Jet Flow Validation of Positron Emission Particle Tracking Utilizing High Speed Video

Seth Thomas Langford
University of Tennessee - Knoxville, slangfo1@vols.utk.edu

Recommended Citation

This Thesis is brought to you for free and open access by the Graduate School at Trace: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Masters Theses by an authorized administrator of Trace: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.
To the Graduate Council:

I am submitting herewith a thesis written by Seth Thomas Langford entitled "Jet Flow Validation of Positron Emission Particle Tracking Utilizing High Speed Video." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Nuclear Engineering.

Arthur E. Ruggles, Major Professor

We have read this thesis and recommend its acceptance:

Lawrence H. Heilbronn, John Auixer IV

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)
Jet Flow Validation of Positron Emission Particle Tracking Utilizing High Speed Video

A Thesis Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Seth Thomas Langford
May 2016
Acknowledgments

The author would like to thank the US Department of Energy (DOE) National Nuclear Security Administration (NNSA) and the Stockpile Stewardship Academic Alliance (SSAA) Program as well the Institute of Nuclear Power Operations and the National Academy for Nuclear Training Fellowship for funding me and this research.

The author would also like to thank Dr. Arthur Ruggles for guiding this research. Additionally, the author would like to thank the advising committee for their constructive suggestions and feedback.

This work was funded by the US DOE NNSA through the SSAA program at the University of Tennessee, Knoxville (UTK) under contract DE-NA-0001983. The views expressed here are those of the author and do not necessarily reflect those of the US DOE, the NNSA, or the SSAA program.
Abstract

Positron Emission Particle Tracking (PEPT) generates 4D Lagrangian particle trajectories and is used to evaluate flow in granular media and complex geometries where optical interrogation methods are not possible. A Multi-Particle PEPT (Multi-PEPT) approach was developed by the University of Tennessee Thermal Fluids Group capable of finding and tracking many particles simultaneously to extend the utility of the PEPT method. This thesis compares 4,014 trajectories generated using the Multi-PEPT method with 3,055 trajectories generated from High Speed Video (HSV) data. All trajectories are acquired in an acrylic test section with water flow using resin beads. The flow geometry includes a flow restriction producing a jet of Reynolds number 23,500 ± 800, with mean velocity 1.08 ± 0.04 m/s, and two recirculation zones. Variation between measurement outcomes is generally less than 0.1 m/s, and measured variations fall within validation uncertainties. Data co-location uncertainty contributes most to variation between Multi-PEPT and HSV velocities in regions of steep velocity gradients.
# Table of Contents

1  Introduction 1  
2  PET Scanner 3  
3  Test Section Design 5  
4  Fluid Flow Delivery System 7  
5  Instrumentation and Calibrations 10  
5.1  Absolute Pressure Transducers Calibration 12  
5.2  Orifice Plate Calibration 12  
6  Flow Description 14  
7  High Speed Video Experiment 16  
7.1  HSV Data Handling 16  
7.2  HSV Pixel Calibration and Distortions 21  
7.2.1  Index of Refraction Distortion 21  
7.2.2  Camera and Lenses Based Arbitrations and Distortions 25  
8  Positron Emission Particle Tracking Experiment 28  
8.1  PEPT Data Handling 28  
9  Lagrangian to Eularian Conversion Software Suite (LECSS) 32  
9.1  LECSS Validation Utilizing Synthetic Data 32  
9.2  LECSS Experimental Configuration 33  
10  Results 35  
11  Discussion 46  
12  Conclusions 49  
13  Division of Effort 50  
References 51  
Appendices 54  
Appendix A - Test Section Engineering Drawings 55  
Appendix B - Experimental Procedure 61  
   Equipment 61  
   Testing Procedure 61  
   Day Before 61  
   Day of the Test 64  
   Day after the Test 67  
Appendix C - Instrumentation Calibration Certificates 68
| Appendix D - Particle Activation Procedure | 72 |
| Appendix E - PEPT LECSS          | 74 |
| Appendix F - HSV LECSS          | 104 |
| Vita                                      | 128 |
List of Figures

Figure 1 - Siemens Inveon (Siemens, 2014) ................................................................. 4
Figure 2 - Inveon Detector Rings with Example Detected LORs in Blue (Siemens, 2014) .... 4
Figure 3 - Test Section Engineering Drawing ........................................................................ 6
Figure 4 - FFDS Geometry A (Dimensions in cm) ................................................................. 8
Figure 5 - FFDS Geometry B (Dimensions in cm) ................................................................. 8
Figure 6 - FFDS Geometry C (Dimensions in cm) ................................................................. 9
Figure 7 - FFDS Geometry D (Dimensions in cm) ................................................................. 9
Figure 8 - Experimental Flow Image, Flow is From Left to Right ........................................ 15
Figure 9 - HSV Still .............................................................................................................. 17
Figure 10 - HSV Flow rate, 25 hz sample rate, 0.62±0.02 L/s ................................................ 18
Figure 11 - HSV Inlet Absolute Pressure, 25 hz sample rate 112.03 kPa ± 0.48 kPa .......... 19
Figure 12 - Mosaic Settings ................................................................................................. 20
Figure 13 - A Sample of Mosaic Trajectories Prior to Filtering ........................................... 22
Figure 14 - HSV Geometry and Refraction Visualization ..................................................... 22
Figure 15 - Refraction Offset as a Function of View Angle ................................................... 26
Figure 16 - Refraction offset as a Function of Distance From Focal Point ............................ 27
Figure 17 - PEPT Flow Rate, 25 hz sample rate, 0.63±0.02 L/s ........................................... 29
Figure 18 - PEPT Inlet Absolute Pressure, 25 hz sample rate, $\sigma = \pm 0.36$ kPa .................. 30
Figure 19 - PEPT Outlet Absolute Pressure, 25 hz sample rate, $\sigma = \pm 0.32$ kPa ............. 31
Figure 20 - Power Law, Synthetic Data vs LECS Reconstruction ....................................... 34
Figure 21 - PEPT and HSV Co-location ............................................................................. 34
Figure 22 - PEPT Trajectories ............................................................................................ 36
Figure 23 - HSV Trajectories ............................................................................................... 36
Figure 24 - PEPT Arrow Plot .............................................................................................. 37
Figure 25 - HSV Arrow Plot ............................................................................................... 37
Figure 26 - PEPT X-Velocity .............................................................................................. 38
Figure 27 - HSV X-Velocity ............................................................................................... 38
Figure 28 - PEPT vs HSV X-Velocity Variation .................................................................... 39
Figure 29 - PEPT Y-Velocity Component .......................................................................... 39
Figure 30 - HSV Y-Velocity Component ............................................................................ 40
Figure 31 - PEPT vs HSV Y-Velocity Variation ................................................................... 40
Figure 32 - Probability Density Function for X-Velocity at X= 54 mm and Y =6 mm, Bin Width 
=0.05 m/s ......................................................................................................................... 41
Figure 33 - Probability Density Function for X-Velocity at X= 54 mm and Y =19 mm, Bin Width 
=0.05 m/s ......................................................................................................................... 41
Figure 34 - Integral Average X-Velocity Profile for PEPT and HSV .................................... 42
Figure 35 - 3D PEPT Trajectories Colored by Velocity, Top View ....................................... 43
Figure 36 - 3D PEPT Trajectories Colored by Velocity, Side View ..................................... 44
Figure 37 - 3D PEPT Trajectories Colored by Velocity, Angled View ................................ 45
Figure 38 - Test Section Design Drawing A ...................................................................... 55
Figure 39 - Test Section Design Drawing B ................................................................. 56
Figure 40 - Test Section Design Drawing C ................................................................. 57
Figure 41 - Test Section Design Drawing D ................................................................. 58
Figure 42 - Test Section Design Drawing E ................................................................. 59
Figure 43 - Test Section Design Drawing F ................................................................. 60
Figure 44 - Absolute Pressure Transducer 444462 Calibration .................................... 68
Figure 45 - Absolute Pressure Transducer 427575 Calibration .................................... 69
Figure 46 - Differential Pressure Transducer 441686 Calibration .................................... 70
Figure 47 - Differential Pressure to Flow Rate Correlation (Dwyer, 2009) ..................... 71
Nomenclature

keV Kilo electron volts
RPM Revolutions per minute
W Watts
Hz Hertz
PSI Pounds per square inch
P Absolute pressure (PSI)
dP Differential pressure (PSI)
d Orifice plate bore
D Pipe inside diameter
K Flow coefficient
Y Expansion factor
Fa Thermal expansion factor
h/w Differential pressure (inches water column)
P L Density at line flowing conditions $\frac{lbs}{ft^3}$
R n Reynolds number
C Flow constant
$\beta$ Beta Ratio ($\frac{d}{D}$)
GPM Flow rate (gallons per minute)
FPS Frames per second
Ni Index of refraction
$\Theta_i$ Refraction angle
D i Distance from normal
T i Thickness
$\Delta$ Distance between where camera detects light and where it originates
$\sigma$ Standard deviation
NaN Not a number
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V$</td>
<td>Synthetic trajectory velocity</td>
</tr>
<tr>
<td>$V_{\text{max}}$</td>
<td>Synthetic trajectory peak velocity</td>
</tr>
<tr>
<td>$r$</td>
<td>Synthetic trajectory radial location</td>
</tr>
<tr>
<td>$R$</td>
<td>Synthetic trajectory pipe radius</td>
</tr>
<tr>
<td>$\delta V$</td>
<td>Uncertainty in PEPT and PTV velocity</td>
</tr>
</tbody>
</table>
1 Introduction

Positron Emission Tomography (PET) produces three-dimensional renderings of activity distribution and time resolved (4D) distributions of activity in clinical settings. A tomograph returns an activity value for every voxel in the field of view (FOV) of a scanner, and image resolution is limited to a few millimeters due to positron range and non-collinearity of the opposed 511 keV gamma-rays created when the positron annihilates (Moses, 2011). Positron Emission Particle Tracking (PEPT) (Parker, 1993), tracks an activated particle as it moves in the image volume of the scanner. The count sequence measured by the scanner is broken into time segments, and the particle is located in each time segment using the lines of response (LORs) created by the twin gamma-rays from positron annihilations near the particle position. A recent example from a research group in Norway (Chang, 2015) uses a clinical Siemens TruePoint scanner to locate a single activated particle to 100 microseconds every millisecond in a hydrocyclone flow with velocities near 10 m/s. Another recent application of PEPT studies water motion in a dishwasher (Perez-Mohedano, 2014). A group at Stanford University simulated single cell tracking using PEPT (Lee, 2015). A cooperation between UC Davis and the University of Birmingham yielded a study on mixing inside of a static mixer (Mihailova, 2015).

The PEPT method was originally developed around tracking of a single particle trace in the field of view of the scanner. This method was extended to follow multiple particles which were of very different activity (Yang, 2006). Our group developed a method to locate and track a random number of particles in the scanner’s field of view to extend the utility of the method for fluid flow measurement. This method of Multiple Positron Emission Particle Tracking (Multi-PEPT) has been shown to locate 17 particles arbitrarily located and articulated in the bore of a Concord Microsystems P4 preclinical PET scanner (Wiggins, 2016). The Multi-PEPT method can also distinguish two particles when separated by more than 5 mm in the P4 scanner bore.

The position accuracy of a measurement has been investigated for the PEPT method and the position uncertainty goes as one over the number of true coincident counts from the particle to power one half (Bickell, 2012). While this theoretical limit is well accepted, there are several other parameters that influence the position uncertainty, such as the scanner sensitivity gradients, the scanner detector crystal size and crystal position location uncertainties, and the scintillation detection and reporting uncertainties. Particle position uncertainties contribute to noise in velocity when position derivatives are used to establish particle velocity. Smoothing approaches have been developed to address this. Lee et al. (Lee, 2015) developed an approach linking cubic splines to create smooth trajectories for low activity particles and tested the fidelity of this approach using data generated from a GATE (Jan, 2004) simulation of the scanner response. Our group reported performance of the Multi-PEPT algorithm in tracking particle trajectories using smoothing functions for data generated using a particle mounted on an electrodynamic shaker (Wiggins, 2016). Chang and Hoffman (Chang, 2015) also used smoothing approaches to condition their data.
In this thesis, Multi-PEPT is validated for fluid flow measurements. This work is a continuation of earlier work to validate Multi-PEPT for a channel flow (Langford, 2016). We compare high-speed video (HSV) data of particle trajectories with PEPT trajectories for turbulent flow featuring a jet region and recirculation zones. Data are collected for both measurement approaches using identical tracer particles and flow matched conditions. This offers an end-to-end comparison of an established optical flow measurement technique with our Multi-PEPT method.
2 PET Scanner

The PEPT measurements are performed using a Siemens Inveon preclinical PET scanner (Bao, 2009). In this scanner, 25,600 lutetium oxyorthosilicate (LSO) detector crystals are in a cylindrical array of diameter 161 mm and axial extent 127 mm, having a useable bore of 117 mm diameter. This scanner can collect up to 1.6 million LORs per second, and provides timing resolution down to 200 microseconds. Figure 1 depicts the PET scanner that is used to collect PEPT data. Figure 2 illustrates the Inveon’s detector rings and coincidently detected LORs. The widespread use of PET scanners at medical centers as well as the availability of positron emitting radio-isotopes at these centers facilitates PEPT measurements.
Figure 1 - Siemens Inveon (Siemens, 2014)

Figure 2 - Inveon Detector Rings with Example Detected LORs in Blue (Siemens, 2014)
3 Test Section Design

The test section was engineered by the author and then contracted out for construction. In order to satisfy the requirements that the test section be capable of being interrogated by optical and PEPT measurements, the test section was constructed of clear acrylic. The test section is designed to have exterior dimensions of 116.8 cm long, 8.9 cm wide and 5.2 cm tall when the lid is installed in order to fit into the Inveon. The internal flow dimensions are 111.4 cm long 4.1 cm wide and 3.8 cm tall. The test section features ¼-20 UNC threaded holes along the interior walls to permit the installation of flow obstructions. The lid seals to the test section body using an O-ring groove designed for 1/8 inch O-ring cord stock in a static face seal configuration. Design drawings of the test section can be found in Appendix A. A sketch of the test section is shown in Figure 3 including the location of the flow obstruction blocks used for these studies.
Figure 3 - Test Section Engineering Drawing
4 Fluid Flow Delivery System

A fluid flow delivery system (FFDS) was constructed from ¾” schedule 40 PVC pipe to supply water to the test section. The FFDS is interspersed with unions allowing for parts of the system to be interchanged. This permits rapid exchange of instrumentation, test sections and flow modulating devices. The flow is driven by a 3000 RPM 150W Little Giant Pump and features a tank to supply water to the loop, recirculation lines as well as a particle insertion/vent line. Images of the FFDS as it was utilized for the PEPT experiment are shown in Figure 4 through Figure 7 with dimensions given in units of cm.
Figure 4 - FFDS Geometry A (Dimensions in cm)

Figure 5 - FFDS Geometry B (Dimensions in cm)
Figure 6 - FFDS Geometry C (Dimensions in cm)

Figure 7 - FFDS Geometry D (Dimensions in cm)
5 Instrumentation and Calibrations

The Fluid Flow Delivery System (FFDS) was instrumented utilizing pressure transducers to provide absolute pressure measurements and a differential pressure transducer across an orifice plate to provide volumetric flow rate.

Table 1 summarizes the instrumentation used in the FFDS along with vendor supplied uncertainties. It is important to note that for the case of the orifice plate, the flow rate is calculated by correlating the differential pressure across the orifice plate to the flow rate. The pressure signal included noise from a range of sources including pump vane pass pressure pulses, electrical noise and loop flow induced vibration. The uncertainty of the flow rate measurement is a function of the uncertainty of every instrument involved in the data acquisition chain used to infer the flow rate. The pressures are sampled at 25 Hz. Parameters such as flow rate and pressure are henceforth described by the signal’s mean value ± standard deviation. Appendix B details the experimental procedure used to gather data from the FFDS during the PEPT experiment as well as the HSV experiment.
Table 1, FFDS Instrumentation

<table>
<thead>
<tr>
<th>Device Type</th>
<th>Vendor</th>
<th>Model Number</th>
<th>Quantity</th>
<th>Vendor Supplied Uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute Pressure Transducers</td>
<td>Omega</td>
<td>PX409-150A5V</td>
<td>2</td>
<td>± 0.08%</td>
</tr>
<tr>
<td>Differential Pressure Transducer</td>
<td>Omega</td>
<td>PX409-005DWU5V</td>
<td>1</td>
<td>± 0.08%</td>
</tr>
<tr>
<td>Orifice Plate</td>
<td>Dwyer</td>
<td>PE-B-3</td>
<td>1</td>
<td>± 0.7%</td>
</tr>
<tr>
<td>Data Acquisition Module</td>
<td>National Instruments</td>
<td>9215</td>
<td>1</td>
<td>± 0.7%</td>
</tr>
<tr>
<td>Data Acquisition Software</td>
<td>National Instruments</td>
<td>LabVIEW 2014</td>
<td>1</td>
<td>N/A</td>
</tr>
</tbody>
</table>
5.1 Absolute Pressure Transducers Calibration

All pressure and flow measurements were originally acquired in English units, which are consistent with the calibrations provided by the vendors using the following fit equations. Reported values are converted from English to SI units. Absolute pressure transducer model number PX409-150A5V, serial number 444462 came with the calibration certification shown in Appendix C, Figure 44. A straight line was fitted through the ordered pairs (-0.002, 0.00), (2.500, 75.00) and (5.001, 150.00) to form the fit equation 1. Where $P$ is pressure in units of PSI and $V$ is the signal voltage in volts.

$$P = 29.982V + 0.055$$

Similarly, the absolute pressure transducer model number PX409-150A5V, serial number 427575 has certificate shown in Appendix C, Figure 45. A calibration curve with a fit equation is given in equation 2, which is used in LabVIEW to report pressure during the experiments.

$$P = 30.048V - 0.2003$$

The differential pressure transducer with model number PX409-005DWU5V and serial number 441686 also has certification shown in Appendix C, Figure 46. This sensor is implemented into LabVIEW using the linear equation 3 to convert the voltage signal into differential pressure. Where $V$ is the signal voltage and $dP$ is differential pressure in units of PSI.

$$dP = 0.9994V - 0.0028$$

5.2 Orifice Plate Calibration

The Dwyer PE-B-3 orifice plate uses calibration equations recommended by the vendor shown in Appendix C, Figure 47 to correlate the differential pressure across the orifice plate to flow rate in gallons per minute (GPM). The flowrate is first evaluated as a dependent function of the independent variable $h/w$ (inches of water column), at Reynolds number 42,000. Reynolds number 42,000 corresponds to the intended test flow rate of 10 GPM and ¾ inch hydraulic diameter of the pipe leading to the orifice plate. The parameters used from Appendix C, Figure 47 include the Beta ratio for the instrument, 0.7, provided by the vendor and the nominal pipe size upstream of the orifice plate of ¾ inch. As advised by the vendor for water,
the expansion factor and thermal expansion factor are taken to be unity. The density is taken to be 62.3 lbs/ft^3.

Once the function is evaluated with pressure in terms of water column, a sixth order polynomial is developed describing flow rates between 5 and 12 GPM as a function of differential pressure in units of PSI. The polynomial in equation 4 is implemented in LabVIEW with $dP$ as differential pressure in PSI, and the flow rate given in GPM. Pressure units of PSI are used for all instruments in the experiment for consistency.

$$GPM = -0.0161dP^6 + 0.1971dP^5 - 1.0109dP^4 + 2.8687dP^3 - 5.1647dP^2 + 8.4097dP + 1.7834$$
6 Flow Description

The test section is fitted with four rectangular baffle plates to form a jet flow with recirculation regions as seen in Figure 8. The first two baffle plates are placed 18 hydraulic diameters from the inlet. These restrict the flow area by a factor of 2.5. The open area between the first two baffle plates and second two baffle plates create two recirculation zones. Particles may reside in these regions and recirculate until they get entrained by the jet flow. A second region of recirculating flows exists downstream of the second set of baffles. The Reynolds number in the constricted flow region is $23,500 \pm 800$ based on a hydraulic diameter of $2.191 \pm 0.016$ cm and a mean flow velocity of $1.07 \pm 0.03$ m/s. An image of the test channel with baffles is offered in Figure 8. Flow behavior is highlighted here by bubbles in the flow. Bubbles are removed prior to formal testing.
Figure 8 - Experimental Flow Image, Flow is From Left to Right
7 **High Speed Video Experiment**

This thesis validates PEPT’s flow measurement capability using high-speed video (HSV) particle tracking velocimetry (PTV). This section describes the HSV PTV experiment and measurements. An average flow rate of 0.62±0.02 L/s (9.8 ±0.3 GPM) is achieved in the FFDS during these studies. Non-activated particles of the same kind used in the PEPT experiment are used to seed the flow. Four seconds of high speed video was acquired with an Olympus i-SPEED 2 high speed video camera at 1000 FPS at 5X shutter speed with a Fujinon 1:1.4/25mm cf25ha-1 lens. The camera is placed approximately 0.5 m above the test section. A still from the HSV is shown in Figure 9. The camera was borrowed from Oak Ridge National Laboratory (ORNL) and the HSV was acquired with assistance from David Felde, a staff researcher at ORNL.

Figure 10 shows the flow rate signal with associated noise at a sample rate 25 Hz. Figure 11, shows the inlet pressure going into the test section. The outlet pressure transducer was malfunctioning during this experiment, and was later replaced.

7.1 **HSV Data Handling**

High-speed video is imported to ImageJ (Rasband, 2015) in .avi format. The video is cropped to fit the walls of the test section as well as to include only the first 20 mm before and after the baffle plates. The Mosaic plugin (Sbalzarini, 2005) for ImageJ is used to track the particles as they travel through the flow. Mosaic identifies particles across a series of image frames using multiple parameters. These include the brightness of the particle in relationship to its environment, proximity of a particle size to a user input size, and a cutoff score relating particle size and brightness to the average of all other particles detected in a given frame.

Linking of particles between frames is controlled with a user-input number of frames (or “range”) to consider for candidate links, a maximum particle displacement between frames, and the particle behavior classification. Brownian, straight or constant velocity dynamics options are available for particle behavior. The particle pixel radius is set to 3, the cutoff score is set to 3, the linking range is left at the default setting of 2, the maximum displacement is set to 12 pixels and the particle dynamic classification is set to straight for these measurements, as seen in Figure 12.

The nearly 4000 frames processed by ImageJ for Mosaic yield 12,723 trajectories. These are visually checked for accuracy to ensure that the detection and linking of particles is accurate. It is found that particles were falsely detected on top of the baffle plates as seen in Figure 13. Incorrect linking also occurs, resulting in what appears to be large spikes in velocity. Trajectory exclusion criteria were developed to resolve this. A trajectory is excluded if the change in velocity between two serial points in a trajectory changes by more than 200% within one time step. If a trajectory has a total length shorter than 5.0 mm, it is also excluded to remove stationary particles from the data field. At the end of filtering the original 12,723 trajectories are reduced to 3,055 trajectories. Figure 13 indicates a sample of PTV trajectories prior to filtering.
Figure 9 - HSV Still
Figure 10 - HSV Flow rate, 25 hz sample rate, 0.62±0.02 L/s
Figure 11 - HSV Inlet Absolute Pressure, 25 hz sample rate 112.03 kPa ± 0.48 kPa
**Figure 12 - Mosaic Settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius</td>
<td>3</td>
</tr>
<tr>
<td>Cutoff</td>
<td>3.0</td>
</tr>
<tr>
<td>Per/Abs</td>
<td>0.70000</td>
</tr>
<tr>
<td>Absolute</td>
<td></td>
</tr>
<tr>
<td>Convert to Gray8</td>
<td>(recommended)</td>
</tr>
</tbody>
</table>

**Particle Linking:**

- **Link Range**: 2
- **Displacement**: 12
- **Dynamics**: straight

Advanced options

---

Please refer to and cite:

7.2 HSV Pixel Calibration and Distortions

In order to convert the trajectories generated by Mosaic into a description of the flow field, it is necessary to convert the image pixels to a description which can be directly compared to other measured flow trajectories. A calibration is created for the whole image field by first measuring the distance between the top of the two left most baffle plates in units of pixels, 85 ±4 pixels, and then dividing that number by the physical distance as measured by a micrometer, 15.4 ± 0.1 mm, to yield a calibration value of 5.5 ± 0.3 pixels/mm.

7.2.1 Index of Refraction Distortion

The HSV image calibration has uncertainty that varies across the field of view. One source of uncertainty is refraction, a phenomenon described by Snell’s law (Peatross, 2015). According to Snell’s law, the index of refraction, $N$, and incoming angle of incident light, $\theta$, in the incident medium is proportional to the index of refraction and the angle of refraction in the refracting medium, as given by equation 5.

$$N_i \sin(\theta_i) = N_{i-1}\sin(\theta_{i-1})$$  \hspace{1cm} 5

Where subscript $i$ denotes the medium, with $i$ indexing across the plane of contact. The specific case of the test section is shown in Figure 14. The figure illustrates the difference between the true location of a particle and where the camera lens places the reflected light from the particle in the image plane. Light must travel from the particle to the camera as demonstrated Figure 14. The largest distortion occurs in the acrylic and water as a result of the camera receiving light that has been refracted by the air. The focal point of the video is 9.0 mm to the right of the right side of the lower left baffle, and 0.5 mm down from the upper right corner of the lower left baffle. Regions that are further away from this focal point have a greater angle $\theta_3$ and thus suffer from a larger distortion from refraction.
Figure 13 - A Sample of Mosaic Trajectories Prior to Filtering

Figure 14 - HSV Geometry and Refraction Visualization
Equation 6 may be evaluated to determine where a particle appears in the camera image. Where $D_{\text{Apparent}}$ is where the apparent position, and $T_i$ is the thickness of the medium the light is passing through as defined in Figure 14.

$$D_{\text{Apparent}} = (T_1 + T_2 + T_3)Tan(\theta_3)$$  

6

The distance between $D_{\text{Apparent}}$, and the actual position $D_{\text{Actual}}$ is defined as delta, $\Delta$, the refraction offset, is given by equation 7.

$$\Delta = D_{\text{Apparent}} - D_{\text{Actual}}$$  

7

The distance that a particle is actually away from the camera’s normal, $D_{\text{Actual}}$, is given by equation 8. Where $D_i$ is the distance from the normal in each medium.

$$D_{\text{Actual}} = D_1 + D_2 + D_3$$  

8

$D_i$ is given by equation 9.

$$D_i = T_i Tan(\theta_i)$$  

9

$\theta_{i-1}$ is given by equation 10.

$$\theta_{i-1} = \sin^{-1}(\sin(\theta_i) \frac{N_i}{N_{i-1}})$$  

10

Where $i$ may vary from 1 to 3 to populate the $D_i$ in equation 8. Using equations 9 and 10, it is possible to solve for all of the distances $D_{1-3}$ and thus $D_{\text{Actual}}$ as well as $\Delta$ as a function of either $\theta_3$ or distance away from the camera’s focal point. The parameters used in equations 6-10 are summarized in Table 2.
**Table 2 - HSV Geometric and Refraction Parameters**

<table>
<thead>
<tr>
<th>Material</th>
<th>Thickness, T (mm)</th>
<th>Index of Refraction, N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air</td>
<td>500.0 = T₃</td>
<td>1.00 (Stone, 2011)</td>
</tr>
<tr>
<td>Acrylic</td>
<td>12.3 = T₂</td>
<td>1.33 (Kasarova, 2007)</td>
</tr>
<tr>
<td>Water</td>
<td>38.4 = T₁</td>
<td>1.49 (Schiebner, 1990)</td>
</tr>
</tbody>
</table>
The offset as a function of view angle $\theta_3$, and its corresponding distance away from the focal point are shown in Figure 15 and Figure 16. The high speed video imaged the whole depth of the water cross section, $T_3$ equal 38.4 mm. The amount of refraction based optical distortion that a particle exhibits varies with its depth in the test section, $T_1$.

7.2.2 Camera and Lenses Based Arbitrations and Distortions

Distortions other than index of refraction also exist (Soloff, 1997). One affect that can cause uncertainties in the measurement technique is focusing arbitrations which may cause the size of the particles to be larger than they should be. The depth of water in the test section may exceed the depth of field causing some particles to be out of focus such that particles will vary in size through the depth of the test section. A second kind of arbitration causes the worth of a pixel to vary throughout the image field. Soloff explains that this second kind of arbitration can be caused by lens imperfections, refraction as described earlier, and finally misalignment with the image. Soloff, goes on to mention that one must relate the displacement measured by the camera to a corresponding displacement in the experimental frame of reference. However, since the camera measures in two dimensions, while the particle travels in three dimensions, it can be difficult to properly assign a pixel to physical space conversion factor that is valid for all depths and locations. Soloff, states that this difficulty exists even without other distortions present as a result of perspective. A solution that would work if only a single image plane was used is an experimental calibration of the image field where each region of the flow gets assigned its own pixel to distance conversion factor.
Figure 15 - Refraction Offset as a Function of View Angle
Figure 16 - Refraction offset as a Function of Distance From Focal Point
8 Positron Emission Particle Tracking Experiment

Particles tracked in this experiment are anion exchange resin beads of 560-700 micron diameter with a density of 1.255±0.005 g/cm³. For this experiment ¹⁸F (half-life of 109.8 minutes) is produced using a Siemens Eclipse cyclotron at the University of Tennessee Medical Center, via the bombardment of ¹⁸O-enriched water with an 11 MeV proton beam. Roughly 20 of our tracer particles are soaked in an aqueous solution containing 30 mCi of ¹⁸F. Each particle is activated to around 1 mCi, resulting in a total sample activity of 20 mCi. Particles are introduced to the flow loop via the particle injection port shown in Figure 4, and a scan is performed for 30 minutes. As seen in Figure 17 the mean flow rate during the experiment is 0.63±0.02 L/s (10.0 ±0.3 GPM). The scanner is set to accept coincident gamma rays in the 425-625 keV range. Using this energy window, each 1 mCi particle results in roughly 450,000 counts per second (cps) near the center of the FOV and roughly 100,000 cps near the axial edge. Figure 18 and Figure 19 indicate the inlet and outlet pressure in the test section as seen during the PEPT experiment. The drift in inlet and outlet pressure is attributable to an increase in fluid temperature resulting from running the pump over a prolonged period of time. The particle activation procedure approved by radiation safety is in Appendix D.

8.1 PEPT Data Handling

Using list mode data generated by the scanner, our PEPT algorithm is used to reconstruct the trajectories of the particles. In this reconstruction, data is sliced into 1 millisecond frames, resulting in roughly 450 LOR per particle per frame for particles near the center of the FOV. LOR crossings are tallied over an 8 mm³ cube grid, and an Anderson-Darling critical value of 20 is used in our clustering routine (Wiggins, 2016). The average calculated position uncertainty is found to be 0.23 mm in the radial direction and 0.17 mm in the axial direction.

Over the course of the 30 minute scan, a total of 16,887 trajectories are detected by our algorithm. However, upon analysis of these, it is determined that 3 particles became stuck in the FOV of the scanner throughout this scan. This results in a large number of spurious and corrupted trajectories, not reflecting the true fluid flow of the system in proximity to the stuck particles. A number of techniques are used to remove such trajectories. Smoothing of retained trajectories is also performed consistent with approaches used in other PEPT research (Wiggins 2016).

First, in order to remove trajectories related to stuck particles, trajectories which had a total position standard deviation of less than 5 mm are not considered. Secondly, it is found that some trajectories exhibit a great deal of erraticism as they pass near the stuck particles. This is likely due to our clustering algorithm accepting a cluster that should be split, resulting in a false detection site in between two true particles. Such trajectories are removed by rejecting any that exhibit single frame displacements above 2.5 mm. The remaining trajectories are smoothed via a non-weighted moving average filter of size 5. A total of 4,014 smoothed particle trajectories remain after filtering and are used herein.
Figure 17 - PEPT Flow Rate, 25 hz sample rate, 0.63±0.02 L/s
Figure 18 - PEPT Inlet Absolute Pressure, 25 hz sample rate, $\sigma = \pm 0.36$ kP
Figure 19 - PEPT Outlet Absolute Pressure, 25 hz sample rate, $\sigma = \pm 0.32$ kPa
9  Lagrangian to Eularian Conversion Software Suite (LECSS)

A MATLAB 2015A script is written to convert both the PEPT and high-speed video trajectories from their natural Lagrangian frames to a Eularian frame as seen in Appendices E and F. This code will henceforth be referred to as the Lagrangian to Eulerian Conversion Software Suite (LECSS). First, the trajectory files containing information about the position, frame number, and particle identifier for each trajectory are read into LECSS. Next, LECSS calculates velocity components for the trajectories utilizing the forward divided difference method. Using gridding parameters chosen by the user, a two-dimensional, time-dependent grid is constructed over the region of interest. After the grid is created, the imported trajectories are traced over this grid. Velocities at each grid element are taken to be the averages of the instantaneous velocities of all the trajectories passing through that element. If no trajectories pass through a particular grid space, it is assigned a velocity of “Not a Number” (NaN). Additionally, if the number of trajectories passing through a grid element is found to be smaller than a user-defined cutoff, the grid space can also be ignored and set to NaN to flag that the statistics of that grid space are weak.

9.1  LECSS Validation Utilizing Synthetic Data

Since LECSS is an original code written for this thesis effort and is used for the basis of the PEPT to HSV comparison, it was deemed necessary to provide a simple validation for LECSS. A set of 10,000 synthetic trajectories were created with velocities dictated on a randomly seeded radial position and the 1/7th power law (Johnson, 1970). This velocity profile is typically used as an approximation for fully developed turbulent pipe flow with Reynolds number near 10^5. The 1/7th power law profile is given as equation 11.

\[ V = V_{max} \left(1 - \frac{r}{R}\right)^{1/7} \]  

Where, \( V \) is the velocity of the trajectory in m/s, \( V_{max} \) is the peak velocity used, 2 m/s. \( R \) is the radius of the pipe, 1 meter, and \( r \) is the randomly seeded trajectory radial position in meters.

Once the radial position and velocity of the trajectories are generated, they are initialized to an axial position of 0.0 meters and allowed to travel until they reached an axial position of 1 meter. The trajectories are then output in a series of text files with the same format as the text files generated by MPEPT and are then read into LECSS. 1000 radial grids and 4 axial grids are used resulting in a 2.0 mm by 0.25m grids. The velocity profile from the synthetic data is plotted...
against a 1000 grid slice from LECSS as well as an analytical solution of the power law as seen in Figure 20.

The synthetic data reaches a lower velocity along the edges than the LECSS reconstruction because the most extreme radial grid is only 2 mm away from the wall. For $1/7$th power law to reach the no slip condition, gridding much finer than 2 mm would be necessary. Additionally, since only 10,000 synthetic trajectories were generated, it is unlikely that the very steep gradient near the wall would be resolved. The result of the LECSS reconstruction of the synthetic data indicates the program is functioning properly.

9.2 LECSS Experimental Configuration

For the case of the PEPT vs HSV validation experiment, square grids of dimension 1.3x1.3 mm are used to create a grid that is 86 units long in the x (flow) direction, and 30 units wide in the y (transverse flow) direction. Trajectories passing through a grid generates x and y velocity values for that grid element. Velocity values are summed and averaged for each grid element. To remove velocities resulting from stationary (i.e. stuck) particles, a minimum particle velocity magnitude of 0.05 m/s is set, below which, it is no longer counted in a grid average. A lower level discriminator of 10 populated occurrences per grid element is used to ignore grid elements with poor statistics. The PEPT version of the code is available in Appendix E and the HSV version is available in Appendix F.

Co-location of PEPT and HSV data is achieved using the forward and rear surfaces of the baffle plates. These locations are inferred from the PEPT data trajectories since the surfaces are not visible in PEPT data form. The result of the co-location process is checked using the HSV image location for the inside corners of the baffle plates. These corners are plotted on top of the PEPT trajectories in Figure 21. The co-location of HSV and PEPT data is not perfect and improvements on this process are planned.
Figure 20 - Power Law, Synthetic Data vs LECSS Reconstruction

Figure 21 - PEPT and HSV Co-location
10 Results

PEPT trajectories are shown in Figure 22. In Figure 22 trajectories are seen to have traveled over the baffle plates or passed between the lid and the walls of the test section. This behavior is also observed visually and is facilitated by the small gap between these parts of the flow channel components. The HSV trajectories are shown in Figure 23. Trajectories in the gap between the lid and test section are not seen in Figure 23 because the video is cropped at the walls.

The spatiotemporal grid averaging process produces Eulerian flow data represented as arrow plots in Figure 24 (PEPT) and Figure 25 (HSV). Recirculation zones are clearly seen in the region between baffle plates and part of a recirculation zone is apparent downstream of the second set of baffles.

Further comparison is performed using color-maps of the x (flow direction) and y (transverse flow direction) components of the velocity as measured by PEPT and HSV. X-component velocity color-maps are seen in Figure 26 and Figure 27.

The absolute x-velocity variation between the two measurement techniques is offered in a color plot in Figure 28. There is less than 0.1 m/s variation across the majority of the FOV, with relatively larger variation in regions of large velocity gradient. For visualization purposes, the extent of the color palate in Figure 28 is limited to 0.25 m/s. Grids which surpass this are shown in dark red. The peak grid value in Figure 28 is 1.18 m/s and is located on the upper left corner of the lower left baffle plate.

Figure 29 and Figure 30 show color-maps of the y-component of the velocity as measured by PEPT and HSV, respectively. Figure 31 shows the grid-by-grid differences between the PEPT and HSV measurements of the y-component of velocity. This plot shows variation in the y-velocity component are less than 0.05 m/s over the bulk of the FOV, with an increase in variation along the edges of the baffle plates. For visualization purposes, the extent of the color palate in Figure 31 is limited to 0.1 m/s. Grids which surpass this are shown in dark red. The peak grid value in Figure 31 is 0.67 m/s and is located on the lower left corner of the upper left baffle plate.

Figure 32 and Figure 33 are probability density functions for x-velocity taken at x=54 mm and y=6 mm and y=19 mm respectively. The values that make up the grid average tend to be closer to the mean as the position moves toward the channel center. Higher variations about the mean are expected in the recirculation zones, and this contributes to a requirement for more trajectories in a grid location to compute a stable mean value. The probability density function is examined to expose differences in the HSV and PEPT data attributable to noise or data filtering.

A scalar velocity profile comparison is obtained by averaging across all grids in the x-direction at every height in the y-direction. This integral average comparison is shown in Figure 34.
Figure 22 - PEPT Trajectories

Figure 23 - HSV Trajectories
Figure 24 - PEPT Arrow Plot

Figure 25 - HSV Arrow Plot
Figure 26 - PEPT X-Velocity

Figure 27 - HSV X-Velocity
Figure 28 - PEPT vs HSV X-Velocity Variation

Figure 29 - PEPT Y-Velocity Component
Figure 30 - HSV Y-Velocity Component

Figure 31 - PEPT vs HSV Y-Velocity Variation
Figure 32 - Probability Density Function for X-Velocity at X = 54 mm and Y = 6 mm, Bin Width = 0.05 m/s

Figure 33 - Probability Density Function for X-Velocity at X = 54 mm and Y = 19 mm, Bin Width = 0.05 m/s
Figure 34 - Integral Average X-Velocity Profile for PEPT and HSV

Figure 35, Figure 36 and Figure 37 are PEPT Lagrangian trajectories as colored by velocity. The figures demonstrate the native format of the PEPT data prior to being converted into a 2D grid averaged Eulerian frame.

The process of going from positron electron annihilations to the images presented in this thesis involves a complex transformation of data. During the 30 minute scan used to generate the data for this thesis, a 15 GB binary .lst file was created. This .lst file is then read into the Multi-PEPT code (Wiggins, 2016) and after filtering of spurious trajectories a series of text files are generated indicating position as a function of time with a total file size on the order of 10 MB. After further processing images such as those presented in Figure 35 through Figure 37 are created with a file size on the order of 1 MB each. Using videos, such as those available on https://www.youtube.com/watch?v=ut3DZlslBD8, the entire flow field can be described in under 30 MB, a data format which is 500 times smaller than the original binary .lst file.
Figure 35 - 3D PEPT Trajectories Colored by Velocity, Top View
Flow Direction

Figure 36 - 3D PEPT Trajectories Colored by Velocity, Side View
Figure 37 - 3D PEPT Trajectories Colored by Velocity, Angled View
11 Discussion

High-speed PTV and PEPT are Lagrangian techniques that generate trajectories with an associated time history. The HSV data used in this comparison are 2D, while the PEPT data are 3D. This limits the dimensionality of the measurement comparison. Within the dimensionality limitation there is the additional challenge that Lagrangian data are inherently difficult to compare across measurement modalities without converting to the more familiar Eulerian format. The process of converting these data from Lagrangian to Eulerian frame causes loss of information and complicates the primary goal of uncertainty quantification.

Within these Eulerian comparisons offered herein, the co-location of data makes the largest contribution to uncertainty in the reported data variation. From Figure 8 it is determined that a co-location error of up to 1.8 mm exists. Considering this and the maxima of the gradients of the PEPT velocity components, the maximum theoretical error as a result of co-location is determined to be 0.64 m/s for x-velocity and 0.37 m/s for y-velocity. It is expected that the difference between the measurements will approach these values in regions where the gradients are the greatest. However, using the average velocity gradients across the FOV, we find average expected co-location velocity differences of 0.052 m/s in the x-velocity and 0.029 m/s in the y-velocity. While these are larger than we would like, this does allow the measured velocity variations to fall within our uncertainties. Improvements in co-location precision will facilitate a more demanding test of the PEPT measurement.

This thesis provides a top down approach to validating PEPT by directly comparing the outcomes of a PEPT experiment to a HSV experiment, another approach is a bottom up approach. A bottom up approach entails propagating uncertainty through both the PEPT and PTV measurements starting at data acquisition uncertainty, and working upward until the uncertainty in the individual Eulerian measurements are estimated. Uncertainty in HSV and PEPT measurements, along with uncertainties associated with the direct comparison, should allow outcomes of the two measurements to overlap. Another student is working through the individual contributions to PEPT measurement uncertainty, but for completeness, a discussion of the contributions to uncertainty in these measurements is offered here.

For PTV, there is uncertainty in the camera’s ability to place the particles in the correct location in the image frame. Refraction caused by the water and test section, and image magnification caused by the camera lens contribute to position uncertainty. Prior analysis in section 7.2.2 indicates these contributions are a function of depth in the test section as well as distance from the center of the lens focal point. The optical uncertainty also includes conversion from the camera’s coordinate system based on pixels to the physical test section in terms of mm. The actual conversion will depend on position in the camera field of view, as discussed in section 7.2.1, but a single representative value is used in this thesis determined as 5.5 ± 0.3 pixels/mm.

Once the particle image is recorded by the camera, Mosaic’s ability to correctly identify the centroid of the particle further contributes to uncertainty. The uncertainty in centroid location requires quantifying how well a particle centroid corresponds to the image centroid, and is a function of test section lighting, and blurring due to particle speed and shutter speed. Mosaic’s
settings allow a particle to be identified as a function of its brightness relative to the background, and the particle radius. These setting choices also influence the particle identification and centroid location to some degree. Inappropriate linking of particles sometimes results from particles passing nearby bright spots due to reflections from lighting, and from test section structure. Inappropriate linking was largely controlled by filtering of data based on trajectory length and acceleration limits as discussed in the next paragraph.

Particles moving with the flow, and passing through the camera field of view normally create trajectories several cm long. The first filter applied to the PTV data, removes trajectories shorter than 5 mm. This removes the stuck/stationary particles that Mosaic constantly identified in the field of view. This filter reduced the data set from 448,283 points down to 230,706. The next filter removes trajectories that have a fractional velocity change between frames greater than 200. With a frame collection every millisecond, this corresponds to an acceleration exclusion of near 200,000 m/s$^2$ in the main flow with mean velocity near 1 m/s. This filter reduced the data set from 230,706 points down to 187,794 points.

Finally, a statistical uncertainty exist in how well the mean velocity value is defined in each grid based on the number of trajectories penetrating each grid area, and including propagation of all the previously mentioned uncertainties. This process should yield a mean ± uncertainty value for each grid in the PTV field of view. Uncorrelated uncertainties will benefit from a larger number of collected trajectories. Equation 12 offers a conservative representation of the individual contributions to HSV particle tracking velocimetry measurements as discussed in this section. Where $\delta V$ is the uncertainty in velocity from each contributor.

$$\delta V_{PTV} = \delta V_{Optical} + \delta V_{Mosaic} + \delta V_{Filtering}$$  

The statistical uncertainty for computed average velocity in a local grid element is not included since it may not be additive.

PEPT particle location uncertainty goes as one over the number of true coincident counts from the particle to power ½ (Bickell, 2012). The number of counts received for any particle is a function of activity and position in the scanner. The absolute position uncertainty depends on the positron range and scanner resolution. The $^{18}$F positron range in water is near 0.6mm, and the Inveon resolution depends on position in the bore, but ranges near 1.5mm near the bore center. A time window of 1 ms may allow 200 true coincident counts (LORs) from a particle, typically allowing a location accuracy near 100 microns. Work is ongoing to better characterize these uncertainties for our scanner and particle tracking algorithm.

The PEPT algorithm links particles to each other across time windows. The raw PEPT data had 3,718,681 individual particle positions. Most particles traversing the scanner bore with the flow will have trajectories of length several cm. A filter removed trajectories with a positional standard deviation less than 5 mm. This reduced the number of particle positions to 237,728 and removed
stationary particles. The next filter removed trajectories with frame to frame displacement of greater than 2.5 mm. With a frame collected each ms, this filter removed particle velocities in excess of 2.5 m/s. This filter removed false linking of moving particles to stationary particles which could result in large spurious velocity values. This reduced the number of positions from 237,728 to 235,847.

Uncertainty is introduced in the measurement during smoothing of the trajectory data. The smoothing is performed with a centered 5 point moving average filter. The smoothing is designed to remove jaggedness in the linked positions of trajectories associated with random uncertainty in location of the particle from time window to time window. The smoothing biases the data where high gradients in velocity exist. For example, the leading corner of the first baffle is a sharp 90 degree corner, and the flow must accelerate around this obstruction. The trajectory smoothing causes particle trajectories near this corner to curve through the baffle volume as can be observed in Figure 22. Smoothing in this way is common to current PEPT implementation, and other methods are being explored to control the random errors.

Finally, a statistical uncertainty exists when LECSS constructs the average in each grid based on the trajectories passing through the grid area. The flow has real random velocity variations. A large number of trajectories contributes to more accurate reported average velocity values.

The total uncertainty in a velocity reported from PEPT is conservatively represented in equation 13, where $\delta V$ is the uncertainty in velocity from each identified uncertainty.

$$\delta V_{PEPT} = \delta V_{LOR} + \delta V_{Filtering} + \delta V_{Smoothing}$$

As before, the statistical contribution associated with averaging in LECSS’s grid location is not represented. Work is ongoing to quantify these contributions to error, and build rules for a protocol that leads to optimal experiment design. The PEPT approach has been in use for single particle tracking since the 1990’s, and Parker et al. (Parker, 1993) have examined the position resolution of the approach, and examined capability to record trajectories. Our research requires extension to examination of velocities and accelerations of particles, and this thesis starts us on that path.

Differences in experimental conditions during the PEPT and PTV measurements also contribute to uncertainty in the comparison reported herein. The bulk liquid flow rate was measured as 0.62 ± 0.02 L/s for the PTV measurements and as 0.63 ± 0.02 L/s for the PEPT measurements. The additional contribution to comparison of PEPT and PTV uncertainty is in the spatial co-location error for the measurement fields. This contribution is treated in the end to end uncertainty assessment presented in section 11. Some more integral flow measurement distortions may also be present, as the comparison of velocity profiles in Figure 34 suggests. These are under investigation.
12 Conclusions

The HSV and PEPT measurement techniques both involve a chain of processes from the initial set-up of the instruments through to the final presentation of accepted trajectories. Each link in this chain offers opportunity to add to the uncertainty in the final reported particle trajectory. The steps in the data acquisition and conditioning process are presented for the HSV and PEPT methods. The long-term objective of the research is to systematically improve the PEPT method for particle tracking in flows. This paper offers initial exploration of our ability to perform conventional optical measurement of particle trajectories and then systematically compare those measurements with trajectories measured using the multi-PEPT technique.

Particle location accuracy using the PEPT method is well established and the location uncertainty can be reduced by increasing the number of LOR (Bickell, 2012). It has been shown in experiments using the P4 scanner that the particle can be located to precision 0.34 mm in the radial direction and 0.32 mm in the axial direction using the multi-PEPT algorithms (Wiggins, 2016). These studies are useful, but the measurement of moving activated particles in a flow adds greatly to the particle location challenge. Flow test sections add to the scattering of emitted gammas, and distort the apparent sensitivity gradients of the scanner. The moving particle produces a different pattern of emission than a stationary one during each time segment used to capture LOR. These effects can be simulated, and this type of simulation is planned going forward to allow better experiment planning. In the interim, this paper offers an end-to-end comparison of PEPT with HSV where the measurement uncertainties are all wrapped into the process. This has helped to scale the importance of parameters contributing to uncertainty in both PEPT and in HSV data, and guide priorities for improvement.

A validation for Multi-PEPT is performed on a jet flow with Reynolds number 23,500 ± 800 utilizing HSV. Lagrangian trajectories are generated by HSV and PEPT. Both methods featured similar trajectory acceptance rates with HSV keeping 24% of reported trajectories, and PEPT keeping 23.8% of reported trajectories. HSV and Eulerian velocity fields are inferred from the HSV and PEPT trajectories and are compared on a grid-to-grid basis. It is found that the difference between the two measurements is less than 0.1 m/s for x-velocity and 0.05 m/s for the y-velocity for most of the field of view. However, in regions where the gradient is steep, co-location error results in larger discrepancies between the two measurements. Future work includes improvements to the co-location via optical and radiolabeled markers and the use of either a stereoscopic camera or a laser plane to illuminate the particles for the HSV. This will limit the depth of field for the HSV data and permit the control and correction of optical aberrations contributing to uncertainty in that measurement.

Work is underway to further quantify the uncertainty associated with our PEPT technique using both experimental methods and data from simulations. A complete GATE model of the Inveon scanner is available (Lee, 2013), and simulated data from moving particles are being generated to allow testing of the multi-PEPT software and to facilitate planning of experiments.
13 Division of Effort

The validation discussed in this thesis are the result of a group of people working together to accomplish a set of common goals. Contributors include Cody Wiggins, who developed the Multi-PEPT algorithm. Matthew Buttrey, assisted in the performance of the PEPT experiment as well as the original design of the fluid flow delivery system. The author would also like to thank ePlastics for interpreting the original 3D test section model as 2D design drawings and providing manufacturability advice as well as manufacturing the test section used in this experiment. Next, the author would like to thank Dr. David Felde of Oak Ridge National Lab for filming the HSV used in this thesis. The author also thanks Dr. Arthur Ruggles for leading the group, supporting and advising my research efforts, granting me academic freedom and permitting me the occasional mistake.

Prior to the experiment described in this thesis I, Seth Langford, led the prototype experiment discussed in Langford et al., 2016 with the help of Daniel Tenpenny. The effort involved spending many hours in the basement of the nuclear engineering building ensuring that the experimental setup was watertight and ready to be tested at the University of Tennessee Medical Center. With the assistance of Cody Wiggins, Daniel Tenpenny and Dr. Arthur Ruggles I wrote a conference paper which resulted in our group being recognized with a best paper award at the 2015 American Nuclear Society Student Conference as well as a best presentation award. The paper has since been published in Nuclear Engineering and Design. With the assistance of Daniel Tenpenny, I designed the test section that was used in this experiment. I also worked with Nitant Patel, to design the instrumentation suite used in the experiments presented in this thesis. I wrote the Lagrangian to Eulerian Conversion Software Suite (LECSS) used to process the trajectories generated by PEPT and HSV.
References


Dwyer, 2009, “Series OP Orifice Plate Flow Meter: Specifications- Installation and Operating Instructions,” FR# 443396-00 Rev. 1


http://emtoolbox.nist.gov/Wavelength/Documentation.asp


Peatross and Ware, 2015, “Physics of Light and Optics”, available at optics.byu.edu.


Siemens, 2014, “Inveon™ User Notes: Inveon Scanners and Inveon Acquisition Workplace 2.0”.


Appendices
Appendix A - Test Section Engineering Drawings

Figure 38 - Test Section Design Drawing A
Figure 39 - Test Section Design Drawing B
Figure 40 - Test Section Design Drawing C
Figure 41 - Test Section Design Drawing D
Figure 42 - Test Section Design Drawing E
Appendix B - Experimental Procedure

Equipment

- Flow loop with instrumentation
- Test section
- Inveon (PET Scanner)
- Fluid barrier
- Tank
- Cart
- 3 Styrofoam bases
- Bucket
- Mop
- 2 extension cords
- 3 power strips
- Motor power cables
- DAQ computer
- DAQ with printer cable
- Activated particles in microfuge tube
- Glycerin
- Geiger counter
- Towels
- Camera stand
- Jack
- Plastic Base

Testing Procedure

Day Before

- Turn on the scanner and move bed to the bottom and full forward position
  - If scanner gets turned off, do not turn it back on without first removing the test section and bore protector from the scanner bore
- Place the fluid barrier in scanner bore
- Ensure that the far alignment tick is aligned with the edge of the scanner bore that is closest to the pump. The two test area ticks denote the active scanning area of the scanner.
Tank Filling Procedure

- Place the tank on the yellow Styrofoam base which is on the cart (away from the scanner) and ensure that the tank valve is closed.
- If ready, fill the tank to the 4 gallon mark with DI water
- Attach tank inlet and outlet components of the loop and valve them off
- Move cart into position

- Continue to assemble the flow loop with the desired test section inside the bore of the scanner

- Ensure that the test section is straight level and centered with respect to the bore
  - Use foam cutout to cradle the test section and absorb vibration
  - Use jack on the bed side of the Inveon to level and center the test section
  - Use camera stand on the “window” side of the Inveon to level and canter the test section

- Ensure that the pump, AC/DC convertor and instrumentations are plugged into the three different power strips with an on/off switch that is beyond the shielding plates
  - On the first power strip plug the Pump
  - On the first power strip plug in the second power strip for the AC/DC convertor
  - On the second power strip plug in the AC/DC convertor
  - On the third power strip plug in the pressure transducers
- Open tank valve 1 and tank valve 2, recirculation valve, vent valve, isolation valve and motor valve (as indicated by red ticks on the valve being aligned)
- Ensure that drain valves handle is parallel with the flow and pointed towards the pump
- Ensure that the flow loop is filled with adequate amount of water and turn on the pump
- If pump does not have suction, gently tilt the flow loop towards the pump
- Ensure that the dp lines are not leaking and are water solid. If not bleed them.
- Close the recirculation valve
- Open and close the vent valve to force any avoids out of the test section
- Open and close the recirculation valve to remove any trapped air
- Once the loop is water solid, open the recirculation valve and close the vent valve and tank valves
- Ensure flow loop is not leaking

**Activating the instrumentation**

- Throttle the recirculation valve to at least 2/3 of the way closed prior to powering on instrumentation
- Ensure that all the pressure transducers are properly placed in the flow loop and attached to the correct DAQ channels
- Turn on the Instrumentation
- Ensure that the folder where the files are being saved is cleared
- Open the LabView file name “FlowLoop Using Orifice Plate PE-B-3” located on the desktop
- To start collecting data, click on Operate → Run
- To stop collecting data, click on “Stop” button located on top of the graph
  
  Note: Do not click on the red button because this will not save the data to a text file
- Ensure that the absolute pressure transducers are reading around 14 PSI and differential pressure transducer is reading between 0.1-5 PSID.
- Follow the previous three steps to ensure that all the pressure transducers are working properly and it is saving all the desired data.
- Turn off pump and all the instrumentations
- Close all valves and return the day of the test to ensure flow loop is not leaking and is still water solid

**Day of the Test**

- Open the recirculation valve
- Fill the injection line
  - Open the particle injection valve and fill up to just below the valve body with DI water
  - Close particle injection valve
• Turn on the pump and ensure that there are no leaks and that the loop is water tight.
• Actuate valves to remove bubbles if needed.
• Activate instrumentation using previously mentioned instrumentation procedure and verify the instrumentation is working properly.
• Using recirculation valve and instrumentation to reach desired flow rate (10 GPM, dP around 2 psi).
• Ensure that the scanner is ready to go and that scanning profile exists.

• Press stop on the instrumentation and turn off the pump.
• Move the existing data files to a new folder.
• Close labview.
• Turn off the pump.
• Particle Dump.
- Drop glycerin into microfuge tube and shake
- Open the particle injection valve and dump in particles, rinse walls and valve body with DI water if needed
- Let the particles sink to the bottom of the “clear Tee”

- Data Acquisition
  - Start work flow on the Inveon and begin acquiring data
  - Simultaneously, as soon as the Inveon begins to acquire data turn on the pump and begin acquiring data from labview
  - As soon, Inveon scan is complete, stop collecting the data from labview
  - Scan for 30 minutes on the region of interest

- To stop collecting data in labview, click on “Stop” button located on top of the graphs
  Note: Do not click on the red button because this will not save the data to a text file

- Once test is complete, shut off the pump and motor valve and close valves
- Return after 24-48 hours

Expected Radiation Exposure (Assuming 30 mCi Activated Particle)
Day after the Test

- Remove the kick-stand and place a container under the drain valve
- Open the drain valve

- After the container is filled, shut the drain valve and dispose of the water
- Repeat previous two steps until all the water is drained out of the loop (Gently tilt the flow loop to get all the water out)
- Remove the flow loop in such a manner as to reduce the risk of spilling water on the Inveon
Appendix C - Instrumentation Calibration Certificates

Model Number: PX409-150A5V  
Serial Number: 444462  
Date: 5/19/2015  
Job: WHS0001807  

Pressure Connection: 1/4-18 NPT Male

WIRING CODE

Electrical Connection: Integral Cable 4-Cord  
BLACK = - EXCITATION  
WHITE = + OUTPUT  
GREEN = N/C  
RED = + EXCITATION

CALIBRATION WORKSHEET

<table>
<thead>
<tr>
<th>Pressure PSIA</th>
<th>OUTPUT Vdc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>-0.002</td>
</tr>
<tr>
<td>75.00</td>
<td>2.500</td>
</tr>
<tr>
<td>150.00</td>
<td>5.001</td>
</tr>
<tr>
<td>75.00</td>
<td>2.500</td>
</tr>
<tr>
<td>0.00</td>
<td>-0.002</td>
</tr>
</tbody>
</table>

LIST Traceable Number(s): C-1956, C-1310

Omega Eng. Inc., certifies that the above instrumentation has been calibrated and tested to meet or to exceed the published specifications. This calibration was performed using instrumentation and standards that are traceable to the National Institute of Standards and Technology. This document also ensures that all testing performed complies with MIL-STD 45662-A, ISO 10012-1, and ANSI/NCSL Z540-1-1994 requirements. After Final Calibration our products are stored in an environmentally controlled stock room and are considered in bonded storage. Depending on environmental conditions and severity of use, factory calibration is recommended every one to three years after the initial service installation date.

Bruce Lott  
Accepted and Certified By  
5/19/2015  
Date

Omega Engineering Inc., One Omega Drive, Stamford, CT 06907  
http://www.omega.com  
email: info@omega.com  
phone (800) 826-6342

Figure 44 - Absolute Pressure Transducer 444462 Calibration
OMEGA ENGINEERING INC.
CERTIFICATE OF CALIBRATION

Model Number: PX409-150A5V
Serial Number: 427575
Date: 10/10/2014
Job: R10814

Pressure Connection: 1/4-18 NPT Male

WIRING CODE

Electrical Connection: Integral Cable 4-Cond
BLACK = - EXCITATION
WHITE = + OUTPUT
GREEN = N/C
RED = + EXCITATION

CALIBRATION WORKSHEET

<table>
<thead>
<tr>
<th>Pressure (PSIA)</th>
<th>OUTPUT (Vdc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.006</td>
</tr>
<tr>
<td>75.00</td>
<td>2.504</td>
</tr>
<tr>
<td>150.00</td>
<td>4.998</td>
</tr>
<tr>
<td>75.00</td>
<td>2.505</td>
</tr>
<tr>
<td>0.00</td>
<td>0.005</td>
</tr>
</tbody>
</table>

NIST Traceable Number(s): C-1956, C-1330

Omega Eng. Inc., certifies that the above instrumentation has been calibrated and tested to meet or to exceed the published specifications. This calibration was performed using instrumentation and standards that are traceable to the National Institute of Standards and Technology. This document also ensures that all testing performed complies with MIL-STD 46662-A, ISO 10012-1, and ANSI/NCSL Z540-1-1994 requirements. After Final Calibration our products are stored in an environmentally controlled stock room and are considered in bonded storage. Depending on environmental conditions and severity of use, factory calibration is recommended every one to three years after the initial service installation date.

Bruce Lott
Accepted and Certified By

Omega Engineering Inc., One Omega Drive, Stamford, CT 06907
http://www.omega.com  email: info@omega.com  phone (800) 826-6342

Figure 45 - Absolute Pressure Transducer 427575 Calibration
OMEGA ENGINEERING INC.
CERTIFICATE OF CALIBRATION

Model Number: PX409-005DWU5V
Serial Number: 441686
Date: 3/9/2015
Job: WHS0000818

Capacity: 5.00 PSID
Excitation: 24.00 Vdc
Technician: SS

Pressure Connection: 1/4-18 NPT Male

WIRING CODE
Electrical Connection: Integral Cable 4-Cond
BLACK = - EXCITATION
WHITE = + OUTPUT
GREEN = N/C
RED = + EXCITATION

CALIBRATION WORKSHEET

<table>
<thead>
<tr>
<th>Pressure PSID</th>
<th>OUTPUT Vdc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.003</td>
</tr>
<tr>
<td>2.50</td>
<td>2.504</td>
</tr>
<tr>
<td>5.00</td>
<td>5.006</td>
</tr>
<tr>
<td>2.50</td>
<td>2.503</td>
</tr>
<tr>
<td>0.00</td>
<td>0.003</td>
</tr>
</tbody>
</table>

NIST Traceable Number(s): C-1952, C-2470

Omega Eng. Inc., certifies that the above instrumentation has been calibrated and tested to meet or to exceed the published specifications. This calibration was performed using instrumentation and standards that are traceable to the National Institute of Standards and Technology. This document also ensures that all testing performed complies with MIL-STD 45662-A, ISO 10012-1, and ANSI/NCSL Z540-1-1994 requirements. After Final Calibration our products are stored in an environmentally controlled stock room and are considered in bonded storage. Depending on environmental conditions and severity of use, factory calibration is recommended every one to three years after the initial service installation date.

Bruce Lott
Accepted and Certified By

3/9/2015
Date

Omega Engineering Inc., One Omega Drive, Stamford, CT 06907
http://www.omega.com email: info@omega.com phone (800) 826-8342

Figure 46 - Differential Pressure Transducer 441686 Calibration
WATER AND LIQUID FLOW - CONCENTRIC BORE

\[ Q \text{ (GPM)} = 44.748 \times d^4 \times K \times Y \times F_a \times \sqrt{(\rho w) / (P_i)} \]

Where:
- 44.748 = physical constant
- d = bore in inches
- D = Pipe inside Diameter (inches)
- K = flow coefficient
- Y = expansion factor (Water and most liquids normally = 1)
- F_a = thermal expansion factor (Water and most liquids normally = 1)
- \(h/w\) = differential pressure (inches w/c)
- \(P_i\) = density @ Line (flowing) conditions in lbs./ft³

\[ C = 0.5959 + 0.0312\beta^{0.1} - 0.1840\beta^4 + 91.71\beta^{0.5} \times R_n^{0.75} \]

\[ \beta = \text{Beta Ratio (d/D)} \]

\[ K = C \times (1) / \left( \sqrt[1-\beta]{\rho} \right) \]

Figure 47 - Differential Pressure to Flow Rate Correlation (Dwyer, 2009)
Appendix D - Particle Activation Procedure

Large Bead Activation Procedure:

PI Arthur Ruggles

Resin Beads are activated with positron emitter F18, and later inserted into a flow test section where the bead trajectories are tracked in a PET scanner. The flow test protocol is already on file with radiation safety. Experience has led to a refined bead activation protocol, and an amendment is offered here.

Activation of Resin Beads, (Amberlyst A26 OH form is current favorite).

1. Place small centrifuge vial into syringe shield.
2. Place resin beads, 1 to 20, into vial via forceps.
3. Remove F18 Bolus (30 mCi max) from shield box and inject into vial.
4. Place vial in syringe shield behind shield blocks.
5. Place empty syringe in waste pig.
6. Leave the hood.
7. Allow the activated solution to remain in contact with the bead for 8 minutes. Every two minutes, with the centrifuge vial facing alternatingly upwards and downwards, tap the side of the centrifuge vial 4-6 times. This will force the bead to move through the solution into the opposite end of the syringe. Each time, replacing the centrifuge vial behind the lead shielding until the next agitation.
8. Uncap the centrifuge vial.
9. Evacuate the excess liquid from the centrifuge vial into the shielded waste vial using a syringe at an approximate rate of 20 μl/sec.
10. Draw 1 ml of ultrapure water into the syringe.
11. Rinse the particles with the water from the syringe indicated in step 10.
12. Repeat steps 9-11 four more times.
13. Remove the syringe shield.
15. Remove the bead via forceps and place into a glass vial located in the counting well for activity measurement.
16. Alternate to 15, place entire syringe into the counting well.

Particle Injection

a. Transport via rad safety particle vial inside of shielded container to the lab.
b. Ensure experimental flow setup is ready for testing.
c. Add ultrapure water to the 2/3 mark on the particle vial and shake.
d. “Dump” particles down particle injection port. Rinse the port and particle vial with ultrapure water as necessary.
Appendix E - PEPT LECSS
clc
clear all
close all

%%%%%%%%%%%%%%%%%%%%%%%%
%% PTV TRACKER FOR PEPT %%%
%%%%%%%%%%%%%%%%%%%%%%%%

PEPTParticles=4014;
o=1;
for u=1:PEPTParticles

readthis=sprintf('FilteredPosition%d.txt',u);
readin=dlmread(readthis,'t',2,0);

if u==1

PEPTData(1:length(readin),1)=u; % Traject
PEPTData(1:length(readin),2)=readin(:,1)/1000; % Time sec
PEPTData(1:length(readin),3)=readin(:,4)/1000; % x (m)
PEPTData(1:length(readin),4)=readin(:,2)/1000; % y (m)
PEPTData(1:length(readin),5)=readin(:,3)/1000; % z (m)

else

PEPTData(lastlength+1:length(readin)+lastlength,1)=u;% Traject
pt=readin(:,1);
PEPTData(lastlength+1:length(readin)+lastlength,2)=readin(:,1)/1000; % Time msec
PEPTData(lastlength+1:length(readin)+lastlength,3)=readin(:,4)/1000; % x (mm)
PEPTData(lastlength+1:length(readin)+lastlength,4)=readin(:,2)/1000; % y mm
PEPTData(lastlength+1:length(readin)+lastlength,5)=readin(:,3)/1000; % z mm
end
lastlength=length(PEPTData);
o=length(readin);
end

minycutoff=inf%(-19.05-2.28+2.56+.440)/1000; % mm
maxycutoff=inf%(19.05+1+.51-1.06-.320+.4)/1000; % mm
maxxcutoff=.1256-(2.11/1000)%((123.9-.7957+.25-1.15+2.21+.26-1.06)/1000; %mm
minxcutoff=.0038+(10.6/1000)%(14.43+.8+3-10+2.4+.32+2.49-.31-.28)/1000; %mm

i=1;
% Y filter so that particles are only in the channel
while i<length(PEPTData)
    if PEPTData(i,4)<=minycutoff

        PEPTData(i,:)=[];

        i=i-1;
    end
    i=i+1;
end

i=1
while i<length(PEPTData)
    if PEPTData(i,4)>=maxycutoff

        PEPTData(i,:)=[];

        i=i-1;
    end
    i=i+1;
end

i=1
while i<length(PEPTData)
    if PEPTData(i,3)<=minxcutoff

        PEPTData(i,:)=[];

        i=i-1;
    end
    i=i+1;
end

i=1
while i<length(PEPTData)
    if PEPTData(i,3)>=maxxcutoff

        PEPTData(i,:)=[];

        i=i-1;
    end
    i=i+1;
end
i=1

dlmwrite('PEPTTrajectories.txt',PEPTData,'precision', 16,'delimiter','\t','roffset',1,'coffset',1)

divby0forward=0% look for errors ie, dx=0
divby0backward=0% look for errors ie, dx=0
i=1; % index for indexing
t=0; % index for a trajectory number
l=0; % length of a trajectory
f=1,% yet another indexor
r=1,% grid indexor
Trajectory=0;
PreviousTraject=0;
lastTrajectLength=0;
% Convert from video frame of reference to real frame of reference

% User Controls

filename='PEPTTrajectories.txt';
Data=dlmread(filename,'\t',1,1);

minx=min(Data(:,3));
miny=min(Data(:,4));
maxx=max(Data(:,3));
maxy=max(Data(:,4));
minz=min(Data(:,5));
maxz=max(Data(:,5));
mint=min(Data(:,2));
maxt=max(Data(:,2));

% Now lets make the the trajectories start at 0
Data(:,2)=Data(:,2)-mint;
Data(:,3)=Data(:,3)-minx;
Data(:,4)=Data(:,4)-miny;
Data(:,5)=Data(:,5)-minz;

minx=min(Data(:,3));
miny=min(Data(:,4));
maxx=max(Data(:,3));
maxy=max(Data(:,4));
minz=min(Data(:,5));
maxz=max(Data(:,5));
mint=min(Data(:,2));
maxt=max(Data(:,2));
% And now lets take care of the fact that we enter from the far side of the
% scanner.. ie, shift the coordinate system so that xmax=xmin=0

Data(:,3)=maxx-Data(:,3);
Data(:,4)=maxy-Data(:,4);
Data(:,5)=maxz-Data(:,5);

% and for our information for working with the code

minx=min(Data(:,3));
miny=min(Data(:,4));
maxx=max(Data(:,3));
maxy=max(Data(:,4));
minz=min(Data(:,5));
maxz=max(Data(:,5));
mint=min(Data(:,2));
maxt=max(Data(:,2));

% mm in the y direction
% in Y direction;
XGridIncrimentor=1/(30*2.875); % growth of the grid size % Note, Must Fit Evenly into 1
YGridincrimentor=1/30; % growth of the grid size % Note, Must Fit Evenly into 1
dtIncrimentor=1/1; % Time is broken up into chunks that are this sized fractions of the total
frames
StatisticCut=10; % number of crossings a grid must have on it in order to be accepted
MinVelocity=.05 % m/s  min velocity a trajectory must be moving at in order to be averaged
into the grid.
dtstepper=dtIncrimentor;
TotalFrames=max(Data(:,2));

% growth of the grid size % Note, Must Fit Evenly into 1
GridsizeX=XGridIncrimentor % grid size as a fraction of the max of x
GridsizeY=YGridIncimentor % grid size as a fraction of the max of y

XGridLimit=GridsizeX*maxx;
YGridLimit=GridsizeY*maxy;

check=Trajectory;

while i<length(Data)+1

  % The while loop tells the velocity calculator how many frames a
  % trajectory exist for and the number value of the trajectory

  while Trajectory(length(Trajectory))==check
    Trajectory=Data(i,1);
  
    l=l+1;
    i=i+1;
    TrajectoryNumber=t+1;
    TrajectoryFrames=l;
    % kill the while loop when we reach the end
    if i>length(Data)
check=3
else
ccheck=Data(i,1);
end
end

% Calc velocity with forward dif
for b=i-1:i-2
    dt=(Data(b+1,2)-Data(b,2));

    Vx(TrajectoryNumber,f)=(Data(b+1,3)-Data(b,3))/(dt);
    Vy(TrajectoryNumber,f)=(Data(b+1,4)-Data(b,4))/(dt);
    Vz(TrajectoryNumber,f)=(Data(b+1,5)-Data(b,5))/(dt);
    V(TrajectoryNumber,f)=sqrt(Vx(TrajectoryNumber,f)^2+Vy(TrajectoryNumber,f)^2+Vz(TrajectoryNumber,f)^2);

    x(TrajectoryNumber,f)=Data(b,3);
y(TrajectoryNumber,f)=Data(b,4);

    % Num, x,y,Vx,Vy,Vmag,T,Vortmag,Vortk,VxVyMag Vz Z
    TrajectMatrix(1,f+lastTrajectLength)=TrajectoryNumber;
    TrajectMatrix(2,f+lastTrajectLength)=Data(b,3);
    TrajectMatrix(3,f+lastTrajectLength)=Data(b,4);
    TrajectMatrix(4,f+lastTrajectLength)=Vx(TrajectoryNumber,f);
    TrajectMatrix(5,f+lastTrajectLength)=Vy(TrajectoryNumber,f);
    TrajectMatrix(6,f+lastTrajectLength)=V(TrajectoryNumber,f);% vx vy vz mag
    TrajectMatrix(7,f+lastTrajectLength)=Data(b,2);
    TrajectMatrix(8,f+lastTrajectLength)=0;% vorticity x y z mag Place Holder
    TrajectMatrix(9,f+lastTrajectLength)= 0;% vorticity k aka z component Place Holder
    TrajectMatrix(10,f+lastTrajectLength)=
        (Vx(TrajectoryNumber,f)^2+Vy(TrajectoryNumber,f)^2)^.5;% VxVyMag
    TrajectMatrix(11,f+lastTrajectLength)= Vz(TrajectoryNumber,f);
    TrajectMatrix(12,f+lastTrajectLength)=Data(b,5);
f=f+1;
;
end

lastTrajectLength=f+lastTrajectLength-1;
f=1;
% allow the while loop to step forward
t=t+1;
l=0;
Trajectory=0;
check=Trajectory;
end
i=1;
a=1;
for i=1:length(TrajectMatrix)-1
if TrajectMatrix(1,i+1)==TrajectMatrix(1,i) % if we are on the correct traject do this!

   VortI=((TrajectMatrix(11,i+1)-TrajectMatrix(11,i))/(TrajectMatrix(3,i+1)-TrajectMatrix(3,i)))-((TrajectMatrix(5,i+1)-TrajectMatrix(5,i))/(TrajectMatrix(12,i+1)-TrajectMatrix(12,i)));
   VortJ=((TrajectMatrix(4,i+1)-TrajectMatrix(4,i))/(TrajectMatrix(12,i+1)-TrajectMatrix(12,i)))-((TrajectMatrix(11,i+1)-TrajectMatrix(11,i))/(TrajectMatrix(2,i+1)-TrajectMatrix(2,i)));
   VortK=((TrajectMatrix(5,i+1)-TrajectMatrix(5,i))/(TrajectMatrix(2,i+1)-TrajectMatrix(2,i)))-((TrajectMatrix(4,i+1)-TrajectMatrix(4,i))/(TrajectMatrix(3,i+1)-TrajectMatrix(3,i)));

   Vorticity=sqrt(VortI^2+VortJ^2+VortK^2);
   TrajectMatrix(8,a)=Vorticity;
   TrajectMatrix(9,a)=VortK;
a=a+1;

else  % else if we are at the end of the traject give us a nan that we will later delete
TrajectMatrix(8,a)=nan;
TrajectMatrix(9,a)=nan;
a=a+1;
end

end

% remove NaN and Infs from traject matrix
i=1;
while i<=length(TrajectMatrix)
    if ( mean((isnan(TrajectMatrix(:,i))==1)) || mean((isinf(TrajectMatrix(:,i))==1)) ~=0 )
        TrajectMatrix(:,i)=[];
i=i-1;
    end
    i=i+1;
end

XgridFrac=XGridIncrementor;
YgridFrac=YGridincrementor;
% Create a Grid for the Velocities
while XgridFrac <= 1
    % Growth of the xgrid
    XGridLimit(r)=minx+(maxx-minx)*XgridFrac;

    gridlimit(r,1)=minx+(maxx-minx)*XgridFrac

    XgridFrac=XgridFrac+XGridIncrementor;
r=r+1;
end
r=1;
while YgridFrac <= 1
    % Growth of the xgrid
    YGridLimit(r)=miny+(maxy-miny)*YgridFrac;
    gridlimit(r,2)=miny+(maxy-miny)*YgridFrac
    YgridFrac=YgridFrac+YGridincrementor;
    r=r+1;
end

ff=1;
aa=0;
Vmagsum=0;
r=1;
h=1;
tstep=2;
Framelimitstep=0;
% Discritize time
while dtIncrimentor <= 1
    Framelimitstep=mint+(maxt-mint)*dtIncrimentor
    FrameLimit(tstep)=Framelimitstep;
    dtIncrimentor=dtIncrimentor+dtstepper;
    tstep=tstep+1;
end

Vmagsum=0;
Vxsum=0;
Vysum=0;
Vzsum=0;
for tstep=2:length(FrameLimit)

% Calculate lower left trajct
for h=1:1
    for r=1:1
        for ff=1:length(TrajectMatrix)
            oops=1;
            if ((TrajectMatrix(2,ff) <= XGridLimit(r)) && (TrajectMatrix(3,ff) <= YGridLimit(h)) &&
                TrajectMatrix(6,ff) >= MinVelocity && (TrajectMatrix(7,ff) <= FrameLimit(tstep) &&
                TrajectMatrix(7,ff) >= FrameLimit(tstep-1)))
Vmagsum=Vmagsum+TrajectMatrix(6,ff);
Vxsum=Vxsum+TrajectMatrix(4,ff);
Vysum=Vysum+TrajectMatrix(5,ff);
Vzsum=Vzsum+TrajectMatrix(11,ff);
VorticitymagSum=VorticitymagSum+TrajectMatrix(8,ff);
VortkSum=VortkSum+TrajectMatrix(9,ff);
VxVyMagSum=VxVyMagSum+TrajectMatrix(10,ff);

VxValues(h,r,aa+1)=TrajectMatrix(4,ff);% Record all the values that make up the average in a grid
VyValues(h,r,aa+1)=TrajectMatrix(5,ff);
VxVyValues(h,r,aa+1)=TrajectMatrix(10,ff);

aa=aa+1;

else

Vmagsum=Vmagsum+0;
Vxsum=Vxsum+0;
Vysum=Vysum+0;
Vzsum=Vzsum+0;
VorticitymagSum=VorticitymagSum+0;
VortkSum=VortkSum+0;
VxVyMagSum=VxVyMagSum+0;
VxValues(h,r,aa+1)=NaN;
VyValues(h,r,aa+1)=NaN;
VxVyValues(h,r,aa+1)=NaN;

end

end
if aa>= StatisticCut
GridVmagAvg(h,r)=Vmagsum/(aa);
GridVxAvg(h,r)=Vxsum/(aa);
GridVyAvg(h,r)=Vysum/(aa);
GridVzAvg(h,r)=Vzsum/aa;
VorticityMagAverage(h,r)=VorticitymagSum/aa;
VortkAvg(h,r)=VortkSum/aa;
VxVyMagAvg(h,r)=VxVyMagSum/aa;
else
GridVmagAvg(h,r)=nan;
GridVxAvg(h,r)=nan;
GridVyAvg(h,r)=nan;
GridVzAvg(h,r)=nan;
VorticityMagAverage(h,r)=nan;
VortkAvg(h,r)=nan;
VxVyMagAvg(h,r)=nan;

end
Vmagsum=0;
Vxsum=0;
Vysum=0;
Vzsum=0;
VorticymagSum=0;
VortkSum=0;
VxVyMagSum=0;
aa=0;
end
aa=0;

% X min sweep except for r=1 (Left side)
for h=2:length(YGridLimit)
    for r=1:1
        for ff=1:length(TrajectMatrix)
            % Do this if we meet x criterion, y criterion and we are not
            % equal to 0
            if ((TrajectMatrix(2,ff)<= XGridLimit(r) ) && (TrajectMatrix(3,ff)<= YGridLimit(h) &&
                TrajectMatrix(3,ff) >= YGridLimit(h-1)) && TrajectMatrix(6,ff) >= MinVelocity &&
                (TrajectMatrix(7,ff)<= FrameLimit(tstep) && TrajectMatrix(7,ff) >= FrameLimit(tstep-1)))

        end
    end
end

85
Vmagsum=Vmagsum+TrajectMatrix(6,ff);  
Vxsum=Vxsum+TrajectMatrix(4,ff); 
Vysum=Vysum+TrajectMatrix(5,ff);  
Vzsum=Vzsum+TrajectMatrix(11,ff);  
VorticitymagSum=VorticitymagSum+TrajectMatrix(8,ff);  
VortkSum=VortkSum+TrajectMatrix(9,ff);  
VxVyMagSum=VxVyMagSum+TrajectMatrix(10,ff);  

VxValues(h,r,aa+1)=TrajectMatrix(4,ff);  
VyValues(h,r,aa+1)=TrajectMatrix(5,ff);  
VxVyValues(h,r,aa+1)=TrajectMatrix(10,ff);  

aa=aa+1; 

else 

Vmagsum=Vmagsum+0;  
Vxsum=Vxsum+0;  
Vysum=Vysum+0;  
Vzsum=Vzsum+0;  
VorticitymagSum=VorticitymagSum+0;  
VortkSum=VortkSum+0;  
VxVyMagSum=VxVyMagSum+0;  
VxValues(h,r,aa+1)=NaN;  
VyValues(h,r,aa+1)=NaN;  
VxVyValues(h,r,aa+1)=NaN;  

end 

end 

if aa>= StatisticCut  
GridVmagAvg(h,r)=Vmagsum/(aa);  
GridVxAvg(h,r)=Vxsum/(aa);  
GridVyAvg(h,r)=Vysum/(aa);  
GridVzAvg(h,r)=Vzsum/aa;  
VorticityMagAverage(h,r)=VorticitymagSum/aa;
VortkAvg(h,r)=VortkSum/aa;
VxVyMagAvg(h,r)=VxVyMagSum/aa;
else
  GridVmagAvg(h,r)=nan;
  GridVxAvg(h,r)=nan;
  GridVyAvg(h,r)=nan;
  GridVzAvg(h,r)=nan;
  VorticityMagAverage(h,r)=nan;
  VortkAvg(h,r)=nan;
  VxVyMagAvg(h,r)=nan;
end
Vmagsum=0;
Vxsum=0;
Vysum=0;
Vzsum=0;
VorticymagSum=0;
VortkSum=0;
VxVyMagSum=0;
aa=0;
end
aa=0;

% Y min sweep for h=1 (bottom side)
for h=1:1
  for r=2:length(XGridLimit)
    for ff=1:length(TrajectMatrix)
      if ((TrajectMatrix(2,ff)<=XGridLimit(r) && TrajectMatrix(2,ff)>=XGridLimit(r-1)) &&
        (TrajectMatrix(3,ff)<=YGridLimit(h)) && TrajectMatrix(6,ff)>=MinVelocity &&
        (TrajectMatrix(7,ff)<=FrameLimit(tstep) && TrajectMatrix(7,ff)>=FrameLimit(tstep-1)))
        Vmagsum=Vmagsum+TrajectMatrix(6,ff);
      end
    end
  end
end
Vxsum=Vxsum+TrajectMatrix(4,ff);
Vysum=Vysum+TrajectMatrix(5,ff);
Vzsum=Vzsum+TrajectMatrix(11,ff);
VorticitymagSum=VorticitymagSum+TrajectMatrix(8,ff);
VortkSum=VortkSum+TrajectMatrix(9,ff);
VxVyMagSum=VxVyMagSum+TrajectMatrix(10,ff);

VxValues(h,r,aa+1)=TrajectMatrix(4,ff); % Record all the values that make up the average in a
grid
VyValues(h,r,aa+1)=TrajectMatrix(5,ff);
VxVyValues(h,r,aa+1)=TrajectMatrix(10,ff);

aa=aa+1;

else

Vmagsum=Vmagsum+0;
Vxsum=Vxsum+0;
Vysum=Vysum+0;
Vzsum=Vzsum+0;
VorticitymagSum=VorticitymagSum+0;
VortkSum=VortkSum+0;
VxVyMagSum=VxVyMagSum+0;
VxValues(h,r,aa+1)=NaN;
VyValues(h,r,aa+1)=NaN;
VxVyValues(h,r,aa+1)=NaN;

end

end

if aa>= StatisticCut
GridVmagAvg(h,r)=Vmagsum/(aa);
GridVxAvg(h,r)=Vxsum/(aa);
GridVyAvg(h,r)=Vysum/(aa);
GridVzAvg(h,r)=Vzsum/aa;
VorticityMagAverage(h,r)=VorticitymagSum/aa;

end
VortkAvg(h,r)=VortkSum/aa;
VxVyMagAvg(h,r)=VxVyMagSum/aa;
else
    GridVmagAvg(h,r)=nan;
    GridVxAvg(h,r)=nan;
    GridVyAvg(h,r)=nan;
    GridVzAvg(h,r)=nan;
    VorticityMagAverage(h,r)=nan;
    VortkAvg(h,r)=nan;
    VxVyMagAvg(h,r)=nan;
end
Vmagsum=0;
Vxsum=0;
Vysum=0;
Vzsum=0;
VorticitymagSum=0;
VortkSum=0;
VxVyMagSum=0;
aa=0;
end
aa=0;

% loops through values in trajectory matrix for all x and y values.

% Do this if h and r are greater than 1. i.e., this covers the grid
% This part does not cover from 0 to the first grid limit

for h=2:length(YGridLimit)
    for r=2:length(XGridLimit)
        for ff=1:length(TrajectMatrix)

            % Do this if we meet x criterion, y criterion and we are not
            % equal to 0


if ((TrajectMatrix(2,ff)<= XGridLimit(r) && TrajectMatrix(2,ff) >= XGridLimit(r-1)) &&
(TrajectMatrix(3,ff)<= YGridLimit(h) && TrajectMatrix(3,ff) >= YGridLimit(h-1)) &&
TrajectMatrix(6,ff) >= MinVelocity && (TrajectMatrix(7,ff)<= FrameLimit(tstep) &&
TrajectMatrix(7,ff) >= FrameLimit(tstep-1)))

Vmagsum=Vmagsum+TrajectMatrix(6,ff);
Vxsum=Vxsum+TrajectMatrix(4,ff);
Vysum=Vysum+TrajectMatrix(5,ff);
Vzsum=Vzsum+TrajectMatrix(11,ff);
VorticitymagSum=VorticitymagSum+TrajectMatrix(8,ff);
VortkSum=VortkSum+TrajectMatrix(9,ff);
VxVyMagSum=VxVyMagSum+TrajectMatrix(10,ff);

VxValues(h,r,aa+1)=TrajectMatrix(4,ff);% Record all the values that make up the average in a
grid
VyValues(h,r,aa+1)=TrajectMatrix(5,ff);
VxVyValues(h,r,aa+1)=TrajectMatrix(10,ff);

aa=aa+1;

else

Vmagsum=Vmagsum+0;
Vxsum=Vxsum+0;
Vysum=Vysum+0;
Vzsum=Vzsum+0;
VorticitymagSum=VorticitymagSum+0;
VortkSum=VortkSum+0;
VxVyMagSum=VxVyMagSum+0;
VxValues(h,r,aa+1)=NaN;
VyValues(h,r,aa+1)=NaN;
VxVyValues(h,r,aa+1)=NaN;

end

end
if aa>= StatisticCut
    GridVmagAvg(h,r)=Vmagsum/(aa);
    GridVxAvg(h,r)=Vxsum/(aa);
    GridVyAvg(h,r)=Vysum/(aa);
    GridVzAvg(h,r)=Vzsum/aa;
    VorticityMagAverage(h,r)=VorticitymagSum/aa;
    VortkAvg(h,r)=VortkSum/aa;
    VxVyMagAvg(h,r)=VxVyMagSum/aa;
else
    GridVmagAvg(h,r)=nan;
    GridVxAvg(h,r)=nan;
    GridVyAvg(h,r)=nan;
    GridVzAvg(h,r)=nan;
    VorticityMagAverage(h,r)=nan;
    VortkAvg(h,r)=nan;
    VxVyMagAvg(h,r)=nan;
end
Vmagsum=0;
Vxsum=0;
Vysum=0;
Vzsum=0;
VorticitymagSum=0;
VortkSum=0;
VxVyMagSum=0;
aa=0;
end
end

XPosistion=XGridLimit; %m
YPosistion=YGridLimit;% m

[XPosistion,YPosistion]= meshgrid(XPosistion,YPosistion);
xgrid=XPosistion;
ygrid=YPosistion;

%%

% This script tricks the surface plot into allowing the last row and
% column of the Z data set to be plotted. Additionally, it also shifts
% everything to the left so that the surface plot will correctly plot the
% color map since by default it plots the lower left vertex of a grid
% value to be the color of the grid to the right of that vertex.
%%

correctedXgrid=zeros(size(xgrid,1)+1,size(xgrid,2)+1);
correctedYgrid=zeros(size(ygrid,1)+1,size(ygrid,2)+1);
CorrectedVmaggrid=zeros(size(GridVmagAvg,1)+1,size(GridVmagAvg,2)+1);
CorrectedVxgrid=zeros(size(GridVxAvg,1)+1,size(GridVxAvg,2)+1);
CorrectedVygrid=zeros(size(GridVyAvg,1)+1,size(GridVyAvg,2)+1);
CorrectedVzgrid=zeros(size(GridVzAvg,1)+1,size(GridVzAvg,2)+1);
CorrectedVortMaggrid=zeros(size(VorticityMagAverage,1)+1,size(VorticityMagAverage,2)+1);
CorrectedVortKgrid=zeros(size(VortkAvg,1)+1,size(VortkAvg,2)+1);
CorrectedVxVyMaggrid=zeros(size(VxVyMagAvg,1)+1,size(VxVyMagAvg,2)+1);

% correct our positions

for i=1:size(xgrid,1);
    for j=2:length(xgrid)+1;
        correctedXgrid(i,j)=xgrid(i,j-1);
    end
end
correctedXgrid(size(correctedXgrid,1),:)=correctedXgrid(size(correctedXgrid,1)-1,:);
correctedXgrid(:,1)=xgrid(1,1)-(xgrid(1,2)-xgrid(1,1));

for i=2:size(ygrid,1)+1
    for j=1:length(ygrid)
        correctedYgrid(i,j)=ygrid(i-1,j);
    end
end
correctedYgrid(:,size(correctedYgrid,2))=correctedYgrid(:,size(correctedYgrid,2)-1);
correctedYgrid(1,:)=ygrid(1,1)-(ygrid(2,1)-ygrid(1,1));

% Now for the Z ie, color mapped part
for i=1:size(GridVmagAvg,1)
for j=1:length(GridVmagAvg)
    CorrectedVmaggrid(i,j)=GridVmagAvg(i,j);
end
end

for i=1:size(GridVmagAvg,1)
for j=1:length(GridVmagAvg)
    CorrectedVxgrid(i,j)=GridVxAvg(i,j);
end
end

for i=1:size(GridVmagAvg,1)
for j=1:length(GridVmagAvg)
    CorrectedVygrid(i,j)=GridVyAvg(i,j);
end
end

for i=1:size(GridVmagAvg,1)
for j=1:length(GridVmagAvg)
    CorrectedVzgrid(i,j)=GridVzAvg(i,j);
end
end

for i=1:size(GridVmagAvg,1)
for j=1:length(GridVmagAvg)
    CorrectedVortMaggrid(i,j)=VorticityMagAverage(i,j);
end
end

for i=1:size(VortkAvg,1)
    for j=1:length(VortkAvg)
        CorrectedVortKgrid(i,j)=VortkAvg(i,j);
    end
end

for i=1:size(VortAvg,1)
    for j=1:length(VortAvg)
        CorrectedVxVyMagGrid(i,j)=VxVyMagAvg(i,j);
    end
end

%%

%% Calculate Curl for "Vorticity"
PEPTCurlI=(curl(correctedXgrid,correctedYgrid,CorrectedVygrid,CorrectedVzgrid));
PEPTCurlJ=(curl(correctedXgrid,correctedYgrid,CorrectedVzgrid,CorrectedVxgrid));
PEPTCurlK=(curl(correctedXgrid,correctedYgrid,CorrectedVxgrid,CorrectedVygrid));
CurlMag=sqrt(PEPTCurlI.^2+PEPTCurlJ.^2+PEPTCurlK.^2);

%%
Time=FrameLimit; %
deltaTime=(FrameLimit(length(FrameLimit))-FrameLimit(length(FrameLimit)-1));

% Vx
Vxplot=figure(1)
set(Vxplot,'Position',[50 50 800 600])
surf(correctedXgrid,correctedYgrid,CorrectedVxgrid)
colormap(jet);
shading interp
xlabel('X position (meters)','FontSize',20)
ylabel('Y position (meters)','FontSize',20)
c=colorbar
ylabel(c,'Vx (m/s)','FontSize',20)
CorrectedVxgridDeviation=std(CorrectedVxgrid(:),'omitnan');
caxis([-1 2])
caxis(caxis)
set(gca,'fontsize',20)
axis([0 .12 0 .04])
title(sprintf( 'PEPT Vx T= %f (s) dt= %f (s)',Time(tstep),deltaTime), 'FontSize',20)
vxname=sprintf('PEPT Vx %d.png',tstep-1);
saveas(Vxplot,vxname);

% our flow comes in backwards into the scanner, do this to make the entry
% point

saveas(Vxplot,vxname);

set(gcf, 'Renderer', 'painters')
set(gca,'nextplot','replacechildren')

% Vy
Vyplot=figure(2)
set(Vyplot,'Position',[50 50 800 600])

surf(correctedXgrid,correctedYgrid,CorrectedVygrid)

colormap(jet);
shading interp
xlabel('X position (meters)','FontSize',20)
ylabel('Y position (meters)','FontSize',20)
c=colorbar
ylabel(c,'Vy (m/s)','FontSize',20)
CorrectedVygridDeviation=std(CorrectedVygrid(:,),'omitnan');
caxis([-8 .8]);
set(gca,'fontsize',20)
caxis(caxis)
axis([0 .12 0 .04])
title(sprintf('PEPT Vy T= %f (s) dt= %f (s)',Time(tstep),deltaTime), 'FontSize',20)
Vyname=sprintf('PEPT Vy %d.png',tstep-1);
% our flow comes in backwards into the scanner, do this to make the entry
% point

saveas(Vyplot,Vyname);

set(gcf, 'Renderer', 'painters')
set(gca,'nextplot','replacechildren')

% Vx Vy magnitude
VxVyMagplot=figure(3)
set(VxVyMagplot,'Position',[50 50 800 600])
surf(correctedXgrid,correctedYgrid,CorrectedVxVyMagGrid)

colormap(jet);
shading interp
xlabel('X position (meters)','FontSize',20)
ylabel('Y position (meters)','FontSize',20)
c=colorbar
ylabel(c,'VxVyMagnitude','FontSize',20)
CorrectedVxVyMagGridDeviation=std(CorrectedVxVyMagGrid(:,),'omitnan');
caxis([0 2])
set(gca,'fontsize',20)
caxis(caxis)
axis([0 .12 0 .04])
title( sprintf( 'PEPT VxVyMagnitude T= %f (s) dt= %f (s)',Time(tstep),deltaTime), 'FontSize',20)

VxVyMagplotname=sprintf('PEPT VxVyMagplot %d.png',tstep-1);
% our flow comes in backwards into the scanner, do this to make the entry % point

saveas(VxVyMagplot,VxVyMagplotname);

set(gcf, 'Renderer', 'painters')
set(gca,'nextplot','replacechildren')

% K component of Vorticity, ie, into or out of the image field
VortKPlot=figure(4)
set(VortKPlot,'Position',[50 50 800 600])
surf(correctedXgrid,correctedYgrid,PEPTCurlK)
vortkmin=min(min(curl(correctedXgrid,correctedYgrid,CorrectedVxgrid,CorrectedVygrid),[]),[],'omitnan')
vortkmax=max(max(curl(correctedXgrid,correctedYgrid,CorrectedVxgrid,CorrectedVygrid),[]),[],'omitnan')
colormap(jet);
shading interp
xlabel('X position (meters)','FontSize',20)
ylabel('Y position (meters)','FontSize',20)
c=colorbar
ylabel(c,'Vorticity(1/s)','FontSize',20)
CorrectedVortKgridDeviation=std(CorrectedVortKgrid(:),'omitnan');
caxis([-70 70])
caxis(caxis)
set(gca,'fontsize',20)
axis([0 .12 0 .04])
title( sprintf( ' PEPT Vorticity Z Component T= %f (s) dt= %f (s)',Time(tstep),deltaTime), 'FontSize',20)

VortKname=sprintf('PEPT VortKPlot %d.png',tstep-1);
% our flow comes in backwards into the scanner, do this to make the entry % point

saveas(VortKPlot,VortKname);

set(gcf, 'Renderer', 'painters')
set(gca,'nextplot','replacechildren')
% Vorticity Magnitude
VortmagPlot=figure(5)
set(VortmagPlot,'Position',[50 50 800 600])

surf(correctedXgrid,correctedYgrid,CurlMag)

colormap(jet);
shading interp
xlabel('X position (meters)','FontSize',20)
ylabel('Y position (meters)','FontSize',20)
c=colorbar
ylabel(c,'Vorticity(1/s)','FontSize',20)
CurlMagGridDeviation=std(CurlMag(:),'omitnan');
caxis([.333*CurlMagGridDeviation 3*CurlMagGridDeviation])
caxis(caxis)
axis([0 .12 0 .04])
set(gca,'fontsize',20)
title(sprintf('PEPT Vorticity Magnitude T= %f (s) dt= %f (s)',Time(tstep),deltaTime),'
'FontSize',20)

Vortmagnamer=sprintf('PEPT VortmagPlot %d.png',tstep-1);
% our flow comes in backwards into the scanner, do this to make the entry
% point

saveas(VortmagPlot,Vortmagnamer);

set(gcf, 'Renderer', 'painters')
set(gca,'nextplot','replacechildren')

% Vx Vy Vz Velocity Magnitude Plot
VelocitymagPlot=figure(6)
set(VelocitymagPlot,'Position',[50 50 800 600])

surf(correctedXgrid,correctedYgrid,CorrectedVmaggrid)

colormap(jet);
shading interp
view(2)
xlabel('X position (meters)','FontSize',20)
ylabel('Y position (meters)','FontSize',20)
c=colorbar

ylabel(c,'Velocity (m/s)','FontSize',20)
CorrectedVmaggridGridDeviation=std(CorrectedVmaggrid(:),'omitnan');
caxis([0  2])
caxis(caxis)
set(gca,'fontsize',20)
axis([0 .12 0 .04])
title( sprintf( 'PEPT Velocity Magnitude T= %f (s) dt= %f (s)',Time(tstep),deltaTime),
'FontSize',20)
%title( sprintf( 'PEPT Velocity Magnitude'), 'FontSize',20)
Velmagname=sprintf('PEPT VelocMagPlot %d.png',tstep-1);
% our flow comes in backwards into the scanner, do this to make the entry
% point

saveas(VelocitymagPlot,Velmagname);

set(gcf, 'Renderer', 'painters')
set(gca,'nextplot','replacechildren')

figure(7)
arrowplot=quiver(correctedXgrid,correctedYgrid,CorrectedVxgrid./CorrectedVxVyMagGrid,CorrectedVygrid./CorrectedVxVyMagGrid,'g','MaxHeadSize',14)
set(arrowplot,'linewidth',.01)
set(gca,'color',[0 0 0])
set(gca,'fontsize',20)
axis([0 .12 0 .04])
xlabel('X position (meters)','FontSize',20)
ylabel('Y position (meters)','FontSize',20)
title( sprintf( ' PEPT Flow Velocity Vector Plot T= %f (s) dt= %f (s)',Time(tstep),deltaTime),
'FontSize',20)

VectorPlotName=sprintf('PEPT VectorPlot %d.png',tstep-1);
% our flow comes in backwards into the scanner, do this to make the entry
saveas(arrowplot, VectorPlotName);

set(gcf, 'Renderer', 'painters')
set(gca,'nextplot','replacechildren')

%%

end

%% Save Last Tstep Values as .mat
save('PEPTCorrectedVxVygrid.mat','CorrectedVxVyMagGrid')
save('PEPTCorrectedVortKgrid.mat','PEPTCurlK')
save('PEPTCorrectedVxgrid.mat','CorrectedVxgrid')
save('PEPTCorrectedVygrid.mat','CorrectedVygrid')
save('PEPTCorrectedXgrid.mat','correctedXgrid')
save('PEPTCorrectedYgrid.mat','correctedYgrid')

%% PEPT and CFD Specific Quantities
save('PEPTCorrectedVmaggrid.mat','CorrectedVmaggrid')
save('PEPTCorrectedVortmaggrid.mat','CurlMag')
figure(8)
plot(Data(:,3)*1000,Data(:,4)*1000,'.')
hold on
PixeltoMM=5.5195;
plot(214/PixeltoMM,(210-60)/PixeltoMM,'r*')
plot(212/PixeltoMM,(210-144)/PixeltoMM,'r*')
plot(396/PixeltoMM,(210-61)/PixeltoMM,'r*')
plot(379/PixeltoMM,(210-147)/PixeltoMM,'r*')

title('PEPT and HSV Co-Location')
xlabel('Distance (mm)','fontsize',20)
ylabel('Distance (mm)','fontsize',20)
set(gca,'fontsize',20)
legend('PEPT Trajectories','HSV Baffle Corners')
axis(1000*[0 .12 0 .04])

%% Now for the Vx variences
Binning=[-1 -.5:.05:.5 1]
for i=1:size(VxValues,1)
    for j=1:size(VxValues,2)
        for k=1:size(VxValues,3)
            if (VxValues(i,j,k)==0) && (isnan(GridVxAvg(i,j))==0)
                GridVxVarience(i,j,k)=GridVxAvg(i,j)-VxValues(i,j,k);
            else
                GridVxVarience(i,j,k)=nan;
            end
        end
    end
end

for i=15
    figure(i+20)
    histogram(GridVxVarience(i,round(size(GridVxAvg,2)*.5),:),Binning)
xlabel('VxMean-Vx','FontSize',20)
ylabel('Items Per Bin','FontSize',20)
set(gca,'fontsize',20)
title(sprintf('PEPT Probability Density Function VxMean=%f (m/s) X=%f (m) Y=%f (m), GridVxAvg(i,round(size(GridVxAvg,2)*.5)), xgrid(1,round(size(GridVxAvg,2)*.5)), ygrid(i,1), 'FontSize', 14)

VxPDFNum=num2str(i+20);
saveas(gcf, strcat('PEPTVxPDF', VxPDFNum), 'png')
end

%% Now for Vy Variences

for i=1:size(VyValues,1)
    for j=1:size(VyValues,2)
        for k=1:size(VyValues,3)
            if (VyValues(i,j,k)~=0) && (isnan(GridVyAvg(i,j))==0)
                GridVyVarience(i,j,k)=GridVyAvg(i,j)-VyValues(i,j,k);
            else
                GridVyVarience(i,j,k)=nan;
            end

        end
    end
    for i=1:size(GridVyAvg)
        figure(i+120)
        histogram(GridVyVarience(i,round(size(GridVyAvg,2)*.5),:), Binning)
        xlabel('VxyMean-Vxy', 'FontSize', 20)
        ylabel('Items Per Bin', 'FontSize', 20)
        set(gca, 'fontSize', 20)
        title(sprintf('PEPT Probability Density Function VyMean=%f (m/s) X=%f (m) Y=%f (m), GridVyAvg(i,round(size(GridVyAvg,2)*.5)), xgrid(1,round(size(GridVyAvg,2)*.5)), ygrid(i,1), 'FontSize', 14)

        VyPDFNum=num2str(i+120);
        saveas(gcf, strcat('PEPTVyPDF', VyPDFNum), 'png')
    end
%% Now for VxVy Variances

for i=1:size(VxVyValues,1)
    for j=1:size(VxVyValues,2)
        for k=1:size(VyValues,3)
            if (VxVyValues(i,j,k)~=0) && (isnan(VxVyMagAvg(i,j))==0)
                GridVxVyVarience(i,j,k)=VxVyMagAvg(i,j)-VxVyValues(i,j,k);
            else
                GridVxVyVarience(i,j,k)=nan;
            end
        end
    end
end

for i=1:size(VxVyMagAvg)
    figure(i+220)
    histogram(GridVxVyVarience(i,round(size(VxVyMagAvg,2)*.5),:),Binning)
    xlabel('VxyMean-Vxy','FontSize',20)
    ylabel('Items Per Bin','FontSize',20)
    set(gca,'fontsize',20)
    title(sprintf(' PEPT Probability Density Function VxVyMean=%f (m/s) X=%f (m) Y=%f (m)',VxVyMagAvg(i,round(size(VxVyMagAvg,2)*.5)),xgrid(i,round(size(VxVyMagAvg,2)*.5)),ygrid(i,1)),'FontSize',14)
    VxVyMagPDFNum=num2str(i+220);
    saveas(gcf,strcat('PEPTVxVyMagPDF',VxVyMagPDFNum),'png')
end
Appendix F - HSV LECSS

clc
clear all
close all
;
i=1 % index for indexing
t=0 % index for a trajectory number
l=0 % length of a trajectory
f=1% yet another indexor
r=1% grid indexor
Trajectory=0;
PreviousTraject=0;
lastTrajectLength=0;
% Convert from video frame of reference to real frame of reference

% User Controls

filename='HSVTake2Try4.txt';
Data=dlmread(filename,','t',[1 1 448283 4]); %1967208

FrameRate=1000; % FPS that the film was shot at
ImageXPixels=600; % Pixels in the x direction
ImageYPixels=220; % Pixels in the y direction
CalibrationLength=15.4/1000 % mm;
CalibartionPixels=85
pixelsPerMeter= CalibartionPixels/CalibrationLength
XGridIncromentor=1/(30*2.875) % growth of the grid size  % Note, Must Fit Evenly into 1
YGridIncromentor=1/30 % growth of the grid size  % Note, Must Fit Evenly into 1
dtIncrementor=1;% Time is broken up into chunks that are this sized fractions of the total frames

% Cutt offs
MinVelocity=.05 % m/s  min velocity a trajectory must be moving at in order to be averaged into the grid.
MinDisplacemnt=5/1000 % minimum distance a trajectory must travel in order to make it onward to averaging
StatisticCut=10;% number of crossings a grid must have on it in order to be accepted
% Shift coordinate system so that its orgin is in the lower left instead of % upper left.
FractionalVelocityFilter=200 ;% if the velocity between two frames on a traject changes by this much between two frames, delete the traject
% Make the lower left hand corner the orgin instead of the upper left
Data(:,4)=ImageYPixels-Data(:,4);

% Make 0,0 the smallest x,y pair
minx=min(Data(:,3)) ;
miny=min(Data(:,4)) ;
maxx=max(Data(:,3)) ;
maxy=max(Data(:,4)) ;

mint=min(Data(:,2)) ;
maxt=max(Data(:,2)) ;

Data(:,2)=Data(:,2)-mint;
Data(:,3)=Data(:,3)-minx;
Data(:,4)=Data(:,4)-miny;

minx=min(Data(:,3)) ;
miny=min(Data(:,4)) ;
maxx=max(Data(:,3)) ;
maxy=max(Data(:,4)) ;

mint=min(Data(:,2)) ;
maxt=max(Data(:,2)) ;

dtstepper=dtIncritermator;
TotalFrames=max(Data(:,2));

% growth of the grid size % Note, Must Fit Evenly into 1
GridsizeX=XGridIncritermator % grid size as a fraction of the max of x
GridsizeY=YGridincritermator % grid size as a fraction of the max of y

XGridLimit=GridsizeX*ImageXPixels;
YGridLimit=GridsizeY*ImageYPixels;
check=Trajectory;
i=1;
Trajectlength=1;%Frame
DataSpot=1;
%Filter the Data Based on Velocity
TrajectFrames=1;
while i<length(Data)

    if Data(i,1)==Data(i+1,1) % Get Traject Length in frames
        TrajectFrames=TrajectFrames+1;

    else
        TrajectStartSpot=DataSpot-TrajectFrames+1;
        Displacement=sqrt((Data(DataSpot,3)-Data(TrajectStartSpot,3))^2+(Data(DataSpot,4)-Data(TrajectStartSpot,4))^2)/pixelsPerMeter;

        if Displacement < MinDisplacement
            Data(TrajectStartSpot:DataSpot,:)=[];

            i=i-TrajectFrames;
            DataSpot=DataSpot-TrajectFrames;
            TrajectFrames=1;
        else
            TrajectFrames=1;

        end

    end

end

i=i+1;
DataSpot=DataSpot+1;
end

i=1;
while i<length(Data)+1

% The while loop tells the velocity calculator how many frames a
% trajectory exist for and the number value of the trajectory

while Trajectory(length(Trajectory))==check
    Trajectory=Data(i,1);

    l=l+1;
    i=i+1;
    TrajectoryNumber=t+1;
    TrajectoryFrames=l;
    % kill the while loop when we reach the end
    if i>length(Data)
        check=3
    else
        check=Data(i,1);
    end
end

% Calc velocity with forward dif
for b=i-l:i-2

dt=(Data(b+1,2)-Data(b,2))/FrameRate;
Vx(TrajectoryNumber,f)=(Data(b+1,3)-Data(b,3))/(dt*pixelsPerMeter);
Vy(TrajectoryNumber,f)=(Data(b+1,4)-Data(b,4))/(dt*pixelsPerMeter);
V(TrajectoryNumber,f)=sqrt(Vx(TrajectoryNumber,f)^2+Vy(TrajectoryNumber,f)^2);

x(TrajectoryNumber,f)=Data(b,3);
y(TrajectoryNumber,f)=Data(b,4);

% Trajectory, x,y,Vx,Vy,V, Frame, Vorticity, AngleChange
TrajectMatrix(1,f+lastTrajectLength)=TrajectoryNumber;
TrajectMatrix(2,f+lastTrajectLength)=Data(b,3);
TrajectMatrix(3,f+lastTrajectLength)=Data(b,4);
TrajectMatrix(4,f+lastTrajectLength)=Vx(TrajectoryNumber,f);
\begin{verbatim}
TrajectMatrix(5,f+lastTrajectLength)=Vy(TrajectoryNumber,f);
TrajectMatrix(6,f+lastTrajectLength)=V(TrajectoryNumber,f);
TrajectMatrix(7,f+lastTrajectLength)=Data(b,2);
TrajectMatrix(8,f+lastTrajectLength)=0;
TrajectMatrix(9,f+lastTrajectLength)=atand((abs(Data(b+1,4)-Data(b,4)))/abs((Data(b+1,3)-Data(b,3))));
f=f+1;

; end 

lastTrajectLength=f+lastTrajectLength-1; 
f=1; 
% allow the while loop to step forward 
t=t+1;
l=0;

Trajectory=0; 
check=Trajectory; 
end

i=1; 
a=1

i=1
a=1
TrajectLength=1;

% calculate the percent difference in velocity and put nans in at the 
% ends so they can be trimmed out.. ie each derivative has one less 
% point than the original.
while i<length(TrajectMatrix)-1

while ( ( i < length(TrajectMatrix) ) && ( TrajectMatrix(1,i)==TrajectMatrix(1,i+1) ) )
    TrajectLength=TrajectLength+1;

end
\end{verbatim}
TrajectMatrix(10,i)=abs((TrajectMatrix(6,i+1)-TrajectMatrix(6,i)))/TrajectMatrix(6,i); % fractional velocity difference
i=i+1;

previousSpot=i-TrajectLength;
end

TrajectMatrix(10,TrajectLength+previousSpot)=nan;
i=i+1;
TrajectLength=1
end

i=1;

% remove NaN and Infs from traject matrix
while i<=length(TrajectMatrix)
    if ( mean((isnan(TrajectMatrix(:,i))==1)) || mean((isinf(TrajectMatrix(:,i))==1)) ~=0 )
        TrajectMatrix(:,i)=[];
i=i-1;
    end
    i=i+1;
end

% Filter out trajects that have large jumps in them based on the fractional % change in velocity between two steps.
i=1;
a=1;
previousSpot=1;
progress=0;
subtractor=0;
while progress < length(TrajectMatrix)
a=previousSpot;
    while ( progress+TrajectLength < length( TrajectMatrix) ) &&
    (TrajectMatrix(1,a)==TrajectMatrix(1,a+1))
        TrajectLength=TrajectLength+1;
    end
    progress=progress+TrajectLength;
i=i+1;
a=a+1;
end
a = a + 1;

end
subtractor = 0;

if mean(TrajectMatrix(10, previousSpot:previousSpot+TrajectLength-1) >= FractionalVelocityFilter) > 0 )
    TrajectMatrix(:, previousSpot:previousSpot+TrajectLength-1) = [];
    subtractor = TrajectLength;
end

progress = progress + TrajectLength - subtractor;

previousSpot = TrajectLength + previousSpot - subtractor;

TrajectLength = 1;

end

% i=1
% a=1
% Filteredout=1;
% while a<=length(TrajectMatrix)-1
% if ( ( abs(TrajectMatrix(9,a+1)-TrajectMatrix(9,a)) >= 10 ) && (TrajectMatrix(1,a) == TrajectMatrix(1,a+1) ) && ( abs( TrajectMatrix(6,a+1)-TrajectMatrix(6,a) )/TrajectMatrix(6,a) ) >= 1 ) )
    Filteredout=Filteredout+1;
%
% FilteredOutVelocityDif( Filteredout)= abs( TrajectMatrix(6,a+1)-TrajectMatrix(6,a) )/TrajectMatrix(6,a);
% FilteredOutAngleDif( Filteredout)= abs(TrajectMatrix(9,a+1)-TrajectMatrix(9,a));
%
r=1;

XgridFrac=XGridIncementor;
YgridFrac=YGridIncementor;
% Create a Grid for the Velocities
while XgridFrac <= 1
    % Growth of the xgrid
    XGridLimit(r)=0+( max( Data(:,3) ) )*XgridFrac;

    gridlimit(r,1)=0+( max( Data(:,3) ) )*XgridFrac

    XgridFrac=XgridFrac+XGridIncementor;

    r=r+1;
end

r=1;
while YgridFrac <= 1
    % Growth of the xgrid
    YGridLimit(r)=0+( max( Data(:,4) ) )*YgridFrac;

    gridlimit(r,2)=0+( max( Data(:,4) ) )*YgridFrac

    YgridFrac=YgridFrac+YGridIncementor;
    r=r+1;
ff=1;
aa=0;
Vsum=0;
r=1;
h=1;
tstep=2;

Framelimitstep=0;
% Discretize time
while dtIncimentor <= 1

    Framelimitstep=0+(TotalFrames)*dtIncimentor
    FrameLimit(tstep)=Framelimitstep;
    dtIncimentor=dtIncimentor+dtstepper;
    tstep=tstep+1;
end

Vxsum=0;
Vysum=0;
tsum=0;
tstep=2
VorticitySum=0

for tstep=2:length(FrameLimit)

    % Calculate lower left trajct
    for h=1:1
        for r=1:1
            for ff=1:length(TrajectMatrix)
if ((TrajectMatrix(2,ff) <= XGridLimit(r) && TrajectMatrix(2,ff) >= 0) && (TrajectMatrix(3,ff) <= YGridLimit(h) && TrajectMatrix(3,ff) >= 0) && (TrajectMatrix(6,ff) > MinVelocity) && (TrajectMatrix(7,ff) <= FrameLimit(tstep) && (TrajectMatrix(7,ff) >= FrameLimit(tstep-1))))

    Vsum=Vsum+TrajectMatrix(6,ff);
    Vxsum=Vxsum+TrajectMatrix(4,ff);
    Vysum=Vysum+TrajectMatrix(5,ff);
    VorticitySum=VorticitySum+TrajectMatrix(8,ff);

    VxValues(h,r,aa+1)=TrajectMatrix(4,ff);% Record all the values that make up the average in a grid
    VyValues(h,r,aa+1)=TrajectMatrix(5,ff);
    VxVyValues(h,r,aa+1)=TrajectMatrix(6,ff);

    aa=aa+1;

else
    Vsum=Vsum+0;
    Vxsum=Vxsum+0;
    Vysum=Vysum+0;
    VorticitySum=VorticitySum+0;
    VxValues(h,r,aa+1)=NaN;
    VyValues(h,r,aa+1)=NaN;
    VxVyValues(h,r,aa+1)=NaN;
end

end
if aa>= StatisticCut
    GridVAvg(h,r)=Vsum/(aa);
    GridVxAvg(h,r)=Vxsum/(aa);
    GridVyAvg(h,r)=Vysum/(aa);
    VorticityAverage(h,r)=VorticitySum/aa;
else
    GridVAvg(h,r)=nan;
    GridVxAvg(h,r)=nan;
end
GridVyAvg(h,r)=nan;
VorticityAverage(h,r)=nan;
end
Vsum=0;
Vxsum=0;
Vysum=0;
VorticitySum=0;
aa=0;
end
end
aa=0;

% X min sweep except for r=1 (Left side)

for h=2:length(YGridLimit)
  for r=1:1
    for ff=1:length(TrajectMatrix)
      % Do this if we meet x criterion, y criterion and we are not
      % equal to 0
      if ((TrajectMatrix(2,ff)<= XGridLimit(r) && TrajectMatrix(2,ff) >= 0) &&
          (TrajectMatrix(3,ff)<= YGridLimit(h) && TrajectMatrix(3,ff) >= YGridLimit(h-1)) &&
          (TrajectMatrix(6,ff) > MinVelocity) &&
          (TrajectMatrix(7,ff)<= FrameLimit(tstep) &&
          (TrajectMatrix(7,ff) >= FrameLimit(tstep-1))))
        Vsum=Vsum+TrajectMatrix(6,ff);
        Vxsum=Vxsum+TrajectMatrix(4,ff);
        Vysum=Vysum+TrajectMatrix(5,ff);
        VorticitySum=VorticitySum+TrajectMatrix(8,ff);
        VxValues(h,r,aa+1)=TrajectMatrix(4,ff);% Record all the values that make up the
        VyValues(h,r,aa+1)=TrajectMatrix(5,ff);% up the average in a grid
        VxVyValues(h,r,aa+1)=TrajectMatrix(6,ff);
        aa=aa+1;
      else
      end
    end
  end
end
114
Vsum = Vsum + 0;
Vxsum = Vxsum + 0;
Vysum = Vysum + 0;
VorticitySum = VorticitySum + 0;
VxValues(h, r, aa+1) = NaN;
VyValues(h, r, aa+1) = NaN;
VxVyValues(h, r, aa+1) = NaN;
end

end
if aa >= StatisticCut
  GridVAvg(h, r) = Vsum / (aa);
  GridVxAvg(h, r) = Vxsum / (aa);
  GridVyAvg(h, r) = Vysum / (aa);
  VorticityAverage(h, r) = VorticitySum / aa;
else
  GridVAvg(h, r) = nan;
  GridVxAvg(h, r) = nan;
  GridVyAvg(h, r) = nan;
  VorticityAverage(h, r) = nan;
end
Vsum = 0;
Vxsum = 0;
Vysum = 0;
aa = 0;
end
end
aa = 0;

% Y min sweep for h=1 (bottom side)
for h = 1:1
  for r = 2:length(XGridLimit)
    for ff = 1:length(TrajectMatrix)
      % Do this if we meet x criterion, y criterion and we are not
      % equal to 0
if ((TrajectMatrix(2,ff)<= XGridLimit(r) && TrajectMatrix(2,ff) >= XGridLimit(r-1)) && (TrajectMatrix(3,ff)<= YGridLimit(h) && TrajectMatrix(3,ff) >= 0) && (TrajectMatrix(6,ff) > MinVelocity) && (TrajectMatrix(7,ff)<= FrameLimit(tstep) && (TrajectMatrix(7,ff) >= FrameLimit(tstep-1)))))

Vsum=Vsum+TrajectMatrix(6,ff);
Vxsum=Vxsum+TrajectMatrix(4,ff);
Vysum=Vysum+TrajectMatrix(5,ff);
VorticitySum=VorticitySum+TrajectMatrix(8,ff);

VxValues(h,r,aa+1)=TrajectMatrix(4,ff);% Record all the values that make up the average in a grid
VyValues(h,r,aa+1)=TrajectMatrix(5,ff);
VxVyValues(h,r,aa+1)=TrajectMatrix(6,ff);

aa=aa+1;
else
    Vsum=Vsum+0;
    Vxsum=Vxsum+0;
    Vysum=Vysum+0;
    VorticitySum=VorticitySum+0;
    VxValues(h,r,aa+1)=NaN;
    VyValues(h,r,aa+1)=NaN;
    VxVyValues(h,r,aa+1)=NaN;
end

end
if aa>= StatisticCut
    GridVAvg(h,r)=Vsum/(aa);
    GridVxAvg(h,r)=Vxsum/(aa);
    GridVyAvg(h,r)=Vysum/(aa);
    VorticityAverage(h,r)=VorticitySum/aa;
else
    GridVAvg(h,r)=nan;
    GridVxAvg(h,r)=nan;
    GridVyAvg(h,r)=nan;
    VorticityAverage(h,r)=nan;
end
Vsum=0;
Vxsum=0;
Vysum=0;
aa=0;
VorticitySum=0;
end
end
aa=0;

% loops through values in trajectory matrix for all x and y values.

% Do this if h and r are greater than 1. i.e. this covers the grid
% This part does not cover from 0 to the first grid limit
for h=2:length(YGridLimit)
    for r=2:length(XGridLimit)
        for ff=1:length(TrajectMatrix)

            % Do this if we meet x criterion, y criterion and we are not
            % equal to 0
            if ((TrajectMatrix(2,ff)<=XGridLimit(r) && TrajectMatrix(2,ff)>=XGridLimit(r-1))&&
                (TrajectMatrix(3,ff)<=YGridLimit(h) && TrajectMatrix(3,ff)>=YGridLimit(h-1))&&
                (TrajectMatrix(6,ff)>MinVelocity) && (TrajectMatrix(7,ff)<=FrameLimit(tstep) &&
                (TrajectMatrix(7,ff)>=FrameLimit(tstep-1))))

                Vsum=Vsum+TrajectMatrix(6,ff);
                Vxsum=Vxsum+TrajectMatrix(4,ff);
                Vysum=Vysum+TrajectMatrix(5,ff);
                VorticitySum=VorticitySum+TrajectMatrix(8,ff);
                VxValues(h,r,aa+1)=TrajectMatrix(4,ff);% Record all the values that make up the average in a
               VyValues(h,r,aa+1)=TrajectMatrix(5,ff);
                VxVyValues(h,r,aa+1)=TrajectMatrix(6,ff);
                aa=aa+1;

            else
                Vsum=Vsum+0;

            end
        end
    end
end
Vxsum=Vxsum+0;
Vysum=Vysum+0;
VorticitySum=VorticitySum+0;
VxValues(h,r,aa+1)=NaN;
VyValues(h,r,aa+1)=NaN;
VxVyValues(h,r,aa+1)=NaN;
end

e
end
if aa>= StatisticCut
GridVAvg(h,r)=Vsum/(aa);
GridVxAvg(h,r)=Vxsum/(aa);
GridVyAvg(h,r)=Vysum/(aa);
VorticityAverage(h,r)=VorticitySum/aa;
else
    GridVAvg(h,r)=nan;
    GridVxAvg(h,r)=nan;
    GridVyAvg(h,r)=nan;
    VorticityAverage(h,r)=nan;
end
Vsum=0;
Vxsum=0;
Vysum=0;
aa=0;
VorticitySum=0;
end
end

XPosistion=XGridLimit/(pixelsPerMeter); % Get our position into meters for plotting%
YPosistion=YGridLimit/(pixelsPerMeter);

[xgrid,ygrid]= meshgrid(XPosistion,YPosistion);

%%

% This script tricks the surface plot into allowing the last row and
% column of the Z data set to be plotted. Additionally, it also shifts
% everything to the left so that the surface plot will correctly plot the
% color map since by default it plots the lower left vertex of a grid
% value to be the color of the grid to the right of that vertex.
%%

correctedXgrid=zeros(size(xgrid,1)+1,size(xgrid,2)+1);
correctedYgrid=zeros(size(ygrid,1)+1,size(ygrid,2)+1);

CorrectedVgrid=zeros(size(GridVAvg,1)+1,size(GridVAvg,2)+1);
CorrectedVxgrid=zeros(size(GridVxAvg,1)+1,size(GridVxAvg,2)+1);
CorrectedVygrid=zeros(size(GridVyAvg,1)+1,size(GridVyAvg,2)+1);
CorrectedVortgrid=zeros(size(VorticityAverage,1)+1,size(VorticityAverage,2)+1);

% correct our positions

for i=1:size(xgrid,1);
    for j=2:length(xgrid)+1;
        correctedXgrid(i,j)=xgrid(i,j-1) ;
    end
end

for i=2:size(ygrid,1)+1
    for j=1:length(ygrid)
        correctedYgrid(i,j)=ygrid(i-1,j);
    end
end

correctedXgrid(size(correctedXgrid,1),:)=correctedXgrid(size(correctedXgrid,1)-1,:);
correctedXgrid(:,1)=xgrid(1,1)-(xgrid(1,2)-xgrid(1,1));

for i=2:size(ygrid,1)+1
    for j=1:length(ygrid)
        correctedYgrid(i,j)=ygrid(i-1,j);
    end
end

correctedYgrid(:,size(correctedYgrid,2))=correctedYgrid(:,size(correctedYgrid,2)-1);
correctedYgrid(1,:)=ygrid(1,1)-(ygrid(2,1)-ygrid(1,1));
% Now for the Z ie, color maped part

for i=1:size(GridVAvg,1)
    for j=1:length(GridVAvg)
        CorrectedVgrid(i,j)=GridVAvg(i,j);
    end
end

for i=1:size(GridVAvg,1)
    for j=1:length(GridVAvg)
        CorrectedVxgrid(i,j)=GridVxAvg(i,j);
    end
end

for i=1:size(GridVAvg,1)
    for j=1:length(GridVAvg)
        CorrectedVygrid(i,j)=GridVyAvg(i,j);
    end
end

for i=1:size(GridVAvg,1)
    for j=1:length(GridVAvg)
        CorrectedVortgrid(i,j)=VorticityAverage(i,j);
    end
end
%%

%% Calculate Curl for "Vorticity"

HSVCurlK=(curl(correctedXgrid,correctedYgrid,CorrectedVxgrid,CorrectedVygrid));

%%%% PLOTS%%%%
%%%  

%%
Time(tstep)=FrameLimit(length(FrameLimit))/FrameRate;
deltaTime=(FrameLimit(length(FrameLimit)) - FrameLimit(length(FrameLimit)-1))/FrameRate;

% Vx
Vxplot=figure(1)
set(Vxplot,'Position',[50 50 800 600])
surf(correctedXgrid,correctedYgrid,CorrectedVxgrid)
colormap(jet);
shading interp
xlabel('X position (meters)','FontSize',20)
ylabel('Y position (meters)','FontSize',20)
c=colorbar
ylabel(c,'Vx (m/s)','FontSize',20)
CorrectedVxgridDeviation=std(CorrectedVxgrid(:),'omitnan');
caxis([-1 2])
set(gca,'fontsize',20)
caxis(caxis)
axis([0 .12 0 .04])
title(sprintf( ' HSV Vx T= %f (s) dt= %f (s)',Time(tstep),deltaTime), 'FontSize',20)
vxname=sprintf('HighSpeed Vx %d.png',tstep-1);
view(2)
% our flow comes in backwards into the scanner, do this to make the entry
% point

saveas(Vxplot,vxname);

set(gcf, 'Renderer', 'painters')
set(gca,'nextplot','replacechildren')

% Vy
Vyplot=figure(2)
set(Vyplot,'Position',[50 50 800 600])

surf(correctedXgrid,correctedYgrid,CorrectedVygrid)
colormap(jet);
shading interp
xlabel('X position (meters)','FontSize',20)
ylabel('Y position (meters)','FontSize',20)
c=colorbar
set(gca,'fontsize',20)
ylabel(c,' HSV Vy (m/s)','FontSize',20)
CorrectedVygridDeviation=std(CorrectedVygrid(:),'omitnan');
caxis([-0.8 .8]);
caxis(caxis)
axis([0 .12 0 .04])
title( sprintf( ' HSV Vy T= %f (s) dt= %f (s)',Time(tstep),deltaTime), 'FontSize',20)
view(2)
Vyname=sprintf('HighSpeed Vy%d.png',tstep-1);
% our flow comes in backwards into the scanner, do this to make the entry
% point

saveas(Vyplot,Vyname);

set(gcf, 'Renderer', 'painters')
set(gca,'nextplot','replacechildren')

% VxVy magnitude
VxVyMagplot=figure(3)
set(VxVyMagplot,'Position',[50 50 800 600])

surf(correctedXgrid,correctedYgrid,CorrectedVgrid)
colormap(jet);
shading interp
xlabel('X position (meters)','FontSize',20)
ylabel('Y position (meters)','FontSize',20)
c=colorbar
ylabel(c,'VxVyMagnitude(m/s)','FontSize',20)
CorrectedVxVyMagGridDeviation=std(CorrectedVgrid(:,),'omitnan');
caxis([0 2])
caxis(caxis)
axis([0 .12 0 .04])
title( sprintf( ' HSV VxVyMagnitude T= %f (s) dt= %f (s)',Time(tstep),deltaTime), 'FontSize',20)
set(gca,'fontsize',20)
VxVyMagplotname=sprintf('HighSpeed VxVyMagplot %d.png',tstep-1);
% our flow comes in backwards into the scanner, do this to make the entry
% point
view(2)
saveas(VxVyMagplot,VxVyMagplotname);

set(gcf, 'Renderer', 'painters')
set(gca,'nextplot','replacechildren')

% K component of Vorticity, ie, into or out of the image field
CurlKPlot=figure(4)
set(CurlKPlot,'Position',[50 50 800 600])

surf(correctedXgrid,correctedYgrid,abs(CorrectedVygrid./CorrectedVxgrid))

colormap(jet);
shading interp
xlabel('X position (meters)','FontSize',20)
ylabel('Y position (meters)','FontSize',20)
c=colorbar
ylabel(c,'Vorticity(1/s)','FontSize',20)
set(gca,'fontsize',20)
HSVCurlKgridDeviation=std(HSVCurlK(:,),'omitnan');
caxis([-70 70])
caxis(caxis)
title( sprintf( ' HSV Vorticity Z Component T= %f (s) dt= %f (s)',Time(tstep),deltaTime), 'FontSize',20)
view(2)
VortKname=sprintf('HighSpeed HSVCurlKPlot %d.png',tstep-1);
axis([0 .12 0 .04])
saveas(CurllKPlot,VortKname);

set(gcf, 'Renderer', 'painters')
set(gca,'nextplot','replacechildren')

figure(5)
arrowplot=quiver(correctedXgrid,correctedYgrid,CorrectedVxgrid./CorrectedVgrid,CorrectedVygrid./CorrectedVgrid,'g','MaxHeadSize',14)
set(arrowplot,'linewidth',.01')
set(gca,'color',[0 0 0])
set(gca,'fontsize',20)
xlabel('X position (meters)','FontSize',20)
ylabel('Y position (meters)','FontSize',20)
axis([0 .12 0 .04])

title( sprintf( 'HSV Velocity Vector Plot T= %f (s) dt= %f (s)',Time(tstep),deltaTime), 'FontSize',20)
view(2)

VectorPlotName=sprintf('HighSpeed VectorPlot %d.png',tstep-1);

saveas(arrowplot,VectorPlotName);

set(gcf, 'Renderer', 'painters')
set(gca,'nextplot','replacechildren')

end

% Save Last Tstep Values as .mat
save('HSVCorrectedVxVygrid.mat','CorrectedVgrid')
save('HSVCorrectedVortKgrid.mat','HSVCurlK')
save('HSVCorrectedVxgrid.mat','CorrectedVxgrid')
save('HSVCorrectedVygrid.mat','CorrectedVygrid')
save('HSVCorrectedXgrid.mat','correctedXgrid')
save('HSVCorrectedYgrid.mat','correctedYgrid')

figure(8)
plot(1000*TrajectMatrix(2,:)/pixelsPerMeter,1000*TrajectMatrix(3,:)/pixelsPerMeter,'.')
title('HSV')

%% Now for the Vx variances
Binning=[-1 -.5:.05:.5 1];

for i=1:size(VxValues,1)
    for j=1:size(VxValues,2)
        for k=1:size(VxValues,3)
            if (VxValues(i,j,k)~=0) && (isnan(GridVxAvg(i,j))==0)
                GridVxVarience(i,j,k)=GridVxAvg(i,j)-VxValues(i,j,k);
            else
                GridVxVarience(i,j,k)=nan;
            end
        end
    end
end

for i=15
    figure(i+20)
    histogram(GridVxVarience(i,round(size(GridVxAvg,2)*.5),:),Binning)
    xlabel('VxMean-Vx','FontSize',20)
    ylabel('Items Per Bin','FontSize',20)
    set(gca,'fontSize',20)
%% Now for Vy Variances

for i=1:size(VyValues,1)

    for j=1:size(VyValues,2)

        for k=1:size(VyValues,3)

            if (VyValues(i,j,k)~=0) && (isnan(GridVyAvg(i,j))==0)

                GridVyVariance(i,j,k)=GridVyAvg(i,j)-VyValues(i,j,k);

            else

                GridVyVariance(i,j,k)=nan;

            end

        end

    end

end

for i=1:size(GridVyAvg)

    figure(i+120)

    histogram(GridVyVariance(i,round(size(GridVyAvg,2)*.5),:),Binning)

    xlabel('VxyMean-Vxy','FontSize',20)

    ylabel('Items Per Bin','FontSize',20)

    set(gca,'fontsize',20)

    title(sprintf(' HSV Probability Density FunctionVyMean=%f (m/s) X=%f (m) Y=%f (m)',GridVyAvg(i,round(size(GridVyAvg,2)*.5)),xgrid(1,round(size(GridVyAvg,2)*.5)),ygrid(i,1)),'Font Size',14)

    VyPDFNum=num2str(i+120);

    saveas(gcf,strcat('HSVVyPDF',VyPDFNum),'png')

end

%%% Now for VxVy Variances
for i=1:size(VxVyValues,1)
    for j=1:size(VxVyValues,2)
        for k=1:size(VyValues,3)
            if (VxVyValues(i,j,k)~=0) && (isnan(GridVAvg(i,j))==0)
                GridVxVyVarience(i,j,k)=GridVAvg(i,j)-VxVyValues(i,j,k);
            else
                GridVxVyVarience(i,j,k)=nan;
            end
        end
    end
end
for i=1:size(GridVAvg)
    figure(i+220)
    histogram(GridVxVyVarience(i,round(size(GridVAvg,2)*.5),:),Binning)
    xlabel('VxyMean-Vxy','FontSize',20)
    ylabel('Items Per Bin','FontSize',20)
    set(gca,'fontsize',20)
    title(sprintf(' HSV Probability Density Function VxVyMean=%f (m/s) X=%f (m) Y=%f (m)',GridVAvg(i,round(size(GridVAvg,2)*.5)),xgrid(1,round(size(GridVAvg,2)*.5)),ygrid(i,1)),'FontSize',14)
    VxVyMagPDFNum=num2str(i+220);
    saveas(gcf,strcat('HSVxVyMagPDF',VxVyMagPDFNum),'png')
end
Vita

Seth Langford was born in Biloxi, Mississippi on January 18, 1992. He attended grade school in Mississippi, Florida and Tennessee. In 2010 he graduated from McMinn Central high school and began attending the University of Tennessee in pursuit of a Bachelor of Science degree in Nuclear Engineering. In 2014, he received his Bachelor’s degree and started graduate school with the intent of earning a Master of Science degree in Nuclear Engineering. He will receive his degree in spring 2016.