Quad Pixel Width and Speed Estimation

Abstract

The goal of this paper is to provide a novel system of estimating width and velocity of an object passing perpendicularly through a sensor field. By fixing four sensors on an axis at known angles the distance and velocity of an object can be estimated from the times of detection of each sensor using a system of equations. To develop a prototype of this system the scenario was modeled and simulated, a processor and sensors were selected, and algorithms for detection and estimation were developed. The goal of the project was to develop a prototype that would correctly estimate the speed of the object in miles per hour within 15% accuracy, and to estimate the width of the object within 20% accuracy.

**Relevant Equations**

All are known, and the average value of the times detected at one sensor will be counted as the sensor time (objects will trigger multiple detections times, the average of these is the sensors ‘detection time’ used in the subtraction.

With these equations, knowing the dimensions and angles of the sensor array allows estimation of distance and velocity from the stream of detection times.

This equation can be used to estimate the width of the object based on the duration of time individual sensors detected its presence and the estimated velocity.

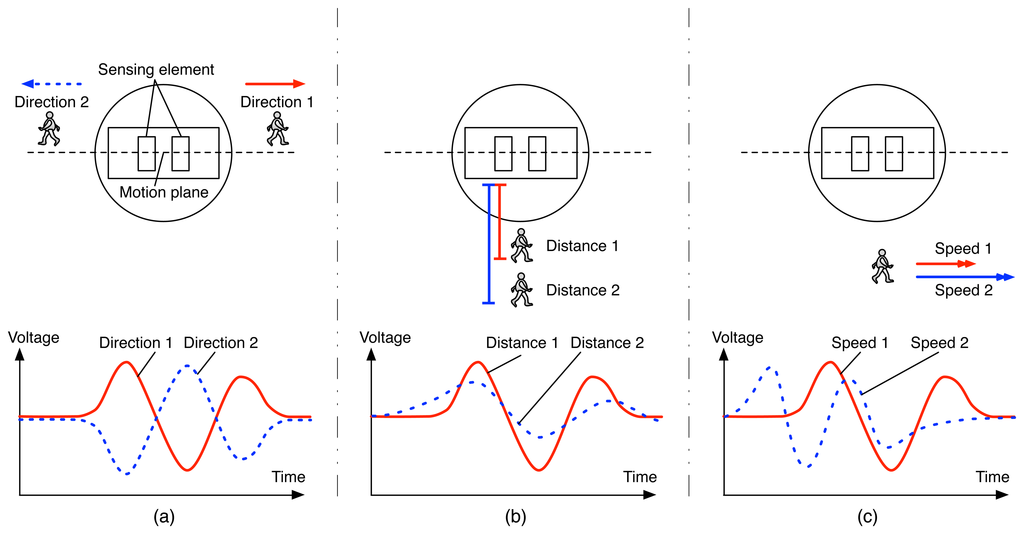
**Tradeoff Matrix Write Up**

For the prototype of the detection system the sensors should have a range of at least 75 feet, a small field of view, and a reasonable sample rate. The small field of view allows each sensor to be treated close to a ‘pixel’ of detection, the smaller the field of view, the more accurate the estimation can be. The decent range of 75 feet allows the geometry of angles of the sensors to spread out enough for full scale (i.e. human or vehicle) tests. The sample rate of the device determines how fast the test objects can move, a full scale vehicle test would require a high refresh rate for a camera, but a low sampling rate for an IR sensor (30 – 60 fps performs well on most vehicle speeds according to simulation).

Four sensors were considered for this project. The first sensor considered was a simple passive infrared sensor, which would be connected through an ADC or GPIO to the Raspberry Pi. The second was an off the shelf IR security sensor which would interface similarly to the Pi. The third option is a standard web camera with a USB connection to the processor. The final option was the FLIR Lepton infrared camera. This option interfaces to the Pi in one of two ways: SPI data and I2C command line or over USB like a webcam with a breakout board.

A standard PIR sensor has a range of up to 20 feet and a wide field of view. This can be narrowed by using lenses, the smallest of which produced a 10 degree FOV. The signal out of a PIR sensor would need to be sampled by an ADC to analyze the waveform of the sensor. Circuits also exist to simply raise a GPIO pin if an intrusion is detected, but these do not provide enough data for this project as they only provide one bit of resolution. The power consumption of these sensors can be relatively low, and they are easily extended to an array of four with a multi-channel ADC or multiple GPIOs. Price is another strong point for this type of sensor as they are very cheap. Unfortunately for this prototype the PIR sensor’s range and wide FOV make them nearly unusable. Ultimately in order to create a low power solution Sandia will look into creating a custom lens system to implement this project, however the lens design is beyond the scope of this project and paying for such a device is beyond the budget.

PIR Sensor wave forms



<http://www.mdpi.com/sensors/sensors-14-08057/article_deploy/html/images/sensors-14-08057f2-1024.png>

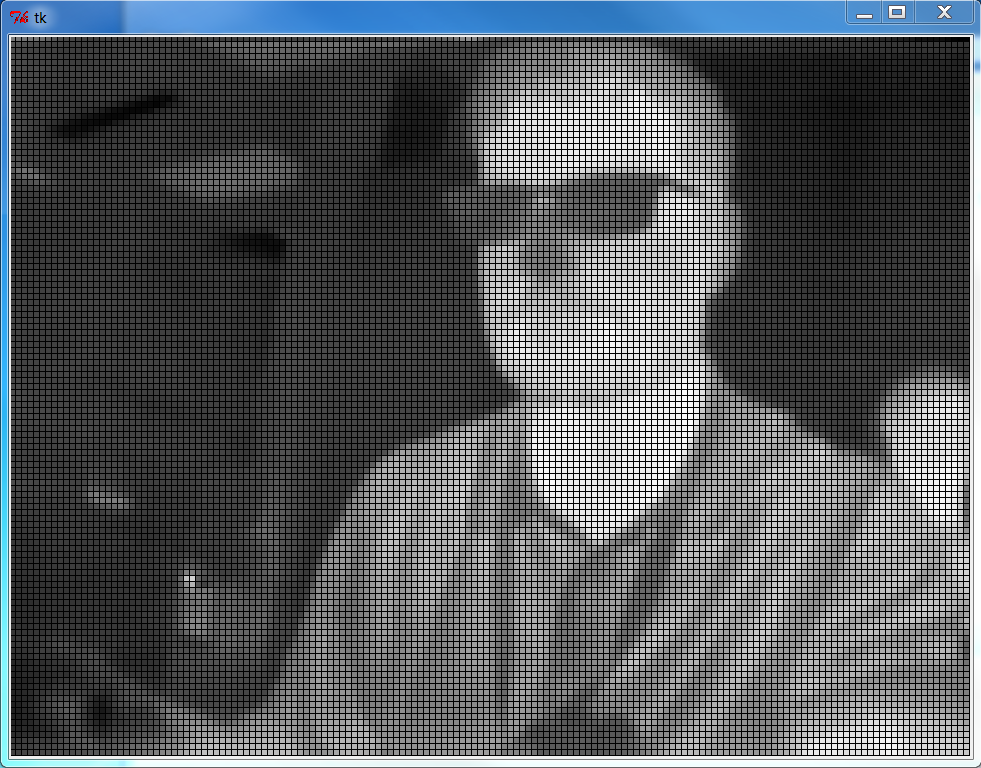
The Takex PIR-50NE is an off the shelf security solution with a PIR sensor at its core that solves the FOV and range problems of the standard PIR sensor. With a 165 foot range and 3.5 degree FOV this sensor has great range and narrow spread. The weaknesses of this sensor come from implementation complexity, power consumption, and cost. All TAKEX sensors are produced ready to be integrated into existing security systems meaning the device would need to be opened and modified to give the Raspberry Pi direct access to the waveform from the PIR sensor at its core. In the out of the box state, the sensor sends a 20V 100 mA pulse for 2 seconds upon an intrusion detection, a sampling rate that is much too low and a power consumption that is much too high. With an individual cost of $520 this sensor is also the most expensive of the four. The TAKEX PIR-50NE contains a sensor and lens system that is nearly ideal for this project, however the price and unknown complexity of modifying the sensor hurt its viability.

The third option was using a basic web camera as the sensor. This option provides flexibility in FOV, great range, low cost, and is easy to implement. However, these cameras simply don’t monitor the radiation this project is interested in. A different image processing project could be completed using web cameras, but for this project an IR sensor is necessary.

The final and chosen option is the FLIR Lepton v3 IR Cameras. This camera has a 160x120 resolution which allows for flexible FOV settings, a good range of 100 feet, medium cost, and low power consumption. The PureThermal 1 breakout board changes the interface from SPI video and I2C command to UVC USB video format. The weakness of this option is its 9 FPS. This sample rate is very low; however a solution is slowing the test objects. The Raspberry Pi has 4 USB ports, so four cameras can be attached to four independent ports (and a mouse and keyboard with a powered hub). Thus through the Linux interface v4l2 (video for Linux 2) frames of data can be captured and processed in many languages (C and Python notably). Due to cost, range, and flexibility the FLIR Lepton v3 Cameras with PureThermal 1 breakout board were chosen as the sensor for this system’s prototype.

**System Prototype**

Given the sensor selection of the FLIR Lepton v3 with PureThermal 1 board the prototype of the system consists of four of these sensors, a Raspberry Pi 3, and a powered USB hub. Firmware for the version 3 was found on GitHub, as the previous versions had half the resolution, and therefore the older firmware is not compatible with the new cameras. After working with v4l2 libraries (in python and C), data from the cameras was successfully captured. A simple Python script (which was later converted to C to similarly validate data was good) outputted a sketch of the thermal measurements by creating Tkinter an array of Tkinter rectangles on a canvas with each rectangle’s color scaled between black (cold) and white (hot) based on that pixels value compared to the minimum and maximum values of the captured data. The resulting image is consistent with the thermal profile of the surroundings.



Given successful data acquisition, the next step was to write algorithms for detection and estimation and run them on the prototype.

**Simulation**

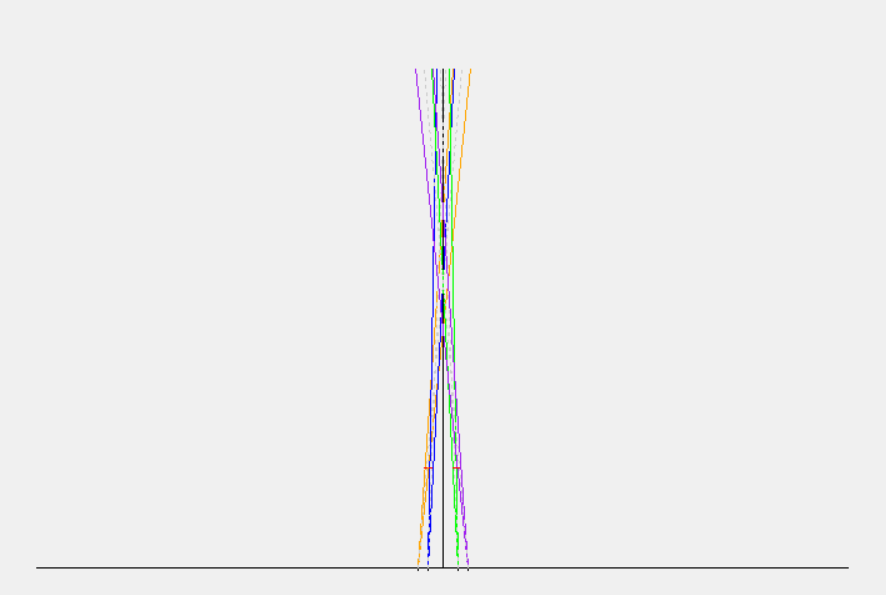
While the sensor-processor interface was being defined a simulation of the scenario was created to better understand the system as well as find optimal geometric parameters for the sensors and optimal testing conditions (width, speed, and distance of the test object. This simulation was written in Python 2.7. It functions by being given three sets of data: test object parameters, geometric parameters, and sensor parameters. Test object parameters are the distance, speed and width of the object passing through the system, the known truth that is being estimated. The geometry parameters define the distances apart and angles from the center the sensors are placed at, data that contributes to the estimation algorithm. Sensor parameters characterize the field of view of the sensor as well as the sampling rate. The test object data set is specified as a range of distances, speeds, and widths with a step size to iterate over. With the system configuration and test objects set, the program begins simulation of the scenarios specified by defining the sensor’s beam width and x-values at the specified distance. Next the test object is placed further negative then the sensor positioned graphically left-most. A loop then steps this object to the “right” by incrementing the front and back coordinates of the test object by the sampling rate multiplied by the speed of the object. This suggests the simulated system has a perfect sampling rate as well as uniform sampling times. As the test object moves across the sensor field, the range of x-values it occupies is tested against the range of x-values that each sensor monitors, if these two ranges coincide, the current time of the system is added to the a list of times for that respective sensor. After the test object completes its run through the sensor array, the average of each list of detection times are found and the simulation uses these times to solve the relevant system of equations to return the estimated distance, velocity, and width estimates. For each of the test object parameters (a specific width, distance from the sensors, and speed) the simulation runs 10 tests at varying starting points within the distance the object can move within one frame (sample rate \* speed) to determine a best, worst, and average estimate for this specific scenario. The simulation will output the percentage of the test set that are above 10% error when estimating velocity and the percentage of these tests above 20% error when estimating width.

A graphical user interface was developed to better show the test scenario specified in the simulation, as well as show the different snapshots the system would get of the object as it passed through the sensor field. When the simulation begins, the geometry of the sensors are drawn on an x-axis, centered on a y-axis. This includes dotted lines to indicate the ‘one pixel wide’ perfect line of sight of the camera, as well as lines on either side of this to indicate the actual field of view of the camera. As the simulation continues, an option can be checked to show the object’s location at each subsequent sampling time. This illustrates how the test object steps through each sensor beam and allows better understanding of sensor misses and other anomalous behavior.

**The Geometry of the System**

The simulation was used to find optimal windows of detection to base tests around. Many configurations of the sensors were considered. To determine which configuration was preferred simulation results were used and considered alongside other qualities of the configuration such as which had the smallest sensor base spread. To determine accuracy a wide range of distances, velocities, and widths were used over a few cases to determine an average performance.

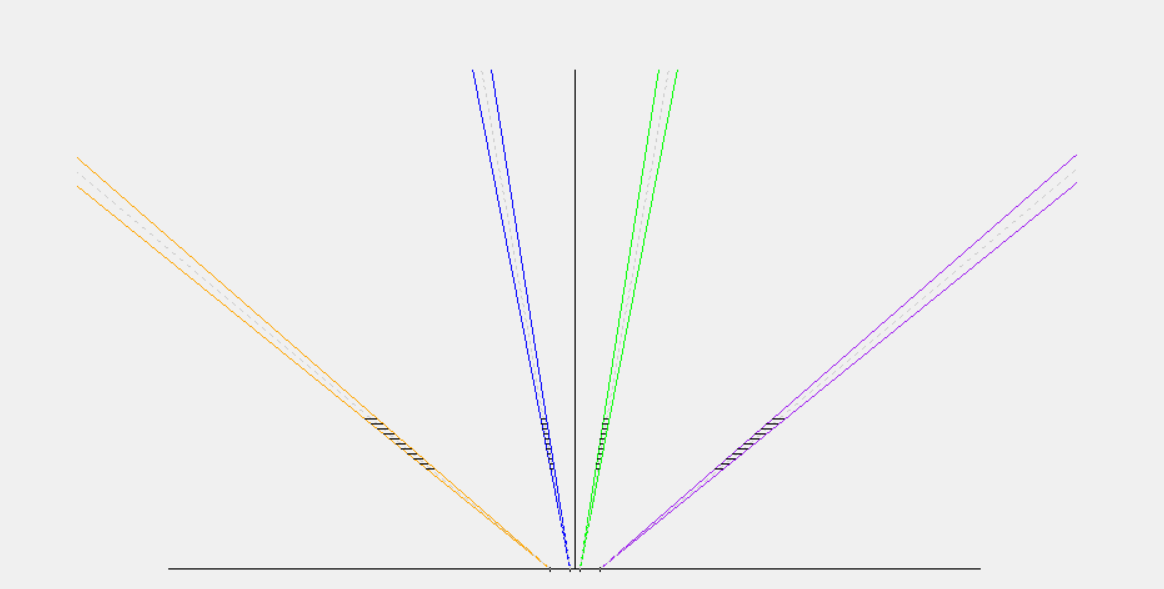
Configuration 1

The first configuration considered had all the sensors at acute angles with respect to the x axis and the y axis. This proved neither accurate nor practical as the distances of the sensor from the base of the axis would need to be large to allow a decent range and spacing between sensors. Recording the same or very similar times for multiple sensors leads to inaccurate estimations.

The project proposal showed the sensors in this array, so for a considerable amount of time this was the assumed configuration. This helped lead to several decisions that would later be to the detriment of the project, for example the selection of the low frame per second sensors. If the configuration required acute angles toward the center, PIR sensors with basic lenses on them become unworkable. This is due to the fact that with off the shelf and cheap lenses, the field of view can be reduced to 10° and the range increased to approximately 20 feet, which is much shorter than the desired range. The spread of the sensors at the edge of the region of detection is 3.5 feet, suggesting for a good amount of the 20 foot range the signals would be colliding and aliasing together as the same detection time. Having multiple sensors with the same average detection time results in division by zero in the system of equations to produce an estimate, and therefore should be avoided. Another issue with this configuration is the width of the base. Since overlapping sensors produce similar average times and decrease the ability of the system to estimate accurately or at all, this configuration favors sensors that are a spread out along the x-axis. This leads to several issues including difficulty in creating a reliable setup procedure or rig for the sensors given such a wide base, ensuring connections for the prototype like USB cables are long enough, and the fact that a compact implementation is preferred.

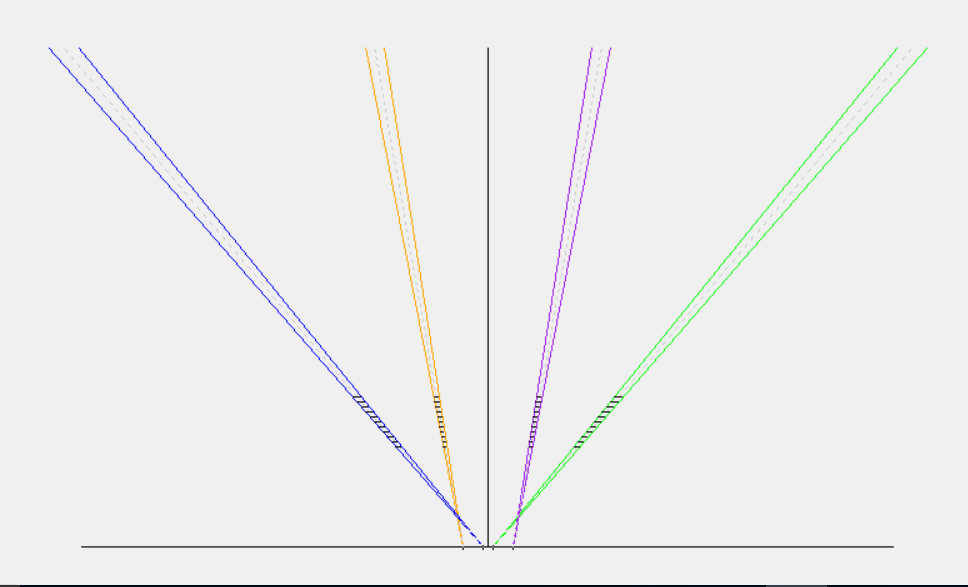
When defining the system’s prototype and selecting sensors, the design challenges of this configuration led to design choices that would greatly impact the success of the prototype, whereas later configuration options were not hampered by the cramped nature of this setup and would have resulted in a different prototype design.

Configuration 2



The second configuration considered works considerable better for both accuracy and the horizontal size of the system, however, the two sensors in the middle are still closer than necessary, which can cause the sensor times to be grouped together, still hurting the accuracy of the estimation.

Configuration 3

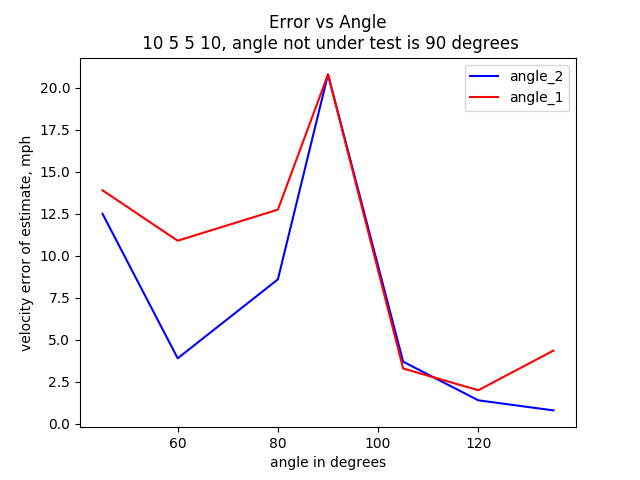


The most accurate and space conscience configuration points the closest grouped cameras at the harshest angles, while allowing the further spaced cameras softer angles. This allows the system to be tightly spaced horizontally while still being accurate. It should be noted that with this configuration the object being observed and estimated should be beyond the points of convergence of the sensors, or aliasing can occur.

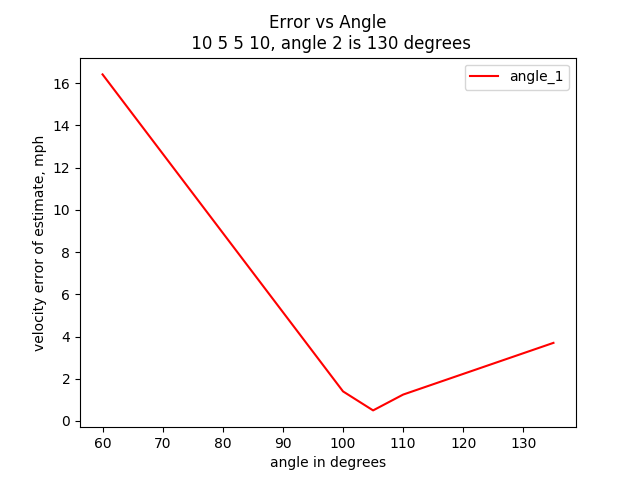
Determining Good Test Objects through Simulation

In order to define both the geometry of the system and the set of test objects, many scenarios were run through the simulation. First relationships between the geometry of the system and the error were developed.

To determine the best angle for the two sets of sensors to use, the angles not under test was held at 90°, the base was set up for the outside sensors to be ten feet from center, the inside sensors to be five feet from center, and the field of view and frames per second were set to 2° and 9 respectively based on the FLIR Lepton camera setup. The distance was set from 30 to 40 feet, the speed set from 5 to 15 mph, and the width was set to 5. All of these parameters were meant to showcase a reasonably favorable test setup, with the base being allowed to be wider than desired in order to fairly consider acute angles.

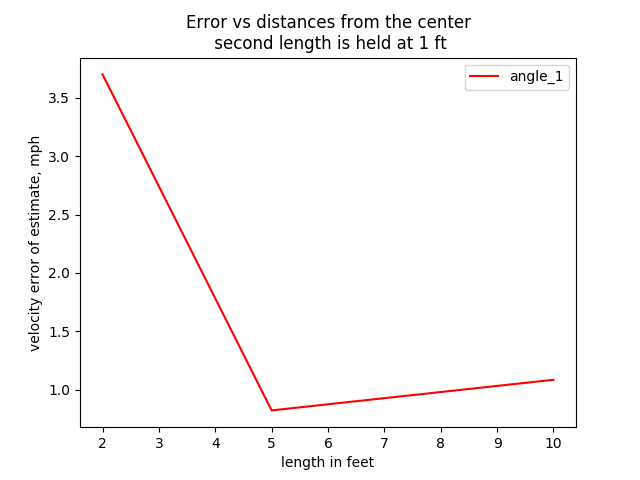


The results suggest the wider the angle, the more accurate the estimation, which agrees with previous reasoning about conflicting regions of detection and decreased sensor accuracy. When sensors are pointed further away from each other, accuracy increases. It is interesting that a local maximum occurs when the angles are equal, suggesting that the estimation equation does not work well when the two geometries used are similar (in this case two rectangles). Given a minimum approximately one mph estimation error occurred when angle 2 and angle 3 were set to 130°, the test to determine the optimal angle 1 and 4 was repeated holding angles 2 and 3 at 130°.

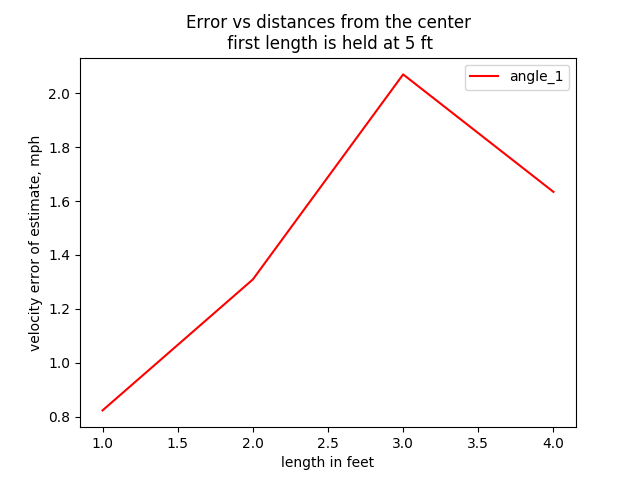


Here the velocity error was minimized beneath 1 mph per estimation when angles 1 and 4 were at 105°. Moving forward the optimal angles for a test configuration were considered to be 105° for angles 1 and 4, and 130° for angles 2 and 3.

The next step in optimizing the geometry of the system was determining the best distances from the center of the system for the sensors to be placed at. Again, sensors 1 and 4 were considered mirrors of each other, as were sensors 2 and 3.



The first test set the angles to their “optimal values” while holding distance 2 at 1 foot in order to consider the smallest configuration possible (2, 1, 1, 2) while biasing the system towards a smaller setup. A minimum of viable sensor distances was found when the outside sensors were placed five feet from the center. Given this optimal location, the set of options for distance of the middle sensors from the center were now {1, 2, 3, 4}.



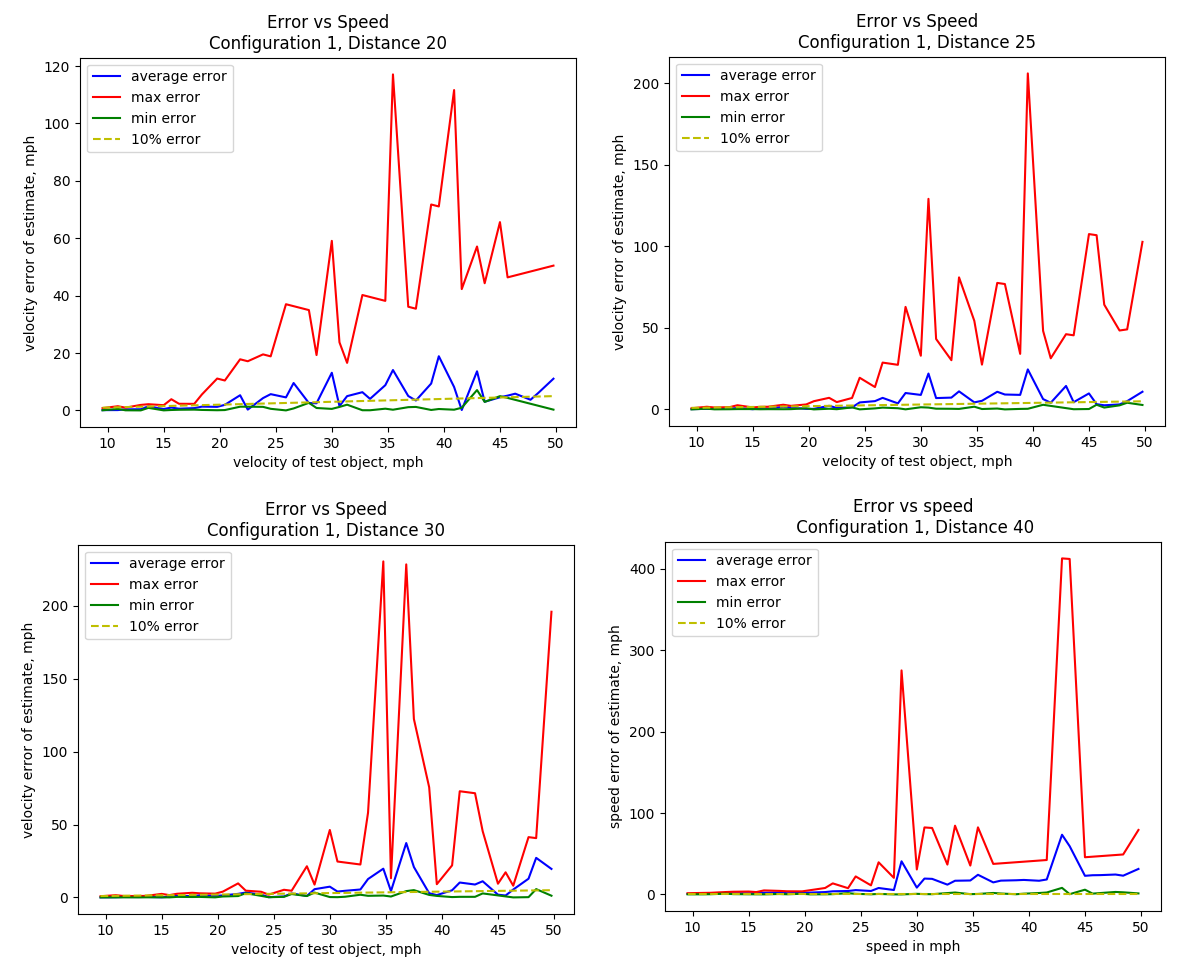
When testing this set, the minimum value was found to be where the inside sensors were furthest from the outside sensors. Given the inside sensors are at a hard angle of 130°, the range in results are much more tightly grouped than previous tests (all are within about 1.2 mph of error).

The optimal geometry of the system was determined to be outside lengths of 5 feet, inside lengths of 1 foot, outside angles of 105° from center, and inside angles of 130° from center.

Determining Test Objects

The next step was to determine optimal test objects (characterized by a distance from the sensors, speed, and width) to test the system with. The first test was to determine a testable range of speeds that would result in a reasonably accurate output. For this test the distance and width (10 feet) were held constant (multiple distances were tested) in order to isolate the effect speed has on the accuracy of the system.

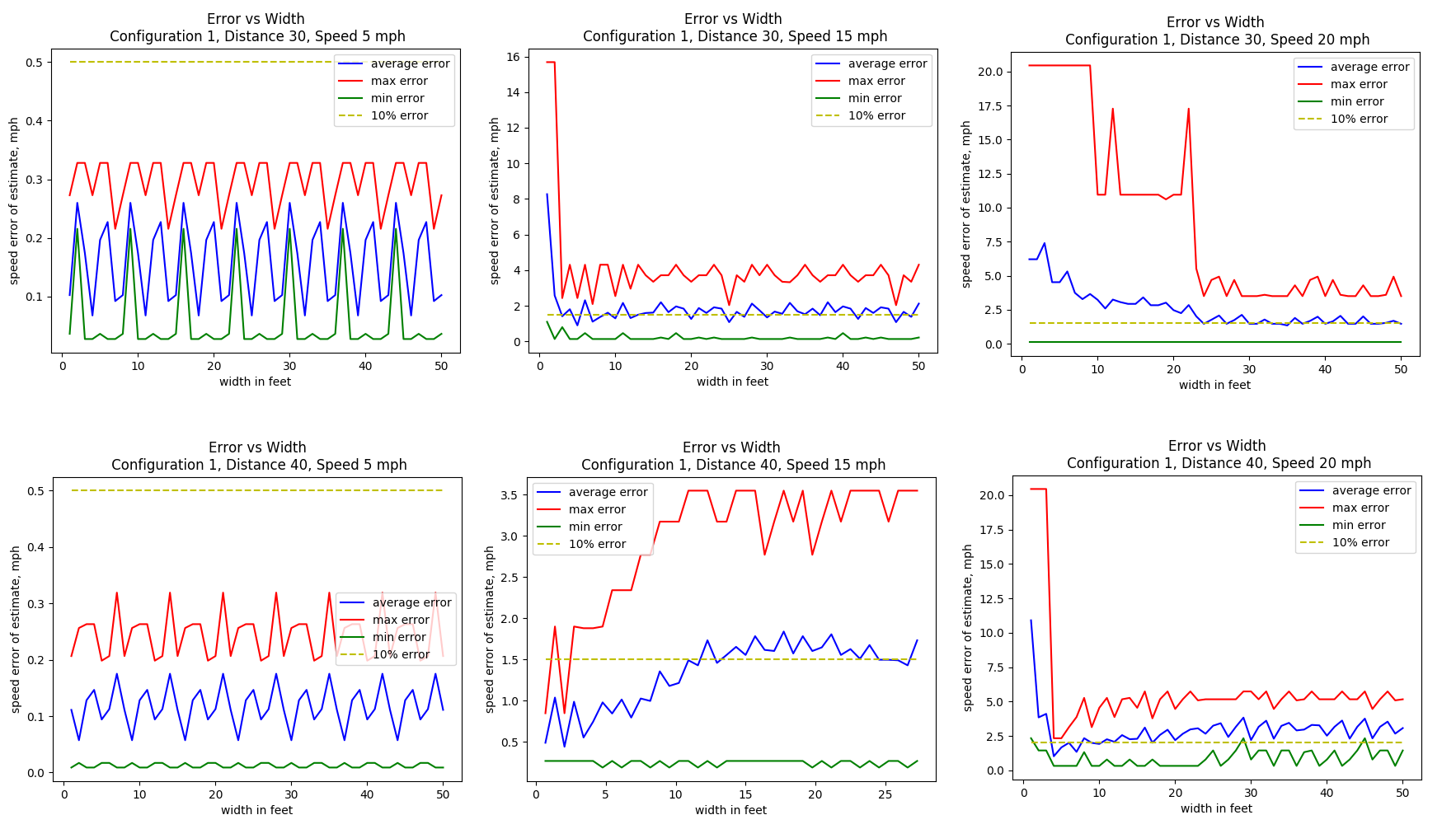
For this part of the analysis, Configuration 1 is considered the previously found optimal geometry of the system.



From a distance of approximately 25 feet to 40 feet and a range of speeds up to nearly 20 mph the minimum and maximum error graphs bound the error by at close to 10%. Thus test objects with a distance range of [25, 40] feet and a speed range of [0, 20] mph or some subset of these ranges would lead to an accurate test.

It should be noted that the effects of the low samplese per second of the system are clearly shown in this test. The error bound quickly grows to ridiculous levels when the speeds of the test objects rise above 20 miles per hour.

The next test was to determine given the previously determined ranges for distance and speed, an optimal width. The same geometric configuration was again used, and constant distances and speeds were picked from within this subset to test a range of widths on. The range was selected to be from one foot to fifty feet in order to include possible test subjects ranging from a person to a semi.



It should be noted that an error of 16 for a 15 mph test indicates a sensor miss caused the estimation algorithm to fail (and return -1), or in general an error of the speed plus one indicates sensor misses led to failure. Thus when a high speed and a small width combine, the maximum error is this number, as seen by the smaller widths on the 15 mph and 20 mph graphs. From these graphs and analysis it can be seen that distances from 30 to 40 feet perform well with a speed range of 5 to 15 mph and width of around 5 to 10 feet.

Thus test objects for this test can optimally be defined as objects with D = [25, 40], S = [5, 15], and W = [5, 10]. Running the simulation over these values with the optimal geometric configuration yields only 0.38% of tests above speed estimation error. Further, to characterize a test object as a golf cart, with a standard width of 8 feet, D = [25, 40], and S = [5, 15] the simulation returns 0% of tests have a speed estimation error of above 15%.

The optimal test conditions:

L1, L4 = 5; L2, L3 = 1; A1, A4 = 105; A2, A3 = 130

D = [25, 40]

V = [5, 15]

W = 8

Expected accuracy: all estimations within 15% of truth.

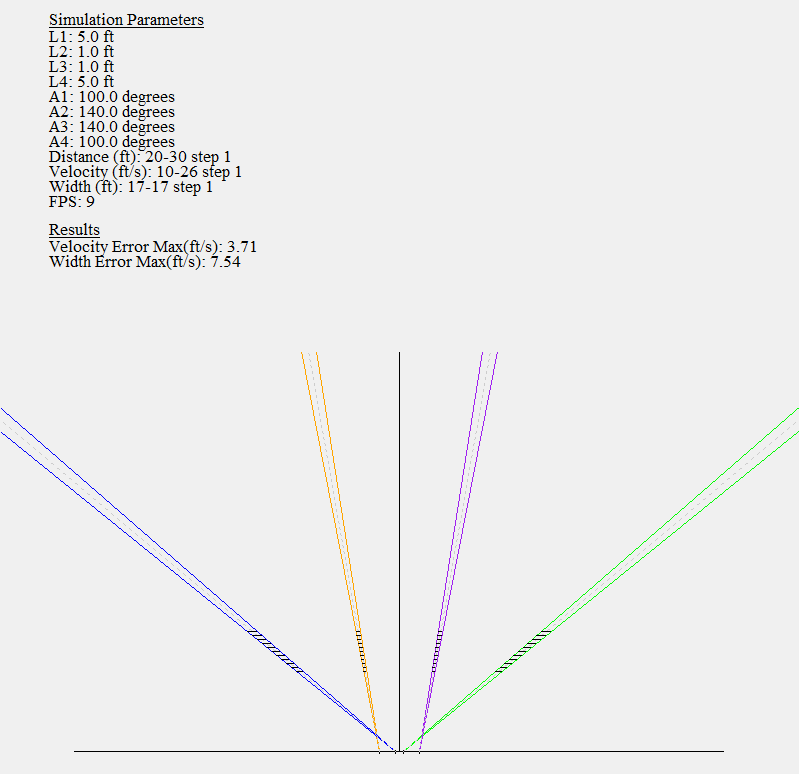
The optimal test object is a golf cart – capable of producing the range of speed and of the width desired.

However, due to the resources that were available for testing, a Chrysler Town and Country was the available vehicle with a built in speedometer, thus a vehicle with a width of 17 feet was instead used.

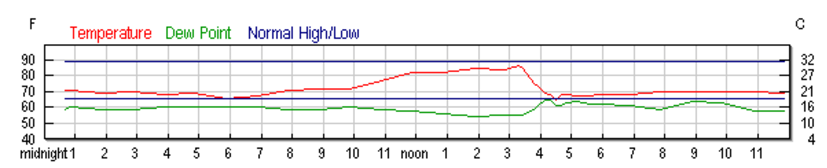
First Data Gathering Report

Matthew DeKoning

The first test site took place in an abandoned neighborhood on Kirtland Airforce base from 9 am to 12 pm. The location features a straight road with a loop to easily turn around. The sensor array was set up 25 feet from the straight road inside of the loop. The farthest two cameras were configured five feet from center focused at an angle 140 degrees from center and the middle two cameras were placed one foot from the center at an angle of 100 degrees from center.



The temperature rose from approximately 70 degrees Fahrenheit to slightly over 80 degrees Fahrenheit. A fairly linear increase of 10 degrees from 10 am to 12 pm.



<https://www.wunderground.com/history/airport/KABQ/2017/7/31/DailyHistory.html?req_city=&req_state=&req_statename=&reqdb.zip=&reqdb.magic=&reqdb.wmo>=

A dodge caravan van (17 feet wide) was driven at 15 – 25 miles per hour in both directions along the straight road as one test subject; a person walking in front of the cameras was used as the second test subject.

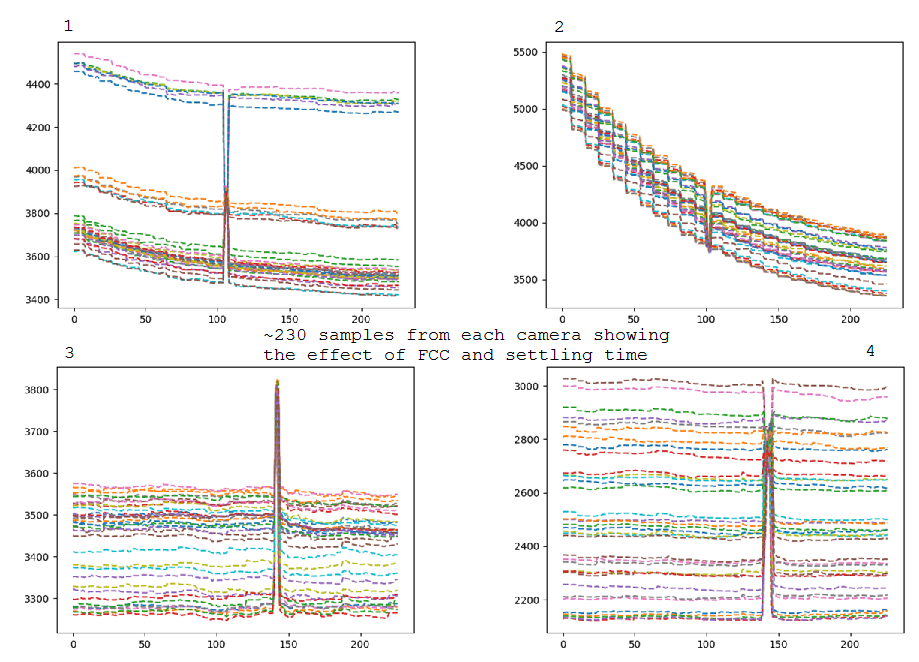
The Raspberry Pi, cameras, monitor, and input devices were powered with a large battery. A powered USB hub compatible with the Raspberry Pi had not yet been found, so a script was created to trigger the data gathering program on a timer, allowing the keyboard to be unplugged and the fourth Lepton to be plugged in (a Raspberry Pi only has 4 USB ports, so without a hub those are monopolized by the cameras).

The algorithm tested gathers a window of thirty two samples for each of the central six by six pixels of the cameras. The mean, median, and standard deviation (extracted via the Median Absolute Deviation calculation) are tracked and incoming data is tested against a certain statistical threshold to determine if it has changed due to an intrusion or simply a shift in ambient temperature. The threshold used here was six sigma (giving a probability of 99.9999998026825 percent that an incoming temperature reading would lie within the acceptable bounds, given the thirty two previous samples have created a reasonable statistical model for the temperature drift the camera is experiencing). When an intrusion is detected one of two things could happen: the alerted pixel count for this camera is incremented and the value is added to the thirty two value window, or the alert flag is incremented and the value is not added. The latter method attempts to leave the running statistics undisturbed by the intrusion, but the effect of the rise or fall in temperature on the value it eventually returns/settles to is unknown. However, the implementation of discarding intrusions was flawed during the first test and a pixel’s standard deviation could become zero. Given any statistically unlikely data will not be added to the running average, this freezes the window and causes constant alerting.

A variety of outputs for the tests were possible: camera number, time stamps, and pixel count changed (intrusion information), pure data – the value of each pixel analyzed paired with its index (camera, location within the six by six window) and statistics (mean, median, standard deviation), or pure data and intrusion information. The most useful proved to be both pure data and intrusion information.

The algorithm did not perform well on the first test for a variety of reasons. The first and largest reason is the settling time of the cameras was not taken into account. Since the USB plugs were constantly being swapped out to allow editing and script starting with the keyboard or run the algorithm with all four cameras, the cameras being swapped were not given time to adjust to the background temperature.

All graphs were created using Python and pyPlot.

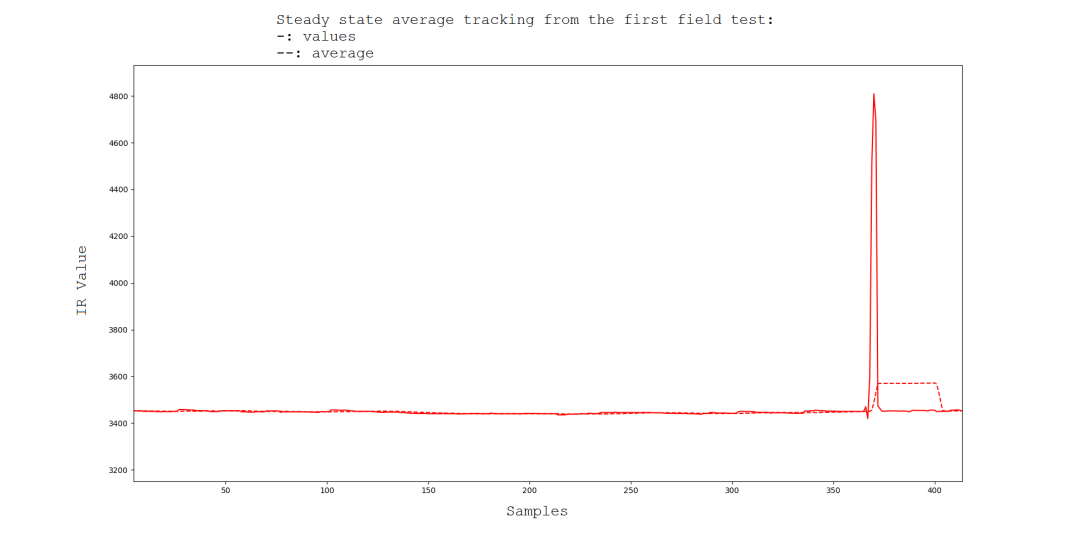
Problems

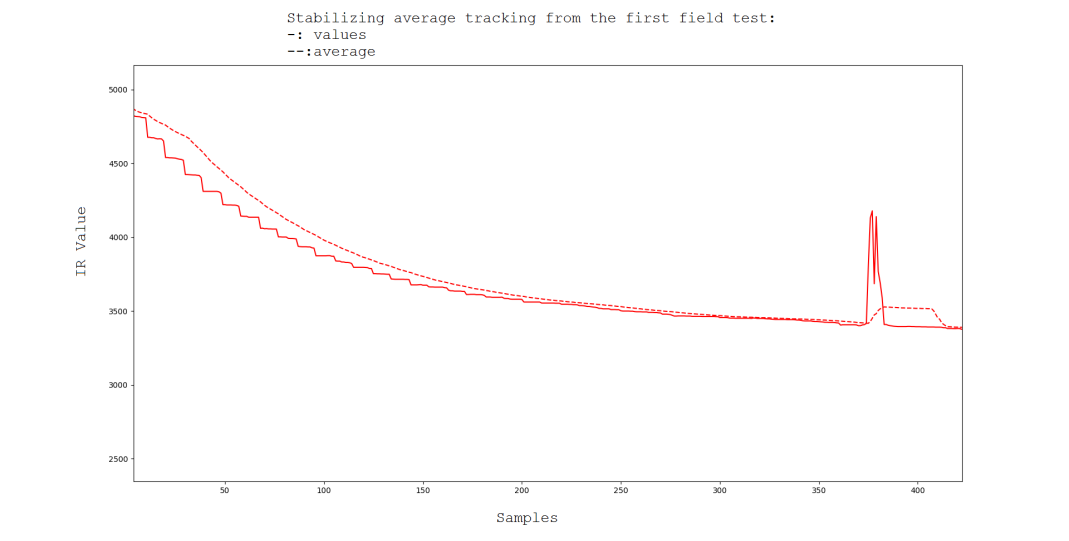
Figures 3 and 4 show cameras that have been allowed to adjust to the ambient temperature landscape they are viewing. Figure 4 shows a good range of temperature scenery being monitored, yet the intrusion around sample 140 still clearly changes each pixel value. Figures 1 and 2 show the cameras response as they perform flat-field correction to adjust to the ambient surrounding temperature. Figure 2 in particular shows the effect of the FCC, which is triggered ever second or every nine frames.

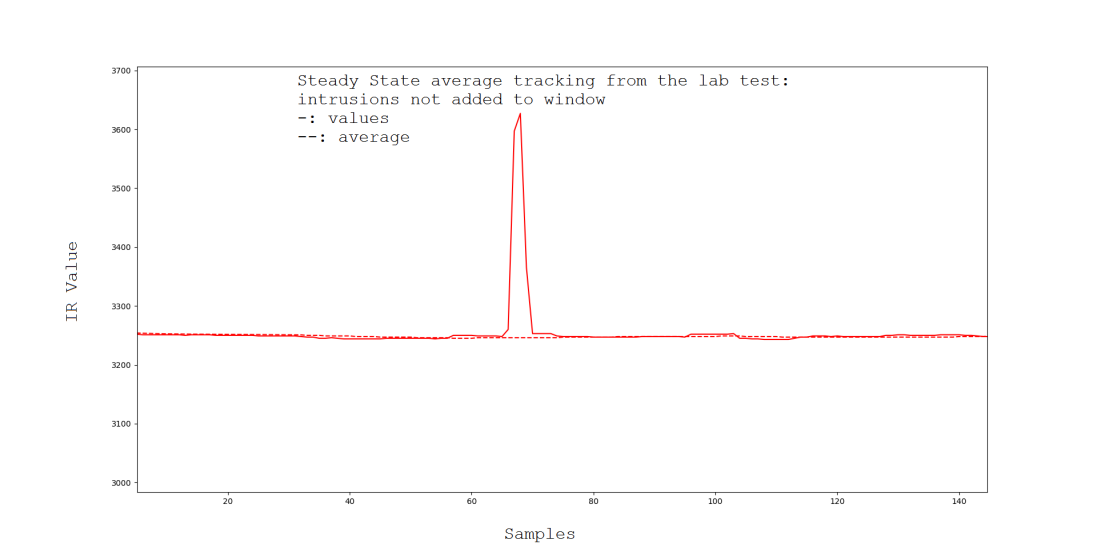
These graphs were obtained during post-processing. In the field I was aware the algorithm was not behaving well, but unsure of where to begin in terms of fixing it. After I began looking at the data, I set up the cameras in the lab, all 90 degrees from the ‘x axis’ and spaced 3 feet and 1 foot from the center. With this setup I determined the cameras need at least a minute if not two to settle before reading their data, as well as found the standard deviation bug. A powered USB hub that allowed all peripherals to always be plugged in was also located.

Validations

The statistics (aside from steady state standard deviation) worked well.

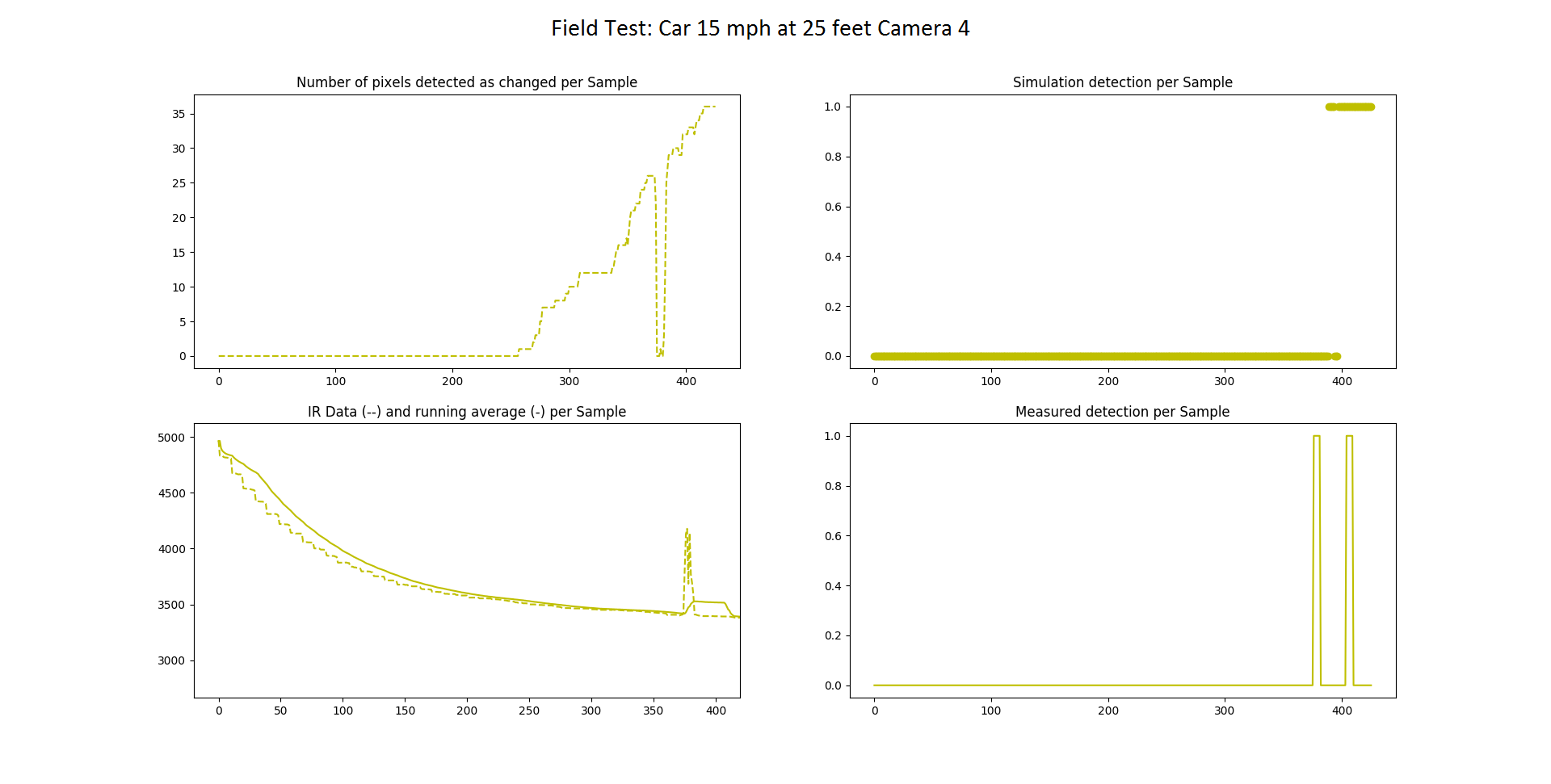
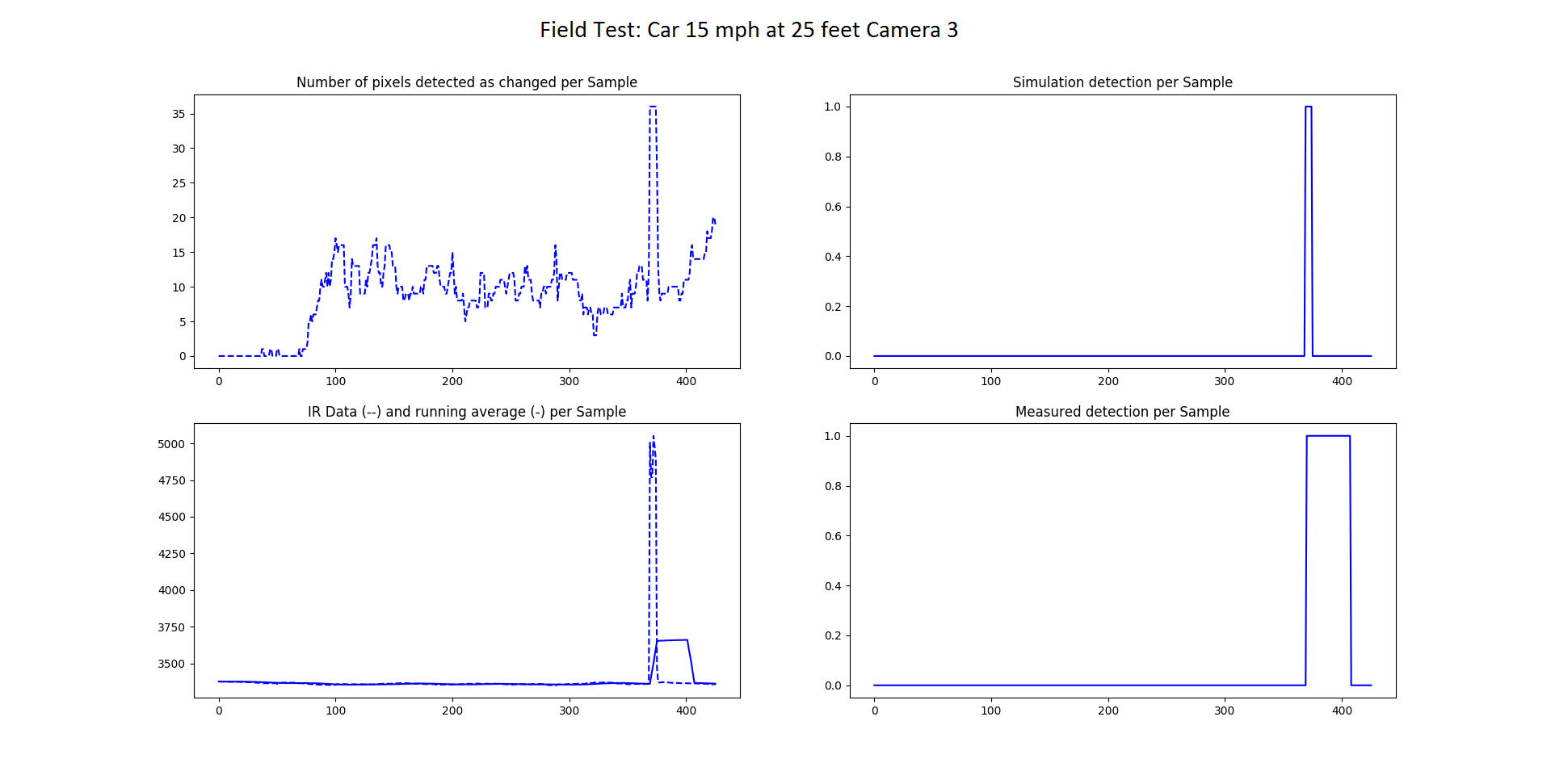
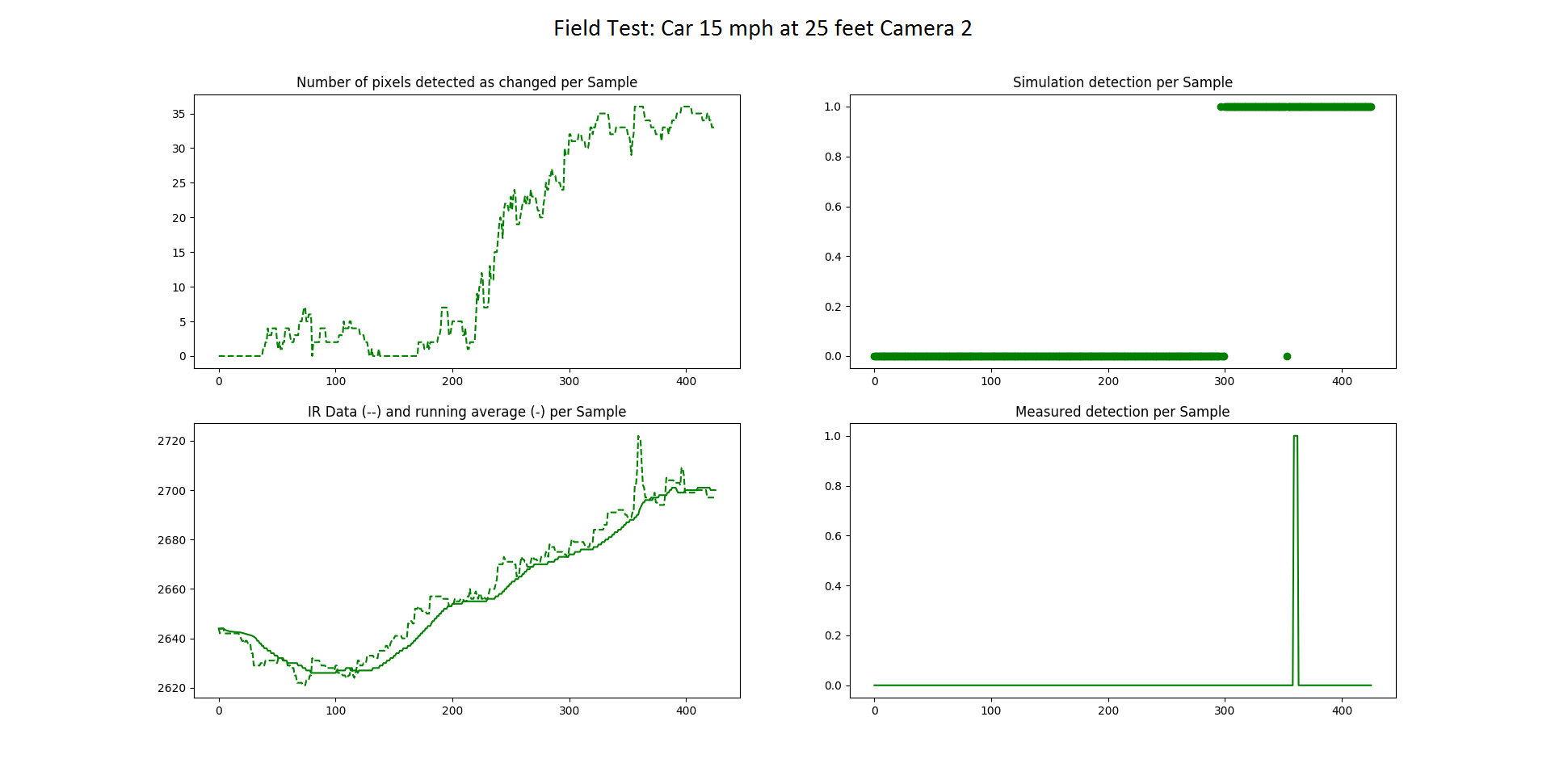
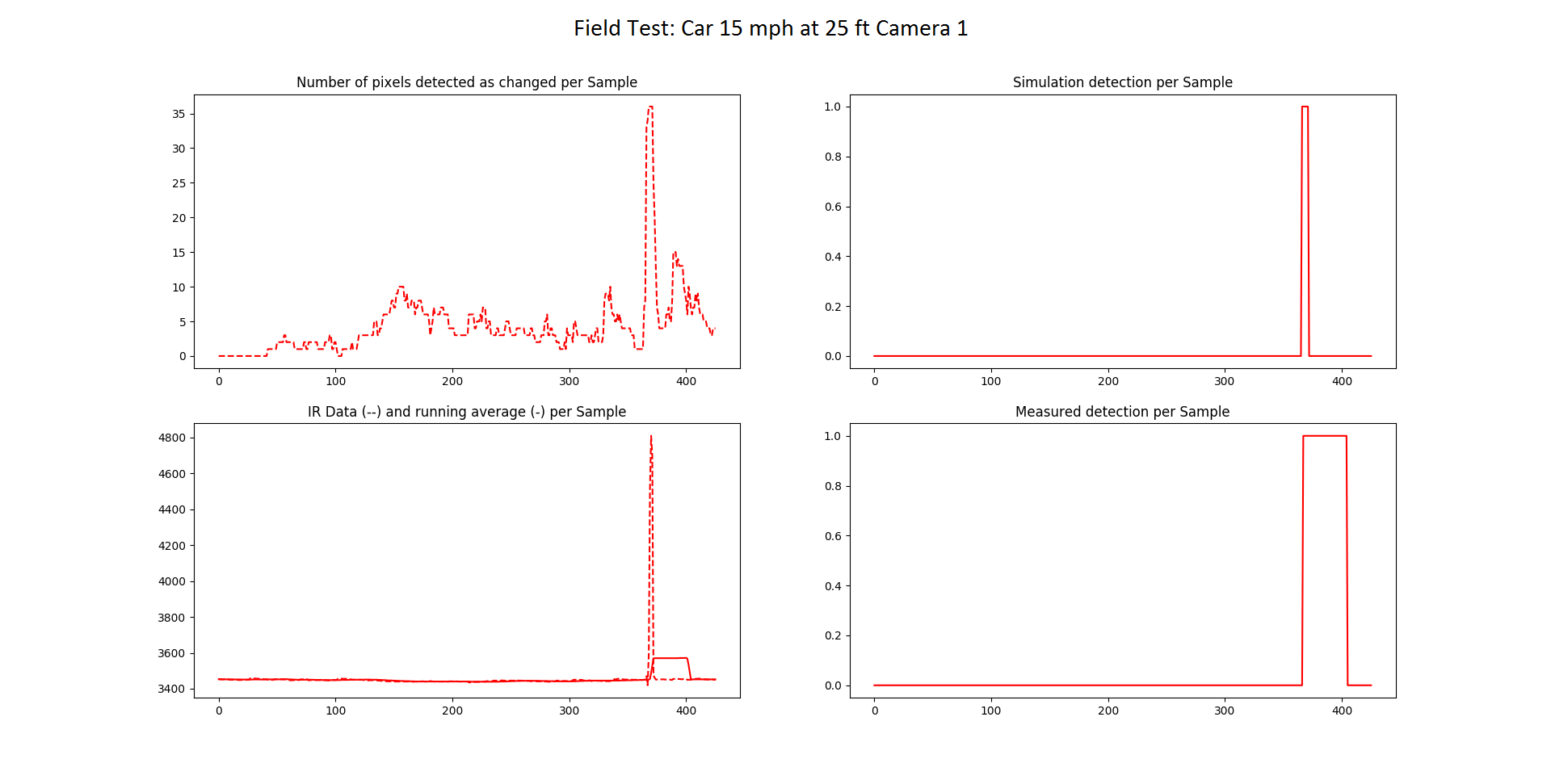
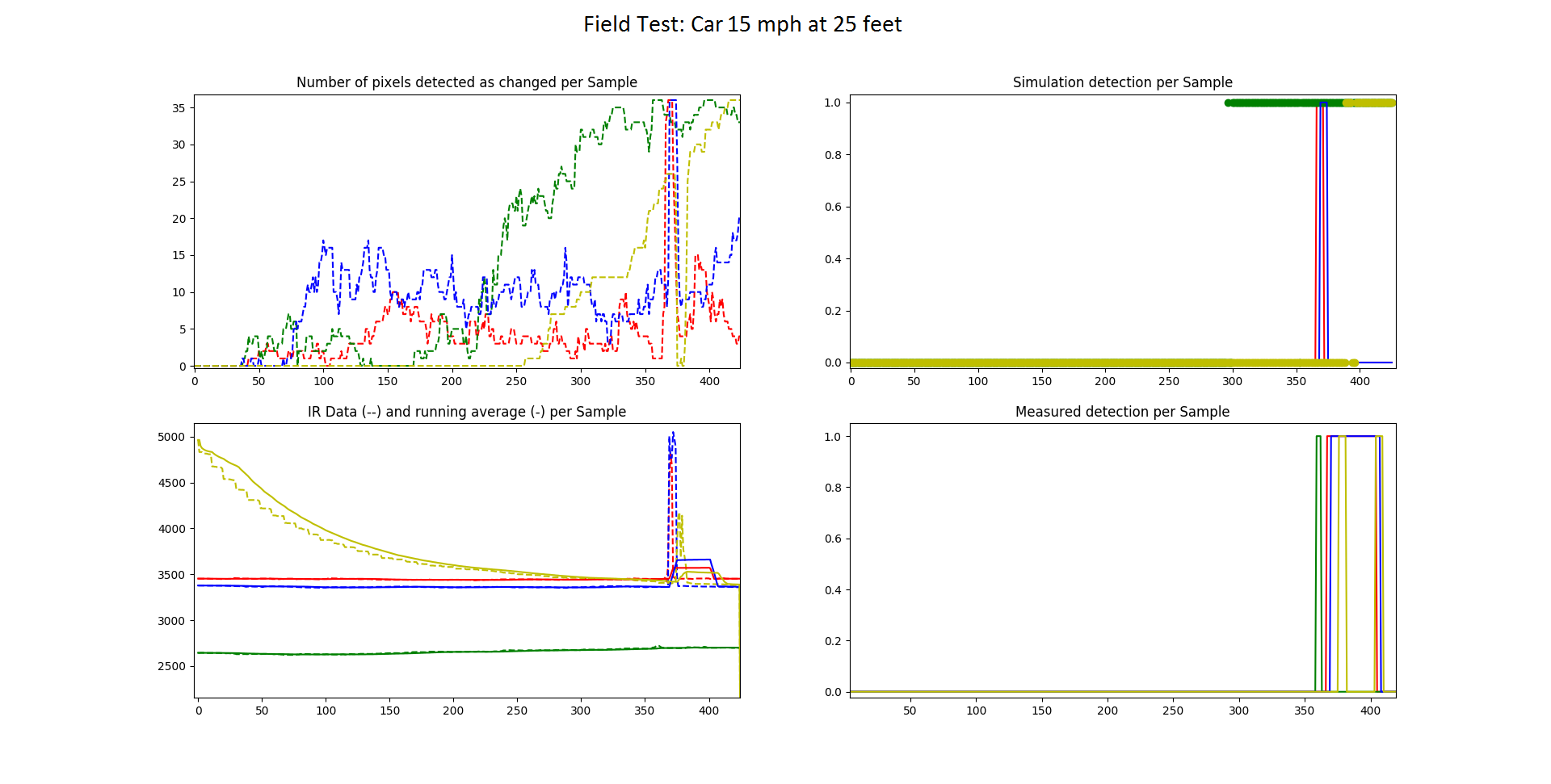






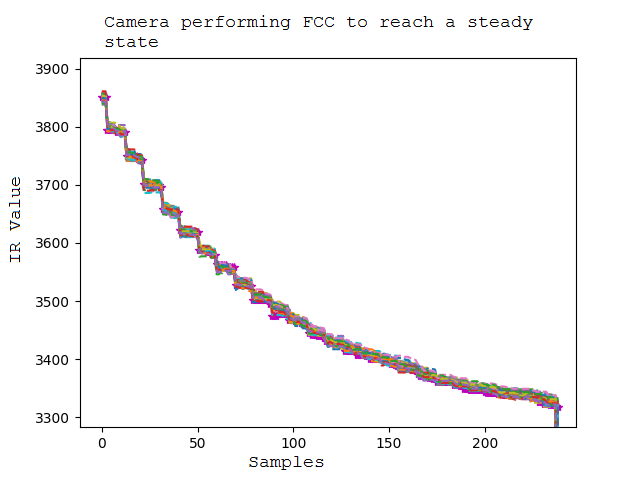
Results

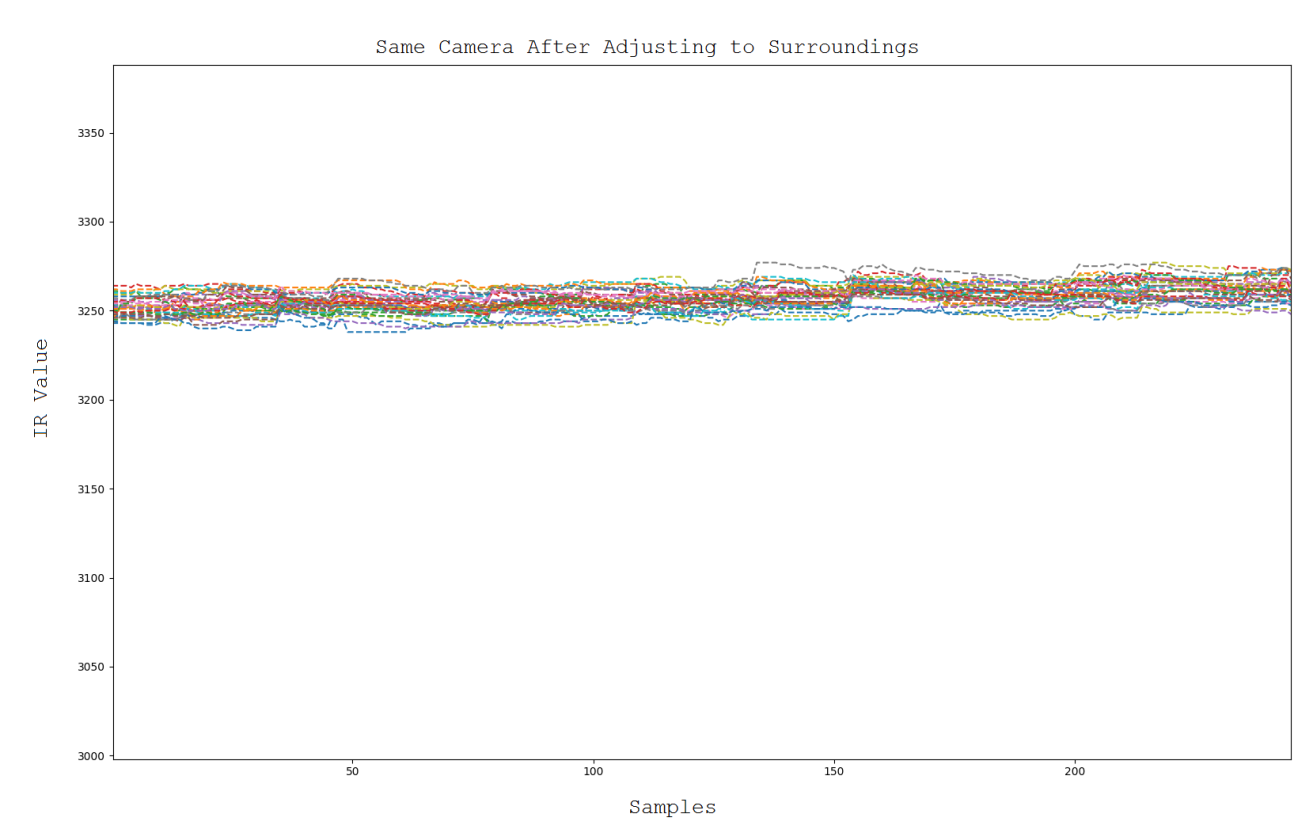
The following graphs show data gathered and algorithm results versus simulated results run over the same data. Both simulation and the Raspberry Pi algorithm used a statistical threshold of six sigma, but the field test had a changed pixel to detection threshold of 25 while the simulation had 30.



Investigating Discontinuities

The data below was captured in an air conditioned (more thermally stable than outside) lab when trying to determine the cause of the discontinuities in the previous graphs.

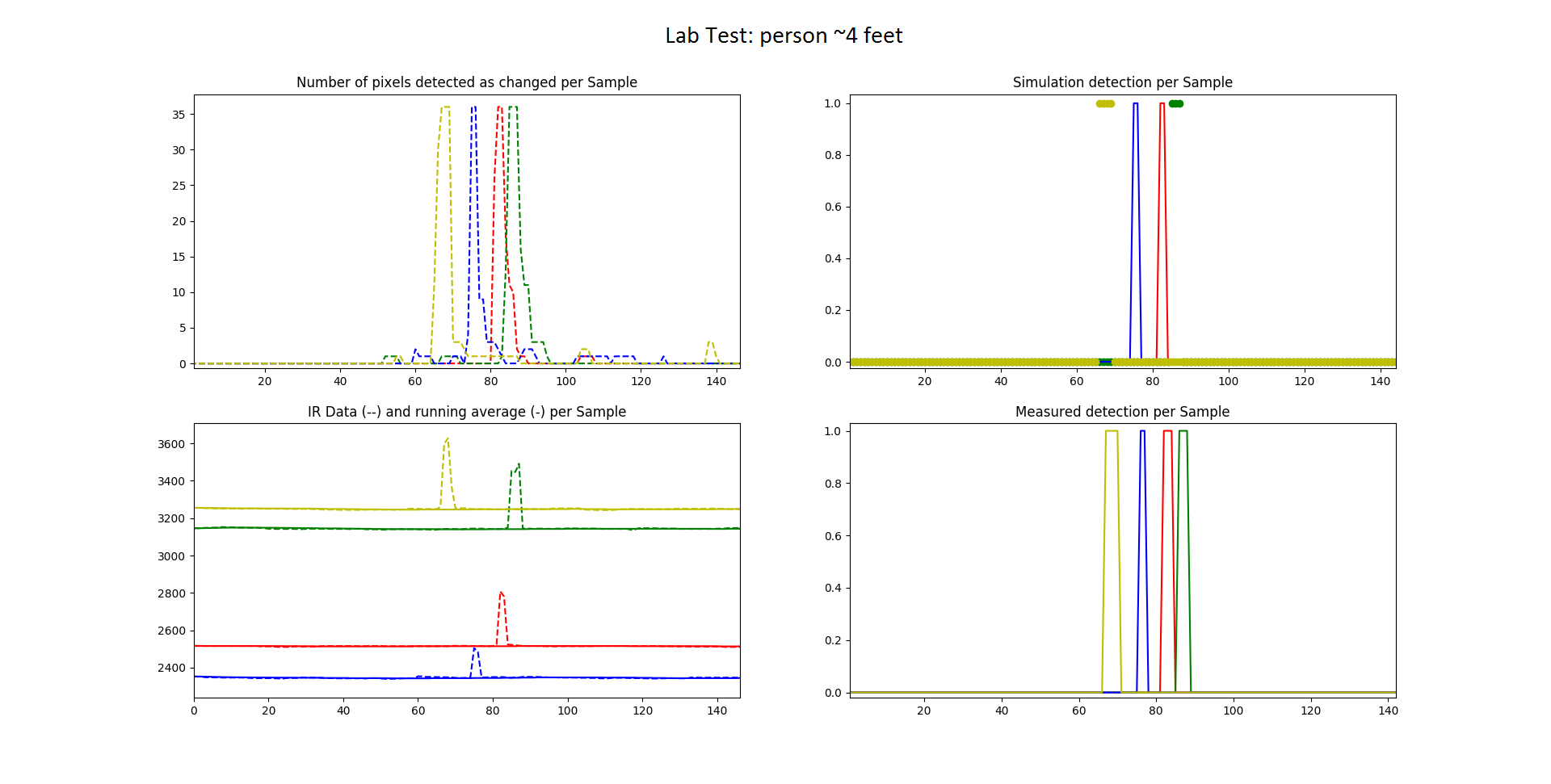


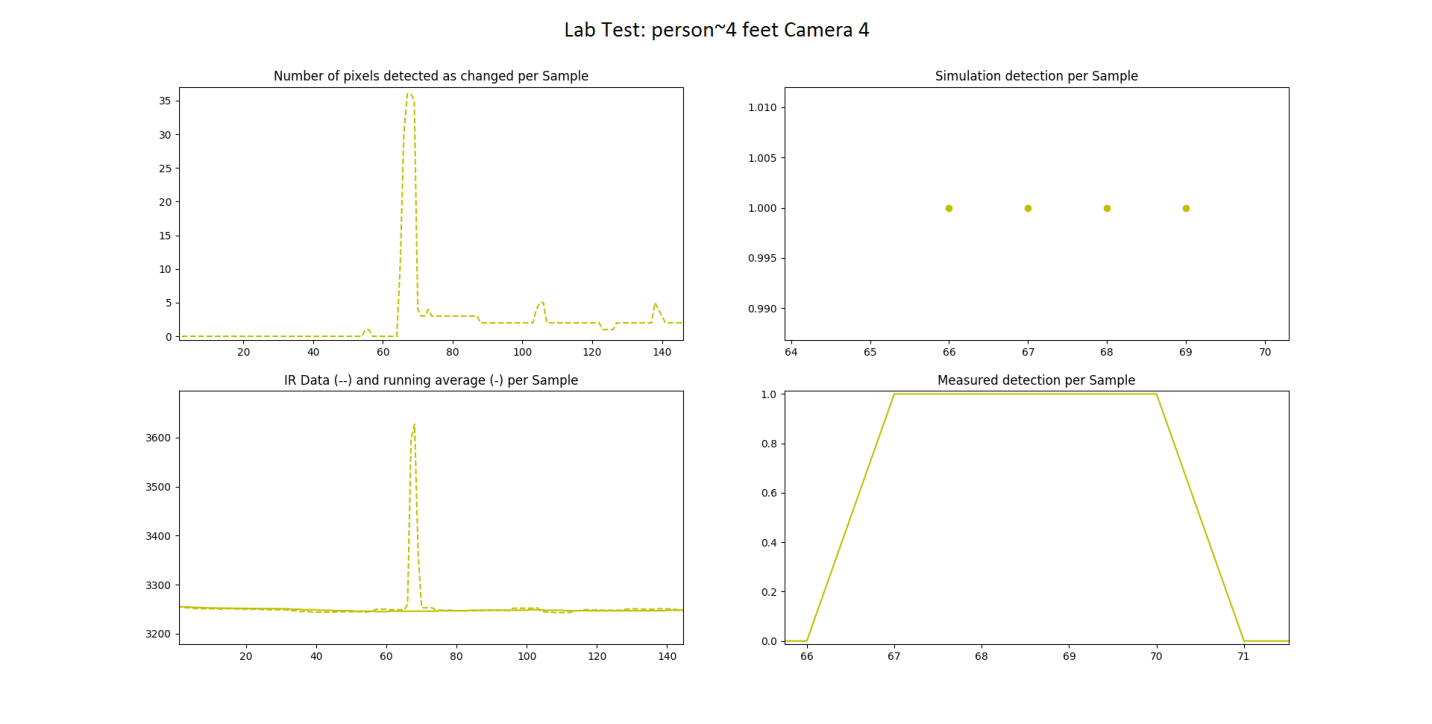
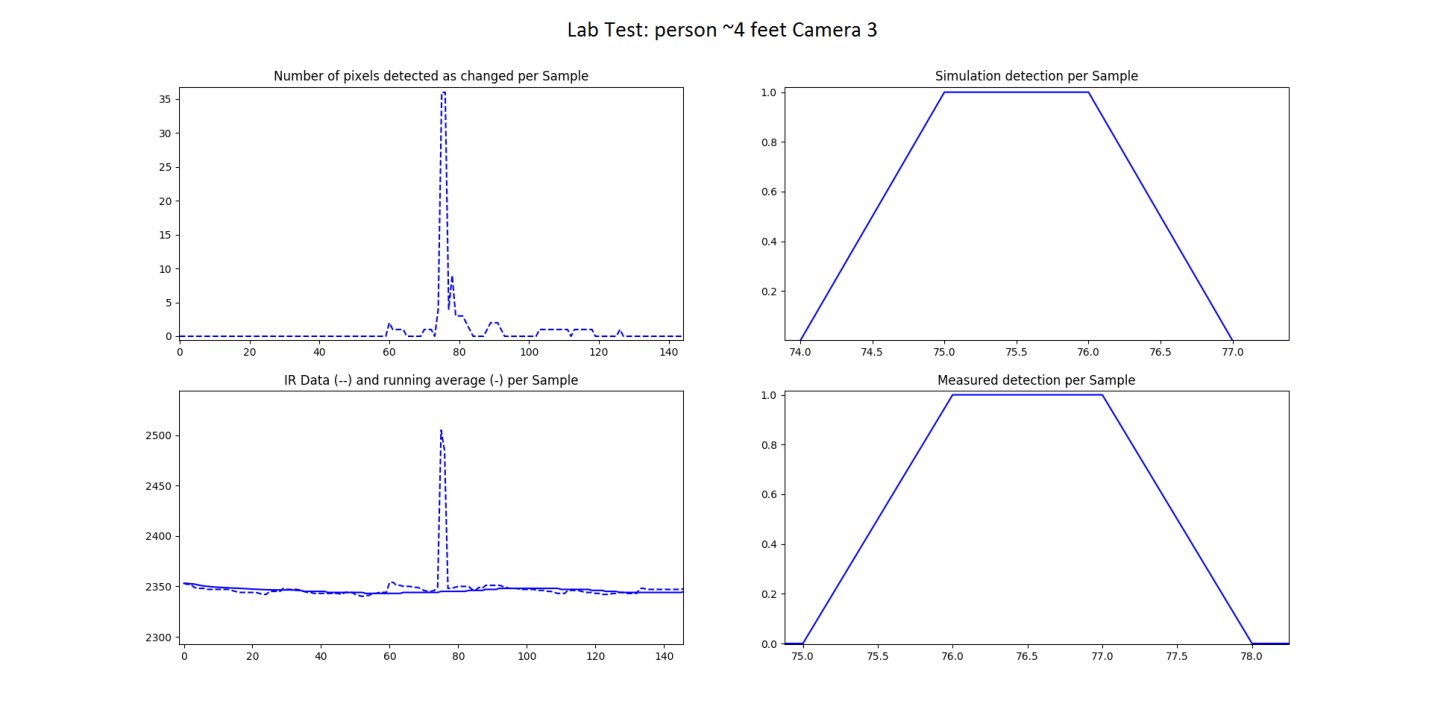
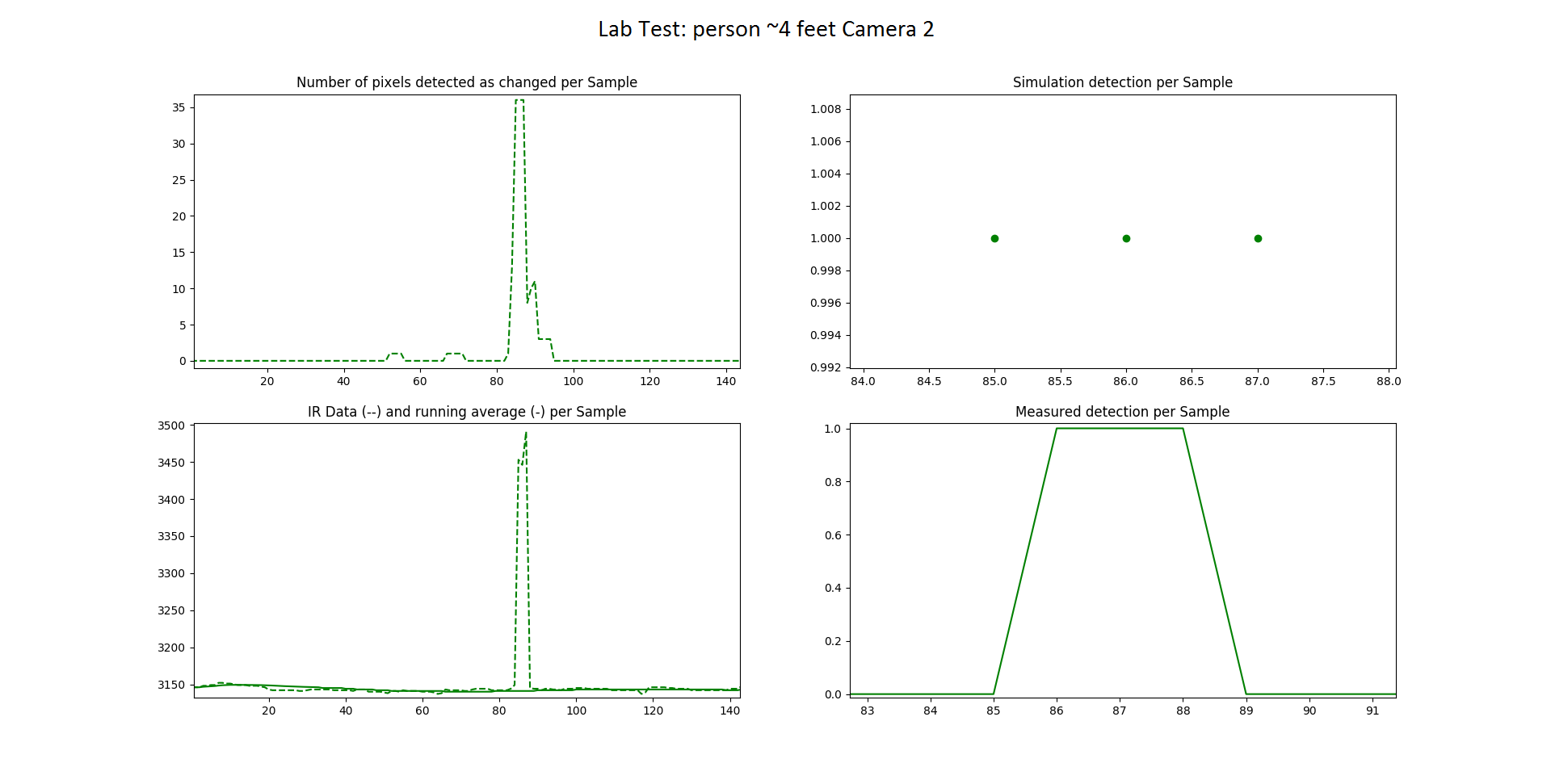
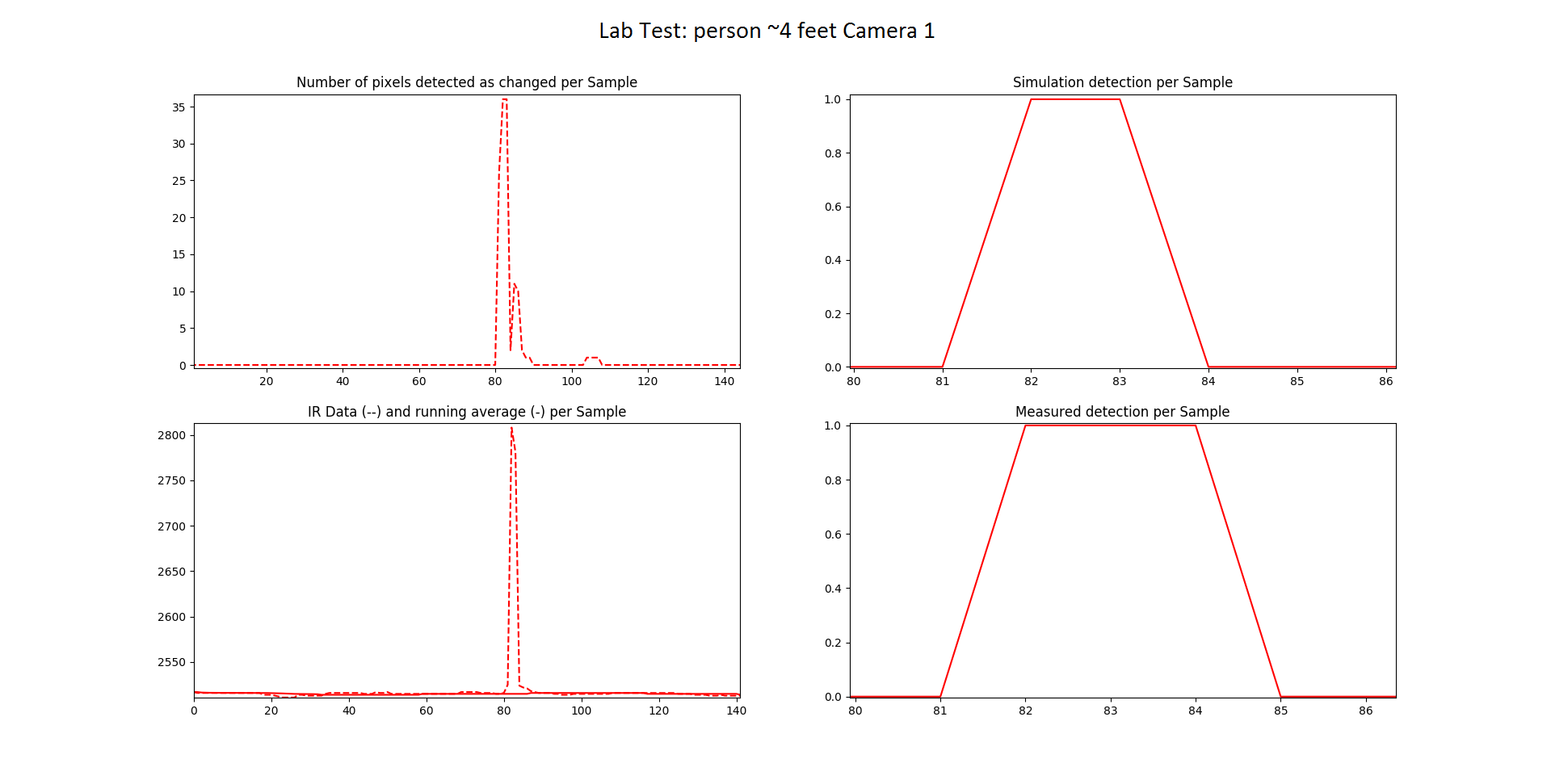


Given the powered USB hub will now allow cameras to remain plugged in during all testing, the rough estimate of 90 seconds for camera stability is acceptable, the cameras will be plugged in for multiple minutes before any future tests, so the problem should be resolved.

Further Work

The following are graphs created from data in a lab after debugging the algorithm. The statistic threshold on the Raspberry Pi was changed to 27, aside from that the test remained the same.





This data shows nearly a one to one detection in simulation to detection in practice after adjusting the algorithm and sensor setup based on the knowledge gained from the first test.

False alarm changed pixels – those that are triggered by thermal noise or other interferers – in this scenario are kept beneath five per frame while residual pixels – those registering a changed due to a recent intrusion – are kept beneath sixteen. This suggests the changed pixel threshold could be lowered to around twenty and the results would remain the same. However this is in a much less noisy lab environment. The next step is to do some basic tests outdoors again, then set up another large data gathering day.

Calibration Tests

Second Test Documentation

Post Processing

Conclusion