Indoor Follow Me Drone

Wenguang Mao, Zaiwei Zhang, Lili Qiu, Jian He, Yuchen Cui, and Sangki Yun[†]
The University of Texas at Austin, [†] Hewlett Packard Labs
{wmao,zaiwei92,lili,jianhe,yuchen93}@cs.utexas.edu, [†] sangki.yun@hpe.com

ABSTRACT

Professional video taping is a costly and time consuming process. With the availability of inexpensive and powerful drones, it is possible to let drones automatically follow a user for video taping. This can not only reduce cost, but also support video taping in situations where otherwise not possible (*e.g.*, during private moments or at inconvenient locations like indoor rock climbing). While there have been many follow-me drones on the market for outdoors, which rely on GPS, enabling indoor follow-me function is more challenging due to the lack of an effective approach to track users in indoor environments.

To this end, we develop a holistic system that lets a mobile phone carried by a user accurately track the drone's relative location and control it to maintain a specified distance and orientation for automatic video taping. We develop a series of techniques to (i) track a drone's location using acoustic signals with sub-centimeter errors even under strong propeller noise from the drone and complicated multipath in indoor environments, and (ii) solve practical challenges in applying model predictive control (MPC) framework to control the drone. The latter consists of developing measurement-based flight models, designing measurement techniques to provide feedback to the controller, and predicting the user's movement. We implement our system on AR Drone 2.0 and Samsung S7. The extensive evaluation shows that our drone can follow a user effectively and maintain a specified following distance and orientation within 2-3 cm and 1-3 degree errors, respectively. The videos taped by the drone during flight are smooth according to the jerk metric.

Keywords

Drone; Tracking; Acoustic signals; FMCW; MUSIC; MPC

1. INTRODUCTION

Motivation: The global drone industry has grown rapidly in the past few years. It reached \$8 billion in 2015, and is expected to reach \$12 billion by 2021 [14]. It has a long list of applications in agriculture, energy, news media, and film production. In particular, one attractive application of drones is to follow a subject for autonomous video taping. This is motivated by the fact that professional video taping has been a costly and time-consuming process, and sometimes not even possible, such as during private moments or at dangerous locations. The wide availability of powerful and inexpensive drones makes it possible for low-cost autonomous video taping via drones.

Inspired by this vision, lots of commercial products are becoming available on the market. [18] lists top 10 drones with the follow-me mode, which are designed for the outdoor usage. They rely on GPS to follow the subject. Their typical follow-me distances are from tens of meters to a few kilometers [19]. In this case, meter-level variation on the following distance has negligible impact on the video quality. However, in indoor environments, the follow-me distance is only up to a few meters, and meter- or even decimeter-level distance variation can significantly degrade the video quality. Therefore realizing the follow-me mode indoors poses several new challenges: (i) How to accurately and efficiently track the user indoors without GPS signals. This requires accurate distance and orientation estimation. (ii) How to control the drone to accurately maintain a specified following distance and orientation from the user for high-quality video taping.

Tracking: When GPS is not available, computer vision (*CV*) is commonly used for a drone to track a user [21, 30, 35, 43, 8, 22, 36]. However, the accuracy of current CV-based tracking is usually limited because the vibration of a drone during flight may blur the captured images and degrade tracking accuracy. CV-based tracking also degrades in a busy background [8, 22]. As a result, the error of follow-me distance for CV-based approaches is decimeter- to meter-level [35, 43, 8, 36], which is insufficient for our purpose. Moreover, CV-based methods tend to be too expensive for mobile devices [25].

In addition to CV-based methods, there are other tracking techniques, including RF-based localization (*e.g.*, [54, 56, 5, 4]), acoustic-based tracking (*e.g.*, [37, 61, 39, 62, 55]), and coarse movement tracking using depth sensors and infrared cameras (*e.g.*, [1, 2, 33]). We find that acoustic based tracking is attractive because (i) its slow propagation speed makes it possible to achieve high tracking accuracy, (ii) it can be generated and received using widely available speakers and microphones on mobile devices, and (iii) it can be efficiently processed in software without any extra hardware. Therefore, we mount multiple speakers on a drone to send special acoustic signals to a mobile device (*e.g.*, a smartphone or a smart watch) carried by a user. The mobile device processes the received signals to track the drone's position in real-time.

While there have been several accurate acoustic-based motion tracking systems proposed recently, tracking in the context of a drone poses several new challenges. First, there can be many objects in indoor environments, which results in significant multipath

MobiSys '17, June 19–23, 2017, Niagara Falls, NY, USA.
© 2017 ACM. ISBN 978-1-4503-4928-4/17/06. . . \$15.00
DOI: http://dx.doi.org/10.1145/3081333.3081362

between the drone and mobile device. Moreover, these paths are all changing since both the drone and user are moving. Therefore the acoustic-based tracking schemes developed in [39] and [55] are not applicable because they assume the tracking target is the only moving object in the environment and all other paths are static. Second, some paths may have similar lengths to that of the direct path, which causes signals traversing these paths to interfere and leads to large distance estimation error. This issue degrades [37], a state-of-the-art acoustic-based tracking. Therefore, we need to develop a motion tracking that has high resolution to separate the paths with similar lengths. Third, when a drone is flying, the propellers on a drone generate large noise over a wide spectrum, which may interfere with the acoustic signals used for tracking. The existing acoustic-based tracking approaches use different frequency from the background noise and do not need to explicitly handle noise. Fourth, tracking must be performed efficiently in real-time on a mobile device.

Motivated by these challenges, we develop a **R**obust **A**coustic-**B**ased **I**ndoor **T**racking system, called Rabbit. It is based on distributed Frequency Modulated Continuous Wave (FMCW) [37] for estimating the distance between the drone and mobile. Furthermore, it applies MUltiple SIgnal Classification (MUSIC) during FMCW detection to significantly improve the capability of resolving multiple paths and enhance distance estimation. This is very different from the traditional use of MUSIC for deriving the Angle-of-Arrival (AoA) using multiple antennas [51, 58, 32], which is not feasible in our setup due to the lack of a microphone array.

Recently, a few work [60, 3, 28] explore applying MUSIC in frequency domain for distance estimation. Compared to these work, our approach enables accurate and efficient acoustic tracking under strong propeller noise and complicated multipath due to four unique designs. First, since MUSIC is a model-based parameter estimation method, which is sensitive to distortion, we develop an effective method to calibrate and compensate for the speaker distortion in acoustic signals. Second, multipath causes multiple frequency peaks during FMCW detection and strong noise from a drone may generate false peaks that do not correspond to any real propagation path. We develop an effective mechanism to select the correct frequency peak corresponding to the direct path for distance estimation. Third, MUSIC needs eigenvalue decomposition, which is expensive, so we develop an effective subsampling to improve the efficiency without compromising accuracy. Finally, we fuse the distance and velocity estimation using Kalman filter to improve the accuracy under propeller noise. While this paper uses Rabbit to track a drone, its robustness and efficiency make it suitable for *other* applications (*e.g.*, gaming, augmented reality/virtual reality, and remote control for IoT devices).

Control design: To address the second challenge, we apply control theory. Despite a variety of control algorithms available, there are several important practical issues. (i) Which control algorithm is appropriate to accurately control low-end drones? (ii) How can we capture the unique dynamics of our drone to maximize effectiveness? (iii) How to measure the deviation from the desired distance and orientation, which will be used as feedback to the controller ? (iv) How to predict a user's movement and leverage this information to enhance the control performance?

Motivated by these questions, we conduct systematic measurement between the control inputs to the drone (*e.g.*, pitch, yaw, and roll) and its actual movement. We find that pitch and yaw have predictable impact on the forward and rotation movement, whereas the relationship between the roll and lateral movement has high variability in the range of interest. We develop measurement-based models for the first two dimensions by applying aerodynamics principles and compensating nonlinear system behaviors, and using model predictive control (MPC) to control movement along these dimensions. We apply Proportional Integral Derivative (PID), which does not require a model, to control the lateral movement, since MPC does not work well when the model accuracy is low. Moreover, our acoustic-based tracking is applied to measure the distances from the mobile to multiple speakers mounted on the drone. Based on the measurements, we use geometry methods to derive the distance and orientation between the user and drone, which serve as feedback to our controller. Finally, based on the Doppler shift of acoustic signals, we develop a simple yet effective way to predict a user's movement so that the controller can respond in advance to optimize the follow-me performance. This paper is a pioneering work that realizes an indoor follow-me drone based on MPC framework. Our procedure and insights are valuable for realizing other important drone functionalities in the future.

System: We develop a follow-me drone for indoor use, called DroneTrack, on top of Rabbit. As shown in Figure 1, Rabbit processes the received signals to estimate the distances between the mobile and speakers on the drone. The observer derives the relative location of the drone with respect to the user, and the predictor estimates the user's movement in the near future. The controller computes the command to move the drone in an appropriate direction. We implement a prototype on AR drone 2.0 and Samsung Galaxy S7, and evaluate it in real indoor environments by either holding the mobile in hand or putting it in a bag. The evaluation shows our system measures the distance between the drone and user with a median error within 1 cm, and maintains the specified following distance and orientation within 2-3 cm and 1-3 degree errors, respectively. Also, our system is robust to various interference including propeller noise, human voices, and music. The processing delay for the tracking part and the whole system is 9 ms and 13 ms, respectively, and the corresponding CPU usage is 13% and 42%, respectively. Moreover, the videos taped by the drone during flight are stable according to the jerk metric [16].

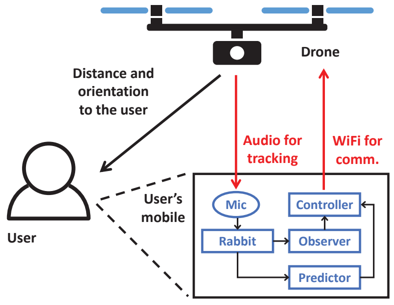


Figure 1: System diagram.

Summary: Our main contributions include (i) developing an accurate, robust, and efficient acoustic-based tracking system on mobile devices (Section 2), (ii) a holistic design of a follow-me drone based on the MPC framework, which includes developing measurement based models, continuously estimating deviation from the specified distance and orientation, and predicting the user's movement (Section 3), and (iii) a prototype system and extensive evaluation that demonstrate the effectiveness of our approaches (Section 4 and Section 5).

2. ROBUST ACOUSTIC TRACKING

In this section, we first review existing approaches for estimating velocity and distance, and identify the limitations for distance estimation. Then we present Rabbit – our approach to enhance robustness of tracking by (i) developing an efficient and robust distance estimation approach based on FMCW and MUSIC, and (ii) fusing distance and velocity measurements using Kalman filter to further improve the accuracy. These techniques are not only useful under environments with significant multipath, but also help tolerate strong noise from a drone.

2.1 Overview of Existing Approaches

Estimating velocity based on Doppler shift: The Doppler shift is a well known phenomenon where the frequency changes with the relative velocity between the sender and receiver as $F^s{=}(v/c)F$, where F is the original frequency, F^s is the amount of frequency shift, v is the receiver's speed towards the sender, and c is the speed of sound waves. Therefore, we can estimate the receiver's velocity by measuring the amount of frequency shift.

According to [61], we use the following steps to estimate the relative velocity between the drone and mobile : (i) The speaker on the drone continuously transmits sine waves at frequency F. (ii) The mobile applies Fast Fourier Transform (FFT) to extract the spectrum of the received signals. We use 1764 acoustic samples in a period of 40 ms to compute FFT. Then, we find a peak frequency and compare it with F. The difference between the two is the frequency shift F^s. (iii) We compute the velocity based on the estimated frequency shift using $v{=}(F^s/F)c$.

Estimating distance based on distributed FMCW: FMCW is an effective way to estimate the distance traveled by acoustic signals. Compared to phase-based approaches [55], FMCW is more robust to multipath. Compared to pulse-based methods [62], FMCW achieves higher accuracy using the same amount of bandwidth.

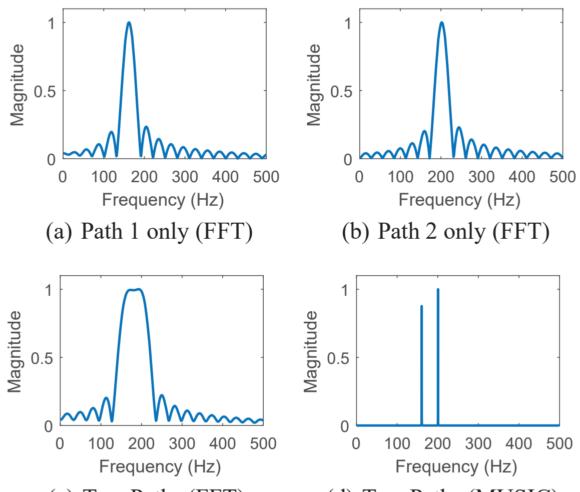
We first review FMCW for a synchronized sender and receiver, and then describe how to support an unsynchronized sender and receiver. In FMCW, a speaker transmits periodic chirp signals, whose frequency linearly increases over time as $f{=}f_{min}{+}\frac{Bt}{T}$ in each period, where B is the signal bandwidth and T is the duration of a period. Thus, the transmitted signal in the k-th period is given by $v_t{=}\cos(2\pi f_{min}t'{+}\frac{\pi B t'^2}{T})$, where $t'{=}t{-}kT$, and the received signal is $v_r{=}\alpha\cos(2\pi f_{min}(t'{-}t_d){+}\frac{\pi B(t'-t_d)^2}{T})$, where α is the attenuation factor and t_d is the propagation delay. The receiver mixes (*i.e.*, multiplies) the received signal with the transmitted signal. After passing a low-pass filter, the mixed signal is given by

$$v_m(t){=}\alpha\cos(2\pi f_{min}t_d{+}\frac{\pi B(2t't_d{-}t_d^2)}{T}). \qquad (1)$$

Then we can derive the propagation delay t_d by finding the peak frequency in the spectrum of the mixed signal (called *FMCW peak frequency*), and estimate the distance based on $c{\cdot}t_d$. Traditional FMCW works for a co-located sender and receiver with a shared clock. To support a separate and unsynchronized sender and receiver as in our context, we apply a distributed FMCW approach developed in [37]. It first estimates the initial distance and then determines the current distance based on the distance change from the initial position as follow: $R_k{=}(f_k^p{-}\frac{f_{min}v_k}{c}{-}\frac{Bv_k}{c}{-}f_1^p)\frac{cT}{B}{+}R_1$, where R_k is the distance between the sender and receiver at the k-th period, f_1^p and f_k^p are FMCW peak frequencies in the first and k-th periods, v_k is the velocity measured by Doppler shift, and R_1 is the initial distance.

2.2 Limitations of Existing FMCW

Traditionally, Fast Fourier Transform (FFT) is used to detect the FMCW peak frequency, which is then used to estimate the distance, as discussed above. However, multipath can result in multiple peaks in the spectrum of mixed signals. In this case, we need to locate the first peak, which corresponds to the direct path. When there exist some paths close to the direct path, the peaks for these paths may merge together due to limited frequency resolution of FFT-based FMCW [31]. This will cause erroneous peak detection.



(a) Path 1 only (FFT) (b) Path 2 only (FFT)
(c) Two Paths (FFT) (d) Two Paths (MUSIC)

Figure 2: Peak merging in FFT-based FMCW.

Figure 2 illustrates an example. We generate two paths in simulation, where the first and second paths are 0.8m and 1m long, respectively. The two paths have the same magnitude, and their phase difference is π. There is no noise in the received signals. All other settings are the same as those we used for experiments. When the signals from the two paths naturally add up at the receiver, we can only detect one peak using FFT-based FMCW as shown in Figure 2(c). The peak occurs at 194 Hz, corresponding to a distance estimation of 0.96m, which gives an estimation error of 16 cm. When there is ambient noise (*e.g.*, from drone's propellers), the error of FFT-based FMCW detection can be even larger.

The limited resolution of FMCW comes from the Fourier transform [31]. When applying the transform to time-domain signals with finite duration, the frequency-domain representation of the signals will spread out. This degrades the ability to differentiate signals at different frequencies. MUltiple SIgnal Classification (MUSIC) [48] is a potential solution for this problem. It is a super-resolution parameter estimation algorithm and widely used in radar and localization applications for determining angle-of-arrival (AoA). According to Equation 1, the mixed signals can be expressed as a sum of cosines when multiple paths are present. This model is the same as that used for AoA detection [48]. Thus, we can apply MUSIC to determine the frequency components in the mixed signals to enhance the ability of resolving multipath and improve the distance estimation with FMCW. In our previous example, if we apply MUSIC to derive a pseudo-spectrum of the mixed signals, we can clearly see the two peaks corresponding to the two paths, as shown in Figure 2(d).

2.3 Enhancing Robustness of FMCW

The simulation results in the previous section shows the promise of MUSIC. However, there are several challenges in applying MUSIC to our context. First, in real environments, the signals suffer from various distortions, which make them deviate from the ideal model required by MUSIC (*i.e.*, a sum of cosines). These distortions significantly degrade the resolution of the algorithm. Second, the distortion and strong noise from the drone may cause the peak frequencies derived by MUSIC not to correspond to a real propagation path. We need to carefully filter out these false peaks and select the true peak corresponding to the direct path between the sender and receiver. Third, MUSIC requires eigenvalue decomposition, which is computationally expensive. To run MUSIC on smartphones, we need to significantly improve its efficiency. Below, we first review MUSIC, and then present our approaches to enhance the robustness and efficiency of MUSIC.

Basics of MUSIC: Based on Equation 1, the mixed signals under multipath propagation becomes

$$v_n = \sum_{i=1}^{M_p} \cos(2\pi f_i n t_s + \phi_i), \tag{2}$$

where M_p is the number of paths, n is the sample index, t_s is the sample interval, and t' in Equation 1 is replaced by nt_s. f_i is the frequency for the i-th cosine wave, and it is proportional to the propagation delay of the i-th path. ϕ_i is the phase in Equation 1. This expression follows the model required by MUSIC [48].

To apply MUSIC, we calculate the auto-correlation matrix R for the mixed signals v as $V^H V$, where V is the Toeplitz matrix of v with the size $(N-M+1)\times M$. M is the *order* of the auto-correlation matrix (*i.e.*, R has the size $M\times M$) and N is the number of samples in mixed signals. The i-th column of V is given by $[v_{M+1-i}, v_{M+2-i}, \ldots, v_{N+1-i}]^T$. Following that, we apply eigenvalue decomposition to R, and sort the eigenvectors in a descending order in terms of the magnitude of corresponding eigenvalues. The space spanned by the first M_p eigenvectors are called *signal space*, and the space spanned by the other eigenvectors are called *noise space*. Let R_N denote the noise space matrix, whose i-th column is the i-th eigenvector for the noise space. It can be shown [48] that

$$R_N^H \cdot s(f_i) = 0, \tag{3}$$

for any f_i in Equation 2, where $s(f)$ is a *steering vector* defined as

$$[1, e^{j2\pi f t_s}, \ldots, e^{j2\pi f (M-1) t_s}]^T. \tag{4}$$

Based on this property, we can define a pseudo-spectrum of the mixed signals as

$$p(f) = \frac{1}{s(f)^H R_N R_N^H s(f)}. \tag{5}$$

We find f_i by locating peaks in the pseudo-spectrum as shown in Figure 2(d). Alternatively, we can determine f_i using a variant of MUSIC called *Root-MUSIC* [45]. Based on Equation 3, it can be shown that the complex number $e^{j2\pi f_i t_s}$ is the root for *the characteristic polynomial* $\sum_{l=-M+1}^{M-1} (\sum_{j-k=l} r_{kj}) z^l$ [45], where r_{kj} is an entry in the matrix $R_N R_N^H$. Root-MUSIC is equivalent to MUSIC except that it directly computes the solution whereas MUSIC searches for f that gives the highest $p(f)$. Therefore, Root-MUSIC is more efficient, and we use Root-MUSIC in our implementation. However, for the ease of visualization, we use MUSIC to generate the pseudo-spectrum plots to help explain our design.

Frequency response compensation: MUSIC is a model-based approach. Its performance is determined by how accurately the signals follow the model. A major source of distortion in the acoustic channel is non-flat frequency response of the speaker. Figure 3 shows the speaker gain decreases with the frequency. The non-flat frequency response is common for inaudible bands (*e.g.*, above 16 KHz), which we use for transmission, because general purpose speakers are not optimized for these bands. The non-flat frequency response distorts the transmitted chirps, which causes the mixed signals to deviate from the model in Equation 2. When applying MUSIC to these signals, the peak becomes less sharp, which degrades the resolution of MUSIC.

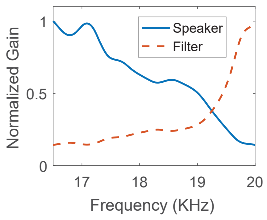
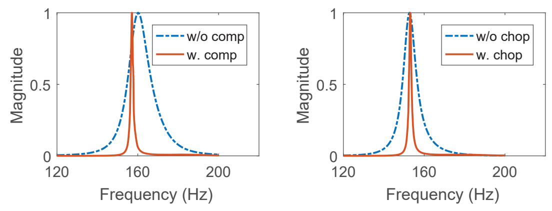


Figure 3: The frequency response of the speaker and filter.



(a) Compensation (b) Chopping

Figure 4: The sharpness of peaks: (a) applying signal chopping and comparing the performance with/without frequency response compensation; (b) applying response compensation and comparing the performance with/without signal chopping.

To minimize the impact of non-flat frequency response of the speaker, we measure the response and compensate its effect. We apply the gated frequency response measurement (*i.e.*, extracting multipath-free samples to derive the response) as [20]. The advantage of this method is that we do not need any special environment (*e.g.*, an anechoic room), and the impact of multipath can be minimized effectively. Once the speaker response is measured, we implement a digital filter whose frequency response is the reciprocal of the measured response, as shown in Figure 3. The transmitted samples need to pass this digital filter before forwarding to the speaker. As a result, the acoustic signals coming out of the speaker have a flat gain at all frequencies in the interested band. To eliminate overhead for users, the speakers sold together with the drone can be calibrated in advance before shipping and compensated chirp signals are saved in a file so that a user can simply play it as usual. From our experience, speakers of the same model experience similar distortion and do not need to calibrated individually. Figure 4(a) shows that the peaks in the pseudo-spectrum derived by MUSIC become much sharper after frequency response compensation. The peak width (the difference between frequencies where the magnitude is half of the maximum) reduces from 14 Hz to 1.2 Hz, which significantly improves the resolution.

Chopping the mixed signals: Another distortion on the mixed signals occurs at their boundary (*i.e.*, the first and last few samples) due to the following two reasons. First, the mixed signals are the product of a transmitted chirp and received chirp, but the boundary of a chirp has discontinuity in frequency. As a result, the boundary of the mixed signals suffer from distortion caused by the discontinuity. Second, in FMCW detection, a low pass filter is applied to the mixed signals before performing MUSIC, and the low pass filter distorts the samples at the beginning [26].

Based on these insights, we chop the mixed signals before applying MUSIC to minimize the impact of distortion. In our scheme, the mixed signals for each period contain 1764 samples (corresponding to 40 ms), and we discard the first 140 samples and the last 100 samples. As shown in Figure 4(b), the peak in the pseudo-spectrum becomes much sharper after chopping.

Peak frequency selection: By applying MUSIC, we can determine f_i in Equation 2. Each f_i corresponds to a propagation path

Matrix order	20	30	40	60	90	120
Time (ms)	1.6	2.9	4.5	11	22	39

Table 1: MUSIC Running time.

between the speaker and microphone, and its value is proportional to the path length. Without noise, the minimum among f_i, denoted by $f_{\min}$, corresponds to the direct path. However, this may not be true under strong noise in our context. In fact, we need to determine the signal space and noise space in the MUSIC derivation. When strong noise is present, some eigenvectors in the noise space may have larger eigenvalues than certain eigenvectors for the signal space. Because we sort the eigenvectors based on the magnitude of the corresponding eigenvalues, some eigenvectors in the noise space will be treated as eigenvectors in the signal space. In this case, when applying MUSIC, we may find some peak frequencies do not correspond to any real propagation path. It is possible that such false frequency peaks occur before the peak corresponding to the direct path. To address the issue, we leverage the following two properties to filter out false peak frequencies:

- Root magnitude: As mentioned, we use Root-MUSIC to find peak frequency f_i. Based on each root of the characteristic polynomial, we can derive a peak frequency. The root corresponding to a true peak frequency has a magnitude close to 1 [45]. Therefore, we remove roots whose magnitude is larger than b_u or smaller than b_l. Based on our experiments, we select b_u as 1.03 and b_l as 0.97.
- Temporal continuity: we track the distance between the speaker and microphone every 40 ms. During this period, the distance will not change too much. As a result, $f_{\min}$ in consecutive periods should be close to each other. Thus, we only consider peak frequencies falling into $[f^p_{\min}-\delta f, f^p_{\min}+\delta f]$, where $f^p_{\min}$ is the frequency corresponding to the direct path detected in the previous period. Based on experiments, we select δf as 25 Hz. 25 Hz corresponds to 12.46 cm movement in 40 ms (*i.e.*, over 3 m/s), which is a loose bound that can tolerate errors in the previous measurement.

Then, we select the minimum peak frequency among the remaining unfiltered frequencies.

Sub-sampling for mobile implementation: The high computation cost of MUSIC makes it challenging to implement on a mobile. As mentioned above, we need to perform eigenvalue decomposition on the auto correlation matrix R. The computation time rapidly increases with the matrix size. Table 1 shows the time of running Root-MUSIC under different M. The test is performed on Samsung Galaxy S7. The algorithm implementation is based on Android NDK to maximize the performance. We use the *eigen* function provided by OpenCV package [41] to perform eigenvalue decomposition. When M is 120, the running time for Root-MUSIC is almost equal to our tracking period (*i.e.*, 40 ms). In this case, the processing speed cannot catch up with incoming samples, and the delay will accumulate over time.

Simply choosing a small M reduces the computation time, but also degrades the resolution. When M becomes smaller, the difference between the steering vectors for a peak frequency f and its nearby frequency $f+\delta f$ is also smaller according to Equation 4. As a result, the pseudo-spectrum magnitude at f and $f+\delta f$ (*i.e.*, $p(f)$ and $p(f+\delta f)$) becomes similar based on Equation 5. In this case, the peak around f becomes less sharp, which results in degraded resolution.

To reduce the computation cost without compromising the performance, we apply sub-sampling on the mixed signals v by a factor K. The sub-sampling gives us a series of sub-sequences $v^i_K = [v_i, v_{i+K}, \cdots, v_{i+lK}]$, where $i \in [1, K]$ and $l = \lfloor (N-K)/K \rfloor$. Then the auto-correlation matrix R_{sub} is estimated by $R_{\text{sub}} = \frac{1}{K}\sum_i V_i^H V_i$, where V_i is the Toeplitz matrix of v^i_K. Following that we apply the remaining steps of MUSIC to R_{sub}. It can be shown that for any f_i in Equation 2, there is a corresponding peak in the pseudo-spectrum generated based on R_{sub}. Moreover, since sub-sampling is applied, the effective sampling interval becomes Kt_s. Thus, the steering vector in this case is given by $[1, e^{j2\pi f K t_s}, \ldots, e^{j2\pi f (M_s-1) K t_s}]^T$, where M_s is the order of R_{sub}. Therefore, with sub-sampling, we can select a small M_s to reduce computation cost while achieving similar resolution as choosing $M = (M_s-1)K$ in the case without sub-sampling. Based on our experiments, we set the sub-sampling factor $K=25$ and the matrix order $M_s=25$.

Multipath resolution: MUSIC algorithm has infinite resolution when the signals perfectly follow the required model (*i.e.*, a sum of cosines) and no noise is present [63]. In this case, our approach can always separate the direct path from other paths. In practice, there are noise and distortion that make the signals deviate from the ideal model. To determine the resolution of our approach, we place a cardboard near the direct path between the speaker and microphone to create a reflected path whose length is close to the direct one. When the length difference between these two paths is 5 cm, we can still separate them in the pseudo-spectrum generated by MUSIC. Thus, the resolution of our approach is 5 cm.

2.4 Kalman Filter

To filter out noise and further improve the accuracy of the distance estimation, we combine distance and velocity measurements using a Kalman filter as follows. Let x_k denote the actual distance between the speaker and microphone in the k-th period, t denote the period duration, v_k denote the measured Doppler velocity, n_k capture the error in Doppler measurements, y_k denote the measured distance, and w_k denote the distance measurement error. These variables have the following relationship:

$$\begin{aligned} x_k &= x_{k-1} + v_k \cdot t + n_k \\ y_k &= x_k + w_k, \end{aligned}$$

Note that y_k and v_k are given by distance and velocity measurements, respectively. Large noise from the propellers degrades the accuracy of these measurements. The basic idea of Kalman filter is to exploit the redundancy between these two measurements to reduce the impact of noise. According to [17], the optimal distance estimation $\hat{x}_k$ is given by

$$\hat{x}_k = \hat{x}_{k-1} + v_k t + \frac{\hat{p}_{k-1} + q_k}{\hat{p}_{k-1} + q_k + r_k}(y_k - \hat{x}_{k-1} - v_k t),$$

where $\hat{p}_k = \frac{r_k(\hat{p}_{k-1}+q_k)}{\hat{p}_{k-1}+q_k+r_k}$. q_k and r_k are the standard deviation for n_k and w_k, respectively. Based on our experiments, we set q_k and r_k to 0.0005 and 0.01, respectively.

3. CONTROL DESIGN

In this section, we describe the control system for our indoor follow-me drone.

3.1 Primer for Control Approaches

Controller: For many applications, we need to ensure that the system output takes its desired value. To this end, we add a controller to the system as shown in Figure 5. The controller manipulates the system input to drive the output to its desired value. Thus, in a controlled system, the input is called *manipulated variable*, while

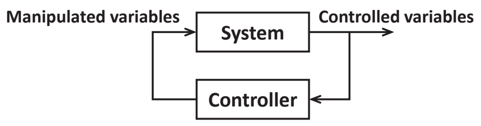


Figure 5: The system with a controller

the output is called *controlled variable*. The relationship between the input and output is called *system model*. In some systems, such a relationship is known. A straightforward way to control these systems is to derive the input for the desired output based on the system model. However, this approach is sensitive to disturbance and modeling error. To solve this issue, many controllers exploit *feedback*, *i.e.*, the mechanism that adjusts the input based on the current measurement of the output.

Primer for PID Control: **P**roportional **I**ntegral **D**erivative (PID) controllers have been widely used in industrial control systems for its simplicity. It computes the manipulated variable based on the error of the controlled variable as $u(t)=K_pe(t)+K_i\int_0^t e(\tau)d\tau+K_d\frac{de(t)}{dt}$, where $u(t)$ is the manipulated variable and $e(t)$ is the error of the controlled variable, which is computed as the difference between its measured value and desired value. K_p, K_i and K_d are non-negative coefficients of the proportional, integral, and derivative terms, respectively. The main advantage of PID is that it does not require knowledge of the system and can control the output simply based on the error $e(t)$. However, the lack of knowledge of the system does not come for free, and may affect the stability and convergence rate of the control system.

Primer on MPC: Model Predictive control (MPC) is a model-based control framework that works as follows [50]:

1. **Build the system model:** Determine controlled variables and manipulated variables, and build a system model to capture their relationship.
2. **Predict the future output:** Based on the system model, the future values of the controlled variable can be predicted in terms of the manipulated variable. Let $\mathbf{u}$ denote the values of the manipulated variable, where the i-th element stands for the value in the i-th period of the future. Let $\mathbf{y}_p(\mathbf{u})$ denote the controlled variable in the next N_p periods predicted based on the model, where N_p is called *prediction horizon*.
3. **Correct the prediction with feedback:** $\hat{\mathbf{y}}_p(\mathbf{u})=\mathbf{y}_p(\mathbf{u})+m-y_0$, where $\hat{\mathbf{y}}_p(\mathbf{u})$ is the corrected prediction, m and y_0 are the measured and predicted controlled variable during the current period, respectively. $m-y_0$ captures the prediction error (*e.g.*, coming from modeling error).
4. **Find the optimal input for the next period:** Let $\mathbf{y}_e(\mathbf{u})$ denote the difference between $\hat{\mathbf{y}}_p(\mathbf{u})$ and the desired values of the controlled variable in the next N_p periods. MPC minimizes the objective $|\mathbf{y}_e(\mathbf{u})|^2+w_u|\Delta\mathbf{u}|^2$, where w_u is a regularization parameter and $\Delta\mathbf{u}$ is the variation of the manipulated variable. The regularization term in the objective is to avoid significant variation in the manipulated variable, which can cause instability. Let $\mathbf{u}^*$ denote the optimal solution for the objective function. In the next period, the manipulated variable is set to the first element of $\mathbf{u}^*$.

The steps 2-4 are repeated every control period. We select MPC due to its following advantages. First, the controller can take action in advance by leveraging prediction, which is greatly helpful to improve convergence rate and avoid fluctuation of controlled variables. This is critical for high quality video taping. Second, MPC has well-understood tuning parameters, such as the prediction horizon and regularization parameter, which allows us to easily tune the controller for the optimal performance.

	Manipulated var.	Controlled var.	Method
1	pitch	drone-to-user dist.	MPC
2	yaw	drone orientation	MPC
3	roll	lateral velocity	PID

Table 2: Controllers in our system.

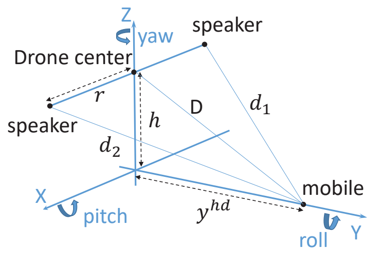


Figure 6: System 3D view. y^{hd} is the drone-to-user distance. d_1 and d_2 are the distances between the speakers and microphone. r is the drone's radius. h is the altitude difference between the drone and mobile.

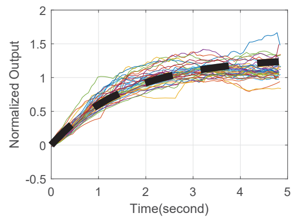
3.2 Control Design

To make a drone follow the user for high-quality video taping, we need to ensure (i) the drone-to-user distance is maintained at the desired value, (ii) the camera on the drone is facing the user, (iii) there is no random swing in the direction perpendicular to the forward motion of the drone (random swing along the forward direction is already captured in (i)). Thus, the controlled variables in our system are the drone-to-user distance, drone's orientation, and lateral velocity, as shown in Table 2.

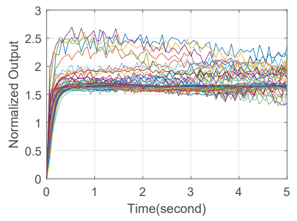
Our goal is to adjust the manipulated variables to make the controlled variables close to the desired values. The manipulated variables in most commercial drones are pitch, yaw, and roll. As shown in Figure 6, pitch captures a rotation angle along the x-axis and determines the forward velocity; yaw captures an angle along the z-axis and determines how the drone's orientation is changed over time; and roll captures rotation along the y-axis and determines the lateral velocity.

We develop three controllers to maintain the drone-to-user distance, drone's orientation, and lateral velocity at their desired values using pitch, yaw, and roll, respectively. As shown in Table 2, we use MPC for pitch and yaw since we can accurately model the relationship between the manipulated and controlled variables. We use PID for roll since the relationship between the lateral velocity and roll has large variability in the range of interest and a simple PID, which does not require a model, works better.

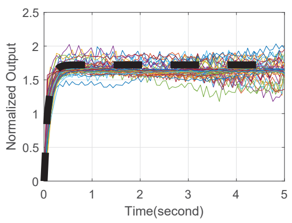
In order to apply MPC to our system, we should address several major challenges. First, MPC requires modeling the relationship between the manipulated and controlled variables, as mentioned in Section 3.1 (Step 1 in MPC). Every drone is different (*e.g.*, different weight). We need a simple method to model the drones. Second, one of the controlled variables in our system is the drone-to-user distance, which requires us to predict not only the drone movement but also the user movement (Step 2 in MPC). Third, we should accurately measure the controlled variables so that we can use the measured values as the feedback to our controllers to improve the robustness against modeling errors and disturbance (Step 3 in MPC). Below we address these challenges by (i) developing measurement-based models, (ii) measuring the controlled variables using Rabbit, and (iii) predicting a user's movement.



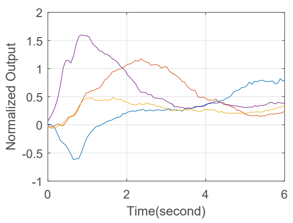
(a) $-v^f/u^p$ over time



(b) $-v^r/u^y$ over time



(c) Compensated $-v^r/u^y$



(d) $-v^l/u^r$ over time

Figure 7: Model measurements

3.3 Measurement-based Models

First, we conduct extensive measurements to develop system models that describe the relationship between our manipulated and controlled variables.

Drone forward movement vs. pitch: We build a model between the forward displacement of the drone and the pitch as follows.

1. **Find the relationship between forward velocity (v^f) and pitch (u^p).** Based on the aerodynamic principle [12], their relationship over time can be expressed as

$$v^f(t) = -\frac{\alpha}{\beta}(1 - e^{-\beta t})u^p(t), \qquad (6)$$

where t denotes time, and α and β are unknown constants and need to be determined.

2. **Determine the parameters in Equation 6 based on measurements.** To this end, we choose different values for the pitch, and measure the forward velocity of the drone over time. The velocity is estimated by the camera on the bottom of the drone using optical flow [12]. For each value of pitch, we plot a line representing $-v^f/u^p$ over time, as shown in Figure 7(a). Based on Equation 6, we know that all these lines follow the expression $\frac{\alpha}{\beta}(1-e^{-\beta t})$. Thus, we find α and β that fit these measurements best by minimizing the mean squared fitting error.

3. **Integrate Equation 6.** After integrating speed over time, we get the relationship between the forward displacement and pitch.

Drone orientation vs. yaw: We build a model between the drone orientation and yaw (u^y) using similar steps as above. We first find the expression for the rotation rate (v^r) in terms of u^y, then integrate the expression to obtain the model between the orientation and u^y. However, when we determine the relationship between v^r and u^y using an equation similar to Equation 6, we observe discrepancy in the collected data as shown in Figure 7(b). The stable value of $-v^r/u^y$ for small yaw (u^y) is higher than that for large u^y. As a result, we cannot find coefficients α and β in Equation 6 that fit all measurements.

To solve this problem, we design a compensator to reduce the discrepancy for different yaw (u^y) values. When we set $u^y = x$, we actually specify the yaw value $x/(1+e^{-cx})$ to the drone, where c is a constant. If x is small, the compensation gain $1/(1+e^{-cx})$ is smaller than 1, which reduces the rotation rate (v^r) compared to the uncompensated case. If x is large, $1/(1+e^{-cx})$ is close to 1, and v^r remains the same as the uncompensated case. As a result, $-v^r/u^y$ for small u^y is reduced, while that for larger u^y remains the same. We can tune the parameter c to make $-v^r/u^y$ for different u^y agree with each other as much as possible. Based on experiments, we set $c=15$. The values of $-v^r/u^y$ over time after compensation are shown in Figure 7(c). We see that the discrepancy among collected data is significantly reduced, and we are able to find a curve (*i.e.*, the dashed line) to fit all measurements.

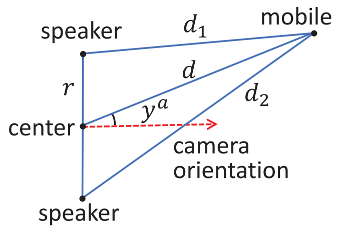


Figure 8: System top-down view.

Drone lateral velocity vs. roll: We try to model the relationship between the lateral velocity (v^l) and roll (u^r) using a similar procedure as above. Since we want to maintain the lateral velocity close to zero (so that there is no side-to-side swing), we focus on small lateral velocity. However, in this case, the relationship has large unpredictability and cannot be accurately modeled as shown in Figure 7(d). Without an accurate model, the effectiveness of MPC degrades. A simpler PID, which does not require any model, performs better. Thus, we use PID to stabilize the lateral velocity to zero. Our implementation uses $K_p=4$, $K_i=0.4$, $K_d=0.1$ based on our experiments.

3.4 Measuring Controlled Variables

Both MPC and PID need feedback to improve robustness against disturbance or model inaccuracy. Introducing feedback requires measuring the controlled variables (*i.e.*, the drone-to-user distance, drone's orientation, and lateral velocity) so that we can compare them with the desired values and adjust the system accordingly. For this purpose, we attach two speakers at each side of the drone and use Rabbit developed in Section 2 to derive the drone-to-user distance and drone's orientation as follows. For lateral velocity, we use the measurement provided by the drone as feedback [12].

Drone-to-user distance: The drone-to-user distance y^{hd} is defined as the horizontal distance between them as shown in Figure 6, while Rabbit measures the distance between the speakers and mobile (d_1 and d_2). To determine y^{hd}, we first calculate the distance from the mobile to the line defined by two speakers, denoted by D. Based on simple geometry, we can derive $D^2 = d_2^2 - (\frac{4r^2+d_2^2-d_1^2}{4r})^2$, where r is the drone's radius and can be easily measured. Let h denote the altitude difference between the drone and mobile. It is easy to see $y^{hd} = \sqrt{D^2 - h^2}$. The drone can measure its altitude using the ultrasound transceiver on its bottom [12] and the mobile's altitude can be approximated based on the user's height. Moreover, we find y^{hd} mainly depends on d_1 and d_2, and is robust to the estimation error in h: it sees little difference even when h has 10 cm error.

Drone orientation As shown in Figure 8, the orientation of the drone with respect to the user, *i.e.*, y^a, can be derived based on the distances between speakers and microphone (*i.e.*, d_1 and d_2). Using simple geometry, we obtain $y^a = \arcsin(\frac{d_2^2 - d_1^2}{4r\sqrt{\frac{d_1^2+d_2^2-2r^2}{2}}})$.

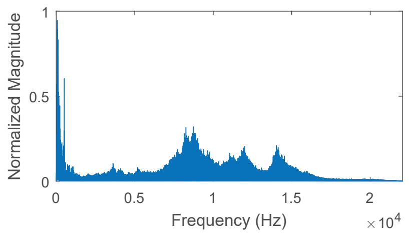


Figure 9: The spectrum of propeller noise.

3.5 Predicting a User's Movement

Standard MPC predicts the controlled variable using the future manipulated variable value and their relationship. However, in our system, certain controlled variable (*e.g.*, drone-to-user distance) depends not only on the manipulated variable (*e.g.*, pitch) but also on the user's movement. Therefore, to predict such controlled variables and apply MPC, we need to predict the user's movement.

We predict drone-to-user distance $y^{hd}(t)$ as $(t-t_0)v^h - y^d(t) + m$, where t denotes the time in the near future and t_0 is the current time. y^d is the displacement of the drone from t_0 to t, which can be predicted based on the model in Section 3.3. m is the currently measured drone-to-user distance. v^h is the user's moving speed and assumed to be constant during the prediction horizon (*i.e.*, how far future we need to predict), which is 1 s in our implementation.

Predicting y^{hd} requires the knowledge of v_h. We estimate it based on Doppler shift of acoustic signals, which provides the relative velocity between the user and drone. From Figure 6, it can be shown:

$$
\begin{aligned}
-v^{D,1} &= (v^h - v^d)\frac{y^{hd}}{d_1} + v^z\frac{h}{d_1} - v^l\frac{r}{d_1} \\
-v^{D,2} &= (v^h - v^d)\frac{y^{hd}}{d_2} + v^z\frac{h}{d_2} + v^l\frac{r}{d_2}
\end{aligned}
$$

where v^l, v^d, v^z are the drone velocities in the x, y, and z axes, respectively, and $v^{D,1}$ and $v^{D,2}$ are Doppler velocities with respect to the two speakers. In our system, y^{hd} is controlled to its desired value y_0^{hd}, and d_1 and d_2 are maintained equal to each other so that the camera on the drone perfectly faces the user. In this case, d_1 and d_2 are given by $d_0 = \sqrt{(y_0^{hd})^2 + h^2 + r^2}$ as shown in Figure 6. Moreover, since the drone flies at a constant altitude, v^z is zero. Thus, v^h can be estimated by $v^h = v^d - \frac{d_0}{2y_0^{hd}}(v^{D,1} + v^{D,2})$, where $v^{D,1}$ and $v^{D,2}$ are measured by Rabbit, while v^d is measured by the drone [12]. In addition, when we detect the user does not move using accelerometer readings, we directly set $v^h = 0$. This improves stability when the user is stationary as shown in our experiments (Section 5).

3.6 Important Parameters

We need to determine two parameters for MPC: (i) prediction horizon N_p, which represents how many control periods to predict in advance, and (ii) regularization parameter w_u, which determines if large variation of the manipulated variables is favored. We select these parameters based on experiments in Section 5.

4. IMPLEMENTATION

We implement our system on *A.R. Drone 2.0* and Samsung Galaxy S7. To support our scheme, we attach two speakers on the left and right sides of the drone, along with an audio amplifier board (Sure Electronics AA-AB32231). To power the amplifier and the speakers, we add a light-weight li-ion battery (EBL 6F22) in the drone's battery compartments. Altogether we add 140 grams to the drone.

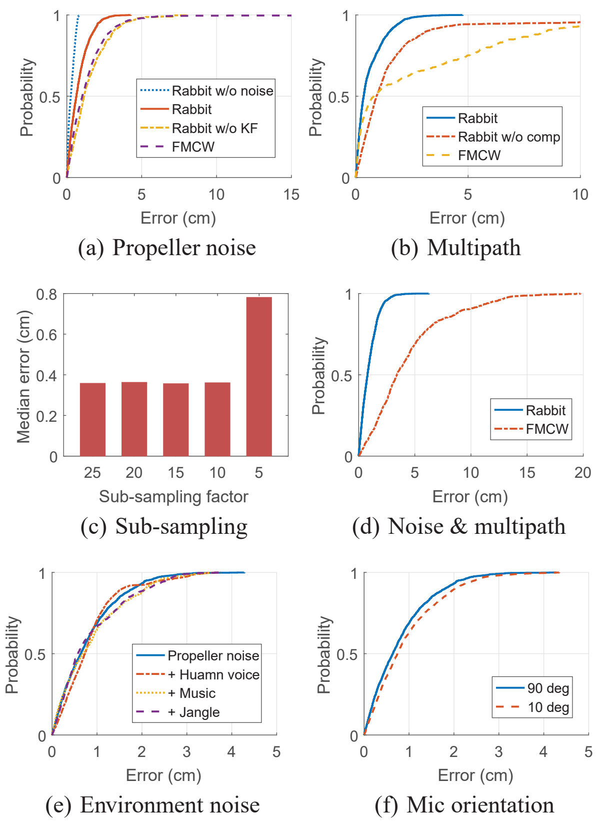


Figure 10: Distance estimation accuracy.

We use Galaxy S7 for tracking and controlling. It runs Android 6.0.1. We develop an application including all modules presented in the paper. The tracking part is implemented on Android NDK for efficiency, and other parts are based on Android SDK. We use the speakers on the drone to send chirp signals and sinusoid signals for distance and velocity measurements, respectively. The smartphone app analyzes the received signals to track the drone and derives control instructions. Then it uses APIs provided by Javadrone [27] to send instructions to the drone through Wi-Fi.

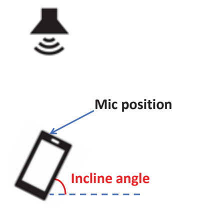
Figure 9 shows the spectrum of the propeller noise of our drone during flight. In the spectrum above 10 KHz, there is a dip between 13 KHz and 13.5 KHz. Also, the noise strength reduces noticeably beyond 17 KHz. Therefore, we use 17 - 19.5 KHz to send chirp signals for distance estimation. We also generate sinusoid signals at 13 KHz and 13.3 KHz for Doppler shift estimation. Signals around 13 KHz are audible, but they are not noticeable due to large noise from the propellers.

5. EVALUATION

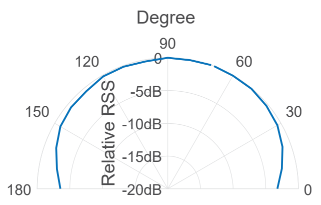
We evaluate Rabbit and DroneTrack in this section.

5.1 Distance Estimation

To evaluate the performance of distance estimation using Rabbit, we use a multi-camera 3D positioning system [44] to measure the distance between the speakers on the drone and the microphone on the mobile during the experiments. The positioning error of the system is less than 1mm. The camera measurements serve as the ground truth. In the experiments, the user holds a mobile, and faces its microphone towards the drone. The distance between the microphone and speakers is 1.5m. We collect 2000 distance measurements for each scheme in an experiment.



(a) Mic orientation



(b) Receiving beam pattern

Figure 11: Sensitivity of microphone orientation.

Performance under propeller noise: We evaluate the performance of distance estimation under propeller noise by having the drone hover in the air. Figure 10(a) compares three schemes: (i) Rabbit, (ii) Rabbit without Kalman filter, and (iii) distributed FMCW [37]. We also compare with Rabbit without propeller noise to understand the impact of the noise. We use [37] to estimate the initial distance required by these schemes. It has 6 mm error, which is not included in the figure. In our application, a small constant error has no impact on the user experience. A user only needs the drone to follow him at a constant distance for stable video taping, and does not care if the constant is a few millimeter away from the specified value. Among these schemes, Rabbit performs best with a median error of 0.63 cm. When the Kalman filter is disabled, Rabbit and distributed FMCW have similar performance. Kalman filter reduces the median error of Rabbit from 1.1 cm to 0.63 cm.

Performance under multipath: Next, we evaluate Rabbit under multipath. To exclude the impact of propeller noise, we put the drone on a stand instead of having it fly. As discussed in Section 2.2, the major interference caused by multipath comes from the paths that have similar lengths to the direct path. Therefore, we put a large cardboard near the direct path between the speaker and microphone, and move it during the experiment to generate interfering paths with varying lengths. We compare three schemes: Rabbit, Rabbit without compensating speaker distortion, and distributed FMCW. As shown in Figure 10 (b), Rabbit has 0.36 cm median error, which is much smaller than Rabbit without compensation. Rabbit also outperforms FMCW by using MUSIC since MUSIC is better at resolving multipath.

Performance of sub-sampling: We use the raw acoustic signals recorded from the above experiment to evaluate the impact of sub-sampling offline. We use different sub-sampling factors K but keep KM to 640 for fair comparison, where M is the order of the auto-correlation matrix. Figure 10(c) shows the median errors as K varies. For most K, the performance does not change. However, when K is small, the performance degrades because we need to use a large M for a small K and the eigenvalue decomposition for a large matrix has much larger numerical errors [46]. Thus, sub-sampling not only improves efficiency, but also enhances the accuracy of eigenvalue decomposition.

Performance under noise and multipath: We evaluate Rabbit under both propeller noise and multipath. As shown in Figure 10(d), Rabbit continues to perform well with a median error of 0.78 cm, and significantly outperforms FMCW.

Performance under environment noise: We evaluate Rabbit under both propeller noise and environment noise. We consider three types of environment noise: 1) human voices: two people continuously talk during the experiment; 2) music: different genres of music (Jazz, Pop, and Classic) are played together using the same volume as that for our tracking signals; 3) jangles generated by continuously jangling metal keys. These noise sources are 1m away from the microphone. As shown in Figure 10(e), the performance

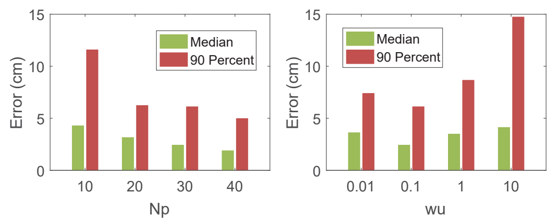


Figure 12: Errors on drone-to-user distance.

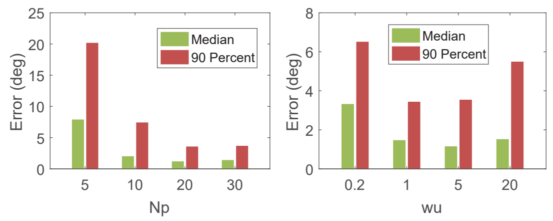


Figure 13: Errors on drone orientation.

under both propeller noise and environment noise is close to that with only propeller noise, because 1) the frequency of environment noise, such as human voices and music, is mostly below 10 KHz [38], while our tracking signals are higher than 13 KHz, and 2) the environment noise is much weaker than that of propeller noise.

Sensitivity of Mic orientation: To evaluate the sensitivity of the microphone orientation, we vary the inclination angle of the mobile from 0 to 180 degree, as shown in Figure 11(a). We record the received signal strength (RSS) for different inclination angles. When the angle is 90 degree, the microphone perfectly aligns with the speaker and RSS is maximized. We plot RSS at different angles relative to that at 90 degree in a polar coordinate system as shown in Figure 11(b), where the angular coordinate stands for the inclination angle and the radical coordinate represents the relative RSS. The RSS at the inclination angles of 10 or 170 degrees is only 3dB weaker. Thus, 3-dB beam width of the microphone is 160 degree. Figure 10(f) compares the distance estimation accuracy when the inclination angle is 10 and 90 degrees. The results indicate that Rabbit is not sensitive to microphone orientation.

5.2 System Evaluation

In this section, we evaluate the performance of DroneTrack system. The user holds a mobile in hand, and faces the microphone towards the drone. Since 3-dB beam width of the microphone is 160 degree, it does not need to align with the speakers perfectly. We use DroneTrack to make the drone follow the user with a desired distance of 1.5m. We quantify the user following performance using three metrics: the drone-to-user distance, the drone's orientation with respect to the user, and the drone's lateral velocity. To evaluate the following performance, we record the difference between their measured and desired values over time. We define the difference as *following errors*. We measure these metrics using Rabbit, whose accuracy is established in the previous section. We do not use the multi-camera system, because it only covers a 2m×3m area, but we fly the drone across tens of meters. It is too expensive to deploy 3D camera systems over such a large area.

Parameter study: We first study the impact of the MPC parameters on the control performance. There are two critical parameters: prediction horizon N_p and regularization parameter w_u.

For the drone-to-user distance control, the user moves along a 15 m straight line three times with a regular walking speed. Figure 12 compares the following errors as we vary N_p and w_u. We see that the following errors reduce as N_p increases. However, further

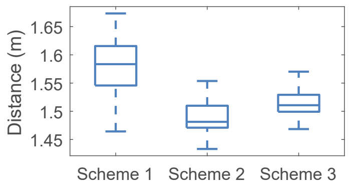
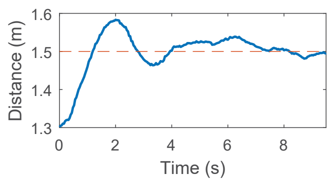
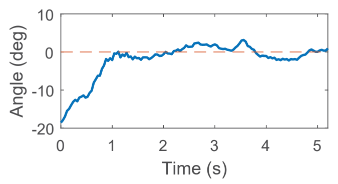


Figure 14: Stability of the drone.



(a) Drone-to-user distance



(b) Drone orientation

Figure 15: Convergence behavior.

increasing N_p beyond 40 helps little, because it is challenging to accurately predict far into the future and increasing the prediction error degrades the effectiveness of MPC. Our results show N_p=40 and w_u=0.1 yield the best performance. In this case, the median error of the drone-to-user distance is 1.9 cm.

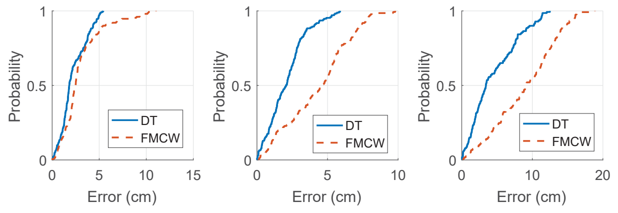
For the drone orientation control, the user moves along a circle with the radius equal to 1.5m three times . Figure 13 shows the following errors as N_p and w_u vary. Under the most favorable setting (N_p=20 and w_u=5), the median following error is 1.1 degree.

Below we use the parameters that yield the best performance. For the distance control, N_p=40 and w_u=0.1. For the orientation control, N_p=20 and w_u=5.

Stationary scenario: We study the stability of DroneTrack when the user does not move. We compare three schemes: (i) hovering mode: a built-in function of our drone, which makes the drone stay at the same position; (ii) DroneTrack without detecting if the user is stationary; (iii) DroneTrack, which detects when the user is stationary and sets v^h=0 whenever the user is stationary. To evaluate these schemes, we measure the drone-to-user distance over one minute. Figure 14 shows the results with the box-plot. The span (in y-direction) of each box reflects the fluctuation in the distance. Rabbit, which detects the stationary user performs the best due to the smaller prediction error when it knows the user is stationary.

Convergence behavior: To evaluate how fast drone-to-user distance converges to its desired value, the user stands at 1.3 m from the drone and does not move. Then we let the drone follow the user. We record the distance over time. As shown in Figure 15(a), it takes 4 s to converge to the desired distance. We conduct a similar experiment to evaluate the convergence behavior of drone orientation. As shown in Figure 15(b), given a 20-degree disturbance, it takes less than 1 s to converge to the desired orientation. These results show that our system can quickly respond to sudden changes.

Impact of user movement speeds: We let a user move along a 15 m straight line at various speeds (3 times for each speed), and measure the drone-to-user distance error. For slow, normal, and fast walking, the speeds are 0.4 m/s, 1 m/s, and 2 m/s, respectively.



(a) Slow speed (b) Normal speed (c) Fast speed

Figure 16: Performance under various user speeds.

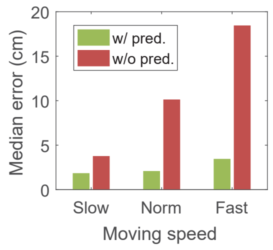


Figure 17: The impact of user movement prediction.

Figure 16 shows the performance of DroneTrack (denoted by "DT") under various speeds. For comparison, we replace Rabbit with distributed FMCW (denoted by "FMCW") in DroneTrack and evaluate its performance. DroneTrack is accurate across different speeds: the 90-percentage error for slow and norm speeds is 4 cm, and that for fast movement is 10 cm. In comparison, distributed FMCW has much larger error. The improvement of DroneTrack over distributed FMCW is higher for faster speeds. FMCW may have a large tracking error due to multiple nearby propagation paths (*e.g.*, caused by tables, chairs, or walls), which results in a large following error. When the user moves faster, a large tracking error makes it harder to converge to the desired distance.

Impact of user movement prediction: Next we compare the performance with and without predicting user movement using the same setup as the previous experiment. As Figure 17 shows, removing prediction significantly increases the median error of drone-to-user distance by 106%, 391%, and 438% for slow, normal, and fast moving speeds, respectively. In comparison, prediction helps to reduce the performance gap between the fast and slow movement since the drone is able to take actions in advance and maintain a constant distance to the user.

Impact of following distances: We evaluate DroneTrack by varying the desired distances between the drone and user. For each distance, we let a user move along a 15 m straight line for three times. Figure 18 shows that the median following error only marginally increases from 2 cm to 3 cm as the desired distance increases from 2 m to 5 m. As the distance increases, both the signals used for tracking and propeller noise attenuate. Thus the SINR does not change significantly, and the accuracy of distance estimation remains similar.

Video taping experiment: Next we let a user move along a fixed

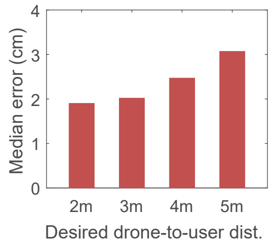


Figure 18: The impact of following distances.

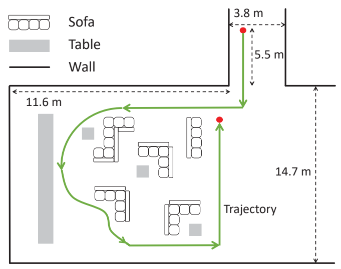


Figure 19: The trajectory for video experiments.

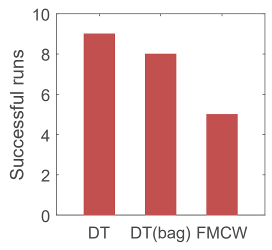


Figure 20: Successful runs.

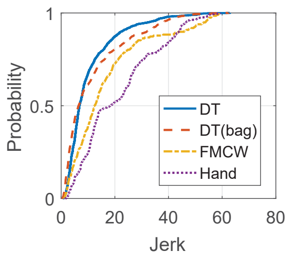
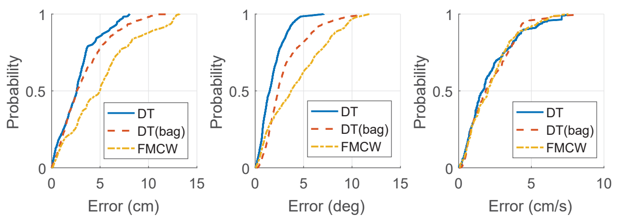


Figure 21: Video stability.

trajectory in our department building as shown in Figure 19. We use the fixed trajectory for fair comparison across different schemes. The trajectory includes various natural walking patterns, such as straight paths, curved paths, and left and right turns. There are furniture (*e.g.*, tables and sofa) and walls along the trajectory. When the user walks, the drone follows him and the camera on it is used to record videos. We compare the performance of three schemes: 1) DroneTrack where the user holds the mobile in hand (denoted by "DT"); 2) DroneTrack where the mobile is placed in a PC sleeve with its microphone facing the drone and the user carries the case (denoted by "DT (bag)"); 3) using distributed FMCW for distance estimation instead of Rabbit (denoted by "FMCW"). As the baseline, we let a person follow the user and take videos with a camera held in his hand (denoted by "Hand"). For each scheme, we conduct experiments along the trajectory for 10 times. An experiment is considered as a failure if the user is outside the view at any point in the video. This may happen due to inaccurate distance estimation or significant WiFi delay which leads to the loss of control instructions.

Figure 20 compares the number of successful runs across different schemes. As we can see, DroneTrack in hand yields the best performance, closely followed by DroneTrack in bag. The gap between the two is small, which demonstrates the robustness of DroneTrack. In comparison, distributed FMCW based approach succeeds only 50% since it cannot effectively resolve multipath.

Figure 21 and Figure 22 compare the video stability and the following errors of the successful runs for each scheme, respectively. For video stability, we use the standard jerk metric [16], which captures the smoothness between two consecutive frames in the video. The lower the jerk metric, the more stable the videos. As we can see, DroneTrack with the mobile in hand performs the best and followed by DroneTrack with the mobile in bag. The hand-held video taping performs the worst due to hand vibration during walking and there is no optical-image-stabilization system on our camera. From Figure 22, we can observe that DroneTrack performs well: the median drone-to-user distance error is 2.5 cm (in hand) and 2.6 cm (in bag); the median drone orientation error is 1.5 deg (in hand) and 2.3 deg (in bag); and the lateral velocity error is 1.6 cm/s (in hand) and 1.8 cm/s (in bag). Note that the lateral velocity control does not depend on the performance of distance estimation, and thus different schemes perform similarly.



(a) Drone-user dist. (b) Drone orientation (c) Lateral velocity

Figure 22: Following errors.

5.3 Computation cost

Finally, we quantify the computation cost. Table 3 summarizes the mean CPU usage and processing delay of DroneTrack running on Galaxy S7. We use *adb shell top* command to sample the CPU usage every second during a 200-second period. As we can see, the CPU usage for tracking is only 13%, and that for the whole system is 42%. Tracking takes 9 ms to process, and the whole system takes 13 ms. Also, we observe the battery of Galaxy S7 (3000 mAh) drops from 100% to 97% after 10-minute running of DroneTrack with the screen off. We plan to further improve the energy efficiency by porting more codes from Android SDK to NDK.

	CPU Usage (%)	Delay (ms)
Tracking	13	9
Overall	42	13

Table 3: Computation cost.

6. DISCUSSION

Loss of direct path signals: Our approach estimates the distance between the drone and user by extracting the signals propagated through the direct path between them. Since acoustic signals can penetrate through some materials, such as clothes, our system continues to work when the mobile is in a bag.

However, there are a few cases where the received signals via the direct path are too weak to detect. This can occur when the microphone is completely opposite to the speaker. Note that we can still easily detect signals from the direct path when the incline angle of the mobile is within 180 degree as shown in Figure 11(a). This is easy to satisfy in most cases since our control algorithm can automatically adjust the drone's orientation when the user makes turns. The direct path may also be blocked by obstacles, which acoustic signals cannot penetrate, such as human body and concrete pillars. If blocking is temporary, it can be handled by extrapolation using the recent past measurements. Long-term blocking is more challenging and may require using other signals in addition to acoustic signals for tracking. We plan to explore it in future.

7. RELATED WORK

Our work is related to the following research areas: (i) tracking and localization, (ii) drone control and modeling, and (iii) autonomous user following with a drone.

Tracking and localization: Acoustic signals are attractive for localization and tracking due to their potential to achieve high accuracy and widely available speakers and microphones. Several acoustic-based tracking systems are developed. [61] uses Doppler shifts of acoustic signals for tracking and achieves 1.4 cm median tracking error. [39] develops a device-free tracking based on cross correlation of acoustic signals and OFDM, while [55] uses the phase change of acoustic signals. Both [39] and [55] assume that the target to be tracked is the only moving object in the environment. However, in our case the drone and mobile are both moving and all paths can change over time. [42] estimates the distance between two phones by having them transmit and receive the beeps to cancel out the processing delay and clock difference. [62] further improves [42] for mobile gaming applications. [37] proposes a distributed FMCW-based approach to support distance estimation between a separate sender and receiver, and combines various measurements to achieve 6 mm tracking error. We apply the distributed FMCW in [37], but make several important improvements: 1) compensate for the speaker distortion and use MUSIC to process FMCW to achieve better resolution of resolving multiple paths, 2) use velocity measurements and Kalman filter to reduce noise in distance estimation.

RF signals are also widely used for tracking and localization. [59] uses an array of RF antennas to achieve 23 cm median error. [53] uses multiple frequency bands for localization with a single AP point. Its median distance estimation error is 14.1 cm in light-of-sight (LOS) and 20.7 cm in non-LOS case. [54] relies on multiple RFID antennas and angle-of-arrival (AoA) estimation to achieve 3.7 cm tracking error. [5] and [4] use RF-based FMCW signals to localize a person with 11.7 cm error. [56] achieves 0.8 cm tracking error using highly directional and steerable 60 GHz antennas.

MUSIC is used to derive AoA using multiple antennas. Combined with propagation delay measurements (*e.g.*, using FMCW [57, 7, 49, 13]) or channel state information (*e.g.*, [51, 58, 32]), the target can be localized. However, since smartphones do not have a microphone array, our acoustic tracking system cannot derive AoA using MUSIC. [60] applies MUSIC in the frequency domain to estimate the propagation delay, and combines multiple frequency bands to improve the estimation accuracy. In [3, 28], MUSIC is applied to enhance the resolution of FMCW-based propagation delay measurement. Our tracking system also leverages this idea, but significantly improves the performance by (i) calibrating and compensating for the speaker distortion, (ii) improving the peak selection under strong propeller noise, (iii) improving the efficiency for mobile implementation, and (iv) combining Doppler measurements using Kalman filter.

Drone control and modeling: There have been significant works on controlling drones [47, 40, 52, 15, 6, 29, 10, 24]. We can broadly classify them into three categories: 1) stabilize a drone at a target location, 2) make a drone follow a pre-defined path, 3) make a drone reach a set of positions at specified time. In all these works, the target positions or paths are given in advance, which is different from our application, where the drone is required to follow a user, whose movement is not known in advance.

Many algorithms have been applied to control drones, such as PID [47], nonlinear guidance low [52], MPC [24], vector field control [40], and gain-scheduled control [29]. Among them, we find MPC best suits our need due to its predictive power. We address practical challenges in applying MPC to our context, including building the system models for the drone, measuring the distance and orientation between the drone and user as the control feedback, and predicting the user movement.

Several research papers model drone dynamics. [9, 34, 11] model the relationship between the drone movement and its rotor thrusts. However, we do not have access to the low-level attributes like rotor thrusts. Instead, we model the relationship between the drone displacement and its inclination angles. Our modeling procedures are closer to [23, 12], but differ from these works in that we find the non-linear behavior of the drone, and apply the compensation to address the issue.

User following: There have been some works that let a drone automatically follow a user [18, 21, 30, 35, 43, 8, 22, 36, 53]. Most off-the-shelf drones rely on GPS [18] to enable follow-me. GPS-based approaches are only feasible outdoors. When GPS is not available, computer vision (CV) is most widely used for tracking a user [21, 30, 35, 43, 8, 22, 36]. However, CV has a few limitations. First, the accuracy of CV-based tracking is limited due to two main reasons: 1) A drone vibrates during flight due to air disturbance. The vibration blurs the images captured by the drone, which significantly degrades the tracking performance. 2) CV based tracking needs depth information. Depth sensors or multiple cameras are required to get the depth information, but they are not available on most drones. Therefore, existing works use the relative size of a target in the image for tracking [30, 43, 8, 36]. However, such an approach is not accurate and sensitive to target deformation and view angle variation. Due to these facts, the error for CV-based user following is typically decimeters or higher (*e.g.*, [35, 43, 8, 36]). Second, CV based approaches are sensitive to visual interference, and its performance degrades in a busy background [8]. As a result, state-of-the-art CV approaches can successfully detect the target for only 67% - 88% [22]. Our acoustic based indoor follow-me drone is not affected by visual interference. To avoid acoustic interference, we use frequencies higher than 13 KHz, which are different from noise in the environment, such as human voices and music, whose frequency is mostly below 10 KHz [38]. We also develop several mechanisms to combat with noise generated by the propellers. Thus, our system is robust to the interference. In addition, the computation of CV based approaches is too heavy for mobile devices. [35, 8, 22] use sophisticated CV approaches, which are more accurate and robust than simple CV methods, such as color filtering, but require extracting and matching image descriptors for the target. These operations take from 30 ms to 3 s for each image when running on a mobile device [25]. In contrast, our tracking approach takes 9 ms for processing 40 ms acoustic samples, and the CPU usage during this period is only 13%.

[53] develops a novel tracking approach using WiFi signals to achieve decimeter-level tracking, and uses one experiment to show their tracking can be applied to follow-me drones. We believe an effective control algorithm is essential to enable follow-me drones under general user movement. Our paper advances [53] by significantly improving the tracking accuracy, designing a holistic system including both tracking and control, and extensively evaluating user-following performance.

Remarks: In short, our work complements existing work by enhancing the tracking accuracy and efficiency and developing a practical control system to allow a drone to follow a user with desired following distance and orientation.

8. CONCLUSION

This paper presents a novel design and implementation of a system that accurately tracks the drone and controls its movement to maintain a specified distance and orientation to the user. Our tracking approach (Rabbit) advances existing tracking technologies by improving robustness under multipath and has applications beyond drones. We further build a holistic follow-me drone system (DroneTrack) on top of Rabbit for low-end drones to achieve high following accuracy in indoor environments through measurement-based modeling, deriving accurate feedback using Rabbit, and predicting user movement.

9. REFERENCES

[1] Microsoft X-box Kinect. http://xbox.com.

[2] Nintendo Wii. http://www.nintendo.com/wii.

[3] M. Abou-Khousa, D. Simms, S. Kharkovsky, and R. Zoughi. High-resolution short-range wideband FMCW radar measurements based on MUSIC algorithm. In *IEEE Instrumentation and Measurement Technology Conference*, page I2MTC, 2009.

[4] F. Adib, Z. Kabelac, and D. Katabi. Multi-person localization via RF body reflections. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 279–292, 2015.

[5] F. Adib, Z. Kabelac, D. Katabi, and R. Miller. WiTrack: motion tracking via radio reflections off the body. In *Proc. of NSDI*, 2014.

[6] A. P. Aguiar, J. P. Hespanha, et al. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE Transactions on Automatic Control*, 52(8):1362–1379, 2007.

[7] F. Belfiori, W. van Rossum, and P. Hoogeboom. 2D-MUSIC technique applied to a coherent FMCW MIMO radar. In *Radar Systems (Radar 2012), IET International Conference on*, pages 1–6. IET, 2012.

[8] V. Bevilacqua and A. Di Maio. A computer vision and control algorithm to follow a human target in a generic environment using a drone. In *International Conference on Intelligent Computing*, pages 192–202. Springer, 2016.

[9] P. Bouffard. On-board model predictive control of a quadrotor helicopter: Design, implementation, and experiments. Technical report, DTIC Document, 2012.

[10] E. Bregu, N. Casamassima, D. Cantoni, L. Mottola, and K. Whitehouse. Reactive control of autonomous drones. MOBISYS, 2016.

[11] T. Bresciani. Modelling, identification and control of a quadrotor helicopter. *MSc Theses*, 2008.

[12] P.-J. Bristeau, F. Callou, D. Vissiere, N. Petit, et al. The navigation and control technology inside the AR. Drone micro UAV. In *18th IFAC world congress*, volume 18, pages 1477–1484, 2011.

[13] J. Choi, J. Park, and D. Yeom. High angular resolution estimation methods for vehicle FMCW radar. In *Proceedings of 2011 IEEE CIE International Conference on Radar*, volume 2, pages 1868–1871. IEEE, 2011.

[14] The drones report: market forecasts, regulatory barriers, top vendors, and leading commercial applications. http://www.businessinsider.com/uav-or-commercial-drone-market-forecast-2015-2.

[15] P. Encarnação and A. Pascoal. Combined trajectory tracking and path following: an application to the coordinated control of autonomous marine craft. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 1, pages 964–969. IEEE, 2001.

[16] F. D. et al. Time-optimal path following for robots with trajectory jerk constraints using sequential convex programming. 2013.

[17] R. Faragher et al. Understanding the basis of the Kalman filter via a simple and intuitive derivation. *IEEE Signal processing magazine*, 29(5):128–132, 2012.

[18] Top 10 drones with follow me mode in 2016. http://www.top10drone.com/top-10-drones-follow-me-mode/.

[19] 12 best follow me drones and follow me technology reviewed. https://www.dronezon.com/drone-reviews/best-follow-me-gps-mode-drone-technology-reviewed/.

[20] Making gated-impulse frequency measurements using ARTA. http://opencv.org/.

[21] J.-E. Gomez-Balderas, G. Flores, L. G. Carrillo, and R. Lozano. Tracking a ground moving target with a quadrotor using switching control. *Journal of Intelligent & Robotic Systems*, 70(1-4):65–78, 2013.

[22] K. Haag, S. Dotenco, and F. Gallwitz. Correlation filter based visual trackers for person pursuit using a low-cost quadrotor. In *Innovations for Community Services (I4CS), 2015 15th International Conference on*, pages 1–8. IEEE, 2015.

[23] A. Hernandez, C. Copot, R. De Keyser, T. Vlas, and I. Nascu. Identification and path following control of an AR. Drone quadrotor. In *System Theory, Control and Computing (ICSTCC), 2013 17th International Conference*, pages 583–588. IEEE, 2013.

[24] A. Hernandez, H. Murcia, C. Copot, and R. De Keyser. Model predictive path-following control of an AR. Drone quadrotor. In *Proceedings of the XVI Latin American Control Conference (CLCAÂŠ14), Cancun, Quintana Roo, Mexico*, pages 14–17, 2014.

[25] M. A. Hudelist, C. Cobârzan, and K. Schoeffmann. Opencv performance measurements on mobile devices. In *Proceedings of International Conference on Multimedia Retrieval*, page 479. ACM, 2014.

[26] V. K. Ingle and J. G. Proakis. *Digital signal processing using MATLAB*. Cengage Learning, 2016.

[27] Javadrone. https://github.com/codeminders/javadrone.

[28] W. Jiang, S. Pennock, and P. Shepherd. A novel w-MUSIC algorithm for GPR target detection in noisy and distorted signals. In *2009 IEEE Radar Conference*, pages 1–6. IEEE, 2009.

[29] I. Kaminer, A. Pascoal, E. Hallberg, and C. Silvestre. Trajectory tracking for autonomous vehicles: An integrated approach to guidance and control. *Journal of Guidance, Control, and Dynamics*, 21(1):29–38, 1998.

[30] J. Kim and D. H. Shim. A vision-based target tracking control system of a quadrotor by using a tablet computer. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 1165–1172. IEEE, 2013.

[31] H.-H. Ko, K.-W. Cheng, and H.-J. Su. Range resolution improvement for FMCW radars. In *Radar Conference, 2008. EuRAD 2008. European*, pages 352–355. IEEE, 2008.

[32] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti. Spotfi: Decimeter level localization using WiFi. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 269–282. ACM, 2015.

[33] Leap motion. https://www.leapmotion.com/.

[34] T. Lee, M. Leoky, and N. H. McClamroch. Geometric tracking control of a quadrotor UAV on SE (3). In *49th IEEE conference on decision and control (CDC)*, pages 5420–5425. IEEE, 2010.

[35] F. Lin, X. Dong, B. M. Chen, K.-Y. Lum, and T. H. Lee. A robust real-time embedded vision system on an unmanned rotorcraft for ground target following. *IEEE Transactions on Industrial Electronics*, 59(2):1038–1049, 2012.

[36] J. J. Lugo and A. Zell. Framework for autonomous on-board navigation with the AR. Drone. *Journal of Intelligent & Robotic Systems*, 73(1-4):401–412, 2014.

[37] W. Mao, J. He, and L. Qiu. CAT: high-precision acoustic motion tracking. In *Proc. of ACM MobiCom*, 2016.

[38] The frequency spectrum, instrument ranges, and EQ tips. http://www.guitarbuilding.org/wp-content/uploads/2014/06/Instrument-Sound-EQ-Chart.pdf.

[39] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota. FingerIO: Using active sonar for fine-grained finger tracking. In *Proc. of ACM CHI*, pages 1515–1525, 2016.

[40] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529, 2007.

[41] Opencv. http://audio.claub.net/tutorials/FR%20measurement%20using%20ARTA.pdf.

[42] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan. BeepBeep: a high accuracy acoustic ranging system using COTS mobile devices. In *Proc. of ACM SenSys*, 2007.

[43] J. Pestana, J. L. Sanchez-Lopez, S. Saripalli, and P. Campoy. Computer vision based general object following for GPS-denied multirotor unmanned vehicles. In *2014 American Control Conference*, pages 1886–1891. IEEE, 2014.

[44] Phasespace motion capture. http://www.phasespace.com/impulse-motion-capture.html.

[45] B. D. Rao and K. Hari. Performance analysis of root-MUSIC. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):1939–1949, 1989.

[46] H. Ren. On the error analysis and implementation of some eigenvalue decomposition and singular value decomposition. 1996.

[47] R. Rysdyk. UAV path following for constant line-of-sight. In *2th AIAA Unmanned Unlimited. Conf. and Workshop and Exhibit, San Diego, CA*, 2003.

[48] R. O. Schmidt. A signal subspace approach to multiple emitter location spectral estimation. *Ph. D. Thesis, Stanford University*, 1981.

[49] M. Schoor and B. Yang. High-resolution angle estimation for an automotive FMCW radar sensor. In *Proc. of Intern. Radar Symposium (IRS), Cologne, Germany*, 2007.

[50] D. E. Seborg, D. A. Mellichamp, T. F. Edgar, and F. J. Doyle III. *Process dynamics and control*. John Wiley & Sons, 2010.

[51] S. Sen, J. Lee, K.-H. Kim, and P. Congdon. Avoiding multipath to revive inbuilding WiFi localization. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 249–262. ACM, 2013.

[52] P. Sujit, S. Saripalli, and J. B. Sousa. Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. *IEEE Control Systems*, 34(1):42–59, 2014.

[53] D. Vasisht, S. Kumar, and D. Katabi. Decimeter-level localization with a single WiFi access point. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 165–178, 2016.

[54] J. Wang, D. Vasisht, and D. Katabi. RF-IDraw: virtual touch screen in the air using RF signals. In *Proc. of ACM SIGCOMM*, 2014.

[55] W. Wang, A. X. Liu, and K. Sun. Device-free gesture tracking using acoustic signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 82–94. ACM, 2016.

[56] T. Wei and X. Zhang. mTrack: high precision passive tracking using millimeter wave radios. In *Proc. of ACM MobiCom*, 2015.

[57] P. Wenig, M. Schoor, O. Gunther, B. Yang, and R. Weigel. System design of a 77 ghz automotive radar sensor with superresolution DOA estimation. In *2007 International Symposium on Signals, Systems and Electronics*, pages 537–540. IEEE, 2007.

[58] Y. Xie, Z. Li, and M. Li. Precise power delay profiling with commodity WiFi. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 53–64. ACM, 2015.

[59] J. Xiong and K. Jamieson. Arraytrack: A fine-grained indoor location system. In *Proc. of NSDI*, pages 71–84, 2013.

[60] J. Xiong, K. Sundaresan, and K. Jamieson. Tonetrack: Leveraging frequency-agile radios for time-based indoor wireless localization. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 537–549. ACM, 2015.

[61] S. Yun, Y. chao Chen, and L. Qiu. Turning a mobile device into a mouse in the air. In *Proc. of ACM MobiSys*, May 2015.

[62] Z. Zhang, D. Chu, X. Chen, and T. Moscibroda. Swordfight: Enabling a new class of phone-to-phone action games on commodity phones. In *Proc. of ACM MobiSys*, 2012.

[63] C. Zhou, F. Haber, and D. L. Jaggard. A resolution measure for the MUSIC algorithm and its application to plane wave arrivals contaminated by coherent interference. *IEEE Transactions on signal processing*, 39(2):454–463, 1991.