EXPERIMENTS AND DATA PROCESSING FOR TURBULENT FLOWS USING EULERIAN AND LAGRANGIAN TECHNIQUES

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Stephanie Jean Neuscamman

May 2009

© 2009 Stephanie Jean Neuscamman

ABSTRACT

In the study of turbulent flows, two reference frames exist in which fluid properties can be measured: a frame fixed in space, the Eulerian viewpoint, or a frame moving with the particle trajectory, the Lagrangian viewpoint. Turbulence research has been advanced primarily on experiments conducted using Eulerian techniques, but the developing Lagrangian methods are needed in order to determine the full acceleration, its temporal and spatial variation, of fluid particles. This research looks at two different problems involving turbulence: a turbulent boundary layer evolving beneath a turbulent free stream and the Lagrangian tracking of particles in turbulent flows.

The results of Eulerian measurements of a turbulent boundary layer evolving beneath free-stream turbulence using hot-wire anemometry are reported. The flat-plate boundary layer was created on a glass plate in a low-speed wind tunnel and free-stream turbulence was generated by an active grid. Systematic variation of the free-stream conditions from very low turbulence (0.25% turbulence intensity) to high turbulence (10.5% intensity) showed effects well within the boundary layer. The free-stream Reynolds number based on the Taylor micro-scale varied between 20 and 550; the boundary-layer momentum-thickness Reynolds number varied from 550 to almost 3,000. At high turbulence intensities, the effects of the free-stream turbulence extend deep into the boundary layer: affecting especially the velocity variances and the energy spectra. The energy spectra display a double-peak, for both near-laminar and turbulent free-streams. At very-low free-stream turbulence intensities, the two peaks represent the inner and outer scales of the turbulent boundary layer. With higher intensity free-stream turbulence present, the energy associated with the free-stream peak dominates the outer peak of the boundary layer.

A detailed description of the Lagrangian particle tracking framework used in experiments at Cornell University is presented. The theory and detailed instructions on the implementation of Lagrangian particle tracking are included. Camera calibration, both from a mask of points and using found particle data output from the tracking code from actual experiments (dynamic calibration) is described. The code used to conduct the analysis of particle image data obtained from the cameras is presented in detail. The code performs three main steps: 1) particle center finding, 2) stereomatching (determining particle 3D coordinates, if more than one camera is used), and 3) tracking particles through the time-series of images to construct trajectories. Each of these steps is conducted by a function called by the controller program. The controller program takes information such as the camera image filenames, the number of movies to be processed, the location and name of the calibration parameters file, and image intensity threshold values and outputs the reconstructed particle trajectory information in a data file. In order to assist future users of the code, the details of the original code and all edits made by the author have been included.

BIOGRAPHICAL SKETCH

Stephanie Neuscamman received her Bachelor's of Science degree in Aerospace Engineering with summa cum laude honors from the University of California at Los Angeles in June 2006. She was a student employee of Lawrence Livermore National Laboratory (LLNL) at Livermore, California from June 2004 to September 2005 where she worked for the National Atmospheric Release Advisory Center (NARAC).

For my sisters, Christine and Jackie, may your ambitions always be your own.

ACKNOWLEDGMENTS

I acknowledge Nicole Sharp's significant contribution to the turbulent boundary layer experiment and results, including the creation of all Figures in Sections 2.2 and 2.3. I also wish to thank Nicole for her support and friendship throughout my time at Cornell. I acknowledge Sergiy Gerashchenko for the design and construction of the flat-plate apparatus used to create the turbulent boundary layer. I thank Doug Kutz for his help with the experiments.

Haitao Xu developed the camera calibration code based on Tsai's radial alignment constraint and Greg Voth developed the dynamic calibration described in Section 3.2. The Lagrangian tracking code was developed by Nicholas Ouellette, Haitao Xu, and Eberhard Bodenschatz. All codes described in this document are used with permission of the original authors. I thank Haitao Xu and Greg Voth for their help in understanding the Lagrangain tracking code.

I thank my committee members Zellman Warhaft and Lance Collins for their guidance during this research.

This work was supported by the National Science Foundation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The goal of this work is to study several aspects of turbulence, one of the quintessential problems in fluid mechanics. It is a phenomenon that has remained unsolved for more than 100 years, yet nearly all flows engineers encounter are turbulent. Research that provides insight into this phenomenon is important for a wide range of disciplines and applications. For example, nearly all mixing processes, whether it be chemicals in a laboratory or pollutants in the atmosphere, are turbulent processes. Many flows in aeronautical applications involve turbulent boundary layers and it is often the case that turbulence is present above industrial or naturally occurring boundary layers. The introduction of particles other than the host fluid in turbulent flows brings a new dimension of complexity. If the particle density is different than the surrounding fluid, the motion of the particles (called inertial particles) will not follow this fluid exactly and settling or clustering effects can be observed. If the particle is larger than the smallest scale of the flow, the motion may also be different than a fluid particle. This research looks at two different problems involving turbulence: a turbulent boundary layer evolving beneath a turbulent free stream and the Lagrangian tracking of particles in turbulent flows. Though this thesis does not directly connect the two projects (i.e. Lagrangian tracking of particles in a turbulent boundary layer in the presence of free-stream turbulence) both components were directly utilized in a study that looked at the acceleration of inertial particles in such a boundary layer (Gerashchenko et al 2008).

There are two ways to describe flow characteristics in a moving fluid: variables defined at points fixed in space (the Eulerian reference frame) or in terms of the

trajectories of fluid particles (the Lagrangian reference frame). These designations apply to theoretical descriptions as well as the experimental methods used in turbulence research. A stationary probe can be used to make Eulerian measurements as the flow passes a fixed point. Alternatively, the Lagrangian fluid properties can be determined by following a particle as it moves in the flow. Eulerian measurements are much more common in the laboratory and have advanced the study of fluid mechanics and turbulence a great deal. However, to fully understand what is happening to the fluid particles in a highly turbulent flow, the ability to follow a fluid particle is required. Processes such as turbulent transport and mixing are best described in a Lagrangian reference frame. The research discussed here involves experiments of both kinds: hot wire anemometry to make Eulerian measurements and high speed cameras to make accurate Lagrangian measurements of illuminated particles.

Turbulent flow is characterized by significant and irregular velocity variation in both position and time. In order to make meaningful observations of turbulence despite its random nature, statistical quantities are used. Of particular use are the moments of the velocity variable U. In turbulence, the velocity can be defined as a mean motion plus fluctuations, $U = \langle U \rangle + u$. The first moment is the mean or expectation, $\langle U \rangle$, which is the probability weighted average of the velocity. The second moment is the variance, $\langle u^2 \rangle$, which is a convenient measure of the width of the velocity probability density function (PDF). The square root of the variance is perhaps the more familiar statistic: the standard deviation or root-mean-square amplitude, σ. Higher order moments are also useful quantities to measure in a turbulent flow. The third order moment, $\langle u^3 \rangle$, when normalized by the r.m.s. ($\langle u^3 \rangle / \sigma^3$) is the skewness, a non-dimensional measure of the asymmetry of the PDF. Positive skewness indicates that large positive values of the velocity fluctuation are more frequent than large negative

values. The fourth moment, $<u^4>/\sigma^4$, is the kurtosis or flatness factor. The moments of velocity fluctuations are useful statistics; profiles of the $1^{st} - 4^{th}$ order moments are presented in Chapter 2 for a turbulent boundary layer.

In addition to velocity statistics, turbulence can be characterized by various length scales. Turbulent fluid flow can be considered to be composed of eddies of different sizes. Richardson's energy cascade (introduced in 1922) states that for isotropic turbulence, the kinetic energy enters the flow at the largest scales of motion through the production mechanism. This energy is transferred to smaller and smaller eddies (scales) until it is finally dissipated by viscosity at the smallest scales present in the flow. Therefore, in addition to velocity statistics, information on the size of these scales of motion and the dissipation rate of turbulent kinetic energy are important properties to determine in turbulent research. The size of the smallest eddies (the eddies responsible for dissipating the turbulent energy) can be determined using Kolmogorov's first similarity hypothesis, which states that in every turbulent flow of sufficiently high Reynolds number, the statistics of the small-scale motions have a universal form with can be uniquely determined by the viscosity of the fluid, ν, and the rate of kinetic energy dissipation, ε. The length scale of the smallest eddies in a turbulent flow is then defined as $\eta \equiv (\nu^3/\varepsilon)^{1/4}$. Similarly, a velocity and time scale associated with the smallest eddies can be defined using dimensional arguments: $u_\eta \equiv (\varepsilon\nu)^{1/4}$ and $\tau_\eta \equiv (\nu/\varepsilon)^{1/2}$. The various length scales present in turbulent flows are evident in energy density spectra, $E(\kappa)$, which is a function of the wavenumber ($\kappa = 2\pi/l$, where l is a length scale).

For isotropic turbulence of sufficient Reynolds number, the spectra have a characteristic shape, as shown in Figure 1.1, above, for nearly isotropic turbulence.

The energy containing range (highest energy density) occurs at low κ, or large l. The mid-κ range displays power-law behavior with p = - 5/3. Finally, at high wavenumber (small scales), the spectra decays more rapidly than a power law, this is the dissipation range. One-dimensional energy spectra are presented in Chapter 2 for a turbulent boundary layer.

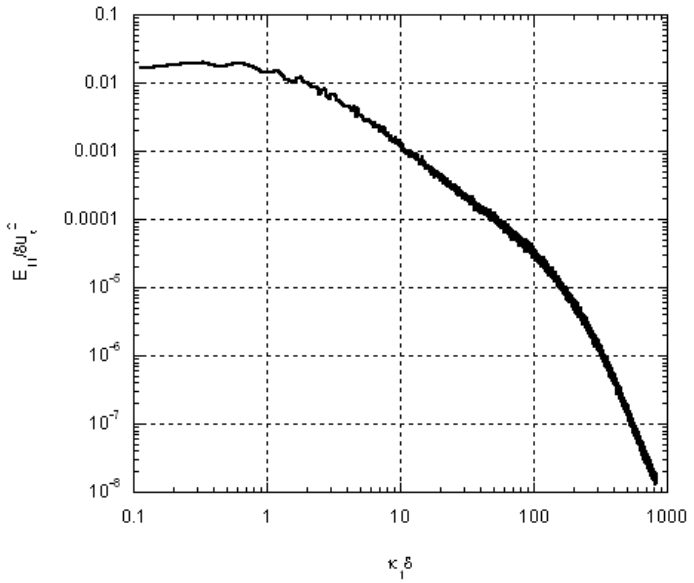


Figure 1.1: Energy density spectra for nearly isotropic turbulence. Data from Sharp et al (2009) Spectrum is from free-stream data for $Re_{\lambda 0}$ = 550.

Turbulence is a multi-scale, non-linear problem. In clouds, for example, which can be significantly turbulent, there are eddies on the order of the size of the cloud (1 km) down to millimeter-sized eddies and these scales are interacting. One flow property that indicates the effects of different lengthscales present in turbulent flows is the energy spectrum, $\kappa_1 E_{11}(\kappa_1)$, which shows how much energy is at a given wavenumber, κ_1, in the flow. The energy spectrum of a turbulent boundary layer is a subject of recent experimental interest. Hutchins and Marusic (2007) and Sharp et al (2009) present energy spectra results for Eulerian measurements of a turbulent boundary layer with and without free-stream turbulence present, respectively. The studies show that by identifying lengthscales associated with different locations in the flow, one can

compare the relative influence of each scale at different locations in the boundary layer and that the presence of energy at one scale can modulate the energy present at another scale in the turbulent flow. The energy spectra and the results of these studies are covered in detail in Chapter 2.

Hot wire anemometry is a useful experimental technique for studying the fluctuating velocity and other velocity-based statistics such as spectra of turbulent flows. Studying turbulence from a Lagrangian perspective can also give insight into velocity, but it is the ability to track particle motion through time that allows acceleration statistics to be determined (through differentiation). Although there have been indirect attempts to derive acceleration from Eulerian measurements (for example, Gylfason, Ayyalasomayajula &Warhaft (2004) have used fourth order structure functions to deduce the acceleration variance from hot wire anemometry data of isotropic turbulence), the full acceleration (its temporal and spatial variation) can only be determined in the frame of the fluid particle motion. The implementation of Lagrangian experiments has necessitated the development of new experimental methods. Different methods have been employed in order to record particle trajectories, such as acoustic techniques (Mordant et al 2001), silicon strip detectors (Voth et al 2002, Mordant et al 2004), and high-speed cameras to record particle motion via images of illuminated seeded particles in the turbulent flow (Ayyalasomayajula et al 2006, Ouellette et al 2006, Gerashchenko et al 2008). The framework described in this thesis for conducting Lagrangian particle tracking experiments uses one or multiple high-speed Phantom v7 cameras.

Recent experiments conducted at Cornell provide examples of experiments using one camera (Ayyalasomayajula et al 2006, Gerashchenko et al 2008) and multiple cameras

(Brown et al 2008). In the case of the Ayyalasomayajula and Gerashchenko experiments, the flow was turbulent wind tunnel flow, with water particles injected into the flow upstream of the measurement section. In order to perform Lagrangian measurements, the camera was accelerated to the mean speed of the tunnel flow, and the particle tracks were captured by the moving camera. In this way, the mean velocity was effectively subtracted from the flow. When the mean velocity is subtracted from the flow, the effect of the fluctuating component of velocity arising from acceleration of the fluid by turbulent eddies on the particle motion can be discerned. In the case of the Brown experiment, the flow in question was water in a tank with turbulence generated by counter-rotating disks. There is no mean flow in this case, and stationary cameras were used. Three cameras were used so that the three dimensional particle tracks could be determined using stereomatching techniques covered in Chapter 3. Future work at Cornell may include a moving track with multiple cameras in order to obtain three-dimensional tracks of particles in the wind-tunnel flow. Whether one or multiple cameras are recording data, a vital part of using camera images to record particle tracks is calibration. In order to determine particle locations in space from their image locations, a calibration must be conducted. This thesis focuses on the calibration technique needed to ensure accurate particle tracking and a discussion of the computer codes that assemble the three dimensional particle tracks from camera image data in Chapter 3.

The remainder of this thesis is divided, like turbulence research itself, into sections involving Eulerian and Lagrangian research. Chapter 2 of this thesis describes an experiment conducted at Cornell University by Nicole Sharp and the author that investigates a turbulent boundary layer evolving in the presence of free-stream turbulence. Section 2.1 details previous work in turbulent boundary layers, both with

and without free-stream turbulence present. When the free-stream has little to no turbulence, it is referred to as the canonical turbulent boundary layer. Section 2.2 describes the experimental set-up, including details on the wind-tunnel, the turbulence generation methods, and the equipment used to sample and record the flow properties. Section 2.3 summarizes the results of the study, specifically the velocity and turbulence intensity profiles, variance and higher order moments, and spectral results. Chapter 3 presents a detailed guide to the operation of a variety of computer codes that are used to conduct Lagrangian particle tracking experiments. Section 3.1 describes the main components of Lagrangian particle tracking and the method used to conduct each step in Zellman Warhaft's laboratory at Cornell University. Section 3.2 describes the camera calibration, a vital step in Lagrangian particle tracking, which is necessary to relate the physical coordinates of an object in three dimensions with the images captured by the cameras. Section 3.3 presents a detailed instructional walk-through to the use of the Lagrangian particle tracking code as developed by the Bodenschatz group at Cornell University and the Max Planck Institute in Göttingen, Germany. Section 3.4 describes the changes made to the code in the process of conducting a specific Lagrangian particle tracking experiment in 2008.

CHAPTER 2

MEASUREMENTS OF A TURBULENT BOUNDARY LAYER WITH INTENSE FREE-STREAM TURBULENCE

Section 2.1: Introduction

Hotwire anemometry (HWA) is a technique for measuring the velocity fluctuations in a turbulent gas flow (Bruun 1995). It is this method that has led to the majority of the accumulated knowledge of turbulence and its characteristics. Although some focus in experimental turbulence research has shifted to techniques such as Laser Doppler Velocimetry (LDV) and Particle Image Velocimetry (PIV) that also measure fluid velocity and related properties, HWA is still contributing to our understanding of a variety of turbulent flows because of its ability to resolve high frequency fluctuations and obtain well resolved energy spectra. In this chapter, I describe hotwire measurements of a turbulent boundary layer evolving in the presence of free-stream turbulence. The work described here was initially motivated by the flow described in Gerashchenko *et al* (2008), in which a turbulent boundary layer is used as a means of generating shear in a turbulent flow in order to study its effects on the acceleration statistics of inertial particles introduced into the flow. In order to augment the Lagrangian particle tracking (see Chapter 3) measurements of acceleration conducted in that experiment, we employed hotwire anemometry to determine the velocity field of the turbulent boundary layer with high free-stream turbulence. In the course of the initial measurements, we observed that the presence of the free-stream turbulence had a significant impact on the velocity statistics of the boundary layer compared to the canonical case of a turbulent boundary layer beneath a laminar free-stream flow.

Previous work on turbulent boundary layers can be divided into two categories: those with a laminar free-stream flow (canonical turbulent boundary layers) and those evolving in the presence of free-stream turbulence. The canonical turbulent boundary layer is a well-studied flow. Texts such as *Turbulent Flows* (Pope 2000) report in detail the structure of the flow and defined regions of the boundary layer. These regions are shown as a function of height above the boundary and the free-stream Reynolds number in Figure 2.1.

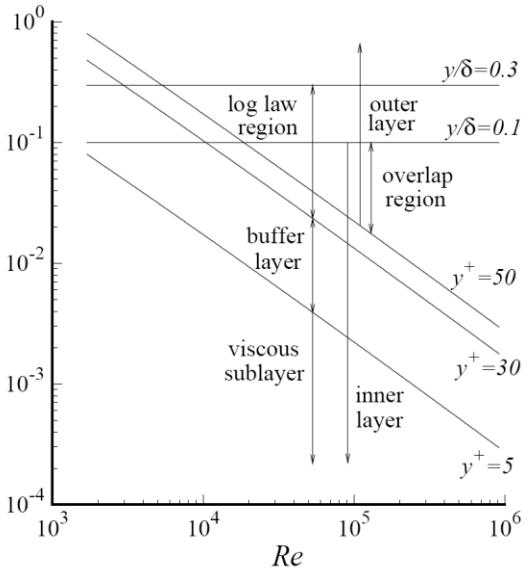


Figure 2.1: Regions and layers in Wall Bounded Flows (Pope, 2000)

The mean velocity profiles are described both in terms of the law of the wall and the velocity deficit law. The Reynolds stress profiles and other turbulence characteristics such as production and dissipation are shown in Figures 2.2 and 2.3.

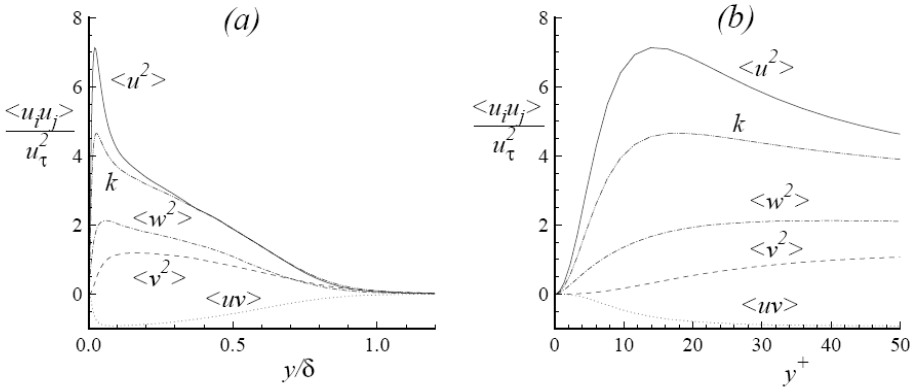


Figure 2.2: Profiles of Reynolds stresses normalized by the friction velocity from the DNS data of Spalart (1988). a) across the boundary layer and b) in the near-wall region (from Pope 2000)

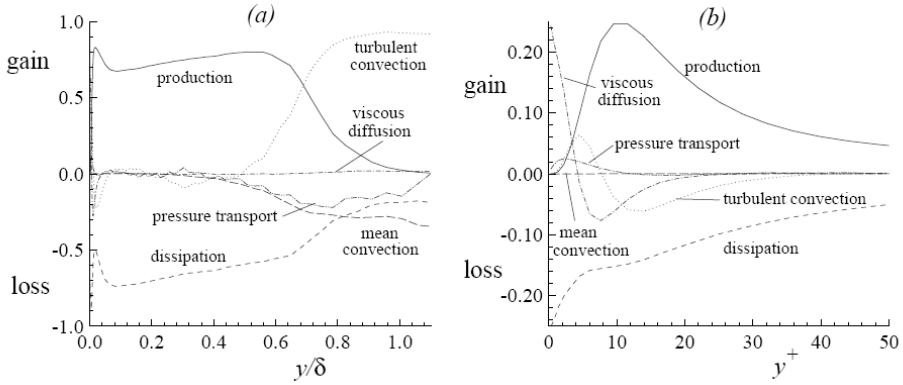


Figure 2.3: Turbulent kinetic energy budget for a turbulent boundary layer from the DNS data of Spalart (1988) (source: Pope 2000)

The data in the graphs above were obtained using direct numerical simulation at a Reynolds number based on boundary layer thickness of 1410 (Spalart 1988). The profiles depend upon the Reynolds number, but there is not much data on this dependence. Kim, Kline and Reynolds (1971) used hydrogen bubble PIV and hotwire measurements together to determine the production, energy spectra, r.m.s. velocities, and autocorrelation in a turbulent boundary layer. They found that the energy production in a turbulent boundary layer occurs in intermittent bursting periods. Robinson's 1991 review of the turbulent boundary layer focuses on the structures (or

coherent motions) in the boundary layer, with compelling visual depictions of the nature of this flow. DeGraaff and Eaton (2000) used laser doppler anemometry to study the scaling of the Reynolds stresses. Saddoughi and Veeravalli (1994) documented spectra and look at the Kolmogorov (K-41) assumptions of local isotropy for 2nd order quantities using hot wire anemometry.

The introduction of turbulence to the free-stream adds complexity to the flow, and the structure of the boundary layer can be significantly altered by its presence. Several important studies have been made investigating this flow phenomenon. Hancock and Bradshaw (1989) present detailed measurements of the boundary layer evolving in the presence of free-stream turbulence with a range of external length scales. The authors construct a complete turbulent kinetic energy balance and find that the dissipation length scale is little affected by free-stream turbulence. However, this study was limited to low intensity free-stream turbulence (around 6%, where turbulence intensity is defined as the ratio of the r.m.s. velocity fluctuations to the mean velocity, ($<u^2>^{1/2}/U_0$)) and free-stream length scales that were on the order of the boundary layer thickness, δ. Bott and Bradshaw (1997) also consider a boundary layer with free-stream turbulence. The focus of that effort was on the determination of mean flow characteristics such as skin friction coefficient and heat transfer, properties which are of interest for applications involving turbine blades – since the impinging of the turbulent wake of one stage of blades to the next can be modeled as a boundary layer evolving under free-stream turbulence. Thole and Bogard (1997) provide even more insight into this flow, studying a boundary layer in the presence of much higher free-stream intensity values, up to 20%, and relatively large free-stream length scales. They show how the r.m.s. velocity profiles change with free-stream turbulence conditions and that the free-stream integral length scale penetrates well into the

boundary layer. The resulting energy spectra do not show strong variation with height compared to the canonical boundary layer. These results lead the authors to conclude that the free-stream turbulence has significant effects on turbulence properties even very close to the wall. The goal of our current work is to expand on the understanding of the physics of the flow in a turbulent boundary layer with high intensity free-stream turbulence by conducting detailed measurements of such a boundary layer with varying free-stream conditions using hotwire anemometry.

Many of the following results are normalized by the flow's friction velocity u_τ and the viscosity ν; these are called "inner variables." The friction velocity can be determined by use of the log-law of the wall (Pope 2000) for turbulent boundary layers with little to no free-stream turbulence. Thole and Bogard (1996) showed that the log-law of the wall is valid even for flows with free-stream turbulence intensity of up to 20%. The law states that the near-wall velocity is determined by

$$\frac{U}{u_\tau} = \frac{1}{\kappa}\ln\frac{yu_\tau}{\nu} + C \qquad 2.1$$

where κ = 0.41 is the von Karman constant, C is a constant with value of 5.0, and u_τ is the friction velocity. The friction velocity for each case in this experiment is determined using a fit to the log-law. Quantities normalized by inner variables are indicated with the superscript '+'.

Section 2.2: Experimental Apparatus

We conducted measurements in an open-circuit, low-speed wind tunnel with a square cross-section of dimensions 0.91 m^2 x 0.91 m^2 and length 9.41 m (Yoon and Warhaft 1990). The boundary layer formed on a flat, smooth glass plate, 3.3 m x 0.67 m by

0.012 m, which sat 0.35 m from the tunnel floor, see Figure 2.4. An active grid generated high intensity free-stream turbulence in the test section of the tunnel. The grid consisted of evenly spaced bars with attached triangular wings or flaps that rotate randomly (Mydlarski and Warhaft 1996). This configuration creates turbulence with much higher free-stream Reynolds numbers and larger length scales compared to a passive grid, which consists of biplanar bars with even spacing. The active grid had a mesh length (inter-bar spacing) of M=11.4 cm. In order to insure approximately isotropic turbulence from the position where the free-stream is incident on the plate, the leading edge of the plate was located no less than 30M (343 cm) from the active grid. In order to achieve a range of free-stream Reynolds numbers (i.e. a range of free-stream intensities), we varied both the tunnel speed and the configuration of the grid. For higher intensity turbulence, the grid operated with the flaps rotating randomly (referred to as the "active configuration"). For lower intensity turbulence, we aligned the flaps with the flow direction, such that the profile resembled that of a passive grid (referred to as the "passive configuration").

We use constant-temperature hot wire anemometry to measure the time-series of velocity fluctuations in this flow, from which we can determine relevant flow parameters such as mean and r.m.s. velocities and spectral data. We use a TSI 1243-T1.5 two-channel X-wire probe with tungsten wires 3.05 μm in diameter specifically designed to measure boundary layer flow. The length to diameter ratio was approximately 200. The x-wire allowed us to record velocity data for both the x- (free-stream mean flow) direction and the y- (wall-normal) directions. The hot wires were connected to Disa 55M01 constant-temperature bridges, and signals from these passed through high-pass (0.01 Hz) and low-pass (between 1,000 and 10,000 Hz)

filters to reduce large-scale disturbances and high-frequency noise before digitization (Sharp et al 2009).

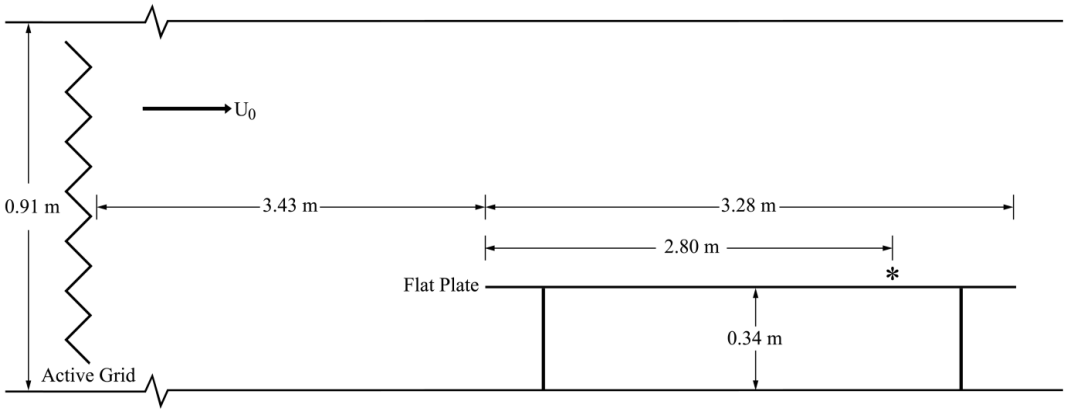


Figure 2.4: The experimental set up. The position of the measurement station is indicated by *

Section 2.3: Results and Discussion

This section presents a summary of the results of the study of the effects of free-stream turbulence on a turbulent boundary layer reported in Sharp, Neuscamman and Warhaft (2009). The free-stream conditions over the turbulent flat-plate boundary layer were varied; eight different free-stream conditions were investigated, as shown in Table 2.1. The data were all taken 2.8 m from the flat plate's leading edge. The table includes the following flow properties: the free-stream mean velocity, U_0 (m/s); the free-stream Reynolds number, $Re_{\lambda 0} \equiv < u^2 >^{1/2} \lambda/\nu$, (where $\lambda =< u^2 >^{1/2}/< (du/dx)^2 >^{1/2}$ is the Taylor micro-scale); the free-stream turbulence intensity, $(<u^2>^{1/2}/U)_0$; the boundary layer thickness based on 99.5% of the free-stream velocity, δ (cm); the ratio of the free-stream length scale, L, to δ; the boundary layer momentum thickness, $\theta \equiv \int_0^\infty (U/U_0)(1-(U/U_0))\, dy$ (cm); the friction velocity, u_τ (m/s); the Reynolds number based on momentum thickness, $Re_\theta \equiv U_0\theta/\nu$; the Reynolds number based on the wall stress, $Re_\tau \equiv u_\tau\delta/\nu$; and the ratio of the

longitudinal free-stream velocity fluctuation r.m.s. to the friction velocity, $< u^2 >_0^{1/2}/u_\tau$.

Table 2.1: The flow parameters for the eight experimental cases

$Re_{\lambda 0}$	U_0 (m s^{-1})	δ (cm)	θ (cm)	$(<u^2>^{1/2}/U)_0$	u_τ (m s^{-1})	Re_θ	Re_τ	L/δ	$<u^2>_0^{1/2}/u_\tau$
20	6.25	6.99	0.59	0.25%	0.2671	2460	1245	-	0.059
60	7.71	7.08	0.55	1.4%	0.3293	2840	1560	-	0.315
160	2.29	8.38	0.36	7.8%	0.1211	550	680	2.8	1.480
260	3.70	7.31	0.31	8%	0.1878	775	915	4.4	1.557
450	6.73	6.56	0.33	10.0%	0.3145	1465	1375	5.5	2.173
500	7.52	5.34	0.28	10.2%	0.3523	1400	1250	7.0	2.131
550	8.15	8.09	0.36	10.2%	0.3747	1980	2020	5.2	2.256
550	8.49	7.27	0.32	10.5%	0.3963	1810	1920	5.1	2.229

In two cases, the free-stream Reynolds number, $Re_{\lambda 0}$, is low (20 and 60) in order to compare the results to a turbulent boundary layer with no free-stream turbulence present. We refer to these cases as "near-canonical" flows. The free-stream turbulence is then increased (up to $Re_{\lambda 0}$ = 550) to study its increasing effects on the boundary layer. The intensity of the free-stream at the measurement station ranged from 0.25% to 10.5%.

The mean velocity profiles for each case are shown in Figure 2.5a, normalized by the friction velocity u_τ and the viscosity ν. All cases show a well-defined log region, with the near-canonical boundary layers showing the largest log-region. Inspection of Figure 2.5a shows that the eight profiles collapse into three distinct cases. These cases correspond to 1) a developed boundary layer with very low free-stream turbulence (circles and squares, $Re_{\lambda 0}$ = 20 and 60), 2) a developed boundary layer with high free-stream turbulence (inverted triangles, right triangles, and open squares, $Re_{\lambda 0}$ = 450, 500, 550), and 3) a developing (low Re_θ) boundary layer with significant free-stream turbulence (diamonds and triangles, $Re_{\lambda 0}$ = 160, 260). The mean velocity profiles for

an example of each case ($Re_{\lambda 0}$ = 20, 260, 550) are shown in Figure 2.5b for clarity. Also included in Figure 2.5a and b are data for a boundary layer without free-stream turbulence from DeGraaff and Eaton (2000) which show a good agreement with our near-canonical data. The turbulence intensity, ($<u^2>^{1/2}/U_0$), for all cases is shown in Figure 2.6. The three example cases are clear here as well.

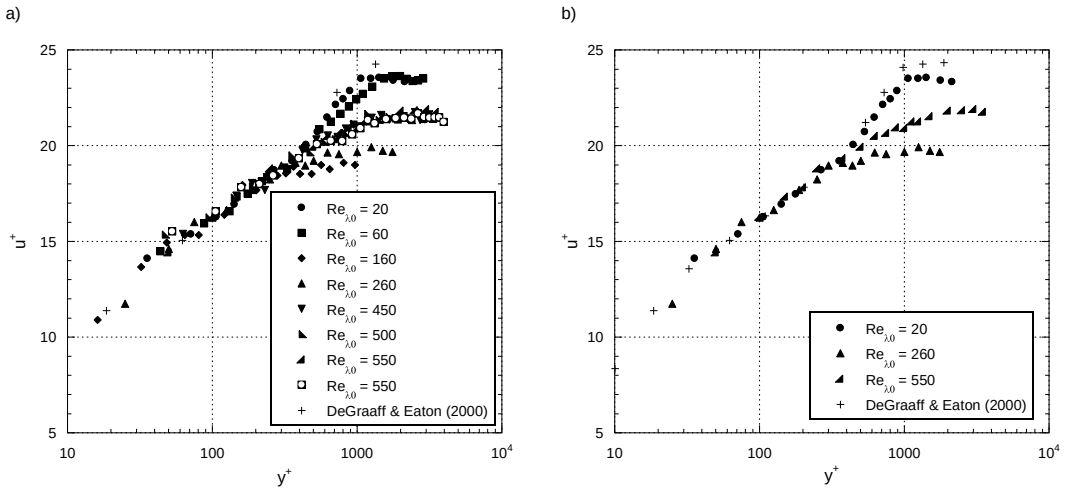


Figure 2.5: Mean velocity profiles normalized by inner variables for: a) all cases and b) example cases. Data for a boundary layer without free-stream turbulence from DeGraaff and Eaton (2000) are included for comparison

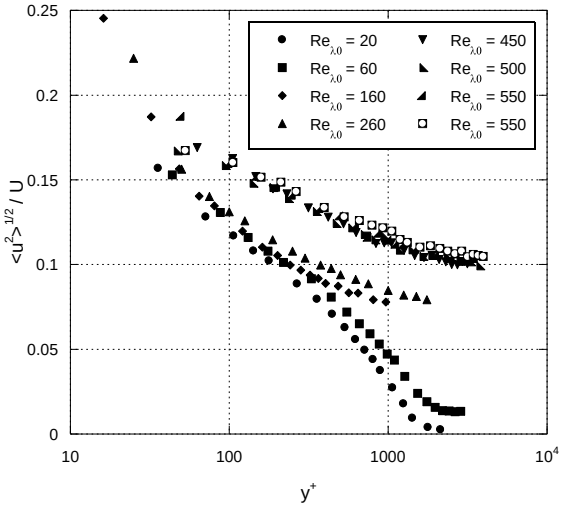


Figure 2.6: Turbulence intensity profiles.

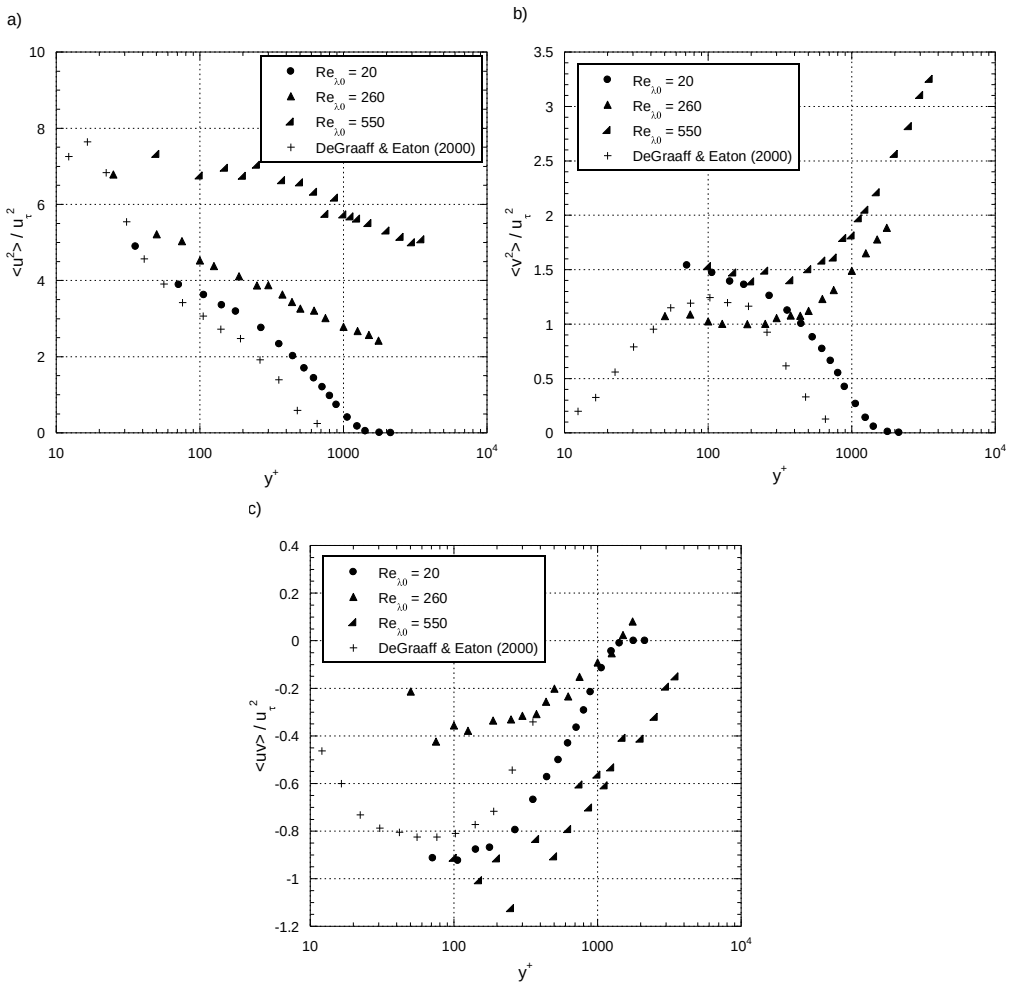


Figure 2.7: Normalized variances and covariances: a) $\langle u^2 \rangle$, b) $\langle v^2 \rangle$, and c) $\langle uv \rangle$

Figure 2.7 shows the normalized variances and covariances for the three example cases. Again, data from DeGraaff and Eaton (2000) for a boundary layer without free-stream turbulence are included for comparison. For the near-canonical case, the variances start at zero in the free-stream and increase into the boundary layer, showing good agreement with the turbulent boundary layer data with no free-stream turbulence. For the cases with free-stream turbulence, the stream-wise variance, $\langle u^2 \rangle$, starts at a non-zero value in the free-stream and increases into the boundary layer. With free-stream turbulence, the plate-normal variance, $\langle v^2 \rangle$, starts at a non-zero value in the free-stream and decreases as the plate is approached. This trend for boundary layers in

the presence of free-stream turbulence is also evident in the work of Thole and Bogard (1996) and Bott and Bradshaw (1997). This decrease in the plate-normal velocity variance can be attributed to the influence of the wall. In order to satisfy both the free-stream conditions and the no-slip condition, $\langle v^2 \rangle$ must decrease from its free-stream value inside the boundary layer (Hunt and Graham 1978). Figure 2.7c shows the covariances, $\langle uv \rangle$, for the three example cases. The near-canonical case again shows good agreement with the data of DeGraaff and Eaton (2000). The highly turbulent free-stream case has a stronger negative covariance than when very low free-stream turbulence is present. The intermediate case (diamonds, $Re_{\lambda 0} = 260$), has a lower magnitude covariance than the near-canonical case. This difference may indicate a dependence on the boundary layer thickness Reynolds number, Re_θ. While the free-stream turbulence in the $Re_{\lambda 0} = 260$ case is only slightly lower than the $Re_{\lambda 0} = 550$ case (turbulence intensity of 8% versus 10%), the boundary layer Reynolds number is smaller than both the $Re_{\lambda 0} = 550$ and $Re_{\lambda 0} = 20$ cases (775 versus 1980 and 2460, respectively).

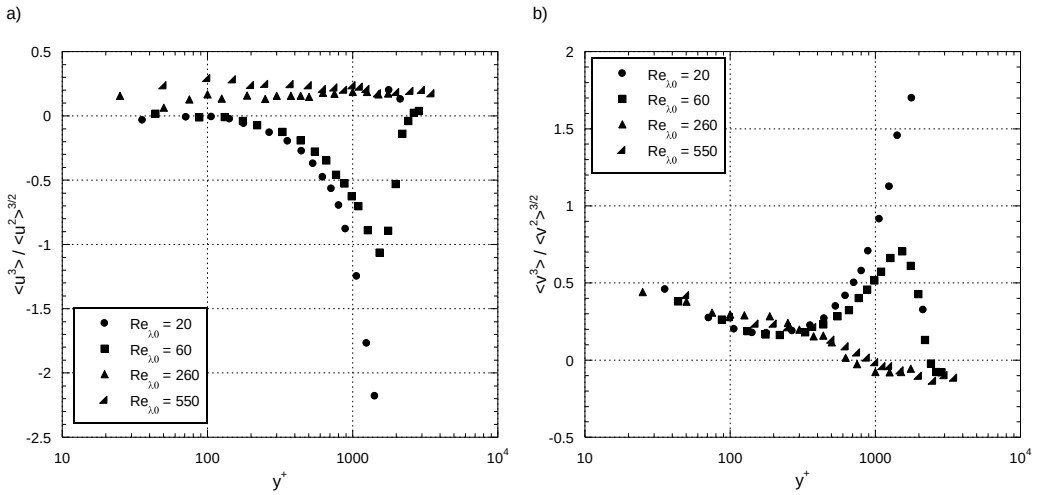


Figure 2.8: Large-scale skewness profiles in the: a) *u*-direction and b) *v*-direction

The normalized third velocity moments, skewness, are shown in Figure 2.8. The cases with significant free-stream turbulence show a departure from the near-canonical results for both stream-wise ($<u^3>$) and plate-normal ($<v^3>$) skewness profiles. The $Re_{\lambda 0}$ = 60 case has been included in Figures 2.8a and b in order to highlight the transition from the near-canonical cases to the turbulent free-stream cases. For the near-canonical cases, both directions show an increase in the magnitude of the skewness as the edge of the boundary layer is approached from the plate, and a decrease after. The cases with significant free-stream turbulence present, however, show little change in both the stream-wise direction and the plate-normal direction. The presence of free-stream turbulence above a turbulent boundary layer serves to smooth out the profiles, showing the effective mixing of the outer portion of the boundary layer with the free-stream turbulence.

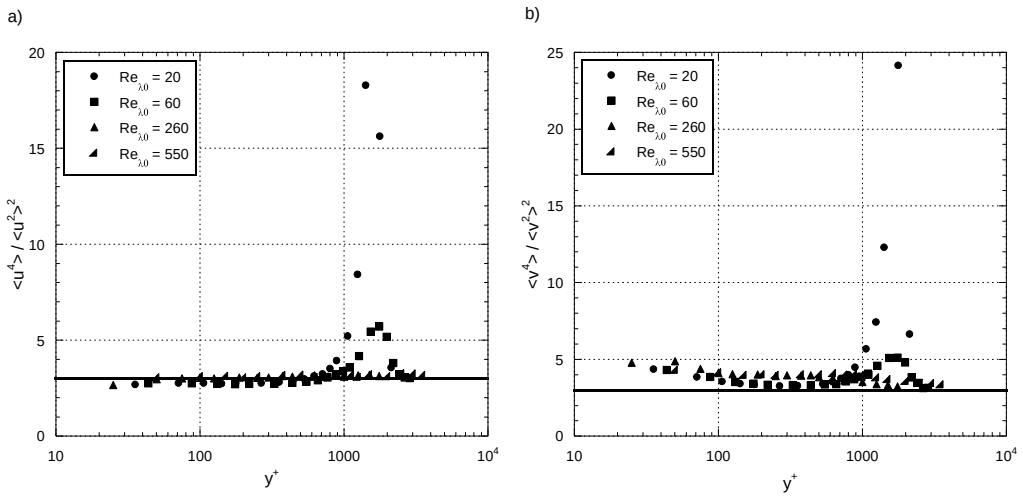


Figure 2.9: Large-scale kurtosis profiles in the: a) *u*-direction and b) *v*-direction

Similar trends are evident in the large-scale kurtosis profiles in Figure 2.9. When significant free-stream turbulence is present, the kurtosis in both *u*- and *v*- directions are nearly constant from deep in the boundary layer to the free-stream. The normalized kurtosis is nearly 3 in these cases, indicating Gaussian velocity PDFs. The

near-canonical cases also show a kurtosis value that is essentially Gaussian, with the exception of the sharp transition at the edge of the boundary layer. Both the skewness and kurtosis profiles indicate that the presence of free-stream turbulences negates the statistical effects of transition at the edge of the boundary layer.

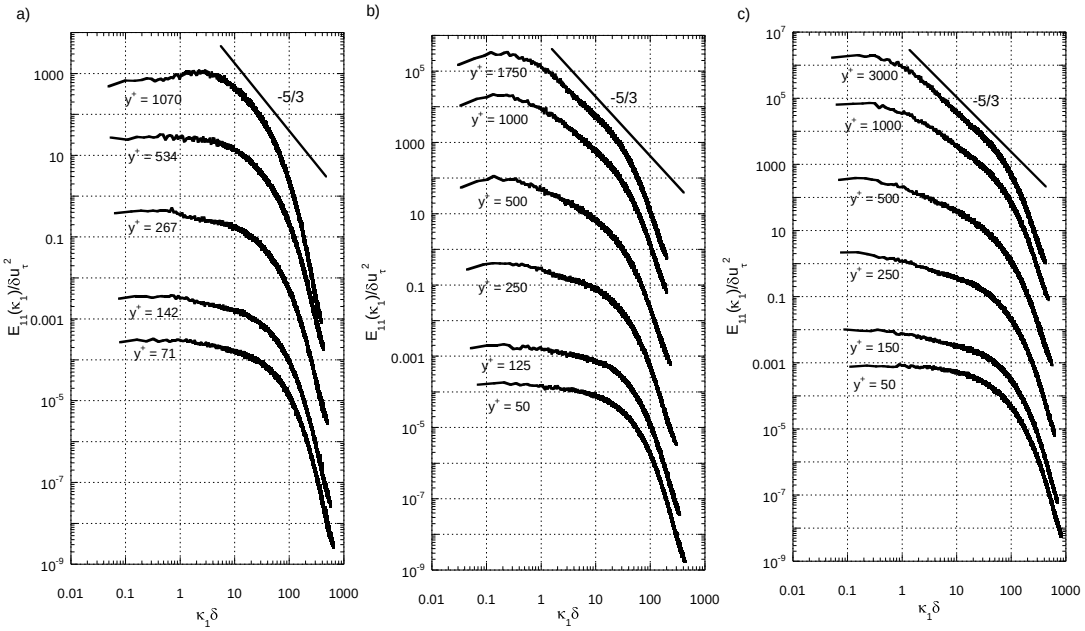


Figure 2.10: Evolution of boundary layer spectra for: a) $Re_{\lambda 0}$= 20, b) $Re_{\lambda 0}$= 260, and c) $Re_{\lambda 0}$= 550. Spectra are staggered relative to the lowest spectrum. From bottom to top, the offset of each spectrum relative to the lowest is: 1, 3, 5, 7 and 8 decades

The evolution of the boundary layer energy density spectra, $E_{11}(\kappa_1)$, where κ_1 is the wavenumber in the stream-wise direction, are shown in Figure 2.10 for each of the example cases. The spectra are staggered from the bottom (closest to the plate) to the top (free-stream) for clarity. In the near-canonical case, the free-stream spectrum is very narrow, indicating low turbulence. The top spectra in Figures 2.10b and c, however, show a flat inertial sub-range that increases with Reynolds number. The -5/3 slope, as predicted by Kolmogorov (1941), has been included to emphasize the inertial sub-range in the cases with significant free-stream turbulence (Pope 2000). As the

plate is approached, the inertial range becomes less pronounced and disappears as the local Reynolds number decreases and the flow becomes more anisotropic.

Finally, the behavior of the energy spectrum, $\kappa_1 E_{11}(\kappa_1)$, for the three example flows is investigated. The energy spectrum shows how much energy is at a given wave-number at a particular location in the flow. The energy spectra taken close to the wall show a double peak for all three example cases, indicating that the energy is primarily located at two lengthscales (see Figure 2.11). In the cases with free-stream turbulence, we will show that the energy at the larger lengthscale is due to the turbulent free-stream, and the smaller lengthscale corresponds to the turbulent energy generated by the boundary layer. With this observation, we can compare the relative influence of the free-stream and the boundary layer at different points in the boundary layer.

However, in the near-canonical case, the near-wall energy spectra also show a double-peak. Hutchins and Marusic (2007) studied the energy spectra of a turbulent boundary layer with no free-stream turbulence and identified two major lenghtscales that appeared under certain conditions in the near-wall energy spectra. In their experiments, a peak in the energy spectrum was found at $\lambda^+ \sim 1000$ (where $\lambda^+ \equiv 2\pi/\kappa$, note that all energy spectra here have been plotted against this variable for ease of comparison) close to the wall (most pronounced at $y^+ \sim 15$). A second peak was present at $\lambda \sim 6\delta$. Hutchins and Marusic (2007) suggest that the inner peak was associated with turbulent production and that the second peak represented superstructures in the boundary layer that modulated near-wall production. They found that the outer scale appeared at a height of $y/\delta \sim 0.06$. When the boundary layer Reynolds number was very high, only the outer scale peak was present at this height. For lower Reynolds number boundary layers (specifically $Re_\tau \equiv u_\tau \delta / \nu \sim 1{,}000$ in the

Hutchins and Marusic (2007) experiment), the separation of scales was reduced and the spectrum at $y/\delta \sim 0.06$ exhibited both peaks. Our near-canonical case shows good agreement with Hutchins and Marusic, as shown in Figure 2.11a: for $y/\delta \sim 0.06$, the inner peak appears at $\lambda^+ \sim 1200$ and the outer peak at $\lambda \sim 6\delta$.

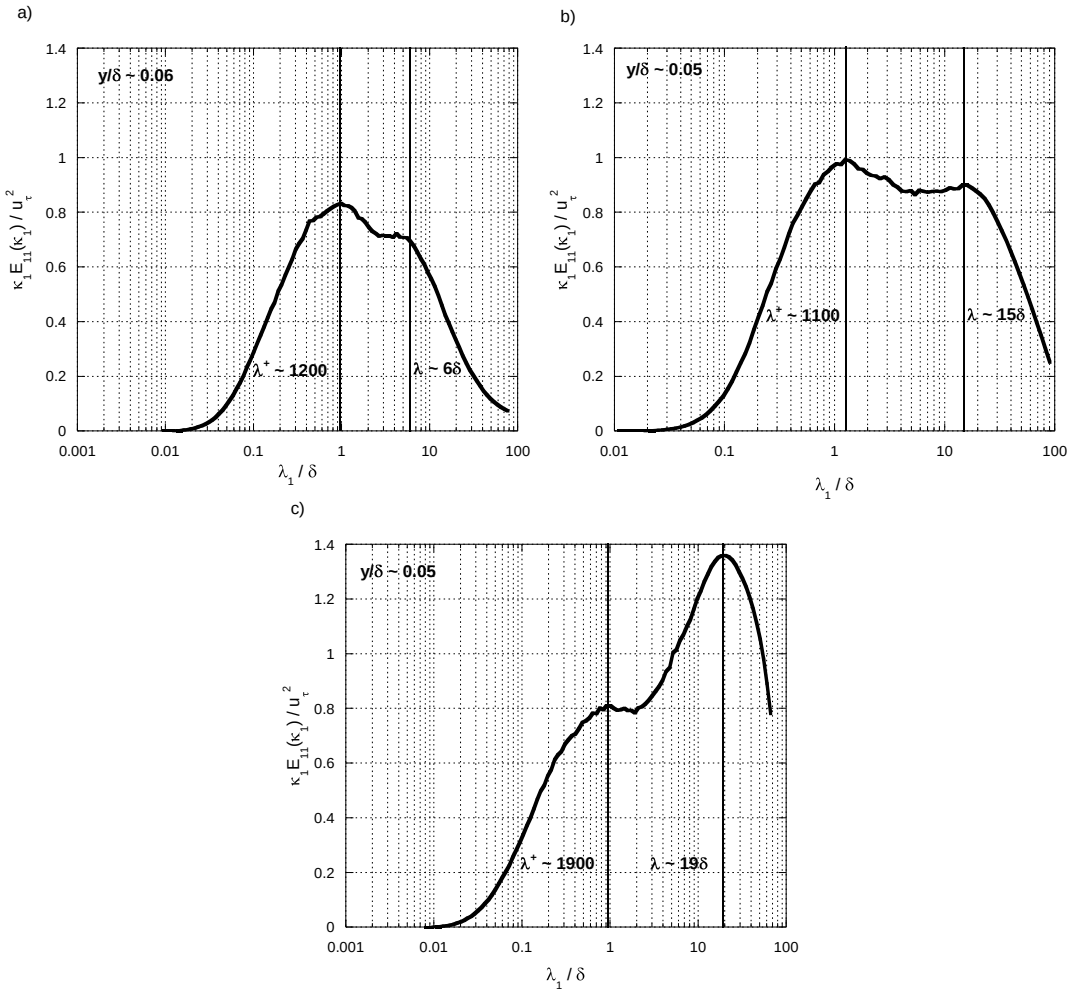


Figure 2.11: Normalized near-wall energy spectra showing double peaks at $y/\delta \sim$ 0.05-0.06 for a) $Re_{\lambda 0} = 20$, b) $Re_{\lambda 0} = 260$, and c) $Re_{\lambda 0} = 550$.

Figures 2.11b and c show the energy spectra at a height of $y/\delta \sim 0.05$ for the two cases of the boundary layer with free-stream turbulence. Again, two peaks are observed at this height. Since the addition of free-stream turbulence introduces a new length scale,

a three-peaked spectrum might be expected (two peaks associated with the boundary layer as in Hutchins and Marusic 2007, and one associated with the free-stream turbulence), but it is not observed. The peaks are shifted from the near-canonical case, occurring at $\lambda^+ \sim 1100$ and $\lambda^+ \sim 15\delta$ for $Re_{\lambda 0} = 260$ (Figure 2.11b)and at $\lambda^+ \sim 1900$ and $\lambda^+ \sim 19\delta$ for $Re_{\lambda 0} = 550$ (Figure 2.11c). Examining the $Re_{\lambda 0} = 260$ case, it is clear that while the lower peak has not changed significantly from the near-canonical case, the shift in the outer peak from $\lambda \sim 6\delta$ to $\lambda \sim 15\delta$ is substantial. Figure 2.12 shows the evolution of the energy spectra with height above the plate for $Re_{\lambda 0} = 260$.

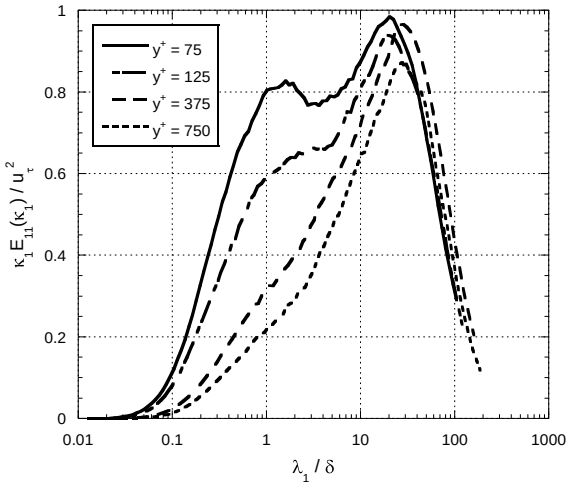


Figure 2.12: Evolution of normalized energy spectra from the wall to free-stream for $Re_{\lambda 0} = 260$.

Near the wall ($y^+ = 75$), the energy spectra has two distinct peaks. Far from the boundary ($y^+ = 750$), the free-stream energy spectrum contains only one peak. The curves for $y^+ = 125$ and $y^+ = 325$ show that as the free-stream is approached, the inner peak disappears, indicating that the inner peak is associated with the boundary layer and the outer peak, which does not change significantly, is associated with the free-stream turbulence. The shift of the bimodal peaks from the near-canonical case is

also clear in the $Re_{\lambda 0}$ = 550, where the peak associated with the free-stream turbulent energy contains high energy compared to the inner length scale.

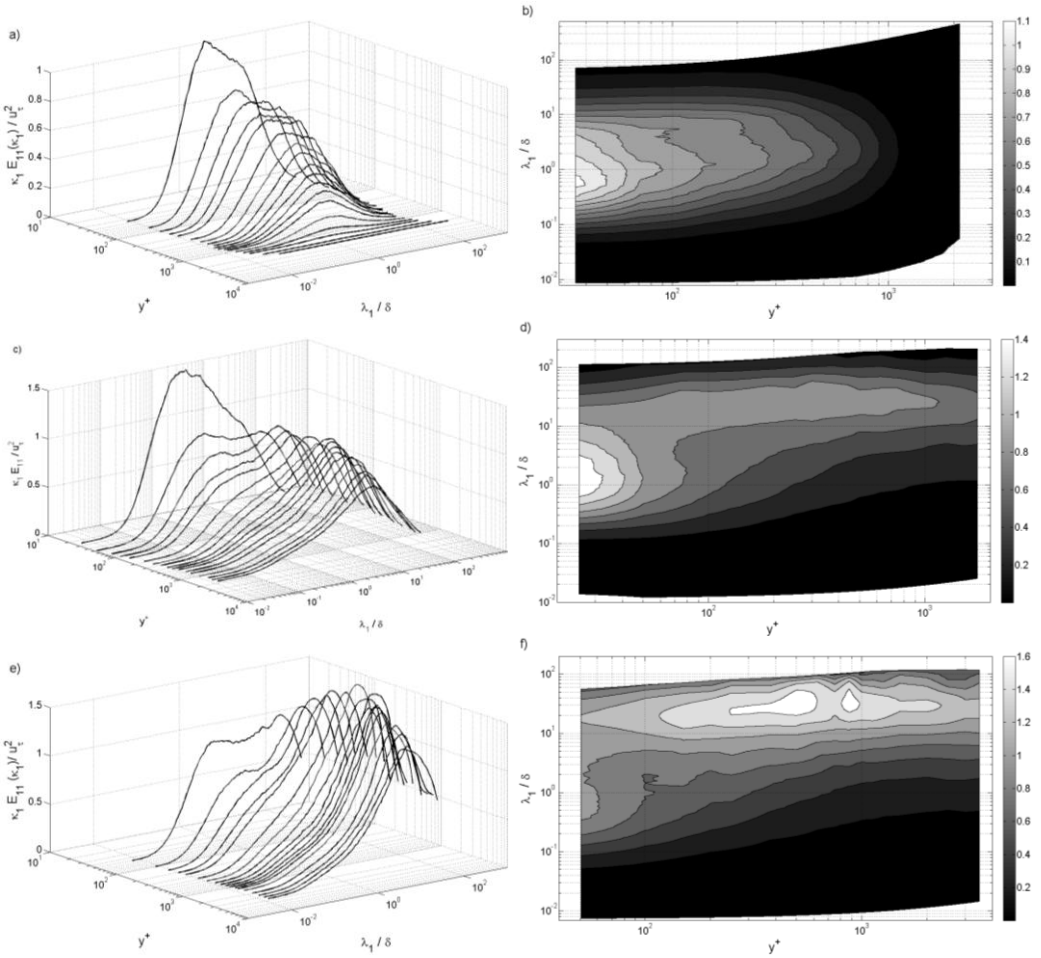


Figure 2.13: Three-dimensional (left) and contour plots (right) of energy spectra throughout the boundary layer: (top row) $Re_{\lambda 0}$ = 20; (middle row) $Re_{\lambda 0}$ = 260; and (bottom row) $Re_{\lambda 0}$ = 550.

Following Hutchins and Marusic (2007), the evolution of the energy spectra with height is shown as a three-dimensional graph in Figure 2.13 for each example case. The data is also presented as a contour plot, showing the three-dimensional plot from above. The top row in Figure 2.13 shows the energy spectra for the near-canonical

turbulent boundary layer. Near the wall, the inner peak dominates the energy spectra. Midway through the boundary layer, we see the double-peaked spectrum as observed by Hutchins and Marusic (2007). Finally, when the free-stream is reached, the turbulent energy spectrum is zero. For the cases with free-stream turbulence, there is a single distinct peak in the free-stream which persists deep into the boundary layer. For the intermediate case, the inner peak develops and then dominates around $y^+ = 40$. For the high free-stream turbulence case (bottom row in Figure 2.13), the free-stream peak is prominent throughout the boundary layer, even as the inner peak develops, to $y^+ = 70$. Comparing the energy spectra of the near-canonical case to the cases with free-stream turbulence, it is evident that the presence of turbulence in the free-stream has a significant effect on the structure of the boundary layer, even very near the wall.

We have presented the results of a study of the effects of free-stream turbulence on a turbulent boundary layer. The results were divided into three categories: a turbulent boundary layer with little to no free-stream turbulence, a developing turbulent boundary layer with significant free-stream turbulence present, and a developed turbulent boundary layer with high free-stream turbulence present. The mean flow and variance profiles were found to be in good agreement with previous studies (DeGraaff and Eaton 2000, Hancock and Bradshaw 1989, and Thole and Bogard 1996) for both the near-canonical case and the cases with free-stream turbulence present. We presented profiles of higher-order moments for the three cases. We observed that the presence of free-stream turbulence resulted in a reduction of the large-scale skewness compared to a canonical turbulent boundary layer. Free-stream turbulence tended to promote Gaussian behavior in the large-scale kurtosis. Spectral analysis showed a double-peaked energy spectrum, as observed by Hutchins and Marusic (2007) for a canonical turbulent boundary layer. They determined that the

large scale motions represented by the spectrum's outer peak had a modulating effect on the smaller scales of the boundary layer. The double-peaked spectra were also observed for the cases with free-stream turbulence present. The free-stream turbulence caused significantly higher energy in the outer (large-scale) peak, energy associated with the length scale of the free-stream turbulence. These results indicate that free-stream turbulence can have a substantial impact on the small-scales of the boundary layer. As shown in Figure 2.13, the energy from the free-stream penetrates deep into the boundary layer (to $y^+ < 100$).

CHAPTER 3

LAGRANGIAN PARTICLE TRACKING

Section 3.1: Introduction

Lagrangian particle tracking is a fundamentally important experimental technique that is being used to broaden the understanding of turbulent fluid flow since many properties of such flows are most evident in the frame of reference moving with a fluid particle. The temporal evolution of a turbulent flow and, more specifically, turbulent mixing and scalar dispersion are best studied from a Lagrangian perspective. The problem of the separation of two nearby fluid elements, or pair dispersion, is also a central component of the Lagrangian description of turbulence and is intimately tied to local concentration fluctuations that affect mixing and transport in turbulent flows (Bourgoin et al 2006).

In order to effectively conduct experiments using Lagrangian particle tracking, several key experimental and computational aspects must come together. The tasks in conducting experiments of this nature are: a) experimental design to create a particle-laden turbulent flow; b) use of proper diagnostics to record particle motion; c) the determination of the particle positions from the data collected; and d) the reconstruction of the particle motion as a time sequence from the data. Particle motion can be recorded using different technologies, e.g. acoustic techniques (Mordant et al 2001) or silicon strip detectors (Voth et al 2002, Mordant et al 2004), but the most popular has become high-speed cameras to record particle motion via images of illuminated seeded particles in the turbulent flow. Several important experiments have been conducted with a single camera: Ayyalasomayajula et al (2006) studied inertial

particle accelerations in a wind tunnel with grid generated turbulence and Gerashchenko et al (2008) also studied the accelerations of inertial particles but in a turbulent boundary layer evolving in the presence of grid generated turbulence. However, images from one camera can only measure the projection of the particle position onto the image plane of the camera to construct the particle paths and accelerations in the camera image plane. For experiments with a defined mean flow, where the accelerations of interest can be aligned with the two dimensions available on the camera (i.e. for Gerashchenko et al (2008), accelerations in the mean flow direction and normal to the boundary) a single camera may be sufficient, but the full three dimensional particle paths, and thus the true three dimensional particle accelerations, can only be determined using multiple cameras. Bourgoin et al (2006) and Ouellette et al (2006) use three cameras to image the turbulent flow between counter-rotating discs and determine particle acceleration from these data. With multiple cameras, the Lagrangian tracking scheme must include an algorithm for reconstructing the 3D coordinates of the particle from its corresponding 2D image plane coordinates on each camera. Finally, the Lagrangian tracking algorithm must connect the particle positions in time through the recorded image sequence.

In order to be effective in Lagrangian particle tracking, the particle finding algorithm must fulfill certain criteria: sub-pixel accuracy, speed, the ability to deal with overlapping particle images, and robustness to noise. The accuracy of the particle trajectories depends directly on the accuracy of the particle finding algorithm. In typical experiments, particle images can cover several pixels. In order to achieve accurate results from these images, measurement error is reduced by using a curve fit to the intensity profile, which allows the center of the particle to be found to a finer resolution than the pixel size (Ouellette et al 2006). The speed of particle finding must

be sufficient to deal with the high data rates from the cameras in experiments involving flow at a large Reynolds number. Although the number of particles in the measurement volume should be limited to avoid particle overlap on the image of a single camera, particle seeding density must also be high enough to obtain reasonable statistics, and so overlap will occur. The particle finding algorithm must have a method of determining the separate particles from overlapping images. Finally, the cameras used in these experiments will record noisy images, and particles must be located despite the inherent noise.

The particle finding algorithm used in our Lagrangian tracking scheme fits two one-dimensional Gaussian functions to the intensity profile of a particle on the image (Ouellette et al 2006). The peaks of these Gaussians determine the position of the particle center in image coordinates. One Gaussian determines the horizontal position of the particle center and the other determines the vertical position.

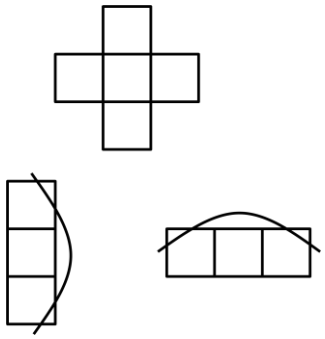


Figure 3.1: Sketch of five consecutive 'pixels' and the two 1D Gaussian fits to determine particle center on the image plane.

This method is less computationally demanding than fitting a fully 2D Gaussian to the pixel group, but still retains some benefits of using a Gaussian fitting; namely, accuracy and the ability to handle overlapping particles using N pairs of 1D Gaussian fits to the N local maxima in a particle cluster on the image. Some considerations for

the particle finding step are: as the particle seeding density for the experiment increases, the error will increase and the yield (total particles found) will decrease; as the particle image size increases, the error stays approximately the same (since overlapping particles become more likely with larger image, tending to decrease the accuracy, but larger particle images provide more intensity information to the algorithm, tending to increase the accuracy of the fit); and as the noise level increases the error increases. Finally, over-saturation of the image (when pixels are at the maximum intensity value the actual intensity may be much higher than the camera can record) can be a problem for particle center finding. When the image is over-saturated, the algorithm must try to determine a maxima in a collection of pixels all at the same (maximum) intensity. Although over-saturation is not a problem specifically addressed in the choice of particle-finding algorithm, care should be taken during the experimental design to avoid the issue.

Once the particles have been located in the image space of each camera, the three dimensional position of each particle in space is determined through stereoscopic reconstruction (stereomatching). Images of the measurement volume are simultaneously recorded by multiple cameras at different view angles. The location of the cameras in relation to the measurement volume (world coordinates) must be known, which requires calibration with images at a known location, usually using a grid of points called a calibration mask. Calibration is an important part of the Lagrangian tracking process, and will be discussed in more detail in a following section. After particle centers are found in the image plane, the 3D coordinates can be determined using the known positions of the cameras. The particle center must be the intersection of the camera lines of sight. From a known particle image on the first camera, A, the “line of sight” is a line passing through the center of the particle image

and the perspective center of camera A. This line of sight is then projected onto the image plane of camera B. Any particle on the image plane of camera B within a tolerance of the projection of the line of sight is added to a list of possible matches. A list is compiled for each pair of cameras and these particles are checked for consistency. Only particles that occur on all cameras are kept. Since the 3D positions are only determined for particles with images on all of the camera image planes, it is important that in the experimental set up that the cameras image an overlapping volume. Any part of the image plane on one camera that is not shared on the other cameras is not utilized in the stereomatching of particles in the Lagrangian tracking scheme described here. Finally, the three-dimensional coordinates for a particle are computed from the least-squares solution of the line-of-sight equations (Mann et al 1999).

When the particle positions are known, they are connected in time to form trajectories. In the scheme used here, a few consecutive frames are considered at a time. Consider a given particle, x_i^n = the i-th particle on the n-th frame. Building the track for this particle requires the determination of the particle x_j^{n+1} such that x_j is the position of the given particle in the n+1 frame. Ideally, there will be an x_j^{n+1} for each x_i^n, however, particles will go out of the image range on one or more cameras, overlap with other particles, or conflict (be possible matches for more than one particle in the previous frame), resulting in the termination of the current particle track and possibly the initiation of a new one. Conflict resolution has been proposed in other tracking schemes (Veenman et al 2001), but Ouellette, Xu and Bodenschatz (2006) determined that the most effective conflict resolution is to terminate the conflicting tracks and create a new one. In our Lagrangian particle tracking scheme, a known particle is matched to its new position in the subsequent frame using a four-frame, best estimate

algorithm. The particle that continues the track from a particle x_i^n in the n+1 frame is determined as follows: x_i^{n-1} and x_i^n are known, and from these two points, the velocity of the particle in frame n is calculated (from change in position over time). The velocity is used to estimate the position of the particle in n+1. The particle in the n+1 frame should be within a specified distance from this estimated position. There may be several particles that satisfy this criterion. These potential matches are used to estimate the possible positions of the particle in frame n+2, using the velocity and acceleration of the particle. The particle in the n+1 frame that gives an estimated n+2 position that most closely matches an actual x_j^{n+2} is selected to continue the particle track. The first two points in a track are chosen by nearest neighbor: the particle in the n+1 frame that is the smallest distance from the given particle in the n frame is added to the track.

When the tracks are constructed, relevant statistics such as acceleration means, variances and probability density function can be determined. The acceleration of the particle can be found from the second derivative of the particle position verses time. Mordant, Crawford and Bodenschatz (2004) calculate the acceleration of particles by convolution of the tracks with a Gaussian smoothing and differentiating filter. A parameter to consider is the time of a particle trajectory over which the acceleration is determined, or the fit time interval, τ_f. Voth et al found that for their frame rate and position measurement error, the best fit time for sampling accelerations was the Kolmogorov time scale, $\tau_\eta = (\upsilon/\varepsilon)^{1/2}$ where υ is the kinematic viscosity of the fluid and ε is the energy dissipation per unit mass. When longer fit times are used, the fit doesn't follow the real trajectory adequately and the particle acceleration tends to be under-estimated. When shorter fit times are used, the fit corresponds to measurement

error and the particle acceleration tends to be over-estimated. Gerashchenko et al (2008) also used a fit time of one τ_η to determine particle accelerations.

The components that make up the Lagrangian particle tracking experimental technique from recording the particle images to determination of particle acceleration have been described theoretically above. The rest of this chapter contains specific details on the calibration, particle tracking and data processing codes developed by the Bodenschatz group, Greg Voth and others. These codes are the framework currently in use for Zellman Warhaft's group at Cornell University.

Section 3.2: Camera Calibration

In order to conduct Lagrangian particle tracking, the position of the recording device relative to the measurement volume must be known. When cameras are used, the position can be determined using images of an object with known 3D coordinates via calibration. The calibration problem can be summarized as the need to compute camera intrinsic parameters (internal geometric and optical characteristics) and extrinsic parameters (relative orientation) based on a number of points whose object coordinates are known in the world frame and whose image coordinates are measured. Camera calibration therefore requires the determination of a large number of parameters. These parameters can be found by a large-scale non-linear search or a set of linear equations which ignores the dependency between parameters and any camera lens distortion. The large-scale non-linear search is computationally demanding and depends strongly on a good initial guess to start the search. Solving linear equations is much simpler, but the accuracy of the calibration is significantly reduced. Tsai (1987) proposes a different approach: using the radial alignment constraint, which says that

the only distortion present in the system is radial, and thus the direction of the vector between the camera optical center and a point in world coordinates is not affected by this distortion nor changes in the effective focal length, *f*. The radial alignment constraint is a function that concerns a subset of the parameters to reduce the dimensions of the parameter space: specifically a function of the relative rotation and translation matrices between the camera and the recorded calibration points. Using this approach, the resulting problem is non-linear but easily solved. Lens distortion is also accounted for to achieve sub-pixel accuracy. Matlab codes have been developed by the Bodenschatz group to implement Tsai's radial alignment constraint method and output the necessary camera parameters for use with the Lagrangian tracking algorithm described in the subsequent section.

The first step in camera calibration is to record images of an object with a known position. The best approach to calibration image acquisition is to use a calibration mask: a precision generated grid of points on a transparent medium with a distinguishing mark to indicate an origin. The mask is illuminated and the points recorded by all cameras in the measurement configuration. Images should be taken with the mask at several different positions: move the mask a small amount in the direction perpendicular to the grid of points; a translation stage is recommended. The position along the translated axis of each set of images (in mm) needs to be recorded in a .dat file. Comments in this file can be made using the # character to comment a line. This file and the .mcin files from the cameras are processed with a Matlab program PTVSetupPrep.m to compute the calibration parameters for the Lagrangian experiments. PTVSetupPrep.m is a very user-friendly program. All needed inputs are handled as prompts in the command line: number of cameras, the moving axis, the name of the axis positions file, grid spacing (two inputs to allow for non-square grids)

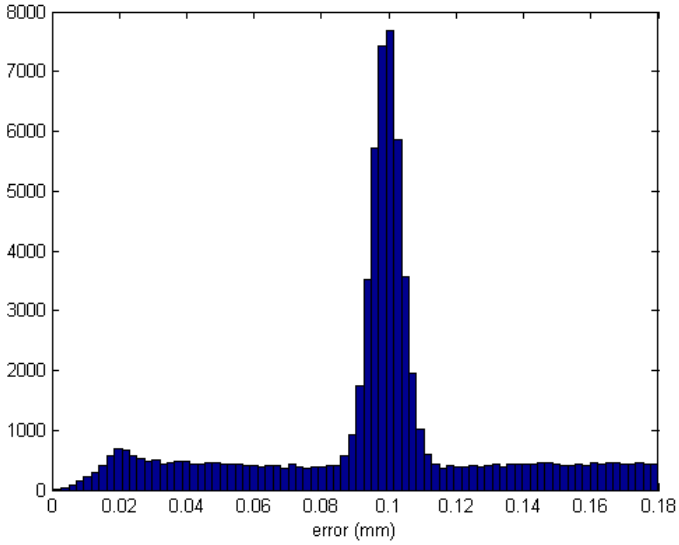
and other information is input by the user. For each camera, the program will prompt for the number of movie files (corresponding to the number of positions of the mask that were recorded) and for each movie, the user will enter the .mcin file name. Enter the movies in the order corresponding to the positions recorded in the position .dat file. For each movie, a figure window will open displaying the recorded mask image. The image can be divided into sub-regions where the point centers will be determined. Input the number of sub-regions when prompted and then select the area on the figure with the mouse. One region enclosing all points is sufficient if the image is not noisy. If there are spurious particles in the measurement volume or other spots on the image, they can usually be recognized and removed at a later step. Alternatively, the region can be divided into as many sub-regions as calibration points, and a small region around each point selected. The program will then prompt for the intensity threshold and minimum particle size in pixels used to identify the grid point images. Once the points have been identified by the program, it will ask the user to flag three base points and give their indices. This process is repeated for each of the movies and for each camera. Now, the 3D world coordinates are known and are recorded in a file calibpointspos.cam*.dat for each camera. The information is passed to the function calib_Tsai, which solves the radial alignment constraint using singular value decomposition. Finally, the PTVSetupPrep program writes the results to a configuration (.cfg) file. The .cfg file contains the number of cameras as well as relevant parameters for each camera including the size of the images (in pixels), the effective focal length (mm), the radial distortion, and the rotation and translation matrices and their inverses.

At the end of the configuration file are two parameters, ‘mindist_pix’ and ‘maxdist_3D’ that are used for the 3D matching (stereomatching) step of the

Lagrangian particle tracking algorithm. These parameters are not determined in the calibration; rather, they must be chosen by the user and manually changed if values other than the default are required. These parameters indicate how far to search for a matching particle image when reconstructing a particle's 3D position. The parameter mindist_pix is the maximum number of pixels on the image plane the algorithm searches to decide if a particle center seen on the image of one camera is a possible match with a particle center on another camera's image plane. The parameter maxdist_3D is in mm, and it is used in a later step of the stereomatching process. Ideally, the lines of sight connecting each camera center to the particle image would intersect at one point: the 3D coordinates of the particle. However, due to a number of errors involved in the process, this is rarely the case. The particle location is defined as the point which has the lowest sum of square distances to each line of sight. If the mean distance to the lines of sight for the particle is larger than maxdist_3D the match is discarded. Large distances can occur, especially when the cameras are in the same plane, and it usually means that the match does not correspond to a true 3D particle position. The default values for these parameters are 1.5 pixels for mindist_pix and 0.1 mm for maxdist_3D. In our experience, these values are a lower bound, and we use less stringent constraints to achieve more matches. At this point, the only way to determine the optimal values for these search parameters is trial and error. The tracking algorithm described in the next section outputs a log file that includes statistics on the number of particles found and matched, as well as the mean and r.m.s. value of the distance of the 3D particle positions from the camera lines of sight. The matching parameters should be set such that a reasonable percent of the found particles are matched, the mean 3D distance does not get too large and, most importantly, the tracking program can find sufficient trajectories to calculate velocity and acceleration statistics.

While the calibration as described above is sufficient to conduct the Lagrangian particle tracking experiments, more accurate calibration may be required, especially when the experiment is of long duration as the experimental set-up may have slight drift over time. A way to improve the calibration without having to redo measurements with a calibration mask (which can be time-consuming) has been developed by Greg Voth and adapted by Voth and the author to match the calibration method and tracking algorithms used at Cornell. The method has been termed dynamic calibration and it makes use of found particles from movies taken during the experiment to adjust the calibration parameters in order to improve particle matching and accuracy. Dynamic calibration is conducted through the following steps: 1) acquire a calibration using a mask of points as described above, 2) analyze actual stereomatched particles in the fluid to obtain 2D and 3D matched positions using the initial calibration, 3) calculate the matching error statistics for unambiguous matches using the current calibration (the distance between the particle position and the lines of sight to the cameras), and 4) iterate a non-linear search to adjust the calibration parameters to decrease the mean distances to the lines of sight. The dynamic calibration is conducted using the Matlab program gv_dynamic_calib.m. After a mask calibration is conducted, a movie of the particles is recorded by the cameras, and the tracking code is used to match particle images between cameras to give a 3D position. The tracking code can be set to output a data file that contains the 3D particle coordinates for matched particles, as well as the matching error and the 2D image positions of the particle as seen on each camera. This file, which uses the naming convention {_movieprefix}_dist.dat (see Section 3.4), is used as an input along with the calibration .cfg file to the Matlab dynamic calibration code. The program first selects a subset of the points to use in the dynamic calibration. The 3D matching is

repeated by the Matlab program for these points using the calibration .cfg in the same manner as the 3D tracking code (see next section). The matching error from this



reconstruction can be compared to the matching error from the tracking code (from the data file), as shown in Figures 3.2 and 3.3 for an example case.

Figure 3.2: Matching error histogram as calculated by the stereomatching code

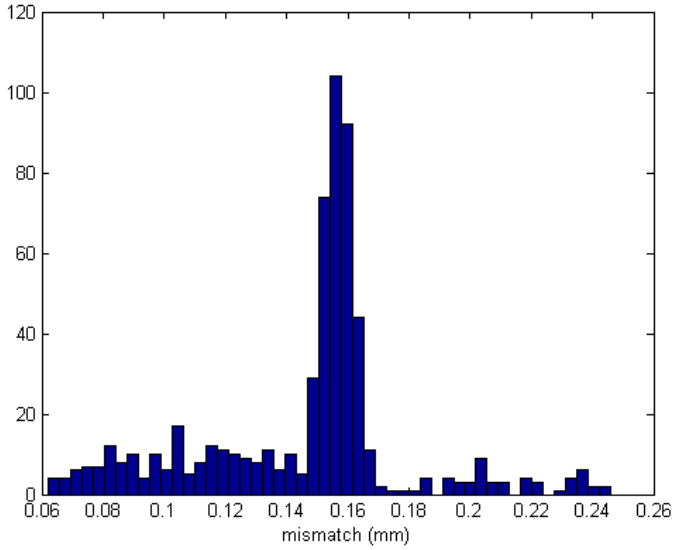


Figure 3.3: Initial matching error histogram as calculated by the dynamic calibration code

These histograms should show a similar distribution in order to achieve a meaningful recalibration. The Matlab code then changes the rotation matrix from the

configuration file into the three corresponding angles (see Tsai 1987). These three angles and the translation vector are the only parameters to be optimized by this code. Effective focal length, distortion and other parameters are held constant. The code completes a non-linear optimization of the rotation angles and translation vector to minimize the matching error. For the same example case as shown above, the mismatch error has the following distribution after the first optimization.

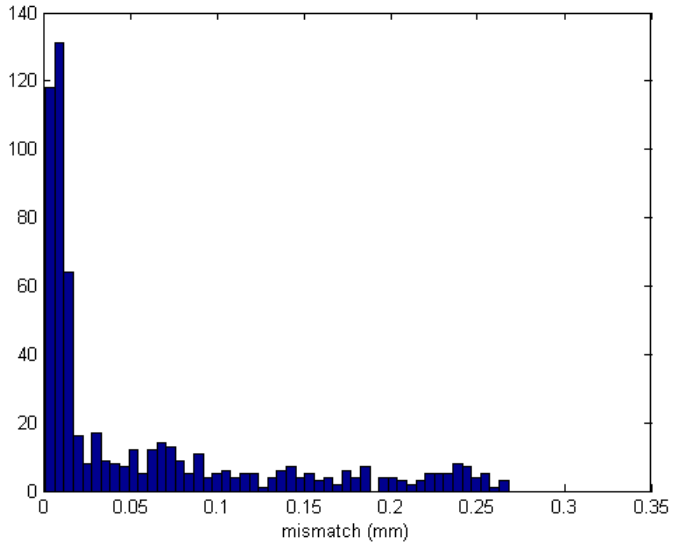


Figure 3.4: Matching error histogram after first optimization

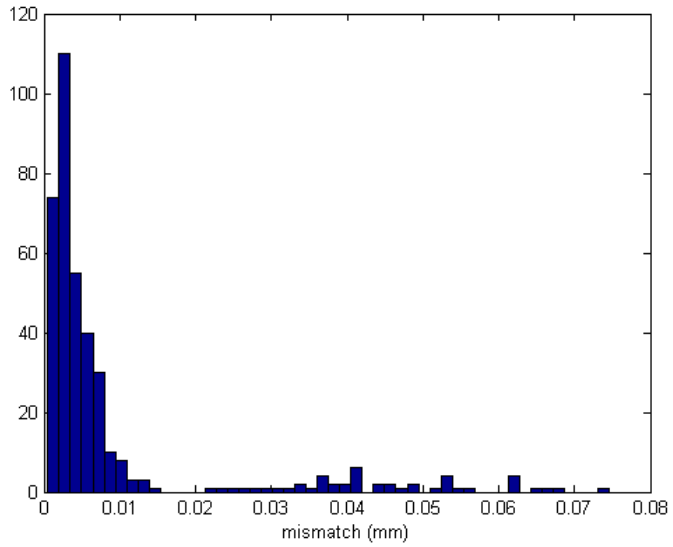


Figure 3.5: Matching error histogram after optimization on good matches

The mismatch error of these particles is much smaller compared to the initial distribution. The optimization is performed a second time, this time on "good" matches, that is, particles from the .dat file that have a matching error less than a given threshold (in mm). As shown in Figure 3.5, the error distribution has shifted even further to smaller error after this second optimization.

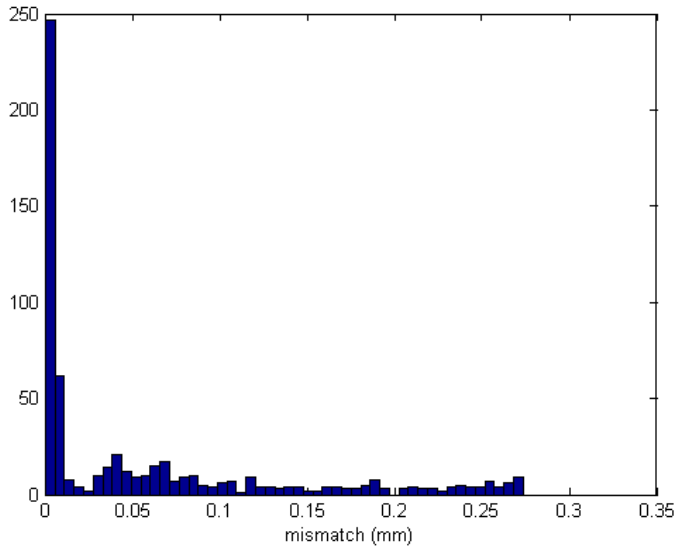


Figure 3.6: Matching error histogram using new calibration parameters on particles not used in the calibration optimization

The new optimized calibration is used to construct 3D positions for particles from the original data set that were not chosen in the initial subset to be optimized, in order to check the validity of the new calibration parameters. The resulting error distribution, Figure 3.6, is significantly improved compared to the initial error distribution. Finally, the program writes a new configuration file with the improved parameters.

Dynamic calibration is a useful tool to achieve more accurate camera parameters for use in the tracking algorithm, and can be used to correct for errors in the initial calibration; however, the importance of a good calibration with the mask must be

recognized. Most importantly, the dynamic calibration cannot correct for a small overlap volume between cameras. Particle matches used in the dynamic calibration can only be determined for particles seen by all cameras. When conducting Lagrangian particle tracking experiments, the calibration should be checked before the bulk of the experiment is conducted. The alignment of the cameras can be checked with the Matlab function checkalign.m, which takes the calibpointspos.cam*.dat files from the mask calibration, the image size in pixels for the calibration and the experiment, and the number of cameras as inputs. The program outputs a graph of the projected overlap area.

Section 3.3: Tracking Code

We use the result of the calibration process, namely the .cfg file, as an input in combination with the tracking code to determine Lagrangian particle tracking. This section presents a walk-through guide of the 3D particle tracking code created by the Bodenschatz group. It is organized according to the arrangement of the main controller program, with focus on the details of the unique data structures and algorithms that are used. The following section will detail the recent additions by the author and Greg Voth.

The Bodenschatz code is designed to work with the Phantom v7.1 cameras to conduct particle finding, stereomatching and tracking from movie image data in real time for a Lagrangian experiment. To conserve hard disc space, only the post-processing track data files are retained; the raw videos are usually discarded. While this approach minimizes storage, it is often desirable to retain the raw experimental data. With this in mind, versions of the controller program were created which could output the

results from the particle finding algorithm (found particle center locations) or take this output data file as an input to continue with the stereomatching and tracking processes. Outputting the particle centers before stereomatching and tracking allows the user to make changes in the program settings which may be necessary after data has been taken, for example, changes to the stereomatching limits discussed in the previous section.

The 3D tracking code consists of several C++ scripts and header files, all of which are contained in the directory TrackingCode/. The main controller program is located in the directory TrackingCode/working/ and is named controller_{identifier}. The original code as delivered by Haitao Xu of the Bodenschatz group is named controller_node. The two edited controller programs are controller_2Dout and controller_2Din, which output and take as input the found particle data, respectively. Header files needed by the controller code are stored in the directory TrackingCode/include/. The corresponding C++ and compiled object files are stored in the directory TrackingCode/lib/. The walk-through details the original Bodenschatz code first, and then Section 3.4 will describe the program with changes made by the author.

Section 3.3.1: controller_node

The Lagrangian tracking program requires several inputs. These can be viewed by running the program without any inputs (typing ./controller_{identifier} into the command line). The code will display the following message:

```
mozart:working $ pwd
/mnt/raid/hd2/stephanie/Code/TrackingCode/working
mozart:working $ ./controller_node
Usage: ./controller_node <directory> <filename prefix> <number of cameras> <threshold
for EACH camera> <num of nodes> <config filename> <track file directory> [<start movie
 number> [<stop movie number> [keep]]]
mozart:working $
```

As the message suggests, the tracking code requires the following inputs to execute correctly:

- Directory: the file directory where the movie files or particle center data files are or will be stored
- Filename prefix: a set prefix that, along with the movie count, makes up the movie file name. Movie files for a given data set must all be saved with the same prefix for use with this code.
- Number of cameras: the number of cameras used in the experimental set up. Currently the maximum number of cameras allowable is four.
- Threshold for each camera: the lowest intensity value at which a particle will be indentified in an image. This input number is repeated for each camera.
- Number of nodes: the number of nodes that will be used to conduct the calculations. For reprocessing after the experiment has been completed, one node is sufficient.
- Configuration file name: the name of the .cfg file output by the calibration code. If this file is saved in a different directory from the controller program, the directory path must also be included in the input.
- Track file directory: where the output track data will be written. Output track files will use the same prefix as the corresponding movies.
- Start movie number: the first movie to be processed.
- Stop movie number: the last movie to be processed.
- Keep: a binary value that determines whether or not the movie file is retained or deleted after tracking.

After the controller program receives the inputs as described (incorrect inputs will result in error messages), it creates a PTVSetup object, ptvsetup, whose constructor reads the camera configuration parameters from the .cfg file. All of the information in the .cfg file is stored in variables in this class. Any text following the comment character (#) on a line is ignored. The first information in the configuration file, the number of cameras, is stored in the variable ncams, which must be less than the maximum number of cameras or it will cause an error. Then the camera parameters for each camera are read in and stored in corresponding variables using the input version of the Camera::camera function. Finally, ptvsetup reads the laser beam specifications and the tolerance parameters for the 3D matching from the .cfg file.

The PTVSetup class also contains the definitions for vectors relating camera positions to one another. These vectors are then used in the stereomatching portion of the algorithm.

With the relevant configuration information now available to the program, it initiates a loop through the movies to be processed. For each movie number, there will be a .mcin file for each camera. The program looks for a file that matches the pattern {filename prefix}{movie count}.cam{camera number 0-ncams}.mcin in the directory specified in the program inputs. For example, prefix_0.cam0.mcin would be the first movie from the first camera; prefix_0.cam1.mcin would be the corresponding movie (taken simultaneously) from the second camera. Once the correct mcin file is located, the program will read in the information from the file. The mcin movie files used in this tracking scheme are binary files with the following information stored from the beginning of the file: the header (which includes information about the size of the movie), the black reference (a frame that gives the intensity value of the image background) and the intensity values for each frame by pixel in the movie. The header consists of the following information: the number of columns of pixels in a frame (2 bytes), the number of rows of pixels in a frame (2 bytes), the number of frames in the movie (4 bytes), the frame rate of the camera during recording (frames per second, 4 bytes), and the exposure time (4 bytes). The movies are interpreted into an MCine object; the header and black reference information is stored in corresponding variables within the MCine class.

Once the controller program has stored the movie file information, it passes the pixel intensity values to a GaussControl object. Then it calls the function GaussControll::FindCenters( ). As the name implies, this function determines the

location of the particle centers on the image. It walks through the array of intensity values and determines those pixels which are a local maximum above the intensity threshold (see the function GaussControl::IsLocalMax). This pixel and the four pixels directly adjacent (to the left, right, above and below) are used to calculate the particle center in image coordinates. This function also calculates the width and height of the pixel image. The center locations, particle image widths, and intensity values for all the found particles in the current frame are stored in a Frame object with the function GaussControl::CreateFrame( ). The controller program repeats this process for each frame in the movie, and for each camera. At the end of this process, a vector of frames will be compiled with the found particles for each frame from each camera's mcin file.

Once the particle centers are found, the controller program initiates the stereomatching process. The vector of frames compiled by GaussControl is passed to the FrameSet object, fs. The controller program notes the time when this step is reached in the log file StereoMatch.log (see Section 3.4 for details on the log file). Then the function FrameSet::Match3D is called by the controller program. The function takes ptvsetup (calibration parameters) information as an input. Match3D calls the function MatchFrame, which is in the FrameSet object and takes ptvsetup and the particle positions for the frame as inputs. MatchFrame uses the function Pairs (also defined in the FrameSet object) to look at each pairing of consecutive cameras 'A' and 'B' (i.e. camera 0 and camera 1 together, camera 1 and camera 2 together, etc.). The function projects particle images from camera A to the image plane of camera B in order to find particle images that most likely corresponded to the same particle in space when the images were recorded. Any particle on the image plane of B that is within a tolerance, mindist_pix (see calibration discussion, above), of the projection from A is

added to a list of pairs of particles for these two cameras. To follow the code for this portion of the algorithm, the user must keep track of several vectors, indicating the particle location or camera center location coordinates in a variety of reference frames. First, the image coordinates for a given found particle on the image plane of A are used with the calibration parameters to determine the vector connecting the particle image to the center of camera A. This vector, xpwA, is referred to as the "line of sight" for this particle. xpwA is projected onto the image plane of camera B, the coordinates of this projection on the image plane of B are stored in the vector xpiAB. The vector from this position (the projection of the line of sight on image plane B) to the location of the center of camera A on image plane B is xppAB. This vector xppAB is then projected onto the vector Lnorm, which is a vector perpendicular to the vector on the image plane of B that connects the two camera centers on the image plane (uCC). This process is repeated for each found particle on image plane A. The found particles on the image plane of camera B are also projected onto this same Lnorm. Finally, the particles with nearly the same coordinates on Lnorm are paired and stored in the array pairAB.

Each list of particle pairs (one for each camera pairing) is checked for consistency. Only the particles that are on each list are retained for the final 3D stereomatching. Match3D calls the function Pos3D. Pos3D determines the particle position in 3D coordinates. The location of the particle is determined using the lines of sight from each camera to the particle location on its image plane. Ideally, these lines would all intersect at the particle 3D location. However, this does not actually occur and the 3D position is determined such that the sum of the perpendicular distances from the particle 3D position to each line of sight is minimized. Pos3D returns the coordinates for the stereomatched particle as well as the measure of the average distance from the

lines of sight for that particle. If this average distance exceeds the parameter maxdist_3D (see calibration discussion above), it is a bad match and will not be used in the tracking portion of this algorithm. Finally, a frame object fs3D stores the array of frames that contain the 3D particle position information. Statistics such as the number of found particles and distance from particle positions to the line of sight are updated. The frame object fs3D is returned to the controller program (stored in 'matched') for use in the particle tracking step to follow. The log file is updated with the total time spent stereomatching for the movie, the average number of particles per frame for each camera, the average number of 3D matches per frame, the average number of overlapping particles, the mean distance of the particle 3D position from the lines of sight to the cameras, and the RMS of this distance.

The controller program now has the vector of frames that contains the three-dimensional coordinates of the particles seen by all cameras. The next step is to connect the particle positions in time to form a track. The controller code creates a Tracker object (t) which includes the vector of matched frames (frames containing 3D particle position information), frame rate, exposure time, mean threshold, and the output file name. Then the controller program calls the function Tracker::MakeTracks, which connects the particles into tracks and outputs the results to a separate file. In MakeTracks, the tracks that can still have points added to them (active tracks) are stored in the list 'activelist.' First, tracks are initialized: one new track for each particle in the first frame is added to activelist. The function loops through each frame, ending two frames from the end of the movie file since the algorithm, as described in Section 3.1, uses the two future frames to determine the best location for the particle position in the next frame. A LinkMatrix (see lib/LinkMatrix.cpp for definition) object 'links' is created to store possible matches.

The number of rows of the matrix is the number of active tracks and the columns are the number of particles in the next frame. This matrix will be populated with the distance between particle position estimates and actual particle positions or particle acceleration estimates and actual acceleration, depending on the initial track length (see below). The function loops through each of the active tracks, using the most recent two points in a track to estimate the velocity of the particle and the position in the next frame (line 209 of Tracker.cpp). If the track has only one point, the next point is determined by the nearest neighbor method: the particle in the next frame that is closest to the current particle position is used to make the velocity and position estimates (lines 149-206). If the track has more than two points, the particle acceleration is also determined (line 214). Then the program loops through the particles in the n+1 frame, finds all particles within a threshold distance (inside a sphere of radius R1) of the estimated position, and determines an acceleration of the particle using each of these points (line 240). For the four frame method, the position of the particle in frame n+2 is estimated using the original position and acceleration and velocity estimates for each possible particle location in the n+1 frame determined in the previous step. The actual particle position in the n+2 frame that is closest to the estimate is determined. The distance between this particle and the estimated position is stored in the link matrix (in the row corresponding to the current track and the column corresponding to the particle number in the frame n+1). When these steps have been completed for the current frame, the matrix is collapsed by removing all entries but the smallest non-zero entry in a row. The particle that corresponds to this distance is added to the corresponding track provided there are no conflicts (i.e. that same particle also continues another active track). In the case of conflicts, the track is terminated and the conflicting point becomes a new active track. If there is a particle in the frame that does not continue any active tracks, it becomes a new track. If a

track has not be updated (added to) this frame, an estimate of the position for the next frame is made; if the track has not be updated for a time longer than a set time, PTVc::OCCLUSION_TIME, it is removed from the active tracks list. Finally, the compiled tracks that are longer than a certain length are output to a file named tracks_{prefix}_movie#_.cam{ncams+1}.dat.

The controller program repeats this process (particle finding, stereomatching, tracking, data output) for each movie in the data set specified in the input. When the tracks have been recorded to the output file, the controller program deletes the movie files unless the input "keep" was specified. This concludes the controller program walkthrough.

Section 3.3.2: Makedata

To obtain velocity and acceleration information from the output tracks, post-processing is needed. The program currently in use by Z. Warhaft's group to do so is Postproc/makedata. Makedata requires the following inputs:

- input_dir: the directory where the track files are stored
- fname_prefix: the track file prefix (including 'tracks_')
- fname_postfix: the track file postfix (.cam{ncams+1}.dat)
- output_dir: the directory to output velocity/acceleration results to
- output_tag: the output filename prefix
- nstart_mov: the number of the first movie to process
- nstop_mov: the last movie to process
- velfilterlen: the filter length (in number of frames) for the velocity calculation (see below)
- velfilterwidth: the filter width (in number of frames) for the velocity calculation (see below)
- accfilterlen: the filter length (in number frames) for the acceleration calculation (see below)
- accfilterwidth: the filter width (in number frames) for the acceleration calculation (see below)

- method_flag (G/P): flag to tell the program which method of fitting to use. P is parabolic least squares method. G is for the Gaussian method (see below)

After receiving the inputs described, makedata reads in the track file information, which is stored as follows from the beginning of the file: the header, and the list of track points. The header consists of the number of tracks (4 bytes), the frame rate (4 bytes), 'exposure' (4 bytes), the intensity threshold (1 byte), 'max_interpolated' (1 byte), and 'exposure' (2 bytes). 'Exposure' and 'max_interpolated' are placeholders for data that are not used in the current application. After the header, the track file contains the following sequence for each track: first, the number of points in the track (4 bytes); then, for each point in the track, the frame number (unsigned long int, 8 bytes), the x, y, and z positions (float, 4 bytes each), the intensity distributions σ_x and σ_y (float, 4 bytes each), the intensity of the particle image in that frame (unsigned char, 1 byte), and whether or not that point was an actual particle image or found via interpolation of the tracking code (unsigned char 1 byte). When the track information has been read in, the velocity and acceleration can be determined by the first and second derivatives of the particle position with time. In order to calculate the derivatives, a function is fit to a portion of the track (determined by the filter inputs) and the derivative of the function is taken. Voth et al (2002) determined that the best fit interval to use when calculating particle acceleration was on the order of the Kolmogorov time scale, τ_η. Larger fit times tend to underestimate the acceleration because the real trajectory of the particle is not resolved. Smaller fit times tend to over-estimate the acceleration because the fit mostly corresponds to the position error or noise. When used with makedata, the filter length for both velocity and acceleration should be the Kolmogorov time in number of frames. The filter width for both velocity and acceleration is one third of the length value.

Two different functional fits to the track are available in the makedata program: a parabolic fit or a Gaussian fit. The parabolic fit method was used by Voth et al (2002) and La Porta et al (2001), but Mordant et al (2004) determined that a Gaussian kernel [3.1] gave more accurate results.

$$g(\tau) = \frac{1}{\sqrt{w^2\pi}} \exp\left(-\frac{\tau^2}{w^2}\right) \quad [3.1]$$

Where τ is time and w is the filter width. The differentiating kernel is the second derivative of the Gaussian kernel:

$$\kappa(\tau) = \frac{2}{\sqrt{\pi}w^3}\left(\frac{2\tau^2}{w^2} - 1\right)\exp\left(-\frac{\tau^2}{w^2}\right) \quad [3.2]$$

The acceleration is then obtained by convolution $a = k * x$ where x is the spatial coordinate of the particle. After calculating the velocity and acceleration, makedata outputs three files: a velocity data file (named {fname_prefix}+{output_tag}.vel_field.dat), an acceleration data file (named {fname_prefix}+{output_tag}.acc_field.dat) and a statistics file (named {fname_prefix}+{output_tag}.stat_field.txt). The velocity data file contains the following information in columns: movie number, track number, frame number, x position, y position, z position, velocity in the x-direction v_x, velocity in the y-direction v_y, velocity in the z-direction v_z. The acceleration data file contains the following information in columns: movie number, track number, frame number, acceleration in the x-direction a_x, acceleration in the y-direction a_y, acceleration in the z-direction a_z. The stat_field file is a text file that contains relevant statistics such as the filters used, the mean and variance velocity and acceleration and the total number of data points used to compute the statistics. Additional post-processing codes are available in the Postproc directory.

Section 3.4: Edits to the Tracking Code

In the course of conducting Lagrangian particle tracking experiments, the tools used to gather and analyze data will evolve to meet the current experiment's specific challenges. This section presents information on versions of the code created to accommodate a particle tracking experiment conducted by Greg Voth and Rachel Brown of Wesleyan University with the author at Cornell in early 2008. Although these changes do not affect the overall particle finding, stereomatching and tracking processes described earlier, they are discussed here to illustrate the evolving capacity of the code and to serve as a record of the changes since, as with any complex computing code, the risk of poor documentation can hinder future users.

The experiment studied the Lagrangian motions of large particles (particles with diameter greater than the Kolmogorov length scale) in a turbulent flow. The experiment was designed to study particle accelerations using neutrally buoyant particles to isolate the effects of large size from the effects of settling and clustering that may be present when the particle density is different from the surrounding fluid. The primary goal was to use a variety of particle sizes to determine the relationship between acceleration variance and particle diameter. This relationship is predicted by Voth et al (2002) to be of the form $< a_x^2 > \propto d^{-2/3}$ (for sufficiently large particles, determined by dimensional arguments and assuming that the motion of large particles will average over eddies smaller than *d*, effectively changing the viscosity seen by the large particles). In order to verify the relationship, measurements were made of spherical polystyrene particles in a von Karman flow between counter-rotating disks (see flow description in Voth et al (2002)). One of the most important requirements of an experiment of this nature (conducted by visitors over a short amount of time) was

to get as much data as possible since there was little to no opportunity to make additional measurements. Thus, ways of efficiently storing the raw data and reprocessing it were needed. Additionally, the stereomatching algorithm was not finding sufficient matches to generate convergent particle acceleration statistics. Due to the limited time table, the proper stereomatching parameters (mindist_pix and maxdist_3D) could not be determined beforehand, also making reprocessing a necessity. Finally, the calibration parameters needed to be improved to improve the matching. Thus, the code was modified to suit the dynamic calibration.

To conserve hard disc space, the version of the code as described in the previous section deletes the raw videos acquired from the cameras after the particle tracks have been determined. The input 'keep' can be specified to retain the video files, but saving the movies from an entire data set can exceed the available hard disc space. In order to accommodate the experiment described above, a version of the code was created that outputs the results from the particle finding algorithm to a binary data file. Found particle image information is saved for each camera. Thus, the information is condensed before any matching or tracking occurs, and changes that may affect those steps (such as changes in the calibration parameters) can still be made long after measurements have been concluded.

The program that outputs the 2D particle positions is called controller_2Dout. The controller_2Dout program operates in a similar manner to the controller_node program described in Section 3.3.1. The user inputs are the same. The data file will be saved in the same directory as the .mcin files, the directory specified by the user as an input to the controller program. The output filename for the 2D positions .dat file is stored in a variable called namename, and it follows the same filename pattern as the

.mcin file: {filename prefix}{movie count}.cam{camera number 0-ncams}.dat. The controller_2Dout program operates in the same way as the controller_node program initially. The calibration information from the .cfg file is read and stored, the .mcin file is located and read in, and the particle finding step of the algorithm is conducted using the GaussControll function FindCenters( ) for a given frame. After the conclusion of the particle finding step, the controller program calls GaussControl::WriteToFile with the variable namename and the current frame number as inputs to the function. WriteToFile opens the binary file for the corresponding movie in append mode (the data for each frame is added to the end of the file each time the function is called). For each frame, the function writes the following to the file: the frame number (int), the number of particles found in the frame (int), and for each particle: the x-position (double), y-position (double), the intensity (int), the particle size σ_x (double) and σ_y (double), and the center of mass c_x (double) and c_y (double).

After outputting the 2D particle information, controller_2Dout continues with the stereomatching and tracking steps of the Lagrangian particle tracking algorithm as described in Section 3.3. Thus, even in cases where the parameters are correctly pre-determined and the data will not need to be reprocessed, controller_2Dout can still be used to compute the particle track files in place of the controller_node program. The benefit of using controller_2Dout is to retain the raw position data in a compressed form. When reprocessing the data is necessary, the 2D .dat files can be read in instead of .mcin movies with the program controller_2Din. The inputs to controller_2Din are the same as described for controller_node; the directory input by the user will be the location of the .dat files. The controller_2Din program uses the function Input2D::load2Ddata to read in the particle information saved in the .dat file and

convert it into frames for use with the subsequent stereomatching process. The function load2Ddata takes the following inputs: the filename where the 2D positions are stored, a bool variable extra_data, and the name of the output vector of frames. The variable extra_data indicates whether the 2D position file was written to include the center of mass information for each particle. For the current controller_2Dout configuration, extra_data should be true. The output vector of frames is f[camid] to conform with the particle frame information needed for the stereomatching process. After reading in the particle position information, the stereomatching and tracking algorithms are conducted as described previously.

Although the output/input of the particle position information was the most significant change made to the tracking program, other edits to the tracking code were made. The input/output edits were made to allow reprocessing of the data and more feasible storage of the raw data. Additional edits were made to aid the development and implementation of the dynamic calibration discussed in Section 3.2. One simple edit adds the name of the calibration .cfg file used by the program to the StereoMatch.log output file. The log file also records the movies processed and statistics such as the average number of particles found for each camera and the average number of stereomatched particles. The file can be found in the same directory as the controller program. It is opened in append mode, so the file can grow quite large unless it is periodically renamed or removed. Adding the .cfg name to the log with the particle matching statistics allows the user to distinguish the effectiveness of different calibrations, and especially illustrates the improvement of a calibration using the dynamic calibration method.

The dynamic calibration program described in Section 3.2 requires data on the stereomatched particles as an input to improve the calibration parameters. This input file is obtained by running the controller program with the initial calibration obtained from the mask. It is output by the FrameSet function MatchFrame into the directory specified for the track files by the user in the controller program inputs. The file contains the 3D particle coordinates, the matching error and the 2D image coordinates on each camera for each matched particle in the movie. The filename is passed to the FrameSet object with the function SetMoviePre( ) which takes the movie name prefix ({filename prefix}{movie count}) as an input and stores it in a variable _movieprefix.

With these edits to the code framework, the development of the dynamic calibration was possible and reprocessing of the data could be conducted for the large particle experiment after the measurements were taken. Unfortunately, it was eventually determined that the measurements were conducted with a small image overlap volume, and even with the improved calibration parameters, there were insufficient tracks to achieve convergence of the acceleration statistics in three dimensions. However, data from the two-dimensional tracks was analyzed and important acceleration results from these measurements were obtained. These tracks showed good agreement with the scaling predictions of Voth et al (2002) as well as highlighting the lack of particle size dependence in the acceleration PDFs (Brown et al 2008).

APPENDIX

The calibration and Lagrangian tracking codes described in this document have been archived and can be obtained by contacting the author (sjn23@cornell.edu) or Zellman Warhaft (zw16@cornell.edu). The code to conduct the dynamic calibrations has many separate functions it depends on. Presented below is the Matlab dependence list so future users know what files they need. All these files are stored in the electronic archive. The flow chart on the next page shows the details of the Lagrangian particle tracking code described in Chapter 3.

Functions called by gv_dynamic_calib.m:

% load_from_PTV -- Reads in calibration parameters from a .cfg file
% calib_Tsai -- Calibrates camera parameters using radial distortion assumption.
(with subfunctions 'myload' and 'importfile')
% gv_rotmat2angles -- Converts a rotation matrix to 3 angles.
% gv_angles2rotmat -- Converts 3 angles to a rotation matrix.
% gv_dynamic_fitfunc -- Repacks parameters and calls gv_calc_ray_mismatch.
% gv_calc_ray_mismatch -- Calculates the distances between lines of sight.
% gv_imgplane2unitvector -- Called by gv_calc_ray_mismatch.
% gv_write_calib_cfg -- Writes the PTVSetup.cfg file after optimization of parameters.

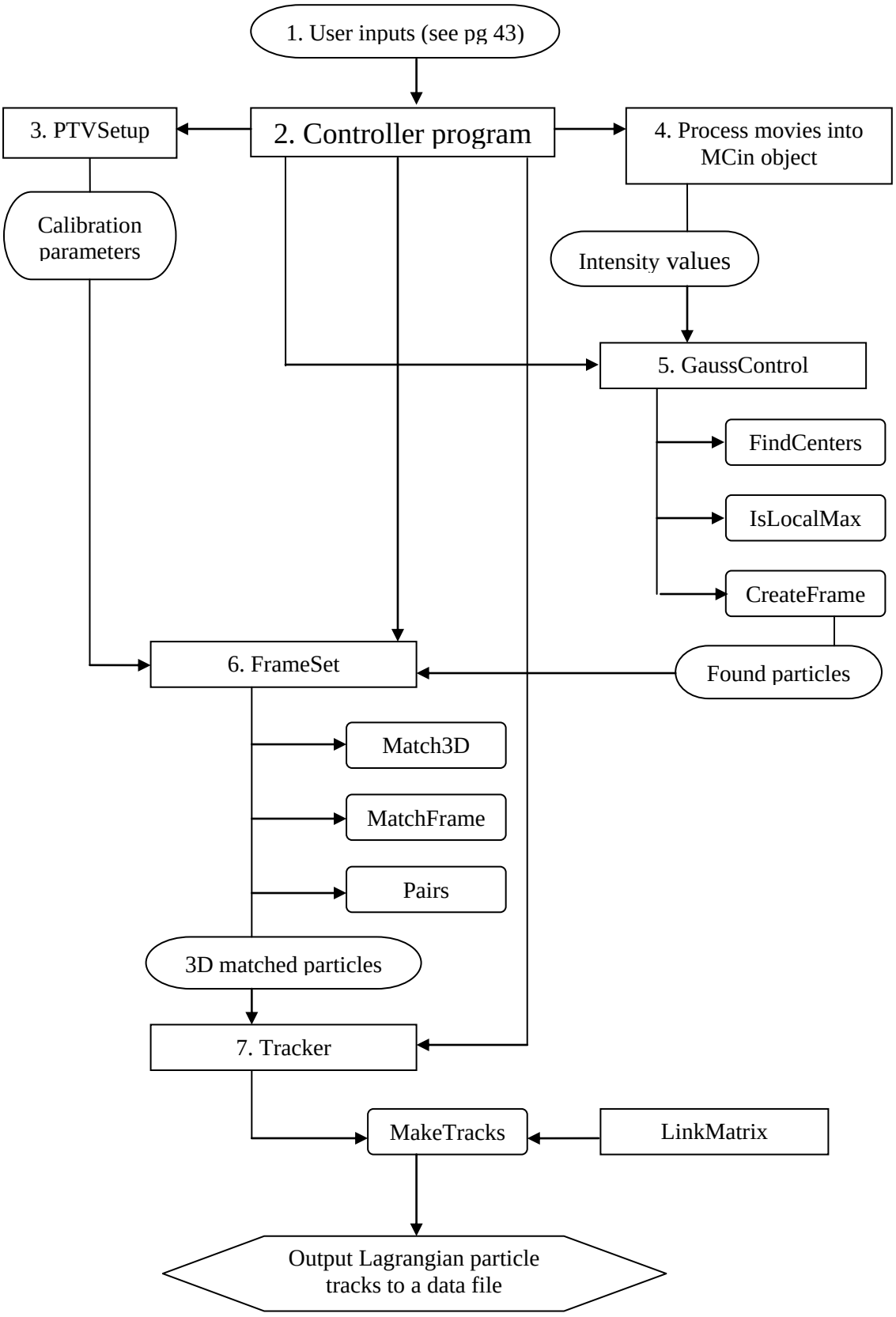


Figure A.1: Flow chart for Lagrangian particle tracking code

REFERENCES

Ayyalasomayajula, S., Gylfason, A., Collins, L. R., Bodenschatz, E. & Warhaft, Z. 2006 Lagrangian measurements of inertial particle accelerations in grid generated wind tunnel turbulence. Phys. Rev. Lett. 97, 144507.

Bott, D. M., and Bradshaw, P. (1997) Effect of high free-stream turbulence on boundary-layer skin friction and heat transfer. Mechanical Engineering Department, Stanford University, Report MD-75.

Bourgoin, M., Ouellette, N. T., Xu, H. T., Berg, J. & Bodenschatz, E. 2006 The role of pair dispersion in turbulent flow. Science. 311, 835–838.

Brown, R. D., S. Neuscamman, G. Voth, Z. Warhaft (2008) Effects of Particle Size on Acceleration Measurements in Intense Turbulence. Presented at 61st Annual Meeting of the APS Division of Fluid Dynamics, San Antonio, TX.

Bruun, H. H. (1995) Hot Wire Anemometry: Principles and Signal Analysis. Oxford University Press.

DeGraaff, D. B., and Eaton, J. K. (2000) Reynolds number scaling of the flat-plate turbulent boundary layer. J. Fluid Mech. 422:319-346.

Gerashchenko, S., Sharp, N. S., Neuscamman, S., & Warhaft, Z. 2008 Lagrangian measurments of inertial particle acceleration in a turbulent boundary layer. J. Fluid Mech 617: 255-281

Gylfason, A., Ayyalasomayajula, S. & Warhaft, Z. (2004) Intermittency, pressure and acceleration statistics from hot-wire measurements in wind-tunnel turbulence. J. Fluid Mech. 501: 213–229.

Hancock, P. E., and Bradshaw, P. (1989) Turbulence structure of a boundary layer beneath a turbulent free stream. J. Fluid Mech. 205:45-76.

Hunt, J. C. R. and Graham J. M. R. (1978) Free-stream turbulence near plane boundaries. J. Fluid Mech. 84:209-235.

Hutchins, N. and Marusic, I. (2007) Large-scale influences in near-wall turbulence. Phil. Trans. R. Soc. A 365: 647–664.

Kim, H. T., Kline, S. J., and Reynolds, W. C. (1971) The production of turbulence near a smooth wall in a turbulent boundary layer. J. Fluid Mech 50:133-160

La Porta, A., Voth, G. A., Crawford, A. M., Alexander, J. and Bodenschatz, E. (2001) Fluid particle accelerations in fully developed turbulence. Nature 409:1017–1019

Mann, J., Ott, S., Andersen J. S. (1999) Experimental study of relative, turbulent diffusion. Risø National Laboratory Report Risø-R-1036(EN)

Mordant, N., Crawford, A. & Bodenschatz, E. 2004 Experimental Lagrangian acceleration probability density function measurements. Physica D 193, 245–251.

Mordant, N., Metz, P., Michel, O. & Pinton, J. F. 2001. Measurement of Lagrangian velocity in fully developed turbulence. Phys. Rev. Lett. 87, 214501.

Mydlarski, L. and Warhaft, Z. (1996) On the onset of high-Reynolds-number grid-generated wind tunnel turbulence. J. Fluid Mech. 320: 331-368.

Ouellette, N. T., Xu, H. & Bodenschatz, E. 2006 A quantitative study of three-dimensional Lagrangian particle tracking algorithms. Exps Fluids 40, 301–313.

Pope, S. B. (2000) Turbulent Flows. Cambridge University Press.

Robinson, S. K. (1991) Coherent motions in the turbulent boundary layer. Annu. Rev. Fluid Mech. (1991) 23: 601-639

Saddoughi, S. G. and Veeravalli, S. V. (1994) Local isotropy in turbulent boundary layers at high Reynolds number. J. Fluid Mech. 268: 333-372.

Sharp, N. S., Neuscamman, S., and Warhaft, Z. (2009) Effects of large-scale free stream turbulence on a boundary layer. Phys. of Fluids (submitted)

Spalart, P. R., (1988) Direct simulation of a turbulent boundary layer up to $R_\theta = 1410$ J. Fluid Mech 187:61-98

Thole, K. A. and Bogard, D. G. (1996) High freestream turbulence effects on turbulent boundary layers. J. Fluids Eng. 118:276-284

Tsai, R. Y. (1987) A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. IEEE J. of Robotics and Automation RA-3(4): 323-344

Veenman C. J., Reinders, M. J. T., Backer, E. (2001) Resolving motion correspondence for densely moving points. IEEE Trans Pattern Anal Mach Intell 23: 54-72

Voth, G. A., La Porta, A., Crawford, A. M., Alexander, J. & Bodenschatz, E. 2002 Measurement of particle accelerations in fully developed turbulence. J. Fluid Mech. 469, 121.

Yoon, K. and Warhaft, Z. (1990) The evolution of grid-generated turbulence under conditions of stable thermal stratification. J. Fluid Mech. 215: 601-638