Automating 3D Wireless Measurements with Drones

Ethan Yu, Xi Xiong, Xia Zhou
Department of Computer Science, Dartmouth College
{ethan.yu, xi.xiong.gr, xia.zhou}@dartmouth.edu

ABSTRACT

Wireless signals and networks are ubiquitous. Though more reliable than ever, wireless networks still struggle with weak coverage, blind spots, and interference. Having a strong understanding of wireless signal propagation is essential for increasing coverage, optimizing performance, and minimizing interference for wireless networks. Extensive studies have analyzed the propagation of wireless signals and proposed theoretical models to simulate wireless signal propagation. Unfortunately, models of signal propagation are often not accurate in reality. Real-world signal measurements are required for validation.

Existing methods for collecting wireless measurements either involve researchers walking to each location of interest and manually collecting measurements, or place sensors at each measurement location. As such, they require large amounts of time and effort and can be costly. We propose DroneSense, a system for measuring wireless signals in the 3D space using autonomous drones. DroneSense reduces the time and effort required for measurement collection, and is affordable and accessible to all users. It provides researchers with an efficient method to quickly analyze wireless coverage and test their wireless propagation models.

Keywords

Wireless measurements; drones; wireless signal propagation

1. INTRODUCTION

Wireless signals and networks are ubiquitous today. Though more reliable than ever, wireless networks still struggle with problems such as weak signals, blind spots, and interference. Key to solving these problems is the understanding of wireless signal propagation. Studies on wireless signal propagation often rely on theoretical models [3, 9, 12, 15, 16, 21, 27, 28]. Although these models have improved our understanding of wireless signal propagation, they often fall far short in accuracy. To gain accurate views of wireless signal distribution in the space, we need actual signal strength measurements.

One method for collecting highly granular wireless measurements is war driving/walking [5, 14, 26]. Researchers must collect measurements at every point of interest by driving or walking to each location. War driving/walking is extremely troublesome, because it requires constant human interaction. Moreover, human interaction can affect the signal propagation and interfere with the wireless measurement results [13, 30]. Another method for collecting wireless measurements is to place sensors at every location of interest, and then repeatedly read measurements from those sensors. However, the granularity of measurements is limited by the number of deployed sensors. It can also entail a high cost for a large number of measurement locations.

Figure 1: DroneSense in action, collecting Wi-Fi signals in a hallway using its built-in Wi-Fi radio. Paper marks on the floor indicate the target measurement locations.

We propose *DroneSense*, a system for automatically collecting wireless signal measurements in the 3D space using drones. To meet our objective, we program a drone to fly along a preset trajectory and collect measurements along its path. The user inputs a list of coordinates of where to collect measurements. Our system then instruments the drone to navigate the 3D space and collect signal measurements at specified 3D locations.

To utilize drone effectively for wireless measurements, we face two challenges. First, it is difficult to track the precise 3D position of the drone indoors. Thus, navigating the drone to collect measurements at specified locations is challenging. Second, the drone is extremely unstable during the flight. Without the proper control, it can frequently undershoot or overshoot a target location. As a result, it spends much time on adjusting its position before reaching the target location, making the measurement time-consuming and inefficient. To address these challenges, we improve the navigation accuracy by leveraging reference points in the environment. We instrument the drone to detect these reference points and then apply Extended Kalman Filter to estimate the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WiNTECH'16, October 03-07, 2016, New York City, NY, USA

© 2016 ACM. ISBN 978-1-4503-4252-0/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2980159.2980168

drone's position. To improve the drone's flight stability, we apply the Proportional-Integral-Derivative Controller to dynamically control the drone's trajectory and improve the flight efficiency.

We implement DroneSense using the Parrot AR Drone 2.0 [23] for its low cost, abundance of open-source APIs, and durability while flying indoors. We implement the navigation and measurement program in JavaScript on the Node.js platform, which is compatible with all operating systems. Our system can ensure 13-cm mean error and 20-cm maximal error from the target location. With this error tolerance, drone flight is efficient, with distance to the target monotonically decreasing, no overshoot or adjustment. Furthermore, our experiments demonstrate that measurement at each location for three seconds is sufficient for accurate results.

We summarize the key contributions of DroneSense as follows:

1. DroneSense provides a fully autonomous process for collecting wireless measurements, which eliminates the human interactions, reduces the required human efforts, improves the efficiency of measurement process, and is easily scalable to a large experiment site.

2. DroneSense enables collecting signal strength measurements in the 3D space, enriching the measurement data for analysis and allowing more accurate understanding of wireless signal propagation in a 3D environment.

3. We present the user an easy way to set the drone's trajectory, various options for collecting signal measurements, and a suite of tools for automation and calibration.

In the rest of this paper, we discuss in detail the technical challenges of creating this system, DroneSense's design, the methods we utilize, the evaluation of our results, summarize DroneSense's effectiveness compared to related works, and discuss its impact and how our work can be extended.

2. OVERVIEW AND CHALLENGES

Overview: DroneSense takes as input a flight path, which is a list of locations for a drone to collect measurements. The user also specifies options for collecting measurements, such as the number of measurements at each point and the time duration between adjacent measurements. For each location of interest, the drone hovers over it for a specified duration, and keeps recording the wireless signal strength received from a given access point (AP) using its built-in radio. When multiple APs are present, the drone's built-in radio can be configured to the monitor mode, where the radio scans all wireless channels and collects received signal strength values from each AP. By averaging the signal strength values at a given location, we can smooth out signal fluctuation caused by fast fading and gain a relatively stable view of the signal strength distribution in the space.

DroneSense is divided into two modules: Navigation and Measurement. The Navigation Module is responsible for moving the drone to the desired locations. The Measurement module is responsible for collecting measurements at each desired location. In creating an automated system for drones to collect wireless measurements, we faced challenges in developing precise control over the drone's movement. Two notable challenges are accurate navigation and efficient flight. Accurate navigation means accurately determining the drone's current location relative to the target's location in order to accurately navigate to the target. Efficient flight means reaching the target location quickly without having to make many large adjustments in flight trajectory.

Challenges: The navigation accuracy challenge arises because currently there is no robust and accurate indoor drone localization system available. It is difficult to leverage GPS to achieve centimeter localization accuracy indoor. Additionally, the characteristics of the drone control API do not provide the ability of dead reckoning. When communicating with the drone, control commands are sent to the drone as a pair of direction and magnitude, such as "left 1.0" or "up 0.5". The magnitude is most analogous to motor speed, but does not map directly any physical unit of movement such as velocity or acceleration. Though theoretically we can infer drone's velocity and heading direction from accelerometer and gyroscope sensors, the sensor data are too noisy and errors accumulate over time so that they could not be used directly.

These observations motivate us to leverage feedback from environment to account for noise from drone's sensor readings and track the its position precisely. We attempted to apply camera-based navigation method [10]. This method investigates drone's front camera for pose estimation. It fuses visual information captured by the front camera with sensor data to infer the drone's state. However, it requires there exists distinct visual features in the images captured by the front camera. If there was only a white wall in front the drone, the navigation would easily go wrong. So we failed in leveraging this method to collect wireless measurements among the entire room. Our solution to this challenge is using Reference Point Detection and an Extended Kalman Filter, discussed in section 3.1 and 3.2.

The other main challenge is to ensure the drone flow in an efficient path. The impact of this problem is not just that the drone will take an indirect path to the target, but more importantly when the drone is near its target, its position will fluctuate wildly as it tries to adjust itself into the exact target position. Since the drone has a limited battery life, it is important to ensure flights as efficient as possible so that more flight time can be achieved. The efficiency of the flight depends on the magnitude of the control commands we send to the drone, which will determine how quickly the drone reaches its target. The problem of efficient flight arises because we do not have an effective method to determine the magnitude of the command to send to the drone.

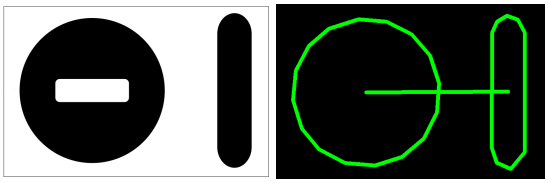
The shortcoming of setting the control magnitude as a constant value is that it does not adjust its speed dynamically according to the distance from a target. This is inefficient since drone has a limited battery life. Thus we attempted to choose magnitudes proportional to the distance to a target. However, this resulted in the drone constantly undershooting or overshooting the target, as the drone was unable to stop right over the target. To make the drone's flight more efficient, we must receive feedback about what is the effect of the magnitude we used, and adjust the magnitude on a continuous and non-linear scale in order to find the optimal magnitude. Our solution to this challenge is the Proportional-Integral-Derivative Controller, discussed in section 3.3.

3. SYSTEM DESIGN

We now describe in detail three key design components in DroneSense to address the above challenges.

3.1 Reference Point Detection

We use reference point detection to help address the challenge of navigation accuracy, by providing position calibrations for the drone. When the drone detects a reference with its camera, we can calculate the reference point's relative position and orientation to the drone in 3D space. We can also predefine each reference point's true position in 3D space so that the program knows the the true position of the detected reference point. Thus, when a



(a) (b)

Figure 2: (a) Reference point. (b) Reference point detected by OpenCV. A line between center of the circle and center of the rounded rectangle is drawn to infer the angle of orientation.

marker is detected, we can use the its true position and its relative position and orientation to the drone to calculate the drone's own position and orientation. We place multiple reference points into the environment to so the drone can constantly have a source of position calibration.

The drone has the capability to detect certain types of markers, including the Oriented Roundel marker that we use. Figure 2(a) shows the Oriented Roundel marker, and we see that it is not only very easy to recognize, but also we can easily determine its orientation. Once a marker is detected, we perform calculations to transform its position in the camera image to its position in the drone's coordinate system, and also find its real position.

Computer Vision Process: First, we convert the image to grayscale. Next, we run the OpenCV Canny Edge Detection algorithm to produce an image where only the edges pixels are turned on. Next, we dilate the image to increase the size of the edges. Next, we use the OpenCV Contour algorithm and find circles and rounded rectangles in the image based on number of sides for each detected contour. Finally, we process detected shapes to determine if a marker is present in the image. If there exists simultaneously a rounded rectangle whose long side length is equal to the side length of a large circle's bounding rectangle, there is a marker in the image. After a marker is detected, we set the center of the circle as the location of the marker. We draw a line between the center of the circle, and the center of the rounded rectangle, shown in Figure 2(b). The angle between that line and the X axis is the angle of orientation. We then send the coordinates and the orientation to be converted to 3D coordinates.

Back Projection: The camera has a frame of 640×360 pixels. We need to translate the location of the detected marker from these 640×360 coordinates to 3D coordinates. In other words, we need to translate the marker's 2D position relative to the center of the drone's camera (in pixel units) to the marker's 3D position relative to the center of the drone itself (in meter units). Using the 3×3 back projection matrix $\mathbf{P}$ that has been calibrated for the drone's bottom camera [4], we can calculate the center of the mark's circle in drone's coordinate space:

$$[x'_c, y'_c, z]^T = \mathbf{P}^{-1}[0.64 * x_c * z, 0.36 * y_c * z, z]^T,$$

where (x'_c, y'_c, z) is the coordinates of the center of the mark's circle in drone's 3D space, (x_c, y_c) is the pixel coordinates of the center of the mark's circle in captured image, and z is the height measured by drone's ultrasonic sensor. We can calculate the coordinates of the center of rounded rectangle (x'_r, y'_r, z) in drone's space as well. Finally, simple trigonometry allows us to determine the orientation of the marker θ:

$$\theta = \arctan(\frac{y'_c - y'_r}{x'_c - x'_r})$$

Thus, we are able to determine the marker's 3D coordinates relative to the drone, as well as the orientation of the drone.

Reference Point Placement: We place markers in the environment 1 meter apart from each other. Fewer markers are needed for the drone to sufficiently navigate, but for more accuracy we try to use as many markers as possible. Because we know that markers are placed at every meter, when a marker is detected, we can assume that marker's true position is the nearest whole-meter coordinate from the drone's current position. If multiple markers are detected, their true positions are the nearest whole-meter coordinates sorted by distance from the drone's current position. For instance, if the drone's current position estimate is (1.7, 2.2), the nearest marker from the drone detected by the camera should have true position (2, 2). Knowing the marker's true position and its relative position and orientation to the drone allows us to calculate the position and orientation of the drone.

With markers placed every meter, locating each marker becomes simpler. It also helps us visually inspect the navigation accuracy of the drone. We can reduce the number of markers by placing them with larger intervals (e.g., every two meters), and configure the drone's program to round its current coordinate to the nearest location two meters away upon detecting a marker. To support irregularly spaced markers, we can indicate the position of the marker on the marker itself with certain predefined special symbol. The system can build a lookup table to map a symbol to a marker location, so that the drone can search through the lookup table to calibrate its location.

3.2 Extended Kalman Filter

We use the Extended Kalman Filter (EKF) [17] to address the challenge of navigation accuracy, by using it to estimate the drone's position at all times. An Extended Kalman Filter is an algorithm that uses a series of measurements over time, containing noise and inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone. This algorithm is frequently used for guidance and navigation of vehicles [2].

The EKF uses a two-step recursive process: predict and update. The predict step uses a position estimate from a previous time as well as sensor data to estimate the current position. Below, we give a brief description of how we predict and update the position estimate. We denote the sensor readings as *sensors*, and the camera reference point observations as the *observations*. We use the sensors and past position estimates to estimate the current position, in other words perform the predict step. We use the observations and past position estimates to correct the model and improve the position estimation, in other words perform the update step.

Predict Step: The predict step utilizes the last position estimate, the velocity readings derived from the accelerometer, the yaw reading derived from the gyroscope, and time interval since last estimation to estimate the current position and orientation. Thus, our position estimate vector $\hat{\mathbf{x}}$ contains the values of x, y, and yaw, and our sensor vector $\mathbf{u}$ contains the values of x velocity and y velocity, yaw, and time interval:

$$\hat{\mathbf{x}} = [\hat{x}, \hat{y}, \hat{\theta}],$$

$$\mathbf{u} = [v_x, v_y, \theta_u, dt]$$

Using our model based on the laws of physics, we know that the current position should be the past position plus the change in position during the last time interval due to velocity, with a modification for rotation. Our position transition functions f are shown:

$$\mathbf{f} = [f_x, f_y, f_\theta]$$

$$f_x(\hat{\mathbf{x}}_k, \mathbf{u}_k) = \hat{x}_k + v_x * dt * cos(\hat{\theta}) - v_y * dt * sin(\hat{\theta})$$

$$f_y(\hat{\mathbf{x}}_k, \mathbf{u}_k) = \hat{y}_k + v_x * dt * sin(\hat{\theta}) + v_y * dt * cos(\hat{\theta})$$

$$f_\theta(\hat{\mathbf{x}}_k, \mathbf{u}_k) = \theta_u$$

Finally, having all of these variables and equations defined, the predicted position $\hat{\mathbf{x}}$ at time k is calculated from the state at time $k-1$ as below:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u_k}),$$

where the $\hat{\mathbf{x}}_{k|k-1}$ denotes a priori state estimate at time k given observations up to $k-1$, and $\hat{\mathbf{x}}_{k-1|k-1}$ represents a posteriori state estimate at time $k-1$ given observations up to and including at time $k-1$.

Update Step: The update step utilizes an observation of the actual position to update, or correct, the model parameters. In our case, when the drone camera detects a reference point, it observes the marker's relative position and rotation, and it knows the marker's true position and rotation. Thus, we can infer the drone's true position based on the discrepancy between the marker's true and relative position. However, although the position estimate derived directly from the reference point detection is likely to be quite accurate, it still may not be perfectly accurate. The update step combines the position estimated in the predict step and the position calculated from the observation, balanced using a weighted average based on uncertainty, to produce a position estimate likely more accurate than both individual calculations.

The observed values will be contained by the true and relative observation vectors $\mathbf{t}$ and $\mathbf{z}$ shown:

$$\mathbf{t} = [x_t, y_t, \theta_t]$$

$$\mathbf{z} = [x_z, y_z, \theta_z]$$

We use the true marker position and current drone position to calculate what the marker's true position should be relative to the drone, or the marker's theoretical relative position. We call the equations for this calculation our observation transition functions. Our observation transition functions h are shown as:

$$\mathbf{h} = [h_x, h_y, h_\theta]$$

$$h_x(\hat{\mathbf{x}}_k, \mathbf{t}_k) = (x_t - \hat{x}) * cos(\hat{\theta}) + (y_t - \hat{y}) * sin(\hat{\theta})$$

$$h_y(\hat{\mathbf{x}}_k, \mathbf{t}_k) = -(x_t - \hat{x}) * sin(\hat{\theta}) + (y_t - \hat{y}) * cos(\hat{\theta})$$

$$h_\theta(\hat{\mathbf{x}}_k, \mathbf{t}_k) = \theta_t - \hat{\theta}$$

Next, we calculate the residual which is the difference between the two relative positions: the relative position of the marker calculated from the camera back projection, and the theoretical relative position of the marker's true position.

Table 1: PID Controller Coefficients

PID	K_p	K_i	K_d
X	0.5	0.1	0.35
Y	0.5	0.1	0.35
Z	0.8	0.2	0.25
Yaw	0.8	0.2	0.25

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}, \mathbf{t}_{k|k-1})$$

Finally, having all of these variables and equations defined, we can calculate the measurement residual covariance and the optimal Kalman Gain with an assumption about the covariance of the observation noise, which in turn allows us to calculate the updated position and the updated covariance.

3.3 Proportional-integral-derivative Controller

We use a Proportional-Integral-Derivative (PID) Controller to address the problem of flight efficiency, by calculating appropriate magnitudes for the control commands for the drone. We want to send the drone control command magnitudes that efficiently move the drone to the target quickly, with few large adjustments, and minimizing overshoot. A PID Controller is suitable for our problem because it allows us to dynamically adjust the command magnitude in a non-linear fashion.

We use a PID Controller for each of the four directions of motion: front or back, left or right, up or down, clockwise or counterclockwise. Our PID controller seeks to minimize distance in a certain direction by setting the control command magnitude u, and uses the feedback to continuously adjust the control variable until a steady state is reached. The PID equation is shown as following:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt},$$

where u is the control variable, e is error from feedback, K_p is proportional coefficient, K_i is integral coefficient, and K_d is derivative coefficient. The proportional term accounts for the present error. The integral term accounts for the total accumulated values of the error, which accelerates the process to the desired level, and eliminates steady state error. And the derivative term accounts for the future values of the error, based on the rate of change of the error. The derivative term helps prevent overshoot by modulating the magnitude. By modulating the derivative of the error rather than the proportional error, the drone can move quickly in the beginning and decelerate quickly as it approaches its target.

Table 1 lists the coefficients we use for each PID. For our system, it is imperative that we avoid overshooting the target, because if the drone moves on top of the wrong reference point, all future flight will be wrong. We use medium or large proportional coefficient, small integral coefficient to eliminate steady state error, and medium derivative coefficient to reduce overshoot. We use larger values for the proportional coefficient of Z and yaw because movement in those directions tend are slower (due to their motor power). We also use smaller derivative terms for Z and yaw because their movement is slower and their sensors are more accurate and thus less prone to overshoot.

4. IMPLEMENTATION

DroneSense is designed to provide an easy-to-use, fast, and flexible way of automatically measuring 3D wireless signals. It is written in JavaScript on the Node.js platform. It is divided into two

modules: navigation and measurement. The navigation module is responsible for moving the drone to the desired locations, estimating its position at all times, and correcting errors in its trajectory. The measurement module is responsible for collecting measurements at each desired location, communicating with the drone to tell it when and how to measure signal. It communicates to the drone's operating system via telnet. It waits for the navigation module to broadcast that the drone has reached a target location, collects the measurement, and informs the navigation module to start moving the drone again.

4.1 Navigation Module

The navigation module is responsible for guiding the drone to the desired locations. It parses the provided flight path for coordinates to visit, and creates a queue of destinations to visit. Once the system is ready, it sends the takeoff signal to the drone. It processes information from the drone such as readings for velocity, height, and marker detection, and sends control commands to the drone. When it determines the drone has reached the target, it tells the measurement module to begin measuring. The navigation module uses a Reference Point Detection and Extended Kalman Filter to estimate the coordinates of the drone, and uses a Proportional-Integral-Derivative Controller to stabilize the flight.

The navigation module takes as an input the user's desired flight plan. The flight plan can be a text file where each line is a 3-dimensional coordinate in meters, and the yaw of the drone is assumed to be 0°. We use a coordinate system with the drone's takeoff position as the origin, initially with yaw of 0°. For Z coordinate simplicity, the user specifies a base level height and between-layer height in meters.

4.2 Measurement Module

The measurement module is responsible for communicating with the drone and collecting wireless signal measurements. When it receives the signal that the drone has reached a target location to collect measurement, it sends a message to the drone's operating system to begin measuring. It receives the response, parses it, and saves it to a text file. We use the drone's on-board wireless radio to collect measurements and work within the constraints of that system.

AR Drone 2.0 Wireless System: The Parrot AR Drone 2.0 uses the Atheros AR6000 mobile 802.11bg chipset for wireless networking. Its computer runs the BusyBox operating system, and has about 10 megabytes of free space. Because of the system constraints, we could not host the control software directly on the drone, but instead must rely on communicating with the drone from the computer program. Directly communicating with the drone requires connecting to the same network as the drone and then using telnet. The drone's wireless driver lacks some important functionality such as monitor mode and accurate noise readings. By default, the drone creates its own access point that computer programs can connect to. However, the drone cannot scan other access points when it is using its own access point. Thus, both the drone and computer must connect to the network we would like to measure, communicate over that network, and collect signals only from its current network.

Measurement Script: We leverage the drone's on-board wireless radio to collect wireless signal measurements. We use telnet to access the drone's command line interface, and place shell scripts for measuring signal on the drone's operating system. The scripts take as parameters the number of times to collect, and the amount of time to wait between collecting. These scripts use *iwconfig* to measure the current network's signal strength. *iwconfig* uses driver meta information to interpret the raw value given by */proc/net/wireless*, displaying the result in units of dBm. We use *iwconfig* because it can be executed quickly in rapid succession and its output is easy to parse.

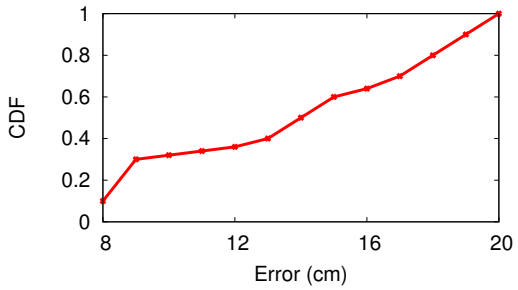


Figure 3: Navigation Error CDF. The mean error is 13 centimeters, and the max error is 20 centimeters.

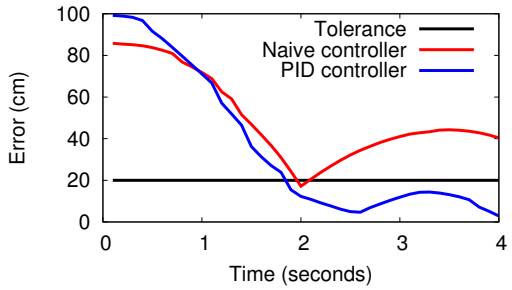


Figure 4: Comparing a naive controller with the PID controller by flying the drone to a target location from 1 meter away. The PID controller successfully reaches the target faster while the naive controller suffers from overshooting.

Measurement Program: The Measurement Program running on the computer communicates with the drone and the navigation module to collect measurements. It allows the user to specify options such as number of collections at each point and amount of time to wait between collecting, as well as where to save the log file. It listens to messages from the navigation module to see when the drone is in the right position. It creates a telnet connection with the drone via the *node-telnet-client* library [25], and then runs one of the measurement scripts we previously placed on the drone. We use a single continuous telnet connection with the drone and error catching mechanisms to restart the connection if it is broken. The program listens to the navigation module to find out when to measure, gets the current coordinates, runs the appropriate measurement script, and writes the result to file.

5. EVALUATION

We evaluate the accuracy and efficiency of the drone's navigation, as well as, the accuracy and speed of the drone's measurements. These evaluations can establish the effectiveness of our solution, and to identify areas of improvement for future work.

5.1 Navigation Accuracy

Experimental Setup: Our test setup involves a 3×2 grid of reference points one meter apart from each other. A string and weight is attached to the bottom of the drone, and while the drone is hovering over a collection location, we mark the projection of the location of

Table 2: Comparing RSS mean and standard deviation for different measurement time durations. We conclude that 3 seconds is a sufficient time length for the drone to collect measurements.

Location ID	Duration (second)	RSS Mean (dBm)	RSS Standard Deviation (dB)
a	1	-73.00	0.8165
	3	-74.33	1.18
	10	-74.35	2.34
	30	-73.78	2.15
b	1	-52.00	1.49
	3	-46.9	1.42
	10	-46.64	3.64
	30	-47.50	3.50

the drone on the ground indicated by the string. Afterwards, we measure the distance between the marked locations and the center of the reference point. The distance between the two points is the navigation error. In the software, we set the accuracy tolerance of the drone to 20 centimeters, which is an acceptable error tolerance and does not cause the drone to spend too long to adjust to reach the target position. We run this experiment 10 times, and find the mean and standard deviation of the error over these 50 data points, and plot the cumulative density function of the results.

Results: The mean error we found is 13 centimeters, and the standard deviation is 4.3 centimeters. Additionally, the error is never more than 20 centimeters, which is important because that is the error tolerance we set in the software. This shows that the drone's navigation is accurate to its specifications. It also means that stricter navigation accuracy standards can be applied, at the expense of navigation time, if greater precision is needed. Figure 3 shows the CDF of the error values.

5.2 Flight Efficiency

Experimental Setup: To evaluate flight efficiency, we have the drone takeoff from one reference point and travel to another reference point 1 meter away, and hover there before landing. We will now work with the drone's estimated coordinates, because it is possible to get multiple position estimates every second. We plot the distance between the drone and the target as estimated by the drone during this flight. We compare the plot of the error when the system uses the current PID Controller implementation, with the plot of the error when the system uses a naive controller, as we mentioned in the flight efficiency challenge in section 2. It sets the magnitude proportional to the error and then rounds the value to the nearest 0.05.

Results: Figure 4 shows the error plots for the naive controller and PID Controller. The point where the drone determines it has reached its target is marked. We see that the PID controller not only helped the drone reach the target faster, but stopped the drone successfully once the target was reached. For the inefficient controller, overshoot is a problem because after the target is reached, the drone does not quickly stop itself, overshoots, and has to adjust back to the target. If all controls commands are decreased in magnitude, then the overshoot problem is reduced, but then it will take much longer to reach the target. Therefore, the PID Controller is a superior controller, and significantly increases the efficiency of the flight.

5.3 Measurement Duration

Experimental Setup: In order to determine an adequate amount of time to collect measurements so that the effect of noise is negligible, we test collecting measurements at two locations for various amounts of time. These two locations have significantly different signal strength, and we test collecting measurements for 1, 3, 10, and 30 seconds. Then we calculate the mean and standard deviation of signal strength collected at each location for each duration and see at which duration the results start to converge.

Results: In Table 2, we list the mean and standard deviation of measurements with different duration at multiple locations. When the drone is collecting measurements upon a location, the drone shakes slightly due to the airflow disturbance, which leads to received wireless signal strength variation. Therefore we need a period of time is to average measurements and counteract signal noise. In Table 2, we found that a minimal duration of three seconds is sufficient for the drone to gain relatively stable measurements that compensate signal fluctuations at each location.

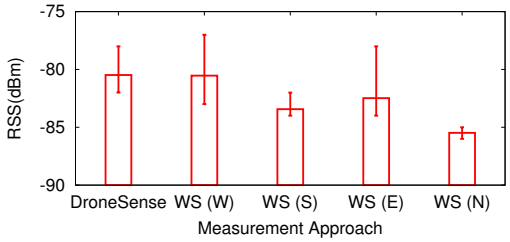


Figure 5: Comparing the signal measurements of DroneSense and the walking sense (WS) approach at a given measurement location. By involving human interactions in the measurement collection, the measurements from WS are affected by the human body.

5.4 Measurement Accuracy

We study two experiment settings to examine DroneSense's accuracy in comparison to the war walking method, and the variation of the measurement results gathered by DroneSense across multiple flight trials. In the first setting, we present that DroneSense improves the accuracy by removing signal variation caused by people interference. The AR Drone 2.0 used in experiments are mostly composed of plastic instead of metal, thus we believe the drone itself is unlikely to alter the signal strength. Then we use DroneSense to collect signal strength along a flight path for multiple times and measurements are quite stable even though during each flight the actual flight path may differ a little due to the navigation error.

5.4.1 DroneSense VS. War Walking

Experimental Setup: Prior studies [13, 30] have shown that the presence of people nearby the receiver interferes with the wireless signal strength, which is the drawback of walk sensing method. To compare our automatic wireless measurement approach with walk sensing (WS) method, we instruct the drone to hover upon a location at 1 meter height for 6 seconds to collect measurements. Then we hold the drone at the exact location for 6 seconds to collect wireless signal strength data, which is repeated for 4 times. Each time the we stand at different orientations relative to the drone, i.e., west, south, east and north.

Results: Figure 5 shows the means and error bars of wireless signal strength measured in above experimental settings. The average RSS is -80.48 dBm when using DroneSense, however, the average RSS fluctuates a lot when we hold the drone toward different orientations while the drone remains upon the location at 1 meter height. It is -80.53 dBm when we stand on the west side of the drone and

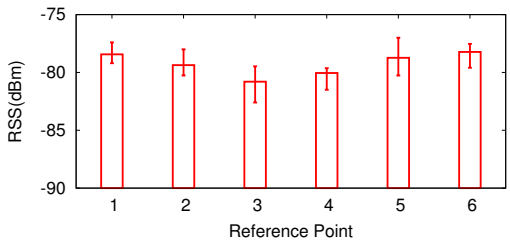


Figure 6: Repeating wireless measurements by DroneSense along a flight path of 6 reference points for 10 times. The measurement variation for a location is around 2dB and no more than 3dB. We conclude that DroneSense is accurate enough for wireless measurement.

-85.48 dBm when at north. According to our observation, when we stand on the north side of the drone, we happen to block some signals by being in a direct path from the transmitter to the drone. The result demonstrates that DroneSense allows collecting more accurate signal measurement by removing the artifacts caused by human body.

5.4.2 Accuracy across Flight Trials

Experimental Setup: Since there has always been a few centimeters navigation error when the drone is hovering upon a location of interest, wireless measurement varies due to drone's movements under different experiment trials. To demonstrate DroneSense is accurate and efficient in wireless measurements, we place 6 reference points with 1 meter interval on the ground, and customize a measurement script to navigate the drone fly over the 6 reference points one by one. When the drone arrives at one reference point, the drone hovers for 3 seconds to collect wireless signal strength data, and calculate the average value as the measurement of the point, then fly to the next point according to the measurement script. We repeat this measurement process for 10 times.

Results: Figure 6 shows the means and error bars of wireless signal strength measured in above experimental settings. The measurement variation of one reference point for 10 times is very small, which is around 2dB and no more than 3dB, which proves for wireless measurement, DroneSense is stable and accurate enough.

6. RELATED WORK

Wireless Measurements: War driving and war walking are common ways to collect Wi-Fi signal data [5, 14, 26]. War driving is to collect data by driving around while war walking is by walkers rather than drivers, both of which are labor-intensive. As the off-the-shelf mobile devices become ubiquitous in recent years, there has been a growing interest in a inspiring method called participatory sensing or crowdsourcing [18, 22, 24, 29]. Though crowdsourcing is in essence war walking, it leverages ubiquitous smartphones to passively collect wireless data and sensor data. Combining inertial sensors (e.g., accelerometer, compass, gyroscope) with dead reckoning, wireless data is automatically labeled with location information, thus no further user participation is required [24].Besides smartphones and laptops, there are also many commercial products that provide all-in-one solutions to facilitate the indoor wireless measurement process, such as the Agilent E6474A [1] and the Nemo Walker Air [20]. They are used to analyze network coverage and evaluate network performance while the user walks around.

Propagation Modeling: A rich set of wireless signal propagation models have been well studied [3, 9, 12, 15, 16, 21, 27, 28]. Studies on propagation modeling require actual signal collection for validation. The receiver is placed manually at a number of locations to perform measurements in previous studies. Thus, research on wireless propagation modeling and our work go hand in hand. For instance, a study based on propagation modeling, WiPrint, tries to shape the wireless coverage in the room [8]. One limitation of propagation models is that they are not fully accurate, which is why our system is complementary to these efforts.

Drone Control and Navigation: There have been extensive efforts in developing reliable drone control and navigation systems [2, 6, 7, 10, 11, 19]. Extended Kalman Filter is also adopted to find the position of a quadcopter by combining GPS and dead reckoning [2]. There have been studies on visual navigation of drones [6, 10, 11]. A vision based drone navigation system leverages a downward looking monocular camera and SLAM algorithm to track the pose of the drone, where a more advanced LQG/LTR based controller is introduced to handle discontinuities of the SLAM pose estimate [6]. In our work, the standard PID controller is sufficient to guarantee the flight efficiency. These methods rely on visual tracking of objects to determine the drone's position and guide the drone to its target. However, these methods require there exist distinct visual features in the images captured by the camera. It suffers from localization error when there is no distinct visual clue, such as a white wall or floor.

7. CONCLUSION AND FUTURE WORK

This paper presents DroneSense, a system leveraging drones to automate the collection of 3D wireless signal measurements. Our system is able to reduce the amount of effort needed to collect measurements, and can be easily adapted to a variety of environments. To achieve this, we address challenges of accurate drone navigation and efficient flight trajectory, using methods of Reference Point Detection, Extended Kalman Filter, and Proportional-Integral-Derivative Controller. Our system uses the Parrot AR Drone 2.0 for its hardware, and JavaScript on the Node.js platform for its software. We summarize the limitations of our current system and plans of future work as follows.

Navigation Accuracy: The current system works well for collecting measurements at whole-meter intervals, with accurate navigation and efficient flight with an error tolerance of 20 centimeters. We plan to improve the accuracy of the system further to allow for sub-meter intervals. This can be achieved through improving the drone's position estimation system, upgrading the model of the drone to one with more accurate velocity readings, setting better parameters for the PID controller. This will help achieve greater resolutions in measurement collection.

Extending to Other Frequency Bands: The current system provides a way to measure Wi-Fi signal strength because of the availability of on-board Wi-Fi radio on the drone. We plan to adapt the system to collect wireless signals in other frequency bands (e.g., visible light). To measure other frequency bands, we can attach external radio transceivers to the drone if the drone's on-board radio cannot measure the frequency band of interests.

Reducing or Eliminating Reference Points: One major limitation of the current system is the need for placing reference points throughout the environment. We will explore methods to reduce or eliminate the need for reference points while maintaining accurate position estimates. One option is placing cameras in the room and using them to determine the drone's location at all times. We will

also study other methods for the drone to locate itself.

Handling Obstacles or Moving Users: The current system relies on the user to input a flight path that does not collide with any obstacles or moving users. Static obstacles cause the drone's self-adjustment of the height, as the drone considers the object as the reference point for the ground. Moving users not only interfere with wireless signals, but also disturb the airflow near the drone, which can make the drone deviate from the planned flight path. Thus, DroneSense is currently more suitable to controlled experimental settings. We plan to develop mechanisms for the drone to maintain certain distances from obstacles and users. Such mechanisms can eliminate the need for the user to input a flight path and allow the drone to continuously collect signal measurements as it flies in an environment.

Multiple Drones: While our current design and experiments consider a single drone, we will extend the functionality of our system to work with multiple drones and speed up the measurement collection process. We will start with running the current program simultaneously in multiple separate processes and instrumenting the drones to cover different parts of the room. We will then further optimize each drone's flight path and coordinate their measurement collection to shorten overall measurement duration.

8. REFERENCES

[1] Agilent indoor wireless measurement system. http://literature.cdn.keysight.com/litweb/pdf/5968-8691E.pdf. Accessed August 6, 2016.

[2] AHAMED HAMZA, M. A., KEPPITIYAGAMA, C., DE ZOYSA, K., IYER, V., HEWAGE, K., AND VOIGT, T. A quadcopter controller to maintain radio link quality. In *Proc. of DroNet* (2015), ACM, pp. 21–26.

[3] ANDERSEN, J. B., RAPPAPORT, T. S., AND YOSHIDA, S. Propagation measurements and models for wireless communications channels. *Communications Magazine, IEEE 33*, 1 (1995), 42–49.

[4] AR Drone camera backproject. https://github.com/tum-vision/ardrone_autonomy/blob/master/calibrations. Accessed August 6, 2016.

[5] BAHL, P., AND PADMANABHAN, V. N. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2000), vol. 2, Ieee, pp. 775–784.

[6] BLÖSCH, M., WEISS, S., SCARAMUZZA, D., AND SIEGWART, R. Vision based mav navigation in unknown and unstructured environments. In *Robotics and automation (ICRA), 2010 IEEE international conference on* (2010), IEEE, pp. 21–28.

[7] BREGU, E., CASAMASSIMA, N., CANTONI, D., MOTTOLA, L., AND WHITEHOUSE, K. Reactive control of autonomous drones. In *Proc. of MobiSys* (2016).

[8] CHAN, J., ZHENG, C., AND ZHOU, X. 3d printing your wireless coverage. In *Proceedings of the 2nd International Workshop on Hot Topics in Wireless* (2015), ACM, pp. 1–5.

[9] DURGIN, G., PATWARI, N., AND RAPPAPORT, T. S. An advanced 3d ray launching method for wireless propagation prediction. In *Vehicular Technology Conference, 1997, IEEE 47th* (1997), vol. 2, IEEE, pp. 785–789.

[10] ENGEL, J., STURM, J., AND CREMERS, D. Camera-based navigation of a low-cost quadrocopter. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on* (2012), IEEE, pp. 2815–2821.

[11] ENGEL, J., STURM, J., AND CREMERS, D. Scale-aware navigation of a low-cost quadrocopter with a monocular camera. *Robotics and Autonomous Systems 62*, 11 (2014), 1646–1656.

[12] GHOSE, A., BHAUMIK, C., AND CHAKRAVARTY, T. Blueeye: A system for proximity detection using bluetooth on mobile phones.

[13] HEREDIA, B., OCAŃA, M., BERGASA, L. M., SOTELO, M., REVENGA, P., FLORES, R., BAREA, R., AND LÓPEZ, E. People location system based on wifi signal measure. In *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on* (2007), IEEE, pp. 1–6.

[14] HUANG, J., MILLMAN, D., QUIGLEY, M., STAVENS, D., THRUN, S., AND AGGARWAL, A. Efficient, generalized indoor wifi graphslam. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (2011), IEEE, pp. 1038–1043.

[15] ISKANDER, M. F., AND YUN, Z. Propagation prediction models for wireless communication systems. *Microwave Theory and Techniques, IEEE Transactions on 50*, 3 (2002), 662–673.

[16] JI, Y., BIAZ, S., PANDEY, S., AND AGRAWAL, P. ARIADNE: a dynamic indoor signal map construction and localization system. In *Proc. of MobiSys* (2006).

[17] KALMAN, R. E. A new approach to linear filtering and prediction problems. *Journal of basic Engineering 82*, 1 (1960), 35–45.

[18] KIM, Y., CHON, Y., AND CHA, H. Smartphone-based collaborative and autonomous radio fingerprinting. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42*, 1 (2012), 112–122.

[19] LUPASHIN, S., HEHN, M., MUELLER, M. W., SCHOELLIG, A. P., SHERBACK, M., AND DÂĂŹANDREA, R. A platform for aerial robotics research and demonstration: The flying machine arena. *Mechatronics 24*, 1 (2014), 41–54.

[20] Nemo walker air indoor benchmarking wireless networks. http://www.anite.com/sites/default/files/Nemo%20Walker%20Air%20Brochure%20Mar2016.pdf. Accessed August 6, 2016.

[21] PANJWANI, M. A., ABBOTT, A. L., AND RAPPAPORT, T. S. Interactive computation of coverage regions for wireless communication in multifloored indoor environments. *Selected Areas in Communications, IEEE Journal on 14*, 3 (1996), 420–430.

[22] PARK, J.-G., CHARROW, B., CURTIS, D., BATTAT, J., MINKOV, E., HICKS, J., TELLER, S., AND LEDLIE, J. Growing an organic indoor location system. In *Proceedings of the 8th international conference on Mobile systems, applications, and services* (2010), ACM, pp. 271–284.

[23] PARROT. Parrot AR Drone 2.0. http://ardrone2.parrot.com. Accessed August 6, 2016.

[24] RAI, A., CHINTALAPUDI, K. K., PADMANABHAN, V. N., AND SEN, R. Zee: zero-effort crowdsourcing for indoor localization. In *Proc. of MobiCom* (2012), ACM, pp. 293–304.

[25] node-telnet. https://github.com/mkozjak/node-telnet-client. Accessed August 6, 2016.

[26] TSUI, A. W. T., LIN, W.-C., CHEN, W.-J., HUANG, P., AND CHU, H.-H. Accuracy performance analysis between war driving and war walking in metropolitan wi-fi localization. *IEEE Transactions on Mobile Computing 9*, 11 (2010), 1551–1562.

[27] VALENZUELA, R. A. A ray tracing approach to predicting indoor wireless transmission. In *Proc. of VTC* (1993).

[28] VALENZUELA, R. A. A ray tracing approach to predicting indoor wireless transmission. In *Vehicular Technology Conference, 1993., 43rd IEEE* (1993), IEEE, pp. 214–218.

[29] WU, C., YANG, Z., LIU, Y., AND XI, W. Will: Wireless indoor localization without site survey. *IEEE Transactions on Parallel and Distributed Systems 24*, 4 (2013), 839–848.

[30] ZHANG, Z., ZHOU, X., ZHANG, W., ZHANG, Y., WANG, G., ZHAO, B. Y., AND ZHENG, H. I am the antenna: accurate outdoor ap location using smartphones. In *Proceedings of the 17th annual international conference on Mobile computing and networking* (2011), ACM, pp. 109–120.