

User's Guide for CFcontrol_Waypoints

By Edward Mikalsen

Ohio University I.D.E.A.S. lab

Updated: October, 2019



Covered in this guide:

- Vicon setup and object creation
- Command station setup procedures
- CrazyFly set up procedures
- CrazyFly testing procedures
- CrazyFly flight procedures
- Analyzing CrazyFly flight data
- Troubleshooting communication issues
- Troubleshooting drone instability issues
- Some high-level python commands for the CrazyFly
- Programs: *CFwaypoints.py*, *demo.py*, *Find_position.py*, *PitchRolltest.py*, *cfControlClass.py* (briefly), *PID_CLASS.py* (briefly), and *SaveVarCF.m*

Not Covered in this guide:

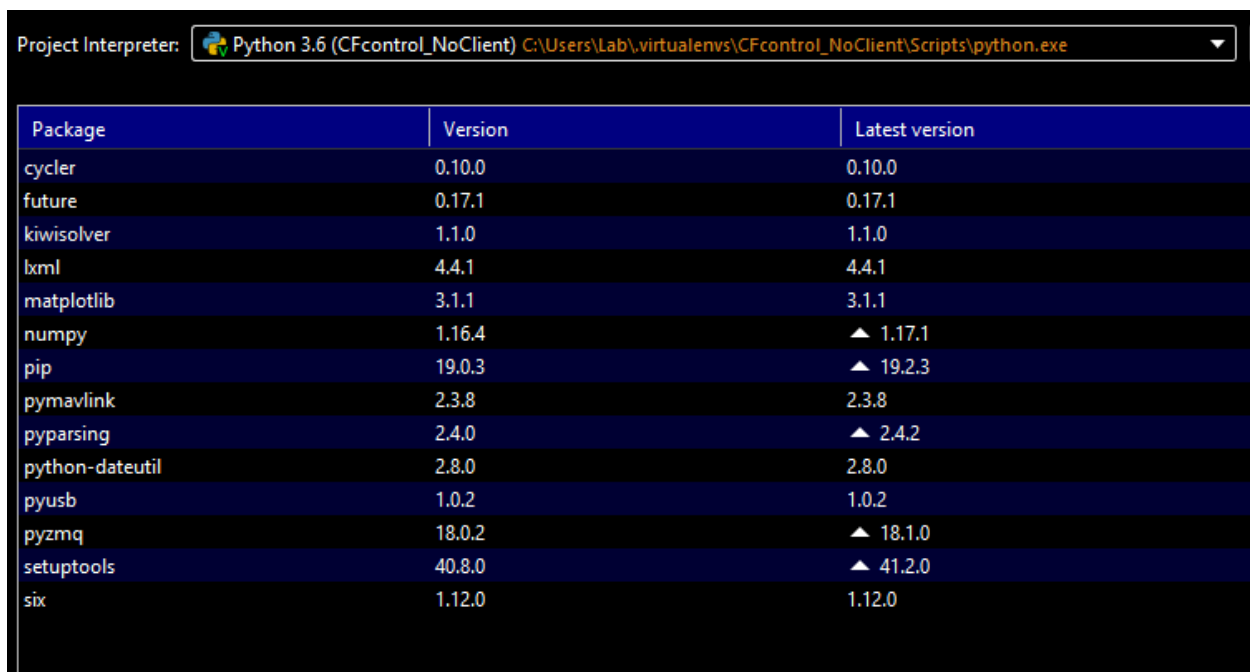
- Most advanced Vicon settings
- Python programming
- MATLAB programming
- Drone repair
- PID control
- Advanced electronic communication concepts
- Any program in the repo other than what is listed above

For any Issues During Parts I – III please refer to Part IV: Troubleshooting

Part I: Setup

Command Station

1. Download entire repo “CFcontrol_Waypoints” from GitHub. Note: Cfcontrol_Waypoints is based off of CFcontrol_NoClient, in that it does not require the CrazyFly Client to run.
2. Start a new project in your Python IDE
 - a. Make sure your interpreter is set to a 32bit version of Python 3 (recommended 3.6).
 - b. Install all the correct packages. Your project interpreter and packages should look something like this:



The screenshot shows a 'Project Interpreter' window with a dropdown menu set to 'Python 3.6 (CFcontrol_NoClient) C:\Users\Lab\virtualenvs\CFcontrol_NoClient\Scripts\python.exe'. Below the dropdown is a table of installed packages.

Package	Version	Latest version
cycler	0.10.0	0.10.0
future	0.17.1	0.17.1
kiwisolver	1.1.0	1.1.0
lxml	4.4.1	4.4.1
matplotlib	3.1.1	3.1.1
numpy	1.16.4	▲ 1.17.1
pip	19.0.3	▲ 19.2.3
pymavlink	2.3.8	2.3.8
pyparsing	2.4.0	▲ 2.4.2
python-dateutil	2.8.0	2.8.0
pyusb	1.0.2	1.0.2
pyzmq	18.0.2	▲ 18.1.0
setuptools	40.8.0	▲ 41.2.0
six	1.12.0	1.12.0

Figure 1: Project Interpreter

- c. If you are having problems installing packages because of pip, see Part IV: Troubleshooting, Command Station Errors and Flight Stability Issues.
3. Set up the CrazyFly Radio
 - a. Plug the transmitter into any USB port. A red light on the transmitter should flash one time. Note: the light may be inside the case.
 - b. Follow these instructions to download the driver for the transmitter: https://wiki.bitcraze.io/doc:crazyradio:install_windows_zadig (start by downloading libusb-win32 but if that doesn't work try libusbK).

Vicon System

This system will NOT cover the standard powering, masking, and calibration procedure for the Vicon. It will cover setting the origin and object creation.

1. Set up cameras
 - a. Turn on cameras and start Vicon Tracker software
 - b. Mask and calibrate the Vicon system as normal
 - c. Try to set the origin in the center of the room in the following orientation (from the top looking down):
 - d. **If you have already calibrated: When two or more cameras begin to flash red, you should always recalibrate.**

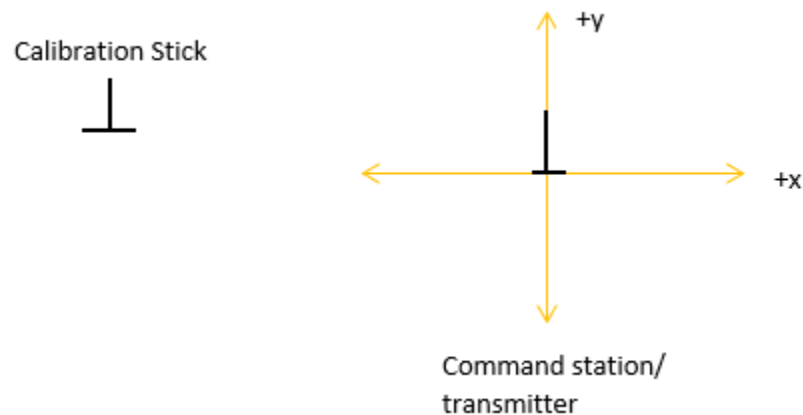


Figure 1: Vicon Calibration Orientation

2. Create CrazyFly Object
 - a. Assemble the CrazyFly in the way described in “CrazyFly” Section 1.
 - b. **If the “CF” (CrazyFly) object already exists in Vicon Tracker when you opened the software, delete it. Always delete and recreate the object every time the Vicon is recalibrated or when you switch drones.**
 - c. With the CrazyFly off, place it in the center of the room (or wherever you set the origin).
 - d. Orient the CrazyFly so that the antennal is pointed down the -y-axis with respect to the Vicon. In other words, line up the CrazyFly’s roll axis (shown in Figure 3) with the Vicon’s -y-axis.

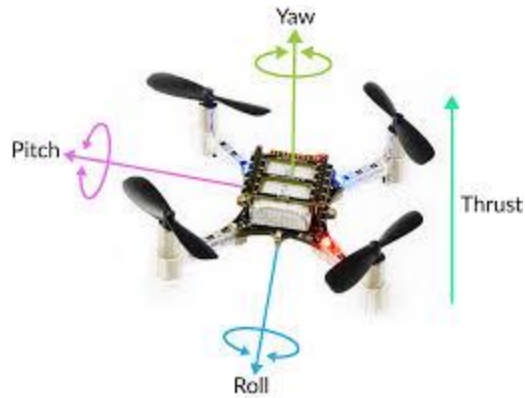


Figure 3: CrazyFly Axes

- e. Create the object in the Vicon Tracker software. **It is important that you name the object “CF” and that ALL of the reflective balls have been selected.** If the Vicon cannot see any or all of the reflective balls see “Part IV: Troubleshooting”.
- f. When the object is created, it should look like Figure 4. Note that the axes of the object and the Vicon (bottom left corner) line up.

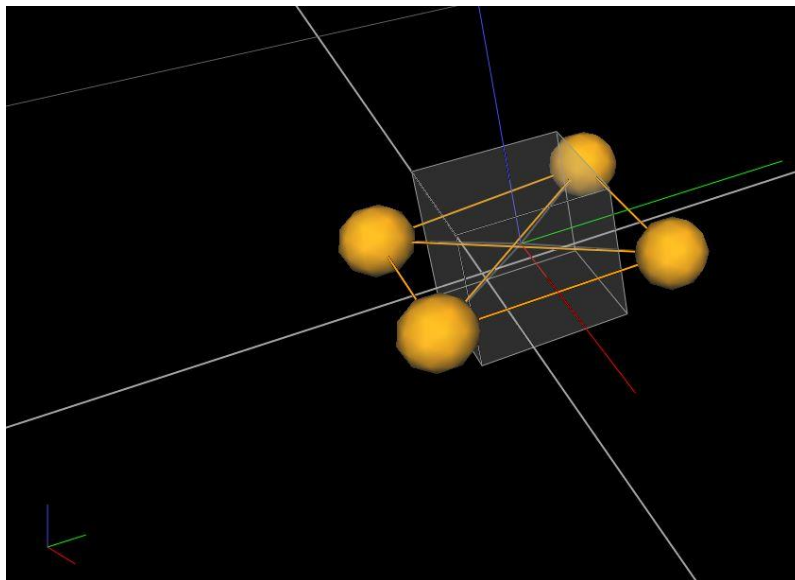


Figure 4: Vicon CF object

CrazyFly

1. Assembling:

- a. Attach plastic motor holders to all arms.
- b. Insert motor into each holder. It should take some force to pull the motor back out.
- c. Thread the wire from the motor up the arm and connect it to the port.
- d. Attach propellers so that the direction they should spin matches up with the arrow on the arm. **Half of the propellers are made to spin one way, half spin the other way, they are not all interchangeable!**
- e. Secure reflective balls in the fashion shown in Figure 5. Note that one of the balls is offset to the side. This is to prevent symmetry, which is important for the Vicon.



Figure 5: CrazyFly with Vicon tracking balls

- f. Proceed with “Vicon System” set up, if you have not already, before moving on to “Powering on”.

2. Powering On:

- a. Secure the battery in the same way as shown in Figure 3 and connect the battery. **If the battery feels “inflated” or excessively hot, pick a different battery and set the old one aside and/or mark as “bad”, and alert the lab safety supervisor as this battery may be in danger of exploding!**
- b. If necessary, press the Power button.
- c. When powered on the CrazyFly will show two solid blue lights. The other two lights will flash once from red to green.
- d. **DO NOT** let the CrazyFly sit still on a level surface for any amount of time until you are ready to calibrate.

3. Calibration

- a. With the CrazyFly on, set it in the center of the room in the same orientation (as close as you can get) to how the object was made in “Vicon System” Section 2b. You will have to be quick about this!
- b. When set down the red light will flash to indicate the battery power. If the red light is solid the battery needs to be replaced.
- c. The green/yellow connection light will flash rapidly then turn off.
- d. All four propellers should spin for a second or so. **Make sure all propellers spin freely.**

Now the setup is complete!

Part II: Flying

In this document, *PitchRolltest.py*, *demo.py*, and *CFwaypoints.py* are all referred to as “command programs” at points because they are designed to be high-level programs with user inputs. *PID_CLASS.py*, *viconStream.py* and others are not referred to as command programs here.

Preflight/ Between Flights Procedures:

1. Before you fly ensure that:
 - a. You have completed Part I with no errors.
 - b. The Vicon System is able to see the object. If more than two Vicon cameras are flashing red, re-do “Vicon System” setup.
2. If this is not your first flight of the day; between flights always turn the CrazyFly off and repeat the “Powering On” and “Calibration” steps. This will prevent two things:
 - a. A “Log entry error” when running a command program such as *demo.py* or *CFwaypoints.py*.
 - b. Sensor drift for the CrazyFly’s onboard sensors.

Pitch-Roll-Thrust Test

1. **Before flight (especially your first flight) you should test the pitch, roll, and thrust.**

To test pitch and roll you will need the CrazyFly client. If you do not have it already, download it from Bitcraze now. With the CrazyFly in your hand, follow these steps in order:

 - a. With the drone connected to the Client, *firmly* hold the drone in your hand as level as possible and *slowly* push the throttle up to 100% using the Xbox controller. If the drone feels like it is trying to pull in a certain direction, it is likely that a propeller is on backwards, or something is preventing a propeller from spinning freely. If you are sure that this is not the case, repeat the “CrazyFly” section in Part I and/or see Part IV: Troubleshooting.
 - b. With the drone connected to the client, tilt the drone to its left and then right slowly. You should see a corresponding change with the roll in the client. Tilting left (pointing M4 and M3 towards the ground) will result in “negative roll” and visa-versa for tilting right.
 - c. With the drone connected to the client, tilt it forward and then backward slowly. You should see a corresponding change with the Pitch in the client. Tilting forward (pointing M4 and M1 towards the ground) will result in “negative pitch” and visa-versa for tilting backwards.
 - d. Disconnect from the client and repeat the same experiment as in part (b) with the CrazyFly in the Vicon testing area while running *PitchRolltest.py*. *PitchRolltest.py* will give you 30seconds to roll the drone to the left and then

right, but if you need more time, change the time.sleep value. You may begin to move the drone as soon as the program prints “MOTOR_UNLOCK_SENT”.

Make sure UAVname is set to ‘CF’; it must be the same name as the Vicon object. From here, move to Part III: Data Processing and check that the “Roll” graph shows data that concurs with your results from part (b), that is left is negative and right is positive.

- e. With the drone disconnected from the client repeat the same experiment as in part (c) with the CrazyFly in the Vicon testing area while running *PitchRolltest.py*. From here, move to Part III: Data Processing. **The “Pitch” recorded by CFcontrol is actually opposite of what is recorded by the client, that is in this case, tilting forwards (M4 and M1 towards the ground) will actually be positive in the raw data, but *SaveVarCF.m* flips the sign so that the data lines up with what the client says.**

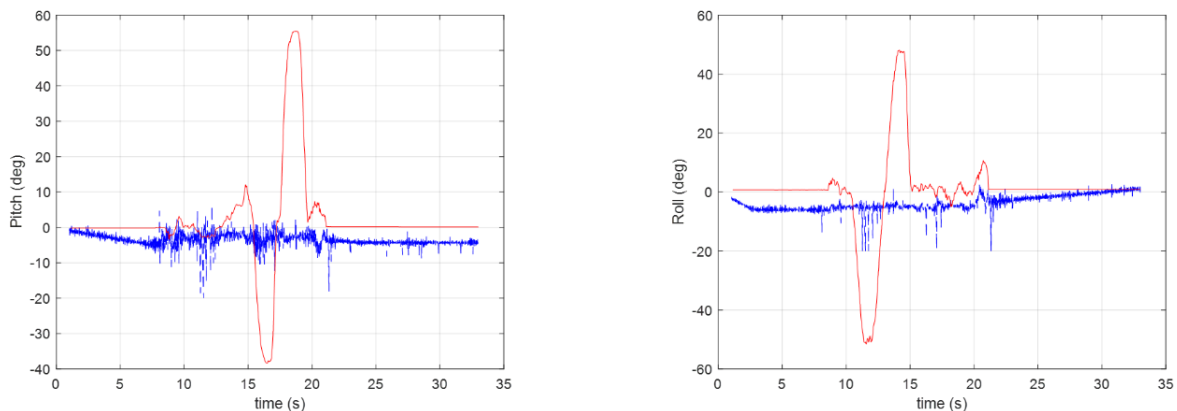


Figure 6: Left: pointing M4 and M1 towards the ground “negative pitch” then pointing M2 and M3 towards the ground “positive pitch”. Right: pointing M4 and M3 towards the ground “negative roll” then pointing M1 and M2 towards the ground “positive roll”.

2. If the Pitch and/or Roll are flipped in parts (b) or (c), that is likely a calibration or sensor issue. Attempt to recalibrate the drone or see Part IV: Troubleshooting. If the Pitch and/or Roll are flipped in parts (d) or (e), re-do the “Vicon System” section of Part 1: Setup.

Once you have completed this test you are ready for flight!!

Running demo.py

1. *demo.py* introduces you to three simple functions that most command programs use:
 - a. takeoff: sends the drone to a specific z-coordinate, x and y are constant.
 - b. goto: sends the drone to a specific x, y, z coordinate
 - a. land: brings the drone back to ground level, x and y are constant.

Note: All distances in CFcontrol are in meters and all times are in seconds.

2. It is recommended that for the first flight you edit the program to perform a simple take off and land, setting `alt = 0.75`, with a `time.sleep` of several seconds between “takeoff” and “land” as well as between “land” and “`uav.active = False`”.
 - a. If that does not work and/or the CrazyFly was very unstable, see “Part IV: Troubleshooting”.
 - b. If it does work, try using the “goto” function as described in sub-section 3 below.
3. Tell the drone to take off from the origin (which you should make as the center of the room) to an altitude of 0.75m, followed by `time.sleep(5)`. Follow this with the command `goto(0.5, 0, alt)`, followed by “land” and “`uav.active = False`”, as per usual. To analyze this test follow this procedure:
 - a. Process your data using the method described in part 3 described in Part III.
 - b. Note that if you set the crazy fly at the origin at the start of the test with the antenna facing the command station (the way that is described and recommended by Part 1), then the CrazyFly would have to move to ITS left, which is negative roll.
 - c. **If you see a significant period of positive roll in the data, then you know something is wrong. Also, if there are significant amounts of pitch or yaw in the data (more than $\sim 10^\circ$ for a significant amount of time) then something is also wrong because there should be almost no pitch or yaw in this test.**
 - d. In either case, go to Part IV: Troubleshooting now if you are experiencing these issues.
 - e. Shown below are pictures of pitch and roll data from a successful execution of this test.

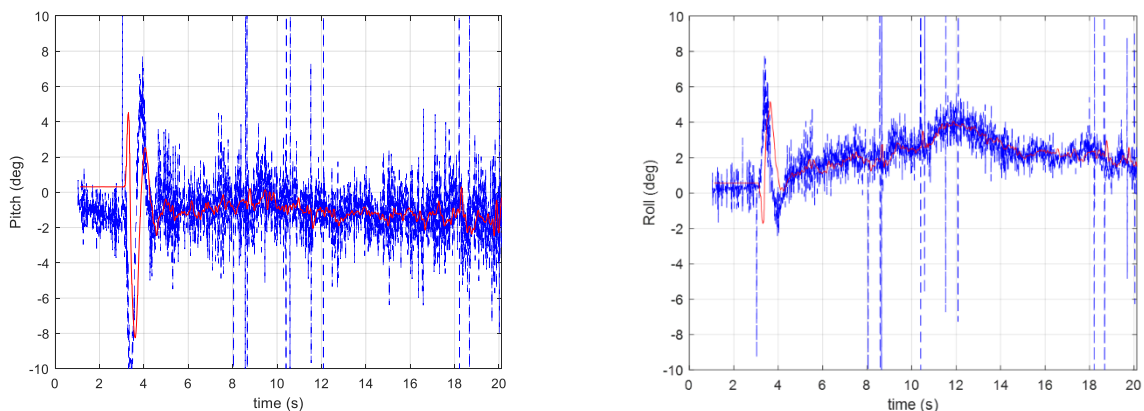


Figure 7: Roll and Pitch Data for an average takeoff and land test using *demo.py*

- f. The average of the blue-line in the pitch and roll graph should be around 0° for this test, but if it is not you may need to “trim” the drone, even if the flight appeared stable. In the case of Figure 7 above, you may want to trim the Pitch up

by about 1° and trim the Roll down by about 2° . This process is described in Part IV: Troubleshooting.

If you have completed this section and are satisfied with the CrazyFly's performance, you can move on to running *CFwaypoints.py*!

Running *CFwaypoints.py*

1. **TO PROCEED YOU MUST HAVE COMPLETED THE ENTIRE PROCEDURE UP TO HERE WITH NO UN-FIXED ERRORS**
2. To run *CFwaypoints.py* follow the instructions in the program provided via the program comments. Several things to note are:
 - a. The same general rules that apply to *demo.py* apply to *CFwaypoints.py*.
 - b. *CFwaypoints.py* will not allow you to input points that are outside of the safety zone, nor will it allow you to input incomplete (x, y, z) coordinate groups.
3. The "Subdivide" function gives you the option to have sub-waypoints. For example, if you want to move from point A to point B "Subdivide" will place sub-waypoints in-between A and B that will form a straight line and help the drone stay on-course. There are several options when creating subpoints:
 - a. You can either set a number of waypoints to divide each section into or a distance that you want between each subpoint (if the distance is not an even division, you will receive a warning that the last waypoint has been truncated off).
 - b. As a rule of thumb: set "R_tol_sub" to be no less than half of the drone's diameter, and have a distance between sub-points that is no less than R_tol_sub. For example, the CrazyFly is about 10cm wide so I set "R_tol_sub = 0.05" and I want to fly a distance of 1m so I set the number of divisions to "10" this creates a string of "spheres" (areas the drone must pass through) that touch each other tangentially, in three dimensions. Of course, you must play around with this to get the right combination of number of waypoints and waypoint size for your experiment.
 - c. **But be careful! Subdividing in increments that are too small may cause a delay in PID information being sent from the command station to the drone, which can cause a crash.**

Part III: Processing Data

This section will not tell you if your data is “good” or “bad”, it will only show you how to process the data. You must draw your own conclusions from there using scientific reasoning and the procedure outlined in the previous sections.

Running *SaveVarCF.m*

1. To process data after a flight, open the program *SaveVarCF.m* in MATLAB
2. To prepare the program to run follow these steps:
 - a. Clear the Workspace by typing “clear” into the Command Window.
 - b. Under the “Home” tab, in the “Variable” section, click “Import Data”
 - c. Find the text file with the data you want and click on it
 - d. Set “Column delimiters:” to “Comma” *and* “Space” (this may take a few seconds to load)
 - e. Set “Output Type:” to Column vectors (this may take a few seconds to load)
 - f. Click on “A” in top of the first column then press “ctrl + a” to select all the data.
 - g. Click the green check-mark; this will import the data.
3. Click the green run arrow in the “Editor” tab. This will run the program though it may take several seconds to print the graphs to your screen.
4. To re-run the program with the same data or different data, you must clear the workspace first then go through the importing process again, unless you save the workspace as a “.mat” file, but that is generally not necessary and not recommended.
5. Notes on *SaveVarCF.m*
 - a. **In the plots, blue dotted lines represent the commands being sent by the command station. Red lines indicate the x, y, and z position and yaw detected by the Vicon, and in the case of roll, pitch and thrust the red line indicates what the sensors on the drone itself were reporting.**
 - b. Plots can be added or suppressed by commenting or un-commenting their respective sections.

Identifying “bad zones” and Choosing the Right “dt” Comparator

1. The “for” loop at the end of the program is recording “bad points” which are points where the refresh rate (the time difference between one data point and the next) was significantly lower than average.
2. You can change what constitutes a “bad point” by changing the comparator for “dt” in line 73. You should not leave this as the default and the “dt” comparator will need to be

manually changed for each data set (each test). Sub-sections 3 and 4 below describes how to do this and why you must do it.

3. **For the CrazyFly to operate normally it needs to be receiving commands at a frequency of about 100Hz on average.** To find the frequency commands are being sent at, you have two options:
 - a. Change the value “dispUpdateRate” to “True” in the command program you are working in (*demo.py*, *CFwaypoints.py*, etc.). this will show the update rate during the test in real time while the drone is running, but it will not display the average, so it is of little use to us for now.
 - b. Recommended: after running *SaveVarCF.m* enter “1/mean(diff(time))” into the command window. This will produce the frequency.
 - c. **While ~100Hz is expected, 90Hz is still usually acceptable, but around 80Hz or less will cause significant issues.** See Part IV: Troubleshooting for information on improving signal frequency.
4. To set the right “dt” comparator in line 73, follow these steps:
 - a. Open up the “time” variable from the “Workspace” by double clicking on it and copy the entire table, using ctrl+a then ctrl+c, into an excel sheet.
 - b. In Excel, find the standard deviation of the data (σ). Use the equation $dt' = avg(dt) + 3\sigma$ where dt' will be your new comparator that is three standard deviations away from the mean ($avg(dt)$). Basically, all time steps that are more than 3 standard deviations away from the average will be counted as a “bad point”. You don’t necessarily have to do it this way, but statistical convention dictates that three standard deviations away from the average constitutes an outlier (in normal distribution, which this data approximately is)
 - c. Now change the comparator to the value you got for dt' . Clear and reimport the data as you would normally, then run the program again.
5. If your bad points plot shows no points at all or too many points, try playing around with the dt comparator.

Part IV: Troubleshooting

This section will only cover “out-of-the-box” errors for CFcontrol_WayPoints. It will not cover errors created by the user if they choose to make modifications of any magnitude to any of the included programs.

Connection Errors:

1. Vicon to Command Station connection issues

- a. Problem: receiving error: “TypeError: 'NoneType' object is not subscriptable”

Solution:

- i. Most likely, the object name in Vicon Tracker does not agree with the object name at the command station. One of these must be changed to agree with the other.
- ii. Note that the object name at the command station can be changed in several different programs, including high level command programs *CFwaypoints.py* and *demo.py*, and they must agree with Vicon Tracker. However, in *cfControlClass.py* the object name is not necessarily the same, which is confusing, but this is how it works.
- iii. If the object names do agree, but you still have the error, simply try running the program again a few more times, or move to the solution below.

- b. Problem: You are not receiving any data from the Vicon at the command station (the program gets stuck somewhere in the “Connecting to Vicon...” phase), but the object *does* appear on Vicon Tracker

Solution:

- i. Open the program *Find_position.py*. This is a simple program that can run without CFcontrol. It will find the x, y, z coordinates and heading of any object, provided that you input the correct name to “if s ==” on line 43. The program can also record these points to a text file by uncommenting the “filehandle” section in the main loop.
- ii. Run the program.
- iii. If the program runs into an error, check that the object name is correct on Vicon Tracker and check that the object is visible on Vicon Tracker. If this is correct, check that the IP address and port number in the function “vicon_connect” are correct for the Vicon system. This is NOT the UDP IP and PORT numbers.
- iv. If everything checks out here and you are still having Issues, it is likely a problem with *python_vicon*, which is discussed later in the section ‘Command Station Errors’.

2. Command Station to CrazyFly connection issues

- a. Problem: receiving error similar to “no module named usb” when running a command program.

Solution:

- i. Remove and reinsert the USB radio transmitter. Make sure the red light flashes when inserting. Then run the command program again.
- ii. If problem persists, check that your interpreter has all the same packages as in Figure 1 at the beginning of this document, especially pyusb.
- iii. Follow this tutorial:
https://wiki.bitcraze.io/doc:crazyradio:install_windows_zadig and select “libusbK” as the driver.

- b. Problem: receiving error “No CrazyFlies found, cannot run” or program runs all the way through with no errors but the drone never turns on when running a command program.

Solution:

- i. Remove and reinsert the USB radio transmitter. Make sure the red light flashes when inserting.
- ii. Power cycle the CrazyFly (including normal recalibration steps)
- iii. Try running the program again. You may need to repeat this several times or possibly try different CrazyFlies, transmitters, or computers.

- c. Problem: When running a command program, the drone receives the first (and possible second) waypoint(s) but does not take off. Propellers may move slightly.

Solution:

- i. Remove and reinsert the USB radio transmitter. Make sure the red light flashes when inserting.
- ii. Check that all the connections are good on the CrazyFly.
- iii. Change the battery of the CrazyFly.
- iv. Power cycle the CrazyFly (including normal recalibration steps)
- v. Try running the program again. You may need to repeat this several times or possibly try different CrazyFlies, batteries, transmitters, or computers.

Vicon Errors:

1. Drone visibility issues

- a. Problem: one or more balls are not being detected by Vicon Tracker when trying to create the object.

Solution:

- i. Try placing the drone a ~6in. away from the origin, **but keep it in the orientation described in Part I: Setup.**
- ii. Try elevating the drone off the ground by placing it on a stool or box.
- iii. If the system has not been recalibrated in some time, and/or two or more cameras are flashing red, then re-mask and recalibrate the system.

- iv. If the problem persists after this, try changing the “Gain” setting for one, multiple, or all of the cameras. In Vicon Tracker go to the System tab, select one or more cameras, click “Show Advanced” then increase the “Gain” multiplier under settings, but this may backfire by detecting things that aren’t there.

2. Camera Issues:

- a. Problem: the same few cameras always begin to flash a red LED light shortly after calibration.

Solution:

- i. Some parts of the room are more prone to vibrations that can disturb the cameras sensors.
- ii. Unfortunately, there is not much we can do about this other than recalibrating the system.
- iii. If this problem becomes too great, you can change the “Bump sensitivity” of the cameras in the same menu as changing the Gain multiplier, but this may produce negative effects for accuracy.

- b. Problem: one or more of the Cameras is flashing red on Vicon Tracker.

Solution:

- i. It is possible that the camera has overheated and must be removed to be allowed to cool down.
- ii. In some cases, one bad camera can cause the entire system to turn red. This may be due to overheating or some other malfunction. It will be up to you to determine which camera(s) is/are causing the problem through trial and error.
- iii. If the issue is severe, see the Vicon manual or contact Vicon Industries Inc.

Command Station Errors and Flight Stability Issues:

1. Drone stability issues due to command station

- a. **Problem: Drone is extremely unstable in flight.**

Solution:

- i. **First, check Problem b. of “CrazyFly Issues”, which is the section after this one.**
- ii. **Go back to Part 1: Setup, Vicon System, Create CrazyFly Object and ensure that the axes of the object match with the axes of the Vicon system.**
- iii. **Look at the data of the most recent test and find the frequency as described in Part III. Again, 100Hz is normal, 90Hz is still usually acceptable, and 80Hz or less will cause significant issues.**
- iv. **Your solution to increasing this frequency, if it is the problem, will depend on the exact program you are running. Often a decrease in**

frequency is due to sending too many setpoints at once, which can overwhelm the CrazyFly, so you may want to decrease the number of setpoints or subdivisions between setpoints you are using. It may also help to be using a computer with a cable connection to the Vicon, instead of wireless.

- v. **Try playing around with the value for “pitch_roll_cap” in *PID_CLASS.py*. the default value is 10° but you may find better performance if you turn this up to 20°. Do not increase past 20!**
- b. Problem: Drone flies straight up into ceiling immediately after takeoff
Solution:
 - i. Most likely you have enabled “fakeVicon”, possibly by accident. “fakeVicon” is an optional debugging function that is not used in this guide’s procedure.
 - ii. To disable “fakeVicon”, go to *cfControlClass.py* and change the value for “fakeVicon” in line 14 to “False”

2. Errors when running command programs

- a. Problem: receiving an error similar to “No module named python_vicon”
Solution:
 - i. Go back to Part 1: Setup and make sure you are running a 32bit version of python.
 - ii. Note: not all releases of Python 3 have a 32bit version.
- b. Problem: receiving an error similar to “log entry error” while running a command program.
Solution:
 - i. The cause of this error comes from not turning off the CrazyFly and recalibrating after the previous flight.
 - ii. This error seems to have no negative effects.
 - iii. It can be corrected by recalibrating the drone before every flight.
- c. Problem: receiving the error “float division by 0” while running a command program.
Solution:
 - i. There is no known solution at this time, however, this error seems to have no negative effects.
- d. Problem: receiving any error of the form: “No module named ___” (except for “python_vicon” or “usb”)
Solution:
 - i. Go back to Part 1: Setup and make sure you have all the same packages shown in figure one.
 - ii. If you cannot install packages because of errors with pip, your version of pip may be newer than the one shown in in Figure 1. Try reverting pip to an older version, preferably the version in Figure 1.

3. Errors running SaveVarCF.m

- a. Problem: receiving the error of the form “Undefined function or variable '_____'.”

Solution (assuming the error occurs on or above Line 21):

- i. Take note of the name of the variable in the error message, then type “clear” into the command window.
- ii. Go to the “Import” window and look for the variable in the error message. If, for example, the error had to do with “VarName17” you would look across the top row and see that the name for column Q is not “VarName17” as it should be, but instead it is something random.
- iii. Correct the misnamed variable and reimport the data then rerun the program.
- iv. This error occurs seemingly at random and the reasoning behind its occurrence is unknown at this time.

- b. Problem: Figure 8 appears as blank or only dotted blue lines are shown.

Solution:

- i. Try expanding the axis of the grid in line 87. Reimport the data and rerun the program.
- ii. Try changing the comparator for dt in line 73. Reimport the data and rerun the program.

CrazyFly Issues:

1. Problems with the drone power:

- a. Problem: no lights turn on when battery is plugged in:

Solution:

- i. The battery is likely dead and will need to be replaced and recharged.
- ii. If you try multiple batteries with no luck check that the leads for the power connection are properly soldered in place.
- iii. Try pressing the Power button
- iv. Try connecting the CrazyFly to any computer or power source via its micro-USB port.

2. Problems with drone stability:

- a. Problem: Drone drifts in one direction and/or the blue line in the pitch and/or roll graphs does not average around 0° when performing a simple takeoff and land test.

Solution:

- i. Perform a simple takeoff and land program that does not attempt to move the drone in any direction other than straight up and down.
- ii. When the flight is complete, analyze the data and see if there is any trend in the roll and/or pitch graphs
- iii. The roll and pitch should both be near zero, but if they are not, take note of what the what the average of the blue line appears to be. It is usually somewhere between -5° and 5°.

- iv. Connect the drone to the client and change the “Roll Trim” and/or “Pitch Trim” values to compensate for the drift. **Be careful of signs here!**
 - v. Note that this will not fix sensor drift, i.e. if the drone is left on a flat surface while connected to the client and the client claims that the pitch and/or roll are more than $\pm 3^\circ$ and constantly increasing in magnitude, then you have a bigger problem. The only known partial-solution to this issue at this time is recalibrating the drone or reflashing the firmware, but this drone may be unrepairable.
- b. Problem: Drone is unstable in flight, but there are no issues with the Vicon System or Command Station signal frequency.
- Solution:
- i. Check to make sure propellers are placed in correct orientation, and that they spin freely.
 - ii. Check that all motors are secure in their housings (the plastic limbs that hold the motor onto the arm) meaning that they cannot easily be pulled out. Also check that every motor’s electrical connections are attached.
 - iii. Re-do the Roll-Pitch-Thrust test described in Part II using the client and an Xbox controller.
- c. Problem: drone sensors are reporting false data:
- Solution:
- i. This is referred to as “sensor drift”. An example being: if the drone is left on a flat surface while connected to the client and the client claims that the pitch and/or roll are more than $\pm 3^\circ$ and constantly increasing in magnitude.
 - ii. The only known partial-solution to this issue at this time is recalibrating the drone or reflashing the firmware, but this drone may be unrepairable.

If there is some problem you have with any part of the system that has no solution here or the solutions provided do not work for you, you always have these options which will solve 99% of your problems:

1. Powercycle the CrazyFly and remove and reinsert the radio transmitter.
2. Try to reflash the CrazyFly firmware. Connect the CrazyFly to the computer or connect to the client via radio and use this repo from GitHub: <https://github.com/bitcraze/crazyflie-firmware>
3. As a last resort: redownload the entire repo “CFcontrol_WayPoints” from GitHub and start over on what you were working on. I know this can be hard, but sometimes you must.