Acoustic Localization System for Precise Drone Landing

Yuan He, *Senior Member, IEEE,* Weiguo Wang, *Student Member, IEEE,* Luca Mottola, *Senior Member, IEEE,* Shuai Li, *Student Member, IEEE,* Yimiao Sun, *Student Member, IEEE,* Jinming Li, *Student Member, IEEE,* Hua Jing, *Member, IEEE,* Ting Wang, *Member, IEEE,* Yulei Wang, *Member, IEEE*

Abstract—We present MICNEST: an acoustic localization system enabling precise drone landing. In MICNEST, multiple microphones are deployed on a landing platform in carefully devised configurations. The drone carries a speaker transmitting purposefully-designed acoustic pulses. The drone may be localized as long as the pulses are correctly detected. Doing so is challenging: *i)* because of limited transmission power, propagation attenuation, background noise, and propeller interference, the Signal-to-Noise Ratio (SNR) of received pulses is intrinsically low; *ii)* the pulses experience non-linear Doppler distortion due to the physical drone dynamics; *iii)* as location information is used during landing, the processing latency must be reduced to effectively feed the flight control loop. To tackle these issues, we design a novel pulse detector, Matched Filter Tree (MFT), whose idea is to convert pulse detection to a tree search problem. We further present three practical methods to accelerate tree search jointly. Our experiments show that MICNEST can localize a drone 120 m away with 0.53% relative localization error at 20 Hz location update frequency. For navigating drone landing, MICNEST can achieve a success rate of 94 %. The average landing error (distance between landing point and target point) is only 4.3 cm.

Index Terms—Localization, Acoustic, Microphone Array, Drone

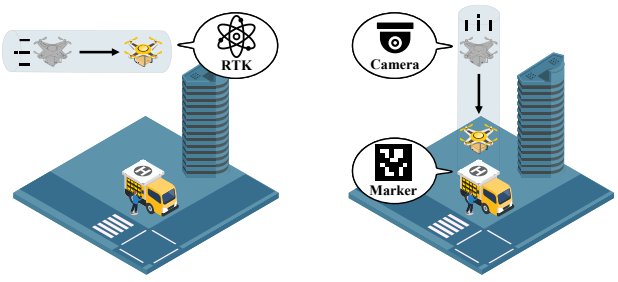
1 INTRODUCTION

Aerial drone technology represents a new breed of computing platform [1], enabling applications in a range of fields such as agriculture, search and rescue, film-making, impromptu networking, and logistics. Landing is a key step in a drone's operation [2], one that is both delicate as the risk of damaging the drone itself or the surroundings is highest [3], and an essential component in many next-generation applications enabled by drone technology.

Target scenarios. With the roll-out of 5G networks, beyond-line-of-sight operation becomes possible: a drone flies autonomously over long distances without a physically co-located human pilot, but connected to an Internet back-end that monitors its operation in real time. This ability unlocks a range of potential applications, such as long-range visual inspections executed by multiple drones in a shared airspace [1] and instant asset delivery with drones.

Let us examine further the latter application. Compared to ground couriers, drones can bypass the complex urban traffic and deliver packages in a much shorter time. This is ideal for time-sensitive deliveries, such as medical supplies [4] and food [5], [6], [7], and also caters for contactless deliveries, which helps reduce the spread of viral diseases. Many companies are exploring the commercial feasibility of instant deliveries with drones; for example, Amazon [8], Alphabet Project Wing [9], Wal-Mart [10], [11], JD.com [12], Domino's [5], UPS [13], Ele.me [6], and Meituan [7].

Instant delivery with a drone unfolds as follows. After the package is placed onboard, the drone takes off, climbs to cruising altitude, and heads towards the destination. The latter is typically a self-collection station[1] near the customer. When the drone is close to the destination, a *precise landing* procedure is initiated, as depicted in Fig. 1. The drone first approaches the landing platform horizontally to achieve vertical alignment, as in Fig. 1(a). Next, the drone starts the descent, shown in Fig. 1(b). Once the drone safely docks onto the landing platform, the package is dropped into the self-collection station, where the customer fetches it.



(a) Horizontal approach. **(b) Vertical descent.**

Figure 1: Two phases of drone landing.

The consequences of not landing precisely can only be underestimated. Loss of the transported good is the most obvious. Should the drone miss the landing platform even partly, it would quickly lose control and crash, possibly damaging objects or hurting people [14]. The key requirements to avoid similar mishaps are as follows: *i) cm-level positioning accuracy ii) at 100+m altitude*, with positioning accuracy improving as the altitude decreases, and *iii) low-latency location updates* to feed the flight control loop. These requirements form the essence of our research problem.

1. These self-collection stations are set up and operated by the drone delivery companies and shared with the nearby residents.

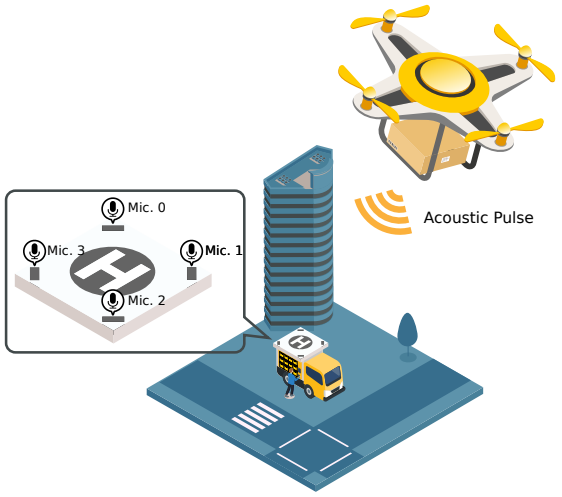


Figure 2: An illustration of MICNEST.

State of the art. During the horizontal approach, shown in Fig. 1, the flight control loop mainly relies on Real-Time Kinematic Positioning (RTK) [15]. During the descent, the flight control loop must keep the drone horizontally centered over the landing platform. As we further articulate in Sec. 2, settings exist, such as urban canyons, where the accuracy of GPS or RTK degrades with decreasing altitude up to making either system essentially unusable [16], [17], [18]. Auxiliary anchor-based systems may assist the drone during the descent such as visual markers [19], [20], [21], [22], [23], laser stations [24], ultra-wide band (UWB) stations [25], [26], or motion capture cameras [27].

Few of these systems, however, fulfill the requirements mentioned earlier. The only technique that may potentially do so is visual markers, read by downward-facing cameras the drone must be equipped with. According to our real-world experiences, we find that these techniques, however, exhibit key limitations:

1) They are *sensitive to lighting variations*; visual markers are difficult to detect in the fog or at night; equipping the drone with a light source may partly ameliorate the problem, however limiting the operational range to 20-30m and only obtaining unstable performance.
2) They are *constrained in horizontal coverage*; depending on the camera field of view and the drone's vertical alignment with the marker, the camera may not completely capture the marker, resulting in a localization failure [28].
3) They *limit system throughput*; as visual markers require line of sight between the camera and the marker, drones necessarily need to land one by one; otherwise, the first drone that commences the descent visually blocks the marker for all others.

MICNEST. We present MICNEST, an acoustic localization system to assist drones in precise landing, summarized in Sec. 3. As illustrated in Fig. 2, A speaker carried by a drone broadcasts purposefully-designed acoustic pulses. Multiple microphones are deployed on the landing platform in carefully devised configurations. By localizing the speaker from the microphone signals, we localize the drone during the descent. The localization results are transmitted to the drone via WiFi and taken as an input for navigation, closing the control loop that drives the drone onto the landing platform.

MICNEST is rooted in the unique features of acoustic signals. The spatial resolution of a signal is proportional to its speed and inversely proportional to its bandwidth [29]. Thus, the slower a signal is, the finer spatial resolution it can provide. Acoustic signals with a limited bandwidth, say 24 kHz, can provide a fine-grained spatial resolution, around 0.71 cm when the sampling rate is 48 kHz. In comparison, the spatial resolutions of RF signals like UWB with a 1.3 GHz bandwidth or mmWave with a 4 GHz bandwidth are 10 cm [30] and 3.75 cm [31], respectively.

Compared to the visual techniques, using acoustic signals further allows MICNEST to *i)* operate regardless of lighting conditions and *ii)* provide larger horizontal coverage. Further, we adopt Pseudo-Random Noise (PRN) to modulate acoustic pulses emitted by a drone. Because PRN pulses are orthogonal as long as they are statistically independent of each other, we can detect these pulses separately from the collided signal and identify which drone each pulse corresponds to. This makes MICNEST able to *iii)* provide concurrent detection and localization of multiple drones.

MICNEST provides, nonetheless, additional benefits. The use of PRN pulses makes MICNEST friendly to human ear. As drones may operate in populated areas, pulses should not cause acoustic discomfort, yet PRN has the same acoustic characteristics as white noise and is almost imperceptible to human ears [32], [33], [34]. Finally, MICNEST is resistant to impersonation attack, because pulses are (pseudo) randomly generated and it is difficult for third parties to generate the same pulse and impersonate a drone.

We want to point out that MicNest is a complement, not a replacement, to the existing localization solutions. The safety of commercial drones can not be overemphasized. To assist drones in precise landing, MicNest will not work alone but will cooperate with RTK and the visual marker to provide a more reliable and accurate localization service.

Challenges and contribution. Localizing drones via acoustics must tackle *three fundamental challenges*.

First, the SNR of acoustic pulses is inherently low. The transmission power of the speaker must be limited to avoid acoustic discomfort. MICNEST needs to achieve long-range localization, thus acoustic pulses experience significant attenuation. Further, background noise in many cities is intrinsically strong, around 40-75 dB SPL [35], and when airborne, drone propellers generate much acoustic interference [36], possibly up to 104 dB SPL [37].

Second, acoustic signals experience non-linear signal distortions due to Doppler effects. The severity of this effect is inversely proportional to signal speed. Compared to RF signals, the sound speed is much lower. It can be expected that acoustic pulses experience serious distortion when drones are airborne. Modern flight control loops take flight decisions at 400+ Hz, rapidly changing the drone velocity. An acoustic pulse thus experiences various degrees of Doppler effect, ultimately undergoing non-linear distortions.

Third, signal processing must withstand the latency constraint imposed by the nature of flight control loops. The latter consumes location information as one of their most critical inputs. Evidence shows that increasing the latency of location updates may represent a source of system in-

stability [3]. To make things worse, MICNEST must provide low-latency location information at a time when the system dependability is most important: during landing.

The key enabling technology behind MICNEST is a *novel pulse detector* called Matched Filter Tree (MFT) and presented in Sec. 4. The key idea is to model pulse detection as a tree search problem. We cut one pulse into multiple short segments. The time span of each segment is short enough that the drone velocity in the three dimensions can be considered as constant. Therefore, Doppler distortion within each segment is linear. We then build a search tree, where each level's nodes correspond to each segment's candidate drone velocities. For each segment, we check all possible velocities to compensate its distortion. If all segments are compensated with the right velocities, the problem of non-linear distortion is thus addressed. In addition, MFT allows us to increase pulse length on demand to further address the low-SNR problem. MFT addresses the first and second challenges above.

To enable low-latency localization, we present three techniques to accelerate tree search: *i)* tree pruning for reducing the search space by reducing the branching factor of the search tree; *ii)* correlation acceleration for reducing the time cost of each search by accelerating correlation; and *iii)* heuristic search for reducing the number of searches by exploiting past history to quickly dive towards the solution. The three techniques, presented in Sec. 5, are used jointly to address the third challenge above.

Although the techniques above significantly improve accuracy and reduce latency, they exclusively operate on single pulses. The specific problem we are to tackle, however, exhibits a further trait that may be exploited to improve overall performance: the localization process must be *continuous*. As the drone approaches the landing platform, acoustic pulses are transmitted continuously from the drone to the platform, and the localization results are similarly sent the other way around. Based on this observation, in Sec. 6 we present two further techniques: *i)* a joint audio transmission and reception scheme, and *ii)* a technique to dynamically adapt the pulse length, based on the instant SNR. Their combination allows MICNEST to further reduce overall latency while maintaining high accuracy.

We report on the performance of MICNEST in Sec. 7. Using a custom foldable landing platform we build, over a total of 60 hours of real-world experiments with multiple drones, we provide evidence of how MICNEST fulfills the requirements at stake. We demonstrate, for example, that MICNEST provides a median error of just 0.043 m below 20 m altitude, that is, where accuracy matters the most for precise landing. In an experiment covering altitudes up to 120 m, nonetheless, the absolute localization error over the distance to the landing platform is only 0.53%. The mean latency to obtain a location update at the drone is 29.7 ms, which is compatible with use in flight control loops [3], [38]. We also show how MICNEST is only marginally affected by factors it cannot control, such as background noise, and can localize multiple drones with limited degradation of accuracy. Finally, we put MICNEST through its paces and test its ability to control drone landing in a realistic setup. The results show that MICNEST can navigate the drone onto the landing platform with a 94 % success rate and an average landing error of only 4.3 cm.

2 RELATED WORK

Our work lies at the intersection of localization in mobile robotics and localization using acoustic signals.

Localization in mobile robotics. GPS is by far the most popular technique providing meter-level accuracy in outdoor settings. RTK is an improved but more expensive version of GPS and can achieve cm-level accuracy. Satellite-based systems become inaccurate or prevented from operating in urban canyons, because the surrounding buildings may reflect or block the GPS signals [16], [17], [18], resulting in the serious multipath issue or NLOS reception, respectively.

Because of these issues, several auxiliary techniques exist to localize mobile robots and especially aerial drones. AprilTags [19], [20], [21] may be used to localize drones by using a downward-facing camera to detect visual markers on the ground. Similar techniques also exist that are customized for testbed operation [39]. These systems can achieve cm-level accuracy. However, they are sensitive to lighting conditions and have a limited horizontal coverage depending on the camera field of view and distance to the marker.

Optics-based systems [24] and motion capture systems [27] can also localize drones with cm-level accuracy. However, their localization range is limited to a few meters, which makes them unsuited to enable precise landing. TrackIO [26] uses UWB tags for localizing drones. The median error is 1+ m, which may not be sufficient to support precise landing. Finally, similar to MICNEST, Rabbit [40] deploys a speaker on the drone to emit frequency-modulated continuous wave (FMCW) signals and uses a mobile phone to track the drone. Although the localization error is less than 3 cm, the range is limited to 1.5 m.

In summary, no existing localization system for mobile robots can simultaneously fulfill the three requirements in the Introduction, thus enabling precise drone landing.

Localization with acoustic signals. Acoustic signals may be used for localization indoor [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51] or outdoor [52], [53], [54], [55]. Many works utilize audible acoustic signals or actively transmit acoustic signals for indoor localization. For example, GuoGuo [42] proposes a fine-grained adaptive ToA (Time of Arrival) estimation approach to improve the location update frequency. VoLoc [46], Symphony [47], and MAVL [56] use a single microphone array to localize sources via wall reflections. Lin et al. [48] transmit the ultrasonic sound and exploit the non-linearity effect to localize devices. Some of these techniques achieve m-level accuracy, with a range limited to less than 20 m.

In outdoor localization, works exist that present the design of wireless acoustic sensor networks to achieve localization in vast areas with m-level accuracy [52], [53]. Li et al. [54] also present a machine learning technique to detect cars via acoustics and exploit the geometric information of roads to localize cars. These works deploy microphones across a vast area, while our deployment is limited to a landing platform not much larger than the drone.

Here again, the conclusion is that no existing technique can simultaneously fulfill the requirements at stake, either

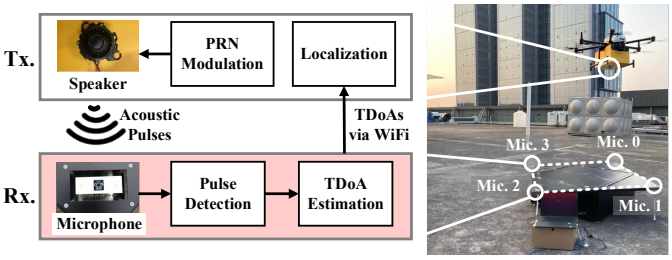


Figure 3: MICNEST system architecture.

because of limited accuracy or coverage, or due to deployment constraints dictated by the target scenario.

3 MICNEST IN A NUTSHELL

Fig. 3 shows the system architecture of MICNEST. The drone carries a speaker that transmits acoustic pulses continuously. Four distributed microphones are deployed at the corners of the landing platform to capture these pulses. MICNEST localizes the drone by localizing the speaker.

PRN modulation. We adopt Pseudo-Random Noise (PRN) modulation to generate the pulses. Specifically, let

$$\boldsymbol{s} = [s_0, s_1, \ldots, s_n, \ldots, s_{N-1}]^T \tag{1}$$

indicate the acoustic pulse for a drone, where s_n denotes the PRN code of a pulse, and N is the pulse length. We take a drone's identifier as a random seed and generate a sequence of N Gaussian random variables as the N codes of the pulse. By doing so, the pulses transmitted by different drones are independent and thus orthogonal to each other. In MICNEST, the code rate equals the sampling rate of the microphones, that is, 48 kHz. The corresponding frequency band of pulses is 0-24 kHz. For continuously localizing the drone, pulses are transmitted repeatedly.

Pulse detection and TDoA estimation. First, we need to detect the pulses from the signals recorded by the microphones. Matched filters are a standard method to detect acoustic pulses. The idea is to consider the transmitted pulse as a template and to correlate it with the received signal. The signal received by a microphone is

$$\boldsymbol{x} = \alpha \boldsymbol{s} + \boldsymbol{w}, \tag{2}$$

where α is the attenuation factor[2], and $\boldsymbol{w} = [w_0, w_1, \ldots, w_{N-1}]^T$ denotes a vector of Gaussian white noise. To detect the pulse $\boldsymbol{s}$, the matched filter correlates $\boldsymbol{s}$ with the received signal $\boldsymbol{x}$. The output is

$$y = \boldsymbol{s}^T \boldsymbol{x} = \alpha \boldsymbol{s}^T \boldsymbol{s} + \boldsymbol{s}^T \boldsymbol{w}. \tag{3}$$

By feeding a stream of $\boldsymbol{x}$ into the matched filter, we obtain a stream of correlations y. Upon observing a correlation peak in the output, we consider a pulse to be detected.

Once an acoustic pulse is detected we calculate the times when the pulse arrived at microphones (ToAs) and time difference of arrivals (TDoAs), that is, the differences in ToAs. In MICNEST, we compute TDoAs between opposite microphones on the landing platform, that is, <Mic. 0, Mic. 2> and <Mic. 1, Mic. 3> in Fig. 3. This is because opposite microphones have the largest inter-microphone distance and therefore yield the largest aperture [29]. The two TDoAs of pairs <Mic. 0, Mic. 2> and <Mic. 1, Mic. 3> are denoted as $\tau_{<0,2>}$ and $\tau_{<1,3>}$, respectively.

2. For simplicity, we assume all codes experience the same attenuation α.

Table 1: Summary of mathematical symbols.

Term	Description
T_c, $T_c^{'}$	Code periods w/o and w/ Doppler effect
G, $G^{'}$	SNR gains w/o and w/ Doppler effect
v	Drone radial velocity
c	Sound Speed
Δ	Doppler time shift experienced by PRN
N	Number of PRN codes in a pulse
L	Number of synchronized PRN codes

Localization. $\tau_{<0,2>}$ and $\tau_{<1,3>}$, are transmitted to the drone, for example, via WiFi. Based on these information, the drone can establish two hyperboloid equations.

We establish a 3D coordinate system and let the center of the landing platform be the origin. The coordinates of microphones are defined as $M_0 = (d, 0, 0)$, $M_1 = (0, -d, 0)$, $M_2 = (-d, 0, 0)$, and $M_3 = (0, d, 0)$, respectively. Given these, the two-sheeted hyperboloids oriented along the x-axis and y-axis are given by:

$$\begin{cases} |PM_0 - PM_2|^2 = 2a_0 = \text{abs}(\tau_{<0,2>} \times c) \\ |PM_1 - PM_3|^2 = 2a_1 = \text{abs}(\tau_{<1,3>} \times c) \end{cases}, \tag{4}$$

where $P = (P_x, P_y, P_z)$ is the drone coordinates, $|\bullet|^2$ is the Euclidean distance (2-norm), and c is the speed of sound.

Note that Eq. (4) is undetermined, since it provides only two constraints while drone coordinates have three unknowns, that are, P_x, P_y, and P_z. It would be possible to introduce additional constraints by using TDoAs of other microphone pairs. However, due to far-field effect, the solution to this system of equations would likely lead to oscillating behaviors, especially in the vertical direction, with the increase of drone distance from the landing platform. Differently, we estimate P_z based on sensors found aboard modern drones, such as barometers, ultrasound sensors, and downward-facing LIDARs. Similar techniques are routinely used in drone testbeds [39]. By determining P_z, we can use Eq. (4) to determine P_x and P_y.

4 TACKLING PULSE DETECTION

The *low SNR* of received pules and their *non-linear Doppler distortion* concur to make pulse detection difficult. This section elaborates on how we tackle these two challenges; then it introduces our novel pulse detector: Matched Filter Tree (MFT). As a reference through this section, Tab. 1 lists the mathematical symbols we use.

4.1 Low SNR

Because of limited transmission power, propagation attenuation, background noise, and propeller interference, the SNR of received pulses is intrinsically low.

An effective solution to address this problem is to increase pulse length, thus improving the *SNR gain* of the

matched filter. To understand the SNR gain as seen by the matched filter, we compare the SNRs of the received signal $\boldsymbol{x}$ and the output y. Considering Eq. (2), the SNR of $\boldsymbol{x}$ is

$$SNR_x = \frac{E\left[|\alpha \boldsymbol{s}|^2\right]}{E\left[|\boldsymbol{w}|^2\right]} = \frac{|\alpha|^2 \boldsymbol{s}^T \boldsymbol{s}}{E\left[\sum |w_n|^2\right]} = \frac{|\alpha|^2 \boldsymbol{s}^T \boldsymbol{s}}{N\sigma^2}, \quad (5)$$

where σ^2 is the noise variance. Similarly, the SNR of y is

$$SNR_y = \frac{E\left[|\alpha \boldsymbol{s}^T \boldsymbol{s}|^2\right]}{E\left[|\boldsymbol{s}^T \boldsymbol{w}|^2\right]} = \frac{|\alpha|^2 \boldsymbol{s}^T \boldsymbol{s} \cdot \boldsymbol{s}^T \boldsymbol{s}}{\boldsymbol{s}^T E\left[\boldsymbol{w}\boldsymbol{w}^T\right] \boldsymbol{s}} = \frac{|\alpha|^2 \boldsymbol{s}^T \boldsymbol{s}}{\sigma^2}, \quad (6)$$

where $E\left[\boldsymbol{w}\boldsymbol{w}^T\right]$ is the covariance matrix of $\boldsymbol{w}$, which is $\sigma^2 \boldsymbol{I}$. The SNR gain of the matched filter is thus given by

$$G = \frac{SNR_y}{SNR_x} = N. \quad (7)$$

Eq. (7) leads to a fundamental insight: *the SNR gain G equals the pulse length N*. This means that, at least in principle, we may trade time for a higher SNR. Unfortunately, doing so backfires in the presence of non-linear Doppler distortion, as explained next.

4.2 Non-Linear Doppler Distortion

The Doppler effect introduced by drone dynamics may seriously distort the acoustic pulses. The severity of Doppler effect is generally inversely proportional to signal speed. Due to the low speed of acoustic pulses, they heavily suffer from such distortion.

Code desynchronization. Doppler effect gradually makes the received pulses misaligned with the transmitted original pulses in the time domain, and ultimately desynchronize the received codes with the transmitted ones.

Consider the continuous-time waveform of the transmitted pulse $\boldsymbol{s}$ which is given by[3]

$$s(t) = \sum_{n=0}^{N-1} s_n \cdot \text{rect}(\frac{t - nT_c}{T_c}), 0 \le t \le NT_c. \quad (8)$$

Here, rect(t) is the rectangular function that is 0 outside the interval $[0, 1)$ and 1 inside of it, whereas T_c denotes the code period, that is, $1/48kHz$ s in our implementation.

In the presence of Doppler effect, the codes of the received pulse expand or compress in time. The period of these codes can be calculated as [57]

$$T_c^{'} = \left(1 - \frac{v}{c}\right) T_c, \quad (9)$$

where v denotes the drone velocity[4]. Given this, the received pulse distorted by Doppler effect is

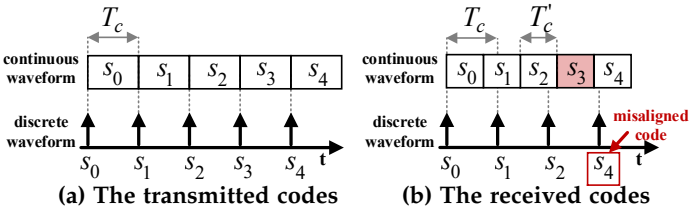
$$s^{'}(t) = \sum_{n=0}^{N-1} s_n \cdot \text{rect}(\frac{t - nT_c^{'}}{T_c^{'}}), 0 \le t \le NT_c^{'}. \quad (10)$$

However, the receiver simply samples the pulse with the original code period T_c. The discrete-time waveform of the received pulse is

$$s_n^{'} = s^{'}(t) \cdot \delta(nT_c), n = 0, 1, \ldots, \quad (11)$$

3. For simplicity, we use the Zero-Order Hold (ZOH) to model the Digital-to-Analog Converter (DAC).

4. Unless otherwise specified, drone velocity refers to radial velocity of drone with respect to the ground port. The velocity is negative if the drone moves towards the ground port.



(a) The transmitted codes **(b) The received codes**

Figure 4: Illustration of code desynchronization.

where $\delta(t)$ is the Dirac delta function. Note that the period of received codes is $T_c^{'}$ while the sampling period is T_c. This means that each sampled code is shifted in time by

$$\Delta = T_c - T_c^{'} = \frac{v}{c} \cdot T_c. \quad (12)$$

To make things worse, the time shift accumulates over multiple samples. Consider Fig. 4 as an example. It can be calculated that when $n \ge L = \lfloor c/v \rfloor$, the accumulated time shift that $s_n^{'}$ experiences is larger than the original code period T_c. This means that the codes received hereafter are desynchronized with the transmitted codes, therefore the following holds:

$$\begin{cases} s_n^{'} = s_n & \text{if } n < L \\ s_n^{'} \ne s_n & \text{if } n \ge L \end{cases} \quad (13)$$

Impact of code desynchronization. The derivation of the SNR gain in Eq. (7) assumes that the transmitted codes, that is, the template, are synchronized with the received codes. This assumption may not hold due to code desynchronization. In such a case, the SNR gain of the matched filter can be re-written as

$$G^{'} = \frac{SNR_y^{'}}{SNR_x^{'}} = \left(\frac{|\alpha|^2 \boldsymbol{s}^T \boldsymbol{s}^{'} \cdot \boldsymbol{s}^T \boldsymbol{s}^{'}}{\sigma^2 \boldsymbol{s}^{'T} \boldsymbol{s}^{'}}\right) / \left(\frac{|\alpha|^2 \boldsymbol{s}^{'T} \boldsymbol{s}^{'}}{N\sigma^2}\right)$$
$$= N\left(\frac{\boldsymbol{s}^T \boldsymbol{s}^{'}}{\boldsymbol{s}^{'T} \boldsymbol{s}^{'}}\right)^2, \quad (14)$$

where $\boldsymbol{s}^{'}$ denotes the received pulse $[s_0^{'}, s_1^{'}, \ldots, s_{N-1}^{'}]^T$.

Based on Eq. (13), if the number of codes n is less than L, then $\boldsymbol{s}^{'} = \boldsymbol{s}$. In this case, there is no code desynchronization, and $G^{'} = G = N$. Instead, if $n \ge L$, then only the first L codes of $\boldsymbol{s}^{'}$ equals those of $\boldsymbol{s}$ and the remaining codes are desynchronized. Therefore,

$$G^{'} = N\left(\frac{\sum_{n=0}^{L-1}(s_n^{'})^2 + \sum_{n=L}^{N-1} s_n s_n^{'}}{\sum_{n=0}^{N-1}(s^{'})^2}\right)^2 = N\left(\frac{\sum_{n=0}^{L-1}(s_n^{'})^2}{\sum_{n=0}^{N-1}(s_n^{'})^2}\right)^2$$
$$\approx N(\frac{L}{N})^2 = \frac{L^2}{N}, \quad N > L. \quad (15)$$

Here, the reason for $\sum_{n=L}^{N-1} s_n s_n^{'} = 0$ and for the approximation in the second line is that codes in our pulse are a sequence of independent pseudo-random Gaussian variables [58]. As a result, once the pulse length N exceeds L, the SNR gain degrades with the increase of pulse length.

We are now facing a catch-22 situation. Eq. (7) suggests that a long pulse length helps mitigate the low-SNR problem. Conversely, Eq. (15) indicates that the maximum pulse length that can improve SNR gain is $L = \lfloor c/v \rfloor$, whose value is upper-limited by the (low) speed of acoustic signals c. Let us consider a concrete example to illustrate the problem.

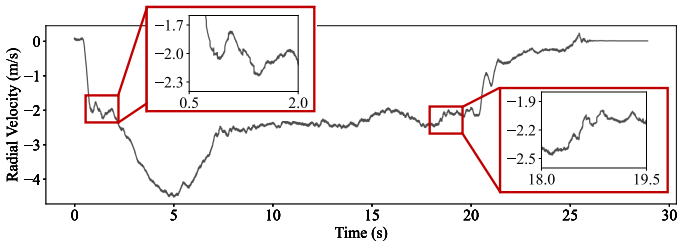


Figure 5: A drone's radial velocity during landing.

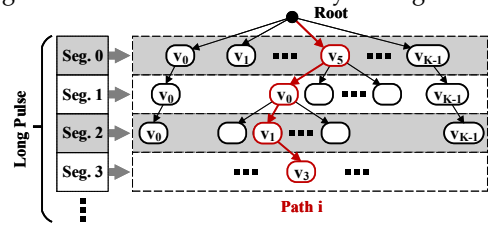


Figure 6: Illustration of a search tree.

Suppose the sound speed is 343 m/s and the drone speed is 6 m/s. Thus, a complete code desynchronization will occur after $L = \lfloor 343/6 \rfloor = 57$ codes, as per Eq. (13). On the other hand, as we indicate in Sec. 7, the pulse duration should be at least 50 ms so as to localize a long-range drone, thus the corresponding pulse length N is 2400 at a sampling rate 48 kHz. The huge gap between $N = 2400$ and $L = 57$ indicates that Doppler distortion significantly hinders addressing the low-SNR problem by increasing the pulse length.

Physical drone dynamics. Should the drone velocity be constant, the corresponding Doppler distortion would be linear. If we were in this situation, we might simply determine what drone velocity compensates the Doppler effect. This is precisely the idea of the matched filter bank [59], [60].

In practice, however, the drone velocity is not constant, leading to non-linear distortion. As an example, Fig. 5 plots the radial velocity of a drone when it is landing automatically from a 50 m altitude, based on location and velocity information obtained by an RTK system in an open area and the on-board IMU. The drone velocity fluctuates rapidly during landing in response to commands from the on-board flight control loops [38], which rapidly change the drone motion to maintain stable flight on a predetermined route.

Flight control loops operate at 100Hz-32kHz [61], [62], [63]. This means that a drone may change its velocity at sub-10-ms scales. On the other hand, Eq. (7) shows that in order to take advantage of the SNR gain of the matched filter, we should increase pulse length. Our real-world experimental evaluation, reported in Sec. 7, indicates pulse duration should be no less than 50 ms to localize the drone robustly. We may thus expect that multiple motions occur within the duration of one pulse. Therefore, pulses undergo non-linear distortion and different codes experience different degrees of Doppler effect, hindering detection.

4.3 Matched Filter Tree

We present a dedicated pulse detector, called Matched Filter Tree (MFT), to detect a low-SNR pulse subject to non-linear distortion. The key idea we exploit is to model pulse detection as a tree search.

Intuition. Fig. 6 illustrates the idea. We split one pulse into M equal *segments* denoted as Seg. 0, 1, ... with segment length $N_{seg} = N/M$. The segments are short enough that the drone velocity can be considered constant within the duration of a segment. Therefore, each segment experiences a *linear* Doppler distortion.

We build a search tree where the nodes at each level correspond to the possible drone velocities during the transmission of a segment. For each segment, we consider the K possible velocities to compensate the Doppler shift it experiences. Ideally, if all segments are compensated with the correct velocities, the new pulse spliced by the compensated segments restores its code synchronization with the received pulse, eliminating the non-linear distortion. Sec. 5 discusses the setting of the parameters at hand, including the choice of M, and the search resolution for velocity.

Searching the solution. Let velocities $< v^{(0)}, v^{(1)}, \ldots, v^{(M-1)} >$ be one possible combination of candidate velocities, for example, corresponding to path i in Fig. 6, where $v^{(m)}$ denotes the candidate velocity for Seg. m. We perform the following steps to check whether the candidate velocities compensate the non-linear distortion:

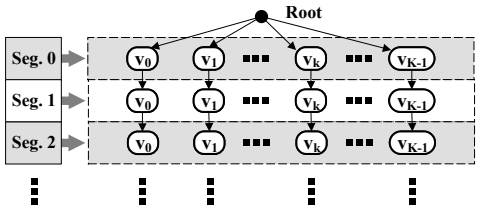
- **S1 (compensation):** for each velocity $v^{(m)}$, we estimate the Doppler shift that Seg. m suffers as $(v^{(m)}/c) \cdot T_c$, according to Eq. (12)); we compensate this Doppler shift by resampling this segment with spacing $(c - v^{(m)})/c$.
- **S2 (concatenation):** we concatenate the resampled versions of M segments into a new pulse, denoted as $s_i^{'}$.
- **S3 (correlation):** we take the new pulse $s_i^{'}$ as a new template and correlate it with the received signal.

If velocities $< v^{(0)}, v^{(1)}, \ldots, v^{(M-1)} >$ along path i match the actual drone velocities, the new pulse $s_i^{'}$ is again synchronized with the received pulse; thus it has maximum correlation with the received signal because the non-linear distortion is minimized. Therefore, the problem of detecting pulses corresponds to finding a solution path in the search tree that can *minimize the non-linear distortion.*

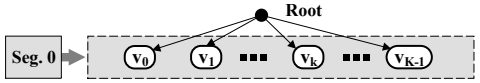
A straightforward solution is exhaustive search, that is, visiting every path of the search tree. For each such path, we use the velocities along the path to compensate the Doppler shift of the pulse, as per **S1** and **S2**, and to calculate the correlation, as per **S3**. After visiting all paths, we select the path whose corresponding correlation has the maximum value; the maximum correlation value means that the non-linear distortion is minimized.

The processing overhead of an exhaustive search is, however, unacceptable for MicNest, because of the low-latency requirement discussed earlier. The search space indeed grows exponentially with the number of segments M. Given the computational complexity of the correlation operation in **S3**, it is nearly impossible to visit every path and return a solution at low latency. Note that it is also infeasible to search for the solution path incrementally or greedily, that is, choose the candidate velocity that can maximize the correlation at each level of the tree. This is because the length of each segment is so short that the SNR gain of segment-audio correlation is immaterial. The noise typically dominates the correlation. Sec. 5 explains how we accelerate the tree search.

With the help of MFT, we can compensate for the non-linear distortion in the received pulses. This means that the problem of code desynchronization can be tackled, and



(a) Detecting linearly distorted pulses.



(b) Detecting high SNR pulses.

Figure 7: Two specific cases MFT can be narrowed down to.

we can avoid the dilemma that a long pulse length can undermine the SNR gain. In other words, it is safe for us to choose a longer pulse on demand to boost the SNR of the MFT's outputs (see Eq. (7)). In Sec. 7, we study the pulse length that MICNEST demands.

4.4 Other MFT Uses

Above, we show that MFT is feasible to detect pulses suffering from both *low SNR* and *non-linear distortion*, which is indeed a challenging task. In fact, MFT can also be used for detecting pulses in other cases, as shown in Fig. 7:

- **Detecting linearly distorted pulses:** Fig. 7(a) shows this case. When the received pulses suffer from linear distortion[5], all segments experience the same degree of distortion. Based on this, we remove the unfeasible search paths and reduce the search space from an exponential one to a linear one. This kind of MFT is equivalent to matched filter bank [59], [60], as used in GPS receivers.
- **Detecting high SNR pulses:** A high SNR of received pulses can lower the requirement for the SNR gain of MFT. This means that the pulse length can be reduced, and correspondingly the depth of the search tree can be reduced. As illustrated by Fig. 7(b), when the SNR is sufficiently high, the search tree can be chopped to a single layer, reducing the search space.

MFT therefore enjoys the flexibility to adapt to detection tasks according to signal quality. Meanwhile, MFT is an extension to a matched filter. By replacing the template of MFT, MFT can detect not only PRN pulses used by MICNEST, but also other types of signals, such as FMCW signals and OFDM signals. We believe MFT is a general detection tool and may be equally applied to other signal processing tasks.

5 TACKLING THE LATENCY CHALLENGE

We present three methods to accelerate the tree search.

5.1 Tree Pruning

We note that the drone velocity does not change abruptly: the velocity of the next segment is unlikely to considerably deviate from that of the current segment. This observation

5. In addition to Doppler effect, many hardware imperfections can also introduce linear distortion, such as carrier frequency offset (CFO) and sampling frequency offset (SFO).

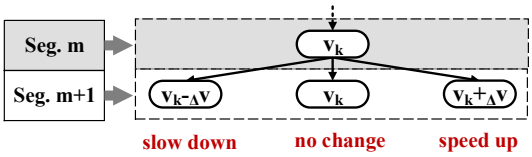


Figure 8: To prune the search space, the branching factor of the search tree is aggressively reduced to three.

allows us to reduce the branching factor of the tree, that is, the number K of candidate velocities. We expect this to abate the processing overhead.

In our design, we aggressively reduce the branching factor to three, as shown in Fig. 8. Suppose v_k is chosen as the velocity of the m-th segment. Only three candidate velocities exist for the next segment, that is $v_k - \Delta v$, v_k, and $v_k + \Delta v$, where Δv denotes the search resolution. By doing so, the search space is significantly reduced.

However, to safely prune the tree, we must satisfy the following two constraints.

- **C1:** ensure that the solution path is within the pruned search space. That is

$$\Delta v \geq N_{\text{seg}} \cdot T_c \cdot a_{\max}, \tag{16}$$

  where $a_{\max}$ denotes the maximum drone acceleration. Intuitively, this requirement expects a large Δv so that the candidate velocities along the considered paths can catch up with the rapid change of drone velocity.
- **C2:** ensure that the search resolution Δv is fine enough that the Doppler shift of each segment can be compensated. In fact, each candidate velocity is a numerical representation of the actual velocity with resolution Δv. Due to numerical error, the Doppler shift may not be completely compensated. According to Eq. (12), the accumulated Doppler shift of segment with length N_{seg} can be calculated as $\sum_{n=0}^{N_{\text{seg}}-1} \frac{v[n]}{c} \cdot T_c$, where $v[n]$ denotes the drone velocity during code s_n. After compensating the Doppler shift with v_k, the residual Doppler shift δ_{shift} is $\delta_{shift} = \sum_{n=0}^{N_{\text{seg}}-1} \frac{v[n]-v_k}{c} \cdot T_c$. Given the resolution Δv, the upper bound of δ_{shift} is $N_{\text{seg}} \cdot \frac{\Delta v}{2c} \cdot T_c$. To avoid introducing code asynchronization additionally, the residual Doppler shift δ_{shift} should be less than T_c:

$$N_{\text{seg}} \cdot \frac{\Delta v}{2c} \cdot T_c < T_c. \tag{17}$$

Obviously, constraints **C1** and **C2** are conflicting since **C1** expects a larger Δv while **C2** expects a smaller one.

We notice that segment length N_{seg} plays a key role in fulfilling both **C1** and **C2**. N_{seg} needs to be small enough that during each segment the drone velocity can be assumed as constant. This assumption also contributes to the residual Doppler shift because the actual velocity is not constant. The resulting maximum possible deviation between the actual velocity and the candidate velocity is $N_{\text{seg}} \cdot T_c \cdot a_{\max} + \Delta v$. Therefore, Eq. (17) should be modified as

$$N_{\text{seg}} \cdot \frac{N_{\text{seg}} \cdot T_c \cdot a_{\max} + \Delta v}{2c} \cdot T_c < T_c. \tag{18}$$

In a nutshell, we should satisfy the constraints of Eq. (16) and Eq. (18) before aggressively pruning the tree to three

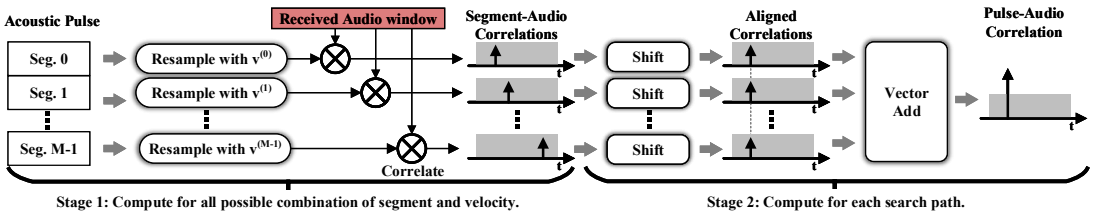


Figure 9: To accelerate the operation of correlation, we take two stages to do so: We first calculate and cache all possible segment-audio correlations. During the tree search, we efficiently compute one pulse-audio correlation by shifting and adding up the segment-audio correlations elementwisely.

branches. In our implementation, $T_c = 1/48k$ s. The maximum acceleration of our drones $a_{\max}$ is 8 m/s^2. Given these, we set N_{seg} to 240 and Δv to 0.1 m/s, ensuring a sufficient search space and providing a promising search resolution.

5.2 Correlation Acceleration

Correlation is the most time-consuming operation during tree search (**S3** in Sec. 4.3). Here, we reduce its time cost.

Let us define $Cor_{<\boldsymbol{v_0},\boldsymbol{v_1}>}[\tau]$ as the correlation function between vectors $\boldsymbol{v_0}$ and $\boldsymbol{v_1}$, and $\tilde{\boldsymbol{s}}_{\text{seg}}^{(m)}$ as the compensated version of Seg. m, and $N_{\text{seg}}^{(m)}$ as the length of $\tilde{\boldsymbol{s}}_{\text{seg}}^{(m)}$.

We find that the correlation between the compensated pulse $\tilde{\boldsymbol{s}}$ and the received audio $\boldsymbol{y}$ can be calculated by summing the M compensated segment's correlation with the received audio $\boldsymbol{y}$:

$$Cor_{<\tilde{\boldsymbol{s}},\boldsymbol{y}>}[\tau] = \sum_{m=0}^{M-1} Cor_{<\tilde{\boldsymbol{s}}_{\text{seg}}^{(m)},\boldsymbol{y}>}[\tau + L^{(m-1)}], \tag{19}$$

where $L^{(m)}$ is defined as $\sum_{k=0}^{m} \tilde{N}_{\text{seg}}^{(k)}$. We can rewrite the compensated pulse $\tilde{s}$ as a concatenation of M compensated segments:

$$\tilde{\boldsymbol{s}}[n] = \sum_{m=0}^{M-1} \text{bool}[L^{(m-1)} \le n < L^{(m)}] \cdot \tilde{\boldsymbol{s}}_{\text{seg}}^{(m)}[n - L^{(m-1)}], \tag{20}$$

where bool[*condition*] is a boolean function that equals 1 if *condition* is true and 0 otherwise. Substituting Eq. (20) into the definition of correlation leads to Eq. (19):

$$\begin{aligned}
Cor_{<\tilde{s},y>}[\tau] &= \sum_{n=0}^{\tilde{N}-1} y[n+\tau]\tilde{s}[n] \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} y[n+\tau]\mathbf{1}[L^{(m-1)} \le n < L^{(m)}] \cdot \tilde{s}_{seg}^{(m)}[n - L^{(m-1)}] \\
&= \sum_{m=0}^{M-1}\sum_{n=L^{m-1}}^{L^m} y[n+\tau] \cdot \tilde{s}_{seg}^{(m)}[n - L^{(m-1)}] \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{\tilde{N}_{seg}^{(m)}} y[n+\tau+L^{(m-1)}] \cdot \tilde{s}_{seg}^{(m)}[n] \\
&= \sum_{m=0}^{M-1} Cor_{<\tilde{s}_{seg}^{(m)},y>}[\tau + L^{(m-1)}].
\end{aligned} \tag{21}$$

Eq. (19) reveals a key fact that can be exploited to accelerate correlation operation. That is, the pulse-audio correlation $Cor_{<\tilde{s},y>}$ can be decomposed into multiple segment-audio correlations. In other words, if segment-audio correlations are available, pulse-audio correlation can be calculated by adding M vectors, that is, M segment-audio correlations.

Given this, we apply a two-step process, shown in Fig. 9:

- **Stage 1:** this stage is performed before the tree search. We calculate and cache all possible segment-audio correlations. Specifically, for each Seg. m and for each candidate velocity $v^{(m)}$, we resample this segment with $v^{(m)}$ and then correlate it with a window of received signals. The resulting segment-audio correlation is then saved to a lookup table with key $< m, v^{(m)} >$.
- **Stage 2:** this stage is performed during the tree search. For one search path, we retrieve all required segment-audio correlations from the lookup table. For each Seg. m, we shift its segment-audio correlation by L^{m-1}. We then add up M shifted segment-audio correlations elementwisely.

Note that the total number of segment-audio correlations is not an exponential function of segment number M, but a linear function, that is, $\sum_{m=0}^{M-1}(2m+3)$. This is because after reducing the branching factor to three (see Sec. 5.1), Seg. m has only $2m+3$ candidate velocities. So Stage 1 can be finished in a short time (about 3.1 ms)

Also note that, the vector add in Stage 2 can be efficiently parallelized. We take advantage of native NVIDIA CUDA kernel[6] to further accelerate Stage 2. In our implementation, searching one tree path takes only 5.3 μs on the NVIDIA RTX 3070, on average.

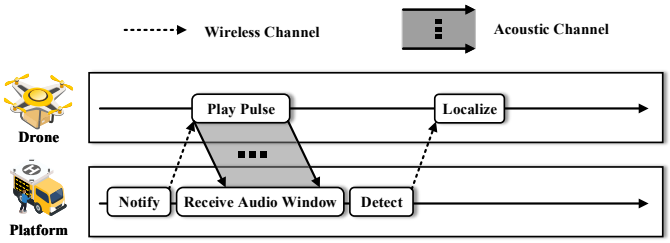
5.3 Heuristic Search

Instead of visiting all tree paths in a brute-force way, we adopt a heuristic method to reduce the total visit count.

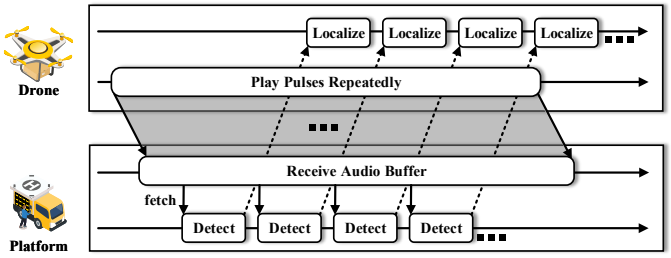
Our method is similar to Monte-Carlo Tree Search (MCTS) [64], [65], [66]. Our insight is: for each search path, its corresponding maximum correlation values contain the useful information, which can be exploited to guide towards the solution path in the search tree. The path that has a larger correlation value is more likely to be closer to the solution path, and thus the nodes along this path are more promising to be the nodes of the solution path. Therefore, we pay more attention to nodes that look promising, so as to avoid traversing the search tree exhaustively.

Our method consists of repeated rounds. For each round, we perform three procedures:

6. Native CUDA kernel also allows us to perform shift operation efficiently. This is because by passing different memory offsets of vectors to the CUDA kernel, we can implicitly perform shift operation without memory copy.



(a) On-Demand localization.



(b) Pipelined localization.

Figure 10: Localization modes.

- **Batch path selection:** we first generate a batch of tree paths in the search tree. The batch size is denoted as B. During the generation of each path, we start from the root node and choose the next child with the highest Upper Confidence Bound (UCB) value [65], defined as

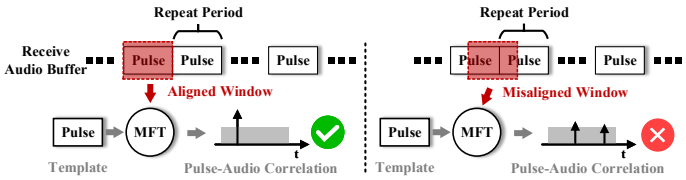
$$\text{UCB(node } j) = \text{score}^j + K_{ucb}\sqrt{\frac{\log N_{\text{vis}}}{N^j_{\text{vis}}}}, \qquad (22)$$

  where score^i denotes the *score* of node j, explained later, N_{vis} is the total number of paths that has been visited, N^j_{vis} denotes the number of times that node j have been selected, and K_{ucb} is an empirical parameter that used to trade off between *exploration* and *exploitation* [66].
- **Batch path evaluation:** For each path from the B generated tree paths, we compute its corresponding correlation using the accelerated method introduced in Sec. 5.2. We treat each path as a comparison game. The paths whose correlations are top B_0 largest are regarded as *win* ($B_0 < B$), and other paths as *lose*.
- **Backpropagation:** we then use the game results of B paths to update the score of nodes. The score of node j is simply defined as the winning rate of paths that pass through it: $\text{score}^j = N^j_{win}/N^j_{\text{vis}}$, where N^j_{win} denotes the winning times of paths passing through node j.

It can be expected that, as the number of rounds grows, the nodes of the solution path will be visited more and more frequently as their node score^j are gradually growing. Therefore, our method will converge to the solution path. In our implementation, the batch size B is empirically set to 20, B_0 is 1, and K_{usb} is 0.4. The total visit count is limited to 5000 (see Sec. 7.3). By doing so, the time cost of detecting one pulse is less than 30 ms.

6 Continuous Drone Localization

To carefully navigate the drone onto the landing platform, we further need to detect a continuous sequence of pulses so as to localize the drone continuously with high precision. This aspect offers avenues for further optimizations.



(a) Aligned window. (b) Misaligned window.

Figure 11: Toy example of continuous pulse detection.

6.1 Localization Mode

Our key design principle for continuous localization is to *reduce the interaction between the drone and the landing platform as much as possible*.

The alternative to continuous localization would be operating on-demand, as illustrated in Fig. 10(a): the platform asks the drone via WiFi to transmit one PRN pulse and consequently records a window of signals. Next, the platform uses MFT to detect pulses and then calculate TDoAs, which are transmitted back to the drone for localization. The on-demand operation would suffer for multiple reasons:

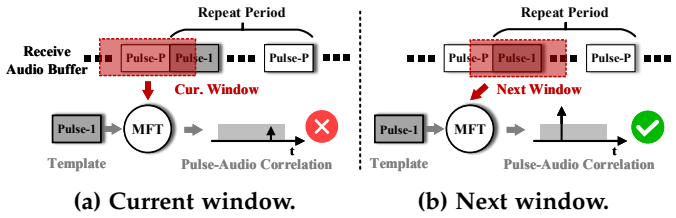
- **Acoustic spectral leakage:** whenever the speaker is switched on/off, the signal energy leaks to undesired frequency components [67]. This not only distorts the signal additionally, but also makes sounds noticeable since the leaked components may fall into the frequency range of the human ear.[7]
- **Additional waiting latency:** given fluctuating wireless latency, the signal window should be somewhat larger than the PRN pulse length, so to ensure the entire pulse can be recorded. The landing platform would need to increase waiting times, increasing latency.
- **Limited update rate:** for each round of localization, MicNest must pay for extra communication latency, besides the above-mentioned additional waiting times.

To tackle the above issues, MicNest adopts a pipelined localization mode, as shown in Fig. 10(b). The main feature is that signal transmission is fully decoupled from signal reception. The drone simply continuously repeats the pulse without spectral leakage and without any interaction with the landing platform. The platform can immediately launch pulse detection upon arrival of an entire pulse, which significantly compresses the entire process over time, compared to the on-demand mode. The pipelined localization mode, however, requires a joint design of audio transmission and reception schemes, as described next.

6.2 Audio Transmission and Reception

Let us consider Fig. 11 as an example. As the drone simply repeats its PRN pulses continuously, they are saved at the platform into a received signal buffer sequentially. The platform then repeatedly performs the following procedures: it

7. These two problems may be mitigated by applying a Hamming or Blackman-Harris window to the audio pulse, however, at the cost of sacrificing pulse strength and quality.



(a) Current window. (b) Next window.

Figure 12: To search for the target pulse-1, we set the template of MFT to the pulse-1, and try to detect it window by window from the receive audio buffer.

fetches a window of audio signals from the buffer, then uses MFT to detect the corresponding PRN pulse.

Ideally, the signal window should align with the received pulse, as in Fig. 11(a). Thus, MFT is expected to produce the pulse-audio correlation with a single correlation peak, whose index indicates the ToA of the pulse. However, the most general case is that the window is misaligned with the pulse, as shown in Fig. 11(b). In this case, the signal window contains two parts from two adjacent identical pulses. Since both parts are correlated with the template, multiple correlation peaks will be generated, resulting in the following two crucial problems:

- **Peak ambiguity:** the ambiguity is two-fold. *i)* since there are multiple candidate peaks, MICNEST cannot determine which peak to choose to calculate ToA; *ii)* as explained in Sec. 5.3, we exploit the correlation values of history paths to guide MFT to find more promising search paths, but multiple peaks confuse MFT, slowing down the tree search.
- **Reduced SNR gain:** Eq. (7) indicates that the SNR gain is proportional to the number of the correlated codes. The misaligned window reduces the number of correlated codes, and thus reduces the SNR gain. In other words, the misalignment distributes the signal energy over multiple peaks, rather than concentrating it in a single peak.

The crux of the problems is the setting of the pulse repetition period and window size: to avoid introducing peak ambiguity, the same PRN pulse should not appear multiple times within a window, not even partly. This requires the pulse repetition period to be larger than the window size. We address this issue by making the drone repeat a group of *different* PRN pulses, rather than a single one, which therefore increases the pulse repetition period several times.

To avoid reducing the SNR gain, the signal window should be extended to reserve some guard interval, so that the entire pulse corresponding to the template case can be captured even when the two are misaligned. We set the window size to twice the target pulse length, and the step size used for sliding the window is set to the pulse length. This ensures that the target pulse is captured in a window.

Fig. 12 gives an example of how the platform detects the pulse with the design above. Suppose the drone transmits P different PRN pulses repeatedly ($P \geq 3$). The landing platform initially chooses pulse-1 as a target to detect. It then sets the template of MFT to pulse-1, and attempts to detect pulse-1 from the signal buffer, window by window. In case that the current signal window covers pulse-1 only partially, the detection fails as shown in Fig. 12(a). However, our design ensures that pulse-1 is captured by the next window and thus detected by MFT, as shown in Fig. 12(b)[8].

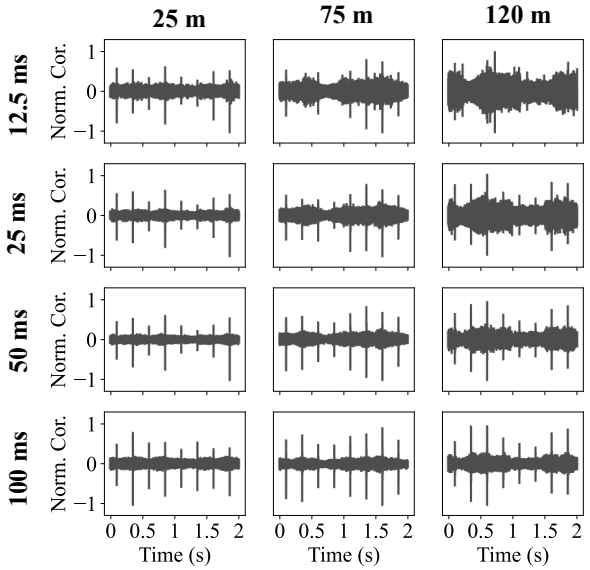


Figure 13: Impact of pulse length and altitude on pulse-audio correlation.

Note that the P pulses and their transmission sequence are known to the landing platform. Once pulse-1 is detected, the platform knows what pulses arrive next, and when. The platform can consequently select the right templates to detect pulses from given signal windows.

6.3 Adaptive Pulse Length

The design above offers an additional knob to the landing platform, as we can adaptively tune the pulse length at runtime, especially as a function of the instantaneous SNR.

Fig. 13 experimentally shows how information on altitude and length of PRN pulses impact the pulse-audio correlation. We make the drone hover at altitudes of 25 m, 75 m, and 120 m. At each altitude, the drone repeatedly transmits acoustic pulses of 250 ms length. We correlate the received audio with pulse templates with variable lengths: 12.5 ms, 25 ms, 50 ms, and 100 ms. When the pulse length is only 12.5 ms, we still observe distinguishable correlation peaks at 25 m altitude, but the correlation is almost entirely superseded by the noise floor when the altitude is increased to 120 m, indicating the drone altitude has a greater impact on the instantaneous SNR. Therefore, the lower the altitude, the shorter the pulse length required by MICNEST. This offers an opportunity to decrease transmission delay.

In our design, the landing platform continuously estimates the signal quality and accordingly tunes the pulse (template) length. We set the maximum pulse length to 2400, corresponding to 50 ms at 48 KHz sample rate. This is reasonable since a pulse with 2400 length can still provide reasonable detection results at 120 m, as shown in Fig. 13,

8. Note that, a possible issue is that if the platform fails to detect pulse-1, it must wait for another pulse repetition period. To alleviate this problem, we limit the pulse repetition period to 250 ms.

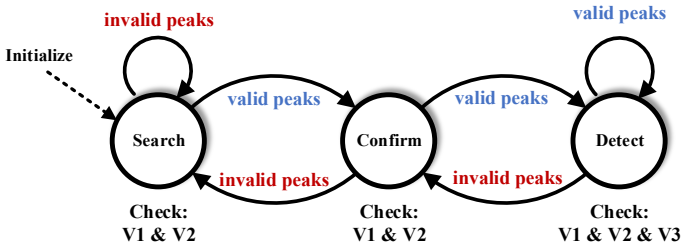


Figure 14: Finite state machine for pulse search and detection.

striking a balance between detection accuracy and transmission delay. The minimum pulse length is empirically set to 600, corresponding to 12.5 ms at 48 KHz sample rate.

At a drone commences the approach to the landing platform, the latter uses the template with the maximum length to search for the first pulse. For each computed pulse-audio correlation, we calculate the peak-to-noise-floor ratio (PNFR), which is empirically defined as the ratio between the correlation peak value and the noise floor. We decrease the next template length by 120, corresponding to 2.5 ms at 48 KHz sample rate, when PNFR $>$ 3.5; increase the next template length by the same quantity when 1.5 $<$ PNFR $<$ 2.5; and resort to maximum pulse length if PNFR $<$ 1.5 or the detected peak fails the validity checks. We leave the length unchanged otherwise.

In summary, the template length for the t-th detection N_t is

$$N_t = \begin{cases} 2400, & \text{if } t = 1 \text{ or PNFR} < 1.5 \\ & \text{or invalid peaks;} \\ \max(N_{t-1} - 120, 600), & \text{if PNFR} > 3.5; \\ \min(N_{t-1} + 120, 2400), & \text{if PNFR} < 2.5; \\ N_{t-1}, & \text{otherwise.} \end{cases} \tag{23}$$

Note that, in addition to PNFR, MICNEST performs three additional validity checks on detected peaks so as to minimize false positives. The valid peaks should meet all of the following requirements.

- **V1: Reasonable TDoA.** The TDoAs calculated by the peak indexes should not be larger than the maximum possible value, corresponding to the inter-microphone distance divided by the sound speed.
- **V2: Well-conditioned problem.** The parameter matrix of Eq. (4) should not be ill-conditioned. We empirically set the threshold for the condition number to 500.
- **V3: Promising index.** If the current pulse is successfully detected, we may predict the index of the next peak index to some degree. Meanwhile, the Doppler effect might slightly bias the actual peak index[9]. MICNEST checks the time interval of the predicted and detected peak. A time interval less than $\frac{v_{max}}{c} T_c N_t + 1$ is acceptable, where v_{max} is the maximum drone speed.

6.4 Integration

Fig. 14 illustrates the operation of pulse search and pulse detection. The process unfolds through three main states: *search*, *confirm*, and *detect*.

Search. After initializing the necessary components, MICNEST searches the audio buffer for the first 50 ms of PRN pulses through a sliding window. In this state, the window size and the sliding size of the audio window are set to 100 ms and 50 ms, respectively. For each correlation peak detected within each window, we perform validity checks **V1** and **V2**. Note that we cannot perform **V3** checks because they require information about previously detected peaks, which is not available at this stage. If the checks are valid, the processes transitions to the *confirm* state.

Confirm. The main purpose of this state is to disambiguate the *search* and *detect* states. The validity checks are empirical in nature; therefore, the peaks that pass these checks are not necessarily correct. MICNEST now performs all validity checks **V1**, **V2**, and **V3** to reduce false positives. In practice, MICNEST may also miss one pulse during continuous pulse detection. In this case, we give MICNEST a single chance to detect the pulse again. We transition back to the *search* state only if the pulse detection fails again. The pulse length is set to 50 ms and the sliding size is equal to the pulse length. We set the size of the audio window to be only slightly larger than the pulse length, around 55 ms, since we can predict when the target pulse arrives.

Localize. Ideally, MICNEST should be able to detect pulse sequentially at this stage. Each peak that passes the validity check (**V1**, **V2**, and **V3**) is transmitted to the drone for the final localization. We apply here the adaptive length scheme to tune the pulse length N_t, as seen in Eq. (23). The window size equals the pulse length plus 5 ms. The sliding size of the next audio window S_{t+1} is tuned based on the detected peak index P_t to track the next pulse, that is,

$$S_{t+1} = P_t + N_t - 240. \tag{24}$$

where 240 represents the number of samples for 5 ms audio.

7 EVALUATION

Our evaluation of MICNEST is composed of five parts and entirely based on real-world experiments. Following a description of the implementation we use and of the experimental setting in Sec. 7.1, we report in Sec. 7.2 on the crucial performance metrics we target: localization accuracy and processing latency. We proceed by investigating the influence of key system parameters in Sec. 7.3. Further, we study in Sec. 7.4 the impact on MICNEST of external factors, such as drone speed, background noise, and ambient temperature. We demonstrate that MICNEST can robustly and accurately guide the drone to the landing platform in Sec. 7.5. We conclude in Sec. 7.6 by demonstrating MICNEST's ability to concurrently localize multiple drones.

The results we collect across a total of 60 flight hours lead us to six key conclusions:

1) MICNEST provides a range up to 120 m and attains *cm-level* accuracy as the drone approaches the platform;
2) The rate of location updates returned by MICNEST is *compatible with use in flight control loops*;
3) The performance of MICNEST *improves as the drone approaches the platform*, that is, where it matters the most;
4) MICNEST is *marginally affected* by factors it cannot control, such as drone speed, noise, and temperature;

9. Recall,Eq. (9) shows that the Doppler effect expands or compresses the received pulse, thus deviating the next peak index.

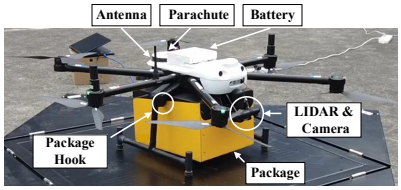


Figure 15: The delivery drone made by Meituan.

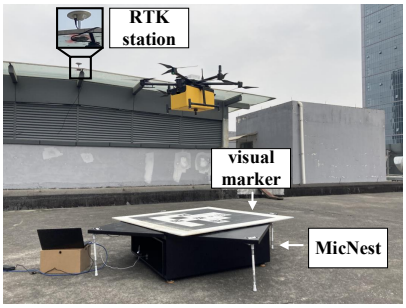


Figure 16: Experimental setting. Our website [68] presents a vivid demonstration.

5) MICNEST can *navigate the drone onto the landing platform robustly and accurately*, with a success rate of at least 94%.
6) MICNEST can *localize multiple drones* with a limited degradation of the localization accuracy.

The rest of this section provides experimental evidence.

7.1 Implementation and Setting

We use drone equipment and a deployment setting that closely mimic actual applications.

Drone. As shown in Fig. 15, we use a custom drone manufactured by Meituan that are currently exploring the feasibility of instant deliveries with drones. The drone is equipped with six propellers each hooked to a brushless TMotor and is steered by the PX4 [69] flight controller, running at 400 Hz. The drone has a payload capacity up to 2.6 kg at liftoff, which is the maximum load that local regulations allow. The altitude is jointly estimated by an on-board barometer and a Benewake downward-facing LIDAR.

The only additional equipment, other than what would normally be present on any professional-grade drone, are the speakers. They are attached to the bottom of the drone and should be as light as possible not to negatively impact the payload capacity. We use a VISTEON speaker weighing a mere 47 g. The applied voltage and operating current are 12.6 V and 45 mA. According to our measurements, the speaker draws less than 0.1% of the total battery power. The speaker volume is empirically set to 70-75 dB SPL (measured at 1 m distance), which is arguably moderate.

Landing platform and software. We build a squared foldable landing platform, shown in Fig. 16, measuring 1 m x 1 m and 1.41 m x 1.41 m when folded or unfolded, respectively. Four omni-directional SPK0641HT4H digital microphones are installed at the corners. The distance between two microphones along the diagonal is 1.86 m.

We use an XMOS XU216 data acquisition board to drive and sample the microphones, so that the four signals are synchronized. The sampling rate is 48 KHz. The board then streams the audio signals to a laptop via USB UAC 2.0 with a latency lower than 0.5 ms. We use a high-pass filter with a cutoff frequency of 500 Hz to pre-process the audio signals.

MFT is implemented in C++ with the CUDA 11.0 library, running on a machine with an Intel i9-11900H CPU, 32 GB memory, and an NVIDIA RTX 3070 GPU.

Ground truth. We conduct the experiments in a secluded area on a building roof, where the reception of GPS signals is of very high quality. We deploy an RTK base station close to the experimental site, as shown in Fig. 16, which keeps rebroadcasting the phase of the GPS signal it observes.

In such a setting, the RTK processing on the drone works at high fidelity, especially because it does not experience the performance degradation or outage problems that occur in an operational site, for example, in a urban canyon, as mentioned in Sec. 2. Therefore, we use the localization results of RTK as ground truth. We compare the performance of MICNEST with ArUco markers [70], a state-of-the-art visual localization system. We place an ArUco marker of 1.5 m $\times$ 1.5 m on top of the landing platform, as shown in Fig. 16.

Flight trajectories. The drone operates automatically during the experiments, exactly as it would in an actual application. We use QGroundControl [71] as ground station control software to plan the flight trajectories. In addition to a hovering mode that keeps the drone stable at a given position, we consider three possible flight trajectories:

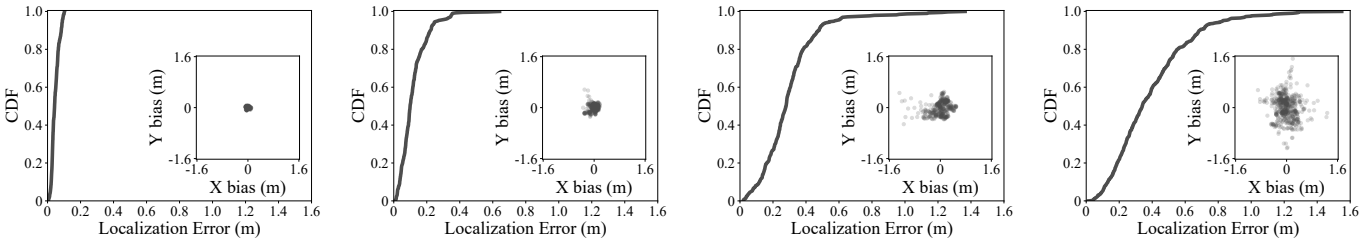
- In *vertical flight*, the drone takes off and vertically climbs to a given altitude; next, it lands back onto the platform by following the same trajectory in the opposite way.
- In a *squared spiral*, the drone takes off vertically, climbs to a given altitude, and flies twice along a squared spiral trajectory at constant altitude; next, it flies horizontally back to the starting point and lands vertically.
- In a *dense squared spiral*, the trajectory is the same as the squared spiral above, yet the side length of the square is increased by 2 m every two turns, instead of 10 m; the drone flies for ten rounds in total, rather than two.

7.2 Accuracy and Latency

Localization accuracy is a function of several factors.

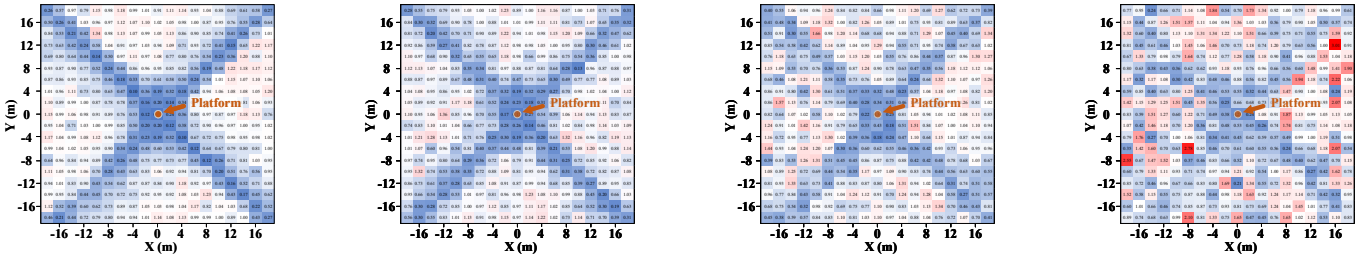
Impact of altitude. We program the drone to perform a *vertical flight* up to 120 m. We plot the cumulative distribution function (CDF) of the localization error, compared to RTK that represents ground truth, at different altitude intervals in Fig. 17(a)-(d). Note that MICNEST calculates the horizontal coordinates of the drone at a given altitude. The scattered plot in each figure shows all the localization biases of MICNEST within the specified altitude interval.

When the altitude is below (above) 20 m (80 m), the median error is 0.043 m (0.339 m). On average, the relative error, that is, the absolute localization error over the distance to the platform, is only 0.53%. The plots also demonstrate that the localization error *decreases as the drone approaches the platform*. This is indeed a desired characteristic for a localization system enabling precise landing, that is, *the performance improves when it matters the most*. In MICNEST, this is due to: *i)* the far-field effect: when the drone flies low, its slight movements may result in a notable fluctuation in TDoA, whereas at higher altitudes, the TDoA fluctuations become less and less distinguishable; and *ii)* signal attenuation: acoustic signals experience more attenuation at longer distances and appear to be noisier, this also explains why there are more and more outliers as the altitude increases.



(a) 0-20 m altitude (b) 20-50 m altitude (c) 50-80 m altitude (d) 80-120 m altitude

Figure 17: Localization error compared to altitude.



(a) 25 m altitude (b) 50 m altitude (c) 75 m altitude (d) 100 m altitude

Figure 18: Heatmaps of localization error on the horizontal plane at different altitudes.

Impact of horizontal distance. We program the drone to fly a *dense square spiral*. Fig. 18 shows a heatmap representation of the distribution of the localization error across the bidimensional plane at four different altitudes.

Generally, MICNEST 's localization error tends to decrease as the drone is horizontally close to the platform. It enjoys the highest localization resolution when the drone horizontally aligns with the center of the platform. The reasons for this behavior are similar to the ones explaining the performance at different altitudes, discussed above.

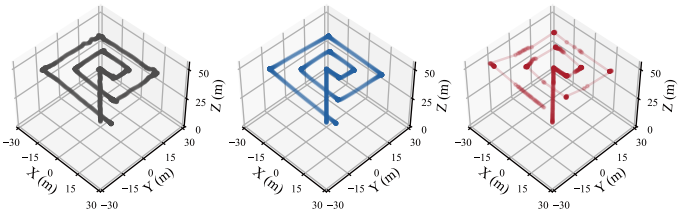
An interesting observation is the visible "X" pattern in the heatmaps, which corresponds to the higher localization accuracy. This pattern corresponds to the two vertical bisectors of the diagonal microphones. TDoA-based localization has indeed the highest spatial resolution in these conditions.

Localization trajectory. As an example, Fig. 19(a)-(c) show the localization results as the drone flies a *squared spiral* at 50 m altitude. An illustrative video is available [68].

The plots demonstrate how MICNEST and RTK successfully localize the drone throughout the whole flight. In contrast, the visual marker works intermittently, because it is difficult for the camera on the drone to capture the visual marker, especially at higher altitudes. Results at different altitudes are nonetheless available on our website [68].

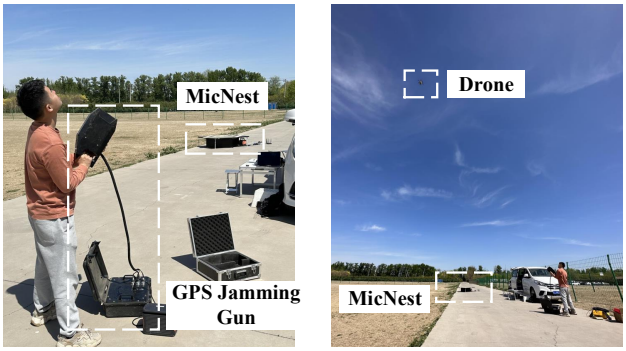
To further motivate MicNest, we evaluate the performance of MicNest when RTK is hampered. We deliberately destroy the signals of RTK using a GPS jamming gun in the experiment. We program the drone to fly along a squared spiral trajectory at an altitude of 50 m, while collecting localization results from both RTK and MicNest. To simulate GPS signal interference, we use the GPS jamming gun to simultaneously jam two GNSS frequency bands: 1197-1288 MHz and 1541-1622 MHz. As shown in Fig. 20(a) and (b), during the drone's flight, a human operator fires the jamming gun every 5 seconds to affect the RTK's performance.

Fig. 21(a) shows the localization results of MicNest, and Fig. 21(b) and (c) plot the localization results of RTK when



(a) MICNEST (b) RTK (c) Visual marker

Figure 19: Drone trajectories localized by MICNEST, RTK, and using the visual marker. We provide an illustrative video of this experiment as well as the results at other altitudes on our website [68].

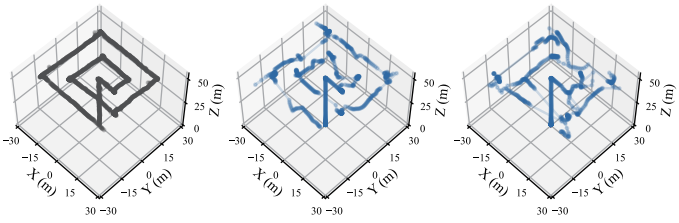


(a) (b)

Figure 20: Using a jamming gun to hamper GPS signals.

the GPS jamming power is 12w and 31w, respectively. The results clearly show that MicNest can localize the drone throughout the whole flight, while, as expected, RTK works only when the jamming gun is off. This comparison clearly demonstrates how MicNest can supplement the deficiency of RTK, particularly when the GPS signal quality is poor.

Latency. We measure the localization latency as the time between the moment a PRN pulse is transmitted and the moment the corresponding localization result is obtained.



(a) MICNEST (b) RTK (12W) (c) RTK (31W)

Figure 21: Drone trajectories localized by MICNEST, and RTK at different jamming powers.

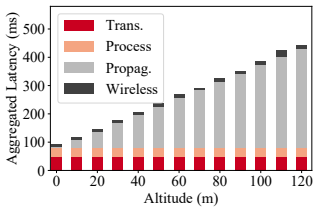


Figure 22: Aggregated latency.

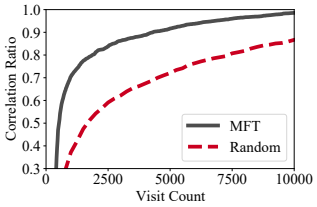


Figure 23: Visit count of MFT.

Four components contribute to this quantity: *i)* the *propagation delay*, that is, the time needed for acoustic signal to reach the microphones; *ii)* the *transmission delay* that equals the PRN pulse length (50 ms); *iii)* the *processing delay* that is dominated by the time for MFT to detect pulses, which we examine later; and *iv)* the *communication delay*, that is, the time for TDoAs to be sent back to the drone via WiFi.

Fig. 22 plots the aggregated latency of localization using MICNEST at different altitudes. As expected, the only varying latency component is the propagation delay. Our measurements indicate that the mean processing delay is 29.7 ms with a standard deviation of 0.6 ms, whereas the mean WiFi delay is 11 ms. The transmission delay is also fixed. The plot shows that the aggregated latency changes with the distance linearly. Since the sound speed is relatively slow, the aggregated latency is dominated by the propagation delay when the drone is far from the platform. As a whole MICNEST can provide localization updates at a rate more than sufficient to feed the flight control loop [3], [38].

7.3 Maximum MFT visit count

The processing delay of MFT is determined by the number of tree paths visited while looking for the solution. The more search paths we visit, the larger correlation value the MFT outputs and thus the larger SNR gain the MFT yields.

We execute an experiment flying a *squared spiral* at 120 m altitude. Fig. 23 plots the correlation ratio[10] of MFT outputs as a function of the maximum visit count. In comparison, the results of a random search, that is randomly picking a tree path that is not yet visited, are also shown. Fig. 23 shows that when the visit count is 5000, the correlation ratio of the MFT and if the random search is 0.92 and 0.72, respectively. In our implementation, we set the maximum visit count of MFT to 5000, which ensures that pulse detection can be completed in a deterministic period. The corresponding time required for detecting a pulse is 26.5 ms.

10. For visit count N_{vis}, the correlation ratio is defined as the ratio of the best correlation values of the first N_{vis} paths to the maximum correlation value of the brute-force search.

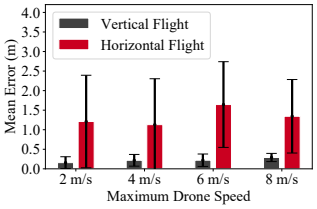


Figure 24: The impact of drone speed.

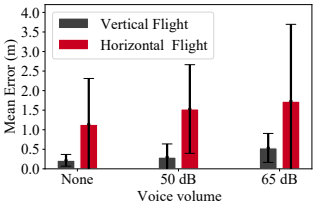


Figure 25: The impact of background noise.

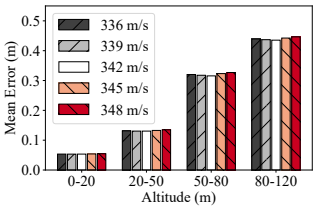


Figure 26: The impact of sound speed.

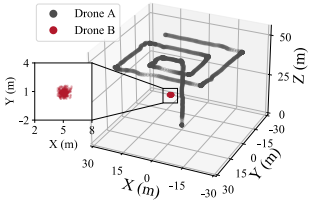


Figure 27: The localization trajectories of multiple drones.

7.4 Impact of External Factors

Factors that are not under the direct control of MICNEST may influence its performance, including drone speed, background noise, and sound speed.

Drone speed. The drone flies a *squared spiral* at 50 m altitude with different speeds: 2 m/s, 4 m/s, 6 m/s, and 8 m/s[11].

Since the whole trajectory consists of both vertical and horizontal parts, Fig. 24 plots the localization errors in either dimension. The error along the vertical parts appears lower than along the horizontal ones. This is because the drone is horizontally aligned to the center of the landing platform during the vertical parts. Most importantly, the plot provides evidence that the drone speed has a negligible impact on the accuracy performance of MICNEST. In turn, this demonstrates that the MFT can search for the correct drone speed and compensate the Doppler effect effectively.

Background noise. We place a loudspeaker 1.5 m away from the center of the landing platform to emulate a source of noise. The speaker plays music continuously at a frequency between 200 Hz and 3.5 kHz with two volumes: 50 dB and 65 dB. Note how the latter setting is effectively close to the volume of the MICNEST speaker aboard the drone. The drone flies again a *squared spiral* at 50 m.

Fig. 25 illustrates the performance degradation with increasing noise. The mean error of the horizontal flight is 1.13 m in the case of no noise and increases to 1.73 m at 65 dB noise, which represents a case of strong background noise. By adopting advanced noise reduction techniques, its impact can be further mitigated [72], [73].

Sound speed. The speed of sound changes with temperature. As a rule of thumb, a 1 °C increase corresponds to a 0.6 m/s increase in sound speed [74]. To investigate this aspect, we program the drone to perform a *vertical flight* up to 120 m altitude. We conduct the flight when the sound speed is 342

11. Note that the drone's speed cannot be perfectly fixed during flight; these values may be regarded as the maximum speed that the drone can reach while flying a predetermined trajectory.

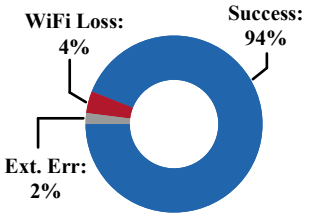


Figure 28: The statistics of the landing results.

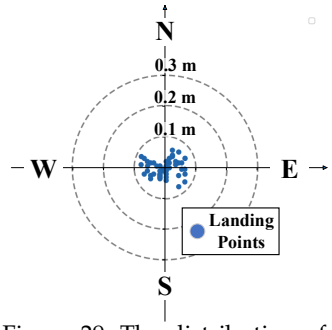


Figure 29: The distribution of the landing points.

m/s. Then we parameterize the sound speed with different values: 336 m/s, 339 m/s, 342 m/s, 345 m/s, and 348 m/s. Fig. 26 shows the results in localization error at different altitudes. The performance difference at different sounds speeds is negligible, yet in real operations one can calibrate the parameter of sound speed in MICNEST by measuring the sound speed[12] to further refine the performance.

7.5 Drone Landing

Guiding the drone onto the landing platform accurately and robustly is the ultimate design goal of MICNEST. In this experiment, we feed the localization results of MICNEST to the flight controller of the drone. For each experiment, the drone first flies to an altitude of 120 m and begins to land. During drone landing, the flight controller is forced to rely solely on the localization results of MICNEST to navigate the drone onto the center of the landing platform.

We consider the landing operation as *successful* whenever the drone hits the center of the landing platform with a maximum error of 10 cm. This is smaller than the frame size of the drone and an acceptable margin of error for applications, such as drone deliveries, where the drone drops a packet after landing or performs actions that require aligning the drone with the landing platform. We repeat this experiment 50 times to gather statistically relevant metrics.

Fig. 28 reports the results. MICNEST navigates the drone onto the landing platform with a success rate of 94%. Only 3 cases of failure exist, and all of them are not caused by MICNEST. Two of them are caused by the WiFi connection loss and one is caused by an error independent of MICNEST operation, as the object avoidance module was mistakenly triggered. Fig. 29 further illustrates the spatial distribution of landing points. The average landing error is only 4.3 cm.

A demonstration video that shows the process of navigating drone landing is available [68].

7.6 Localization of Multiple Drones

We conclude the evaluation by studying MICNEST's ability to concurrently localize multiple drones. We use two drones A and B. Drone A flies along a *squared spiral* at 40 m altitude, while drone B hovers at 20 m altitude. The two drones play different PRN pulses.

Fig. 27 plots the localization results of the two drones. MICNEST can detect both drones' pulses from the collided signal. The mean localization errors for drones A and B are 1.77 m and 0.38 m, respectively. Below 20 m altitude, this metric for drone A is reduced to 0.17 m, that is, where accuracy is most important for precise landing.

Compared to the single-drone localization performance, MICNEST performance is marginally degraded because the presence of multiple drones adds up the degree of background interference. Besides the countermeasures mentioned in Sec. 7.4 to tame background noise, one may also improve the coding scheme of pulse and adopt an adaptive power control scheme [75], [76], [77].

8 DISCUSSION

We elaborate next on the rationale behind some key design decisions in MICNEST design and implementation.

Why not using inaudible or higher-frequency sounds? The attenuation of a signal increases as the signal frequency increases. It can be expected that a higher-frequency signal that spans the same bandwidth as our current pulses (24 kHz) experiences much more attenuation. This would inherently limit the operating range of the system.

Is acoustic signal propagation a limitation? Fig. 22 shows that the signal propagation becomes a limiting factor for latency only at high altitudes. Here, the propagation delay can be tolerated to some degree as long as it can be estimated and reported to the navigation system [78]. As the drone approaches the platform, that is, where the highest location update rate is required, the contribution of signal propagation to processing latency becomes increasingly immaterial.

What about the number of microphones? Our implementation of MICNEST uses four distributed microphones. We may further improve localization performance by deploying more microphones and, for example, using beamforming techniques to further enhance the signal [79].

Why not using frequency-modulated continuous-wave (FMCW) signals? FMCW signals are resistant to Doppler effect [80]. However, it is the linear Doppler effect that FMCW can resist, not the non-linear one. In addition, FMCW signals cannot satisfy the practical requirements we outline in the Introduction, such as being friendly to the human ear or resistant to impersonation attacks.

What about the performance of multi-drone localization? There exist methods to further improve the performance of multi-drone localization. For example, each drone may adopt an adaptive volume strategy: reducing the volume when the altitude decreases. Therefore, the PRN pulses transmitted by high-altitude drones are less interfered by the pulses of low-altitude drones. Another remedy is to improve the orthogonality of PRN pulses.

Why not accelerating tree search using information from drone IMU? It is the *radial* drone velocity with respect to the microphones that MFT searches for, not the velocity with respect to the Earth. Before converting the estimated velocity to the radial one, we should know the location of the drone with respect to the microphones. This actually leads to

12. Measuring sound speed is simple for MicNest. We can put a speaker in line with two diagonal microphones, and measure the TDoA of PRN pulses. The sound speed can be calculated by dividing the microphone distance by the TDoA.

a "chicken-and-egg" problem: tree-search acceleration and drone localization are a prerequisite of each other.

Can we reversely deploy microphones on the drone and a speaker on the ground? This may be feasible, but a practical issue is that the drone size is limited, which sets an upper bound on the inter-microphone distance (aperture), resulting in a lower localization resolution. In addition, the computing resource of drones is generally not sufficient to support real-time pulse detection.

How does multipath effect impact MicNest? In general, MICNEST can tolerate multipath effect as long as there is a line-of-sight (LOS) path between the microphones and the speaker. Thanks to the low sound speed, a slight difference in path lengths will lead to a distinguishable time difference of arrival. Therefore, the paths reflected from, for example, the surrounding buildings will not overlap with the LOS path in the time domain. Given that the LOS path is the strongest, we can implicitly determine the LOS path by choosing the most significant correlation peak.

9 CONCLUSION

MICNEST enables precise landing of drones using acoustic signals. The key enabling technologies we present are MFT, a novel pulse detector that models the problem as a tree search problem, and its efficient low-latency implementation. These allow MICNEST to localize a drone 120 m away with 0.53% relative localization error at 20 Hz location update frequency. We demonstrate that MICNEST can accurately and robustly navigate a drone onto a landing platform with a 94 % success rate and an average landing error of only 4.3 cm.

REFERENCES

[1] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, p. 460, 2015.

[2] BOEING, "Statistical summary of commercial jet airplane accidents," https://www.boeing.com/resources/boeingdotcom/company/about_bca/pdf/statsum.pdf, 2021, accessed: 2022-02-06.

[3] A. Patelli *et al.*, "Model-based real-time testing of drone autopilots," in *Proceedings of DRONET, Singapore, June 26, 2016*, 2016.

[4] Zipline, "In the ari with zipline's medical delivery drones," https://spectrum.ieee.org/in-the-air-with-ziplines-medical-delivery-drones, 2022, accessed: 2022-03-25.

[5] Dominos, "Pizza-by-drone a reality with world-first customer deliveries in new zealand," https://www.dominos.com.au/inside-dominos/media/november-2016-pizza-by-drone-a-\\reality-with-world-first-customer-deliveries-in-new-zealand, 2016, accessed: 2022-02-06.

[6] Ele.me, "Ele.me cleared to use delivery drones in china," https://www.pymnts.com/news/delivery/2018/eleme-food-delivery-drones-china/, 2022, accessed: 2022-02-06.

[7] Meituan, "Food delivery giant meituan unveils drones for delivery service, offering new user experience," https://pandaily.com/food-delivery-giant-meituan-unveils-drones-for-delivery-service/, 2021, accessed: 2022-02-06.

[8] Amazon, "Amazon prime air," https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011, 2022, accessed: 2022-02-06.

[9] Alphabet, "Project wing," https://wing.com/, 2022, accessed: 2022-02-06.

[10] Walmart, "Walmart invests in droneup, the nationwide on-demand drone delivery provider," https://corporate.walmart.com/newsroom/2021/06/17/walmart-invests-in-droneup-the-nationwide-on-demand-\\drone-delivery-provider, 2021, accessed: 2022-02-06.

[11] DroneUp, "Droneup," https://www.droneup.com/, 2022, accessed: 2022-02-06.

[12] JD.com, "Jd.com's drone delivery program takes flight in rural china," https://jdcorporateblog.com/jd-coms-drone-delivery-program-takes-flight-in-rural-china/, 2016, accessed: 2022-02-06.

[13] UPS, "Ups operates first ever u.s. drone covid-19 vaccine delivery," https://about.ups.com/us/en/our-stories/innovation-driven/drone-covid-vaccine-deliveries.html, 2021, accessed: 2022-02-06.

[14] A. Today, "Drone delivery crash in switzerland raises safety concerns as ups forms subsidiary," https://www.aviationtoday.com/2019/08/08/drone-delivery-crash-in-switzerland-raises-safety-concerns/, 2019, accessed: 2022-02-06.

[15] Wikipedia, "Real-time kinematic positioning," https://en.wikipedia.org/wiki/Real-time_kinematic_positioning, 2022, accessed: 2022-02-06.

[16] P. D. Groves, "Shadow matching: A new gnss positioning technique for urban canyons," *The journal of Navigation*, vol. 64, no. 3, pp. 417–430, 2011.

[17] A. Budiyono, "Principles of gnss, inertial, and multi-sensor integrated navigation systems," *Industrial Robot: An International Journal*, 2012.

[18] L.-T. Hsu, "Analysis and modeling gps nlos effect in highly urbanized area," *Springer GPS solutions*, vol. 22, no. 1, pp. 1–12, 2018.

[19] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China, May 9-13, 2011*, 2011.

[20] M. Krogius, A. Haggenmiller, and E. Olson, "Flexible layouts for fiducial tags," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Macau, SAR, China, November 3-8, 2019*, 2019.

[21] J. Wang and E. Olson, "April*f*Tag 2: Efficient and robust fiducial detection," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Daejeon, South Korea, October 9-14, 2016*, 2016.

[22] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose tracking from natural features on mobile phones," in *Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality, Cambridge, UK, September 15-18, 2008*.

[23] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, June 20-26, 2005*, 2005.

[24] Bitcraze AB, "Lighthouse positioning system," https://www.bitcraze.io/documentation/system/positioning/ligthouse-positioning-system/, 2021.

[25] bitcraze, "Loco positioning system," https://www.bitcraze.io/documentation/system/positioning/loco-positioning-system/, 2022, accessed: 2022-02-06.

[26] A. Dhekne, A. Chakraborty, K. Sundaresan, and S. Rangarajan, "Trackio: Tracking first responders inside-out," in *Proceedings of USENIX NSDI, Boston, MA, February 26-28, 2019*, 2019, pp. 751–764.

[27] Bitcraze AB, "Motion capture positioning," https://www.bitcraze.io/documentation/system/positioning/mocap-positioning/, 2021.

[28] P. Autopilot, "Landing phases flow diagram," https://docs.px4.io/master/en/advanced_features/precland.html\#landing-phases-flow-diagram, 2022, accessed: 2022-02-06.

[29] M. Soumekh, *Synthetic aperture radar signal processing*. New York: Wiley, 1999, vol. 7.

[30] decawave, "Dwm1000 datasheet," https://www.decawave.com/sites/default/files/resources/DWM1000-Datasheet-V1.6.pdf, 2022, accessed: 2022-02-06.

[31] T. Instruments, "Introduction to mmwave sensing: Fmcw radars," https://training.ti.com/sites/default/files/docs/mmwaveSensing-FMCW-offlineviewing_4.pdf, 2022, accessed: 2022-02-06.

[32] A. Wang, J. E. Sunshine, and S. Gollakota, "Contactless infant monitoring using white noise," in *Proceedings of ACM MobiCom, Los Cabos, Mexico, October 21-25, 2019*, 2019, pp. 1–16.

[33] M. L. Stanchina, M. Abu-Hijleh, B. K. Chaudhry, C. C. Carlisle, and R. P. Millman, "The influence of white noise on sleep in subjects

exposed to icu noise," *Elsevier Sleep medicine*, vol. 6, no. 5, pp. 423–428, 2005.

[34] M. A. Nobile, "Identifying prominent discrete tones in noise emissions," *The Journal of the Acoustical Society of America*, vol. 78, no. S1, pp. S33–S33, 1985.

[35] W. H. Organization *et al.*, "Environmental noise guidelines for the european region," 2018.

[36] A. Bannis, H. Y. Noh, and P. Zhang, "Bleep: motor-enabled audio side-channel for constrained uavs," in *Proceedings of ACM MobiCom, London, United Kingdom, September 21-25, 2020*, 2020.

[37] B. Schäffer, R. Pieren, K. Heutschi, J. M. Wunderli, and S. Becker, "Drone noise emission characteristics and noise effects on humans—a systematic review," *MDPI International Journal of Environmental Research and Public Health*, vol. 18, no. 11, p. 5940, 2021.

[38] E. Bregu *et al.*, "Reactive control of autonomous drones," in *Proceedings of ACM MobiSys, Singapore, June 26-30, 2016*, 2016.

[39] M. Afanasov, A. Djordjevic, F. Lui, and L. Mottola, "Flyzone: A testbed for experimenting with aerial drone applications," in *Proceedings of ACM MobiSys, Seoul, Republic of Korea, June 17-21, 2019*, 2019, pp. 67–78.

[40] W. Mao, Z. Zhang, L. Qiu, J. He, Y. Cui, and S. Yun, "Indoor follow me drone," in *Proceedings of ACM MobiSys, Niagara Falls, NY, USA, June 19-23, 2017*, 2017.

[41] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik, "Indoor localization without infrastructure using the acoustic background spectrum," in *Proceedings of ACM MobiSys, Bethesda, MD, USA, June 28 - July 01, 2011*, 2011, pp. 155–168.

[42] H. Chu, P. Huang, R. R. Choudhury, and F. Zhao, "Guoguo: Enabling fine-grained indoor localization via smartphone," in *Proceedings of ACM MobiSys, Taipei, Taiwan, June 25-28, 2013*, 2013, pp. 235–248.

[43] F. Li, H. Chen, X. Song, Q. Zhang, Y. Li, and Y. Wang, "Condiosense: High-quality context-aware service for audio sensing system via active sonar," *Springer Personal and Ubiquitous Computing*, vol. 21, no. 1, pp. 17–29, 2017.

[44] T. Akiyama, M. Sugimoto, and H. Hashizume, "Time-of-arrival-based indoor smartphone localization using light-synchronized acoustic waves," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 100, no. 9, pp. 2001–2012, 2017.

[45] W. Huang, X. Li, Y. Xiong, P. Yang, Y. Hu, X. Mao, F. Miao, B. Zhao, and J. Zhao, "Stride-in-the-loop relative positioning between users and dummy acoustic speakers," *IEEE JSAC*, vol. 35, no. 5, pp. 1104–1117, 2017.

[46] S. Shen, D. Chen, Y. Wei, Z. Yang, and R. R. Choudhury, "Voice localization using nearby wall reflections," in *Proceedings of ACM MobiCom, London, United Kingdom, September 21-25, 2020*, 2020, pp. 1–14.

[47] W. Wang, J. Li, Y. He, and Y. Liu, "Symphony: localizing multiple acoustic sources with a single microphone array," in *Proceedings of ACM SenSys, Virtual Event, Japan, November 16-19, 2020*, 2020, pp. 82–94.

[48] Q. Lin, Z. An, and L. Yang, "Rebooting ultrasonic positioning systems for ultrasound-incapable smart devices," in *Proceedings of ACM MobiCom, Los Cabos, Mexico, October 21-25, 2019*, 2019, pp. 1–16.

[49] Y. Tung and K. G. Shin, "Echotag: Accurate infrastructure-free indoor location tagging with smartphones," in *Proceedings of ACM MobiCom, Paris, France, September 7-11, 2015*, 2015, pp. 525–536.

[50] H. Han, S. Yi, Q. Li, G. Shen, Y. Liu, and E. Novak, "Amil: Localizing neighboring mobile devices through a simple gesture," in *IEEE INFOCOM, San Francisco, CA, USA, April 10-14, 2016*, 2016, pp. 1–9.

[51] A. Mandal, C. V. Lopes, T. Givargis, A. Haghighat, R. Jurdak, and P. Baldi, "Beep: 3d indoor positioning using audible sound," in *IEEE Consumer Communications and Networking Conference, Las Vegas, NV, USA, January 3-6, 2005*, 2005, pp. 348–353.

[52] L. Girod, M. Lukac, V. Trifa, and D. Estrin, "A self-calibrating distributed acoustic sensing platform," in *Proceedings of ACM SenSys, Boulder, Colorado, USA, October 31 - November 3, 2006*, 2006, pp. 335–336.

[53] A. S. Pinna, G. Portaluri, and S. Giordano, "Shooter localization in wireless acoustic sensor networks," in *IEEE Symposium on Computers and Communications, Heraklion, Greece, July 3-6, 2017*, 2017, pp. 473–476.

[54] S. Li, X. Fan, Y. Zhang, W. Trappe, J. Lindqvist, and R. E. Howard, "Auto++ detecting cars using embedded microphones in real-time," *Proceedings of the ACM IMWUT*, vol. 1, no. 3, pp. 1–20, 2017.

[55] J. Zhang, T. Yan, J. A. Stankovic, and S. H. Son, "Thunder: towards practical, zero cost acoustic localization for outdoor wireless sensor networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 1, pp. 15–28, 2007.

[56] M. Wang, W. Sun, and L. Qiu, "MAVL: multiresolution analysis of voice localization," in *Proceedings of USENIX NSDI, April 12-14, 2021*, 2021, pp. 845–858.

[57] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.

[58] B. Farhang-Boroujeny, *Adaptive filters: theory and applications*. John Wiley & Sons, 2013.

[59] U. Cheng, W. Hurd, and J. Statman, "Spread-spectrum code acquisition in the presence of doppler shift and data modulation," *IEEE Transactions on Communications*, vol. 38, no. 2, pp. 241–250, 1990.

[60] J. Iinatti, "On the threshold setting principles in code acquisition of dsss signals," *IEEE JSAC*, vol. 18, no. 1, pp. 62–72, 2000.

[61] betaflight, "Betaflight flight controller," https://github.com/betaflight/betaflight, 2021.

[62] cleanflight, "Cleanflight flight controller," https://github.com/cleanflight/cleanflight, 2021.

[63] rs2k, "Raceflight flight controller," https://github.com/rs2k/raceflight, 2021.

[64] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *Proceedings of Springer International conference on computers and games, Turin, Italy, May 29-31, 2006*, 2006, pp. 72–83.

[65] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Proceedings of Springer European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006*, 2006, pp. 282–293.

[66] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[67] R. A. Roberts and C. T. Mullis, *Digital signal processing*. Addison-Wesley Longman Publishing Co., Inc., 1987.

[68] MicNest, "Supplemental page of micnest," https://micnest.github.io, 2022, accessed: 2022-06-21.

[69] P. Autopilot, "Px4 autopilot," https://docs.px4.io/master/en/, 2022, accessed: 2022-02-06.

[70] OpenCV, "Detection of aruco markers," https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html, 2022, accessed: 2022-02-06.

[71] DroneCode, "Qgroundcontrol," http://qgroundcontrol.com/, 2022, accessed: 2022-02-06.

[72] S. V. Vaseghi, *Advanced digital signal processing and noise reduction*. John Wiley & Sons, 2008.

[73] R. Bentler and L.-K. Chiou, "Digital noise reduction: An overview," *Trends in amplification*, vol. 10, no. 2, pp. 67–82, 2006.

[74] Wikipedia, "Speed of sound," https://en.wikipedia.org/wiki/Speed_of_sound, 2022, accessed: 2022-03-25.

[75] A. J. Viterbi, *CDMA: principles of spread spectrum communication*. Addison Wesley Longman Publishing Co., Inc., 1995.

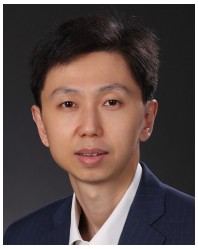
[76] R. M. Rao and S. A. Dianat, *Basics of code division multiple access (CDMA)*. SPIE Press, 2005, vol. 67.

[77] M. Rintamäki *et al.*, *Adaptive power control in CDMA cellular communication systems*. Helsinki University of Technology, 2005.

[78] R. C. Dorf and R. H. Bishop, *Modern control systems*. Pearson Prentice Hall, 2008.

[79] J. Benesty, J. Chen, and Y. Huang, *Microphone array signal processing*. Springer Science & Business Media, 2008.

[80] A. G. Stove, "Linear fmcw radar techniques," in *IEE Proceedings F-Radar and Signal Processing*, vol. 139, no. 5. IET, 1992, pp. 343–350.



Yuan He is an associate professor in the School of Software and BNRist of Tsinghua University. He received his B.E. degree in the University of Science and Technology of China, his M.E. degree in the Institute of Software, Chinese Academy of Sciences, and his PhD degree in Hong Kong University of Science and Technology. His research interests include wireless networks, Internet of Things, pervasive and mobile computing. He is a member of IEEE and ACM.

Weiguo Wang is currently a PhD. student in Tsinghua University. He received his B.E. degree in the University of Electronic Science and Technology of China (UESTC). His research interests include acoustic sensing and mobile computing.

Luca Mottola is a Full Professor at Politecnico di Milano (Italy) and a Senior Researcher at RI.SE Sweden. He is past General Chair for ACM/IEEE CPS-IoT Week 2022 and past PC chair for ACM MOBISYS, ACM SENSYS, ACM/IEEE IPSN, and ACM EWSN. He received the ACM SENSYS Test of Time Award in 2022, two ACM SigMobile Research Highlights, and is a Google Faculty Award winner. He holds or held visiting positions at Uppsala University, NXP Technologies, TU Graz, and USI Lugano.

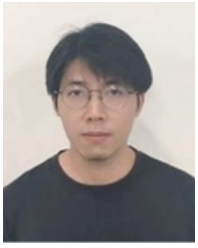
Shuai Li is currently a Master student at the School of Software, Tsinghua University. He received the B.E. degree in Software Engeering at Tsinghua University in 2020. His research interests include Internet of Things and wireless sensing.

Yimiao Sun is currently a PhD. student in Tsinghua University. He received his B.E. degree in the University of Electronic Science and Technology of China (UESTC) in 2021. His research interests include Internet of Things and wireless sensing.

Jinming Li is currently a graduate student in Tsinghua University. He received his B.E. degree in Tsinghua University. His research interests include wireless networks and Internet of Things.

Hua Jing works as a software director at Meituan. He received his M.E. degree and B.E. degree in Beihang University. His research interests include heterogeneous computing and realtime communication.

Ting Wang works as a system developer at Meituan. She received her B.E. degree in Sichuan University and her M.E. degree in Tsinghua University. She works in the fields of UAV and RFID.



Yulei Wang works as a software architecture manager at Meituan. He received his B.E. degree and M.E. degree in Beihang University. He works in the fields of UAV and robot.