Development of graphical user interface, data acquisition, and computer vision software for the Outwash project

Patrick W. Spencer

Aeromechanics Intern,

NASA Ames Research Center,

Mountain View, CA

Mechanical Engineering Student,

JB Speed School of Engineering,

Louisville, KY

Abstract

The goal of the Outwash project is to better characterize horizontally outflowing air from rotors close to the ground. To do this, software is required that: coordinates data collection, displays test safety information, eases sensor calibration, and displays collected data post-test. Additionally, software to automatically determine the angle of aerodynamic tufts was desired. Lacking suitable preexisting options for these tasks, software to do them was developed in-house using the Python language, the Qt framework for GUI, and the OpenCV library for computer vision.

Contents

Introduction

The DataWash program, written in Python and XML and making heavy use of the Qt framework (specificially its Python port PyQt), handles data acquisition and display for outwash characterization testing. It talks to various other devices and displays and logs a plethora of sensor data. The functionality of this software is the main focus of this document. The initial prototype for this software was built by Meenakshi Manikandan, a summer 2023 intern.

The TuftVision software, written in Python and relying heavily on the OpenCV framework, uses computer vision to find the pointing angle of aerodynamic tufts in images.

DataWash

Requirements and goals

The DataWash software had a number of goals during its development. These were divided into goals for the user experience and goals for the codebase (to ease current and future development and make bugs less common).

The main user experience goal was simply to implement all of the functionality required by users: collecting data, displaying live data in various ways, displaying collected data, and various related functions. This functionality was ascertained by discussing with the future users of the software what they would need out of it and talking with people who had used similar software in the past. Some future users were specifically interested in certain parts of the software and thus had specific requirements for those parts, but in general desired functionalities were gathered from discussions with the entire Outwash project team.

A secondary user experience goal was ease of use. This was pursued in a variety of different ways: easily-understandable naming for any named component (button, window, data, etc.), as-expected functionality (things generally work how the user would expect them to), clear program state feedback to the user, and clear documentation.

An additional user experience goal was configurability: the program was designed to be highly configurable without needing to change the code. The exact use conditions of any given software are only known approximately while it is being developed, so giving the user various (not in code) settings to change and making it clear how exactly to change them lets them adapt it to their exact use conditions.

Some parts of the software are safety-critical, specifically the Safety of Flight window. At-a-glance readability was a consideration for each window, but took a more central role for the design of the Safety of Flight window.

Code modularity - making the easily and quickly adaptable - was the central goal for the codebase layout. This is analogous to user-facing configurability, but in fact they are implemented in two entirely different ways. This focus greatly accelerated the development of the software as it progressed, as already-developed components or functionality could be adapted and reused, and each time a modification was made to a given functionality, care was taken to make future modifications even easier and quicker.

A secondary goal for the codebase was sensible and readable code division, so that future developers can more easily determine where the code for a certain functionality is located. Care was also taken to explain the codebase structure in the program's documentation.

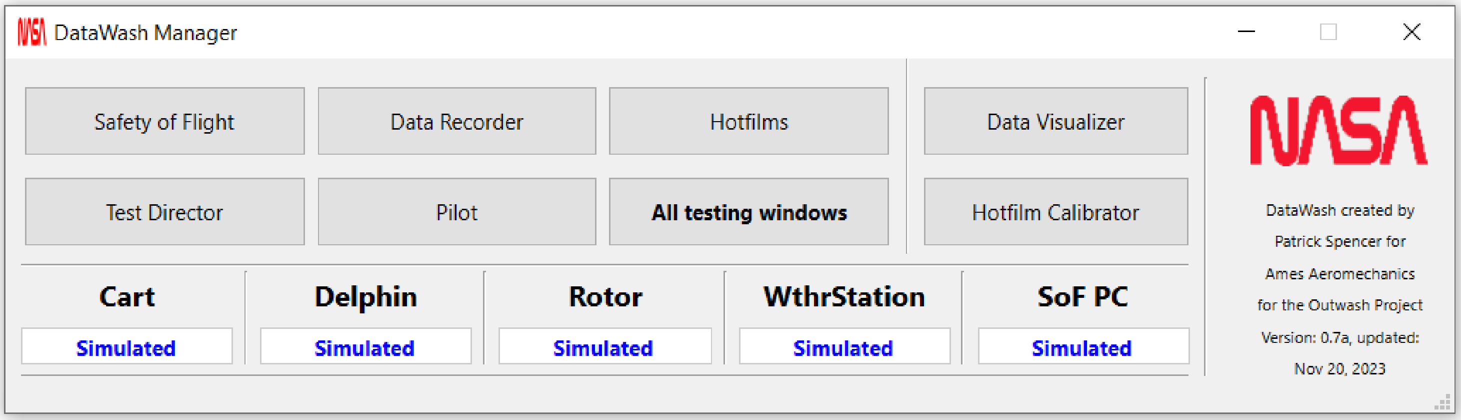
Results: Testing Windows

Term Definitions

Two units of data collection are the run and point. A point is a sequential collection of sensor data over some short time period, usually a few seconds. A run is a collection of points (which are usually acquired sequentially) for which a single variable is varied in a sweep.

Raw data is the exact numbers supplied to the program. Most raw values are in volts, but some are already in their respective engineering units (because it was for some reason more convenient to do the conversion on the device that collected the data). Processed data is always in engineering units, and is created from raw data by applying sensor conversions (for those raw values that are in volts - no conversion is needed if a value is already in engineering units) and zeros (offsets). Engineering unit and processed data are used interchangeably elsewhere in this document.

All user-configurable options are set in the config file.



DataWash created by Patrick Spencer for Ames Aeromechanics for the Outwash Project Version: 0.7a, updated: Nov 20, 2023

Fig. 1: Manager Window

Manager

The manager window (Fig. 1) is the central window of the DataWash program. It appears when the program is started, and cannot be closed until it is the only window open of the program. All other windows are launched from here. The windows are divided into the during-testing windows (on the left) that must all be displayed while a test is going on, and the other windows (on the right) that are used pre- (Hotfilm Calibrator) or post-test (Data Visualizer). All of the during-testing windows are described in this section, the others are described in the Pre/Post Test Windows section. The statuses of various data links are displayed below the buttons to open windows and automatically update (in normal use they read 'Connected' or 'Not Connected'; 'Simulated' is only for program testing and development). On the right of the window is general information about the program and version being used.

Safety of Flight

The Safety of Flight window (Fig. 2) is for monitoring safety-critical test conditions. There are plots for loadcells, accelerometers, resistance temperature detectors (RTDs), and RPM. The y-axis on each plot is a percent of a user-specified maximum, and on each plot there is a red line and a yellow line at user-specified percentages of the maximum. When bars pass these lines, their color changes to yellow or red. Various values are calculated from all the data collected in a preceding user-specified period (usually 1 second): maximum, minimum, half-peak-to-peak, and average; and some of these are displayed on each plot, depending on which are safety-relevant for each sensor. Each plot has textboxes below it (one per bar per sensor) that show the engineering unit value of each bar. Note that this is the only location in the program where live-displayed data is processed; elsewhere it is always the instantaneous sensor value. The update rate of the plots and textboxes is user-configurable.

Users can submit events by clicking 'submit event'. This saves the current time and all of the most recently collected data (for every sensor, not just those displayed on Safety of Flight) to a log file. 'Start/stop Recording' starts or stops recording. While recording, collected data and its timestamp is saved to a log file at a relatively low rate, so that recording can be continued for hours without files getting too large. This log is intended to go for the entire period that testing is being carried out, so that if there is any safety event, during testing or not, data about it was recorded and can hopefully help to find a cause.

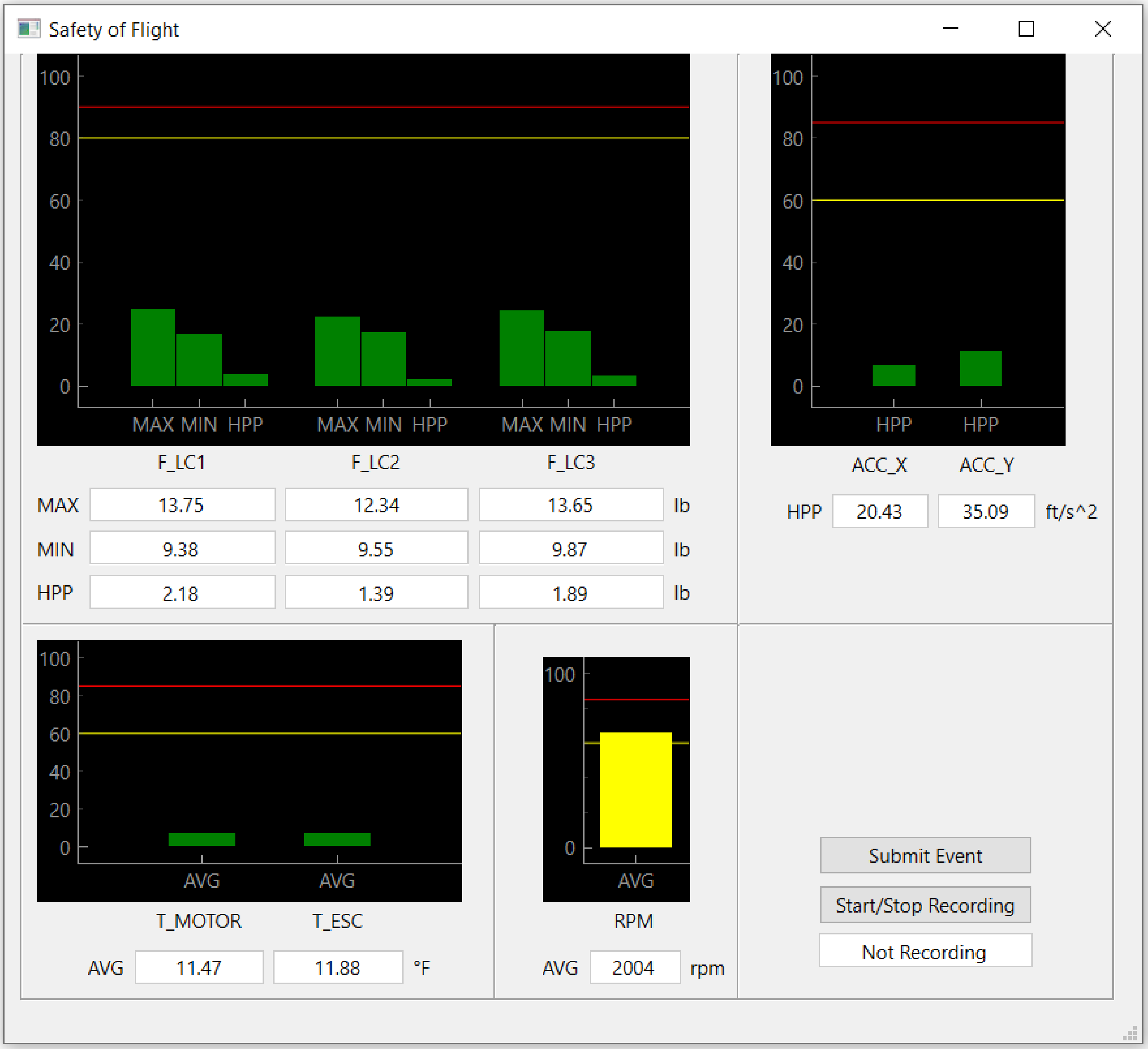
Test Director

The Test Director window (Fig. 3) is a read-only window that displays important test information to the test director. Various sensor values are displayed. The current value changes color between green, yellow, and red according to user-set limits. The central plot displays wind velocities from different types of anemometers; yellow points are hotfilms and red is ultrasonic. The x-range of this graph is user-set and the y-range is automatically changed based on what values the user sets for hotfilm heights when creating a new run in Data Recorder (described in Data Recorder section).

On the top-right of the window is the current run and point and information about the current run (its goal parameters that were set when creating it in Data Recorder). On the bottom-right is the current zero (run and point), if any, and the current hotfilm calibration, if any.

Pilot

The Pilot window (Fig. 4) displays information about the rotor status: servo positions, collective, torque, RPM, and current. The central plot is the target moment (blue) and the actual moment (yellow). The dimensions of the circle on the plot can be changed so that the pilot can use it as a limit condition



	F_LC1	F_LC2	F_LC3	
MAX	13.75	12.34	13.65	lb
MIN	9.38	9.55	9.87	lb
HPP	2.18	1.39	1.89	lb

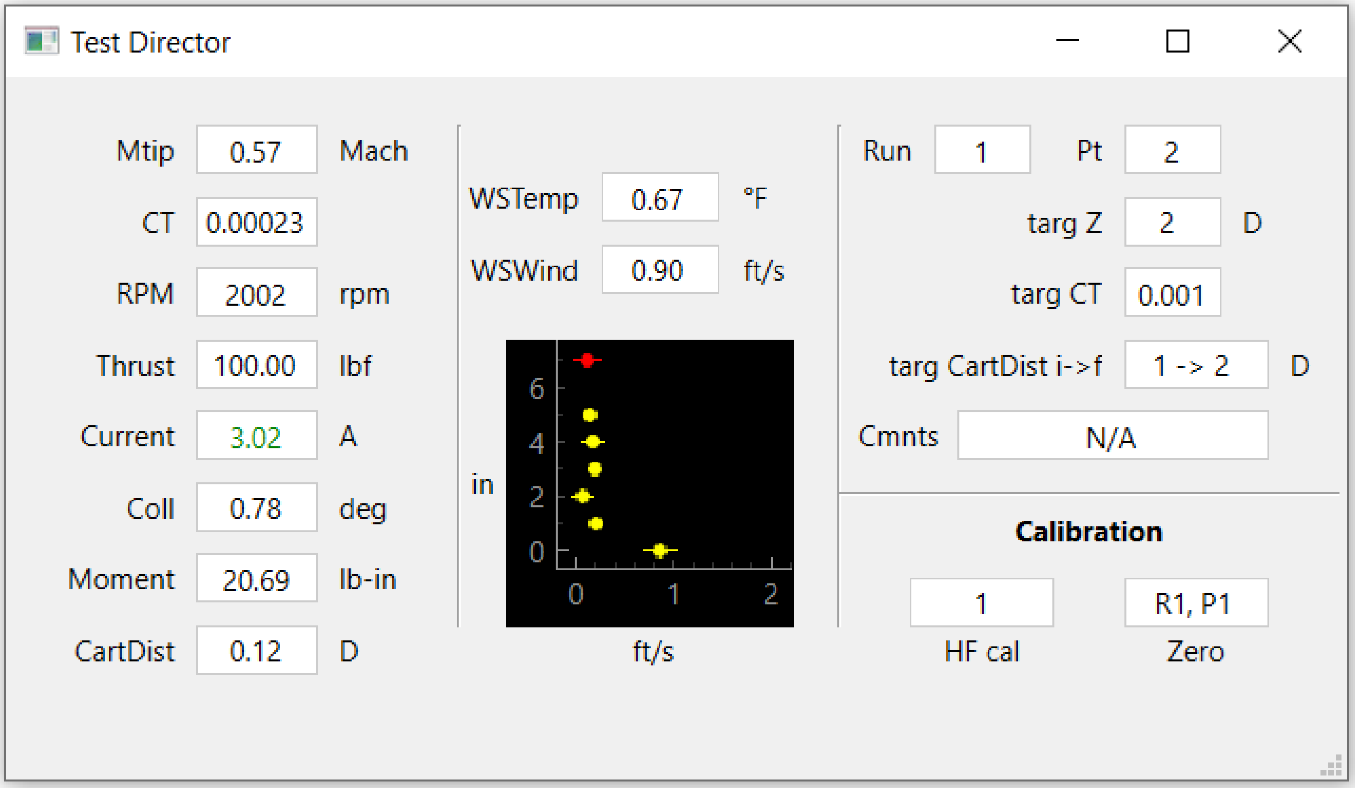
	ACC_X	ACC_Y	
HPP	20.43	35.09	ft/s^2

	T_MOTOR	T_ESC	
AVG	11.47	11.88	°F

	RPM	
AVG	2004	rpm

Not Recording

Fig. 2: Safety of Flight Window



Test Director

Mtip	0.57	Mach
CT	0.00023	
RPM	2002	rpm
Thrust	100.00	lbf
Current	3.02	A
Coll	0.78	deg
Moment	20.69	lb-in
CartDist	0.12	D

WSTemp	0.67	°F
WSWind	0.90	ft/s

Run	1	Pt	2	
		targ Z	2	D
		targ CT	0.001	
		targ CartDist i->f	1 -> 2	D
Cmnts	N/A			

Calibration

1	R1, P1
HF cal	Zero

Fig. 3: Test Director Window

or target. The plot can be zeroed (resetting the yellow dot to its center) with the ’Zero’ button. When clicked, a progress bar is displayed while data is recorded, and the offset is calculated as the average of the recorded data. This does not change any recorded data, only the appearance of the plot, and the magnitude of the offset applied is then displayed as MOffset. The refresh rate and plot range are user-set.

Hotfilms

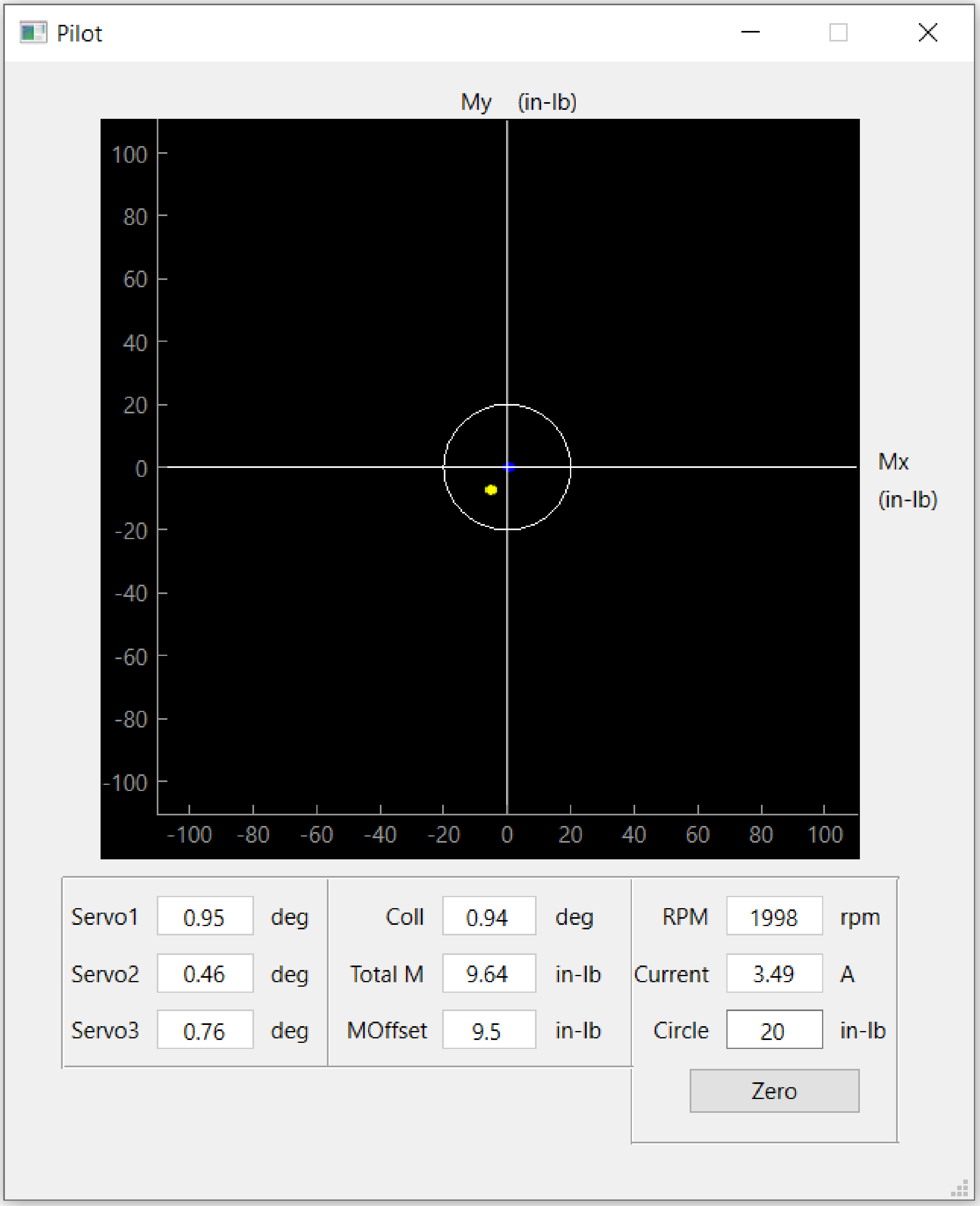
The Hotfilms window (Fig. 5) displays all anemometer readings on the left, a plot of these readings in the center (the same plot as in the Test Director window), and information about ambient conditions (largely from the weather station sensor) on the right.

Data Recorder

The Data Recorder window (Fig. 6) allows the user to collect data. On the left are various sensor values in addition to the current date and time. On the right is the data recording interface.

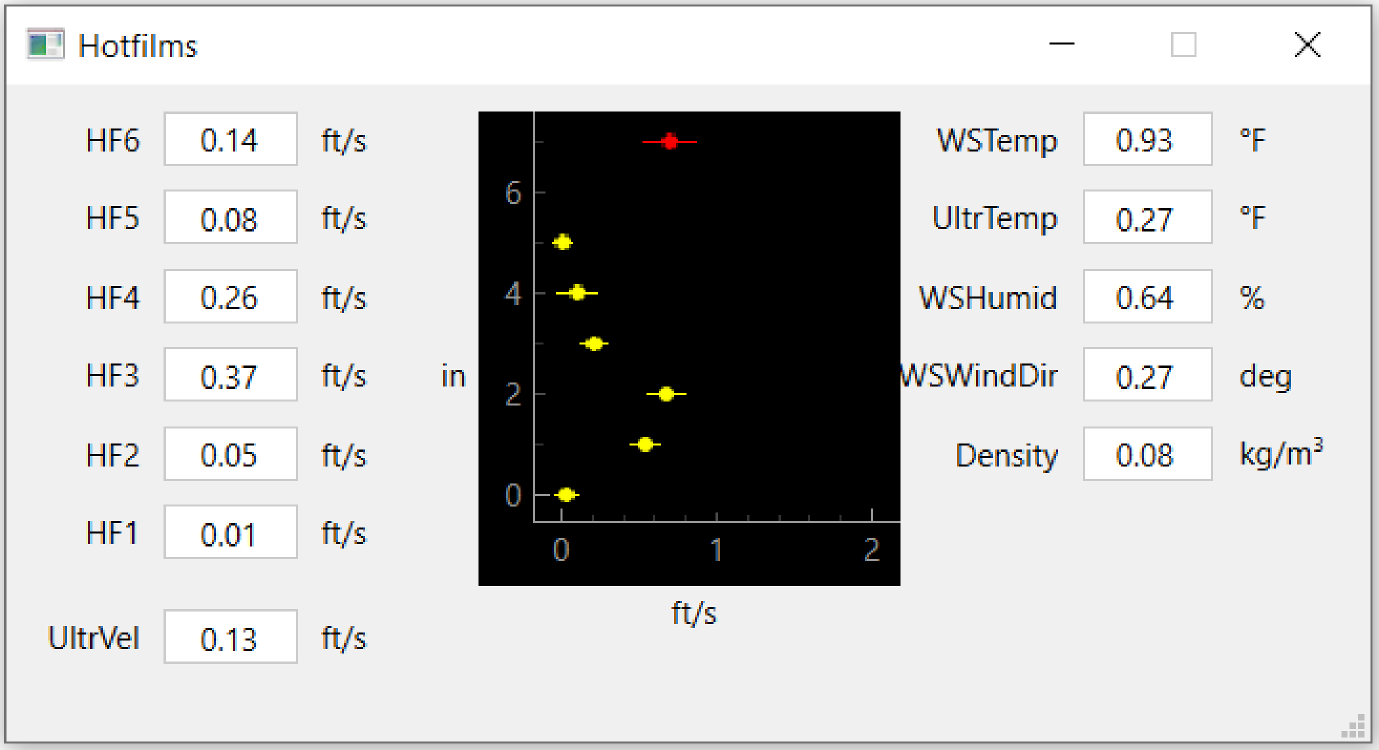
The first thing a user sees when using the program for the first time is the new run popup window (Fig. 7). This window automatically pops up if there are no runs stored, and can be accessed otherwise by clicking the ’New run’ button. The window allows the user to input goal parameters for the run (left, optional) and the height of each anemometer for the run (right, required). The hotfilm heights can be automatically filled-in with the previous run’s values, if they were not changed, by clicking ”Use prev. distances”. As goal parameters are filled in, the log at the bottom of the window is filled in. The run number can be set at the top of the window, and defaults to the smallest unoccupied run greater than the last / current run. Clicking ”Confirm” creates the new run.

After creating a new run, the run’s log name is displayed as ’Run Log’ at the top right of the Data Recorder window. This log name can be edited by editing the text and then clicking ’Save Logname’. The point number indicates what point will be recorded upon the next data collection, so it is always 1 after creating a new run. To collect data, a duration in seconds and data type must be specified. The data type defaults to ’Data’, which is the usual type. Collecting data in ’Zero’ mode creates new offsets for all of the channels that the user specified as zeroed channels. These offsets are applied to all future data collections



Servo1	0.95	deg	Coll	0.94	deg	RPM	1998	rpm
Servo2	0.46	deg	Total M	9.64	in-lb	Current	3.49	A
Servo3	0.76	deg	MOffset	9.5	in-lb	Circle	20	in-lb

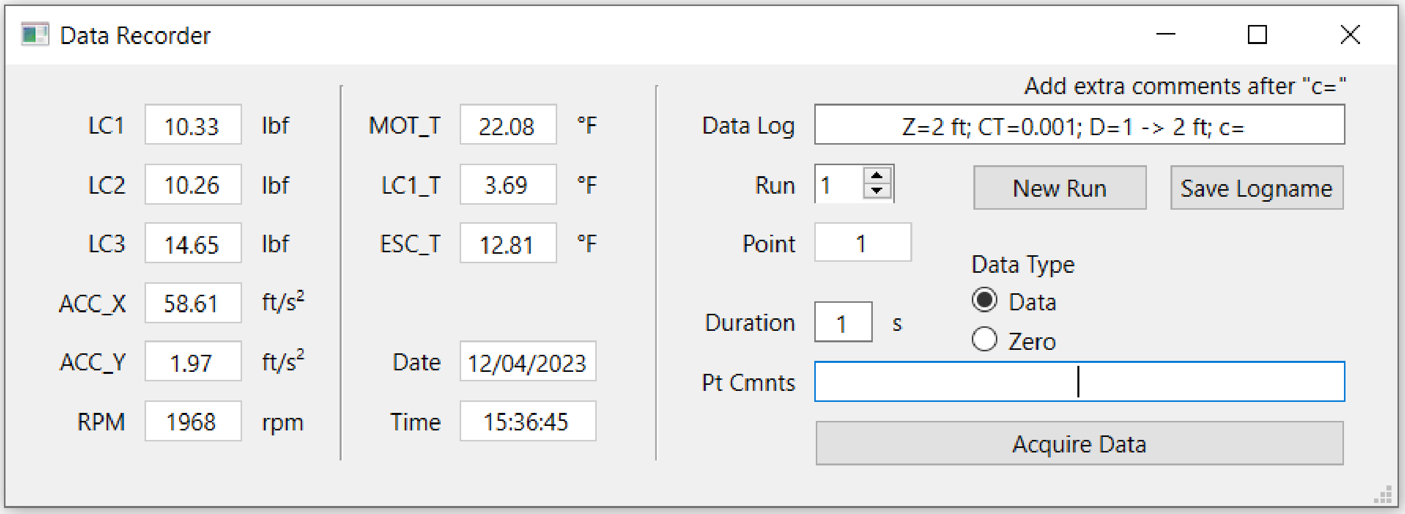
Fig. 4: Pilot Window


Hotfilms

HF6	0.14	ft/s
HF5	0.08	ft/s
HF4	0.26	ft/s
HF3	0.37	ft/s
HF2	0.05	ft/s
HF1	0.01	ft/s
UltrVel	0.13	ft/s

WSTemp	0.93	°F
UltrTemp	0.27	°F
WSHumid	0.64	%
WSWindDir	0.27	deg
Density	0.08	kg/m^3

Fig. 5: Hotfilms Window


Data Recorder

LC1	10.33	lbf	MOT_T	22.08	°F
LC2	10.26	lbf	LC1_T	3.69	°F
LC3	14.65	lbf	ESC_T	12.81	°F
ACC_X	58.61	ft/s^2			
ACC_Y	1.97	ft/s^2	Date	12/04/2023	
RPM	1968	rpm	Time	15:36:45	

Add extra comments after "c="

Data Log: Z=2 ft; CT=0.001; D=1 -> 2 ft; c=

Run: 1

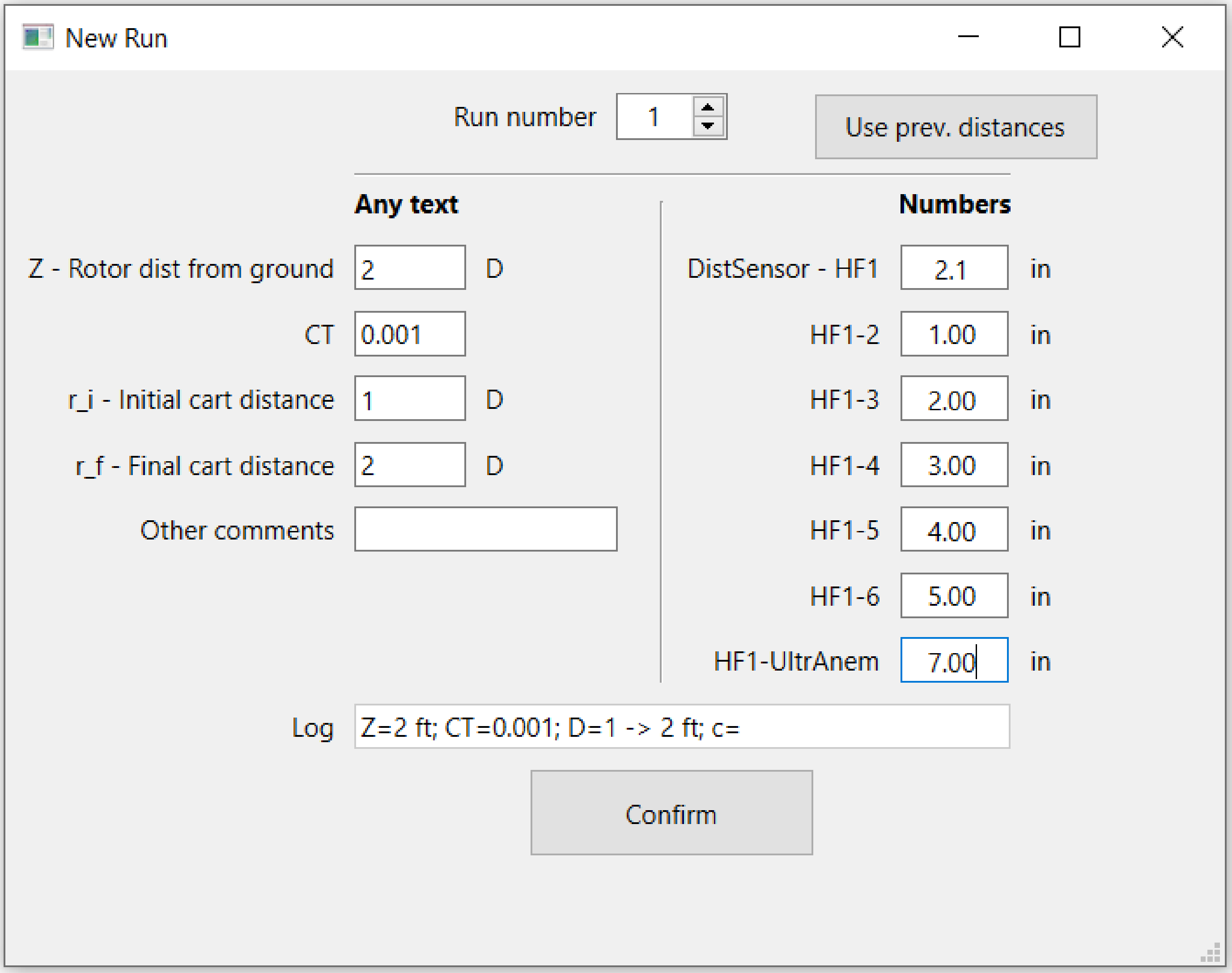
Point: 1

Duration: 1 s

Data Type: [✓ Data] Zero

Pt Cmnts:

Fig. 6: Data Recorder Window


New Run

Run number: 1

Use prev. distances

Any text			Numbers		
Z - Rotor dist from ground	2	D	DistSensor - HF1	2.1	in
CT	0.001		HF1-2	1.00	in
r_i - Initial cart distance	1	D	HF1-3	2.00	in
r_f - Final cart distance	2	D	HF1-4	3.00	in
Other comments			HF1-5	4.00	in
			HF1-6	5.00	in
			HF1-UltrAnem	7.00	in

Log: Z=2 ft; CT=0.001; D=1 -> 2 ft; c=

Confirm

Fig. 7: Data Recorder new run popup window

(only to the processed data) in the same run until a new point is collected in 'Zero' mode. Note that taking a point in 'Zero' mode does everything that 'Data' mode does, in addition to saving offsets.

The run can be changed by typing a new number or clicking the arrows of the 'Run' box. The run can only be set to run values that are already saved. The log name and point will automatically update as the run is changed. The 'Pt Cmnts' box is for comments about the point being collected, and is set to blank after every point collection. A progress bar is displayed during the collection of each point.

Results: Pre/Post Test Windows, miscellaneous

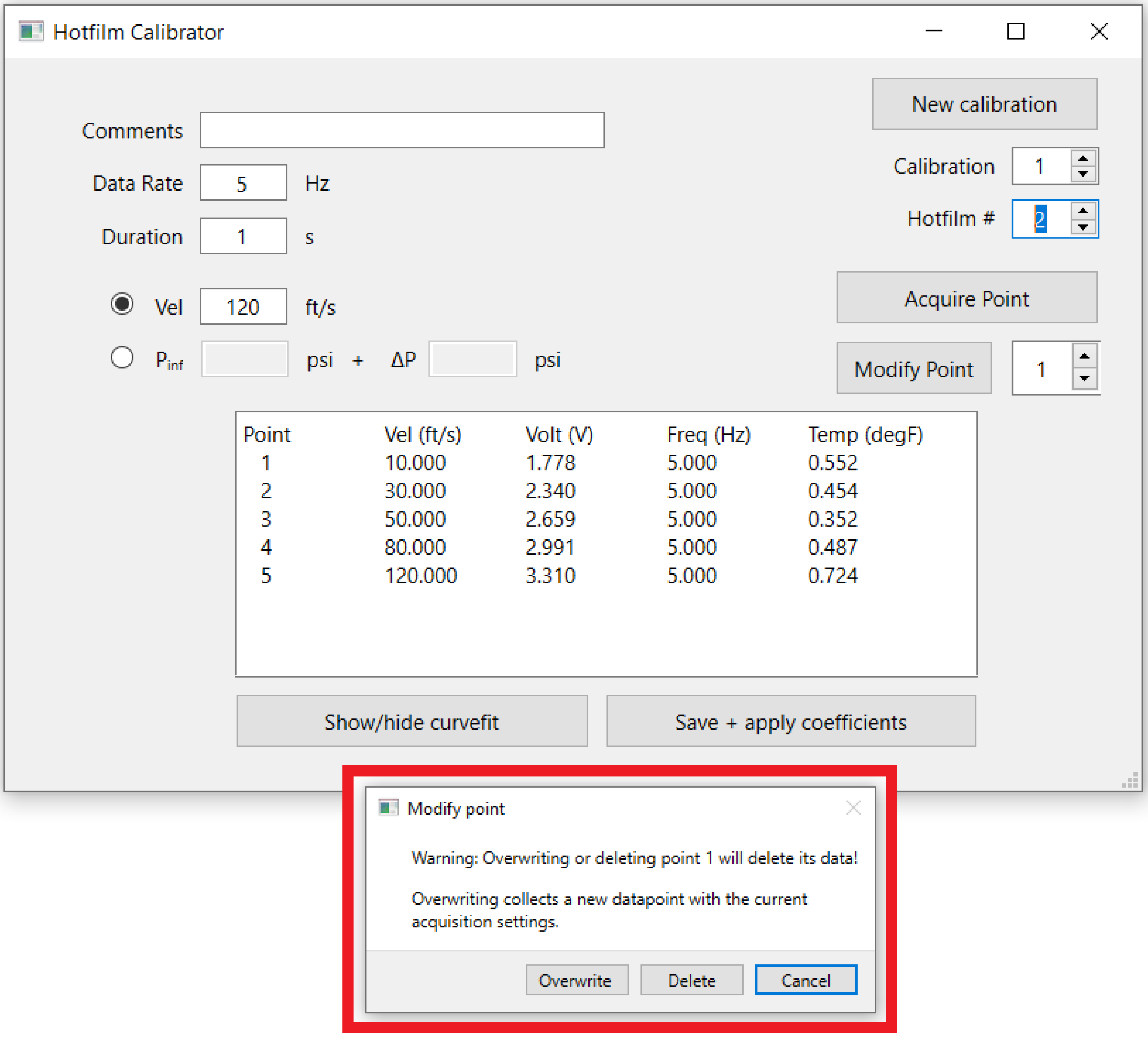
Hotfilm Calibrator

The hotfilm calibrator window (Fig. 8) lets the user calibrate each hotfilm. A blank calibration is automatically created when a user uses the program for the first time. Comments for each run can be entered in the 'Comments' box and saved by clicking 'Save Comment'. A new calibration can be created by clicking 'New calibration'. The calibration can be changed in the 'Calibration' box and the points and calibration comment will update automatically.

To collect a point, values must be entered in the 'Data Rate' and 'Duration' boxes. Additionally, values must be entered in either the 'Vel' box or both the 'Pinf' and 'P' boxes, depending on which option is selected (which textbox(es) are editable). After all required values are entered, clicking 'Acquire Point' collects a point, displaying a progress bar, and then shows it in the central text box.

At least five points must be collected per hotfilm for curvefit coefficients for that hotfilm to be generated. After at least five points have been collected, 'Show/hide curvefit' can be clicked and will show the curvefit generated from the collected points, as well as each coefficient and the R^2 value for the fit (Fig. 9). If the curvefit is not satisfactory, more points can be collected or points can be discarded via the 'Modify Point' button, which gives the option to either delete or overwrite any point. The current hotfilm can be changed by editing the value in the 'Hotfilm ' box.

Once all hotfilms have at least 5 points, 'Save + apply coefficients' can be clicked. This will save the curvefit coefficients for each hotfilm (calculating them if they have not been calculated already). Note that coefficients are not used in any way until they have been saved. The most recently saved coefficients are



Comments:

Data Rate: 5 Hz

Duration: 1 s

Calibration: 1

Hotfilm #: 2

● Vel 120 ft/s

○ P_{inf} psi + ΔP psi

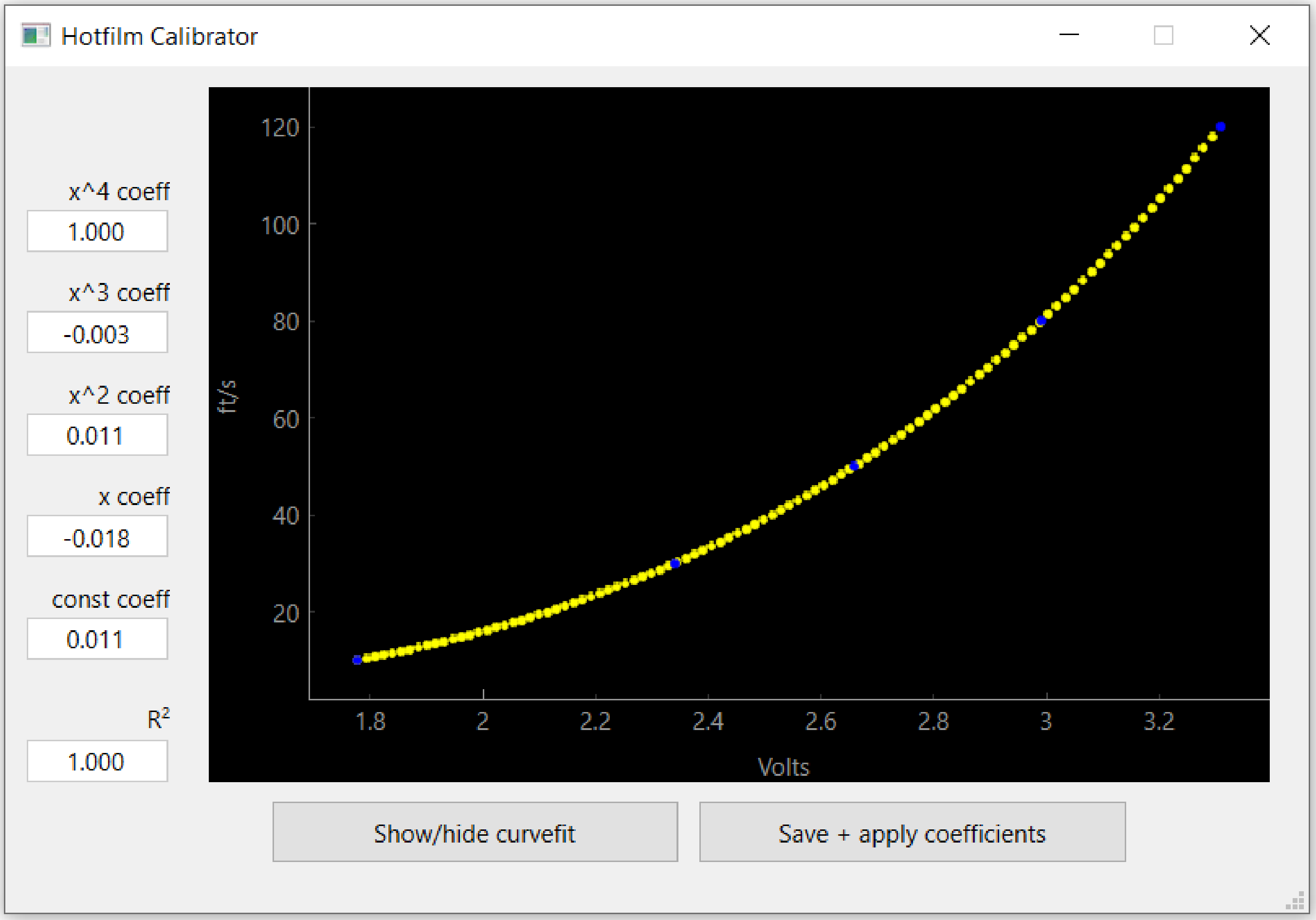
Modify Point: 1

Point	Vel (ft/s)	Volt (V)	Freq (Hz)	Temp (degF)
1	10.000	1.778	5.000	0.552
2	30.000	2.340	5.000	0.454
3	50.000	2.659	5.000	0.352
4	80.000	2.991	5.000	0.487
5	120.000	3.310	5.000	0.724

Warning: Overwriting or deleting point 1 will delete its data!

Overwriting collects a new datapoint with the current acquisition settings.

Fig. 8: Hotfilm Calibrator Window, inset is point modification popup


Hotfilm Calibrator

x^4 coeff
1.000

x^3 coeff
-0.003

x^2 coeff
0.011

x coeff
-0.018

const coeff
0.011

R^2
1.000

Fig. 9: Hotfilm Calibrator Window with fit shown

always the ones used, even if they are not from the current hotfilm calibration.

Data Visualizer

The Data Visualizer window (Fig. 10) lets the user view plots of collected data. Collected data stored in the default directory can be automatically accessed, and data elsewhere can be loaded via the ’Load file’ button. If the data is automatically accessed by its run and point, the run’s comment (if present) will be shown in the ’Run comments’ box. Data can be toggled between raw and engineering units (EU) by clicking the respective box on the left. Additionally, either each point in a run or each data point in a point can be displayed by selecting either ’x = points’ or ’x = time’. Enabling ’Sync x-axes’ causes all graphs’ x-axes to move in unison whenever any graph is panned. Specific channels can be enabled or disabled by toggling their checboxes on the left of each graph. Clicking ’Autoscale’ zooms each graph into the data it is currently displaying, according to which channels are shown or hidden. Entire graphs can be shown or hidden via the top-left dropdown menu. Note that no data can be modified in this window, only viewed.

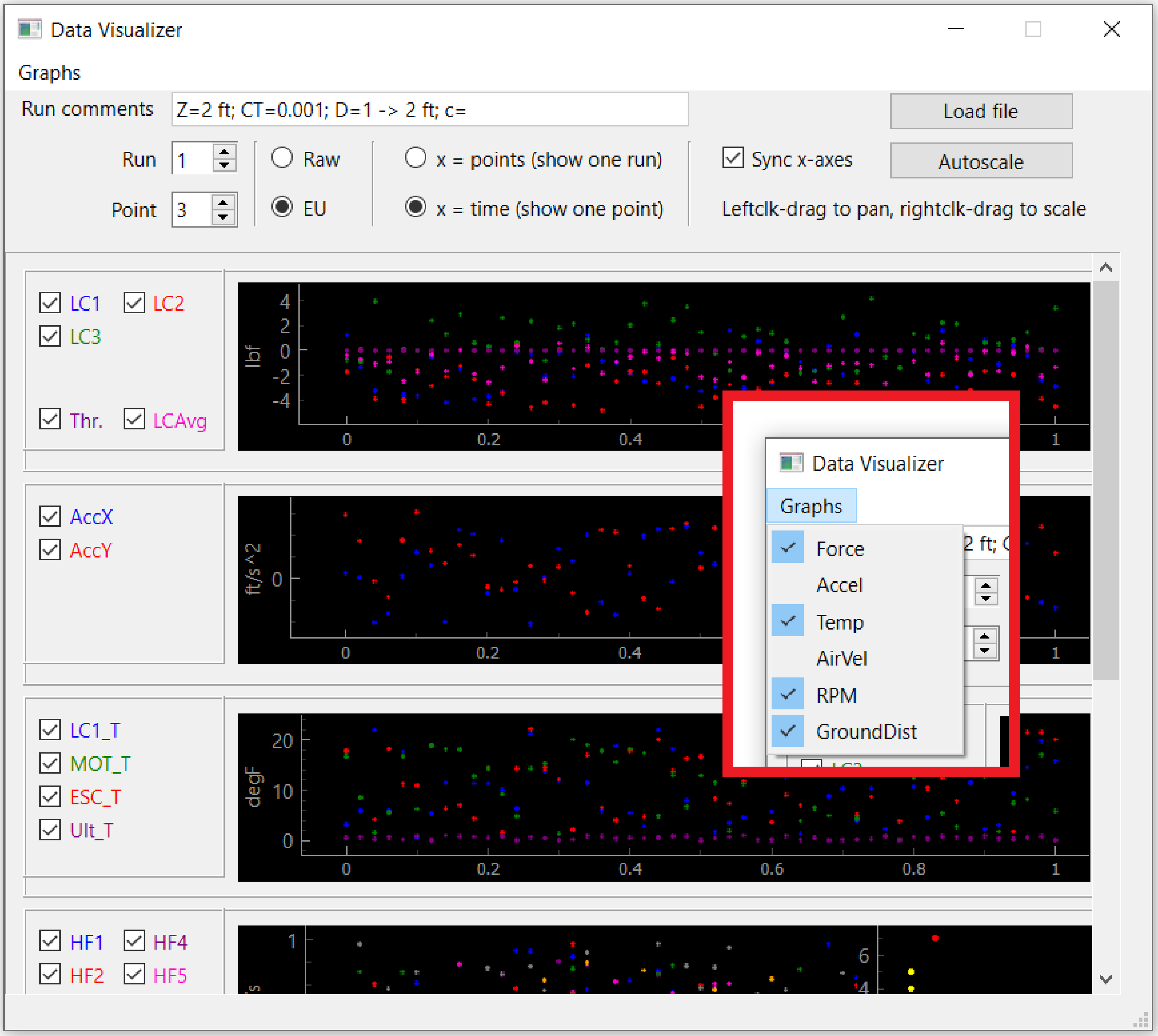
Configuration File

The main way for the user to configure various aspects of the program is the configuration file (Fig. 11 - note many options were removed for ease of reading). This is a text file with various keywords that change how the program works. Comments can be made with (comment) - anything following a is always skipped.

The option in the configuration file that most changes the functionality of the program is ’testingmode’. When this is set to true, no attempt is made to connect to other devices, and instead random data is generated for testing purposes. This is made clear by all device’s statuses in the manager window appearing as ’Simulated’ when this option is enabled.

Another important option is ’path’. By default, the program will create a folder ’outwash’ in the same directory as the program. If ’path’ is set, the ’outwash’ folder will instead be created there. This lets the user locate the data wherever they would like on their computer.

The sensor conversion section (top rectangle) is where the equation to convert raw voltages to engi-


Data Visualizer

Graphs

Run comments: Z=2 ft; CT=0.001; D=1 -> 2 ft; c=

Run: 1

Point: 3

Raw [✓ EU]

x = points (show one run) [✓ x = time (show one point)]

[✓ Sync x-axes]

Leftclk-drag to pan, rightclk-drag to scale

[✓ LC1] [✓ LC2] [✓ LC3] [✓ Thr.] [✓ LCAvg]

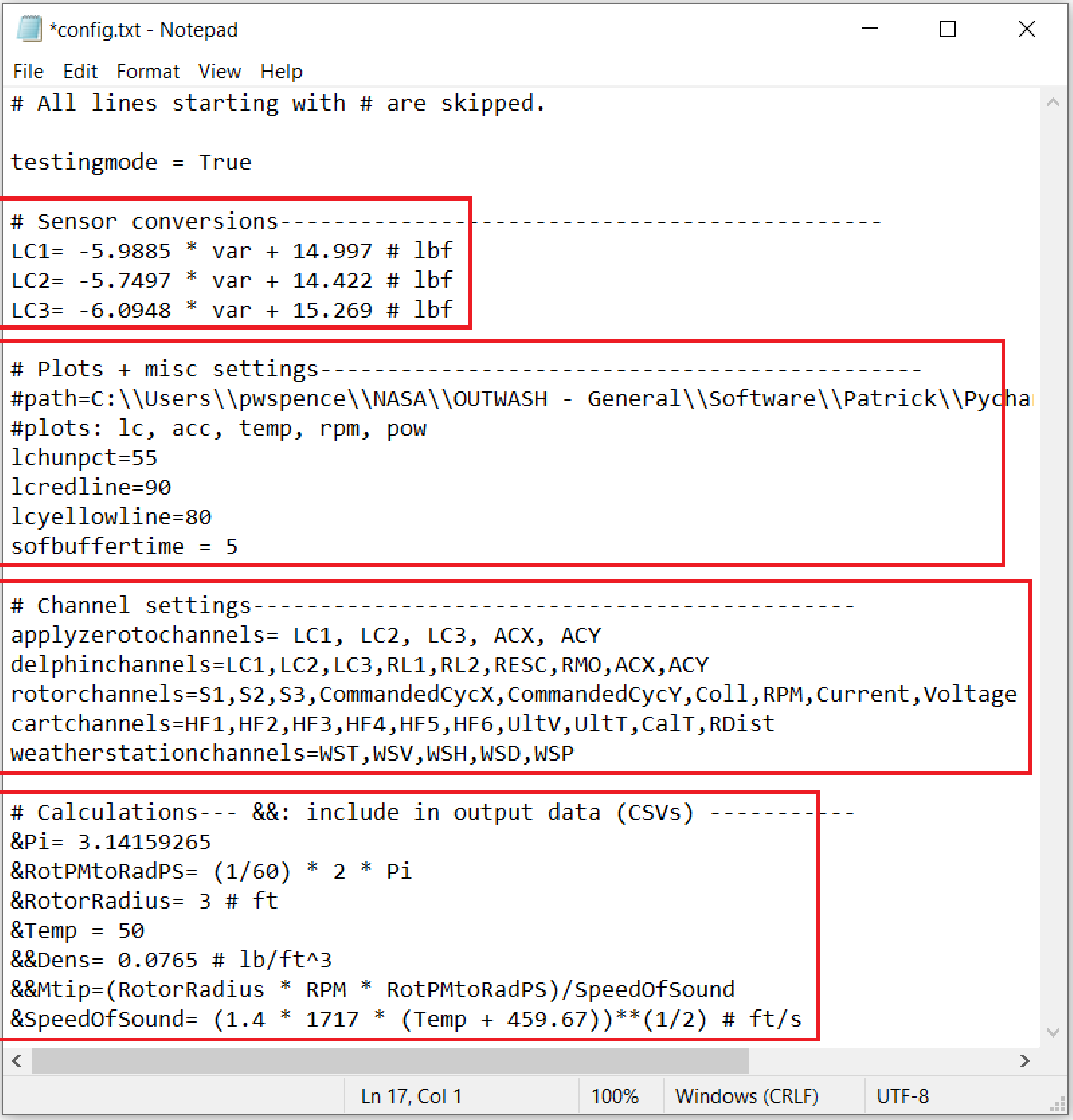
[✓ AccX] [✓ AccY]

[✓ LC1_T] [✓ MOT_T] [✓ ESC_T] [✓ Ult_T]

[✓ HF1] [✓ HF4] [✓ HF2] [✓ HF5]

Graphs: [✓ Force] Accel [✓ Temp] AirVel [✓ RPM] [✓ GroundDist]

Fig. 10: Data Visualizer Window, inset is graph selection drop-down



```
# All lines starting with # are skipped.

testingmode = True

# Sensor conversions-------------------------------------------------
LC1= -5.9885 * var + 14.997 # lbf
LC2= -5.7497 * var + 14.422 # lbf
LC3= -6.0948 * var + 15.269 # lbf

# Plots + misc settings----------------------------------------------
#path=C:\\Users\\pwspence\\NASA\\OUTWASH - General\\Software\\Patrick\\Pycha
#plots: lc, acc, temp, rpm, pow
lchunpct=55
lcredline=90
lcyellowline=80
sofbuffertime = 5

# Channel settings-----------------------------------------------
applyzerotochannels= LC1, LC2, LC3, ACX, ACY
delphinchannels=LC1,LC2,LC3,RL1,RL2,RESC,RMO,ACX,ACY
rotorchannels=S1,S2,S3,CommandedCycX,CommandedCycY,Coll,RPM,Current,Voltage
cartchannels=HF1,HF2,HF3,HF4,HF5,HF6,UltV,UltT,CalT,RDist
weatherstationchannels=WST,WSV,WSH,WSD,WSP

# Calculations--- &&: include in output data (CSVs) -----------
&Pi= 3.14159265
&RotPMtoRadPS= (1/60) * 2 * Pi
&RotorRadius= 3 # ft
&Temp = 50
&&Dens= 0.0765 # lb/ft^3
&&Mtip=(RotorRadius * RPM * RotPMtoRadPS)/SpeedOfSound
&SpeedOfSound= (1.4 * 1717 * (Temp + 459.67))**(1/2) # ft/s
```

Fig. 11: Sections in the config file

neering units is specified, per channel. Conversion equations can be entered in any format as long as they include ’var’ (which is the voltage received on that channel) and otherwise only includes numbers and operations (+, -, *, /, x**y = x^y).

The plot settings section (top middle rectangle) is where various plot settings can be specified. These include the maximum range and red and yellow line locations for Safety of Flight, the maximum range of the Pilot moment plot, the maximum x-range of the wind plots on Test Director and Hotfilms, and the buffer time for Safety of Flight (the preceding period in seconds that Safety of Flight values are calculated from).

The channel settings section (bottom middle rectangle) is where channels can be added, deleted, or reordered. In this section, each device connected to by the software has its own channel list. Some devices (depending on the communication protocol) must be configured so that their sent channel order is the same as their order here. Contained in this section is also the option to specify which channels to zero (when taking data with ’Zero’ mode in Data Recorder). Note that for each channel listed for each device, a corresponding sensor conversion must be present.

The calculated channels section (bottom rectangle) is where channels that are derived from sensor channels can be specified. Channels are composed in the same way as sensor conversion equations, except that the name of any sensor channel, and the name of any other calculated channel, can also be used and the current value of that channel will be used in the equation. Calculated channels are indicated by at least one & preceding their name; one indicates that the channel will not be present in program data output (.csv’s) and two indicates it will.

There is extensive error-checking and feedback for the settings in the configuration file, and the program will not start until the configuration file is set up properly. Some sensor channel and some calculated channel names are searched for by the program and thus cannot be changed without error. Others can be changed, and channels can always be added or reordered. Note that groupings of settings are only for readability - any order of settings will work.

Documentation for the configuration file is discussed in the documentation section, and the actual documentation is in the Appendix.

Other program information

File Structure

The file structure generated by the program (either in the same directory as the program or wherever the user specified, according to the 'path' configuration setting) is shown below.

```
outwash/
├── calibration/
│   └── zeros.csv
│       ├── hotfilmcalibrationdatapoints/
│       │   └── cal_001/
│       │       ├── HF1_P1.csv
│       │       ├── HF1_P2.csv
│       │       ├── ...
│       │       └── HF6_P6.csv
│       └── hotfilmcalibrations/
│           ├── cal_001_coeffs.csv
│           ├── cal_001_vals.txt
│           └── comments.txt
├── data/
│   ├── runlogs.csv
│   ├── AVG/
│   │   ├── EU/
│   │   │   └── Run_001.csv
│   │   └── RAW/
│   │       └── Run_001.csv
│   ├── EU/
│   │   └── Run_001/
│   │       └── P_01/
│   │           ├── CART.csv
│   │           ├── DELPHIN.csv
│   │           ├── ROTOR.csv
│   │           └── WTHRSTATION.csv
│   └── RAW/
│       └── Run_001/
│           └── P_01/
│               ├── CART.csv
│               ├── DELPHIN.csv
│               ├── ROTOR.csv
│               └── WTHRSTATION.csv
└── soflogs/
    ├── events.csv
    ├── EU/
    │   └── EU_SOF_11_47_54.csv
    └── RAW/
        └── RAW_SOF_11_47_54.csv
```

The 'outwash' folder is divided into three directories: 'calibration', 'data', and 'soflogs'. 'calibration' is where hotfilm calibrations and zeros are stored. 'soflogs' is where any data collected by Safety of Flight (either event logs or recorded ('indefinite') logs) is stored. 'data' is where all other collected data is stored. In the folders above, there is one run with one point recorded, and each hotfilm has a number of points recorded. Note that for timed logs, a file per device is saved. This is so that it is clear when data was recorded - data from different devices is received at different times, and those times are preserved by saving different device's data separately, with their respective timestamps. Note that 'EU' means 'engineering units'.

Codebase Structure

The program is split into many different files. XML files are used to define the graphical layout of each window, with all program functionality being in Python. Each window with a lot of functionality (Manager, Data Recorder, Hotfilm Calibrator, Visualizer) has its own Python file, while windows that mostly just display data (Test Director, Hotfilms, Pilot, Safety of Flight) are all in the same file. There is also a file that manages all data acquisition, and another for various functions that multiple windows make use of.

Documentation

The functionalities of the program, available configuration settings and arguments, and general program use instructions are all documented in a 'README' file, among other information about the program. This file can be found in the Appendix.

Error checking is present in many places in the program. In some cases the program does not let the user proceed with certain inputs (like letters in a box that should only contain numbers). In other cases, where the action the user wants to do may not be ideal but will not give an error, the program presents a confirmation dialogue to the user.

Data system integration

The program talks to a number of other devices: PLCs, the Delphin data acquisition system, and other computers. All communication is over Ethernet. The ModbusTCP protocol is used to communicate with the Delphin, the ModbusRTU protocol for the PLCs, and the STOMP protocol is used to communicate with all other computers. The health of each connection is periodically checked by the software and the results are displayed in the status boxes on the Manager window. This informs the user if something is wrong with a certain device or connection, so they can remedy the problem.

TuftVision

Requirements and goals

TuftVision is the software developed to automatically identify the angle of aerodynamic tufts. This capability was desired as a way to validate the wind direction reading of another sensor, and a way to gather wind angle data without any electronic sensor.

The program needed to take in a number of images, that each have a computer vision marker and a number of aerodynamic tufts, and output (in a CSV file) its best guess of the angle of each of the tufts relative to the computer vision marker. The program also needed to give the user a way to discard any bad angle guesses.

The robustness of the angle detection process was the major focus during development. The program needs to be robust to different lighting conditions, different image angles, different image resolutions, the aerodynamic tufts being located in different parts of the image, etc.

Results

The steps of angle detection are shown, in order, in Fig. 12, Fig. 13, Fig. 14, and Fig. 15.

Fig. 12 shows the source image. No cropping or processing of any type is required for source images. Fig. 13 shows the computer vision marker detection (the marker is boxed in blue). From the position of the marker, the area that the aerodynamic tufts are present in is determined (shown boxed in black). This area and the number of tufts to look for in it are easily modifiable; a 2x2 grid of tufts was used for testing but the final configuration will likely be 6 tufts in a vertical line. The dimensions of the tuft area are set by the user as multiples of the marker width and height, to account for different pixel dimensions of the area caused by different image resolutions and distances between the tufts and camera.

In the next step (Fig. 14) the tuft area is projected into a rectangle. This step isolates the tufts from the

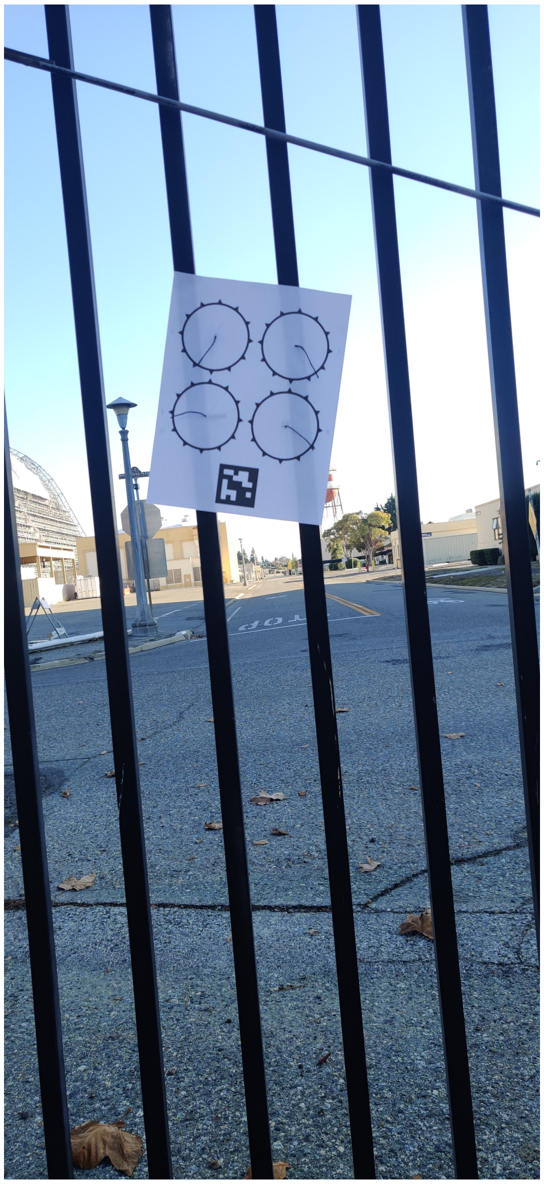


Fig. 12: Source image

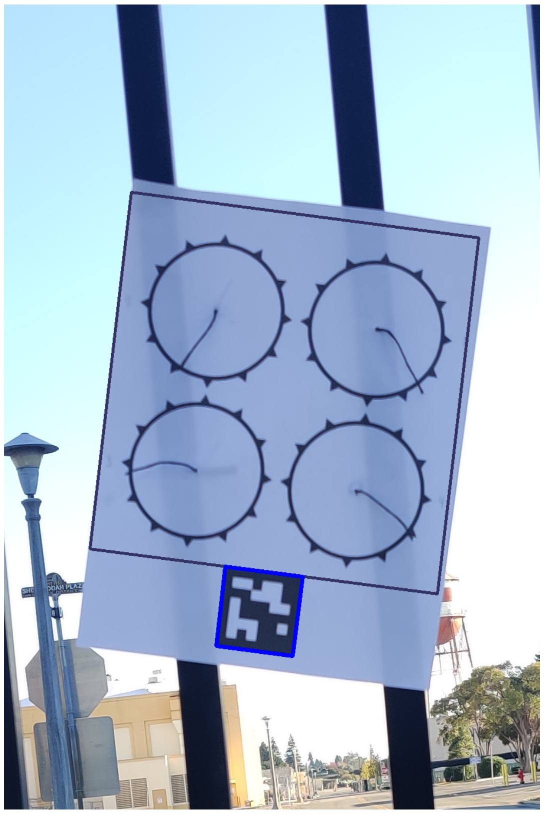


Fig. 13: Marker detection (blue), circle area determination (black)

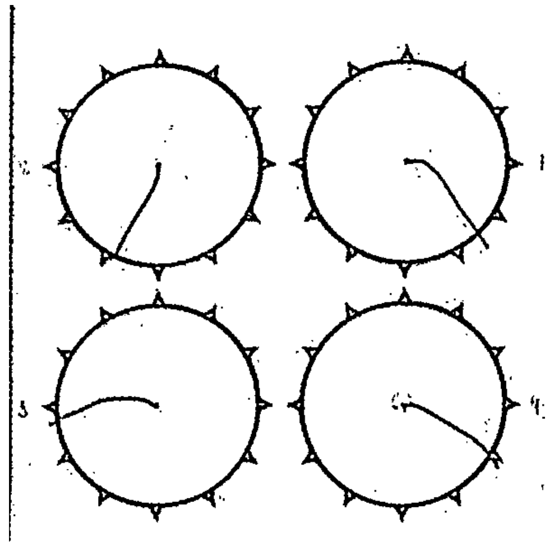


Fig. 14: Crop to circle area, image processing

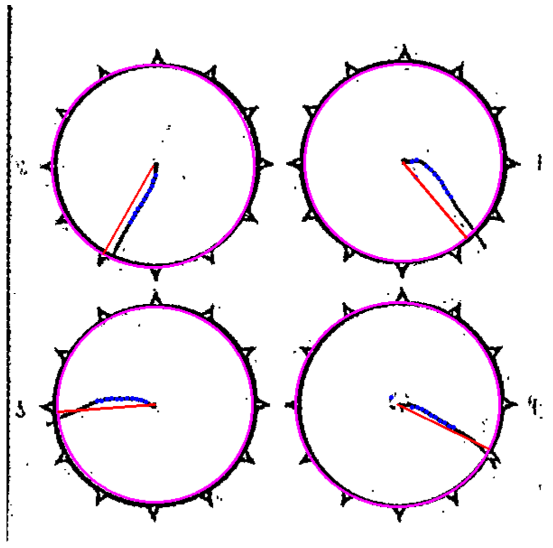


Fig. 15: Circle detection (pink), angle guesses (red)

rest of the image and accounts for image rotation and image squish (caused by non-perpendicular image angle). At this step, processing operations are applied to the image (blurring and thresholding).

In the final step (Fig. 15), any circles within a certain size range are detected. Then, for each circle detected, it is determined what the darkest azimuth is in a number of circles expanding out from the center of the detected circle. For example, with the radius of the detected circle as R_c, the darkest azimuth in a circle of radius $0.05R_c$ is determined, then the darkest azimuth in a circle of radius $0.1R_c$, and so on. These points are highlighted as blue in the figure. Using all of these radii-azimuth pairs, an angle guess can be made. A naive method would be to simply average all of the azimuths, but this is not very robust to erroneous point detections - a single point at a drastically different azimuth from the rest could have a large impact on the final guess. Instead, the program looks at the angle to each point *from the furthest-out dark azimuth* (the azimuth detected at the largest fraction of the circle radius), and takes the average of those angles. This means that even if points are erroneously detected near the center of the circle, the final guess will not be changed by much. Note that this explanation is only a simplified overview of the detection process, and at every step there are various checks to discard outliers, check for erroneous detections, validate that the detected circles are in the expected locations, etc.

Other work

Sensor interfacing and testing

Work was done to interface with and characterize two sensors: an ultrasonic anemometer and a time-of-flight distance sensor. Code was written to interface with each and then each sensor was tested and characterized. The distance sensor was tested by moving it closer and further from a wall and looking at its measurement standard deviation as a function of wall proximity. The measurement period (time over which its internal samples are averaged) was also varied. In both cases, the standard deviation of the sensor was observed as a function of the varied parameter.

The results of the distance sensor characterization are in Fig. 16 and Fig. 17. The standard deviation was found to increase significantly at shorter distances, but stays below 0.15 cm (1.5 mm) even at very short distances, which is acceptable for its use in the project. The standard deviation was also found to increase significantly at shorter measurement periods, but once again stays relatively low, below 2.5 mm in the worst case. Because the distance sensor data is not needed at a high rate, a longer measurement period of around 100 ms will be used to improve accuracy.

The ultrasonic anemometer was characterized by its proximity and orientation relative to a flat surface, because there were concerns about the sonic pulses generated by the sensor reflecting off of surfaces and increasing its measurement error. Thus, standard deviation was examined in different surface proximities and orientations. Outwash and downwash was also measured at various distances away from the wind source to ensure the sensor can accurately measure low windspeeds. In both cases, the wind source used was a tabletop box fan.

The results of the anemometer characterization are in Fig. 18 and Fig. 19. The collected outwash and downwash data seems reasonable and the trend for both is as expected, indicating the anemometer is capable of accurately measuring windspeed even when it is relatively low. The anemometer's proximity and orientation relative to a surface seem to have some impact on its standard deviation but it is not a large effect, which means it is safe to use very close to the ground if desired.

Conclusions

The goal of my internship was to develop graphical user interface, data acquisition, and computer vision software for the Outwash project. Software that meets all of these goals, and additionally is robust, configurable, and future-modifiable, was delivered successfully. Developing this software in-house turned out to be a good choice: In-house development allows for a lower monetary cost (at a higher time cost) and produces a program that is much more specialized to the specific project it is associated with than a commercial solution would be. Using Python, a completely free and very popular language (many specific libraries used were similarly chosen for their large userbases), means there will always be someone available to maintain and improve the software, and no license will ever be required for it.

Acknowledgments

The author graciously thanks the Kentucky Space Grant Consortium for making this work possible by providing the funding for it. The author specifically thanks Jacob Owens for being a helpful and considerate POC of the KYSGC.

The author sincerely thanks Dorsa Shirazi for her amazing mentorship and genuine interest in the author's wellbeing, success, and career development.

The author thanks Bill Warmbrodt for his mentorship, various tour opportunities, history lessons, and reminders to work just a little less hard.

The author thanks the managers of the Aeromechanics intern program: Alyna Anderson, Kristen Kallstrom, Mickey Radotich, and Linda Umana-Galo; as well as the Ames internship program team, for their guidance and significant time investment into each intern.

The author thanks the entire Aeromechanics Office, and especially the Outwash team, for their overwhelming friendliness and kindness.

Finally, the author thanks all of his fellow Ames interns for a wonderful semester of adventures.

Appendix

Datawash Documentation

```
DataWash created by Patrick Spencer for the Outwash Project
Last Update: Version 0.7a on Nov 20 2023

- General:
  - Data types:
    - Raw: Exactly what is sent to the computer. No offsets, conversions, or filtering. Units are volts for
        analog sensors or the respective engineering units for digital sensors. All secondarily
        calculated channels are 0.
    - EU: Engineering units. May have offsets applied from data collections in zero mode. Likely in each
        sensor's respective engineering unit rather than volts.
  - Modifying files:
    - The program needs to be restarted after changing any config settings or editing any data that the
        program reads for the changes to take effect. Editing data that the program reads or writes while
        the program is open is not recommended and could cause overwriting of data or erroneous behavior.
      - Excel does not allow any other program to read or edit a file that is currently open in Excel, so
          the program checks for csvs it needs and prompts the user with an error message if any of them
          are open.
      - Text file can always be read or edited whether they are open or not, so prompting the user to close
          them is not possible. If a currently opened text file is edited by the program, it will not
          update in the text viewing program until reopened.
- Files:
  - config.txt must always be in same directory as datawash.exe.
  - The files created and used by Datawash can be saved anywhere by using the path config setting.
    - If using path=, format filepath like 'C:\\Users\\pwspence\\NASA\\OUTWASH - General\\outwash\\'
    - If not, the default location is 'outwash\\' in the folder that datawash.exe is in.
  - 3 folders will be created at the path: calibration, data, soflogs. The contents of each are explained
      below.
- calibration\:
  - hotfilmcalibrationdatapoints\: Data collected from hotfilm calibration.
  - hotfilmcalibrations\: Generated coefficients and values from hotfilm calibration.
  - csv: Channel averages of each 'zero' data collection.
    - All channels are stored in zeros. Only channels specified in applyzerotochannels config setting have
        an offset applied to them.
- data\:
  - AVG\: Channel averages of all data collections.
  - EU\: All collected engineering unit data for all data collections.
  - RAW\: All collected raw (voltage) data for all data collections.
  - txt: Comments and number of points for each data collection run.
    - Run logs format: Run X: Z=x ft, CT=x, D=x -\> x ft, c= x, Pts= x, HFDists = []
      - Z: Rotor distance from ground
      - CT: Coefficient of thrust
      - D: Initial and final distances of cart to rotor mast
      - c: Extra comments
      - Pts: Number of points in this run
      - HFDists: Distance from distance sensor to HF1, then distance from HF1 to HF2, HF1 to HF3, HF1 to
          HF4, HF1 to HF5, HF1 to HF6, then HF1 to Ultrasonic Anemometer (7 numbers)
- soflogs\:
  - EU\: Engineering unit values from each channel for each Safety of Flight recording.
  - RAW\: Raw (voltage) values from each channel for each Safety of Flight recording.
  - txt: Events submitted from Safety of Flight (timestamp and channel values).
- txt mandatory settings:
  - Notes on channel lists:
    - The name of any channel that is utilized by the program (graphed, displayed in textbox) cannot be
        changed.
    - The order of channel lists can be changed in most cases (exception: HF1...HF6 must always be
        sequential and in order) and will change the column order of saved CSVs.
    - Example of comma separated channel list: 'LC1,LC2,ACX'
    - **Channel names are case-sensitive!**
  - delphchannels=(comma separated channel list) Channels coming from the Delphin.
  - plcchannels=(comma separated channel list) Channels coming from the PLC.
  - cartchannels=(comma separated channel list) Channels coming from the cart.
  - calculatedchannels=(comma separated channel list) Channels calculated from other channels. Equations
      for calculated channels and their names cannot be changed (option to set equation in config.txt
      coming in future version)
  - applyzerotochannels=(comma separated channel list) Channels to apply an offset to after collecting a 'zero' datapoint in the Data Recorder. This offset only applies to EU data.
  - [plot]hunpct=(float) Sets the y-range of each Safety of Flight plot. Red and yellow lines are set as a percentage of this maximum.
    - [plot] can be 'lc', 'acc', 'temp', 'rpm', 'pow'
  - (channel) = func(var) for each channel specified in delphchannels, plcchannels, and cartchannels.
    - Channel functions use Python syntax, so mult = \*, div = /, y\*\*(x) = y^x.
- txt optional settings:
  - potatorange(int/float)(default=20): Range in all directions of potato plot, in in-lb.
  - windrange(int/float)(default=10): Maximum range of wind plot in hotfilm and testdirector windows, in fts.
  - testingmode(boolean)(default=false): If true, instead of connecting to external data sources, generates random vals for each channel.
  - [plot][color]line(defaults: red=85, yellow=60): Sets the red/yellow line for each Safety of Flight plot.
    - [plot] can be 'lc', 'acc', 'temp', 'rpm', 'pow'
    - [color] can be 'red', 'yellow'
- hfheights(list)(default=1,2,3,4,5,6): Defines heights of hotfilms in order from 1 to 6.
- ultranemheight(num)(default=7): Defines height of ultrasonic anemometer.
- sofbuffertime(num)(default=1): Preceding time in seconds that the SOF min/max/avg/hpp are calculated over.
- Calculated channels:
  - &(name)=func(channels): Calculated channel the value of which is not saved. Can use any other channel (calculated or not).
  - &&(name)=func(channels): Calculated channel the value of which is saved. Can use any other channel ( calculated or not).
  - Channel functions use Python syntax, so mult = \*, div = /, y\*\*(x) = y^x.
  - Note that some calculated channel names are utilized by the program to find certain values, so their names should not be changed.
- **Windows:**
  - DataWash Manager: Appears on program startup, window from which all other windows are launched.
    - The connection status of each data source is indicated below window buttons.
  - All testing windows button: Not a window. Launches Safety of Flight, Data Recorder, Pilot, Hotfilms, and Test Director.
  - Safety of Flight: Plots of loadcell, accelerometer, RTD, and RPM data.
    - Preceding period over which data is from can be changed with config option sofbuffertime.
    - Submit event: Saves a log of the current timestamp and all current channel values (raw and processed) into soflogs/events.txt (file not created until an event is submitted)
    - Start/stop recording: Records indefinite-length log of all channel values (raw and processed), at data acquisition rate.
  - Data Recorder: Timed logging of data and displaying data from various sensors.
    - Acquire data: Records a point over the specified duration.
    - Run, point: Displays current run and point and allows for editing of run (editing of old runs should be reserved for correcting errors). Changing the run changes the y-positions (heights) of the wind graphs in Hotfilms and Test Director windows. Current point is always (number of points in current run) + 1. Points cannot be modified once recorded.
    - New run: Creates a new run.
      - Run number: The current run number. This must be a new number (there is no saved run with this number), but it is not required to be sequential (any non-occupied number can be entered).
      - Left boxes (under run number): These relate to the run's position in the test matrix. **Taking care to ensure the correct values are entered is highly encouraged** , although they can be edited later.
      - Right boxes: These are parameters relating to hotfilm and anemometer positioning. These are required to create a new run. Putting a number in the second box will automatically update the other hotfilm boxes. If the automatically generated numbers are incorrect they can be overwritten . After exiting the new run window, these cannot be edited except by editing the runlogs.txt file .
      - All information entered is saved in data/runlogs.txt and non-sensor positioning information is displayed in Test Director and Data Recorder.
    - Save logname: This saves the logname in the editable box to the current run's entry in runlogs.txt. The intended usecase is to add extra comments or correct typoed test parameters. **The logname is not saved until this button is clicked.**
    - Data type
      - Data: normal data collection
      - Zero: normal data collection + after point, averages of each channel are saved in calibration/zeros .csv, and channels set with config setting applyzerotochannels are offset by the respective value in this file. Run and point of zero currently being used is displayed in Test Director.
        - Data in a zero never has an offset applied to it.
        - No offset is applied to any data until a zero is collected.
        - The most recently recorded zero is always used, irrespective of what run or point it was.
        - Zero is never applied to raw data.
  - Pilot: Displays information for controlling rotor including moment plot.
    - Zero: Takes 10 data samples over the course of a second and computes average X and Y moments. Takes
        these averages as a zero and applies offsets to the plot.
      - No data is changed/affected, only the display of it in the plot.
      - The magnitude of the offset is shown in the MOffset box.
    - Circle: Changes the diameter of the circle. Circle is intended to be used as a limit condition or
        accuracy target for the pilot. Circle can be removed by setting to 0.
    - Plot range can be changed with potatorange= config setting.
  - Hotfilms: Displays current hotfilm readings and associated data.
  - Test Director: Displays various test-critical data for the test director.
  - Data Visualizer: Analysis of test data after collection.
    - Graphs can be panned and zoomed with mouse.
    - Extra graph settings available by right clicking any graph, including exporting as image, manually
        setting range, gridlines.
      - Some of these (everything in Plot Options except grid) are nonfunctional because they are
          implemented by default by the plotting library but do not function in this use case.
    - Raw/EU: Switches the data type between raw voltages and engineering units.
    - x=points/x=time: Switches data x-axis between points (dots are the average of all data collected in
        one point) and time (dots are individual data collections in a certain point).
    - "Graphs" button in top left: Shows/hides graphs.
    - Load file: Pops up window to let user select file to display.
      - This is mostly intended to be used for looking at Safety of Flight and hotfilm logs, recorded
          points should be looked at by navigating to the right run/point number.
      - Changing any of: Run, Point, Raw/EU, x =, will cause the file to be unloaded and data will be shown
           from a recorded run, if any exist.
      - Run comments box will always be blank.
    - Sync x-axes: Syncs x-axes of all graphs.
    - Autoscale: Scales all graphs so their data takes up the full graph.
    - Buttons to left of graphs: Each channel can be shown/hidden.
    - Wind plot: The AirVel graph has a mini graph to its right that displays hotfilm velocities in the
        same format as the hotfilm plots in the Hotfilms and Test Director windows.
      - This graph will be displayed **if and only if** the data mode is EU and the x-axis is time.
      - It is not affected by sync axes.
      - It will be displayed if the above conditions are met and a file is loaded, but it may not make any
          sense (or be blank) and hotfilm heights will not be correct.
  - Hotfilm Calibrator: Calibration of hotfilms.
    - Two types of calibration available: input velocity or input ambient temp + ambient pressure +
        pressure delta.
    - Calibration and hotfilm boxes allow navigating between all saved calibrations and all hotfilms (
        whether they have points or not). After clicking New calibration, a calibration is only saved if a
         point is recorded in it. Modify point allows for overwriting or entirely deleting an erroneous
        point.
    - Show/hide curvefit: Calculates and displays, or hides, the 4th-order curvefit of the points in the
        currently loaded hotfilm, as well as the coefficients.
    - Save + apply coefficients: Calculates the coefficients, saves them in calibration/hotfilmcalibrations
        /cal\_xxx\_coeffs.txt, and applies them to the rest of the program. **No coefficients are ever
        saved automatically!**
      - Hotfilm values are -1 if they do not currently have valid coefficients.
      - On startup, the program looks for the coefficients file of the highest calibration that has at
          least 1 point recorded. If there is no such file, no coefficients are applied. Coefficients from
          older calibrations can be applied by loading the desired calibration and clicking the save +
          apply button.
      - Note that coefficients are per-hotfilm and some hotfilms may have valid coefficients while others
           do not. Each hotfilm needs at least 5 points for coefficients to be calculated.
        - If at least one hotfilm has enough points for coefficients but others do not, a message
             indicating the hotfilm did not have enough points is saved as its coefficients data.
- Bugs:
  - Send to {maintainer} on Teams!
- Protocols:
  - Data recording
    - New run: Click New run (if it is the first run it will automatically pop up on window opening). Enter
         in hotfilm height information. Optionally, enter in run target information. Click confirm.
    - Taking a point: Enter a duration. Select a collection type (data or zero). Enter in comments into 'Pt
         Cmnts' if applicable. Click Acquire Data. The point is automatically incremented and the comments
         erased.
    - Editing log: Edit text in 'Data Log'. Click Save Logname.
  - Hotfilm calibration:
    - New calibration: Click new calibration. Enter any comments.
    - Taking a point: Select desired hotfilm. Enter a data rate and duration. Select the appropriate test
        type and enter the required information (Vel or T0 + Pinf + deltaP). Click Acquire point.
    - Modifying/overwriting point: Click modify point. Select desired action.
    - Saving coefficients: With at least 5 points, click show/hide curvefit and verify good curvefit (
        Rsquared is below threshold). Collect more points if curvefit is not acceptable. Otherwise, hide
        curvefit and click save + apply coefficients.
- Maintenance:
  - To build a new version after code has been changed, do:
    - pip install pyinstaller
    - CD into directory containing main.py
    - PyInstaller main.py --onedir --debug all --icon=dwicon.ico
    - Move source and other required files (gui, icons, config, readme) into dist\main\
    - Zip all files in dist\main\
  - Code structure:
    - py: Highest-level program functionality. Window initializations, variables referenced by multiple
        windows, manager window functionality, textbox updating.
    - py: Data acquisition and processing functionality. No GUI stuff. Also parses config.
    - py: Safety of Flight, Pilot, Hotfilm, and Test Director windows (all of these windows are largely
        display elements with little or no interactivity).
    - py: Functions and classes that are used by multiple windows.
    - py: Data visualizer window
    - py: Data recorder window
    - py: Hotfilm calibrator window
    - gui\\: UI elements are mostly stored in .ui files (XML) here, created by QtDesigner. Some static UI
        is set in Python for ease of modification/updating. All dynamic / updating UI functionality is in
        Python. **QtDesigner should be used to edit .ui files.**
      - UI elements are accessed in Python by their name set in QtDesigner.
  - Adding/removing channels on DAQ devices:
    - Delphin:
      - See OUTWASH   General/Software/Vibro Documentation/Configuring ModbusTCP on Vibro.docx for config
      - The first channel must have an address of 0 and each following channel must be sequential. This is
          the order that the channels will be sent to the computer, e.g. if LC1 is set to addr=0 and LC2 is
           set to addr=1, config.txt should have delphinchannels=LC1,LC2,
      - There must be a buffer channel after the final channel (channel with the largest address). It does
          not matter what its content is (it will not be read by the computer). Recommended to set to
          constant=0.
    - PLC: TBD
    - Cart: TBD
    - Rotor: TBD
    - Weatherstation: TBD
```

Sensor characterization results

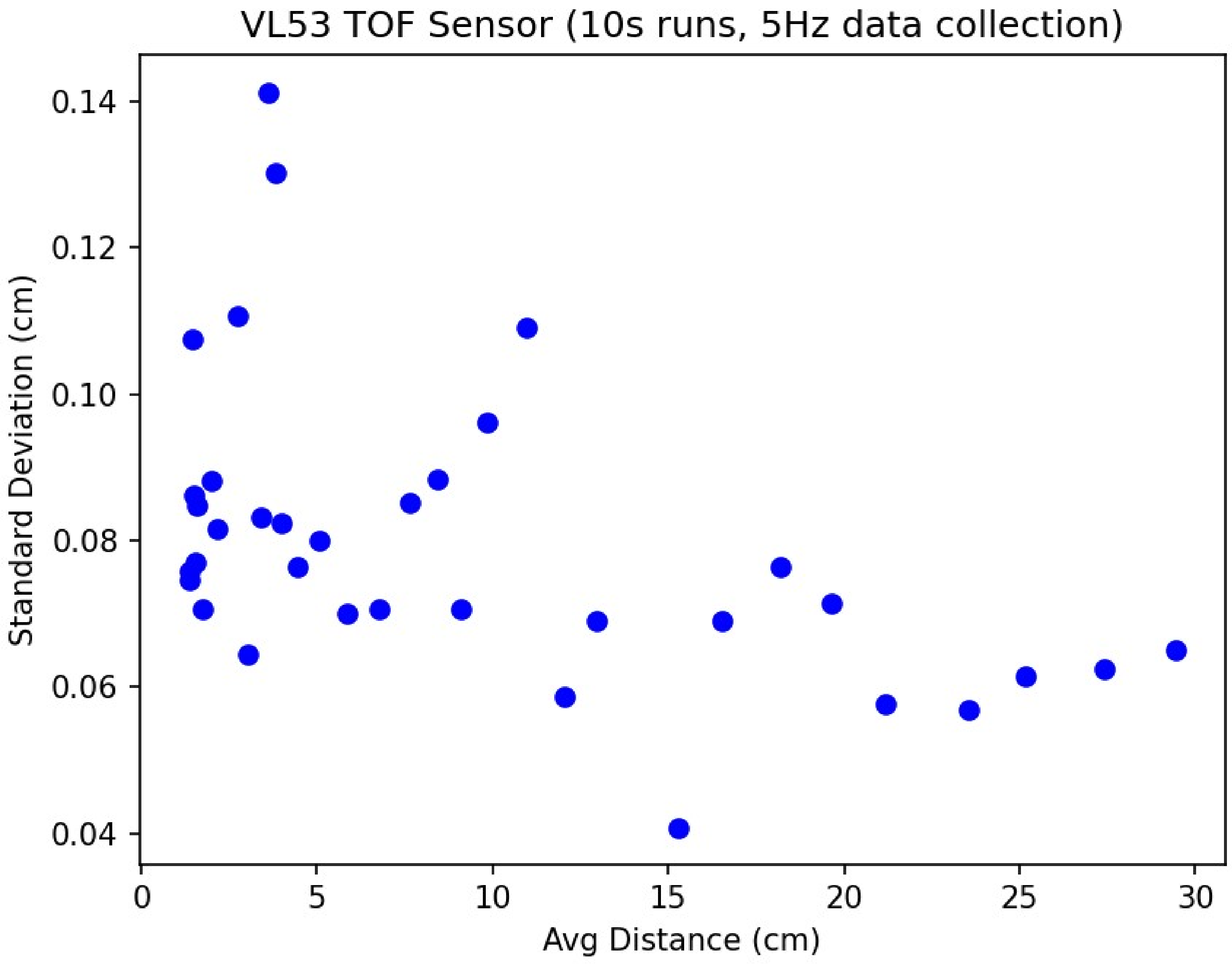


Fig. 16: Distance sensor standard deviation as function of distance

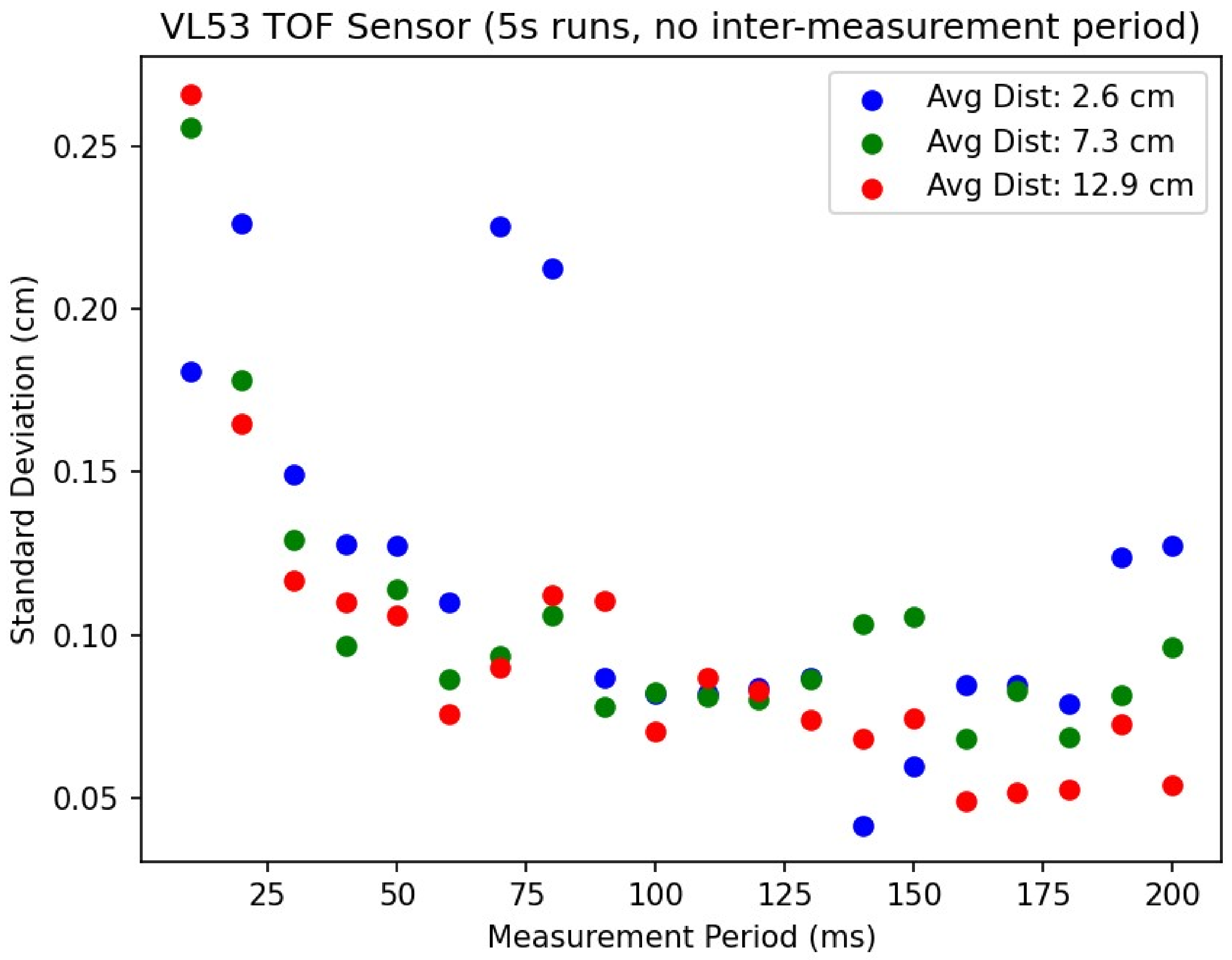


Fig. 17: Distance sensor standard deviation as function of measurement period

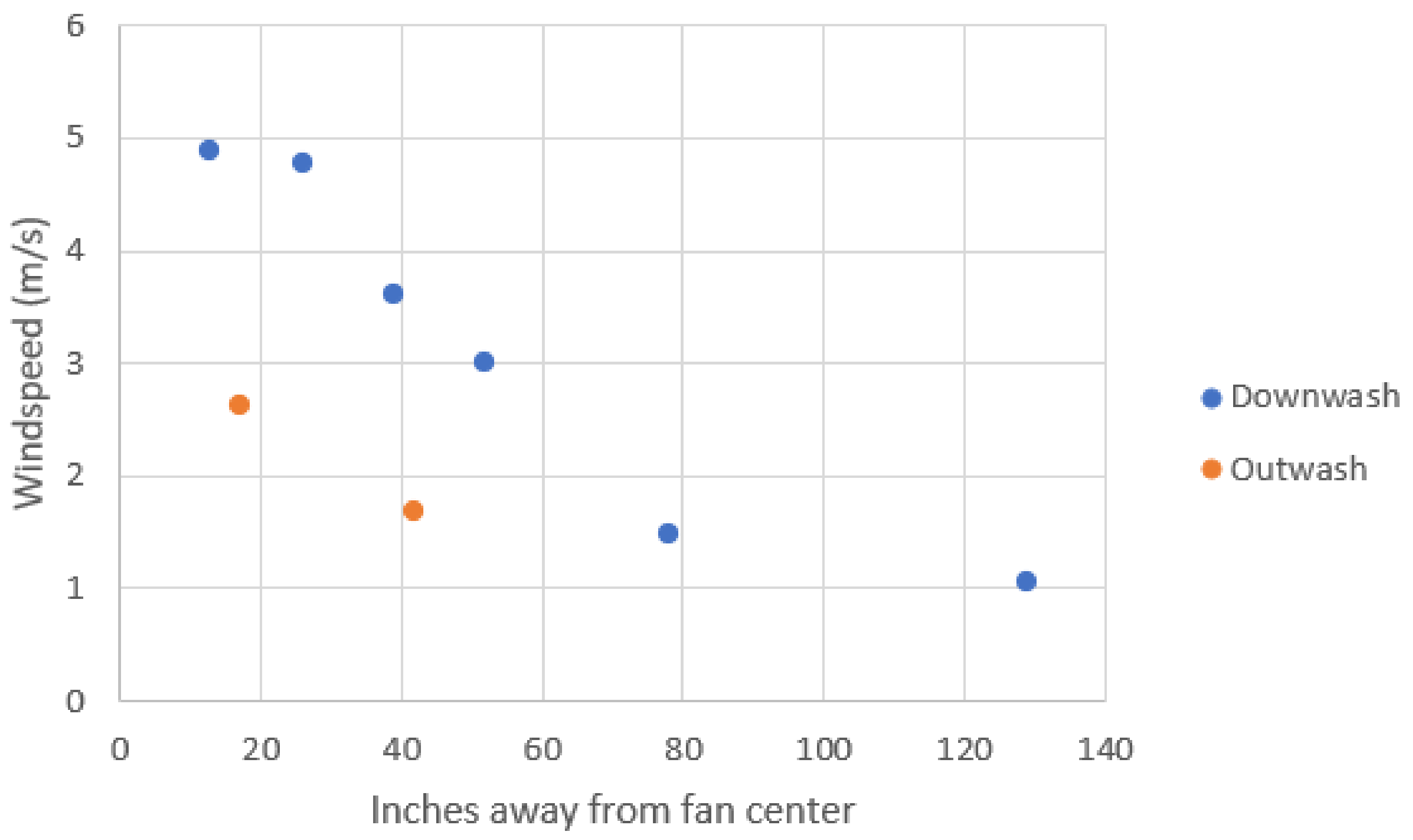


Fig. 18: Outwash and downwash data collected with ultrasonic anemometer

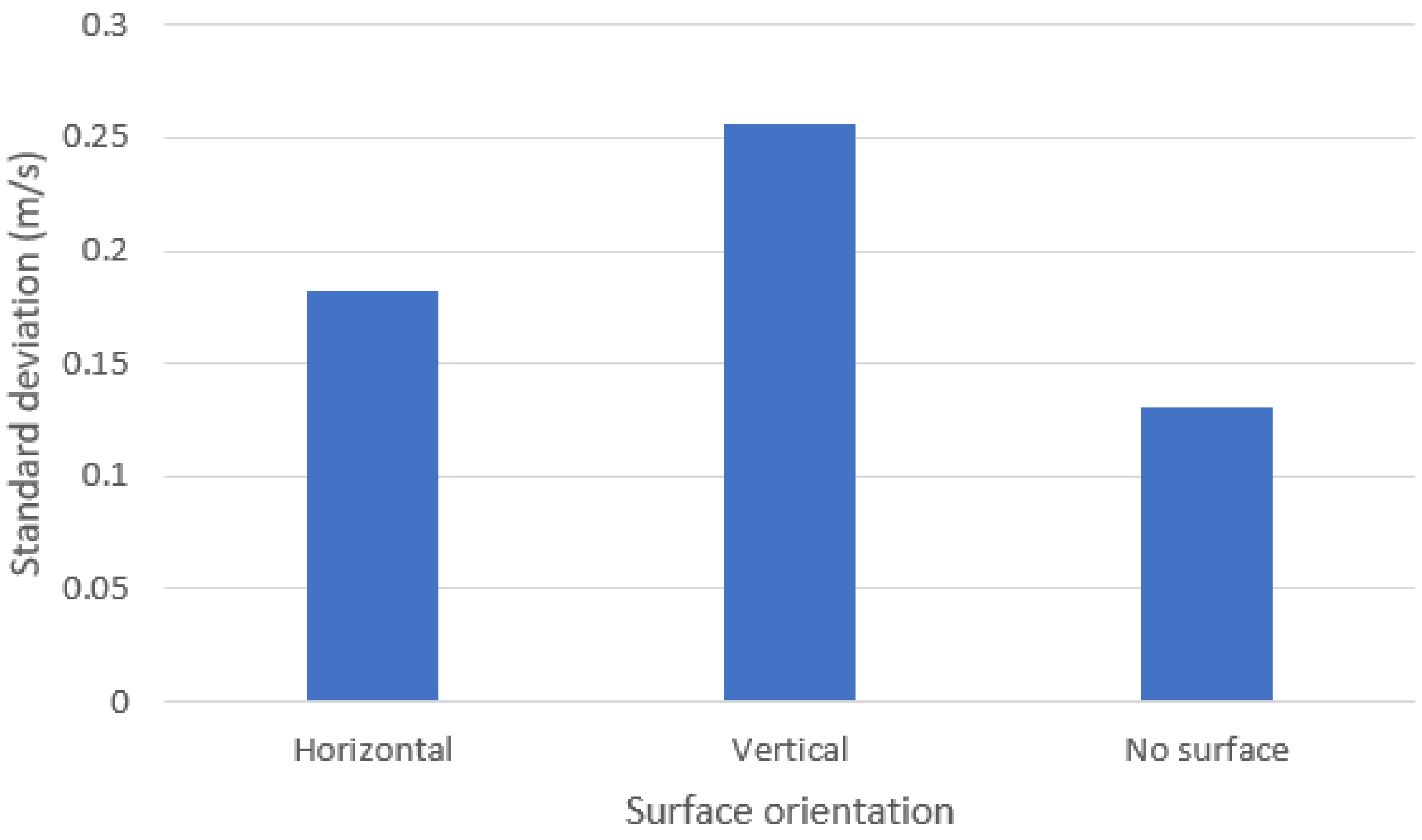


Fig. 19: Ultrasonic anemometer standard deviation as funcion of surface proximity and orientation