \end{pmatrix} \qquad \Delta v := Uv - (Uv\cdot P)\cdot P = \begin{pmatrix} 1.964 \\ -0.227 \\ 0.054 \end{pmatrix}\frac{m}{s} \qquad Uv = \begin{pmatrix} 1.996 \\ 0.029 \\ 0 \end{pmatrix}\frac{m}{s}$$

$$\Delta vn := \frac{\Delta v}{\sqrt{\Delta v\cdot\Delta v}} = \begin{pmatrix} 0.993 \\ -0.115 \\ 0.027 \end{pmatrix}$$

Fiber Material Properties

$E := 255GPa$

Initial guess of shear force on fiber and moment about base. $\quad \mu N := \frac{N}{1000000}$ $\quad l_1 := 0.0\mu m$

$$Cd(y) := \exp\left(-0.67\cdot\ln\left(\frac{\rho\cdot un(y)\cdot d_f}{\mu}\right) + 2.51\right)$$

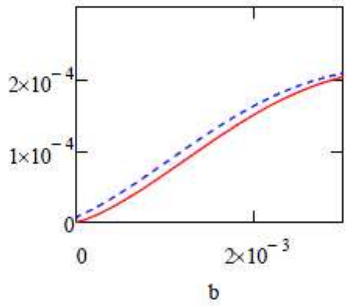
$$F_{sg} := \int_{l_1}^{h_f} \frac{1}{2}\cdot Cd(y)\cdot\rho\cdot un(y)^2\cdot d_f\, dy = 0.31\cdot\mu N$$

$$M_{sg} := \int_{l_1}^{h_f} \frac{1}{2}\cdot Cd(y)\cdot\rho\cdot un(y)^2\cdot d_f\cdot y\, dy = 630.833\cdot\mu N\cdot\mu m$$

The force per unit length calculations using the fiber base as the reference hieght or using the floor as the reference height.

$$q(b) := \frac{1}{2}\cdot Cd(b)\cdot\rho\cdot un(b)^2\cdot\frac{d_f}{\sin(\alpha_f)\cdot\sin(\beta_f)}$$

$$qm(b) := \frac{1}{2}\cdot Cd(b+h_m)\cdot\rho\cdot un(b+h_m)^2\cdot\frac{d_f}{\sin(\alpha_f)\cdot\sin(\beta_f)}$$



Deflection prediction without presence of mound:

$$dr3(y2) := \int_{l_1}^{y2}\frac{q(b)}{E\cdot I_f\cdot(\sin(\alpha_f)\cdot\sin(\beta_f))}\,db \qquad C1 := -\int_{l_1}^{h_f}\frac{q(b)}{E\cdot I_f\cdot(\sin(\alpha_f)\cdot\sin(\beta_f))}\,db = -1.229\times10^4\,\frac{1}{m^2}$$

$$dr2(y2) := \int_{l_1}^{y2}\frac{dr3(b)+C1}{\sin(\alpha_f)\cdot\sin(\beta_f)}\,db \qquad C2 := -\int_{l_1}^{h_f}\frac{dr3(b)+C1}{\sin(\alpha_f)\cdot\sin(\beta_f)}\,db = 25.733\,\frac{1}{m}$$

$$dr1(y2) := \int_{l_1}^{y2}\frac{dr2(b)+C2}{\sin(\alpha_f)\cdot\sin(\beta_f)}\,db$$

$$dr(y2) := \int_{l_1}^{y2}\frac{dr1(b)}{(\sin(\alpha_f)\cdot\sin(\beta_f))}\,db$$

$$\Delta := dr(h_f) = 6.806\times10^{-5}\,m$$

$$\Delta V := \Delta\cdot\Delta vn = \begin{pmatrix}0.068\\ -7.826\times10^{-3}\\ 1.865\times10^{-3}\end{pmatrix}\cdot mm$$

$$\Delta xol := \frac{\Delta V_0}{l_f} = 0.02 \qquad \Delta zol := \frac{\Delta V_2}{l_f} = 6.019\times10^{-4} \qquad \Delta tol := \frac{\left[(\Delta V_0)^2+(\Delta V_2)^2\right]^{0.5}}{l_f}$$

Deflection prediction with presence of mound inclded and assuming flow not disturbed by mound:

$$drm3(y2) := \int_{l_1}^{y2} \frac{qm(b)}{E \cdot I_f \cdot (\sin(\alpha_f) \cdot \sin(\beta_f))}\, db \qquad Cm1 := -\int_{l_1}^{h_f} \frac{qm(b)}{E \cdot I_f \cdot (\sin(\alpha_f) \cdot \sin(\beta_f))}\, db = -1.369 \times 10^4 \frac{1}{m^2}$$

$$drm2(y2) := \int_{l_1}^{y2} \frac{drm3(b) + Cm1}{\sin(\alpha_f) \cdot \sin(\beta_f)}\, db \qquad Cm2 := -\int_{l_1}^{h_f} \frac{drm3(b) + Cm1}{\sin(\alpha_f) \cdot \sin(\beta_f)}\, db = 27.733 \frac{1}{m}$$

$$drm1(y2) := \int_{l_1}^{y2} \frac{drm2(b) + Cm2}{\sin(\alpha_f) \cdot \sin(\beta_f)}\, db$$

$$drm(y2) := \int_{l_1}^{y2} \frac{drm1(b)}{(\sin(\alpha_f) \cdot \sin(\beta_f))}\, db$$

$$\Delta 2 := drm(h_f) = 7.253 \times 10^{-5}\, m$$

$$\Delta 2V := \Delta 2 \cdot \Delta vn = \begin{pmatrix} 0.072 \\ -8.34 \times 10^{-3} \\ 1.988 \times 10^{-3} \end{pmatrix} \cdot mm$$

$$\Delta 2xol := \frac{\Delta 2V_0}{l_f} = 0.023 \quad \Delta 2zol := \frac{\Delta 2V_2}{l_f} = 6.414 \times 10^{-4} \quad \Delta 2tol := \frac{\left[(\Delta 2V_0)^2 + (\Delta 2V_2)^2\right]^{0.5}}{l_f}$$

$$Data := augment\left(\frac{x}{mm}, \frac{\beta}{2 \cdot deg}, \frac{h_f}{mm}, \frac{l_f}{mm}, \frac{\delta_{bl}}{mm}, \frac{\alpha_f}{deg}, \frac{\beta_f}{deg}, \frac{U_{ch}}{\frac{m}{s}}, \frac{\Delta}{mm}, \Delta xol, \Delta zol, \Delta tol, \frac{\Delta 2}{mm}, \Delta 2xol, \Delta 2zol, \Delta 2to\right.$$

FileName := "0deg_E1_nAOA_alt.txt"

PRNCOLWIDTH := 12 PRNPRECISION := 6

APPENDPRN(FileName) := Data

FileName = "NoE_E3_nAOA_alt_revised.txt"

DataZ :=

TCs ← time(π)

for k ∈ 1 .. 10

$U_{ch} \leftarrow k \cdot \frac{m}{s}$

$Cm \leftarrow \frac{U_{ch}}{\left(\frac{ch}{mm}\right)^{M}}$

$U_x \leftarrow Cm \cdot \left(\frac{x}{mm}\right)^{M}$

$y \leftarrow \eta \cdot \left(\frac{M+1}{2} \cdot \frac{U_x}{\nu \cdot x}\right)^{-\frac{1}{2}}$

$\delta_{bl} \leftarrow \eta_{\delta} \cdot \left(\frac{M+1}{2} \cdot \frac{U_x}{\nu \cdot x}\right)^{-\frac{1}{2}}$

$u \leftarrow U_x \cdot uoU$

$v \leftarrow -\left(\frac{2}{M+1} \cdot \frac{\nu \cdot U_x}{x}\right)^{0.5} \cdot \left[intu + \frac{(M-1)}{(M+1)} \cdot \overrightarrow{(\eta \cdot uoU)}\right]$

ui(y2) ← interp(cspline(y, u), y, u, y2)

vi(y2) ← interp(cspline(y, v), y, v, y2)

$uv(y) \leftarrow \begin{pmatrix} ui(y) \\ vi(y) \\ 0 \cdot \frac{m}{s} \end{pmatrix}$

$un(y) \leftarrow \left[uv(y) \cdot uv(y) - (P \cdot uv(y))^2\right]^{0.5}$

$Uv \leftarrow \begin{pmatrix} u_{800} \\ v_{800} \\ 0 \end{pmatrix}$

$\Delta v \leftarrow Uv - (Uv \cdot P) \cdot P$

$\Delta vn \leftarrow \frac{\Delta v}{\sqrt{\Delta v \cdot \Delta v}}$

$Cd(y) \leftarrow \exp\left(-0.67 \cdot \ln\left(\frac{\rho \cdot un(y) \cdot d_f}{\mu}\right) + 2.51\right)$

$q(b) \leftarrow \frac{1}{2} \cdot Cd\left(b + h_m\right) \cdot \rho \cdot un\left(b + h_m\right)^2 \cdot \frac{d_f}{\sin(\alpha_f) \cdot \sin(\beta_f)}$

$$dr3(y2) \leftarrow \int_{l_1}^{y2} \frac{q(b)}{E \cdot I_f \cdot \left(\sin(\alpha_f) \cdot \sin(\beta_f)\right)}\, db$$

$$C1 \leftarrow -\int_{l_1}^{h_f} \frac{q(b)}{E \cdot I_f \cdot \left(\sin(\alpha_f) \cdot \sin(\beta_f)\right)}\, db$$

$$dr2(y2) \leftarrow \int_{l_1}^{y2} \frac{dr3(b) + C1}{\sin(\alpha_f) \cdot \sin(\beta_f)}\, db$$

$$C2 \leftarrow -\int_{l_1}^{h_f} \frac{dr3(b) + C1}{\sin(\alpha_f) \cdot \sin(\beta_f)}\, db$$

$$dr1(y2) \leftarrow \int_{l_1}^{y2} \frac{dr2(b) + C2}{\sin(\alpha_f) \cdot \sin(\beta_f)}\, db$$

$$dr(y2) \leftarrow \int_{l_1}^{y2} \frac{dr1(b)}{\left(\sin(\alpha_f) \cdot \sin(\beta_f)\right)}\, db$$

$$\Delta \leftarrow dr(h_f)$$

$$\Delta V \leftarrow \Delta \cdot \Delta vn$$

$$\Delta xol \leftarrow \frac{\Delta V_0}{l_f}$$

$$\Delta zol \leftarrow \frac{\Delta V_2}{l_f}$$

$$\Delta tol \leftarrow \frac{\left[\left(\Delta V_0\right)^2 + \left(\Delta V_2\right)^2\right]^{0.5}}{l_f}$$

$$\Delta t \leftarrow time(\pi) - TCs$$

$$Out \leftarrow augment\left(\frac{x}{mm}, \frac{\beta}{2 \cdot deg}, \frac{h_f}{mm}, \frac{l_f}{mm}, \frac{\delta_{bl}}{mm}, \frac{\alpha_f}{deg}, \frac{\beta_f}{deg}, \frac{U_{ch}}{\frac{m}{s}}, \frac{\Delta}{mm}, \Delta xol, \Delta zol, \Delta tol, \Delta t\right)$$

APPENDPRN(FileName, Out)

Out

BIBLIOGRAPHY

[1] A F Huber, Air University (U.S.). Center for Strategy, and Technology. *Death by a Thousand Cuts: Micro-air Vehicles in the Service of Air Force Missions*. Occasional paper. Center for Strategy and Technology, Air War College, 2002.

[2] Peter Van Blyenburgh. UAVs - Where Do We Stand? *Military Technology*, pages 29–30, March 1999.

[3] W. R. Jr. Davis, B. B. Kosicki, D. M. Boroson, and D. F. Kostishack. Micro Air Vehicles for Optical Surveillance. *THE LINCOLN LABORATORY JOURNAL*, 9:197–214, 1996.

[4] J.M. McMichael and M.S. Francis. Micro Air Vehicles - Toward a New Dimension in Flight, 1997.

[5] Wei Shyy, Mats Berg, and Daniel Ljungqvist. Flapping and flexible wings for biological and micro air vehicles. *Progress in Aerospace Sciences*, 35(5):455–505, 1999. DOI: 10.1016/S0376-0421(98)00016-5

[6] GV Crowley and LS Hall. Histological Observations on the Wing of the Grey-Headed Flying-Fox (Pteropus-Poliocephalus) (Chiroptera, Pteropodidae). *Australian Journal of Zoology*, 42(2):215, 1994. DOI: 10.1071/ZO9940215

[7] Ch. Brücker, J. Spatz, and W. Schröder. Feasability study of wall shear stress imaging using microstructured surfaces with flexible micropillars. *Experiments in Fluids*, 39(2):464–474, June 2005. DOI: 10.1007/s00348-005-1003-7

[8] Ch. Brücker. Time-resolved wall-shear stress imaging on surfaces coated with arrays of flexible micro-pillars. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Orlando, FL, 2010.

[9] J.M. Zook. The neuroethology of touch in bats: cutaneous receptors of the bat wing, 2005.

[10] Susanne Sterbing-D'Angelo, Mohit Chadha, Chen Chiu, Ben Falk, Wei Xian, Janna Barcelo, John M Zook, and Cynthia F Moss. Bat wing sensors support flight control. *Proceedings of the National Academy of Sciences of the United States of America*, 108(27):11291–6, July 2011. DOI: 10.1073/pnas.1018740108

[11] M Chadha, C F Moss, and S J Sterbing-D'Angelo. Organization of the primary somatosensory cortex and wing representation in the Big Brown Bat, Eptesicus fuscus. *Journal of comparative physiology. A, Neuroethology, sensory, neural, and behavioral physiology*, 197(1):89–96, January 2011. DOI: 10.1007/s00359-010-0590-9

[12] John Montgomery, Sheryl Coombs, and Matthew Halstead. Biology of the mechanosensory lateral line in fishes. *Reviews in Fish Biology and Fisheries*, 5(4):399–416, December 1995. DOI: 10.1007/BF01103813

[13] Sheryl Coombs and Paul Patton. Lateral line stimulation patterns and prey orienting behavior in the Lake Michigan mottled sculpin (Cottus bairdi). *Journal of comparative physiology. A, Neuroethology, sensory, neural, and behavioral physiology*, 195(3):279–97, March 2009. DOI: 10.1007/s00359-008-0405-4

[14] S Coombs. Smart skins: Information processing by lateral line flow sensors. *Autonomous Robots*, 11(3):255–261, 2001.

[15] Jeffrey M. Camhi, Winston Tom, and Susan Volman. The escape behavior of the cockroach Periplaneta americana. *Journal of Comparative Physiology ? A*, 128(3):203–212, 1978. DOI: 10.1007/BF00656853

[16] O Dangles, D Pierre, C Magal, F Vannier, and J Casas. Ontogeny of air-motion sensing in cricket. *The Journal of experimental biology*, 209(Pt 21):4363–70, November 2006. DOI: 10.1242/jeb.02485

[17] M.a b Shimozawa T.a b Kanou. Varieties of filiform hairs: range fractionation by sensory afferents and cercal interneurons of a cricket. *Journal of Comparative Physiology A*, 155(4):485–493, 1984. DOI: 10.1007/BF00611913

[18] T. Kumagai, T. Shimozawa, and Y. Baba. The shape of wind-receptor hairs of cricket and cockroach. *Journal of Comparative Physiology A: Sensory, Neural, and Behavioral Physiology*, 183(2):187–192, July 1998. DOI: 10.1007/s003590050246

[19] T. Shimozawa, T. Kumagai, and Y. Baba. Structural scaling and functional design of the cercal wind-receptor hairs of cricket. *Journal of Comparative Physiology A: Sensory, Neural, and Behavioral Physiology*, 183(2):171–186, July 1998. DOI: 10.1007/s003590050245

[20] B T Dickinson. Hair receptor sensitivity to changes in laminar boundary layer shape. *Bioinspiration biomimetics*, 5(1):16002, 2010. DOI: 10.1088/1748-3182/5/1/016002

[21] T Steinmann, J Casas, G Krijnen, and O Dangles. Air-flow sensitive hairs: boundary layers in oscillatory flows around arthropod appendages. *The Journal of experimental biology*, 209(Pt 21):4398–408, November 2006. DOI: 10.1242/jeb.02506

[22] J. A. C. Humphrey, R. Devarakonda, I. Iglesias, and F. G. Barth. Dynamics of Arthropod Filiform Hairs. I. Mathematical Modelling of the Hair and Air Motions. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 340(1294):423–444, June 1993. DOI: 10.1098/rstb.1993.0083

[23] F. G. Barth, U. Wastl, J. A. C. Humphrey, and R. Devarakonda. Dynamics of Arthropod Filiform Hairs. II. Mechanical Properties of Spider Trichobothria (Cupiennius salei Keys.). *Philosophical Transactions of the Royal Society B: Biological Sciences*, 340(1294):445–461, June 1993. DOI: 10.1098/rstb.1993.0084

[24] F. G. Barth, J. A. C. Humphrey, U. Wastl, J. Halbritter, and W. Brittinger. Dynamics of Arthropod Filiform Hairs. III. Flow Patterns Related to Air Movement Detection in a Spider (Cupiennius salei KEYS.). *Philosophical Transactions of the Royal Society B: Biological Sciences*, 347(1322):397–412, March 1995. DOI: 10.1098/rstb.1995.0032

[25] R. Devarakonda, F. G. Barth, and J. A. C. Humphrey. Dynamics of Arthropod Filiform Hairs. IV. Hair Motion in Air and Water. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 351(1342):933–946, July 1996. DOI: 10.1098/rstb.1996.0086

[26] Höller A Barth F.G. Dynamics of arthropod filiform hairs. V. The response of spider trichobothria to natural stimuli. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 354(1380):183–192, 1999.

[27] F G Barth. How to catch the wind: spider hairs specialized for sensing the movement of air. *Die Naturwissenschaften*, 87(2):51–8, February 2000.

[28] R Kant and J a C Humphrey. Response of cricket and spider motion-sensing hairs to airflow pulsations. *Journal of the Royal Society, Interface / the Royal Society*, 6(40):1047–64, November 2009. DOI: 10.1098/rsif.2008.0523

[29] N. H. Fletcher. Acoustical response of hair receptors in insects. *Journal of Comparative Physiology*, 127(2):185–189, June 1978. DOI: 10.1007/BF01352303

[30] G.K.a Taylor and H.G.b Krapp. Sensory Systems and Flight Stability: What do Insects Measure and Why? *Advances in Insect Physiology*, 34:231–316, 2007. DOI: 10.1016/S0065-2806(07)34005-8

[31] G J Schmitz, Ch Brücker, and P Jacobs. Manufacture of high-aspect-ratio micro-hair sensor arrays. *Journal of Micromechanics and Microengineering*, 15(10):1904–1910, October 2005. DOI: 10.1088/0960-1317/15/10/016

[32] Spatz J Schröder W Brucker C. Wall shear stress imaging using micro-structured surfaces with flexible micro-pillars. *12th International Symposium*, 7(5), 2004.

[33] M.a Schaefer, P.a Jacobs, D.b Bauer, D.a Moll, and A.a Gillner. Investigation and development of a molding process for the production of micro-hairs. *International Journal of Advanced Manufacturing Technology*, 51(9-12):935–944, 2010. DOI: 10.1007/s00170-010-2670-y

[34] M Dijkstra, J J van Baar, R J Wiegerink, T S J Lammerink, J H de Boer, and G J M Krijnen. Artificial sensory hairs based on the flow sensitive receptor hairs of crickets. *Journal of Micromechanics and Microengineering*, 15(7):S132–S138, July 2005. DOI: 10.1088/0960-1317/15/7/019

[35] Bruinink C M Sanders R G P Krijnen G J M Droogendijk H. Non-degenerate parametric amplification and filtering in biomimetic hair flow sensors. In *2011 16th International Solid-State Sensors, Actuators and Microsystems Conference, TRANSDUCERS'11*, pages 2038–2041, Beijing, 2011. DOI: 10.1109/TRANSDUCERS.2011.5969213

[36] Bruinink C M Sanders R G P Siebelder O G Krijnen G J M Droogendijk H. Lowering the sensory threshold and enhancing the responsivity of biomimetic hair flow sensors by electrostatic spring softening. In *Proceedings of IEEE Sensors*, pages 829–832, Limerick, 2011. DOI: 10.1109/ICSENS.2011.6127093

[37] Bruinink C M Sanders R G P Krijnen G J M Droogendijk H. Non-resonant parametric amplification in biomimetic hair flow sensors: Selective gain and tunable filtering. *Applied Physics Letters*, 99(21), 2011. DOI: 10.1063/1.3663865

[38] Bruinink C M Sanders R G P Krijnen G J M Droogendijk H. Application of electro mechanical stiffness modulation in biomimetic hair flow sensors. In *Proceedings of the IEEE International Conference on Micro Electro Mechanical Systems (MEMS)*, pages 531–534, Paris, 2012. DOI: 10.1109/MEMSYS.2012.6170232

[39] J M Engel, J Chen, C Liu, and D Bullen. Polyurethane Rubber All-Polymer Artificial Hair Cell Sensor. *Journal Of Microelectromechanical Systems*, 15(4):729–736, 2006. DOI: 10.1109/JMEMS.2006.879373

[40] Chang Liu. Micromachined biomimetic artificial haircell sensors. *Bioinspiration & biomimetics*, 2(4):S162–9, December 2007. DOI: 10.1088/1748-3182/2/4/S05

[41] L. Prandtl. Über Flüssigkeitsbewegung bei sehr kleiner Reibung. In *Verhandlungen des dritten internationalen Mathematischen Kongresses*, pages 484–491, Heidelberg, 1904.

[42] Hermann Schlichting. *Boundary-layer theory*. McGraw-Hill, New York, 7th edition, 1979.

[43] B T Dickinson. *Detecting Fluid Flows with Bioinspired Hair Sensors*. PhD thesis, Oregon State University, 2009.

[44] A J Aranyosi and Dennis M Freeman. Sound-induced motions of individual cochlear hair bundles. *Biophysical journal*, 87(5):3536–46, November 2004. DOI: 10.1529/biophysj.104.044404

[45] J-H Nam, J R Cotton, and J W Grant. Effect of fluid forcing on vestibular hair bundles. *Journal of vestibular research : equilibrium & orientation*, 15(5-6):263–78, January 2005.

[46] F M White. *Viscous fluid flow*. McGraw-Hill series in mechanical engineering. McGraw-Hill Higher Education, 2006.

[47] M. J. Sytsma and L. Ukeiley. Wind Tunnel Generated Turbulence. In *49th AIAA Aerospace Sciences Meeting*, Orlando, FL, 2011.

[48] Gilberto Narvaez and Stephen T Mcclain. Flow over a Distribution of Obliquely Aligned Elements : Part I Experimental Investigation. In *5th Flow Control Conference*, number July, pages 1–17, Chicago, Illinois, 2010. AIAA.