3D Backscatter Localization for Fine-Grained Robotics

Zhihong Luo Qiping Zhang Yunfei Ma Manish Singh Fadel Adib
MIT Media Lab

Abstract – This paper presents the design, implementation, and evaluation of TurboTrack, a 3D localization system for fine-grained robotic tasks. TurboTrack's unique capability is that it can localize backscatter nodes with sub-centimeter accuracy without any constraints on their locations or mobility. TurboTrack makes two key technical contributions. First, it presents a pipelined architecture that can extract a sensing bandwidth from every single backscatter packet that is three orders of magnitude larger than the backscatter communication bandwidth. Second, it introduces a Bayesian space-time super-resolution algorithm that combines time series of the sensed bandwidth across multiple antennas to enable accurate positioning. Our experiments show that TurboTrack simultaneously achieves a median accuracy of sub-centimeter in each of the x/y/z dimensions and a 99^{th} percentile latency less than 7.5 milliseconds in 3D localization. This enables TurboTrack's real-time prototype to achieve fine-grained positioning for agile robotic tasks, as we demonstrate in multiple collaborative applications with robotic arms and nanodrones including indoor tracking, packaging, assembly, and handover.

1 Introduction

The emergence of agile and miniature robots has led to a novel set of capabilities and sensing tasks. Nanodrones that can fit in your palm are used for mapping indoor environments and are deployed in swarms for emergency response and hazard detection in urban settings [29, 67, 43]. Dexterous robotic arms have shifted manufacturing automation from assembly lines that consist of dozens of robots, each of which is dedicated to a single task, to a pair of multi-functional robots that can collaborate on picking up, assembling, and packaging items [16, 25]. Personal robots like the Roomba are already cleaning our homes, and their roles are expected to expand to folding clothes, washing dishes, and helping with other daily routines [4, 47, 21].

A fundamental challenge that still faces agile robots, however, is their ability to operate in highly cluttered settings [28, 42, 27]. While highly-trained vision systems can perform accurate classification and tracking tasks, their performance suffers in cluttered environments, and fails if the object of interest is fully occluded, e.g., if a robot must pick up an item from under a pile [74, 48]. Moreover, tracking individual nanodrones in a swarm is challenging even in line-of-sight settings due to their constrained size and payload, which prevent instrumenting them with visually identifiable markers [33, 44].

RF-based identification and localization offers an alternative sensing modality that is highly robust to visual clutter, providing an attractive solution and a complementary sensing capability. Motivated by the recent advances in RF-based localization by the networking community [68, 45, 60, 50, 72], in this paper, we set out to build a system for RF-based identification and 3D localization for fine-grained robotic tasks. Building such a system requires meeting requirements along three fronts:

- **Accuracy:** To enable agile manipulation tasks, like grasping and packaging, we need to achieve sub-centimeter localization accuracy [57, 18]. Such accuracy is needed to enable a robot to align its grip with an object for item grasping and manipulation tasks.
- **Mobility:** Robotic arms and nanodrones are in constant mobility as they perform sensing and localization. Hence, a localization system for fine-grained robotic tasks must be fast enough to track them and deal with random mobility patterns.
- **Scalability:** Since robots are expected to manipulate everyday items, we need cost-effective solutions that scale to hundreds or thousands of items, even in relatively confined areas like homes or small businesses.

Unfortunately, no system exists today that can realize all three goals simultaneously. On one hand, WiFi and Bluetooth-based solutions [34, 73, 68] are not scalable in our context since it is not feasible or cost-effective to tag every item with a Bluetooth or WiFi radio. On the other hand, billions of manufactured items are already tagged with few-cent RFIDs, making RFID-based localization attractive from a scalability standpoint. However, RFID localization solutions are limited either in their accuracy or in their ability to deal with mobility. Techniques like Tagoram [75] and RFIDraw [71] can work with mobile RFIDs, but they have decimeter-scale accuracy in obtaining a tag's exact position;[1] this prevents using them for tasks like grasping or manipulation. Others like MobiTagBot [60] and RFCompass [69] achieve high accuracy but require the tag to remain static for multiple seconds as they perform their localization; this prevents them from tracking nanodrones or enabling agile tasks

[1] These techniques track *changes in distance* so they can accurately recover the shape of a trajectory, but relatively low accuracy in obtaining the exact position.

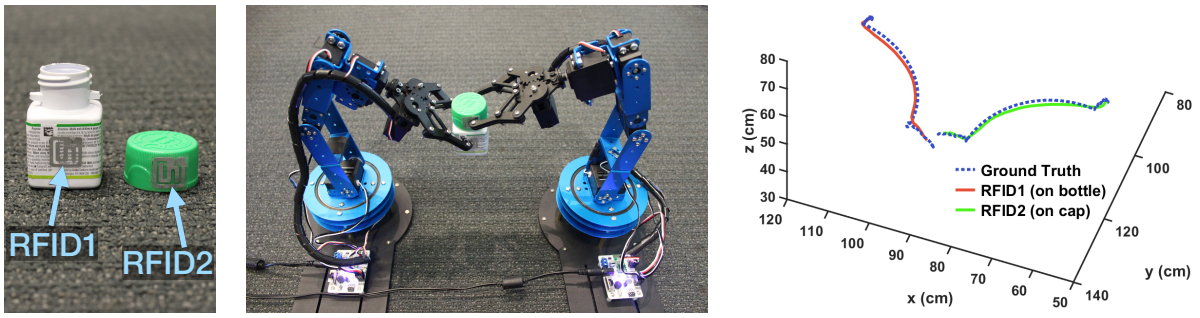


Figure 1: **TurboTrack in an Assembly Task.** Left figure shows a bottle and its cap tagged with RFIDs. Middle figure shows two robots collaborating on assembly, where one robot puts a cap on a bottle carried by another. Right figure shows TurboTrack's tracking output as the robots move the cap (in green) and bottle (in red) into position for assembly and compares it to a ground-truth vision-based system (in blue) to show its accuracy.

where a robot needs to move and manipulate the object of interest simultaneously.

The main contribution of this paper is to build an RF localization system that can achieve all the above requirements. Our system, TurboTrack, introduces two main innovations that together allow it to achieve the high accuracy and unconstrained mobility needed to deliver fine-grained robotic tasks:

(a) One-Shot Wideband Estimation: TurboTrack's first component allows it to estimate an RFID's response over a wide bandwidth – one that is three orders of magnitudes larger than the backscatter communication bandwidth. The large bandwidth can be used to mitigate reflections from other objects in the environment and isolate the RFID's response. This component is inspired by past work that performs frequency hopping for wideband estimation [45]. In contrast to this past work, which needs every RFID to remain static as it repeatedly queries it dozens of times while hopping frequencies, TurboTrack can estimate the wide bandwidth in one shot from every single RFID response. To do so, it introduces a wideband localization helper which acts like a radar. The helper transmits a wideband signal, and measures its reflection off different objects in the environment, including the RFID. In §3, we describe how TurboTrack constructs the helper's wideband signal to be compatible with the RFID protocol, and how it synchronizes the helper with an RFID reader to isolate the RFID's reflection and estimate its wideband channel from every single response. This one-shot estimation component enables TurboTrack to operate correctly with moving targets.

(b) Bayesian Space-Time Super-Resolution: In principle, if one could estimate multiple GHz of bandwidth off an RFID using the above technique, then we could apply standard ultra-wideband ranging methods to directly localize a tag. Unfortunately, the bandwidth that can be estimated from an RFID remains limited by the RFID's antenna response and impedance matching circuitry. Specifically, because of their designs that optimize for energy harvesting efficiency, the ability to sense an RFID's channel significantly degrades beyond a couple hundred MHz, even if we perform frequency hopping [45]. Such bandwidth is still an order of magnitude lower than that required for sub-centimeter localization using ultra-wideband techniques [56, 14].

TurboTrack's second innovation is a space-time super-resolution algorithm that overcomes this challenge. Its key insight is that while few hundred MHz of bandwidth cannot enable sub-centimeter positioning, they narrow down the potential locations of an RFID to a handful of candidates. By combining these candidates over space (multiple antennas) and time, TurboTrack zooms in on the exact location. In §4, we formalize this as a Gaussian mixture problem and incorporate it into a Bayesian framework that fuses spatio-temporal bandwidth measurements. Further, to deal with the nonlinear nature of the measurements, we introduce an approximate inference algorithm that exploits RF and geometric properties of the underlying estimators to design a computationally efficient solution for highly accurate positioning.

We built a prototype of TurboTrack using USRP X310 software radios and tested it with off-the-shelf, battery-free RFIDs. Our evaluation with over a million location measurements demonstrates that TurboTrack can achieve sub-centimeter localization accuracy in each of the x/y/z dimensions. TurboTrack's 99^{th} percentile error remains lower than 2-cm in each of the dimensions, while retaining a 99^{th} percentile latency smaller than 7.5 milliseconds in 3D localization. Further, we compared its performance to two state-of-the-art proposals, RFind [45] and RFIDraw [71]. Our results show that TurboTrack achieves two to three orders of magnitude improvement in localization accuracy of moving targets.

Finally, to demonstrate TurboTrack's capability in mobile and accurate positioning, we tested it in two classes of agile and fine-grained robotic tasks. First, we show how it can accurately track collaborative tasks between robotic arms including packaging, handover, and assembly. Second, we demonstrate how it can accurately track nanodrones as they fly in indoor environments.

Contributions. TurboTrack's contributions are:

- The first system architecture that performs one-shot

wideband estimation from every backscatter packet, enabling low-latency and high-precision localization.
- A spatio-temporal Bayesian framework for RF localization that fuses time series of bandwidth and phase measurements across multiple antennas.
- An approximate inference algorithm with fast convergence time for RF localization. The algorithm achieves computational efficiency by incorporating the properties of RF signals and the geometric nature of its measurements.
- A real-time prototype implementation and evaluation demonstrating the system's ability to track fine-grained robotic tasks performed using robotic arms and nanodrones.

We note that our current implementation inherits some of the limitations of RFIDs. Most importantly, its range of operation is limited by a reader's ability to power up battery-free tags, which is typically within less than 10 m. However, this limitation is not inherent to our design since both TurboTrack's architecture and algorithms are general to any backscatter sensor and do not stop at RFIDs. For example, they could work with battery-assisted or solar-powered tags, which would enable long-range communication [56, 55]. Such tags may be attached to nanodrones or robotic arms to track the robots themselves over tens to hundreds of meters and not just the items they manipulate.

2 Design Overview

TurboTrack is a system that enables ultra-low latency, very high accuracy localization of backscatter sensors for fine-grained robotic applications. TurboTrack's localization works both in line-of-sight and through occlusions. Further, TurboTrack can operate with inexpensive backscatter sensors like off-the-shelf RFIDs without requiring any hardware modifications, and it is fully compatible with today's standard UHF RFID protocol. The sensors can be attached as stickers to objects of interest (e.g., manufacturing items) for object manipulation or to miniature robots like nanodrones for tracking.

Architecturally, TurboTrack combines a standard RFID reader with a wideband localization helper as shown in Fig. 2. The helper transmits wideband signals, and captures their reflections off different objects in the environment. The combination of a standard reader with a wideband helper enables both identification (through the RFID's identifier) and accurate localization (using the wideband signals). To deliver both of these tasks, TurboTrack's centralized controller synchronizes the helper's signals with the RFID reader at the physical layer, and, at the protocol level, it incorporates the helper's operation into the reader's finite state machine.

Algorithmically, TurboTrack leverages the wideband channel estimates for localization. It fuses estimates

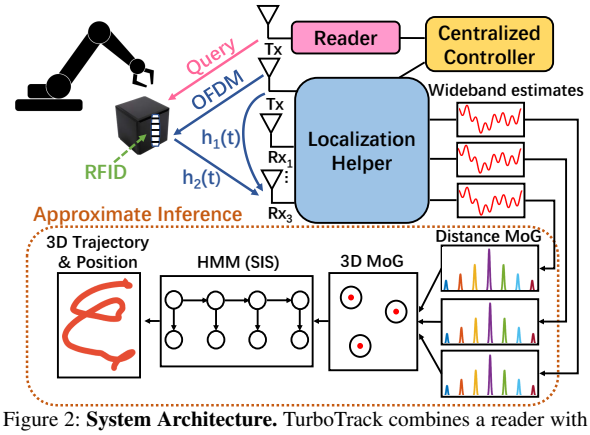


Figure 2: **System Architecture.** TurboTrack combines a reader with a localization helper to obtain wideband estimates. It fuses the estimates across space and time via a Bayesian framework, and solves for accurate 3D positions using approximate inference. MoG refers to Mixture of Gaussians, and SIS refers to Sequential Importance Sampling.

across multiple receive antennas of the localization helper and across time through a spatio-temporal Bayesian framework. It models each antenna's measurements as a Gaussian mixture, and solves for each tag's location and trajectory through an approximate inference algorithm customized for backscatter localization. The algorithm linearizes and approximates the antenna measurements in 3D and propagates their beliefs through a particle filter. The resulting accuracy is high enough to enable TurboTrack to track fine-grained robotic tasks.

The next sections describe TurboTrack's operation at the architectural (§3) and the algorithmic levels (§4).

3 One-Shot Wideband Estimation

In this section, we describe how TurboTrack's helper can obtain wideband estimates from every single (narrowband) RFID response. The one-shot estimation enables it to track changes in the channel at very high speeds. Then, in §4, we describe how it uses the wideband estimates for accurate localization.

3.1 Primer on Backscatter Modulation

In backscatter networking, a wireless device called a reader starts a communication session by sending a signal on the downlink. A backscatter sensor, e.g., an RFID, harvests energy from this signal and powers up. To communicate with the reader, the sensor switches between two states: reflective and non-reflective, to transmit bits of zeroes and ones.

Our past work has observed that backscatter modulation is frequency agnostic [45], meaning that RFIDs not only modulate the reader's signal but also all transmitted signals in the environment. This enables us to estimate an RFID's channel out-of-band. In particular, as an RFID backscatters the reader's signal, we can transmit an unmodulated wave at another sensing frequency and estimate the RFID's channel at that frequency. By hopping the sensing frequency across successive RFID

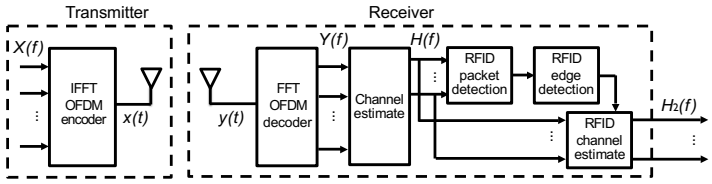


Figure 3: **One-shot Estimation Architecture.** The helper transmits OFDM symbols. The receiver demodulates these symbols by performing an FFT followed by an overall channel estimation step. The overall channel is fed into an RFID packet and edge detection module that allow discovering the RFID's state transitions. Using the output of the edge detection and elimination block, TurboTrack can extract reliable wideband channel estimates.

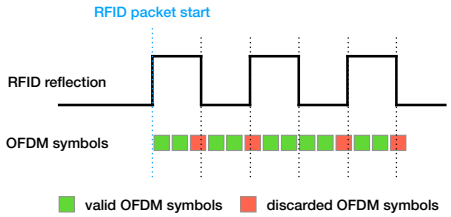


Figure 4: **Two Modulations.** RFIDs communicate by switching between reflective and non-reflective states. TurboTrack's localization helper constructs and decodes its symbols to accommodate the backscatter switching.

responses, we can emulate a large bandwidth – 1000× larger than typical RFID backscatter bandwidths of few 100 kHz. But, such an approach needs to query the same RFID many times before it can sense a wide bandwidth. Thus, it requires the RFID to remain static, reducing the reader's throughput and increasing the tracking latency.

TurboTrack's first component focuses on estimating a wide bandwidth in a single shot, i.e., from every single RFID response. In principle, one could do that by simply transmitting a wideband signal from the localization helper simultaneously with the reader, and estimating the channel across its bandwidth. However, this approach is complicated by two main factors. First, the RFID's switching process introduces a fast fading channel for the wideband signals; ignoring such fading corrupts the wideband estimates. Second, by spreading its transmitted signal over a wide bandwidth rather than a single frequency, the helper's signal-to-noise ratio (SNR) significantly degrades, which limits its ability to detect the RFID's response and precludes channel estimation.

The rest of this section describes how TurboTrack's helper overcomes these challenges. On the transmit side, the process involves constructing backscatter-aware wideband transmissions. And on the receive side, it involves robustly detecting the RFID's response for accurate channel estimation.

3.2 Backscatter-Aware Wideband Transmissions

To simplify channel estimation over a wide bandwidth, TurboTrack borrows the OFDM (Orthogonal Frequency Division Multiplexing) modulation technique from WiFi and LTE systems. At a high level, OFDM divides a wideband channel into an array of narrowband channels, and performs modulation in the frequency domain.

For the purpose of this paper, we do not need to delve into the details of how OFDM operates beyond the FFT and IFFT blocks of Fig. 3. Specifically, an OFDM modulator encodes information in the frequency domain as $X(f)$ then takes an IFFT before transmitting the signal over the air. The receiver demodulates the signal by taking an FFT as shown in Fig. 3, and can estimate the channel $H(f)$ by dividing the FFT's output by $X(f)$.

So, how can we construct OFDM symbols to be friendly with backscatter modulation? To see why OFDM construction is important, consider Fig. 4 which shows both the backscatter modulation and the OFDM symbols over time. OFDM channel estimation assumes that the entire OFDM symbol lies within a channel coherence time (i.e., that the channel does not change during the estimation process). However, as a backscatter sensor switches its impedance, it causes an extremely fast-fading channel and corrupts the entire OFDM channel estimate. Hence, if we choose long OFDM symbols, then all of them will be corrupted with the backscatter switching process. On another hand, if we choose very short symbols, we cannot pack many frequencies into them, which would prevent us from estimating the wideband channel at sufficient frequency resolution to deal with frequency selectivity.

To address this challenge and obtain both non-corrupted and fine-grained wideband estimates, TurboTrack's helper exploits information from the RFID reader about the backscatter switching rate. Specifically, the reader communicates that rate (called backscatter link frequency or BLF) in its downlink command to the backscatter sensor. Hence, by coordinating with the RFID reader, the localization helper can use the BLF to construct its OFDM symbols.

In particular, to ensure that the channel is not corrupted, an OFDM symbol must lie within a specific RFID reflection mode (i.e., transition-free region). Hence, the OFDM symbol duration T_{symbol} must be smaller than half the period of an RFID switching period $T_{switching}$:

$$T_{symbol} < \frac{T_{switching}}{2} = \frac{1}{2BLF}$$

Knowing that an OFDM symbol consists of N samples (i.e., N subcarriers), each with time period T_{sample}, and denoting the overall bandwidth of the helper's OFDM transmission B, this means that we should choose N such that $N < \frac{B}{2BLF}$. For example, if the helper's OFDM bandwidth is 100MHz and the backscatter link frequency is 500 KHz, N must be less than 100.

3.3 Robust Wideband Channel Estimation

Now that we know how the localization helper constructs its transmitted symbols, we switch our focus to how it

can obtain robust channel estimates on the receive side. Recall that the difficulty in wideband estimation arises from the low SNR at each subcarrier since the power is spread across frequencies. This complicates packet detection[2] and reduces the reliability of the sensed channel.

(a) Robust Packet Detection: First, to compensate for the reduced power and robustly estimate the beginning of a backscatter packet, TurboTrack exploits the frequency agnostic property of backscatter modulation. In particular, since the different OFDM subcarriers undergo the same backscatter modulation, we can incoherently average their estimates. Such averaging would enable us to reliably observe changes in the overall reflected power and use them to detect the beginning of the RFID response. Specifically, for every OFDM symbol at time n, we compute $H_{combined}(n) = \sum_f |H(f,n)|$.

Next, we leverage our knowledge of the RFID packet's preamble to detect the packet start. Specifically, every RFID packet payload is preceded by a known preamble $p(n)$. Hence, the localization helper correlates the averaged channel estimates $H_{combined}(n)$ with the preamble to detect packet start. We can write this correlation as:

$$D(\Delta) = \sum_{n=1}^{T} p^*(n) H_{combined}(n+\Delta)$$

where T is the preamble length and Δ is the time instance where correlation is performed. The helper identifies the packet beginning when D rises above a threshold.

(b) Edge Flip Elimination: Next, TurboTrack proceeds to eliminating corrupted OFDM symbols. Recall from §3.2 that TurboTrack constructs the OFDM symbols to accommodate for the backscatter reflection rate. While this ensures that at least one whole symbol is in a reflective or non-reflective state, it does not ensure that all OFDM symbols are non-corrupted.

By leveraging the knowledge from the previous step – namely when an RFID packet starts as well as the RFID's switching frequency – TurboTrack's localization helper can automatically detect and discard erroneous channel estimates (marked in red in Fig. 4). In doing so, it only retains the channel estimates that are obtained when the RFID is reflecting or not reflecting, while eliminating estimates corrupted by RFID state transitions.

(c) Channel Estimation: Now that we have eliminated erroneous OFDM channel estimates, we can proceed to estimating the RFID channels at each of the subcarriers. Note that the OFDM channel estimates $H(f)$ not only consist of the RFID's reflection but also the direct path between the helper's transmit and receive antennas as well as other reflections in the environment. To estimate the RFID's channel, TurboTrack exploits that the difference between the reflective and non-reflective states is due to the RFID, and subtracts them from each other to obtain the RFID's channel.

Mathematically, assume that after discarding the erroneous channel estimates from the preamble, the helper is left with L symbols where the RFID is non-reflective and M symbols where it is reflective, we can estimate these channels at each subcarrier f as:

$$\widehat{H_2}(f) \propto \frac{1}{L}\sum^{L} H(f|\text{reflective}) - \frac{1}{M}\sum^{M} H(f|\text{non-reflective})$$

Note that in the above equation, we only average over the reflective states of a *single* RFID response, enabling one-shot wideband estimation.

Finally, to improve the efficiency of the wideband estimation process, TurboTrack incorporates the localization helper into the finite state machine of the RFID reader. In doing so, the helper only needs to perform OFDM processing (packet detection, edge elimination, etc.) over the short interval of time during which it expects the RFID's response rather than over the entire duration of the reader's communication session. Specifically, the receiver opens a short time window immediately after the reader finishes transmitting its query command. In our implementation, this window is 300μs-long in comparison to the 2ms-long communication session. As a result, this synchronous architecture saves significant computational resources (by $6.6\times$), allowing TurboTrack to achieve ultra-high frame rates and ultra-low latency.

A few additional points are worth noting about TurboTrack's use of OFDM for channel estimation:

- Because the helper's transmitter and receiver are connected to the same oscillator, the estimated channels do not have any carrier frequency offset (CFO) or sampling frequency offset (SFO). Hence, unlike WiFi or LTE, we do not need to correct for them.
- Since the helper transmits the same OFDM symbol back-to-back to estimate the channel, each OFDM symbol acts a cyclic prefix for the subsequent one.
- Since the helper's transmit and receive antennas are co-located, the line-of-sight would dominate the channel estimate, and we do not expect any sampling offset (or packet detection delay) between the transmitted and received OFDM symbols. To correct for any sample offsets introduced by the hardware channel, we perform a time-domain correlation that allows us to detect the beginning (i.e., first sample) of the first OFDM symbol. Moreover, we perform a one-time calibration with a known RFID location in order to eliminate other over-the-wire hardware channels.
- Similar to standard OFDM receivers, the localization helper drops the DC subcarrier as it is less robust to noise, and since dropping it improves the dynamic range of the ADC (Analog-to-Digital Converter).
- To further improve its signal-to-noise ratio (SNR), TurboTrack employs an Exponential Moving Average

[2]The helper cannot simply rely on the reader's packet detection, since the reader uses much lower bandwidth and lower sampling rate.

(EMA) on the channel estimates. The EMA provides more robust channel estimates and higher accuracy without sacrificing frame rate.

4 Bayesian Space-Time Super-Resolution

So far, we have discussed how TurboTrack can estimate an RFID's channel over a wide bandwidth. In this section, we describe how it uses this wide bandwidth to estimate and track an RFID's 3D location.

4.1 Bootstrapping Localization

Before we describe TurboTrack's localization algorithm, we start by asking whether backscattering a large bandwidth would be sufficient for precise localization. In principle, if one could backscatter a very large bandwidth off an RFID, then we could directly use that bandwidth to localize the tag in a manner similar to ultra-wideband (UWB) ranging systems [58]. In particular, UWB systems leverage their large bandwidth to compute the time-of-flight – i.e., the time it takes their signals to travel between a transmitter and a receiver; they then map this time-of-flight to the distance traveled by multiplying it by the speed of propagation. Because time and frequency are inversely related, their distance resolution is inversely proportional to their bandwidth:

$$\text{resolution} = \text{speed}/\text{bandwidth}$$

Since RF signals travel at the speed of light, obtaining sub-centimeter resolution using UWB ranging would require a bandwidth of 30 GHz.

Unfortunately, backscattering 30 GHz off RFID tags is neither feasible nor desirable for multiple reasons. First, backscatter devices have antennas and impedance matching circuits that are optimized to harness energy within a specific frequency band. Hence, signals backscattered significantly outside their optimal frequency band have very poor SNR, making channel estimation infeasible. Second, even if one could backscatter such a wide bandwidth off the tags, generating it would require very costly RF radios and processing its reflections would be compute intensive as it would require processing 30 GS/s.

To achieve higher accuracy without such a large bandwidth, one could leverage the phase of the backscattered signal since it is very sensitive to changes in distances. Specifically, in the noiseless case, we can express the phase ϕ as:[3]

$$\phi = 2\pi\frac{d}{\lambda} \mod 2\pi$$

where d is the distance traveled and λ is the wavelength.

The difficulty in using the phase, however, is that it wraps around every wavelength. Hence, it allows us to accurately recover a fractional distance d_{frac} modulo the wavelength. Said differently, we would know that the actual distance is $d_{frac} + n\lambda$, but n is an unknown integer.

[3] In presence of multipath, we can use a large bandwidth to obtain "sanitized phases" [45] by projecting on the direct path after identifying it.

Since TurboTrack obtains a wide bandwidth and the phase from every antenna, it can combine them to narrow down the candidate locations to a handful. For example, if we consider a bandwidth of 100 MHz and a wavelength of 33 cm, this leaves us with only $resolution/\lambda = 9$ potential candidate locations for the tag. Hence, TurboTrack still needs a mechanism to resolve ambiguity and identify the correct candidate.

4.2 Bayesian Formulation

To localize tags, TurboTrack employs a Bayesian framework that fuses measurements across space and time:

- *Spatially,* each distance candidate maps to an ellipse whose foci are the transmit and receive antennas. This is because TurboTrack's antennas measure the round-trip distance from the transmitter to the backscatter tag, and back to the receiver. Since a given transmit-receive pair has multiple distance candidates, this leads to multiple confocal ellipses as shown in Fig. 5(a). Adding more receive antennas would create other sets of ellipses. However, due to noise, we do not expect the correct ellipses from all antennas to intersect at the same point; hence, we cannot simply rely on voting to identify the correct candidate.
- *Temporally,* as the tag moves, each of the candidate locations traces a different trajectory. The intuition of using temporal series is that because noise is random, we expect the intersection points that correspond to the actual location to be closer to each other across time, providing opportunities to identify them.

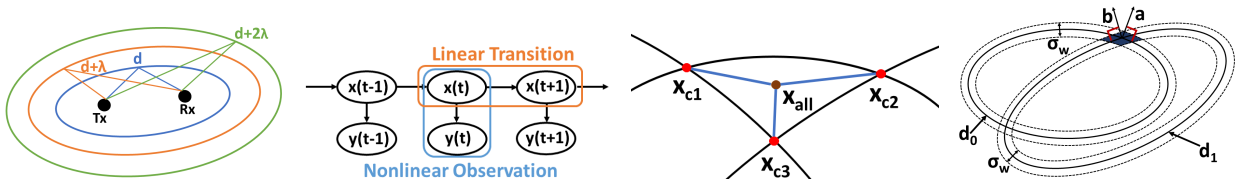
Hidden Markov Model. Given the above intuition, we formulate the localization problem as a Hidden Markov Model (HMM), as shown in Fig. 5(b). HMMs form a class of powerful Bayesian inference models with hidden states and observed variables. In our context, the hidden states correspond to the actual locations of the RFID over time, and the observations are the candidate distances obtained from the different receive antennas.

Most importantly, TurboTrack's HMM has a nonlinear Gaussian observation model and a linear Gaussian transition model: the distance observations are nonlinear in the state variables (the coordinates) as can be seen with the $dist$ function, while state transitions are linearly related due to motion. Unfortunately, the nonlinearity prevents us from adapting common solutions like Kalman Filters to model the distributions [35].

TurboTrack's goal is to find the most likely trajectory $\mathbf{x_1} \ldots \mathbf{x_T}$ given the observations (distance candidates) $\mathbf{y_1} \ldots \mathbf{y_T}$. Formally, it needs to solve for the maximum aposteriori (MAP):

$$\mathbf{x}^*_{\mathbf{0:T}} = \arg\max_{\mathbf{x_{0:T}}} p(\mathbf{x_{0:T}} \mid \mathbf{y_{0:T}})$$

where $\mathbf{x_t}$ is a d dimensional coordinate vector of the tag, $\mathbf{y_t}$ is a k dimensional vector of the estimated dis-



(a) Confocal Ellipses (b) HMM (c) Centroid (d) Linearization

Figure 5: **TurboTrack exploits RF and geometric properties for estimation.** (a) shows that distances map to confocal ellipses. (b) shows the HMM with nonlinear observations. (c) shows centroid approximation. (d) shows geometric linearization

tances from each receiver, and $p(x|y)$ denotes the posterior probability distribution.

Solving the above MAP inference problem requires us to model the likelihood function $p(\mathbf{y_i} \mid \mathbf{x_i})$ and the prior $p(\mathbf{x_{0:T}})$. In particular, per Bayes' rule, we can write the posterior as: $p(\mathbf{x_{0:T}} \mid \mathbf{y_{0:T}}) \propto p(\mathbf{y_{0:T}} \mid \mathbf{x_{0:T}}) p(\mathbf{x_{0:T}})$

Next, we describe how we model the likelihood function and prior given the geometric nature of the problem and the underlying wireless properties of the estimators.

Likelihood Function. We first model TurboTrack's likelihood function. Recall from §4.1 that the distance from a given transmit-receive antenna pair exhibits as multiple candidates. We formulate the likelihood as a mixture of Gaussian (MoG), where each Gaussian is centered at a different integer wavelength. In particular, given the true distance d, we approximate the distance estimates d_{est} with the following distribution:

$$z(d_{est}|d;\sigma_w) = \sum_{i=-N}^{N} w(i)\mathcal{N}(d+i\lambda,\, \sigma_w^2) \qquad \sum_{i=-N}^{N} w(i) = 1 \tag{1}$$

where N is determined by the total number of candidates, i is integer wavelength, and σ_w corresponds to the standard deviation of Gaussian noise. The weights $w(n)$ denote the discrete distribution of different integers.[4]

The above formulation models the likelihood distribution for one receiver. Since TurboTrack employs multiple receivers, each provides a different set of distance estimates. Because estimates from different antennas are independent, the overall likelihood $p(\mathbf{y_t} \mid \mathbf{x_t})$ is a product of those from different receivers:

$$p(\mathbf{y_t} \mid \mathbf{x_t}) = \prod_{j=1}^{k} z(\mathbf{y_t}[j] \mid dist(\mathbf{x_t})[j]; \sigma_w) \tag{2}$$

where $dist : \mathbf{R^d} \rightarrow \mathbf{R^k}$ returns the round trip distance to the RFID, k denotes the total number of receivers, and j is the index of each receiver.

Transition Model. Next, we model the prior $p(\mathbf{x_{1:T}})$. Since the prior consists of a Markov time series of the RFID's location, we use a simple transition model:

$$\mathbf{x_t} = \mathbf{x_{t-1}} + \mathbf{v_t}, \mathbf{v_t} \sim \mathcal{N}(\mathbf{0},\, \Sigma_v) \tag{3}$$

where $\mathbf{v_t}$'s correspond to linear position changes, drawn from a Gaussian distribution with zero mean and covariance matrix Σ_v. Together, Eqs. 2 and 3 form our HMM.

4.3 Approximate Inference via Particle Filters

Due to the nonlinear observation model, we cannot solve the MAP problem analytically. So, TurboTrack resorts to approximate inference solutions, specifically particle filters. Instead of propagating beliefs through analytical probability distributions, these models approximate distributions with a set of particles [30]. Approximating distributions with particles requires us to answer two main questions: First, how can we choose particles so that their discrete distributions accurately represent the analytical distributions? And second, how can we update these particles with new observations (distance candidates)?

In the rest of this section, we describe how TurboTrack's algorithm exploits the RF and geometric structures of our measurements to answer these questions and develop a computationally efficient solution. For simplicity, our exposition focuses on 2D localization using three antennas, but the technique can naturally generalize to 3D with an arbitrary number of antennas.

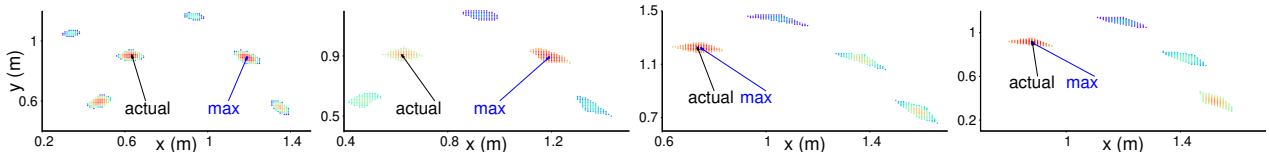
4.3.1 Initial Sampling Function

Standard particle filters bootstrap the sampling process by populating the entire d-dimensional space with exponentially many particles [63]. In contrast, TurboTrack leverages its first observation to bootstrap its inference with a relatively small number of particles. In particular, recall from §4.1 that each receive antenna discovers a finite number of distance candidates, which can be mapped to a set of ellipses in 2D as shown in Fig. 5(a).

To obtain an initial set of particles, one option is to consider all intersection points between ellipses from different antennas. But, this is undesirable for three reasons. First, any such two-way intersection point leaves out important information about distance measurements from other receive antennas. Second, it would result in a polynomially large number of intersection points.[5] Third, such a set of intersection points would not be representative of the distribution as they are sparsely distributed rather than concentrated at the most likely locations.

[4] These weights could correspond to the value of the fractional Fourier transform or a MUSIC projection [32]. In our implementation, we found the overall algorithm performance is the same in both cases, so we used the fractional Fourier as it is less computationally expensive.

[5] The total number of intersection points is the number of distance candidates raised to the power of the number of receive antennas.



(a) Initial Sampling (t=0) (b) t = 1s (c) t = 2s (d) t = 3s

Figure 6: **TurboTrack's Particle Filter.** We show the particles in 2D space over time. Colors indicate particle weights, with red being highest and blue lowest. We label the actual location and that with highest weight. Over time, the highest weight particle converges to the correct one.

Instead, TurboTrack first identifies the most likely intersection points then uses them to sample particles that are representative of the distribution. At a high level, it exploits the fact that each antenna's distance measurements can be modeled as a Mixture of Gaussians as per §4.2 in order to (1) prune out unlikely candidates, and (2) linearize the intersection points into a mixture of 2D Gaussians. Subsequently, it can sample from a mixture of 2D Gaussians then adapt the particle weights as per standard importance sampling [62].

Step 1: Pruning. In the pruning step, TurboTrack uses the distances between intersection points to identify the most likely candidates. In particular, given a tuple of distance measurements from three antennas, $[d_0, d_1, d_2]$, we can plot three ellipses and identify the three closest intersection points between them as shown in Fig. 5(c).[6] In the noiseless case, the three points coincide. Otherwise, we compute the average distance between them and discard all points whose average is above a threshold.

Formally, if we denote the intersection points as $\{\mathbf{x^s_{01}}, \mathbf{x^s_{12}}, \mathbf{x^s_{02}}\}$. TurboTrack will only keep tuple $\mathbf{s}$ [7] in the set $C \coloneqq \{\mathbf{s} \mid r(\mathbf{s}) \leq T\}$ where

$$r(\mathbf{s}) \coloneqq \frac{1}{3}(\|\mathbf{x^s_{01}} - \mathbf{x^s_{12}}\|_2 + \|\mathbf{x^s_{01}} - \mathbf{x^s_{02}}\|_2 + \|\mathbf{x^s_{12}} - \mathbf{x^s_{02}}\|_2)$$

Step 2: Geometric Linearization. Next, TurboTrack approximates each intersection point as a 2D Gaussian using the original distance distributions of the corresponding ellipses. To see why this linearization is possible, consider two intersecting ellipses in Fig. 5(d), whose intersection point we denote as $\mathbf{x^s_{01}}$. Since the noise on the distance measurements is much smaller than the size of the ellipses, the curvature of the ellipses around the intersection point is relatively flat. This allows us to approximate the distribution as a 2D Gaussian centered around $\mathbf{x^s_{01}}$ and whose axes lie along the normals to the two ellipses at the intersection point.

Formally, if we denote the two axes by the normal vectors: $\mathbf{a}, \mathbf{b}$, and define $A = [\mathbf{a}, \mathbf{b}]$, then we obtain R_{ij} as a Gaussian distribution in the 2D coordinate system:

$$R_{ij} \sim \mathcal{N}(\mathbf{x^s_{01}}, \Sigma^s_{01} = AVA^T) \text{ s.t. } V = \begin{pmatrix} \sigma_w^2 & 0 \\ 0 & \sigma_w^2 \end{pmatrix} \quad (4)$$

where V denotes the diagonal covariance matrix, which represents the fact that the distance errors from the different antennas are independent.

Step 3: Centroid Approximation. So far, we have identified the most likely tuples of intersection points, and we have linearized each of the intersection points as a 2D Gaussian. Next, TurboTrack fuses every surviving tuple into a single candidate. This step is important because it combines measurements from all antennas (while the previous step considered antennas only in pairs). To do so, it approximates any given candidate $\mathbf{x_0}$ as the centroid of its three corresponding intersection points (from Step 1). For simplicity, we assume mutual independence between the intersection points, which results in the following Gaussian distribution $f(\mathbf{x_0}; \mathbf{y_0}, \mathbf{s})$:

$$\begin{aligned} f(\mathbf{x_0}; \mathbf{y_0}, \mathbf{s}) &\sim \mathcal{N}(\mu_\mathbf{s}, \Sigma_\mathbf{s}) \\ \mu_\mathbf{s} &= \frac{1}{3}(\mathbf{x^s_{01}} + \mathbf{x^s_{12}} + \mathbf{x^s_{02}}) \\ \Sigma_\mathbf{s} &= (\frac{1}{3})^2(\Sigma^s_{01} + \Sigma^s_{12} + \Sigma^s_{02}) \end{aligned} \quad (5)$$

This provides us with a set of 2D Gaussian distributions, each centered at a likely candidate position.

Step 4: Weighting and Initial Sampling. In the final step, we combine the set of Gaussian distributions into a single 2D mixture of Gaussians. To do so, we assign weights proportional to the product of weights of individual integers to tuples in C. This provides us with the initial sampling distribution $q_0(\mathbf{x_0}; \mathbf{y_0})$:

$$\begin{aligned} q_0(\mathbf{x_0}; \mathbf{y_0}) &= \sum_{\mathbf{s} \in c} \frac{w(\mathbf{s})}{w} f(\mathbf{x_0}; \mathbf{y_0}, \mathbf{s}) \\ w(\mathbf{s}) = w(s_0)w(s_1)w(s_2), w &= \sum_{\mathbf{s} \in c} w(\mathbf{s}) \end{aligned} \quad (6)$$

Given this distribution, we can now efficiently sample a small number of particles $\{\mathbf{x_0^{(i)}}\}$ and use them to approximate the distribution. After sampling, we normalize every particle's weight to the total weights.

Fig. 6(a) shows an example output of initial sampling in 2D space. Each particle's color indicates its normalized weight, where red and blue indicate highest and lowest probability. The particles are concentrated in several clusters which correspond to the fused intersection points. While the actual location is in one of the clusters, the one with the highest weight is in another cluster (due to noise). This shows the power of fusing across antennas and emphasizes the need to resolve ambiguity by exploiting target motion and fusing over time.

[6] Three ellipses can have at most six intersection points, from which we choose the closest three.

[7] $\mathbf{s}$ is integer tuple $[s_0, s_1, s_2]$ for $[d_0 + s_0\lambda, d_1 + s_1\lambda, d_2 + s_2\lambda]$.

4.3.2 Sequential Sampling

So far, we have described how TurboTrack can obtain an initial set of particles by fusing RF observations from multiple antennas at a single point in time $t = 0$. Next, we discuss how TurboTrack can update its particles through new observations over time. To do so, it adapts Sequential Importance Sampling (SIS), a well-known particle update filter, to our problem domain.

As its name indicates, SIS operates through a sequential process. To select representative particles for the probability distribution at some time t, it uses the (new) observations from time t and the probability distribution from the previous state $t-1$. Hence, for every particle i, it is optimal to sample from the following distribution [15]:

$$p(\mathbf{x_t} \mid \mathbf{x_{t-1}^{(i)}}, \mathbf{y_t}) \propto p(\mathbf{y_t} \mid \mathbf{x_t}) p(\mathbf{x_t} \mid \mathbf{x_{t-1}^{(i)}})$$

Recall that we have already modeled both terms on the right hand side. In particular, we know that $p(\mathbf{x_t} \mid \mathbf{x_{t-1}^{(i)}})$ is a Gaussian as per the motion model in Eq. 3. Moreover, $p(\mathbf{y_t} \mid \mathbf{x_t})$ can be approximated as a Gaussian as per Eq. 5. Because the product of two Gaussians is Gaussian, the overall sampling function is both Gaussian and approximately optimal.[8]

Now that we know the sampling distribution for a given particle i, SIS samples a single particle from that distribution for time t. It then repeats the same process for all the particles. After obtaining all the samples at time t, it normalizes their weights then moves to the next time step $t + 1$ and repeats the same process.

Few additional points are worth noting about TurboTrack's inference algorithm:

- *MAP Inference:* At every point in time, TurboTrack can make a decision about the most likely location by choosing the particle with the highest weight. It can also update the past trajectory since every particle can backtrack its entire history. This allows us to continuously update the trajectory with new observations.
- *Resampling:* It is known that even with optimal approximate sampling functions, the SIS algorithm can degenerate to a small number of particles [30, 38]. To avoid degeneracy, TurboTrack follows a standard resampling approach. In particular, at every time t, it estimates the effective sample size $\widehat{N_{eff}} = \frac{1}{\sum_{i=1}^{N} (w_t^{(i)})^2}$, and resamples when it is lower than a threshold.
- *Computational Efficiency:* In the sample update step, TurboTrack exploits that many of the particles i come from the same Gaussian in the Gaussian mixture because they share the same tuple of intersection points. Hence, to optimize computation, it computes the sequential sampling function once for each cluster and reuses it for all particles in that cluster.
- *Outlier Detection & Recovery:* The SIS filter accounts for Gaussian noise but does not incorporate mechanisms to deal with wireless interference or strong leakage from multipath.[9] To deal with such scenarios, TurboTrack employs outlier detection and recovery. We identify two scenarios for outlier detection. The first is when the centroid at t is far from that at $t - 1$. The second is when all particles are assigned near-zero weights, indicating that there is no valid position given current distance estimation. To recover from such outliers, TurboTrack leverages its high frame rate and extrapolates the previous estimates.

Finally, Alg. 4.1 summarizes TurboTrack's approximate inference algorithm and Fig. 6 shows how the algorithm converges to the right candidate over time.

Algorithm 4.1 TurboTrack's Approximate Inference

Input: Distance estimations: $\mathbf{y_{0:T}}$

Initialization: At time $t = 0$

▷ Compute mixture of Gaussian function $q_0(\mathbf{x_0}; \mathbf{y_0})$
- (a) Prune unlikely intersection points of $\mathbf{y_0}$
- (b) Linearize unpruned tuples as Gaussians per Eq. 4
- (c) Approximate Gaussians per Eq. 5
- (d) Assign weights to tuples per Eq. 6

▷ Sample and assign weights

For $i = 1, ..., N$:
- (a) Sample $\mathbf{x_0^{(i)}}$ from $q_0(\mathbf{x_0}; \mathbf{y_0})$
- (b) Assign weights: $\hat{w_0}^{(i)} = p(\mathbf{y_0} \mid \mathbf{x_0^{(i)}})$

▷ Normalize weights: $w_0^{(i)} = \hat{w_0}^{(i)} / \sum_{j=0}^{N} \hat{w_0}^{(j)}$

Iteration: For time $t = 1, ..., T$

For $i = 1, ..., N$:

▷ Compute Gaussian function $f(\mathbf{x_t}; \mathbf{y_t}, \mathbf{s^{(i)}})$
- (a) Identify the tuple $\mathbf{s^{(i)}}$ corresponding to the intersection point[10] closest to $\mathbf{x_{t-1}^{(i)}}$
- (b) Linearize tuple as per Eq. 4
- (c) Approximate Gaussian as per Eq. 5

▷ Sample and assign weights
- (a) Sample $\mathbf{x_t^{(i)}}$ from $\pi(\mathbf{x_t} \mid \mathbf{x_{t-1}^{(i)}}, \mathbf{y_{0:t}}) \propto p(\mathbf{x_t} \mid \mathbf{x_{t-1}^{(i)}}) f(\mathbf{x_t}; \mathbf{y_t}, \mathbf{s^{(i)}})$
- (b) Add to history: $\mathbf{x_{0:t}^{(i)}} = (\mathbf{x_{0:t-1}^{(i)}}, \mathbf{x_t^{(i)}})$
- (c) Assign weights:

$$\hat{w_t}^{(i)} = w_{t-1}^{(i)} \frac{p(\mathbf{y_t} \mid \mathbf{x_t^{(i)}}) p(\mathbf{x_t^{(i)}} \mid \mathbf{x_{t-1}^{(i)}})}{\pi(\mathbf{x_t^{(i)}} \mid \mathbf{x_{t-1}^{(i)}}, \mathbf{y_{0:t}})}$$

▷ Normalize weights: $w_t^{(i)} = \hat{w_t}^{(i)} / \sum_{j=0}^{N} \hat{w_t}^{(j)}$

▷ Compute effective sample size $\widehat{N_{eff}}$.

If $\widehat{N_{eff}} < N_{thres}$, resampling based on $w_t^{(i)}$

Output: MAP Trajectory: $\mathbf{x^*_{0:T}} = \arg\max_{\left\{\mathbf{x_{0:t}^{(i)}}, w_{0:t}^{(i)}\right\}} w_{0:t}^{(i)}$

[8] Note that $p(\mathbf{y_t} \mid \mathbf{x_t})$ is a mixture of Gaussians as per Eq. 6, but we approximate it to the Gaussian nearest to $\mathbf{x_t^{(i)}}$ in the mixture.

[9] Recall that we mitigate (but not eliminate) multipath by projection.

[10] Recall that the intersection point can be computed from $\mathbf{y_t} + \mathbf{s^{(i)}}\lambda$.

5 Implementation & Evaluation

Localization Helper: We implement TurboTrack's localization helper on USRP X310 software radios, all synchronized to the same external clock [11]. We use three USRPs with two UBX daughterboards each. Since each USRP can support two chains, we use one chain at a transmitter and four as receivers. Each transmit/receive chain is connected to a circularly polarized patch antenna [8]. All antennas are arranged in a single plane, with a separation of 40 cm between any two adjacent pairs. USRPs sample at 100 MSps and send data over Ethernet to a computer using the Intel Converged Network Adapter X520-DA2 [3] to support high data rates. The computer runs Ubuntu 16.04 and has an 4-core 8-thread 64-bit Intel Core i7 processor and 16GB RAM.

RFID Reader: We adapt a USRP RFID reader developed by past work [36] and implement it on USRP N210 software radios [12] with SBX daughterboards. The reader implements the EPC Gen2 protocol [2]. The reader powers up and communicate with the RFID tags, setting various parameters such as the BLF and tag selection in multi-tag scenarios.

Real-time Processing. We implement the algorithms described in §3-§4 directly into the USRP's UHD driver in C++. Our implementation consists of a multi-threaded pipelined architecture with worker pools. On the transmit side, we use two threads in total: one for the RFID reader and another one for the OFDM transmitter. On the receive side, we use one thread for extracting buffer-sized packets from the USRPs. In addition, every antenna has a worker pool of two threads; each worker extracts and processes individual RFID responses from the shared buffer, computes and timestamps the wideband estimates, and feeds them to an output buffer. Finally, we aggregate the threads, combining the wideband estimates from all the antennas to perform 3D localization.

Our implementation runs in real-time, transmitting and receiving OFDM symbols, decoding, estimating channels, performing super-resolution and localization ($\Sigma_v = 5mm$)[11]. Our OFDM symbols use $N = 20$ subcarriers over a bandwidth of 100 MHz to increase resilience to RFID state-transitions. In contrast, the RFID BLF is set to 40 kHz.

Baselines: We implemented two baselines:

- *RFind* [45]: represents a state-of-the-art system for centimeter-scale RFID positioning. It emulates a large bandwidth by frequency hopping. As per the implementation described in the original paper, the hopping process takes few seconds (for channel acquisition). We reproduced the implementation and wrote the code directly into the UHD driver of the USRP (as we did for TurboTrack).
- *RF-IDraw* [71]: represents a state-of-the-art system for high-accuracy RFID tracking.[12] As per the authors evaluation, RF-IDraw can achieve high tracking accuracy but decimeter-scale accuracy in exact positioning. It combines various antenna patterns. We faithfully reproduced the authors' implementation on Thingmagic m6e [6] RFID readers.

RFID tags: We evaluated TurboTrack with commercial, off-the-shelf, passive UHF RFID tags. We tested it with multiple tags including Avery Dennison AD-238u8 [1], Alien Squiggle [17], and Smartrac [10]. Each tag costs about 5-10 cents. Because our tags are vertically polarized, we program the robots to maintain their orientation to minimize phase changes from orientation changes (for both TurboTrack and the baselines). Alternatively, one could use circularly polarized tags to avoid this problem, or incorporate an orientation-phase model into TurboTrack's inference algorithm.

Robots: We tested TurboTrack with three kinds of moving robots. For 2D evaluation, we attached an RFID to a Roomba [4]. For 3D evaluation, we attached the RFID to an item carried by LeArm 6DoF robotic arms [5] (as shown in Fig. 1) and to A20 minidrones from Potensic [7] (as shown in Fig. 12).

Ground truth: We use the OptiTrack system [9] to obtain ground truth location measurements. The OptiTrack is an optical tracking system which consists of an array of tripod-mounted infrared cameras that can achieve millimeter-scale accuracy by relying on infrared-reflective markers placed on the objects of interest. Since the OptiTrack can only operate in line-of-sight, in our non-line-of-sight evaluation, we ensure that while the RFIDs of interest are occluded from our antennas, they remain in LOS of the OptiTrack cameras.

Evaluation Environment: We evaluated TurboTrack in a standard office building, fully furnished with tables, chairs, computers. We tested it in both line-of-sight and non-line-of-sight settings, where RFIDs are within $6m$ of the antennas. We performed NLOS testing similar to past work [45, 71] by blocking the visible LOS path between an RFID and TurboTrack's antenna using standard office cubicle dividers made of wood.

[11] In principle, one could learn the HMM parameters, but setting a constant standard deviation worked well for our applications.

[12] Mathematically, RF-IDraw's tracking algorithm is similar to Tagoram's [75] method for tracking movement with unknown track and both systems achieve comparable tracking and localization accuracy.

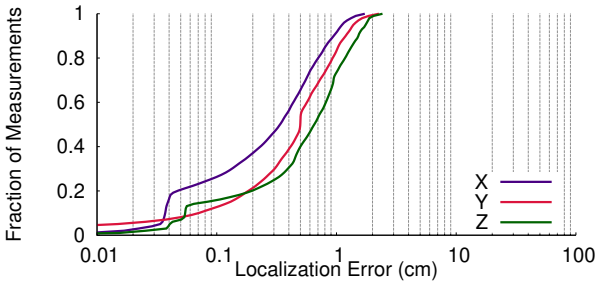


Figure 7: **3D Localization Accuracy in Line-of-Sight.** The figure plots the CDF of TurboTrack's localization error in LOS along each of the x/y/z dimensions.

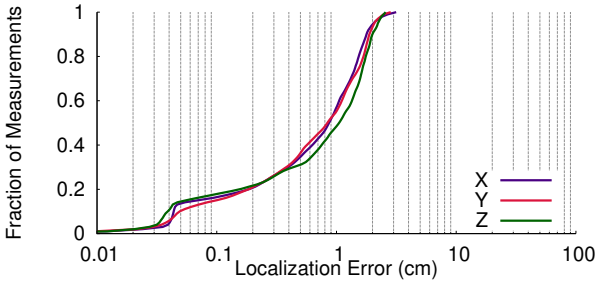


Figure 8: **3D Localization Accuracy in Non-Line-of-Sight.** The figure plots the CDF of TurboTrack's localization error in NLOS in the x/y/z dimensions.

6 Results

6.1 Performance Evaluation

We first evaluated TurboTrack's performance quantitatively. Specifically, we evaluated its localization accuracy, latency, and performance with moving targets.

(a) 3D Localization Accuracy: We tested TurboTrack's localization accuracy in LOS an NLOS settings. We performed 5,000 experimental trials, each lasting between 30 seconds and one minute. This allowed us to collect more than 20 million location measurements. In each trial, we performed different arbitrary movements with the robotic arm or the drone carrying the RFID. We compute the tracking error as the difference between TurboTrack's location estimate and the ground truth from OptiTrack.

Figs. 7-8 plot the CDFs of the location errors in each of the x, y, and z dimensions for both LOS and NLOS settings. Our results show that TurboTrack achieves a median error less than 1 cm and a 90^{th} percentile error is less than 2 cm in each dimension. Further, we note that the accuracy in LOS is slightly better than its accuracy in NLOS settings. This is expected since the SNR is higher in LOS, resulting in higher accuracy.

(b) Latency & Frame Rate: Next, we evaluated the latency of TurboTrack's pipelined architecture in both 2D and 3D localization. Our 2D trials were performed using an RFID attached to a Roomba. The latency is computed as the time difference between the time the USRP obtains an RFID response and the time it outputs a location. We ignore the time-of-flight of the wireless signal since it is only few nanoseconds for the distances of testing (few meters) hence negligible for our latency measurements.

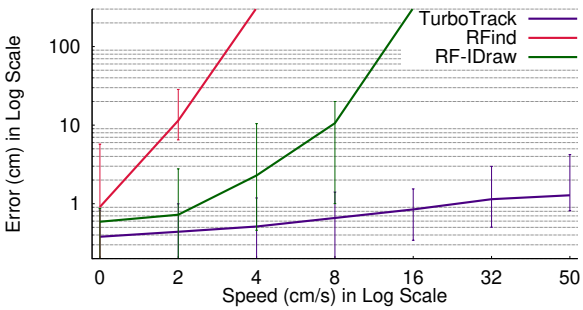


Figure 9: **Tracking Error vs. Speed.** We plot the median 2D tracking accuracy of TurboTrack (violet), RF-IDraw (green), and RFind (red) vs speed. Error bars indicate 10^{th} and 90^{th} percentiles.

Our results shows that the 99^{th} percentile latency measurements for 2D and 3D tracking are 6.6 ms and 7.3 ms, respectively. This latency is primarily limited by the processing time of the computer rather than the latency of the RFID's response or acquisition. Specifically, in both the 2D and 3D experiments, TurboTrack's receivers all obtain the RFID responses at the same time, yet the difference in latency is due to the difference in processing speed. We also note that across these experiments, TurboTrack could achieve a frame rate of 300 frames/second in both 2D and 3D localization. This high frame rate owes to TurboTrack's pipelined architecture which decouples the frame rate from the latency, enabling it to achieve both high frame rates and low latency.

(c) Performance with Motion: TurboTrack's high accuracy, low latency, and high frame rate aim at enabling it to track objects in continuous motion, as necessary for robotic manipulation and tracking tasks. Next, we evaluated TurboTrack's accuracy with motion and compare it to our baselines. In fairness to RF-IDraw, we focused on 2D tracking since the system was only evaluated in 2D.

We perform 400 experimental trials, each time varying the speed at which an RFID moves. We varied the speed by attaching the RFID on a mobile Roomba whose speed can be controlled. Fig. 9 plots the CDF of tracking accuracy for RFind, RF-IDraw, and TurboTrack. It is important to note that the figure is *in log-log scale* to demonstrate how much TurboTrack is more capable in maintaining its accuracy despite high speed motion.

Our results show that TurboTrack outperforms both RF-IDraw and RFind across all speeds. Even at speeds of 50 cm/s, which is the maximum speed of Roomba [4], its error remains under 2 centimeters. This is due to

three reasons: First, it has significantly higher frame rates than RFind. Second, it has more resilience to multipath than RF-IDraw due to its larger bandwidth. And lastly, its Bayesian framework boosts its accuracy and robustness.

We also note that RFind suffers the most with motion, even at speeds as low as 2 cm/s. This owes to its frequency hopping process, which requires the object to remain static for 3 s. On the other hand, while RF-IDraw can deal with some movements, its ability to accurately track the phase diminishes beyond speeds of few cm/sec.

In the above result, we eliminated the initial position error as per RF-IDraw's implementation. In fairness to RFind, we run 100 additional experiments with a static RFID and compute RFind and RF-IDraw's accuracy. We summarize our results in the table below.

	RFind	RF-IDraw	TurboTrack
Median (cm)	0.87	19	0.51
90^{th} percentile (cm)	2.3	63	1.1

Table 1: **Positioning Accuracy.**

The table shows that both TurboTrack and RFind outperform RF-IDraw, which has a median accuracy of 19 cm. This result is expected since RF-IDraw is designed for high tracking accuracy rather than high localization accuracy. In the RF-IDraw paper [71], the authors call this the initial position error. We also note that even though RFind has larger overall bandwidth than TurboTrack (around 200 MHz vs TurboTrack's 100 MHz), TurboTrack takes advantage of motion and fuses measurements across multiple antennas to achieve a 90^{th} percentile error of around 1.1 cm.

6.2 Microbenchmarks

Next, we would like to understand the effectiveness of each of TurboTrack's sub-components. To do so, we ran micro-benchmarks with partial implementations of the system as well as with simplified variants. This enables us to gain deeper understanding into the importance of each component as well as the effectiveness of TurboTrack's design choices from two main perspectives: localization accuracy and computational efficiency.

(a) Decomposing TurboTrack's Gains. We would like to quantify the accuracy gains arising from TurboTrack's space-time super-resolution algorithm and the different sub-components of this algorithm. Hence, we implemented three variants of the algorithm and compared their localization accuracy; we focus on 2D localization for simplicity. All three schemes are given the same distance estimates obtained from TurboTrack's one-shot wideband estimation algorithm over 100 MHz of bandwidth. The schemes differ in how they perform localization based on these distance estimates:

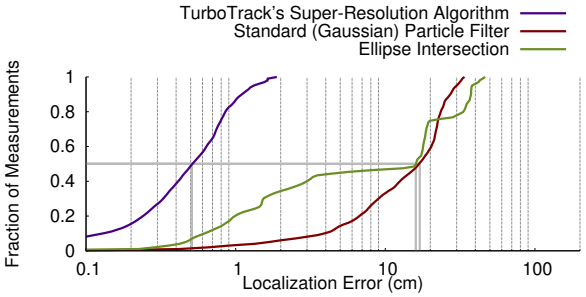


Figure 10: **Comparing Partial Implementations of TurboTrack.** The figure plots the CDF of the 2D localization errors of three different localization schemes, all of which use the output of TurboTrack's one-shot wideband estimation.

1. *Ellipse Intersection:* Recall from §4 that each distance estimate maps to an ellipse whose foci are the transmit and receive antennas. A simple localization method would consider the most likely distance estimate from each receive antenna, map it to an ellipse, and solve for the intersection point of the ellipses in order to identify the tag's location [14, 13]. To implement this scheme, we select the distance estimate with the highest weight from each receive antenna (where the weights are obtained from the amplitude of the interpolated FFT in a manner similar to [13]). Since TurboTrack employs more antennas than the number of ellipses needed for 2D localization, we solve for all the intersection point and assign the one closest to the ground truth as the tag's location. Doing so provides this scheme with more information than we provide TurboTrack's super-resolution algorithm. Hence, TurboTrack's ability to outperform this scheme is a stronger demonstration of its effectiveness.
2. *Standard (Gaussian) Particle Filter:* The second localization scheme employs a standard Gaussian particle filter. The main difference between this scheme and TurboTrack's algorithm is that its likelihood function (i.e., Eq. 1) is a single Gaussian (centered around the most likely distance estimate and with a large standard deviation) rather than a mixture of narrow Gaussians. This scheme also incorporates the SIS filter.
3. *TurboTrack's Super-Resolution Algorithm:* The final scheme implements TurboTrack's space-time super-resolution algorithm, which incorporates the proposed Mixture of Gaussians (MoG) likelihood function and SIS filter.

Fig. 10 plots the CDFs of the localization error for each of the three schemes. We observe the following:

- All three schemes achieve a median error less than 20 cm. This decimeter-scale error of *moving* RFIDs is smaller than that of state-of-the-art systems (as per in §6.1(c)), and is possible because of TurboTrack's one-shot wideband estimation technique.

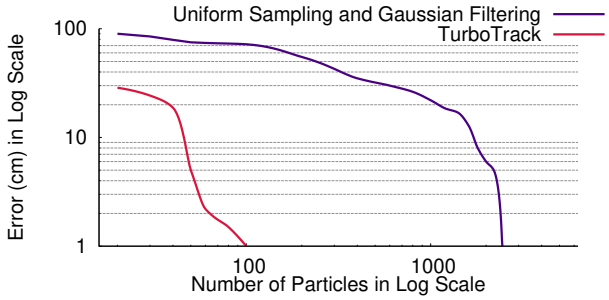


Figure 11: **Tracking Error vs. Number of Particles.** The figure plots the median error of 2D localization as a function of the number of particles for TurboTrack's sampling method and compares it to a uniform sampling method. The figure shows that TurboTrack can converge to sub-centimeter accuracy using $25\times$ less particles.

- The error distribution of the ellipse intersection method exhibits multiple discontinuities. This is because there are multiple candidate distance estimates (as described in §4.2), and the scheme picks one of them independently in every frame. Thus, while the error is small if the correct candidate is chosen, the error is typically larger than a wavelength (33 cm) if the incorrect candidate has a higher weight (e.g., due to noise). We note that this scheme's results are in line with the reported errors in [45] when using the same bandwidth with frequency hopping for static RFIDs.
- Interestingly, the ellipse intersection method has slightly smaller median error (16 cm) than that of a standard particle filter-based method (17 cm). However, the 90^{th} percentile error of the particle filter is smaller than that of the intersection-based method. This is due to two reasons. First, the ellipse intersection scheme is given more information since we allow it to choose the intersection point closest to the ground truth. And second, the particle filter smoothes the trajectory, which increases its median error but enables it to achieve better tail performance.
- Finally, TurboTrack's full implementation significantly outperforms both partial schemes, achieving at least $30\times$ improvement in median accuracy (5 mm median error).

(b) Complexity Gain of Approximate Inference. Next, we would like to quantify the efficiency (complexity) gains arising from TurboTrack's approximate inference algorithm and its initial sampling scheme. To do so, we compare the algorithm to a baseline implementation of a standard Gaussian particle filter. The baseline uniformly samples 2D space and updates particles weights using the Gaussian filtering method described in §6.2(a).

The complexity of a particle filter is a direct function of the number of particles N used to represent the distribution. Hence, to compare the two schemes, we ran both particle filters (the baseline and TurboTrack's) with the same number of particles over 30-second-long trajectories, and we computed the localization accuracy. We repeat this process multiple times, each time with a different number of particles.

Fig. 11 plots the median error (on the y-axis) as a function of the number of particles (on the x-axis) in log-log scale for each of the two schemes. The figure shows that TurboTrack's inference algorithm can converge to sub-centimeter accuracy with only 100 particles; in comparison, to achieve the same accuracy, the uniform sampling-based scheme requires around 2500 particles. Since the complexity of the particle filters is $\mathcal{O}(N)$, this demonstrates that TurboTrack's algorithm is more computationally efficient.

6.3 Qualitative Performance

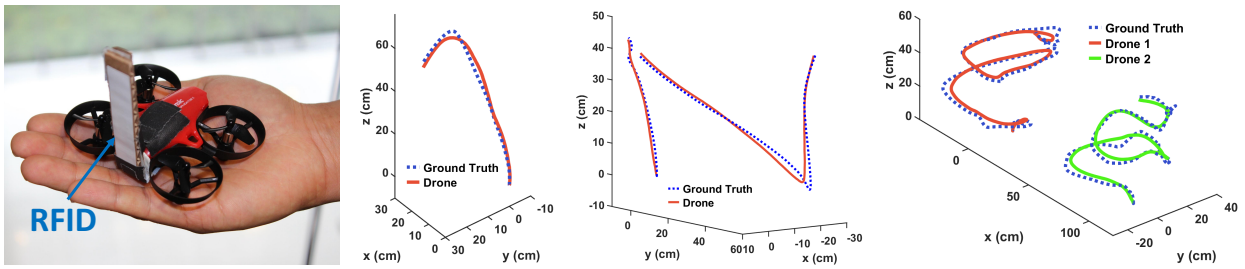
Finally, we evaluated TurboTrack qualitatively in fine-grained robotic tasks. Our results, shown in Fig. 12-13, demonstrate the ability to track nanodrones docking, maneuvering, and even flying simultaneously. The results also show that TurboTrack's fine-grained tracking can be an enabler for collaborative packaging and handover between robotic arms.

7 Related Work & Conclusion

(a) RF-based Localization is a long studied problem in the networking community. Early work relied on measuring the received signal strength (RSS) [54, 76, 24, 26], the angle of arrival (AoA) [49, 77, 20, 40], and the received signal phase [19, 41]. These proposals could operate correctly in line-of-sight but not in the presence of multi-path since constructive and destructive interference make the strength, angle, and the phase of the received signal unpredictable.

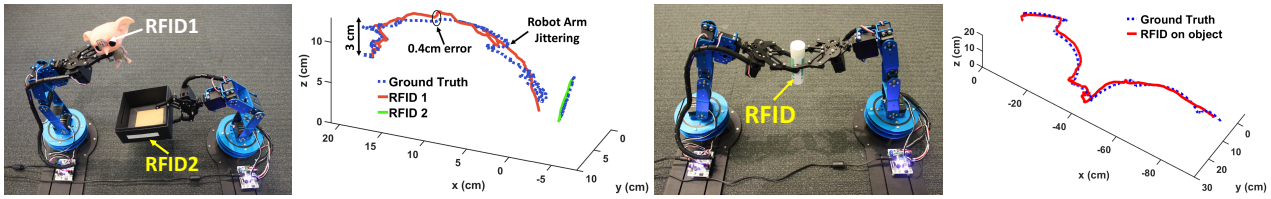
Unfortunately, state-of-the-art proposals that can deal with multi-path cannot deliver on the mobility or accuracy requirements for fine-grained robotic tasks. In particular, solutions that achieve high accuracy require the target to remain static for seconds as their antennas move over multiple meters, collecting measurements from different spatial locations then combining them to localize [70, 60, 69, 51]. Others, like RFind [45], achieve high accuracy without requiring antenna motion, but they still require the object of interest to remain static for few seconds as they perform frequency hopping over a large bandwidth. As demonstrated in §6, this leads to large errors in mobile settings. Fundamentally, even if one could hop frequencies faster, such systems would still suffer from a range-Doppler ambiguity [46] and have lower accuracy, frame rate, and throughput than TurboTrack.

To avoid this latency problem, researchers have looked into recovering the shape of the trajectory while sacrificing exact positioning [71, 75, 61, 39]. These proposals focus on tracking *changes in distances* and can achieve



(a) Nanodrone with RFID (b) Docking (c) Maneuvering (d) Multi-drone Simultaneous Tracking

Figure 12: **Tracking Nanodrones.** (a) shows the nanodrone we use in our experiments. (b), (c), and (d) show TurboTrack's output and the ground truth (dotted blue) in tracking docking, maneuvering, and two drones flying simultaneously.



(a) Packaging (b) Output (c) Handover (d) Output

Figure 13: **Robotic Manipulation.** (a) shows two robotic arms collaborating on packaging an RFID-tagged item in an RFID-tagged box. (c) shows one robotic arm handing over an RFID-tagged item item to another. (b) and (d) show TurboTrack's output for tracking the RFID tagged items.

high accuracy over short periods of time, however, they have decimeter-scale accuracy (i.e., tens of centimeters) in computing the exact location. As a result, while they may recover the shape of a trajectory of an RFID or a WiFi device, they cannot enable precise positioning tasks like grasping or manipulation. In contrast to all of these proposals, because TurboTrack computes the *wideband estimates from every single RFID response*, it can achieve precise positioning and at low latency.

(b) Super-resolution algorithms have been extensively studied both theoretically and practically. Most past work falls in the imaging community, where the goal is to obtain a higher-resolution image by combining images of the same scene across multiple viewpoints [65, 23, 37, 64] or across time frames in a video [31, 22, 52, 53]. While TurboTrack is inspired by this body of work, it differs in two key aspects. First, in contrast to imaging systems which rely only on pixel intensity, TurboTrack also has access to phase information and utilizes it in its super-resolution algorithm. Second, its formulation is unique in how it models and linearizes distances obtained from wideband RF measurements and how it incorporates them into a computationally efficient particle filter.

The RF community has also taken interest in super-resolution algorithms, with famous algorithms like MUSIC [32], smoothed MUSIC [59], and ESPRIT [66]. TurboTrack builds on this body of work as well, and to the best of our knowledge introduces the first Bayesian spatio-temporal framework that combines bandwidth and phase measurements to achieve this level of accuracy.

Naturally, TurboTrack also relates to a growing literature on object manipulation in the robotics community. The majority of past work relies on vision-based or optical systems which, unlike TurboTrack, cannot operate in visually occluded settings [48, 74, 42]. Finally, we note that some of the RF localization solutions mentioned above [69, 60] have been explored in this context as well, but they lacked the localization accuracy and/or the low latency required to deliver on these tasks. TurboTrack is inspired by this work and builds on it to enable highly accurate tracking and identification for fine-grained robotic tasks, particularly in cluttered or occluded settings.

Acknowledgments. We thank Nick Selby for his help in early developments of the system and in the baseline evaluation. We also thank our shepherd, Lili Qiu, and the anonymous NSDI reviewers for their feedback and insights. This research is partially supported by the MIT Media Lab and NSF CPS Award CNS-1739723.

References

[1] Avery denison. `http://rfid.averydennison.com`, 2018. Avery Denison.

[2] EPC UHF Gen2 Air Interface Protocol. `http://www.gs1.org/epcrfid/epc-rfid-uhf-air-interface-protocol/2-0-1`, 2018.

[3] Intel X520. `https://ark.intel.com`, 2018. Intel X520.

[4] irobot roomba. https://www.irobot.com, 2018. irobot Roomba.

[5] Lewansoul learm. http://www.lewansoul.com, 2018. LewanSoul LeArm.

[6] M6e. http://www.thingmagic.com, 2018. ThingMagic Inc.

[7] mini-drone a20. http://www.ipotensic.com/, 2018. Potensic mini-drone.

[8] MTI RFID antenna. http://www.mtiwe.com, 2018. MTI Wireless Edge.

[9] Optitrack. http://www.optitrack.com, 2018.

[10] Smartrac RFIDs. https://www.smartrac-group.com, 2018. Smartrac.

[11] UBX daughterboard. http://www.ettus.com, 2018. ettus inc.

[12] usrp n210. http://www.ettus.com, 2018. ettus inc.

[13] F. Adib, Z. Kabelac, and D. Katabi. Multi-person localization via rf body reflections. In *Usenix NSDI*, 2015.

[14] F. Adib, Z. Kabelac, D. Katabi, and R. C. Miller. 3d tracking via body radio reflections. In *Usenix NSDI*, 2014.

[15] H. Akashi and H. Kumamoto. Construction of discrete-time nonlinear filter by monte carlo methods with variance-reducing techniques. *Systems and Control*, 19(4):211–221, 1975.

[16] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Multi-robot cooperation in the martha project. *IEEE Robotics Automation Magazine*, 5(1):36–47, March 1998.

[17] Alien Technology Inc. ALN-9640 Squiggle Inlay, 2018. www.alientechnology.com.

[18] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation*, 9(2):152–165, April 1993.

[19] D. Arnitz, K. Witrisal, and U. Muehlmann. Multifrequency continuous-wave radar approach to ranging in passive uhf rfid. *IEEE Transactions on Microwave Theory and Techniques*, 57(5):1398–1405, 2009.

[20] S. Azzouzi, M. Cremer, U. Dettmar, R. Kronberger, and T. Knie. New measurement results for the localization of uhf rfid transponders using an angle of arrival (aoa) approach. In *IEEE RFID*. IEEE, 2011.

[21] J. M. Beer, C.-A. Smarr, T. L. Chen, A. Prakash, T. L. Mitzner, C. C. Kemp, and W. A. Rogers. The domesticated robot: design guidelines for assisting older adults to age in place. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 335–342. ACM, 2012.

[22] C. M. Bishop, A. Blake, and B. Marthi. Super-resolution enhancement of video. In *AISTATS*, 2003.

[23] N. K. Bose and N. A. Ahuja. Superresolution and noise filtering using moving least squares. *IEEE Transactions on Image Processing*, 15(8):2239–2248, 2006.

[24] M. Bouet and A. L. Dos Santos. Rfid tags: Positioning principles and localization techniques. In *Wireless Days, 2008. WD'08. 1st IFIP*, pages 1–5. IEEE, 2008.

[25] L. Chaimowicz, T. Sugar, V. Kumar, and M. F. M. Campos. An architecture for tightly coupled multi-robot cooperation. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 3, pages 2992–2997 vol.3, May 2001.

[26] K. Chawla, C. McFarland, G. Robins, and C. Shope. Real-time rfid localization using rss. In *Localization and GNSS (ICL-GNSS), 2013 International Conference on*, pages 1–6. IEEE, 2013.

[27] B. Choi and J. Lee. Mobile robot localization in indoor environment using rfid and sonar fusion system. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2039–2044, Oct 2009.

[28] C. M. Costa, H. M. Sobreira, A. J. Sousa, and G. M. Veiga. Robust and accurate localization system for mobile manipulators in cluttered environments. In *2015 IEEE International Conference on Industrial Technology (ICIT)*, pages 3308–3313, March 2015.

[29] Defense IQ. Small is beautiful: Nano drone tech is advancing. https://www.defenceiq.com/defence-technology/articles/nano-drone-tech-is-advancing.

[30] A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

[31] M. Elad and A. Feuer. Superresolution restoration of an image sequence: adaptive filtering approach. *IEEE Transactions on Image Processing*, 8(3):387–395, 1999.

[32] B. Friedlander. The root-music algorithm for direction finding with interpolated arrays. *Signal processing*, 30(1):15–29, 1993.

[33] M. Hassanalian and A. Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, 91:99 – 131, 2017.

[34] K. Joshi, S. Hong, and S. Katti. Pinpoint: Localizing interfering radios. In *Usenix NSDI*, 2013.

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

[36] N. Kargas, F. Mavromatis, and A. Bletsas. Fully-coherent reader with commodity sdr for gen2 fm0 and computational rfid. *IEEE Wireless Communications Letters*, 4(6):617–620, 2015.

[37] K. I. Kim and Y. Kwon. Single-image super-resolution using sparse regression and natural image prior. *IEEE transactions on pattern analysis & machine intelligence*, (6):1127–1133, 2010.

[38] A. Kong, J. S. Liu, and W. H. Wong. Sequential imputations and bayesian missing data problems. *Journal of the American statistical association*, 89(425):278–288, 1994.

[39] M. Kotaru and S. Katti. Position tracking for virtual reality using commodity wifi. *IEEE CVPR*, 2017.

[40] R. Kronberger, T. Knie, R. Leonardi, U. Dettmar, M. Cremer, and S. Azzouzi. Uhf rfid localization system based on a phased array antenna. In *Antennas and Propagation (APSURSI), 2011 IEEE International Symposium on*, pages 525–528. IEEE, 2011.

[41] X. Li, Y. Zhang, and M. G. Amin. Multifrequency-based range estimation of rfid tags. In *RFID, 2009 IEEE International Conference on*, pages 147–154. IEEE, 2009.

[42] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, Y. Taguchi, T. K. Marks, and R. Chellappa. Fast object localization and pose estimation in heavy clutter for robotic bin picking. *The International Journal of Robotics Research*, 31(8):951–973, 2012.

[43] G. Loianno, C. Brunner, G. McGrath, and V. Kumar. Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu. *IEEE Robotics and Automation Letters*, 2(2):404–411, 2017.

[44] C. Luis and J. L. Ny. Design of a trajectory tracking controller for a nanoquadcopter. *CoRR*, abs/1608.05786, 2016.

[45] Y. Ma, N. Selby, and F. Adib. Minding the billions: Ultrawideband localization for deployed rfid tags. *ACM MobiCom*, 2017.

[46] B. R. Mahafza. *Radar systems analysis and design using MATLAB*. Chapman & Hall, 2013.

[47] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2308–2315. IEEE, 2010.

[48] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *2010 IEEE International Conference on Robotics and Automation*, pages 2308–2315, May 2010.

[49] G. Mao, B. Fidan, and B. D. Anderson. Wireless sensor network localization techniques. *Computer networks*, 51(10):2529–2553, 2007.

[50] W. Mao, J. He, and L. Qiu. Cat: High-precision acoustic motion tracking. In *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking*, MobiCom '16, pages 69–81, New York, NY, USA, 2016. ACM.

[51] R. Miesen, F. Kirsch, and M. Vossiek. Holographic localization of passive uhf rfid transponders. In *RFID (RFID), 2011 IEEE International Conference on*, pages 32–37. IEEE, 2011.

[52] B. Narayanan, R. C. Hardie, K. E. Barner, and M. Shao. A computationally efficient super-resolution algorithm for video processing using partition filters. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(5):621–634, 2007.

[53] M. K. Ng, H. Shen, E. Y. Lam, and L. Zhang. A total variation regularization based super-resolution reconstruction algorithm for digital video. *EURASIP Journal on Advances in Signal Processing*, 2007(1):074585, 2007.

[54] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil. Landmarc: indoor location sensing using active rfid. *Wireless networks*, 10(6):701–710, 2004.

[55] Z. Nitzan, D. Lavee, and G. Guri. Battery-assisted backscatter rfid transponder, July 1 2008. US Patent 7,394,382.

[56] P. Pannuto, B. Kempke, and P. Dutta. Slocalization: sub-μw ultra wideband backscatter localization. In *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 242–253. IEEE Press, 2018.

[57] J. R. Cannon and E. Miles. Utilizing human vision and computer vision to direct a robot in a semi-structured environment via task-level commands. In *Intelligent Robots and Systems, IEEE/RSJ International Conference on(IROS)*, volume 01, page 366, 08 1995.

[58] Z. Sahinoglu, S. Gezici, and I. Gvenc. *Ultra-wideband Positioning Systems: Theoretical Limits, Ranging Algorithms, and Protocols*. Cambridge University Press, New York, NY, USA, 2011.

[59] T.-J. Shan, M. Wax, and T. Kailath. On spatial smoothing for direction-of-arrival estimation of coherent signals. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1985.

[60] L. Shangguan and K. Jamieson. The design and implementation of a mobile rfid tag sorting robot. In *ACM MobiSys*, pages 31–42, 2016.

[61] L. Shangguan and K. Jamieson. Leveraging electromagnetic polarization in a two-antenna whiteboard in the air. In *ACM CoNEXT*, 2016.

[62] R. Srinivasan. *Importance Sampling: Applications in Communications and Detection*. Springer Berlin Heidelberg, 2013.

[63] S. Thrun. Particle filters in robotics. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 511–518. Morgan Kaufmann Publishers Inc., 2002.

[64] B. C. Tom and A. K. Katsaggelos. Reconstruction of a high-resolution image by simultaneous registration, restoration, and interpolation of low-resolution images. In *icip*, page 2539. IEEE, 1995.

[65] R. Tsai. Multiframe image restoration and registration. *Advance Computer Visual and Image Processing*, 1:317–339, 1984.

[66] U. Tureli, H. Liu, and M. D. Zoltowski. Ofdm blind carrier offset estimation: Esprit. *IEEE Transactions on communications*, 48(9):1459–1461, 2000.

[67] US Department of Defense. Department of Defense Announces Successful Micro-Drone Demonstration. https://dod.defense.gov/News/News-Releases/News-Release-View/Article/1044811/department-of-defense-announces-successful-micro-drone-demonstration/.

[68] D. Vasisht, S. Kumar, and D. Katabi. Decimeter-level localization with a single wifi access point. In *Usenix NSDI*, 2016.

[69] J. Wang, F. Adib, R. Knepper, D. Katabi, and D. Rus. RF-Compass: Robot Object Manipulation Using RFIDs. In *ACM MobiCom*, 2013.

[70] J. Wang and D. Katabi. Dude, where's my card? rfid positioning that works with multipath and non-line of sight. In *ACM SIGCOMM*, 2013.

[71] J. Wang, D. Vasisht, and D. Katabi. Rf-idraw: virtual touch screen in the air using rf signals. In *ACM SIGCOMM*, 2015.

[72] T. Wei and X. Zhang. Gyro in the air: tracking 3d orientation of batteryless internet-of-things. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 55–68. ACM, 2016.

[73] J. Xiong and K. Jamieson. ArrayTrack: a fine-grained indoor location system. In *Usenix NSDI*, 2013.

[74] H. Yang and A. Kak. Determination of the identity, position and orientation of the topmost object in a pile: Some further experiments. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 293–298, April 1986.

[75] L. Yang, Y. Chen, X.-Y. Li, C. Xiao, M. Li, and Y. Liu. Tagoram: Real-time tracking of mobile rfid tags to high precision using cots devices. In *ACM MobiCom*, 2014.

[76] J. Zhou and J. Shi. Rfid localization algorithms and applications—a review. *Journal of intelligent manufacturing*, 20(6):695–707, 2009.

[77] J. Zhou, H. Zhang, and L. Mo. Two-dimension localization of passive rfid tags using aoa estimation. In *Instrumentation and Measurement Technology Conference (I2MTC), 2011 IEEE*, pages 1–5. IEEE, 2011.