Development of a ROV Deployed Video Analysis Tool for Rapid Measurement of Submerged Oil/Gas Leaks

Final Report

Reporting Period
May 4, 2014 – March 31, 2016

Ömer Savaş

Submission Date
April 3, 2017

Award No: DE-FE0014314

Submitting Organization
The Regents of the University of California
c/o Sponsored Projects Office,
2150 Shattuck Avenue, Suite 300,
Berkeley, CA 94704-5940

Abstract

Expanded deep sea drilling around the globe makes it necessary to have readily available tools to quickly and accurately measure discharge rates from accidental submerged oil/gas leak jets for the first responders to deploy adequate resources for containment.

We have developed and tested a field deployable video analysis software package which is able to provide in the field sufficiently accurate flow rate estimates for initial responders in accidental oil discharges in submarine operations. The essence of our approach is based on tracking coherent features at the interface in the near field of immiscible turbulent jets.

The software package, **`UCB_Plume`**, is ready to be used by the first responders for field implementation. We have tested the tool on submerged water and oil jets which are made visible using fluorescent dyes. We have been able to estimate the discharge rate within 20% accuracy.

A high end WINDOWS laptop computer is suggested as the operating platform and a USB connected high speed, high resolution monochrome camera as the imaging device are sufficient for acquiring flow images under continuous unidirectional illumination and running the software in the field. Results are obtained over a matter of minutes.

Table of contents

Introduction

Expanded deep sea drilling around the globe makes it necessary to have readily available tools to quickly and accurately estimate discharge rates from accidental submerged oil/gas leak jets (Figure 1). We had proposed developing a field deployable video analysis software package which is able to provide in the field sufficiently accurate flow rate estimates for initial responders in accidental oil discharges in submarine operations. The essence of our approach is based on tracking in the near field flow features at the interface of immiscible turbulent jets. Then, the flow velocities at the edge are intrapolated to the center of the jet after which the flux is estimated.

The task assumed by Berkeley is to develop the software and to test it in small scale experiments designed to obtain detailed, high-resolution PIV velocity maps of coherent visible features and inside a transparent oil jet under very closely controlled conditions. The small-scale water and oil jet experiments allows for viewing and measuring the velocity of visible features and relate the velocity of visible features to the internal velocity profiles.

We have developed a software package, **`UCB_Plume`**, that is ready to be used by the first responders for field implementation. We have tested the tool on submerged water and oil jets which are made visible using fluorescent dyes. We are able to estimate the discharge rate within 20% accuracy.

A high end WINDOWS laptop computer is suggested as the operating platform and a high speed, high resolution monochrome camera as imaging device are sufficient for acquiring flow images under continuous unidirectional illumination and running the software in the field over a matter of minutes.

Berkeley experiments

A series of small-scale water and oil jet experiments are designed for viewing and measuring the velocity of visible features and relate the velocity of these features to the internal velocity profiles. The data are used to fine tune the parameters of the image correlation velocimetry algorithm and flow rate estimation strategy.

Experimental setup

The experiments are setup in a 4'×4'×8' glass tank located in 140 Hesse Hall of UC Berkeley Campus. The first set of experiments conducted in August 2014 employed horizontal discharge tubes. While this is of no concern when discharging water into water, it was obviously a severe restriction when oil (of lower density than water) was injected into water. Therefore, the setup is reconfigured for vertical discharge. Figure 2 shows the overall view of the setup. Figure 3 is a mosaic of various views and components. The schematic of the vertical discharge flow setup is shown in Figure 4 along with the specifications of its major components. The flow fields are recorded using simultaneous schlieren, PIV, and FV cameras at high speed; IDT X3 and Y3 cameras. A 10W (CW) Argon Ion laser is employed for both flow visualization (FV) and PIV. The laser light sheet illuminates the flow from the side of the tank. The PIV and FV cameras are positioned normal to the plane of the laser sheet. The schlieren system is wrapped around the tank by folding the classical Z-configuration using flat mirrors and set at 12 degrees off the normal the laser sheet in order to allow clear 90-degree access for the PIV and FV cameras. The flow is driven by a pump and the flow rate is monitored using an industrial grade 1% accurate turbine flow meter.

In order to minimize the use of oil, the gear pump was driven by a micro-stepper motor under computer control for 10-20 seconds at a time. For simultaneous vertical- and horizontal- knife edge schlieren images, the optics is reconfigured temporarily where the uninterrupted light bean was split by a cubic beam splitter for the two schlieren imaging schemes. For fluorescent-dyed water and oil jet visualization experiments, the *shell* of the jets were visualized under oblique LED illumination at about 35 degrees.

The discharge tube was fixed as 1/2" nominal diameter smooth copper tube with 16-inch length. The flow at the inlet to the tube is tripped with a mesh screen. Therefore, the length of the tube is sufficient for fully developed turbulent flow to establish at sufficiently high Reynolds numbers.

Sample images

Figures 5 and 6 show horizontal discharge oil experiments, sample shadowgraph and PIV runs. Due to the buoyancy of the jet, the flows bend upward quickly. Even though these are reasonable experiments in their own right, they are not amenable to verification of code development. Hence the horizontal discharge experiments are quickly abandoned in favor of vertical discharge runs.

- Figures 7 shows a sample schlieren image of a vertically discharging water jet and Figure 8 a sample PIV image.
- Figure 9 is mosaic of simultaneous shadowgraph and visible images of vertical oil jets of 5cs silicone oil at 45 cm^3/s (Re=900); and of oil jet of 1cs silicone oil at 360 cm^3/s (Re=35,000).
- Figure 10 shows sample flow images at Q=1.0 gallon/min (63 cm^3/s): schlieren with vertical knife edge and horizontal knife edge. The images are not simultaneous.
- Figure 11 shows simultaneous schlieren and cross-sectional fluorescent dye images at Q=1.0 gallon/min (63 cm^3/s), vertical knife edge.
- Figure 12 shows simultaneous schlieren and PIV images at Q=1.0 gallon/min (63 cm^3/s), vertical knife edge.
- Figure 13 shows simultaneous schlieren and PIV images at Q=1.0 gallon/min (63 cm^3/s), horizontal knife edge.

Completed experimental work

We completed the data acquisition phase of the project by the end of January 2015. These final experiments included fluorescent-opaque water discharge experiments, where the shadows of the edge features of the flows are recorded simultaneously with schlieren images. We also recorded *simultaneous* vertical and horizontal knife edge schlieren visualizations of water jets. Lastly, we carried out opaque oil discharge runs using 1cs and 5sc silicone oils which were made fluorescent-opaque using an oil soluble fluorescent dye. These data are cataloged for algorithm development.

Rudimentary data processing

Figures 14 and 15 show the results of rudimentary image correlation velocimetry results corresponding to images in figure 11 and 12, respectively. We have used our inhouse software called WALPT for processing. All image pairs are processed using identical processing parameters, such as interrogation window size, using our in-house code. No preprocessing is done. The results in these figures are not normalized. They are presented for qualitative discussion only. Clearly, results based on the schlieren images and flow visualization images do not even identify the jet. At this stage, it is not immediately clear what the reason for this behavior is. For example, that the flow seems reversed in the right frame of figure 14 may be an artifact of the decreasing dye concentration hence, decreasing image intensity as the jet gets diluted through entrainment. On the other hand, the particle image (Figure 15, right frame) very clearly identifies the jet. Obviously, if an autonomous processing algorithm is desired, much innovation has to be done.

Algorithm development

We have developed of a novel image processing technique for analyzing video decks.. The impetus for this development is the necessity of separating features that are moving at different speeds in a video stream. In the context of flow discharge, eddies of different size move or seem to move at different speeds in video streams. A similar case is observed when looking into discharges via schlieren videography. Our eyes can easily distinguish such features by comparing *successive* images. The approach we devised attempts to mimic human perception. The process is illustrated in figure 16. To identify a feature moving at given speed, we look at successive images. We extract from the video stream, the time history of image intensity at a given pixel $I_{ij}(t)$. A slow moving feature will have a slowly varying signature in $I_{ij}(t)$ and a fast moving, a fast varying signature. Therefore, filtering (or, equivalently, convolution) of $I_{ij}(t)$ allows us selectively extract features for their propagation speeds. We name this process PIXelwise TIme Filtering or PIXTIF for short. The process preserves spatial gradients. Once a video stream is processed for PIXTIF, the images can be further process to extract tractable features, such as Canny filtering to detect edges for image correlation velocimetry. The implementation of PIXTIF using Fourier transforms is straight forward: The Fourier transform of $I_{ij}(t)$ is multiplied by a suitably chosen filter kernel $G(\omega)$ in frequency domain. The resultant spectrum is inverted to construct the PIXTIFed image $\sim I_{ij}(t)$:

$$\sim I_{ij}(t)= F^{-1} \{ F [I_{ij}(t)]*G(\omega) \}. \qquad (1)$$

The process requires large computational resources, especially, computer memory. The whole video stream must be loaded at once into the computer memory for rapid access. The construction of the filter kernel $G(\omega)$ requires delicate attention. In particular, the filter shape (band-pass, low/high-pass) and cut-off parameter as well as the cut of rate are critical to obtaining useful results. For example, when employing a Gaussian filter,

$$G(\omega)=\exp[-(\omega - \omega_o)^2 / \Delta] \qquad (2)$$

the center frequency ω_o and the filter width Δ must be carefully chosen. These parameters are linked to the inertial range of the turbulence spectrum of the flow and must be adjusted for the Reynolds number. We have chosen an iterative approach to settle on the final choices for flow rate estimation. This strategy is built into the software package and is not discussed here further.

In parallel, we have developed methods to process uncompressed AVI files and subroutines to analyze image sequences. Figure 17 shows a PIXTIF example. In this exercise, a schlieren video sequence is PIXTIF'ed as illustrated in figure 16 using a Gaussian bandpass filter in frequency domain. The left image in figure 17 shows the PIXTIF'ed version of the schlieren image on the left in figure 16. The corresponding image pair is processed through the correlation velocimetry algorithm using the very same parameter used earlier. The resulting velocity vector field is shown on the right in figure 17. Clearly, the discharge jet is now well captured.

Discharge rate estimation

We assume that, in the region of interest of a developed or nearly developed turbulent jet, the average velocity profile u(r), averaged over both radial and axial directions, can be adequately described as a Gaussian bell curve (Figure 18)

$$u(r) / U_c = \exp[-r^2/\sigma^2] \qquad (3)$$

where U_c is the average centerline velocity and σ a width parameter. At the mean visible outer shell of the boundary r_c, the velocity of the eddies, the celerity C, is, from Equation (3),

$$C / U_c = \exp[- r_c^2/\sigma^2]. \qquad (4)$$

Similar to the classical shear layer, if we assume that we can relate to U_c to C to as

$$U_c = (1+\varphi^{-1/2})\, C \qquad (5)$$

where φ is the density ratio, we can extract from Equation (4) the width parameter σ as

$$\sigma^2 = r_c^2 / \mathit{ln}(1+\varphi^{-1/2}). \qquad (6)$$

The flow rate Q is now deduced, by integration the Gaussian profile (3) over the interval $[0, r_c]$ as

$$Q = \pi r_c^2 C / [\varphi^{1/2}\, \mathit{ln}\, (1+\varphi^{-1/2})] \qquad (7)$$

where r_c is the radius of the visible discharge radius, C the average celerity of the eddies at r_c and φ is now the specific gravity (sg) of the fluid discharging into water. Note that $Q=1.44\pi r_c^2 C$ when $\varphi =1$, which corresponds to the case where the ambient fluid and the discharging fluid are the same, such as water jet in water.

The key assumption in proceeding approach is that, if the jet and ambient fluids are immiscible, as the case is in an oil spill, flow images show the fluid interface, hence, the velocity information based on those images is the celerity $C(r_c)$ at the interface. The implication is that, one would obtain nearly uniform celerity around during the data analysis. Figure 20 shows a water jet made visible with fluorescent dye at Q=1 gpm, Re=4500. The intricacy of the jet-ambient fluid interface is clearly delineated due to low molecular diffusivity of water. The velocity of the interface features are expected to be uniform around the *interface shell*. Figure 21 shows the velocity profile at the *interface shell* of the jet fluid which is nearly uniform over the width of the jet.

In proposing the flow rate calculation in the above equation (7), we neglected entrainment of the ambient fluid into the jet. Since we focus our attention on the near field of the jet, the fraction of the entrained fluid is still small, hence the error incurred is well within the uncertainty inherent in the calculation of Q. Further, the incurred error will tend to make Q an overestimate of the actual discharged species flow rate, which is less undesirable than an underestimated value. Instead of the Gaussian profile in Eq. (3), on could propose the algebraic profile deduced from the assumption of constant eddy viscosity across the jet. The final results, however, show little difference.

Image Processing

Video stream are processed at multiple steps for flow rate estimation. The flow chart in Chart 1 highlights the procedure. Figure 22 shows a snapshot from the video stream, the average flow field, and its rms field. The details in Figure 22a highlight the flow scales that have to be considered. These scales depend of the jet Reynolds number. An a priori estimate of these scales is later be used in setting the processing parameters for the video image deck. The average picture in (b) is used to establish the jet geometry in autonomous operation. The rms picture in (c) is used to determine the average edge of the jet fluid r_c over the region of interest, ROI.

Figure 23 shows highlights of the image processing prior to correlation velocimetry. Figure 23a shows the PIXTIF'ed result of image in Figure 22a. PIXTIF'ed video deck is used to determine the velocity field via cross correlation of deformed fluid parcels in WALPT. The picture in Figure 23(b) shows the Canny edge-detection result corresponding to the PIXTIF'ed image in 23(a). We are currently continuing with PIXTIF'ed image deck. The highlighted strip in Figure 23(c) marks the region of interest which is used to determine the edge of the jet fluid r_c. The strip, which is determined from the rms picture in Figure 22(c) is also used to mask the results of WALPT to determine the average celerity C at the interface over ROI. Note that ROI is deliberately chosen to be in the near field of the jet, where it may not have fully developed yet. The advantage of this choice is that the flow feature can be clearly identified.

Implementation

The implementation of the flow rate calculation in equation (7) is done by the batch script **plume.bat,** which is outlined in Chart 2. After some minimal interaction from the operator, the software produces its final results on the screen and archives them permanently. Chart 3 shows the screen flow during the execution of the batch process, which simply requires a double click on the **plume** icon to start.

Sample results

Table 1 shows the results from **`UCB_Plume`** processing for the flow rate. Two oil jets are shown: 5cs and 1 cs. Figure 24 shows the salient steps during data processing for three selected flows. Depending of the choice of numerous parameters specified, the estimated flow rate from the video stream can either closely match that determined from the flow meter in the flow loop or deviate substantially.

Accomplishments

Implementation

The script for discharge rate calculation is outlined in Chart 2. After some minimal interaction from the operator, the software proceeds autonomously, and produces its final results on the screen and archives them permanently. The User's Manual of the software package is presented in Appendix 1. The listing of the script may be found in Appendix 2.

Algorithm development

Video stream or image sequences are processed at multiple steps for discharge rate estimation. The flow chart in Chart 1 highlights the procedure. Chart 2 shows the operation block diagram of the software. Chart 3 shows the screen shot of the process from start to end, showing the final output of the software; an estimate of the flow rate in various units. After some minimal interaction from the operator, the software proceeds autonomously, and produces its final results on the screen and archives them permanently. The User's Manual of the software package is presented in Appendix 1. The listing of the script may be found in Appendix 2.

Table 1 shows our most recent runs of the software on our experiments, and Table 2 on OMSETT experiments at the highest flow rate. As can be seen from the end results in the last columns, the software is able to estimate the flow rates within 20% accuracy which is deemed sufficient for the first-response purposes in the field.

Presentation at BSEE

We made a presentation at BSEE in Sterling, Virginia, on March 24, 2016. Franklin Shaffer of NETL and Ömer Savaş of UCB made presentations to the cognizant officers of the program at BSEE. It was well received. Slides of this presentation may be found in Appendix 3.

Hardware requirements for field implementation

A high end WINDOWS laptop computer as the operating platform and a high speed, high resolution monochrome camera as imaging device are sufficient for acquiring flow images and running the software in the field over a matter of minutes. Suggested resources are:

- A high end laptop computer (32GB RAM, Intel i7 core), Windows/DOS operating system
- A high speed, high resolution monochrome camera (3000+ fps, 2Mpix+, e.g. MotionPro Y3). Framing rate must be high enough to record the life span of the inertial eddies at the edge of the jet. Imaging through polarizing filter may be useful to remove reflections at interfaces
- Continuous unidirectional polarized illumination along the jet axis, we suggest self-contained LED lighting which is now ubiquitous. Alternatively, a collimates halogen light source may be used
- Optional third party software for visualization (Adobe CS Professional, IDL, MATLAB, VLC …)

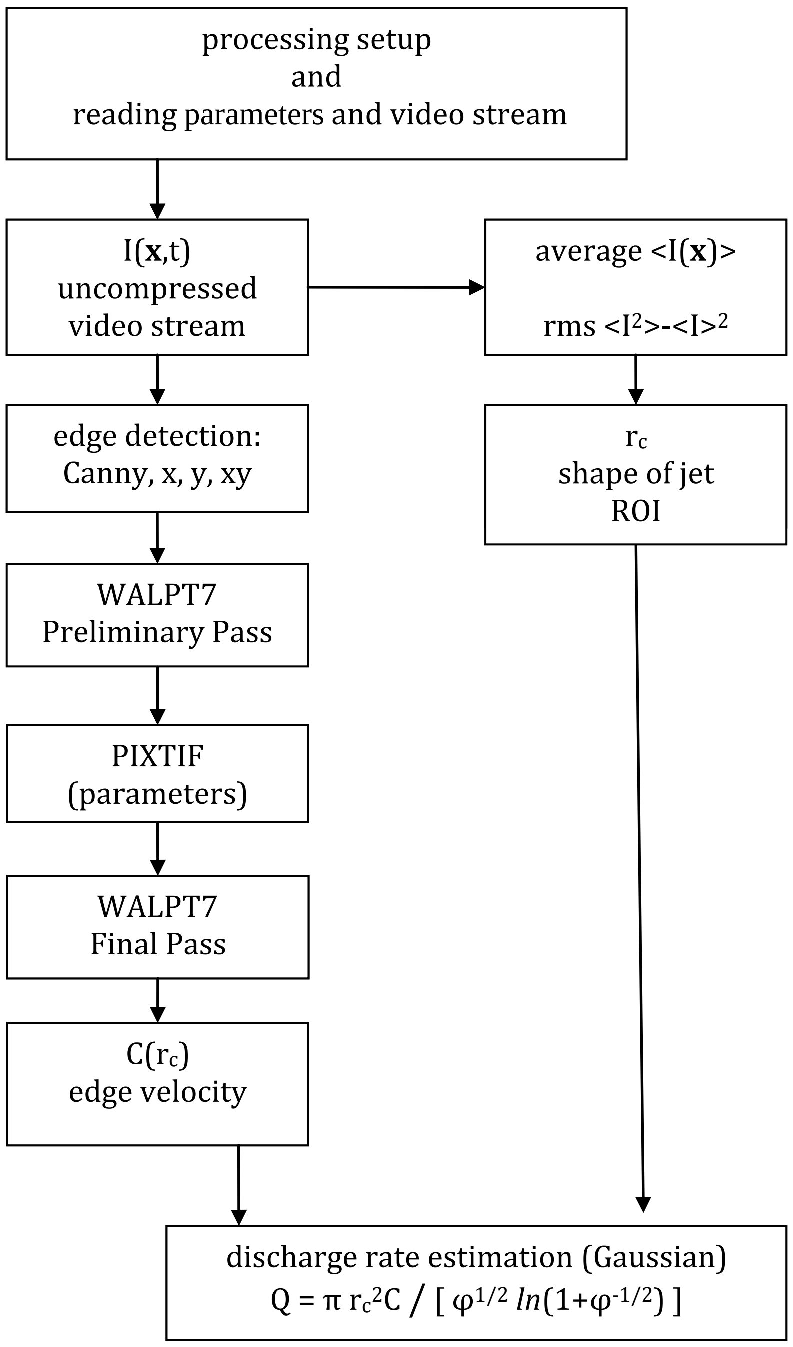

processing setup and reading parameters and video stream
$I(\mathbf{x},t)$ uncompressed video stream
average $\langle I(\mathbf{x})\rangle$
rms $\langle I^2\rangle - \langle I\rangle^2$
edge detection: Canny, x, y, xy
r_c shape of jet ROI
WALPT7 Preliminary Pass
PIXTIF (parameters)
WALPT7 Final Pass
$C(r_c)$ edge velocity
discharge rate estimation (Gaussian)
$Q = \pi r_c^2 C \,/\, [\, \varphi^{1/2}\, ln(1+\varphi^{-1/2})\,]$

Chart 1. **`UCB_Plume`** flow chart.

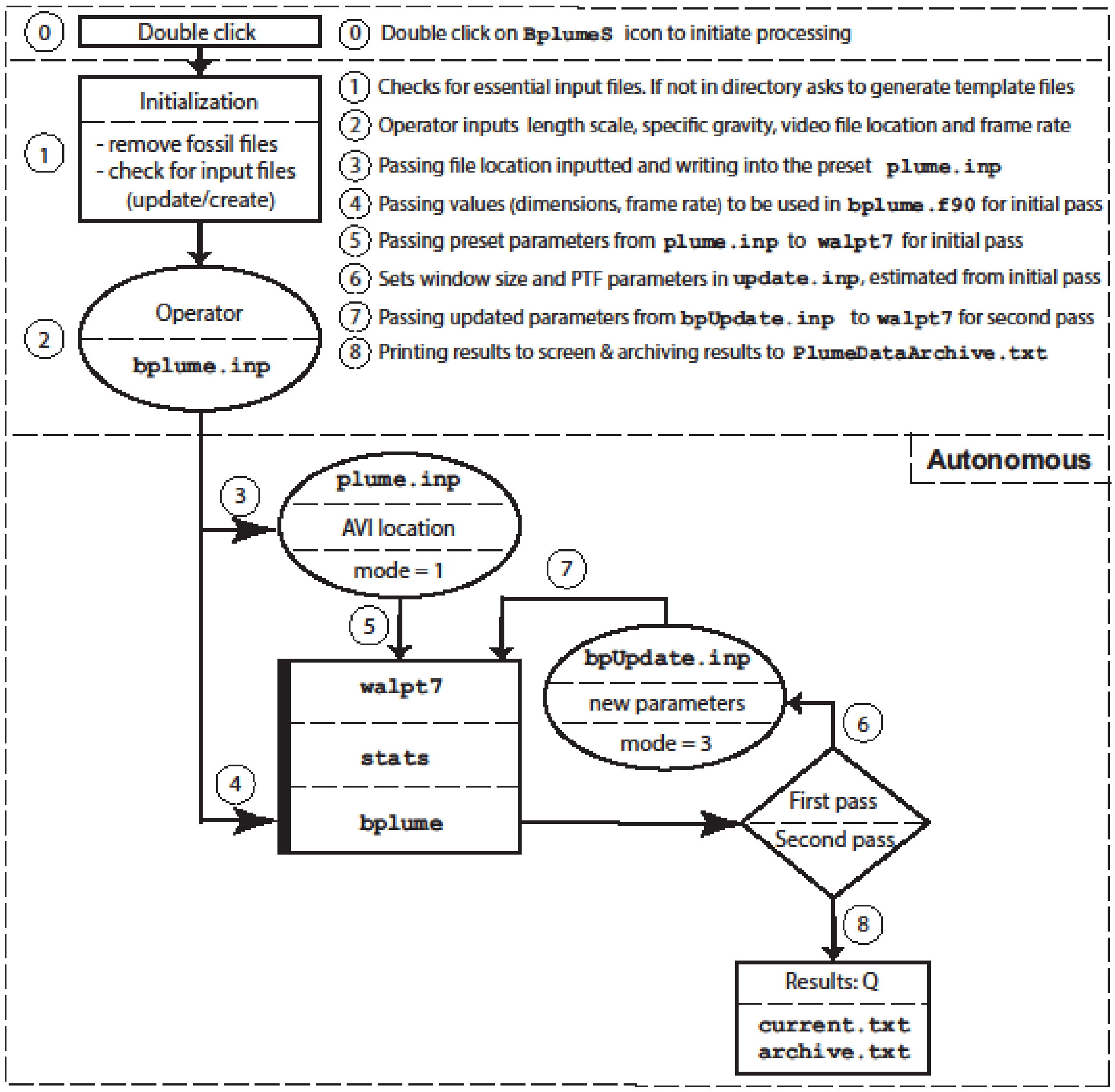
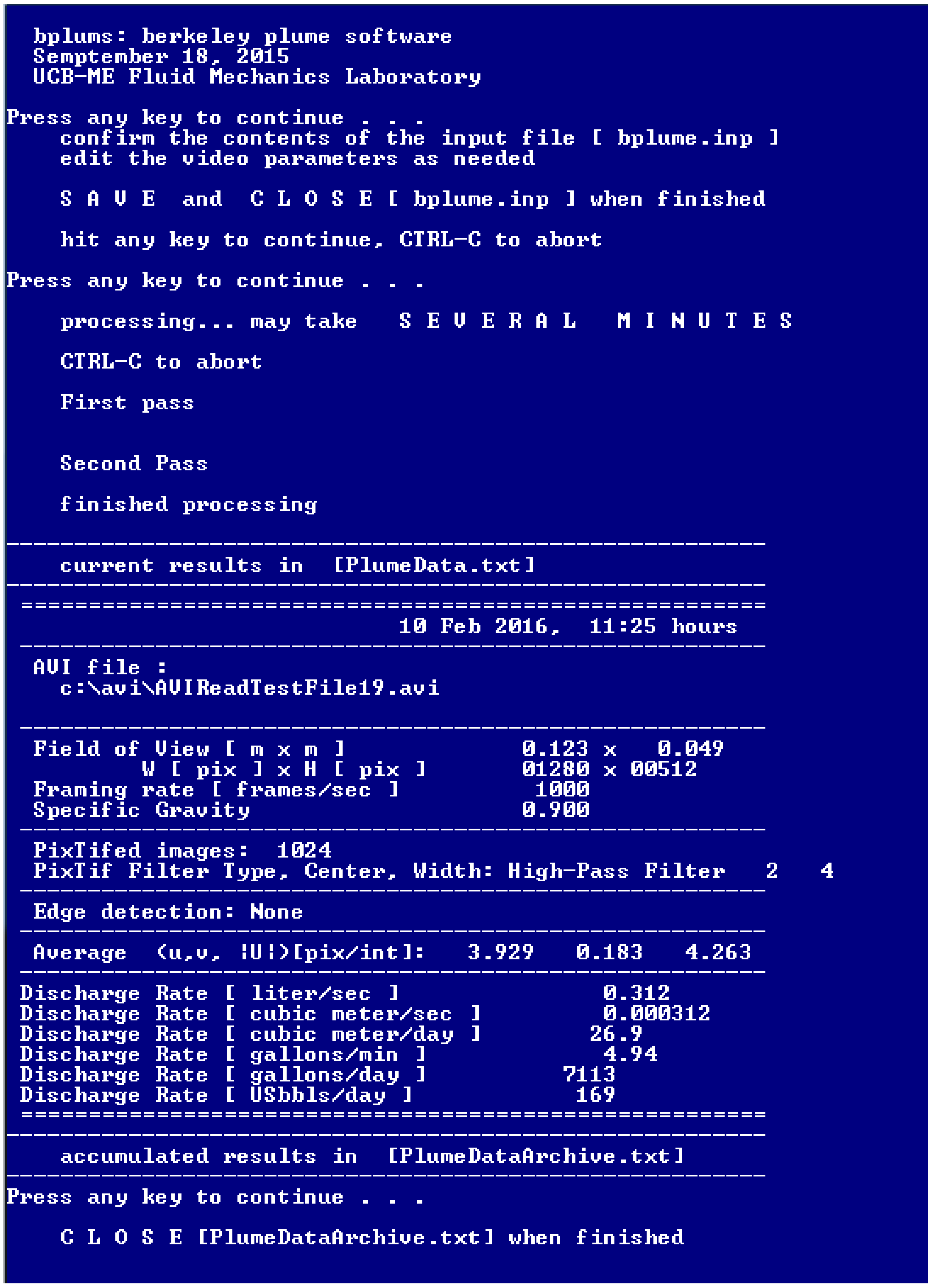

0
Double click
0 Double click on BplumeS icon to initiate processing
1
Initialization
- remove fossil files
- check for input files (update/create)
1 Checks for essential input files. If not in directory asks to generate template files
2 Operator inputs length scale, specific gravity, video file location and frame rate
3 Passing file location inputted and writing into the preset plume.inp
4 Passing values (dimensions, frame rate) to be used in bplume.f90 for initial pass
5 Passing preset parameters from plume.inp to walpt7 for initial pass
6 Sets window size and PTF parameters in update.inp, estimated from initial pass
7 Passing updated parameters from bpUpdate.inp to walpt7 for second pass
8 Printing results to screen & archiving results to PlumeDataArchive.txt
2
Operator
bplume.inp
Autonomous
3
plume.inp
AVI location
mode = 1
7
5
walpt7
stats
bplume
bpUpdate.inp
new parameters
mode = 3
6
4
First pass
Second pass
8
Results: Q
current.txt
archive.txt

Chart 2. Operation of software package: **`UCB_Plume`**.



```
  bplums: berkeley plume software
  Semptember 18, 2015
  UCB-ME Fluid Mechanics Laboratory

Press any key to continue . . .
    confirm the contents of the input file [ bplume.inp ]
    edit the video parameters as needed

    S A V E  and  C L O S E [ bplume.inp ] when finished

    hit any key to continue, CTRL-C to abort

Press any key to continue . . .

    processing... may take   S E V E R A L   M I N U T E S

    CTRL-C to abort

    First pass


    Second Pass

    finished processing

--------------------------------------------------------
    current results in  [PlumeData.txt]
--------------------------------------------------------
 =======================================================
                            10 Feb 2016,  11:25 hours
 -------------------------------------------------------
  AVI file :
    c:\avi\AVIReadTestFile19.avi

 -------------------------------------------------------
  Field of View [ m x m ]                0.123 x   0.049
          W [ pix ] x H [ pix ]          01280 x 00512
  Framing rate [ frames/sec ]             1000
  Specific Gravity                       0.900
 -------------------------------------------------------
  PixTifed images:  1024
  PixTif Filter Type, Center, Width: High-Pass Filter   2   4
 -------------------------------------------------------
  Edge detection: None
 -------------------------------------------------------
  Average  <u,v, |U|>[pix/int]:   3.929   0.183   4.263
 -------------------------------------------------------
 Discharge Rate [ liter/sec ]                 0.312
 Discharge Rate [ cubic meter/sec ]           0.000312
 Discharge Rate [ cubic meter/day ]          26.9
 Discharge Rate [ gallons/min ]               4.94
 Discharge Rate [ gallons/day ]            7113
 Discharge Rate [ USbbls/day ]              169
 =======================================================
--------------------------------------------------------
    accumulated results in  [PlumeDataArchive.txt]
--------------------------------------------------------
Press any key to continue . . .

    C L O S E [PlumeDataArchive.txt] when finished
```

Chart 3. Screen shot **`UCB_Plume`**.

Fluid	Frame rate	Flow rate	Re	error
	(fps)	(gpm)	x 10^4	%
water	500	1.02	0.45	-3
water	500	4.10	2.67	-23
water(*)	500	7.40	4.83	-18
1cs oil	1000	4.14	2.70	-9
1cs oil(*)	1000	5.52	3.60	-7
5cs oil(*)	1000	3.45	0.45	-11
5cs oil	1000	5.80	0.76	-2
5cs oil	1000	6.90	0.90	-15

Table1. Summary of Berkeley experiments. WALP& mode (3.2), PIXTIF filter parameters set at (1, 2, 8). (*)- See figure 23 for images.

WALP Mode,Passes	PIXTIF Filter Parameters	u	v	Calculated Flow Rate	error
				(gpm)	
3.2	1.3.6	0.91	0.047	595	-30%
3.2	1.4.8	0.998	0.072	653	-23%
3.2	1.5.10	1.058	0.058	692	-19%
3.2	1.6.12	1.069	0.059	700	-18%
3.2	1.7.14	1.065	0.059	697	-18%
3.2	1.8.16	1.057	0.056	691	-19%
3.2	1.9.18	1.044	0.048	683	-20%
3.2	1.10.20	1.032	0.869	675	-21%

Table 2. PIXTIF exercise on the OHMSETT Test 18: 850 gallons per minute, video frame rate 500 frames per second.

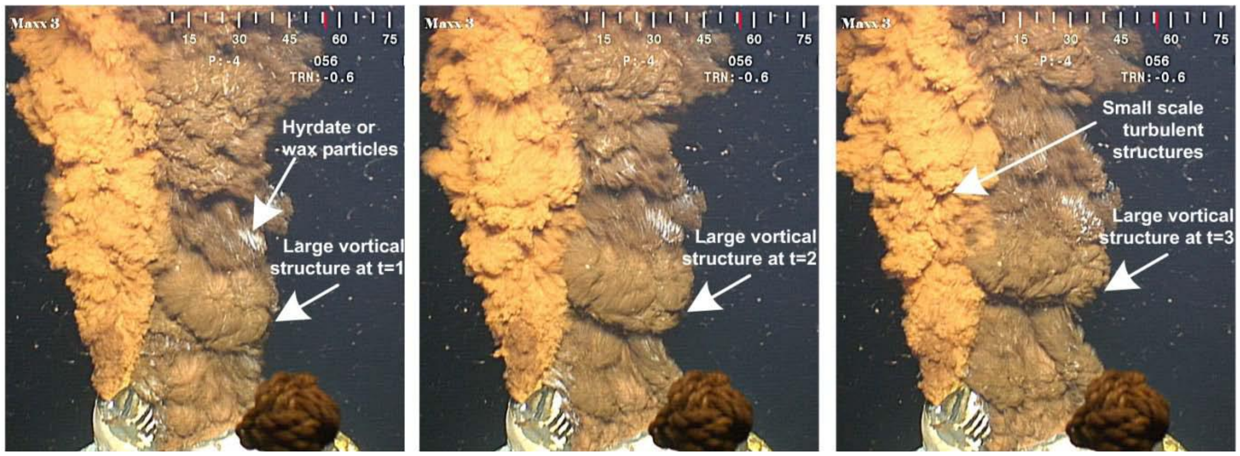

Maxx 3
15 30 45 60 75
P: -4
056
TRN: -0.6
Hyrdate or wax particles
Large vortical structure at t=1
Large vortical structure at t=2
Small scale turbulent structures
Large vortical structure at t=3

Figure 1. Examples of visible features tractable in the flow direction on the Deepwater Horizon oil leak jet.

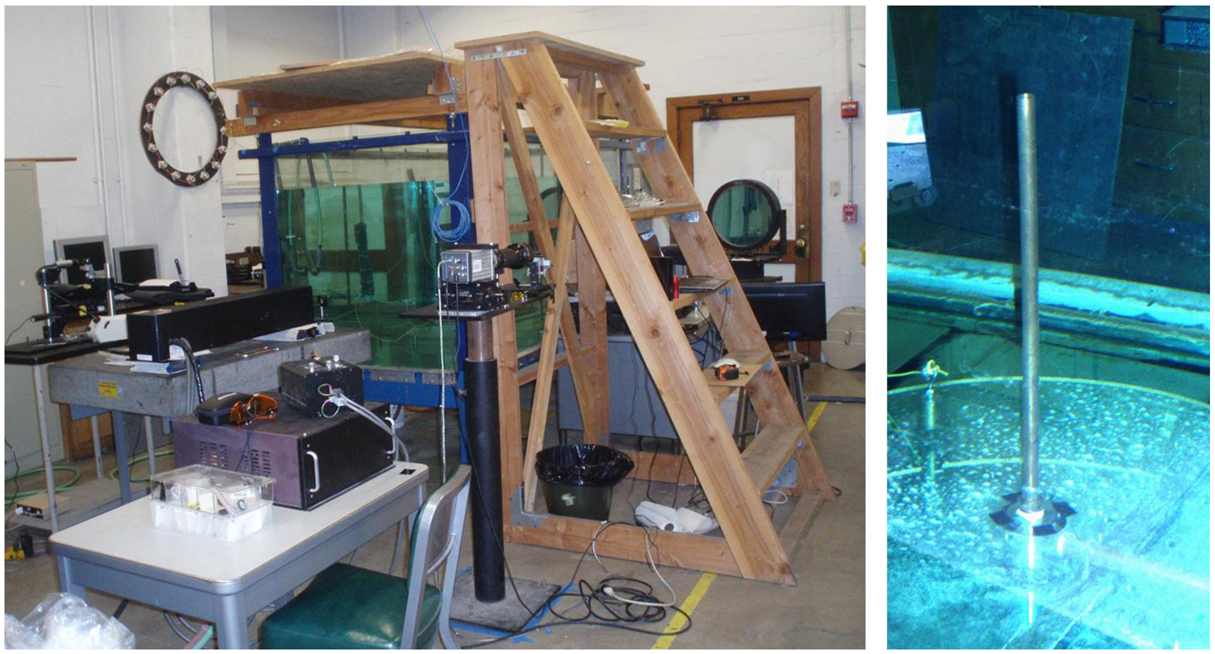


Figure 2. Overall view of the final setup (left) and the ½ inch diameter, 16 inches long discharge tube (right).

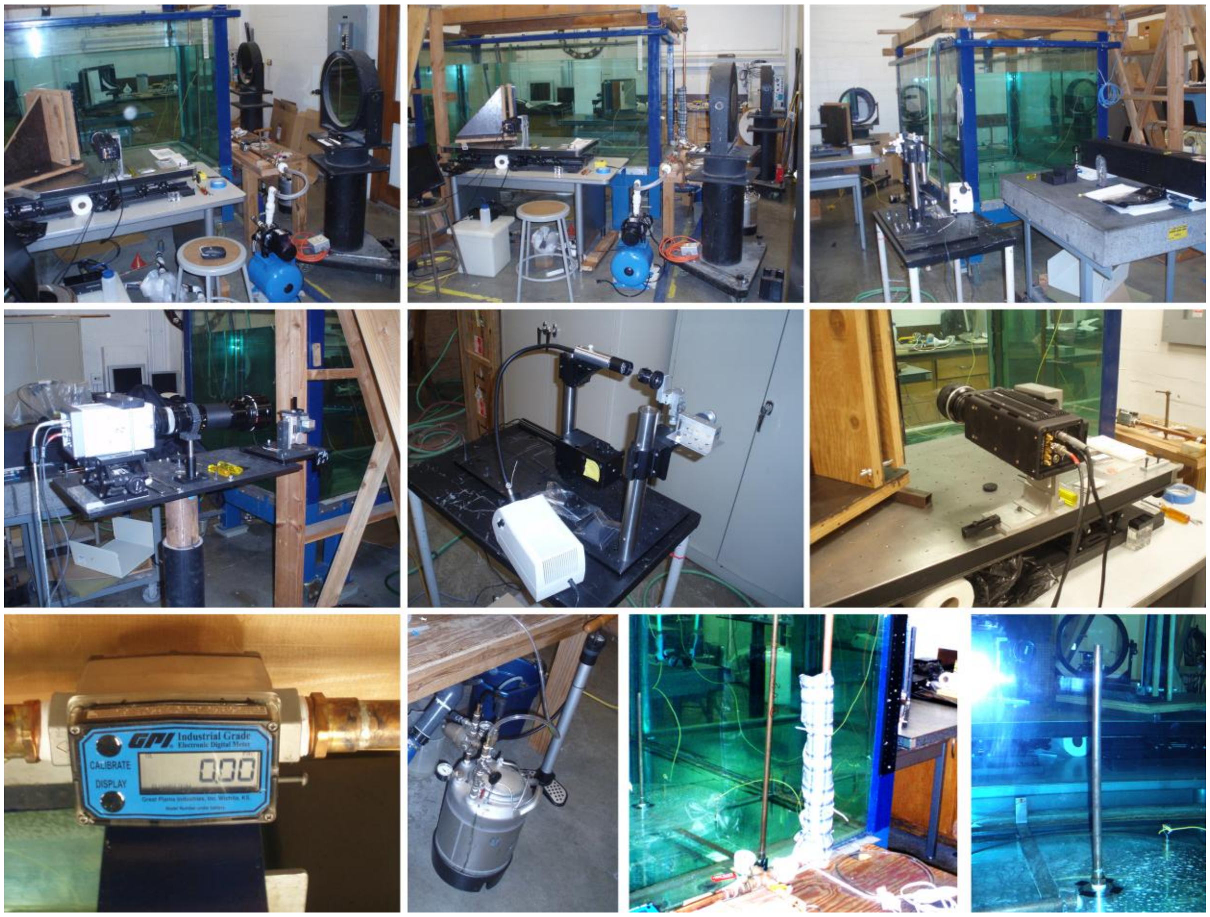


Figure 3. A mosaic of the components of the final setup. Top row: three general views showing the water tank, optical layout and its major components. Middle row: schlieren camera, shclieren light source and optics, and PIV/FV camera, Bottom row: turbine flow meter, pressurized dye tank, heater, and discharge tube.

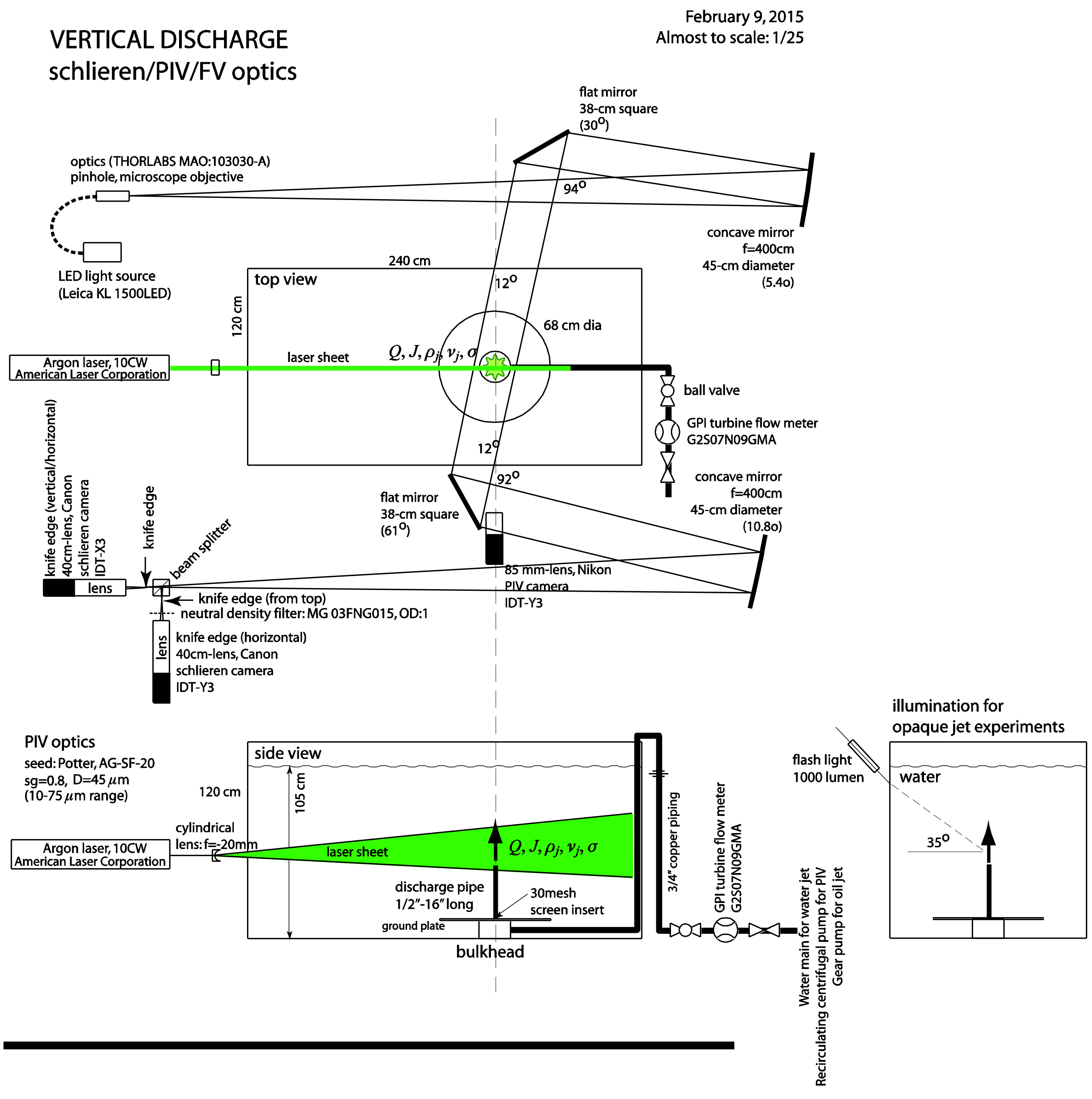


Figure 4. The final configuration of the experimental setup showing the schlieren/shadowgraph optical layout (top), PIV, and FV optical layout for vertical jet discharge experiments. The schlieren/shadowgraph light beam path is skewed 12 degrees to allow for normal clear access for PIV and FV cameras. . Simultaneous horizontal/vertical knife edge schlieren layout is at the left-center of the layout. Illumination scheme for fluorescent-opaque jet experiments is on the lower-right.

Figure 5. Schlieren/shadow photographs of 5 cs, 0.92 g/cm^3 n=1.49 oil discharging horizontally into water tank.

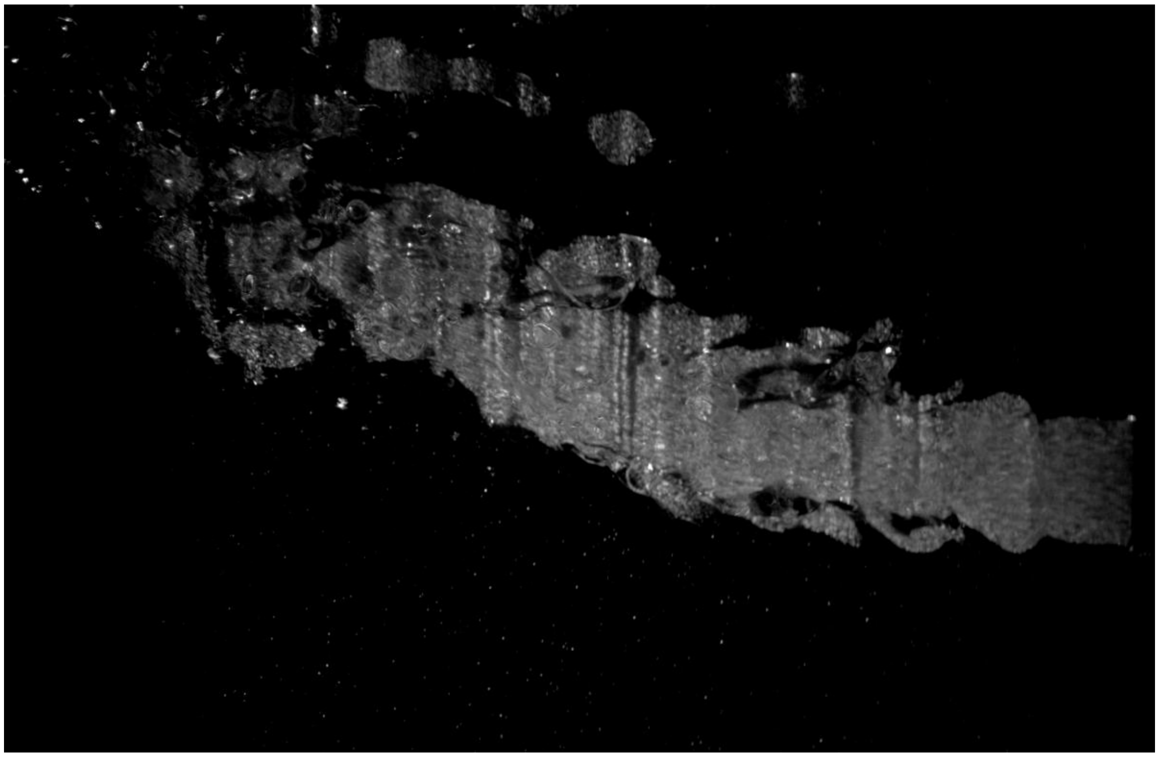


Figure 6. PIV image of 5 cs, 0.92 g/cm^3 n=1.49 oil discharging horizontally into water tank. The flow is made visible by seeding the oil and water with silver coated hallow ceramic spheres,

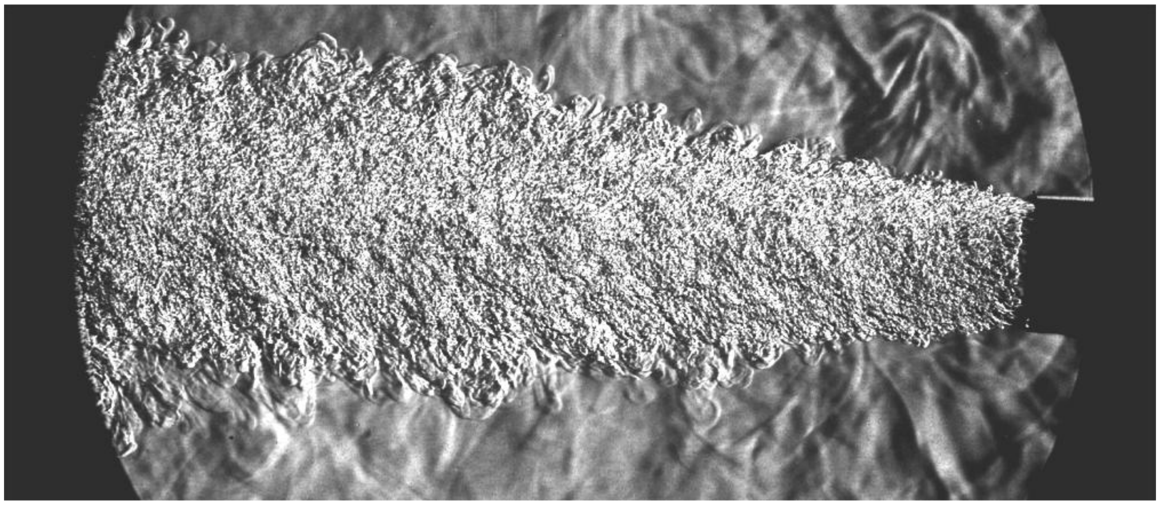


Figure 7. Schlieren photograph of water discharging vertically up into tank. The flow is made visible by slightly heating the discharge pipe exterior to the tank.

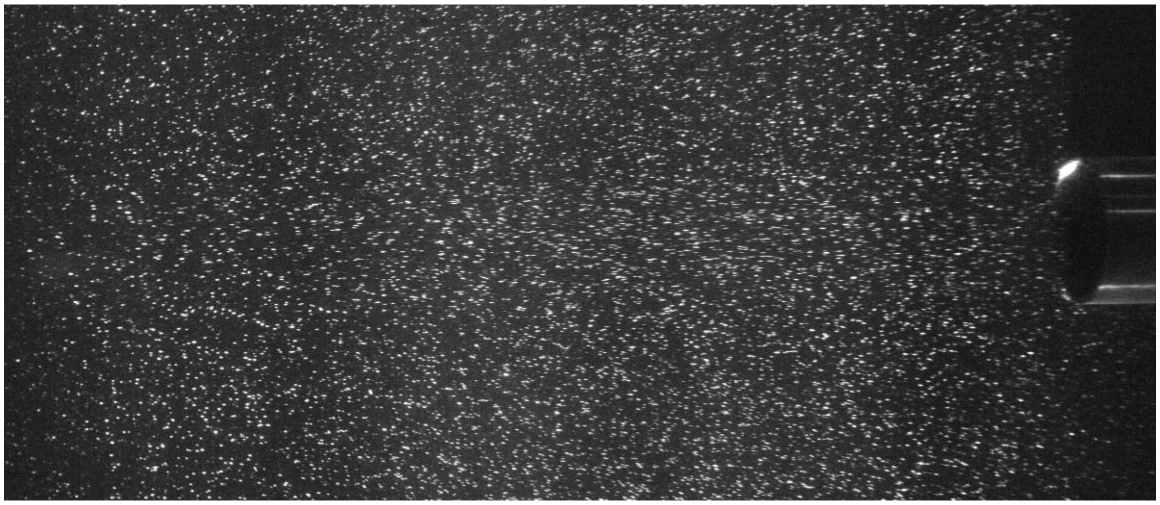


Figure 8. PIV image of water discharging vertically up into tank. The flow is made visible by seeding the water with silver coated hallow ceramic spheres.

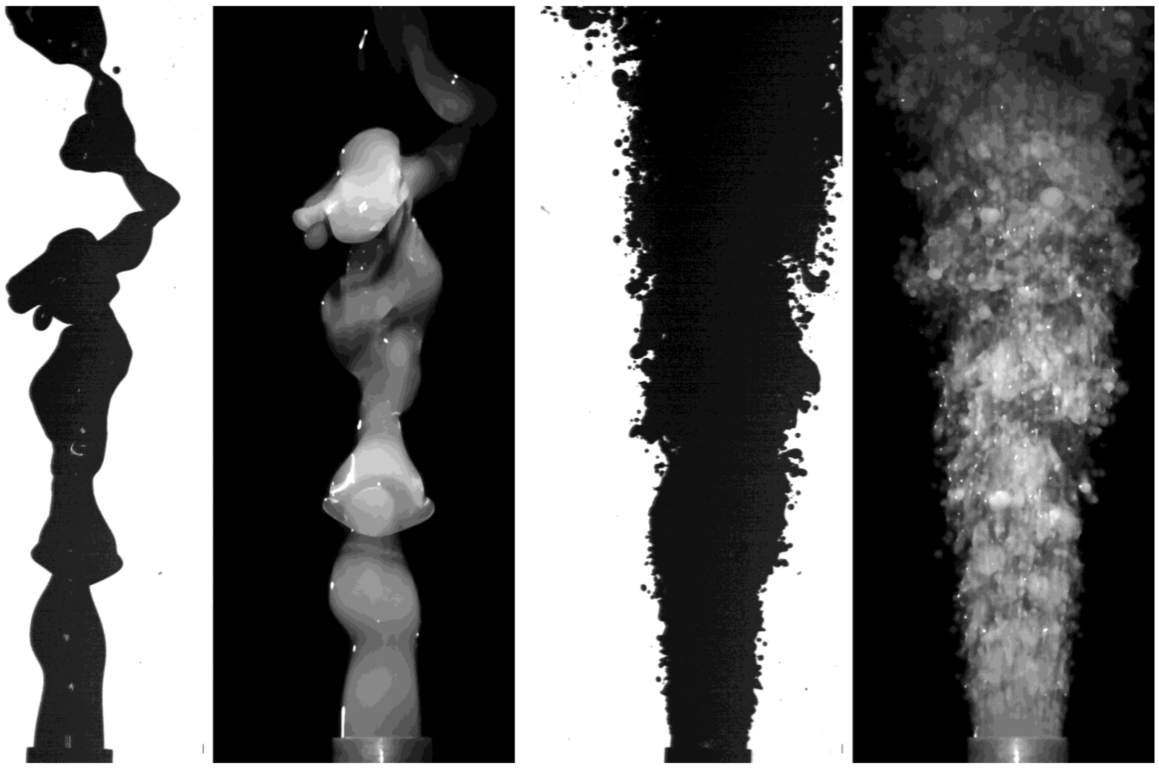


Figure 9. A mosaic of images of vertical oil jets. Left pair: 5cs silicone oil at 45cc/s (Re=900); Right pair: 1 cs silicone oil at 360 cc/s (Re=35,000). In each pair, the left image (white background) is a shadowgraph and the right one (dark background) a direct image of the exterior of the oil jets which is marked with oil soluble fluorescent dye. The image pairs are simultaneous, even though the magnifications are slightly larges for the direct images (1 versus 1.16).

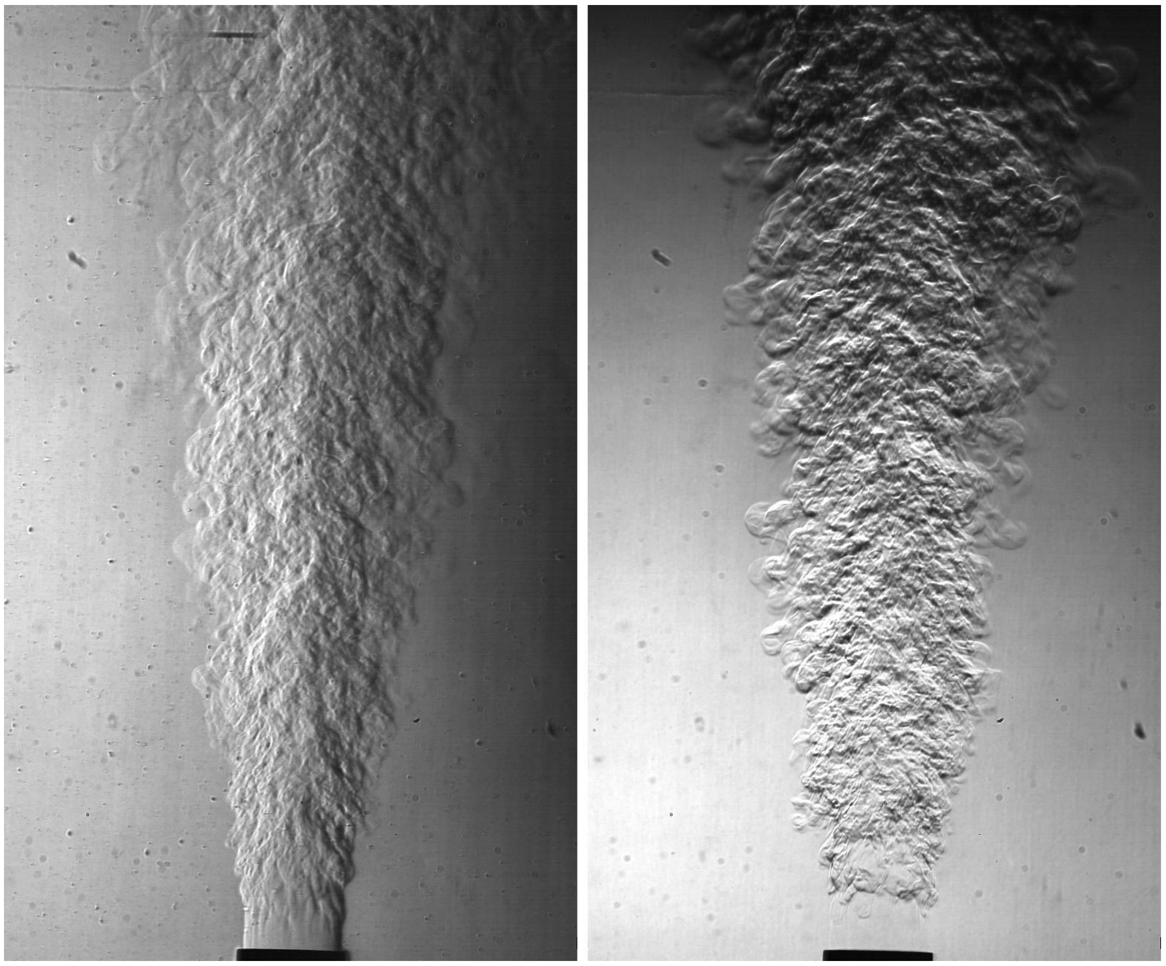


Figure 10. Sample flow images at Q=1.0 gallon/min (63 cm^3/s): schlieren with vertical knife edge and horizontal knife edge. Images 2048/4096 of the video streams are shown. The images are not simultaneous. Both images are 85 mm x137 mm (768x1280 pixels).

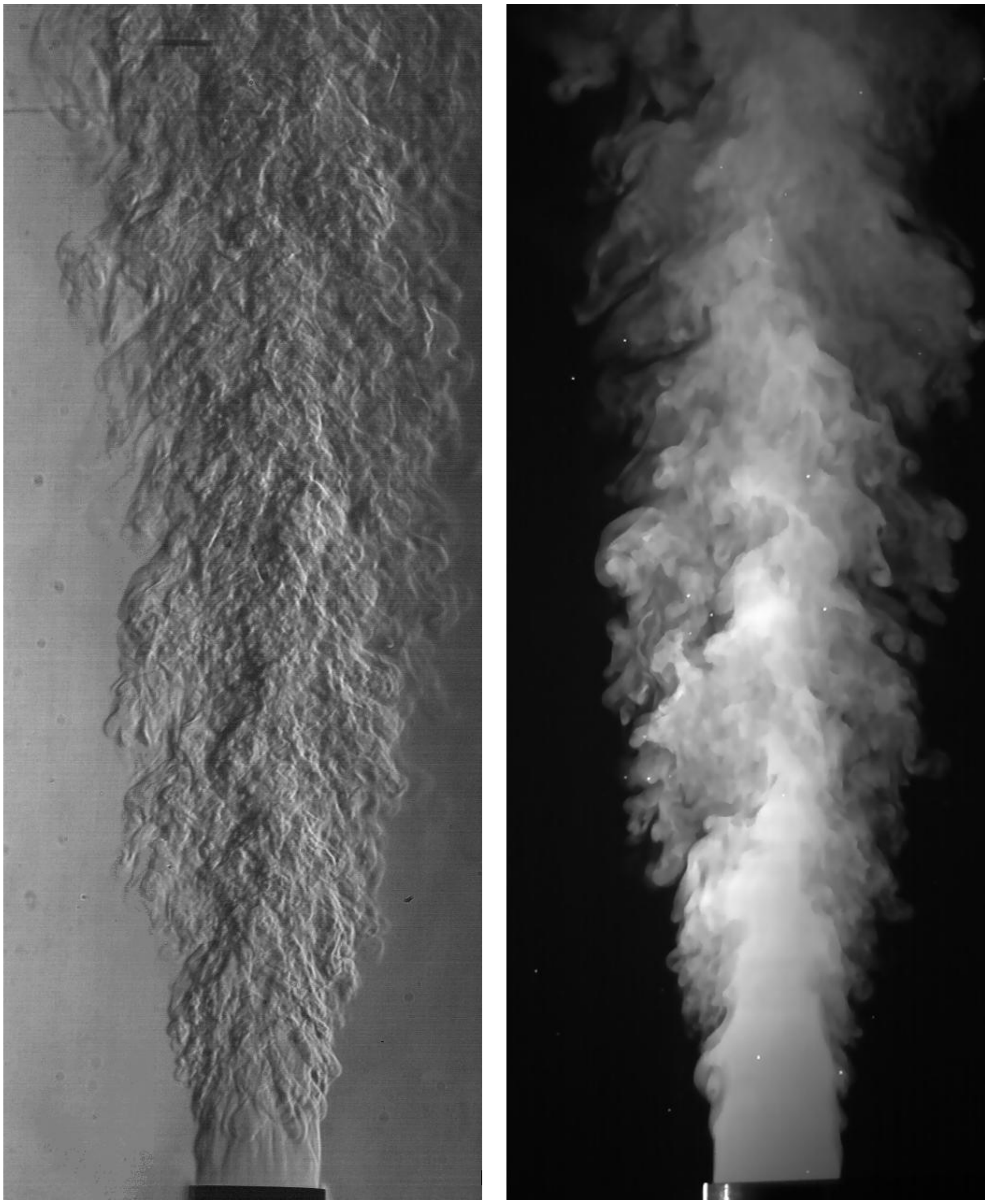


Figure 11. Sample flow images at Q=1.0 gallon/min (63 cm^3/s): simultaneous schlieren and cross-sectional fluorescent dye images. Images 2048/4096 of the video streams are shown. Left image: 56.5mmx137mm, Right image: 52mmx132mm. Both images 512x1280 pixels.

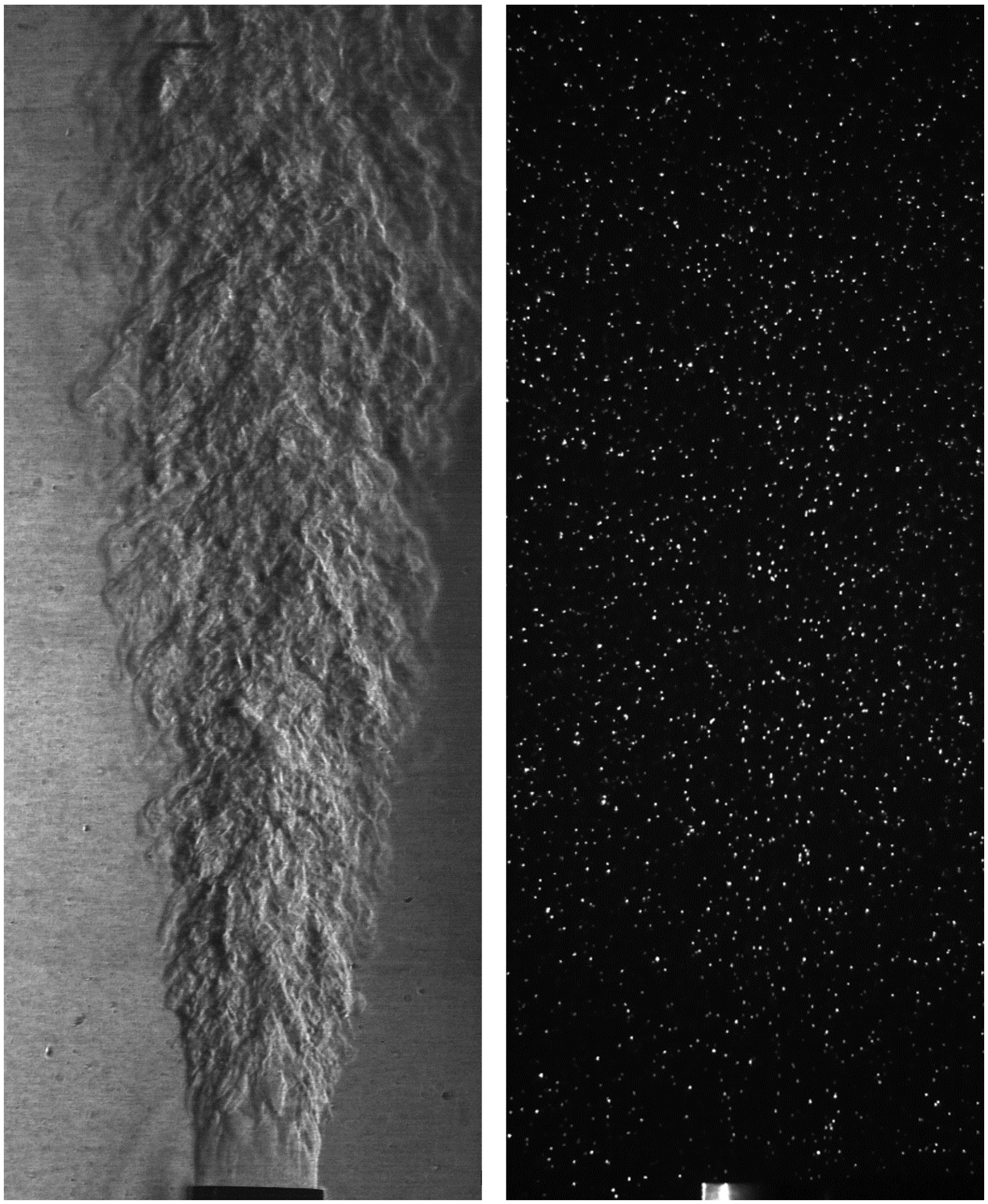


Figure 12. Sample flow images at Q=1.0 gallon/min (63 cm^3/s): simultaneous schlieren and PIV images, vertical knife edge. Images 2048/4096 of the video streams are shown. . Left image: 56.5mmx137mm, Right image: 52mmx132mm. Both images 512x1280 pixels.

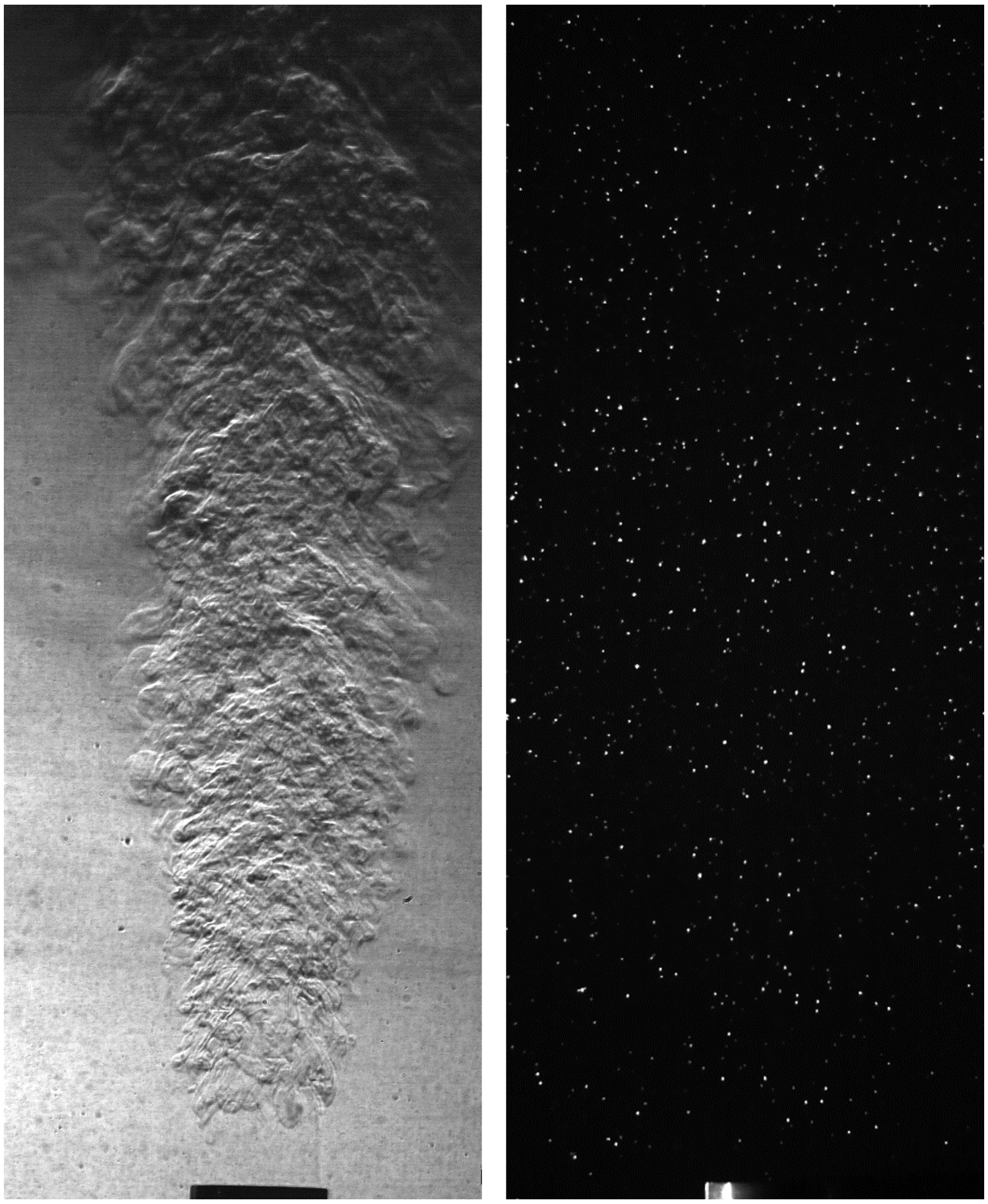


Figure 13. Sample flow images at Q=1.0 gallon/min (63 cm^3/s): simultaneous schlieren and PIV images horizontal knife edge. Images 2048/4096 of the video streams are shown. . Left image: 56.5mmx137mm, Right image: 52mmx132mm.

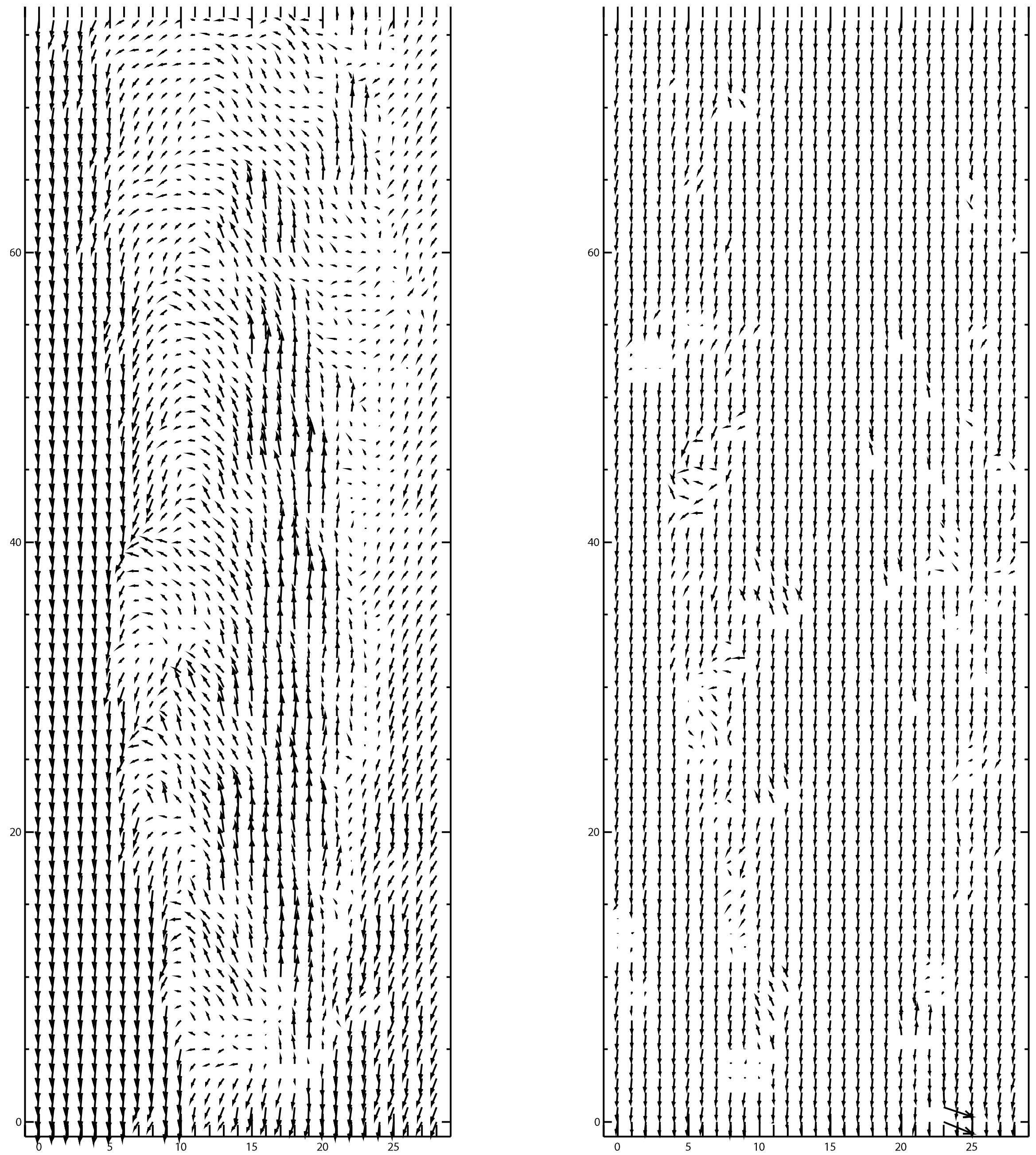
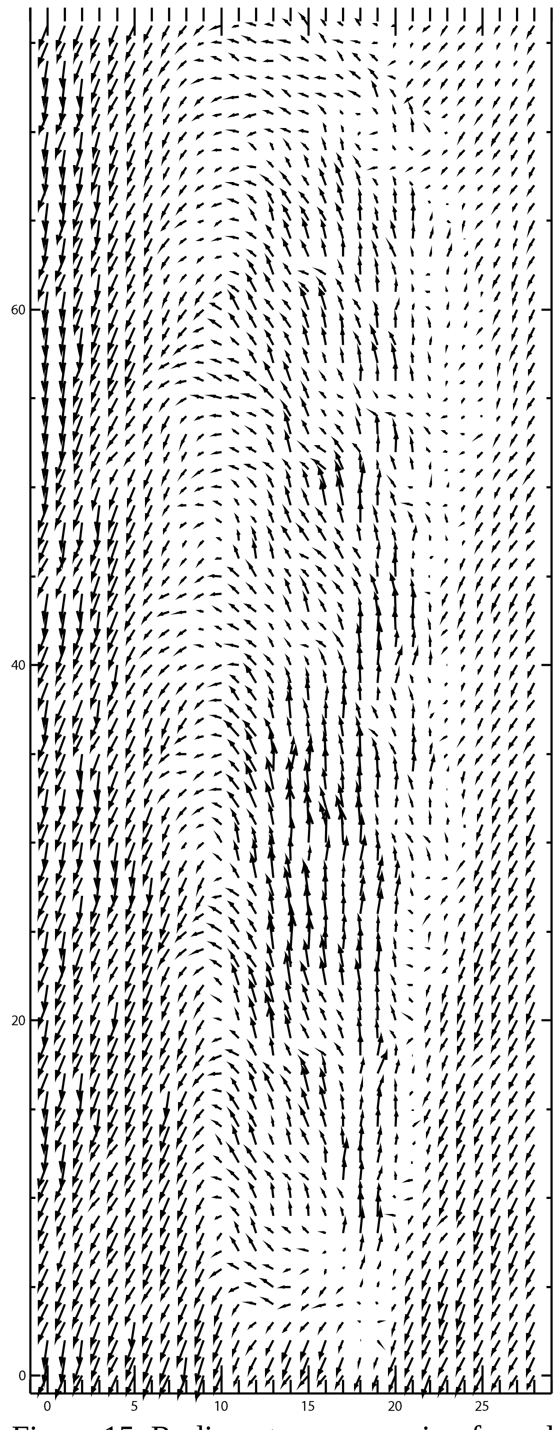


Figure 14. Rudimentary processing for velocity vector field of image pairs corresponding to those in figure 11. Right: schlieren images, Left: cross-sectional dye images. Vectors indicate only their relative magnitudes in each frame.



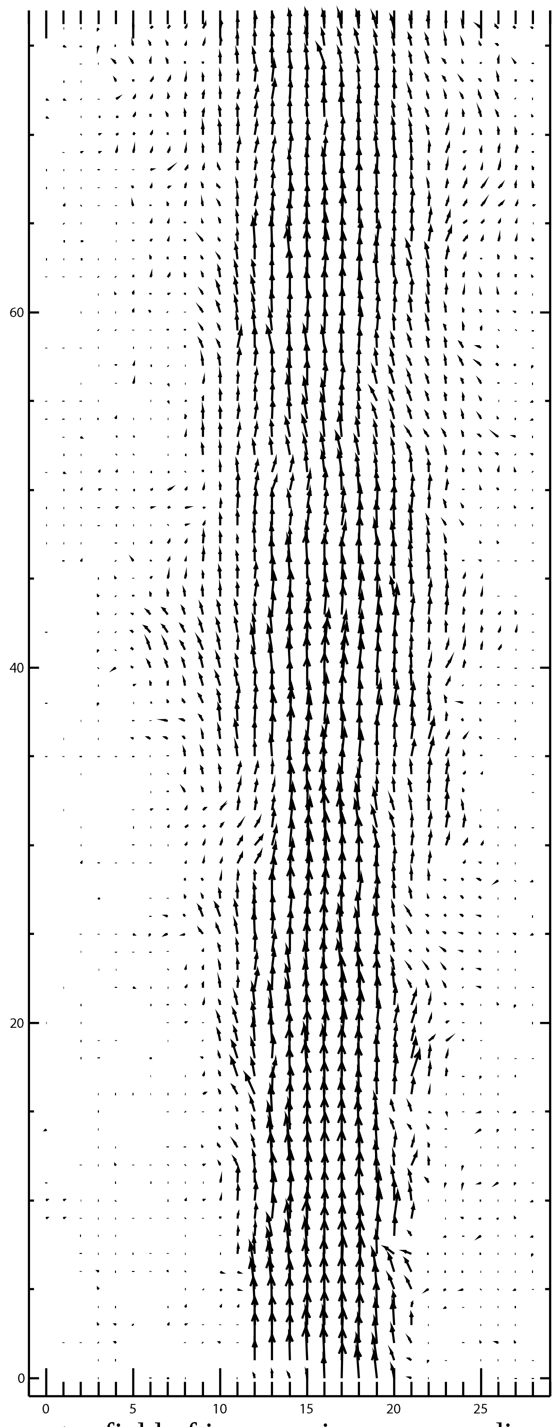


Figure 15. Rudimentary processing for velocity vector field of image pairs corresponding to those in figure 13. Right: schlieren images, Left: cross-sectional particle images. Vectors indicate only their relative magnitudes in each frame.

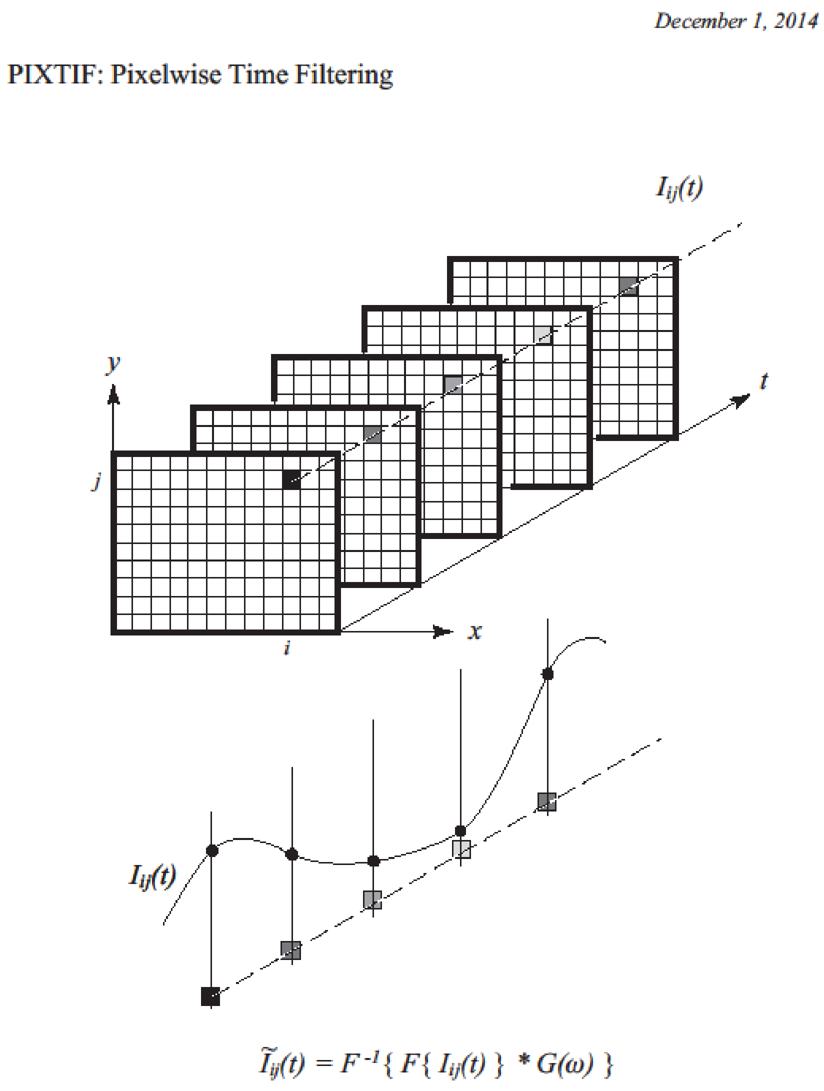

December 1, 2014
PIXTIF: Pixelwise Time Filtering
$I_{ij}(t)$
y
t
j
x
i
$I_{ij}(t)$
$\tilde{I}_{ij}(t) = F^{-1}\{ F\{ I_{ij}(t) \} * G(\omega) \}$

Figure 16. Pixelwise time filtering, PIXTIF, sketched.

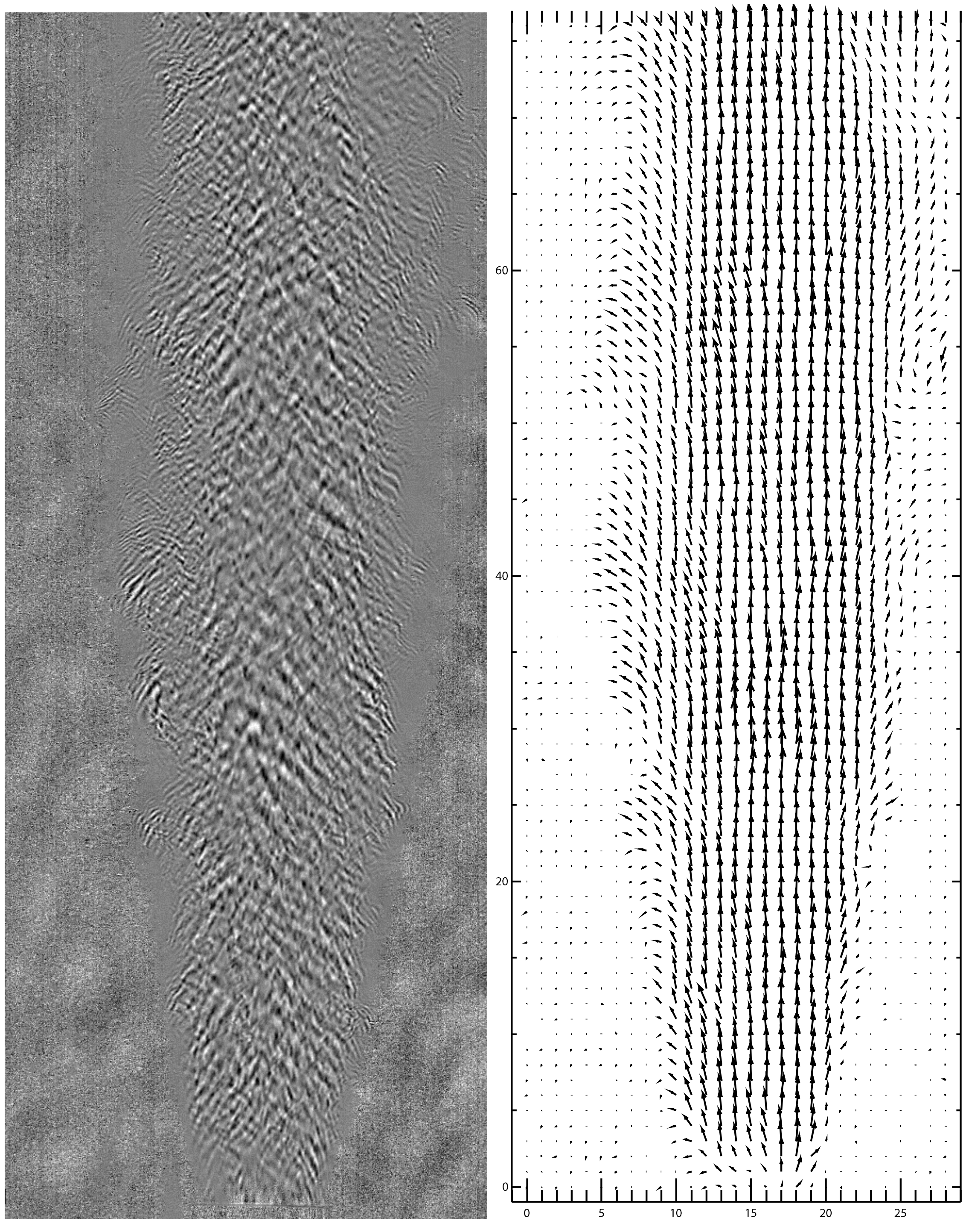


Figure 17. PIXTIF'ed image corresponding to the schlieren image in figure 13, and the velocity vector field determined from the corresponding image pair. Vectors indicate only their relative magnitudes. Image on the left is 56.5mmx137mm.

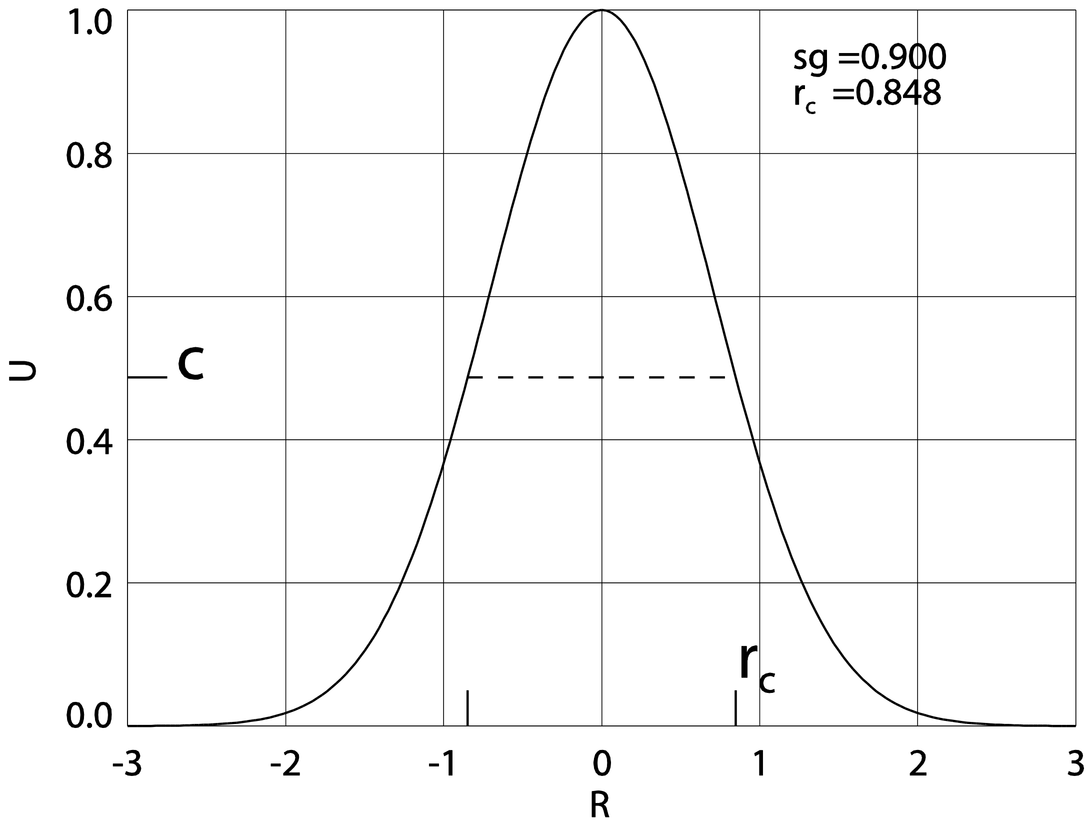


Figure 18. Schematic of a jet and symbol definitions. . Gaussian mean velocity profile, interface r_c, and velocity at the interface C, celerity.

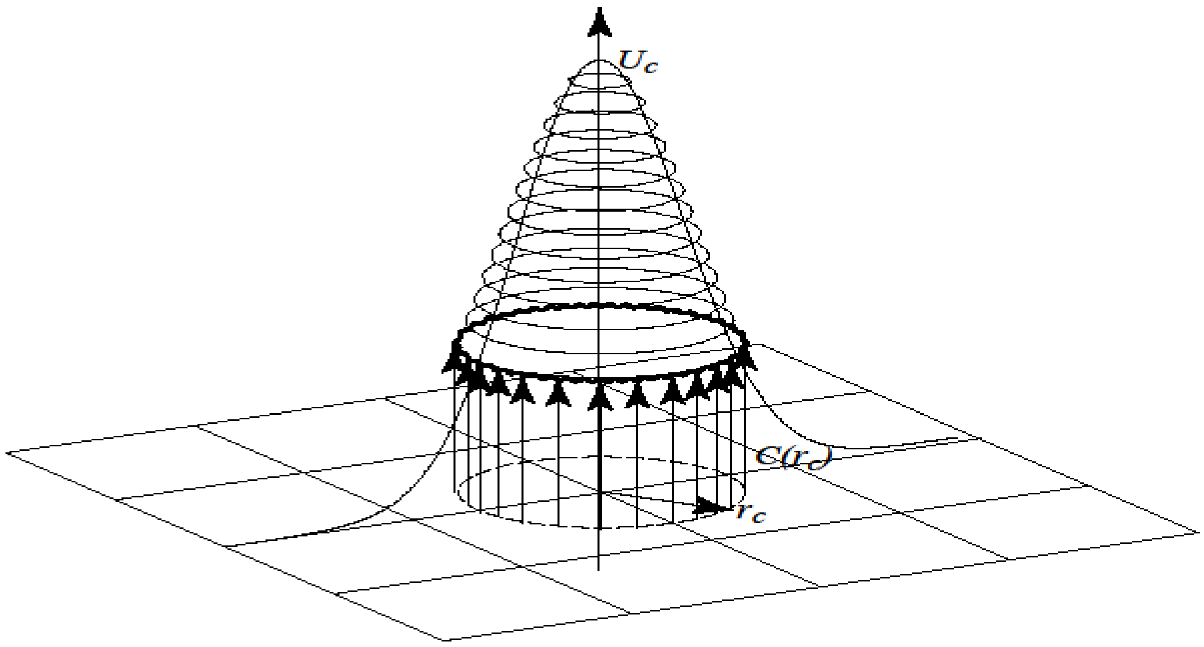


Figure 19. Jet edge velocity illustration.

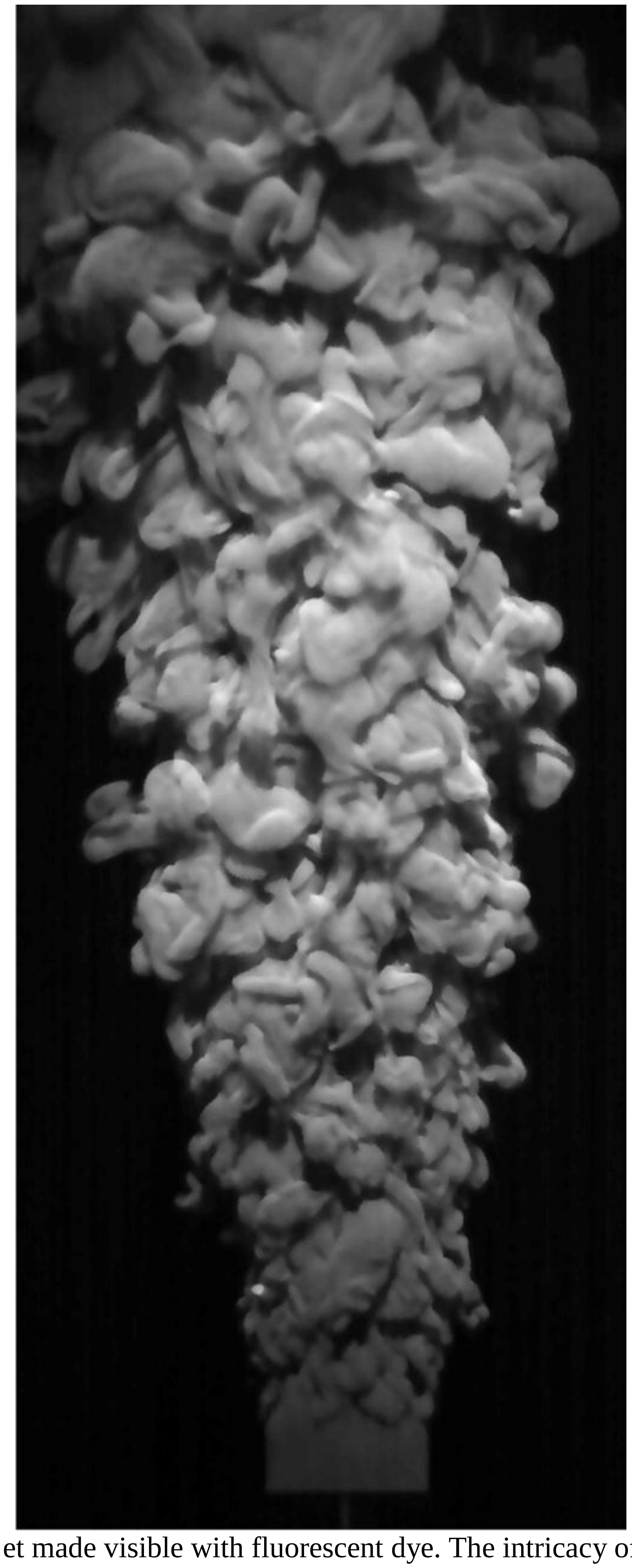


Figure 20. Water jet made visible with fluorescent dye. The intricacy of the jet-ambient fluid interface is clearly delineated. Q=1 gpm, Re=4500.

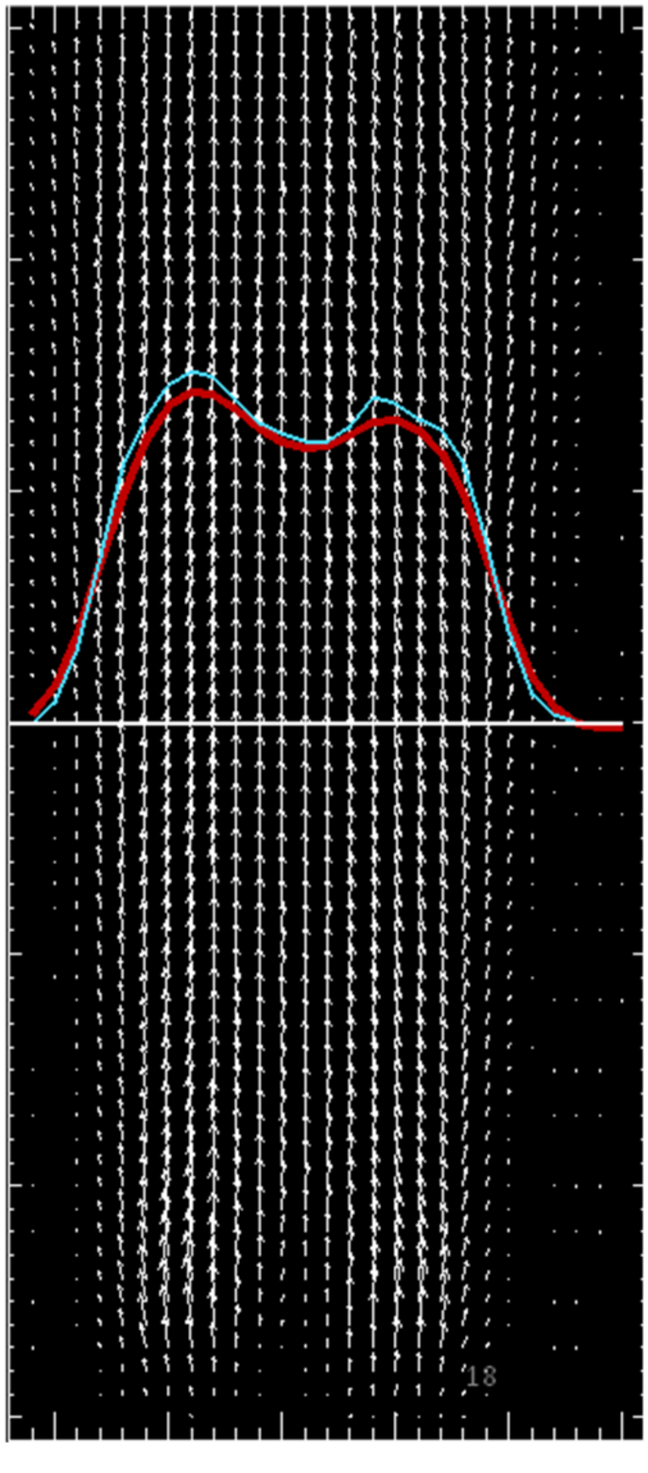


Figure 21. Typical jet edge velocity measurements.

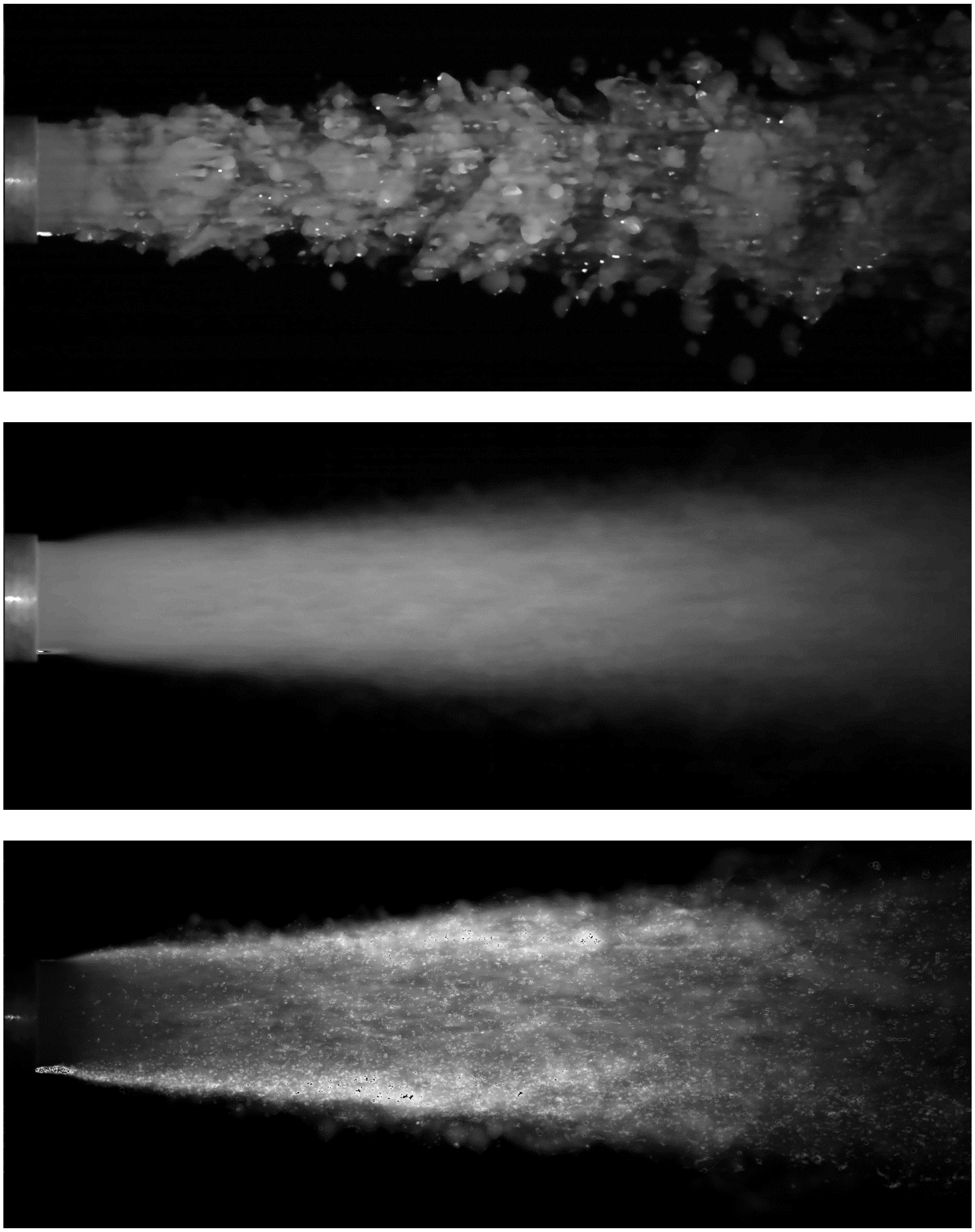


Figure 22. Images at the salient stages of **`UCB_Plume`** processing: 1cs silicone oil, 4.14 gpm, 1000 fps. (a) a raw image, (b) average image, and (c) rms image. (images (a) and (b) are enhanced for clarity)

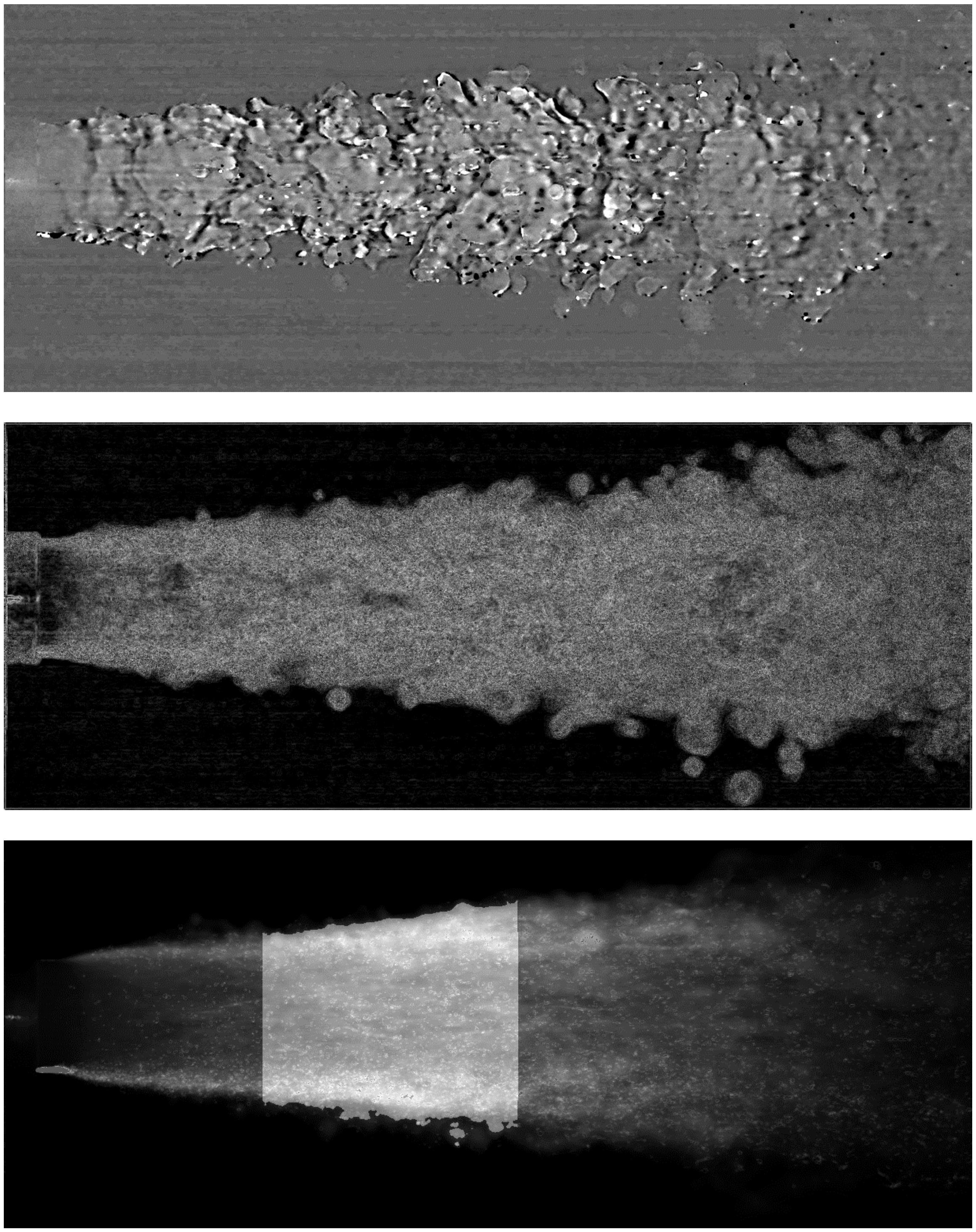


Figure 23. Images at the salient stages of image processing **`UCB_Plume`** processing: 1cs silicone oil, 4.14 gpm, 1000 fps. (a) PIXTIF'ed image (b) Canny Edged PIXTIF image, and (c) striped image, ROI. (images (a) and (b) are enhanced for clarity)

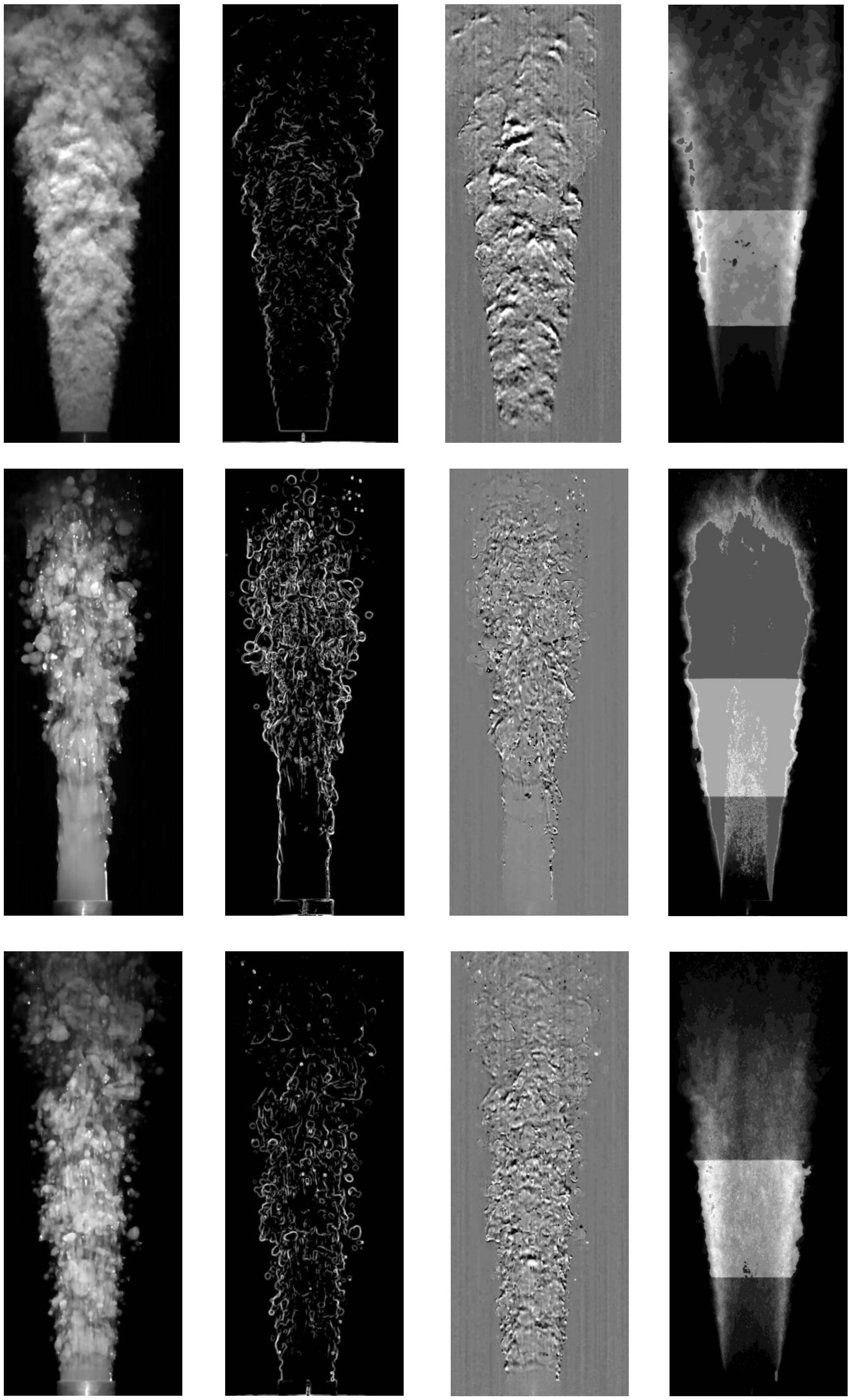


Figure 24.Top: Water jet Re=48,000, Middle: 1cs oil, Re=36,000, and Bottom : 5cs oil, Re=4,500 (see Table 1 for summary).

APPENDIX-1

UCB_Plume

USER’S MANUAL

UCB Plume

UCB Plume Analysis Tool
USER'S MANUAL

Version 1.01
June 2016

Fluid Mechanics Laboratory
140 Hesse Hall
Department of Mechanical Engineering
University of California, Berkeley, California, 94720-1740

Scope of Manual

This manual will describe the steps to use **`UCB Plume`** to estimate the flow rate of jet discharges. This involves both outlining the limitations of the software and describing its ideal application conditions. Methods for interpreting additional data that is calculated, as well as the visualization of results, will also be outlined. A list of common error messages and their solutions will be detailed as well. The name **`UCB Plume,`** UCB Plume Analysis Tool, and **`UCB_Plume.exe`** will be used interchangeably to refer to the software.

Point of Contact

`UCB Plume` was developed at UC Berkeley with support from National Energy and Technology Laboratories and Bureau of Safety and Environmental Enforcement.
Please contact with field use questions.

Contact:	Savaş, Ömer
Email:	savas@berkeley.edu
Phone:	(510) 642 - 5272
Contact:	Ibarra, Eric
Email:	ibarraeric89@gmail.com
Phone:	(323) 327 - 5868

Project References

This software was tested on laboratory experiments whose Reynolds number ranged from 4,500-550,000. These tested experiments were conducted at UC Berkeley and OHMSETT. On average, the flow rate estimation tends to be conservative by ~30%.

Glossary

The following terms are used as operator inputs and calculated results described in the manual.

ROI length/width:

The physical dimension in meters corresponding to the length/width of the image

Specific gravity (φ):

Density of the fluid comprising the Jet/Plume (ρ_{jet}) over the density of the quiescent fluid ($\rho_{ambient}$).

(i.e. Density of Crude oil / Density of Sea Water)

Frame Rate:

The rate at which the video was recorded by the camera, different from the playback rate that will be listed in the details of the AVI file.

Celerity (C):

The velocity of the trackable features. The mean center velocity of the trackable eddies along the visible surface of the opaque jet/plume.

Centerline Velocity (U_C):

The maximum velocity to be found at the center axis of the discharging fluid jet.

Algorithm Development

Cross Correlation: A measure of the similarity between two signals as a function of the delay between the two images. In our correlation, the signal is the intensity value in the image and the lag is the time interval elapsed between successive images.

Particle Image Velocity (PIV): A flow visualization technique that provides the instantaneous velocity measurements to be taken. For PIV, density neutral particles are seeded into a flow. The motion of the particles are used to calculate the velocity field of the flow.

Image Correlation Velocimetry (ICV): ICV performs a cross correlation of interrogation regions in consecutive images to measure the displacements of moving images. Using the delay between the images, a velocity is measured.

PixTif: A pixel-wise filtering done on time. Features in the flow move at a certain speed. At a given pixel, these features leave a distinct signature in the pixel's intensity history. By taking the intensity signal into the Fourier domain, features moving with a certain speed can be filtered with a convolution. We then apply a high-pass filter, and we define the filter center and width using the celerity and radius calculated during the first pass using a Stouhal number.

Parameter generation: During a run of `UCB_Plume.exe`, `WALPT7.exe` is called for two passes over the images. An initial pass is run with preset processing parameters and the user's flow parameters. This initial pass gives a first approximation of the velocities, length scales, and flow rates that are being considered. Based on the results, `WALPT7.exe` will output an updated input file with processing parameters adjusted. The updated input file will be used to run `WALPT7.exe` for the final pass, the results of which will be outputted.

Radius estimation: The root mean square (RMS) of the all the images processed is used to estimate the radius. The RMS of the image deck removes constant features in the ROI, while outlining the location of fluctuating intensity values, such as where moving features in the jet occur.

Celerity: The velocity of the surface features measured and is used as the celerity for the calculation. By using image correlation velocimetry, a velocity field of the jet is calculated, and an average of this field is taken about the region of calculation.

Discharge Rate estimation

Assumptions:

In the integration of the estimated jet profile these assumptions are made.

A *Gaussian velocity profile* of the jet that is axisymmetric is assumed. This is used to relate the observable feature's velocity to what is inside for flow rate calculations.

The velocity profile is evaluated at a distance of 3 pipe diameters downstream from the discharging orifice, which is in the *near field*. In this range the inertial terms are the same order of magnitude as the buoyant terms, allowing them to be neglected in the flow rate calculations.

It is assumed in the *near field* that the velocity of the surrounding fluid, U_2, is small in comparison to the celerity of the jet/plume, allowing $U_2 \approx 0$. The limit of this software is to where jets are not being affected by currents in the same order of magnitude as the jet/plume.

A derivation of the flow rate (Q) is as follows:

$$u(r)\,/U_c \;=\; \exp(\,-r^2/\sigma^2)$$

$$C\,/U_c \;=\; \exp(\,-r_c^2/\sigma^2)$$

$$U_c = \;(1+\varphi^{-1/2})\;C$$

$$\varphi \;=\; \frac{\rho_{jet}}{\rho_{ambient}}$$

$$\sigma^2 = \frac{r_c^2}{\ln(1+\varphi^{-1/2})}$$

$$Q = 2\pi\int_0^{r_c} u(r)\;r\;dr = \;\frac{\pi\; r_c^2\; C}{\varphi^{1/2}\ln(1+\varphi^{-1/2})}$$

Hardware Requirements

A high end Windows laptop computer as the operating platform and a high speed, high resolution monochrome camera for uncompressed video recording are the hardware requirements.

Capable Camera:

A high speed, high resolution camera (2000+ fps, 2Mpix+). Successive images in a video stream must be recorded within the life span of flow structure at the edge of the jet. Camera should use a polarizing filter to remove reflection at jet/environment interfaces. The Y models and X models by Integrated Design Tools Inc. have been used in the laboratory setting for these higher speed flow experiment, and have performed well. An additional benefit, the capturing software provided by IDT Inc. can write AVI files using the Basic Windows bitmap format which is required by `UCB_Plume.exe`.

i7 processor/32 GB Ram or better:

The `UCB_Plume.exe` software has been tested on contemporary WINDOWS operating systems; here are their listings with the average times to processes 512 frames:

-Intel® Xeon® CPU E5607 @ 2.27 GHZ, 128 GB Ram
Running Windows 2008 R2
~60 minutes
-Intel® Core™ i7-4700 MQ CPU @ 2.40 GHZ, 24GB RAM
Running Windows 7
~25 minutes
-Intel® Core™ i7-4710 MQ CPU @ 2.50 GHZ, 16 GB RAM
Running Windows 8.1
~25 minutes
-Intel® Core™ i5-4430 MQ CPU @ 3.00 GHZ, 12 GB RAM
Running Windows 8.1
~35 minutes

Imaging requirements

Uncompressed AVI Gray level

-Compressed, lossless, or lossy video will result in the code aborting. Only uncompressed AVI using the Basic Windows bitmap format codec is supported for gray level.

Static recording position

-Have the ROV record the jet/plume in a fixed reference with respect to the orifice. The camera should be faced perpendicular to the jet's/plume's centerline axis

Lighting

-The flows must be illuminated by a continuous, directed, polarized light source. Continuous light is critical, and must be differentiated from certain light sources that only simulate continuous by pulsing on and off at a high frequency not discernible to the human eye. When recording pulsing light with a high speed camera, consecutive images can alternate from high to low intensity leading to poor results. The light source should originate from near the ROV/camera. Polarization of the light along with the use of a polarized lens will diminish the occurrences of bright reflections of light at the interfaces, these glares lead to less reliable results if not dealt with.

Image centered 3 orifice diameters along the jets axis

-Include the discharging orifice in the image and have the image centered at 3 orifice diameters lengths from the orifice itself.

Video image's length is aligned with axis of the jet

-The jet should discharge left to right in the video recording.

Sufficient Frame Rate to capture evolution of features

-The recording frame rate should be sufficient to continuously capture the motion of the visible features.

Execution

In the directory UCB_Plume, four files are essential to the operation of the software:

- **`UCB_Plume.exe`:** The main batch file that prompts and automates the invocation of the `WALPT7.exe` with the proper input file.

-**`WALPT7.exe`:** The computing heart of this software. **`WALPT7`** source code written in FORTRAN, allowing it to both efficiently and effectively run calculation.

-**`bplume.inp`**: This input file will be edited and verified by the user. It contains parameters pertaining to the AVI as well as the flow.

-**`plume.inp`**: This is a self-contained input file which houses parameters used in analyzing the video.

Running Program:

-Double-click **`UCB_Plume.exe`** and follow the command line instructions.

-The user will be required to input/verify the information in **`bplume.inp`**:

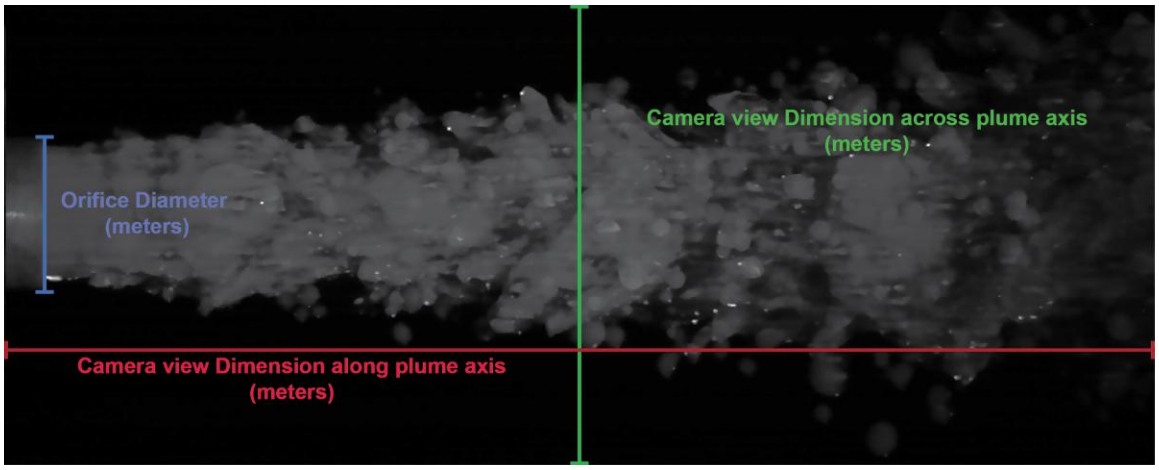
- Camera view dimension along plume axis, in meters (Line 2)

- Camera view dimension across plume axis, in meters (Line 4)

-real world ("Physical") length scale of the captured video

- Discharge orifice diameter in meters (Line 6)

-the diameter of the jet's/plume's orifice


Camera view Dimension across plume axis
(meters)
Orifice Diameter
(meters)
Camera view Dimension along plume axis
(meters)

(Test image for reference)

Each of these length are related to field measurements and estimations

- Camera frame rate (Line 8)

-the capture frame rate of the video

- Discharge fluid specific gravity (Line 10)

-the specific gravity of the jet/plume being discharged

-AVI file location (Line 12)

-the path to the video file storage location

Note: There are many ways to copy the file path. Here is one for reference:

-Locate the AVI file

-Shift + Right-Click the file and select

Copy as path

-Paste the path in bplume.inp in the correct line (line 12)

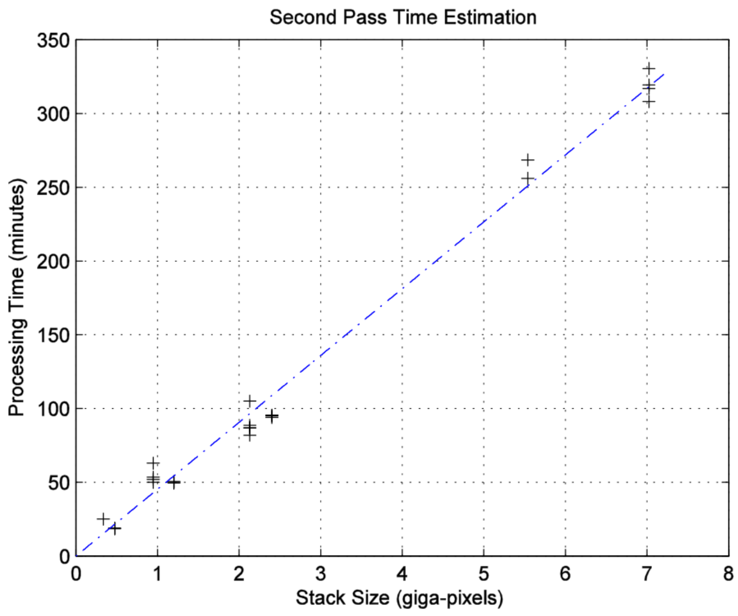
(Remove the quotation marks from the front and end of the pasted file path)

After editing, verifying, saving, and closing the input file, follow the command line instructions.

NOTE: Assume that the calculation time will scale with the number of frames, N_{frames}, by

$$Time \propto N_{frames} \cdot log(N_{frames})$$

2.5 GHz i7 processor with 24 GB ram



Results

Two **`.txt`** files are generated upon completion of the program.

`PlumeData.txt`:
Holds the printed information of the most recent run. This information lists parameters input by the user, such as:

- AVI file location
- Field of view dimensions
- Framing rate
- Specific gravity

Listed also are other processing parameters that were derived from information gained in the initial pass. These parameters are:

- Image resolution
- Number of images used in PixTif'ing
- PixTif filter type, center, and width

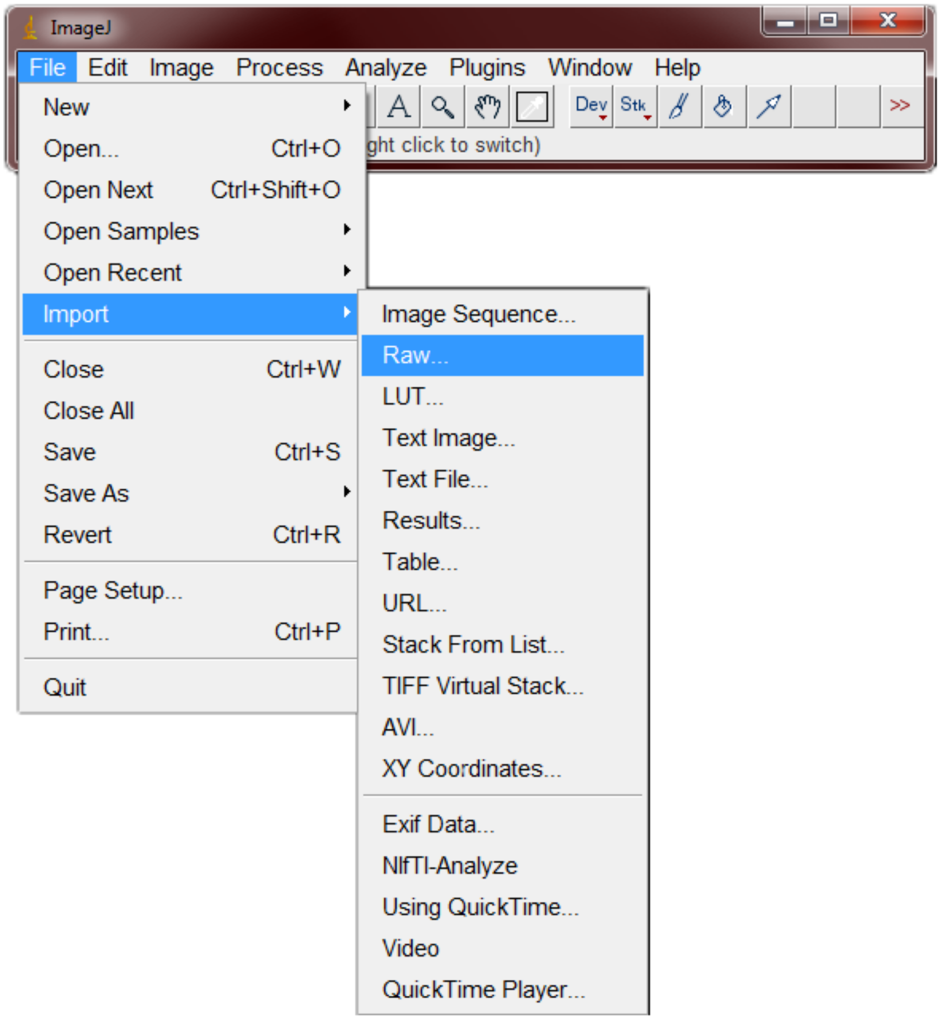
The listing of these parameters allows for a connection between a video and how it was processed to calculate the estimated flow rate, which is listed in a range of units.

`PlumeDataArchive.txt`:
An archive of the all **`PlumeData.txt`** from prior runs of **`UCB_Plume.exe`**. To start a new archive, move/rename the current **`PlumeDataArchive.txt`** prior to running **`UCB_Plume.exe`.**

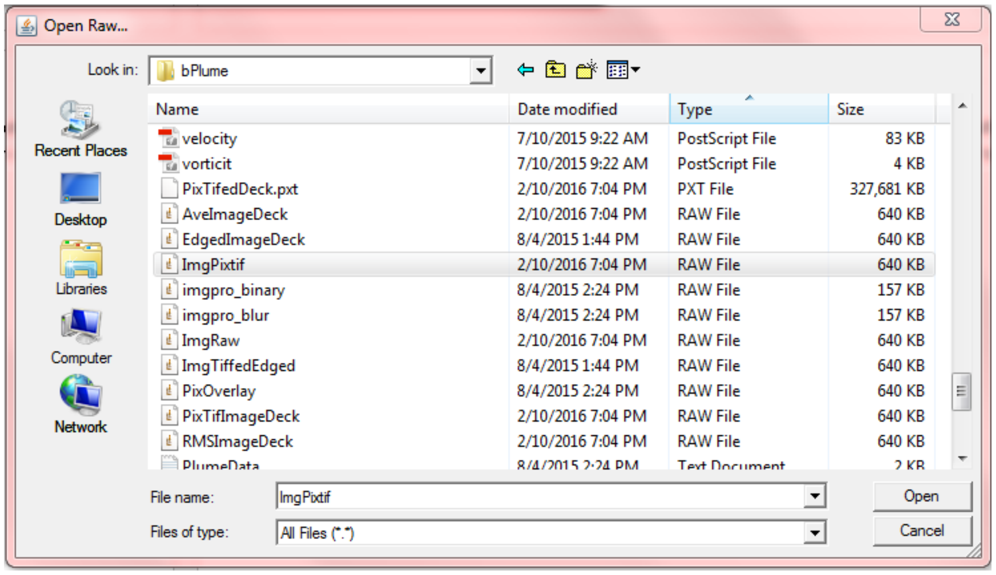
Images: A range of images are saved for diagnostic and process visualization purposes. While other image handling programs can open these file, the process using ImageJ will be described due to its open source and robust use.

For all `.raw`, `.edg`, `.pxt`, begin by running ImageJ.

Click: File > Import > Raw…



Select the file desired to be opened


Open Raw...

Look in: bPlume

Name	Date modified	Type	Size
velocity	7/10/2015 9:22 AM	PostScript File	83 KB
vorticit	7/10/2015 9:22 AM	PostScript File	4 KB
PixTifedDeck.pxt	2/10/2016 7:04 PM	PXT File	327,681 KB
AveImageDeck	2/10/2016 7:04 PM	RAW File	640 KB
EdgedImageDeck	8/4/2015 1:44 PM	RAW File	640 KB
ImgPixtif	2/10/2016 7:04 PM	RAW File	640 KB
imgpro_binary	8/4/2015 2:24 PM	RAW File	157 KB
imgpro_blur	8/4/2015 2:24 PM	RAW File	157 KB
ImgRaw	2/10/2016 7:04 PM	RAW File	640 KB
ImgTiffedEdged	8/4/2015 1:44 PM	RAW File	640 KB
PixOverlay	8/4/2015 2:24 PM	RAW File	640 KB
PixTifImageDeck	2/10/2016 7:04 PM	RAW File	640 KB
RMSImageDeck	2/10/2016 7:04 PM	RAW File	640 KB
PlumeData	8/4/2015 2:24 PM	Text Document	2 KB

File name: ImgPixtif

Files of type: All Files (*.*)

For **single image**, continue by inputting **NXC**, number of pixels along the width of the video image, and **NYC** ,number of pixels along the height of the video image, that correspond to the processed video.

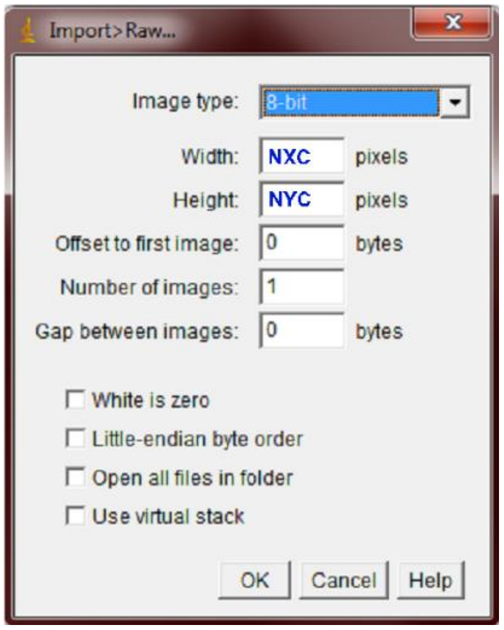

Import>Raw...

Image type: 8-bit

Width: NXC pixels

Height: NYC pixels

Offset to first image: 0 bytes

Number of images: 1

Gap between images: 0 bytes

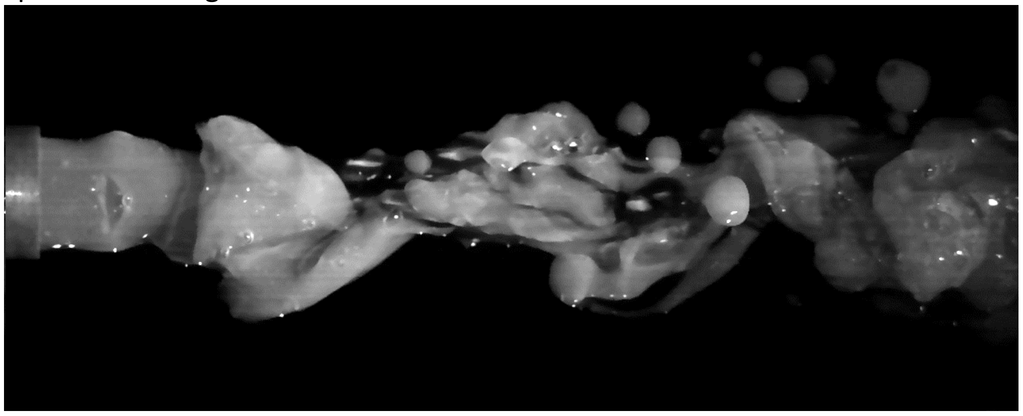
☐ White is zero

☐ Little-endian byte order

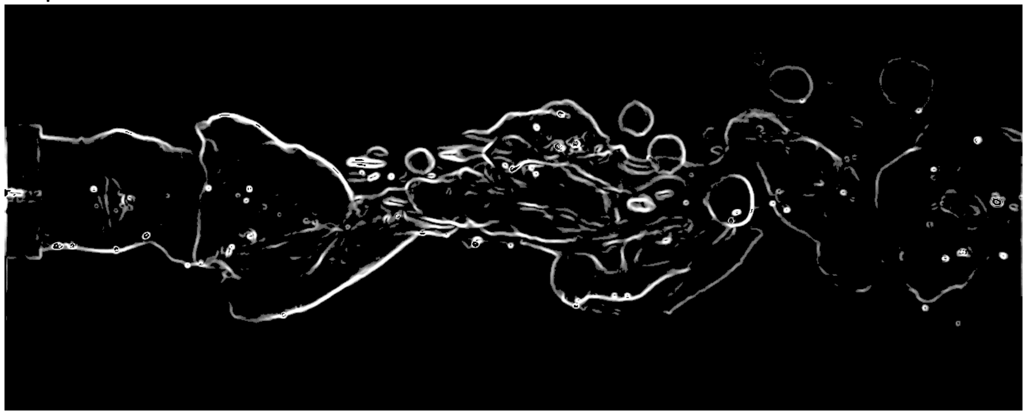
☐ Open all files in folder

☐ Use virtual stack

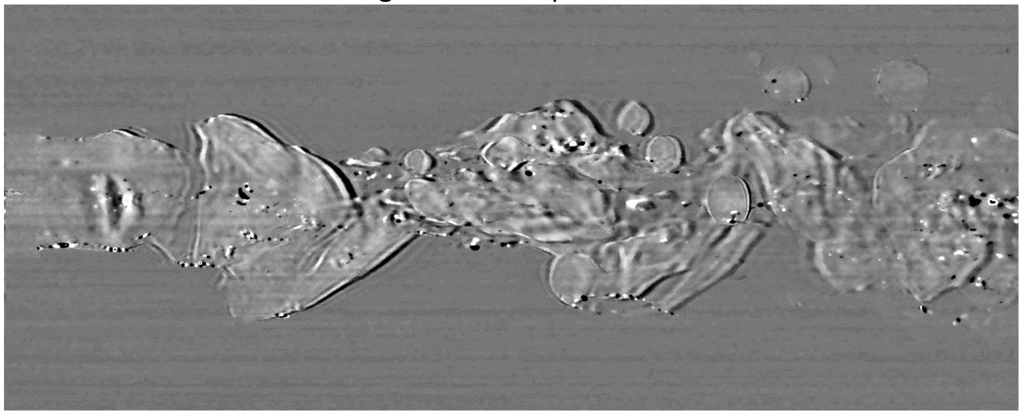
-`ImgRaw.raw`: An image frame taken from the middle of the image deck, unprocessed image for reference.



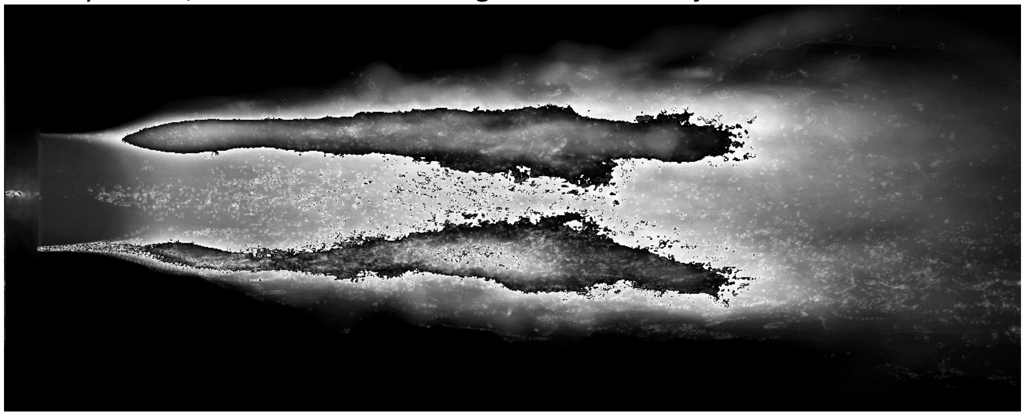
-`ImgEdged.raw`: A Canny edge detected version of image saved in **`ImgRaw.raw`**. This provides insight into which features are being tracked in the first pass.



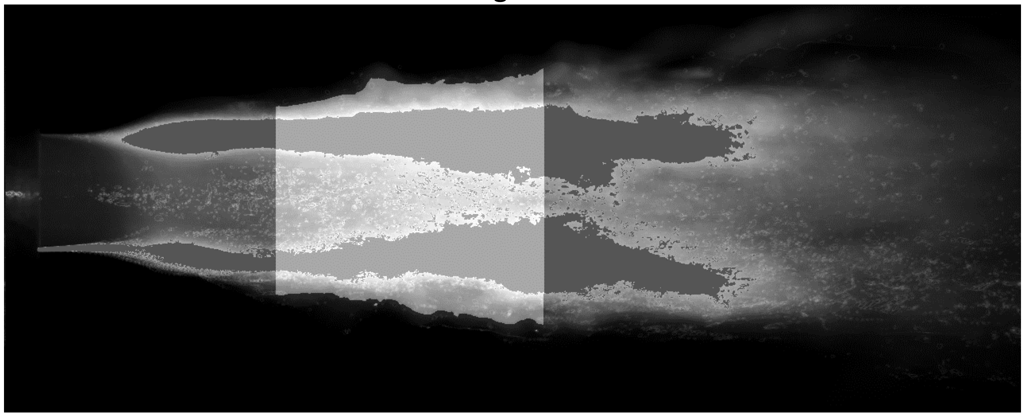
-`ImgPixtif.raw`: A Pixtif processed version of the image saved in **`ImgRaw.raw`**. This provides insight into which features are being pronounced in the flow to be tracked during the second pass.



-`RMSImage.raw`: As stated earlier in assumptions, the RMS of the image deck removes constant features in the ROI, while outlining the location of fluctuating intensity values, such as where moving features in the jet occur.



-`PixOverlay.raw`: The area of calculation is outlined as a whited region overlaid on the RMS of the entire image deck.



For **an image deck**, continue by inputting **NXC**, number of pixels along the width of the video image, and **NYC**, number of pixels along the height of the video image, and **NF** (this number is recorded in PlumeData.txt), number of frames processed, that correspond to the processed video. The "Offset to first image" should be set to 128 bytes to bypass the header information stored along with each image deck.

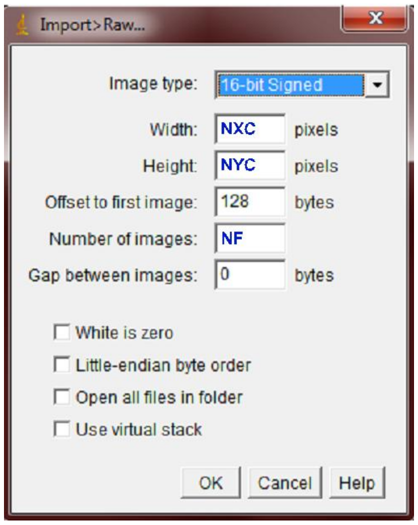


Image type: 16-bit Signed
Width: NXC pixels
Height: NYC pixels
Offset to first image: 128 bytes
Number of images: NF
Gap between images: 0 bytes

☐ White is zero
☐ Little-endian byte order
☐ Open all files in folder
☐ Use virtual stack

-`EdgedDeck.edg`: The raw image deck is ran through canny edge detection algorithm. The **`EdgedDeck.edg`** used for calculation during the first pass of **`WALPT7.exe`** in **`UCB_Plume.exe`**.

-`PixTifedDeck.pxt`: The raw image deck is ran through a pixel-wise filtering using parameters derived from the first pass. The **`PixTifedDeck.pix`** is used for calculation during the second pass of **`WALPT7.exe`** in **`UCB_Plume.exe`**.

Five `.out` files will be generated by `UCB_Plume.exe`. Third party software can be used to interpret the data stored in these files (Adobe CS, IDL, Matlab, VLC, ImageJ …).

The files are as follows:

The instantaneous information will be ordered as **`AviTensor.####`**. The **`.####`** will indicate which pair of images are being examined between. A correlation between Image1 and Image2 of the video deck will correspond to **`AviTensor.0001`**. In **`AviTensor.####`** information is stored as follows:

The output data file contains three arrays, **header, uvtensor, intwinxy,** all in binary. **header** contains processing and id information; **uvtensor** contains velocity data; and **intwinxy** consists of interrogation window dimensions in pixels when adaptive processing mode is used.

```
.
integer(2) header(64)
integer(1),allocatable :: intwinxy(:,:)
real(4),   allocatable :: uvtensor(:,:,:)
.
allocate(uvtensor(nxuv,nyuv,10))
allocate (intwinxy(nxuv,nyuv,2))
   intwinxy=0
.
open(1,file=utensor,form='binary')
write(1) header, uvtensor, intwinxy
close(1).
```

This output file contains a header which is **integer*2(64).**

Current contents are:

header(1)=version number (x100), used to determine compatibility

header(2)=nxc	camera size-x
header(3)=nyc	camera size-y
header(4)=nxuv	output array size-x
header(5)=nyuv	output array size-y
header(6)=nxw	window size-x
header(7)=nyw	window size-y
header(8)=nxs	step size-x
header(9)=nys	step size-y
header(10)=nxf	flow region size-x
header(11)=nyf	flow region size-y
header(12)=xf	flow region offset-x
header(13)=yf	flow region offset-y
header(14)=nbits	pixel depth
header(15)=ipx_ctr	number of boundaries in ipx, 0=no ipx
header(16)=mode	processing mode
header(17)=FileType	input file type 0=raw,1=tiff,2=avi stream
header(18)=(nt-1) or (ntfft-1)	Tensordeck output depth, for AVIbatch or PixTif processing for .raw and .tifdecks

header(19:64)	reserved for future use

The output data array **uvtensor** is a **binary** tensor containing the filtered velocity vector **u**=(u, v), velocity gradient tensor $\partial u_i/\partial x_j$, unfiltered velocity vector **u**=(u, v), and the correlation normalized coefficient in a (*,*,10) array. Horizontal pixel dimension [hpixel] is the length unit. Both u and v velocity components have units of [hpixel/δt]. For example, to calculate circulation, the arc length must be measured in horizontal pixel units. The unit of vorticity is [hpixel/δt] / [xstep].

The velocity data are in a 3D array of dimensions uvtensor(nxuv,nyuv,10) where

uvtensor(*,*, 1) = u	[hpixel/δt]
uvtensor(*,*, 2) = v	[hpixel/δt]
uvtensor(*,*, 3) = ∂u/∂x	[1/δt]
uvtensor(*,*, 4) = ∂v/∂x	[1/δt]
uvtensor(*,*, 5) = ∂u/∂y	[1/δt]
uvtensor(*,*, 6) = ∂v/∂y	[1/δt]
uvtensor(*,*, 7) = u_unfiltered	[hpixel/δt]
uvtensor(*,*, 8) = v_unfiltered	[hpixel/δt]
uvtensor(*,*, 9) = 0 reserved	
uvtensor(*,*,10) = correlation coefficient	[normalized]

1.1. `ImageDeck.out`: image deck consisting of the first image of each pair

`tensordeck.out` : tensor output file is stacked in a deck for each image pair

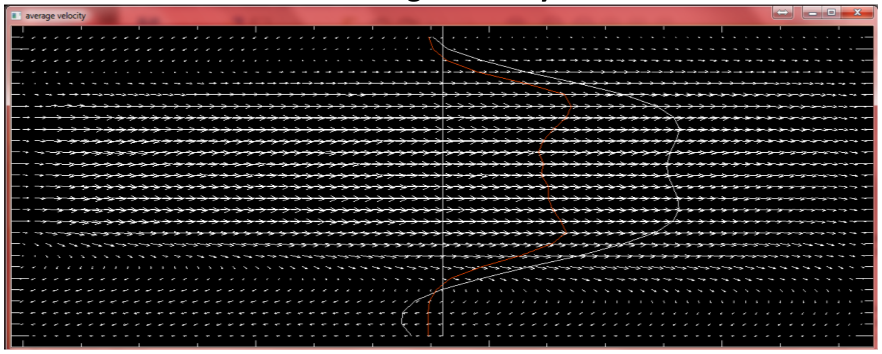
`tensorave.out` : average tensordeck.out, written by stats.f90

`statistics.out` : statistics of results in **`tensordeck.out`**, written by **`stats.f90`**

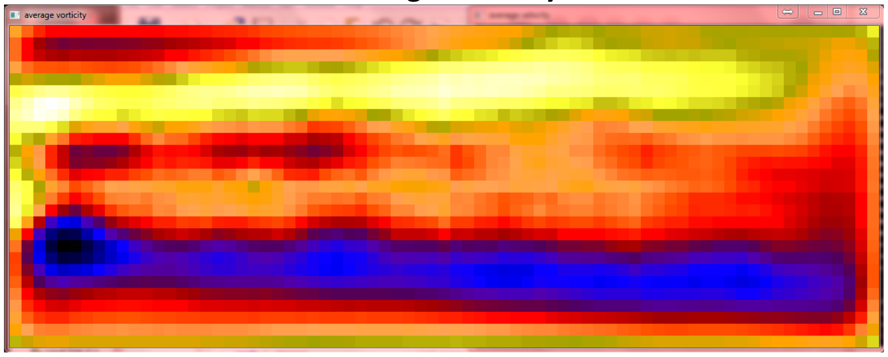
statistics(*,*, 1) = u_ave
statistics(*,*, 2) = v_ave
statistics(*,*, 3) = average vorticity
statistics(*,*, 4) = average speed
statistics(*,*, 5) = u^prime
statistics(*,*, 6) = v^prime
statistics(*,*, 7) = speed^prime
statistics(*,*, 8) = vorticity^prime
statistics(*,*, 9) = u_ave (unfiltered)
statistics(*,*,10) = v_ave (unfiltered)
statistics(*,*,11) = average unfiltered speed
statistics(*,*,12) = unused
statistics(*,*,13) = unused
statistics(*,*,14) = unused
statistics(*,*,15) = unused
statistics(*,*,16) = unused

Using IDL, **`statistics.out`** can be used to visualize:

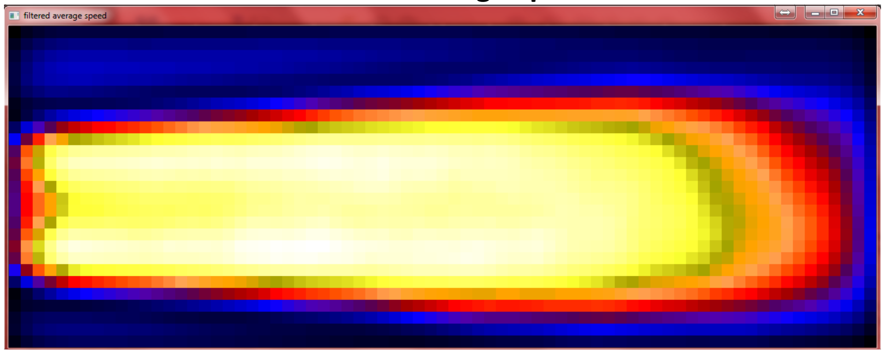
Average Velocity



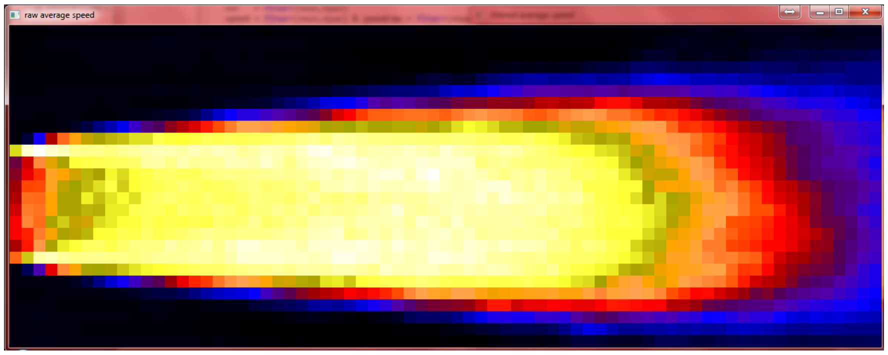
Average Vorticity



Filtered Average Speed



Raw Average Speed



Common Errors

ERROR [`.inp`] does not exist
Do you want to generate a template [`.inp`]?

If **`bplume.inp`** or **`plume.inp`** does not exist in the same directory as **`UCB_Plume.exe`**, the user will be prompted for permission to generate respective template input files. The user input file, **`bplume.inp`**, will still be presented to operator to be modified to run desired video recording.

A B O R T I N G [`File.avi`] video stream file does not exist

This error is stating the code could not find the file in the specified directory. Verify the files existence and that correct file path was specified in **`bplume.inp`** in line 12.

Known Issues

Insufficient Virtual Memory: Range of tests have been completed using hardware:

Windows OS [2008R2 / 7 / 8.1 / 10]
Intel i7 processor
24GB Ram

Using this range of hardware, an error stating insufficient virtual memory has occurred while processing large video files. As currently tested on our hardware, a video file of 7GB is processable.

Noise sensitivity: Noise found in the video recording can be pronounced and lead to the generation of erroneous velocity fields. Noise can be introduced to the recording via substandard recording equipment, compression/decompression of recording, or certain video enhancement processes.

Nulled image: Removal of certain frequencies while using PixTif'ing has resulted in the zeroing of all pixels in the processed image decks. By adjusting the filter width and center this can be corrected.

References

The uses of coherent structure –Coles, Donald

• http://resolver.caltech.edu/CaltechAUTHORS:20141210-131347636

A Computational Approach to Edge Detection –Canny, John

• Canny, John. "A computational approach to edge detection." Pattern Analysis and Machine Intelligence, IEEE Transactions on6 (1986): 679-698.

Deepwater Horizon: A Preliminary Bibliography of Published Research and Expert Commentary

•http://www.lib.noaa.gov/researchtools/subjectguides/dwh.html

Appendix

Advanced Operation

`UCB_Plume.exe` is a program that runs almost autonomously, taking in few inputs from a user to describe the physical parameters of the recording. With this information `UCB_Plume.exe` generates `plume.inp` to run a preliminary pass with `WALPT7.exe` and then uses a generated `update.inp` to run the second pass with estimated filtering parameters. While this functionality is set to be available for quick use, advance use of `WALPT7.exe` directly is possible.

To use `WALPT7.exe` three files will be required to be in the same directory:

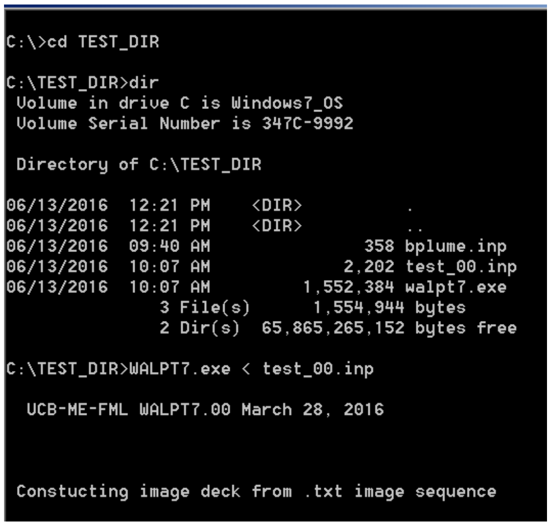
- **`WALPT7.exe`**
- **`bplume.inp:`** outlined in detail on page 8 under **Running Program** of this manual. It would be recommended to make a copy of the `bplume.inp` as generated by `UCB_Plume.exe.` The file name `bplume.inp` must remain unchanged, as `WALPT7.exe` will search the working directory for the exact file name.
- **`test_00.inp:`** The format of this file must be compatible with that of `plume.inp` as generated by `UCB_Plume.exe`. It would be recommended to make a copy of the `plume.inp`, rename as desired while preserving the `.inp` file extension, **`test_00.inp`** will be used to reference this duplicated file.

The directory housing these three files will be referred to as <TEST_DIR>, this can be named as desired. This directory will require an allotment of memory space to save to, roughly thrice the size of the images/.avi used.

From a command window, change the working directory to <TEST_DIR>.
Entering:

```
WALPT7.exe < test_00.inp
```

will use the parameters set in `test_00.inp` to process the images, computing a velocity field per image pair and, along with the information given in `bplume.inp`, the estimated flow rate.



```
C:\>cd TEST_DIR

C:\TEST_DIR>dir
 Volume in drive C is Windows7_OS
 Volume Serial Number is 347C-9992

 Directory of C:\TEST_DIR

06/13/2016  12:21 PM    <DIR>          .
06/13/2016  12:21 PM    <DIR>          ..
06/13/2016  09:40 AM               358 bplume.inp
06/13/2016  10:07 AM             2,202 test_00.inp
06/13/2016  10:07 AM         1,552,384 walpt7.exe
               3 File(s)      1,554,944 bytes
               2 Dir(s)  65,865,265,152 bytes free

C:\TEST_DIR>WALPT7.exe < test_00.inp

  UCB-ME-FML WALPT7.00 March 28, 2016



 Constucting image deck from .txt image sequence
```

Parameter Description: `plume.inp / test_00.inp`

	mode npass silent batch edgeflag PTFlag PTCenter PTWidth
A)	1 1 0 1 0 0 0 0
	file that contains the names of image files, prefix for outputfiles
B)	D:\OHMSETT\OHM\Test_23_input\Test_23_input
C)	TxtTensor
	image size nxc, nyc, nbits, pixr
D)	1040 564 8 1.00
	flow size, nxf, nyf
E)	1040 564
	flow offset, xf, yf
F)	0 0
	window size, nxw, nyw, 2n**
G)	64 64
	amod, min, max windows dimensions 2n, correlation level corlvl**
H)	1 8 32 0.40
	step size, nxs, nys
I)	16 16
	window type, wtype 1-7, see source listing
J)	2
	peak type, ptype 0=grid,1=parabolic,2=gaussian
K)	2
	laundary type, ltype 0=no laundering,1=rejection
L)	0
	extension parameter, 0= none, zero padding, 1= smooth
M)	0
	filter widths fltrwx, fltrwy wavenlength in steps; exponent
N)	9 9 2
	wall parameters: nwalls, parex, motion, intflag, outmask
O)	0 0 0 1 1
	wall geometry file
P)	Wall_Mask.raw
	motion parameters: dxcg, dycg ,rot
Q)	0.00 0.00 0.00
	0.00 0.00 0.00
	9.00 0.00 0.00

A) Primary Processing Parameters

`MODE`: Sets which type of PIV algorithm will be implemented.

`MODE=1`: Standard DPIV. In this mode, only one pass is done with the parameters specified. The code checks for questionable vectors and does some repairs; some ad-hoc, some rigorous. (relative time=1)

`MODE=2`: Predictor corrector DPIV; moving from smaller windows to larger ones. It is quick at the cost of robustness. The code makes the first pass at the specified window size, and repairs suspect vectors at twice or, if needed, at four times the window size to obtain an estimate for the final velocity pass. During the final pass, window pairs that are separated by the estimated velocity are used to determine the final velocity. Some repairs are done at the end. (relative time=2)

`MODE=3:` Lagrangian parcel tracking LPT with multiple passes. The velocity field calculated using MODE=2 is used as the first estimate in LPT. First, the domain is extended to $2^m x 2^n$. Then velocity gradient tensor and its derivative are calculated using Fourier methods after some filtering in frequency domain. Fluid parcels are deformed according to the local velocity field to determine the deformed correlation windows from which a new velocity field is determined. Each pass is done with a newly calculated velocity field. No repairs are done. (relative time=2+2*passes)

`MODE=4:` Adaptive Lagrangian parcel tracking aLPT with multiple passes. Chooses windows commensurate with the velocity vector, thus increases the spatial resolution of the instrument. The processing is the same as that in mode 3 except the dynamic adaptation of the window size and orientation. The velocity field calculated using MODE=3 with 2 passes is used as the first estimate in aLPT. No repairs are done. Since smaller windows are used most of the time, the incremental processing is faster than that in mode 3.

`npass:` is the number of passes for **walpt** processing when **`MODE`** = 3 or 4. Otherwise set to 1. The code expects a valid integer entry for **`npass`** even if the **`mode`** is other than 3 and 4.

`silent:` screen display mode,

`silent = 0 :` Verbose mode slows the CPU to crawl, probably due to poor operating system design (Windows)
`silent = 1 :`silent (minimal display of status). Set to 1 if processing a large number of files (say, larger that 20) when **`batch`** >0.

`batch:`. The contents of files **`file_1`** and **`file_2`** below are interpreted differently depending on the value of **`batch`**.

`batch = 0 :` Single image pair processing
`batch = 1 :` Multiple image pair processing

`edgeflag:` Parameter flag setting whether edge detection processing is ran on the images, and if so which type.

`edgeflag = 0` for no filtering
`edgeflag = 10` for Sobel, all around
`edgeflag = 11` for Sobel, along x-axis

edgeflag = 12 for Sobel, along y-axis
edgeflag = 25 for 5x5 Laplace filtering.
edgeflag = 30 for Canny filtering, all around.
edgeflag = 31 for Canny filtering, along x-axis.
edgeflag = 32 for Canny filtering, along y-axis

PixTifFlag:

PTFlag = 0: no PxTif'in
PTFlag = 1: high-pass filter (tanh)
PTFlag = 2: low-pass filter (1-tanh)
PTFlag = 3: band-pass filter (exp(-t^2))
PTFlag = 4: band-reject filter (notch) (1-exp(-t^2))
PTFlag > 4: returns without filtering, 2^n image deck written out.
PTFlag = 9: special handling

PixTifCenter: PixTif center of filter.

PixTifWidth: PixTif filter width parameter.

B) file_1: File name that is processed differently based on content and batch flag

C) file_2: File name that is processed differently based on content and batch flag

batch =0 ; single pair of images are processed

file_1 and **file_2** are the names of the pair of images for processing, one line per file, include full path to avoid confusion. Both image files must be of the same type, i.e. **.raw** and **.tif** files may not be mixed. The output data are written in **'tensor.out'** in the current directory (described below). If ipx is invoked, the path data are written in **'ipxpaths.out'** in the current directory.

batch =n>0: multiple pairs of images are processed

file_1 is the name of the file containing the names of images for processing. The images are taken as pairs until all entries are processed. If there are odd number of entries, the last one is discarded. Include full path to avoid confusion. Image files must all be of the same type, i.e. **.raw** and **.tif** files may not be mixed.

file_2 is the prefix used to construct the output file names for each of the pairs processed. The output files are numbered consecutively starting with the value of **batch** **.0001** extension. For example if **batch =99** and `file_2=tensor`, then the output files will start from `tensor.0099` in the specified folder. If no folder is specified, the output files will be in `TensorFolder` directory.

2. **AVI image stream**

`file_1` is the name of the avi file to be processed. It must have .**avi** or **.AVI** or **.Avi** extension. Compressed avi files are rejected. Images are processed for every interval; that is, if there are N images in the avi file, there will be N-1 output files. If the images are equi-spaced, then, no special handling is needed. If, however, the images are recorded in 'double exposure' mode for PIV operations, then successive output files will have alternating time intervals, one corresponding to the laser pulse separation and the other corresponding to sampling interval for PIV image pairs. Therefore, the output files must be processed with the timing considerations in mind. Even and odd numbered files must be grouped together

`file_2` is the prefix used to construct the output file names for each of the pairs as well as the consolidate output file for the whole stream

3. **TXT image sequence**

`file_1` is the name of the folder storing all the .txt image file to be processed. It must have i**nput** or **Input** or **INPUT** as its ending characters. Numbering style should be consistent with that exported from ImageJ. If there are N .txt images in the input folder, there will be N-1 output files. If the images are equi-spaced, then, no special handling is needed. If, however, the images are recorded in 'double exposure' mode for PIV operations, then successive output files will have alternating time intervals, one corresponding to the laser pulse separation and the other corresponding to sampling interval for PIV image pairs. Therefore, the output files must be processed with the timing considerations in mind. Even and odd numbered files must be grouped together

`file_2` is the prefix used to construct the output file names for each of the pairs as well as the consolidate output file for the whole stream

When **`ipx`** is invoked, the same prefix is used to generate the ipx'ed image and path files. This time, '**`ipx1/2`**' suffix is inserted in the file names for images and **`ipxpaths`** for path files. For example, the ipx'ed image file names corresponding to **`tensor.0099`** output are **`tensoripx1.0099`** and **`tensoripx2.0099,`** and the corresponding path file is **`tensoripxpaths.0099`**.

If the full path is included in **`file_2,`** e.g., **`file_2 = d:\airfoil\tensor`**, then, the appropriate directory must already exist.

D) Image Size: (figure 1a)

`nxc:` Number of pixels horizontally across the image
`nyc:` Number of pixels vertically across the image
`nbit:` Pix bit depth
`pixr:` camera pixel ratio, vertical/horizontal pixel dimensions

Camera	nxc x nyc	pxr	(µm x µm)	bits	*max frame rate*	
IDT-MP-X3	1280H x 1024V	1.00	(12x12)	8	1000 Hz	digital
Kodak ES1.0	1008H x 1018V	1.00	(9x9)	8	30 Hz	digital
Sony 7500	640H x 480V	1.00	(9.9x9.9)	8	30/60 Hz	analog/ non-interlaced

E) Flow Size: (figure 1a)

`nxf, nyf:` flow region of interest imbedded in **`(nxc,nyc)`** image array

F) Flow Offset window size: (figure 1a)

`xf, yf:` position of **`(nxf,nyf)`** region with respect to the image origin

G) Window size: (figure 1a)

`nxw, nyw`: correlation window size 2^m x 2^n. Consider using rectangular windows in nearly parallel flows

H) Adaptation Parameters

`amod:` Adaptation mode
- **`amod`** = 0: square windows only, no directional adaptation
- **`amod`** = 1: include rectangular windows aligned with axes

`minw, maxw:` are minimum and maximum window dimensions 2^m used in mode 4 LPT. The minimum value may be as low as 8 if image quality permits.

`corlvl:` is the minimum correlation coefficient level to double the window dimensions during LPT.

I) Step Size

`nxs, nys:` step size for scanning the flow region `(nxf,nyf)`

J) Window Type

`wtype:` window type for windowing data before correlating. If the images are periodic, you must use a window.

- `wtype` = 1: square window (no windowing)
- `wtype` = 2: Rosenfeld
- `wtype` = 3: triangle(Parzen, Bartlett)
- `wtype` = 4: parabolic (Welch)
- `wtype` = 5: cosine (Hanning)
- `wtype` = 6: Hamming
- `wtype` = 7: Blackman-Harris

K) Peak Type

`ptype:` method of determining correlation maximum

`ptype =` 0: read off the array values (you might as well not use the program)

`ptype =` 1: paraboloid fitting

`ptype =` 2: gaussianoid fitting (logarithms are used, correlation data must be positive values).

L) Launder Type

`ltype:` bad vector rejection flag,

`ltype` = 0: no rejections

`ltype` = 1: rejections enabled

M) Extension Parameter

`exten`: domain extension flag

`exten` = 0: extend with zero padding (use when far field is quiescent),

`exten` = 1: smooth extension with matching 2nd order derivatives.

N) Filtering parameters

Filtering parameters used in wavenumber domain (Figure 5).

`fltrwx, fltrwy` are the wavelengths (in units of steps) at the 1/e cut-off point of the filter kernel. Note that the step sizes `(nxs,nys)` determine the actual cut-off wavelengths (in pixels) on PIV images. The larger the parameters are, the smoother the output is. Equivalently, features smaller than the parameters are filtered out (blurred). If **`fltrwx=0`**, or **`fltrwy=0`**, then, no filtering is done in x or y, respectively.

`Nfil:` is the exponent in the filter kernel.
Higher values mean sharper cut-off in wavenumber domain.
The filter kernel is $[1-\exp(-1/k^n)]$ where k is the magnitude of the wave vector,
(k_x, k_y) : $k \sim \{ [k_x / (1 / \mathrm{fltrwx})]^2 + [k_y / (1 / \mathrm{fltrwy})]^2 \}^{1/2}$
Alternatively,
(k_x, k_y) : $k \sim \{ [\mathrm{fltrwx} \cdot k_x]^2 + [\mathrm{fltrwy} \cdot k_y]^2 \}^{1/2}$

O) <u>Wall parameters</u> (figure 2).

`nwall`: number of interfaces(s) in PIV images, to be dealt with.
  `nwall` = 0: no interfaces
  `nwall` = +n , n interfaces, **`boundary`** data file is needed (described below).
  `nwall` = -n , n interfaces, boundaries are determined from the first flow image. Requires high contrast data, to clearly delineate the flow region.
`parex:` image parity exchange at walls to extend velocity measurements to interfaces
  `parex` = 0: Off
  `parex` = 1: On
`motion`: interfaces to be dealt with
  `motion` = 0: no motion, fixed walls
  `motion` = 1: walls in rigid body motion (individually or in unison)
  `motion` = -1: compliant interfaces (every interface is treated independently)

`intflag:` used when **`parex`** =1
  `intflag` = 0: nearest pixel,
  `intflag` = 1: interpolation over 3x3 region (recommended)
`outmask:` used when **`wall`** =1 to deal with output data in extended regions.
  `outmask` = 0: output as is (for further processing at interfaces, e.g. shear at wall)
  `outmask` = 1:set wall regions to 1001.0 (suitable for display in IDL using **`MISSING`** keyword in **`velovect`** procedure)

P) Wall Geometry File

`mpairs`=0
file marking the interfaces(s) appearing the first image file **`imagefile_1`**. Must be supplied when **`nwall > 0`**. Has the same structure as the image files **`imagefile_1/2`** above. Byte elements are 255 (-1)(white) when in the metric flow domain and 0(black) when in the non-metric domain (figures 2,4). The interface outlines can be closed loops or open lines. There is no limit to the number of interfaces in the flow field. High curvatures should be avoided. Cusps are not allowed in autonomous processing.

`mpairs`=1
`boundary` is the name of the file containing the names of boundary files for batch processing. A boundary mask file matched to each image pair listed in **`file_1`**.

Q) Motion Parameters

rigid body motion parameters of wall outlines, needed when **`motion=1.`** One line for each boundary.

`dxcg` translation displacement of the centroid of wall outline in x-direction (pixels)
`dycg` translation displacement of the centroid of wall outline in y-direction (pixels)
`rot` rotation of wall outline in radians

`CAUTION`: If the number of line entries are fewer than the boundaries in **`boundary`**, then, the last entry is used for the remaining boundaries. If the walls are in pure translation, then the results are correct. If there is any rotation, the results will be wrong.

`NOTE:` If there are multiple walls which are moving as a solid body in unison, a single entry is sufficient provided that **`nwall=1`**.

See figure 4 for the ordering convention of the interfacess. **`ipx`** looks for interfaces in **`boundary`** starting on **`edge_1`**, going around the four edges counterclockwise. Interfaces starting on the edges are numbered sequentially until the open-ended interfaces are taken care of. Then, **`ipx`** starts scanning vertically the interior of the **`boundary`** file stating from x=0. The numbering of the closed interfaces then continues until all interior interfaces are accounted for. If all interfaces are rigid walls, then, **`ipx`** can automatically perform image parity exchange using the rigid-body-motion parameters prescribed above.

FIGURES

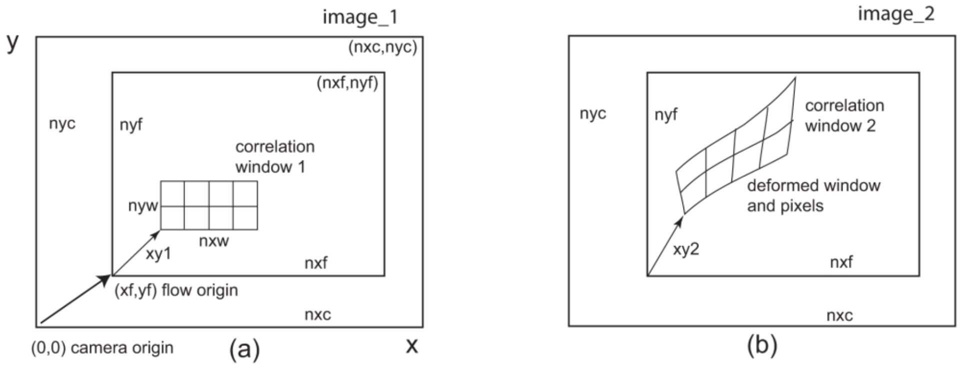


Figure 1. Lagrangian Parcel Tracking -- LPT.

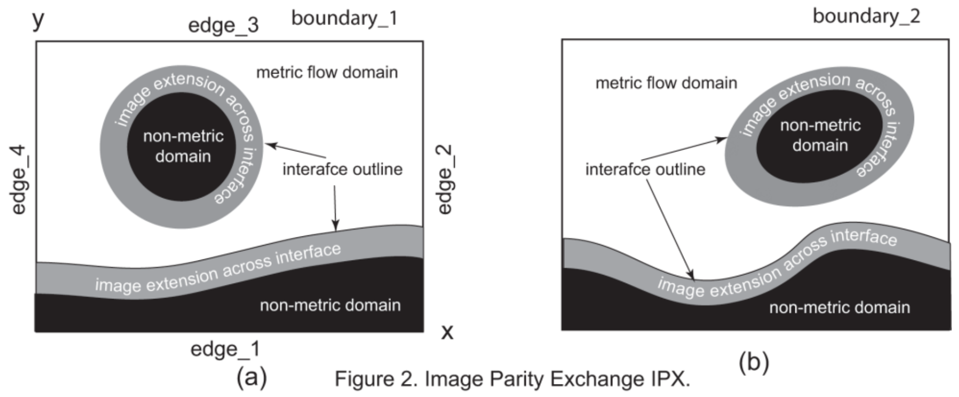


Figure 2. Image Parity Exchange IPX.

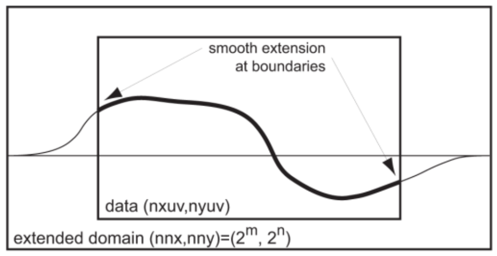

smooth extension at boundaries
data (nxuv,nyuv)
extended domain (nnx,nny)=(2^m, 2^n)

Figure 3. Extending data domain for FFT processing.

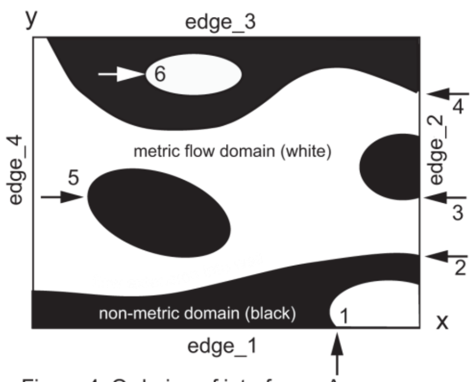


Figure 4. Ordering of interfaces. Arrows indicate starting points of interface outlines.

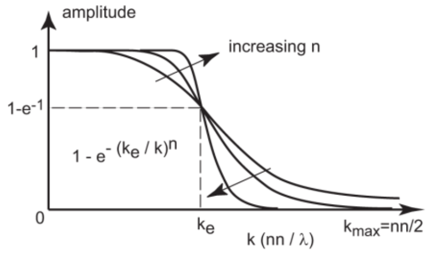


Figure 5. Wavenumber filter kernel.

APPENDIX-2
`Plume.bat` source listing

BplumeS.bat

```
rem executes plume software
rem June 13, 2016
rem Omer Savas, UCB-ME-FML-140HH
rem Eric Ibarra
echo off
title UCB_Plume: berkeley plume analysis tool
         cls
color 1f
setlocal enableextensions  enabledelayedexpansion


set InputFile=plume.inp
set UserInput=bplume.inp
set update=bpUpdate.inp

set tempfile=tmpy.tmp
copy /y nul %tempfile% > nul


set extract=12
set extract2=14
set replace=4
set replace2=7
set replace3=9

echo.
echo   UCB_Plume: berkeley plume analysis tool
echo   June 5, 2016
echo   UCB-ME Fluid Mechanics Laboratory


if     exist %update% (erase %update% > nul)



REM _____ CHECKING FOR bplume.inp _____
if     exist %UserInput% (goto :continue1)

echo.
echo ERROR [ %UserInput% ] does not exist
echo.
```

```
REM Template Generator for UserInput
echo      Do you want to generate a template [ %UserInput%
]?
     set /p INPUT= Type [ Y/N ]:
     SET INPUT_=%INPUT:~0,1%
     If "%INPUT_%"=="Y" goto yes1
     If "%INPUT_%"=="y" goto yes1

echo     A B O R T I N G [ %UserInput% ] does not exist
goto :eof

:yes1
REM iput file generated
echo.
copy /y nul %UserInput% > nul
echo. Camera view dimension along plume axis, in meters >>
%UserInput%
echo. 0.12 >> %UserInput%
echo. Camera view dimension across plume axis,  in meters
>> %UserInput%
echo. 0.049 >> %UserInput%
echo. Discharge orifice diameter in meters >> %UserInput%
echo. 0.013 >> %UserInput%
echo. Camera frame rate >> %UserInput%
echo. 500 >> %UserInput%
echo. Discharge fluid specific gravity >> %UserInput%
echo. 0.90 >> %UserInput%
echo. AVI file location >> %UserInput%
echo. c:\avi\AVIReadTestFile7.avi >> %UserInput%
echo. Recording pixel dimension (Required for .txt
processing) >> %UserInput%
echo. 1280 512 >> %UserInput%

echo Template [ %UserInput% ] successfully generated.
pause

:continue1
REM _____ CHECKING FOR plume.inp _____
if   exist %InputFile% (goto :continue2)

echo.
echo ERROR [ %InputFile% ] does not exist
echo.

REM Template Generator for InputFile
echo      Do you want to generate a template [ %InputFile%
]?
```

```
     set /p INPUT= Type [ Y/N ]:
     SET INPUT_=%INPUT:~0,1%
     If "%INPUT_%"=="Y" goto yes2
     If "%INPUT_%"=="y" goto yes2

echo     A B O R T I N G [ %InputFile% ] does not exist
goto :eof

:yes2
echo.
copy /y nul %InputFile% > nul
echo. mode npass silent batch edgeflag PTFlag PTCenter
PTWidth >> %InputFile%
echo. 1 1 0 1 30 5 4 8 >> %InputFile%
echo. file that contains the names of image files, prefix
for outputfiles >> %InputFile%
echo. c:\avi\AVIReadTestFile7.avi >> %InputFile%
echo. AviTensor >> %InputFile%
echo. image size nxc, nyc, pixr >> %InputFile%
echo. 1280 512 8 1.00 >> %InputFile%
echo. flow size, nxf, nyf >> %InputFile%
echo. 1280 512 >> %InputFile%
echo. flow offset, xf, yf >> %InputFile%
echo. 0 0 >> %InputFile%
echo. window size, nxw, nyw, 2**n >> %InputFile%
echo. 64 64 >> %InputFile%
echo. amod, min, max windows dimensions 2**n, correlation
level corlvl >> %InputFile%
echo. 1 8 32 0.40 >> %InputFile%
echo. step size, nxs, nys >> %InputFile%
echo. 16 16 >> %InputFile%
echo. window type, wtype 1-7, see source listing >>
%InputFile%
echo. 2 >> %InputFile%
echo. peak type, ptype 0=grid,1=parabolic,2=gaussian >>
%InputFile%
echo. 2 >> %InputFile%
echo. laundary type, ltype 0=no laundering,1=rejection >>
%InputFile%
echo. 0 >> %InputFile%
echo. extension parameter, 0= none, zero padding, 1= smooth
>> %InputFile%
echo. 0 >> %InputFile%
echo. filter widths fltrwx, fltrwy wavenlength in steps;
exponent >> %InputFile%
echo. 9 9 2 >> %InputFile%
echo. wall parameters: nwalls, parex, motion, intflag,
outmask >> %InputFile%
echo. 0 0 0 1 1 >> %InputFile%
echo. wall geometry file >> %InputFile%
echo. plane1.raw >> %InputFile%
echo. motion parameters: dxcg, dycg ,rot >> %InputFile%
echo. 0.00 0.00 0.00 >> %InputFile%
echo. 0.00 0.00 0.00 >> %InputFile%
echo. 9.00 0.00 0.00 >> %InputFile%

echo Template [ %InputFile% ] successfully generated.

:continue2

echo.
         pause

echo     confirm the contents of the input file [
%UserInput% ]
echo     edit the video parameters as needed
echo.
echo     S A V E  and  C L O S E [ %UserInput% ] when
finished
         notepad %UserInput%
echo.

rem _________________________________________________

rem extracting video file location

set "var="
set /a extract-=1
for /f "skip=%extract% delims=" %%i in (%userinput%) do if
not defined var set "var=%%i"

rem extracting video file location

set "var2="
set /a extract2-=1
for /f "skip=%extract2% delims=" %%i in (%userinput%) do if
not defined var2 set "var2=%%i"

rem Editing input file %inputfile%
set line=0

for /f "delims=" %%j in (%inputfile%) do (
    set /a line+=1
    if !line!==%replace% (
        echo.%var%>>%tempfile%
     )    else if !line!==%replace2% (
        echo.%var2% 8 1.00>>%tempfile%
    )     else if !line!==%replace3% (
        echo.%var2%>>%tempfile%
    ) else (
        echo %%j>>%tempfile%
    )
)

erase %inputfile%
ren %tempfile% %inputfile%
rem type %inputfile%


rem _________________________________________________

if  not  exist %var% (
echo.
echo     A B O R T I N G [ %var% ] video stream file does
not exist
     pause
goto :eof
)

echo     hit any key to continue, CTRL-C to abort
echo.
         pause
echo.
echo     processing... may take   S E V E R A L   M I N U T
E S
rem         if not exist TensorFolder (mkdir TensorFolder >
nul)
echo.
echo     CTRL-C to abort
echo.
echo     First pass
echo.
         walpt7<%InputFile% >nul  rem First pass
echo.
echo     Second Pass
           walpt7<%update% >nul  rem Second pass

echo     finished processing
rem echo.    >> PlumeData.txt
rem echo         %DATE:/=-%   %TIME::=-% >> PlumeData.txt
rem echo.    >> PlumeData.txt
echo.
echo ------------------------------------------------------
--
echo     current results in  [PlumeData.txt]
echo ------------------------------------------------------
--
         type PlumeData.txt
rem echo ----------------------------------------------------
------
rem         if     exist Plume.txt (copy /y
Plume.txt+PlumeData.txt Plume.txt > nul)
rem         if not exist Plume.txt (copy /y PlumeData.txt
Plume.txt > nul)
echo ------------------------------------------------------
--
echo     accumulated results in  [PlumeDataArchive.txt]
echo ------------------------------------------------------
--
         pause
echo.
echo     C L O S E [PlumeDataArchive.txt] when finished
echo.
         notepad PlumeDataArchive.txt
         pause

 rem        start c:\rsi\idl63\bin\bin.x86\idlde.exe
c:\walpt6\stats.pro

 rem restore DOS color scheme

         color 0f
         title Command Line  Interface

         exit /b
 rem END
```

APPENDIX-3

BSEE Presentation