Mobility Improves Accuracy: Precise Robot Manipulation with COTS RFID Systems

Paper # 81, 14 pages

Anonymous Author(s)

ABSTRACT

This paper presents GLAC, the first 3D localization system that enables millimeter-level object manipulation for robotics using only COTS RFID devices. The key insight of GLAC is that mobility reduces ambiguity and thus improves accuracy. Unlike state-of-the-art systems that require extra time or hardware to boost performance, it draws the power of modeling mobility in a delicate way. In particular, we build a novel framework for real-time tracking using the Hidden Markov Model (HMM). In our framework, multiple Kalman filters are designed to take a single phase observation for updating mobility statues, and a fast inference algorithm is proposed to efficiently process an exponentially large number of candidate trajectories. We prototype GLAC with only UHF tags and a commercial reader of four antennas. Comprehensive experiments show that the median position accuracies of x/y/z dimensions are within 1 cm for both LoS and NLoS cases. The median position accuracy for slow-moving targets is 0.41 cm, which is 2.2×, 17.3×, and 14.9× better than Turbo-Track, Tagoram, and RF-IDraw, respectively. Also, its median velocity accuracy is at least 20× better than all three competitors for fast-moving targets. Besides accuracy, it achieves more than 4× localization time gains over state-of-the-art systems.

ACM Reference Format:
Anonymous Author(s). 2020. Mobility Improves Accuracy: Precise Robot Manipulation with COTS RFID Systems: Paper # 81, 14 pages. In *Proceedings of CoNEXT '20.* ACM, New York, NY, USA, 14 pages. https://doi.org/TBA

1 INTRODUCTION

Robotics has been evolving significantly over the past decades, becoming more capable, agile, and precise [10, 31, 33, 54]. This robot precision is the key enabler to future applications. Different from earlier industrial automation where many repetitive tasks, e.g., car-door painting, do not require high precision, modern robots nowadays are integrating motion control, actuator, and other advanced technologies to complete a wide range of tasks [1, 10, 25, 37], from welding to surgery. For example, recent studies show that delicate eye surgery can be done with a robot that is ten times more precise than a human surgeon [30].

Laser and vision-based solutions have been the mainstream to deliver robot precision [18, 24, 35]. The most widely used clinical robotic surgical system, da Vinci, has mechanical arms with surgical instruments and a camera arm that provides a high-definition, magnified, 3D view of the surgical site [13]. Such systems, however, have two main drawbacks: high cost that prohibits small and medium enterprises adoption, e.g., an infrared VICON motion capture system is priced at hundreds of thousands of dollars [45]; and poor handling of occlusion that is the fundamental weakness for visions. As a result, robotics has been looking for real-time localization solutions that are cost-effective, highly accurate, and capable of dealing with occlusion.

To address aforementioned issues, the wireless community has made much effort in RFID localization systems, which offer an appealing alternative as it can achieve high-precision, auto-identification, and working with Non-Line-of-Sight (NLoS) cases [28, 29, 45–47, 53]. Table 1 lists the key features of most state-of-the-art systems. Through a careful study, we find that those prior arts focus on putting in either more money, .e.g., expensive and powerful equipment, or more time by requiring heavy computation or restrictive mobility. For example, RFind [29] is quite resource-intensive; it requires 4 USRPs of N210 (amounting to $ 8000), more than 66 packets to derive a position estimate (translating to a reading delay of seconds), and a bandwidth of 200 MHz wide (10 times of a typical WiFi channel). Tagoram [53], which needs only a single COTS reader, takes several seconds computation time to localize once, making it unfit for real-time tracking. RF-IDraw [47], which has moderate money and time cost, achieves an unsatisfactory median position accuracy, 19 cm; thus it is only suitable for applications that only require shape fidelity, like drawing, as opposed to precise location control, such as robot-assisted surgery. In short, how to build a real-time localization system for precise robot control with COTS RFID equipment remains a big challenge.

In this paper, we present GLAC, the first 3D localization system that enables millimeter-level object manipulation for robotics using only COTS devices. Compared to prior arts, it achieves the most accurate localization while consuming

CoNEXT '20, December 1-4, 2020, Barcelona, Spain
2020. ACM ISBN TBA…$TBA
https://doi.org/TBA

Table 1: Comparison of state-of-the-art RFID localization systems.

	How much we pay				How well we do	
	Cost ($)	Computation	Reading [1]	Bandwidth	Position Accuracy [2]	Dim.
GLAC	~ 1,000 (1 COTS reader)	~ μs	1 pkt	0.25 MHz	sub-centimeter	3D
TurboTrack [28]	~ 20,000 (3×X310 & 1×N210) [3]	~ μs	4 pkts	100 MHz	sub-centimeter	3D
RFind [29]	~ 8000 (4× N210)	~ μs	>66 pkts	200 MHz	1.1 cm	3D
RF-IDraw [47]	~ 2000 (2 COTS reader)	~ μs	8 pkts	0.25 MHz	19 cm	2D
RF-Compass [45]	~ 2000 (1× N210)	~ μs	60 pkts	0.25 MHz	2.8 cm	3D
PinIt [46]	~ 2000 (1× N210)	~ μs	480 pkts	0.25 MHz	11.2 cm	2D
MobiTagBot [41]	~ 1000 (1 COTS reader)	~ μs	10 pkts	2.5 MHz [4]	2.8 cm	1D
Tagoram [53]	~ 1000 (1 COTS reader)	~ s	4 pkts	0.25 MHz	12.3 cm	2D

[1] The minimal no. of packets needed to derive a location estimate.
[2] The median accuracy when a tag moves at a velocity of less than 10 cm/s.
[3] The extra cost associated with SDRs involves formal FCC certifications, which is necessary to turn prototypes into commercial RFID devices.
[4] The original paper recommends hopping more than 10 channels for better performance.

minimal resources in terms of cost-effectiveness and time-efficiency. Naturally, a question arises; as the saying goes "you get what you pay for", how could we achieve more by expending less? Of course, GLAC is no free lunch. Unlike previous systems that mainly employ more physical resources or assumptions, we take a completely distinct approach, which is to draw power from mobility itself. Our key insight is that ambiguity may hurt once, but not forever. In other words, while differentiating ambiguity at a time may be difficult due to insufficient information, it becomes more tractable when we build a network of candidate trajectories containing all the possible ambiguities. Because there is only one candidate trajectory that most closely matches the ground truth if the object's mobility is reasonably modeled. It is like that we put a great number of gladiators into a fight arena, and the final survivor is the strongest one. That's why we call it GLAC (GLadiator trACking). Turning this idea into a system, however, faces two critical challenges.

1) How to reduce ambiguities with constrained COTS devices?

The basic principle of resolving ambiguities is to acquire more information. Nevertheless, the intrinsic low reading rate of COTS RFID devices cannot meet the demand for high-precision. As a result, many prior systems attempt to overcome this barrier from different perspectives. TurboTrack employs extra localization helpers (wideband USRPs) that can achieve reading rates of upto 300 frames/s [28]. RF-Compass [45], PinIt [46], and MobiTagBot [41] require to collect a number of packets on the scale of several seconds. RF-IDraw draws information from a deliberately designed antenna placement with 8 antennas using plane geometry. On the contrary, we resort to mobility modeling since our goal is to push the envelope of COTS RFID localization with no strings attached. Specifically, we build a novel framework of localization using the HMM where spatial states at a time are discrete and those states over time are continuous. This way, RFID localization is translated into an HMM inference problem. As more observations come in, the likelihood of identifying the optimal trajectory (hidden states) increases, i.e., mobility over time reduces ambiguities. Further, to solve the low-rate COTS reading problem, we design Kalman filters that take a single phase for updating as opposed to multi-phase feeding in all prior arts.

2) How to achieve real-time localization efficiency?

The localization time involves reading time and computation time. While our HMM framework is designed to accommodate single-phase readings, which achieves minimal communication overhead, its computation overhead is prohibitive. Because the number of candidate trajectories increases exponentially over time. To address this, we employ a fast inference scheme that includes nearest neighbor pruning and initial states pruning techniques. Such an expedited scheme ensures that the number of candidate trajectories does not grow over time. Given fully optimized reading and computation time, GLAC can deliver accurate localization with high time-efficiency.

We prototype GLAC with only UHF tags and a commercial reader with four antennas. The main results of our extensive experiments are summarized as follows.

- The median position accuracies of x/y/z dimensions are within 1 cm for both LoS and NLoS cases.
- In slow-moving scenarios, the median position accuracy is 0.41 cm, which is 2.2×, 17.3×, and 14.9× better than TurboTrack, Tagoram, and RF-IDraw, respectively, and those gaps enlarge to 21.8×, 21.7×, and 32.6× for fast-moving scenarios.

- The median velocity accuracy is 2.26 cm/s for slow-moving cases, which is 5.3×, 7.6×, and 7.2× better than TruboTrack, Tagoram, and RF-IDraw, respectively, and those differences grow to more than 20× for all three competitors when the target is fast-moving.
- As to localization time-efficiency, it achieves overall time gains of 4× over TurboTrack, 72× over Tagoram, and 4× over RF-IDraw.

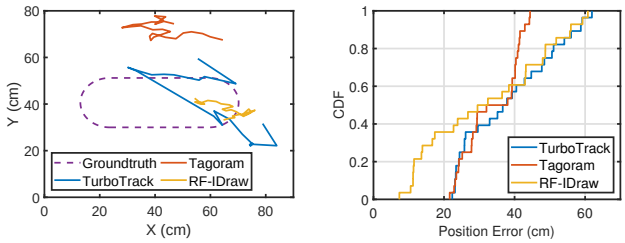
Contributions: We make the following contributions:

- GLAC is the first mobile 3D localization system that supports robot object manipulation using only COTS devices. It enables robots to locate an object with an accuracy of millimeter-level and a timescale of tens of milliseconds. By doing so, it provides a practical tracking solution for robotics, which is cost-effective, highly accurate, and real-time even with NLoS.
- GLAC presents a novel RF-tracking framework formulated using the HMM. It makes three key differences: 1) it leverages the object's mobility for disambiguation; 2) it can estimate the object's velocity in addition to its location using only a single phase measurement; 3) it employs a fast inference algorithm that uses nearest neighbor pruning and solid geometry.
- A real-time prototype of GLAC is implemented, and extensive real-world experiments show its capability of millimeter-level tracking in real-time. As GLAC requires only COTS gadgets, billions of deployed commercial tags and readers are ready to benefit a wide range of agile robot applications.
- To further contribute to the community and support reproducibility, we intend to open source the code and experimental data of our implementation on GitHub after the decision of this submission is notified. The link is not provided here to ensure compliance with anonymity rules.

2 MOTIVATION

To begin with, we investigate how prior systems perform under a standard test, which can help us gain deep insights about what the essentials are for high-precision tracking. It goes as follows. We employ a COTS reader with 4 antennas to continuously localize a tag moving at 40 cm/s along a predefined track. The implemented systems include RF-IDraw [47], Tagoram [53], and TurboTrack [28]. Best efforts have been made to ensure faithfully reproductions, since comparable results are achieved according to the original papers [1]. Here, we make the standard test as simple as possible to identify root causes affecting tracking accuracy by ruling out most of the explicit and implicit assumptions from prior works, such as powerful SDR readers, restrictive mobility patterns for readers or tags.

[1]For details including settings, parameters and detailed results, please refer to Section 5.



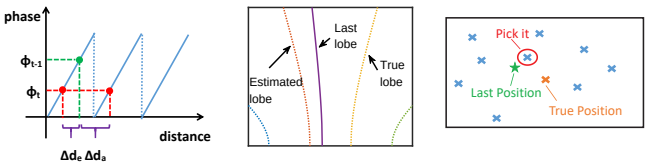
(a) Trajectories at 40 cm/s (b) CDF comparison

Figure 1: Comparison using 1 COTS reader of 4 antennas. (a) When the tag's velocity is 40 cm/s, the trajectories of all three state-of-the-art systems are far from the ground truth. (b) shows they are all unable to deliver high-precision tracking with COTS devices.

Results are shown in Figure 1a. We observe that all three state-of-the-art systems perform poorly as the resulted trajectories are far from the ground truth. Specifically, as shown in Figure 1b, the median accuracies of TurboTrack, RF-IDraw, and Tagoram are 36.6 cm, 29.5 cm, and 32 cm, respectively. After diving deeper into tracking processes, we find two main factors are contributing to this.

Oversimplified mobility models. Even though all three systems aim for tracking, they do not draw sufficient attention to mobility. For example, Tagoram fits the trajectory using the estimated distance difference that is always less than $\frac{\lambda}{4}$. This strategy may lead to significantly errors when the tag moves fast. As shown in 2a, the estimated distance difference is Δd_e as it is less than $\frac{\lambda}{4}$, which is wrong since the actual distance difference is Δd_a. Similar things happen to RF-IDraw and TurboTrack as well, as shown in Figure 2b and 2c. As RF-IDraw tracks the grating lobe that is the closest to the last lobe and TurboTrack looks for the intersection point that is the closest to the particle position of the previous round, both could result in wrong picks due to the tag's fast movement. In short, mobility has to be properly modeled to deal with fast-moving objects.

Low-rate asynchronous reading. One may wonder that as stated in the original paper [28], TurboTrack can track a fast-moving tag upto 50 cm/s accurately. But why it looks so bad in our test. The answer is COTS readers interrogate tags at a median rate of 30 frames/s [53] whereas SDR readers of [28] work at 300 frames/s. Such a low-rate reading further necessitates mobility modeling. In addition to that, differing from SDR readers that can receive backscattered signals simultaneously across multiple antennas, COTS readers only support asynchronous reading for multiple antennas, i.e.,



(a) Tagoram (b) RF-IDraw (c) TurboTrack

Figure 2: Oversimplified mobility assumptions cause tracking failures for all three systems.

one antenna at a time. Suppose that the delay of two successive asynchronous readings is 30 ms [2], and the antenna reading order is 1-2-3-4. Then the delay between the first antenna and the fourth would translate to a position error of 3.6 cm when the tag moves at 40 cm/s, which may lead to even greater accumulated errors. Simply put, all three state-of-the-art systems are not well prepared for low-rate asynchronous readings.

3 GLAC DESIGN

We will first formally formulate the problem and present the solution framework, then propose a fast inference that achieves real-time efficiency.

3.1 HMM Tracking Framework

Almost all COTS readers can output backscatter phase readings, which is a measure of the range between the reader and tag expressed in units of cycles of the carrier frequency. While this phase measurement can be made with very high-precision, the whole number of cycles is not measurable. Given the distance between the reader and tag, z, the measured phase, ϕ, follows,

$$\phi = \frac{2z}{\lambda} + \delta \mod 2\pi, \tag{1}$$

where λ is the wave length, δ is the noise. Since phase is periodic with a period of 2π radians, the phase values will repeat at distances separated by integer multiples of one-half wavelength [3], $z^n = \phi + \frac{n\lambda}{2}, n = 0, 1, 2...$[4]. We can imagine phase as a measuring tape extending from the reader to the tag that has numbered markers every one millimeter. Unfortunately, however, the numbering scheme returns to zero with every one-half wavelength. To remove such ambiguities, many previous systems make various attempts including moving antennas [41, 46], virtual antennas [53], and dedicated localization helpers [28, 29]. Instead, we choose to model mobility into an HMM based on a key observation; phases are accurate and all we need to do is to reduce ambiguities.

[2]The median value measured from ImpinJ R420 and Thingmagic M6e.

[3]For many readers that introduce additional π ambiguity for phase values, separating distances become $\frac{n\lambda}{4}$.

[4]n can be set based on the maximal reading range.

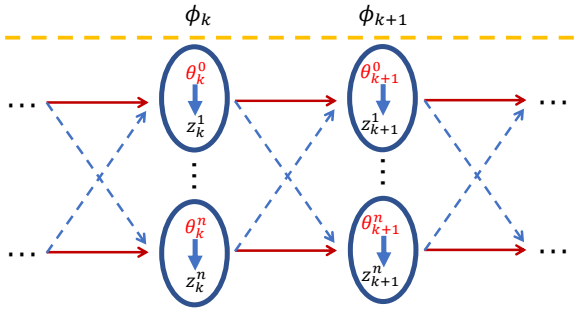


Figure 3: HMM Framework where multiple Kalman filters are used to estimate hidden states from phase observations.

The basic RFID localization problem is that given a time series of phase measurements (ϕ_0,ϕ_2,...,ϕ_κ), find the most probable location series (P_0, P_2,...,P_κ). We translate this process into an HMM, as shown in Figure 3. First, we model the mobility status of each location as $\theta_k = (P_{x,k}, P_{y,k}, V_{x,k}, V_{y,k})^T$, where $P_{x,k}$ and $P_{y,k}$ are x-y coordinates of the location at time k, and $V_{x,k}$ and $V_{y,k}$ are corresponding velocities. Second, we extend phase observation ϕ_k to separating distance observations (ambiguities) $Z_k = (z_k^0, ..., z_k^n)$. Accordingly, state θ_k is discretized into $\theta_k^0, ..., \theta_k^n$, each of which is related to the corresponding distance observation. Hence, the original problem becomes given a set of distance observations (Z_0, Z_1, ..., Z_κ), find the most probable states ($\hat{\theta_0}$, $\hat{\theta_1}$, ..., $\hat{\theta_\kappa}$). Simply put,

$$\hat{\theta_{0:\kappa}} = \arg\max_{\theta_{0:\kappa}} p(\theta_{0:\kappa}|Z_{0:\kappa}). \tag{2}$$

Apparently, there are n^κ possible trajectories with n^κ observation sequences. To find the optimal trajectory, the simplest way is to compute the likelihood of the observation sequence for each trajectory. Then we choose the trajectory with the maximum observation likelihood among all possibilities.

To do so, we need to estimate all the hidden states and this is where Kalman filters come in [23]. From a high level, we put a Kalman filter for each trajectory and estimate its mobility statuses continuously. Before state estimation, we find there are two important parts missing, state transition probabilities and observation likelihoods.

Transition equation. As a general time-varying system, we approximate state transition as a Gaussian process,

$$\theta_{\mathbf{k+1}} = \mathbf{A}\theta_\mathbf{k} + \mathbf{s_k} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \theta_\mathbf{k} + \mathbf{s_k} \tag{3}$$

where Δt is the time difference, $\mathbf{s_k}$ is the system-state Gaussian noise, $\mathbf{A}$ is the system matrix depicting mobility. Both $\theta_\mathbf{k}$ and $\mathbf{s_k}$ follow Gaussian distribution, i.e., $\theta_\mathbf{k} \sim \mathcal{N}(\mathbf{m_k}, \Sigma_\mathbf{k})$, $\mathbf{s_k} \sim \mathcal{N}(0, \mathbf{Q_k})$.

Observation equation. The observed distance, $\mathbf{z_k}$, can be modeled as two points in Euclidean space as follows[5],

$$\mathbf{z_k} = (\sqrt{(P_{x,k} - A_{x,i})^2 + (P_{y,k} - A_{y,i})^2}) + \mathbf{u_k}. \quad (4)$$

where $A_{x,i}$ and $A_{y,i}$ are the positions of the antenna i that reports ϕ_k, and $\mathbf{u_k} \sim \mathcal{N}(\mathbf{0}, \mathbf{R_k})$ is Gaussian noise.

State estimation. When transition and observation equations are in place, Kalman filters can be invoked iteratively to do state estimation. Note that due to the nonlinearity of our observation equation, the Extended Kalman Filter[22] (EKF) is needed. Then the observation equation becomes

$$\mathbf{z_k} = \mathbf{C}\theta_\mathbf{k} + \mathbf{u_k},$$

where $\mathbf{C_k} = (\frac{\partial \mathbf{z_k}}{\partial P_{x,k}}, \frac{\partial \mathbf{z_k}}{\partial P_{y,k}}, \frac{\partial \mathbf{z_k}}{\partial V_{x,k}}, \frac{\partial \mathbf{z_k}}{\partial V_{y,k}})$, is the observation matrix. Then the Kalman filter is able to predict the state at time $k+1$ using the transition equation,

$$\theta'_{\mathbf{k+1}} = \mathbf{A}\theta_\mathbf{k} \quad (5)$$

$$\Sigma'_{\mathbf{k+1}} = \mathbf{A}\Sigma_\mathbf{k}\mathbf{A}^\mathbf{T} + \mathbf{Q} \quad (6)$$

Afterwards, an update has to be done by combing observation equation and predicted $\theta'_{\mathbf{k+1}}$ as

$$\theta_{\mathbf{k+1}} = \theta'_{\mathbf{k+1}} + \mathbf{K_{k+1}}(\mathbf{z_{k+1}} - \mathbf{C_{k+1}}\theta'_{\mathbf{k+1}}) \quad (7)$$

$$\Sigma_{\mathbf{k+1}} = (\mathbf{I} - \mathbf{K_{k+1}}\mathbf{C_{k+1}})\Sigma'_{\mathbf{k+1}}, \quad (8)$$

where $\mathbf{K_{k+1}} = \Sigma'_{\mathbf{k+1}}\mathbf{C}^\mathbf{T}_{\mathbf{k+1}}(\mathbf{C_{k+1}}\Sigma'_k\mathbf{C}^\mathbf{T}_{\mathbf{k+1}} + \mathbf{R_{k+1}})^{-1}$ is the Kalman gain that balances transitions and observations.

The above four equations are for one round. Iterative executions for a candidate trajectory from the beginning to the end and then for all the candidate trajectories make all the hidden states properly estimated. There are several distinct features of our HMM framework with multiple Kalman filters worth mentioning.

- As velocities are estimated together with each position, position estimates that do not match velocity history will be devalued significantly by the Kalman gain, reducing ambiguities.
- To estimate the next state, we only need a single phase measurement, which accommodates the asynchronous reading mode and differs from all previous approaches [28, 29, 45–47, 53].
- The Gaussian mixture model used by previous systems [28, 53] is no longer needed as we already discretized it into multiple separate Gaussian models for Z_k.

[5]We describe it in 2D for simplity.

3.2 Fast Inference Using Nearest Neighbor

While the aforementioned HMM framework seems settled, it has a big drawback, high complexity. Specifically, as more phase readings come in, the candidate trajectories increase exponentially, which conflicts the goal of real-time tracking. As the reading time is already minimized by using only single-phase updates, we design two techniques to achieve computation efficiency, nearest neighbor pruning and initial state pruning.

3.2.1 Nearest Neighbor Pruning. The basic idea of nearest neighbor pruning is to reduce the number of candidate trajectories by only connecting the nearest distance observation. It works as follows. Suppose the state at time k of one candidate trajectory is $\theta^\mathbf{i}_\mathbf{k}$, then at time $k+1$, this state is only transit to state $\theta^*_{\mathbf{k+1}}$ whose observation distance $\mathbf{z}^*_{\mathbf{k+1}}$ is the nearest to the predicted state $\theta'_{\mathbf{k+1}}$. In particular, the pick of $\mathbf{z}^*_{\mathbf{k+1}}$ follows

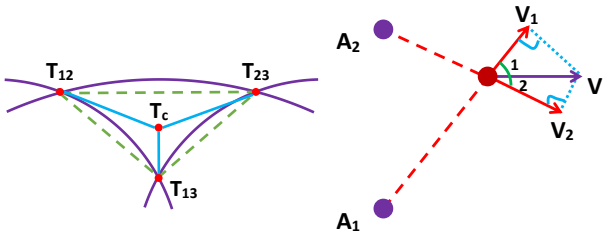
$$\mathbf{z_{k+1}} = \frac{\phi_{k+1}}{2\pi}\lambda + \frac{\lambda}{2}\arg\min_{i \in Z} |\frac{\phi_{k+1} + i\pi}{2\pi}\lambda - d_{k+1}|, \quad (9)$$

where d_{k+1} is the distance between $(P'_{x,k+1}, P'_{y,k+1})$ and antenna i that reports phase at time $k+1$.

Apparently, our nearest neighbor pruning is a greedy algorithm, which may achieve local optimum. But such a sacrifice is worth it because it ensures that the number of candidate trajectories is not growing over time, achieving great time efficiency. While one may think our nearest neighbor strategy is similar to the pitfalls of previous systems shown in Section 2, it has at least two major differences. First, it does not pose any limit on location and velocity estimation, which is different from that prior systems explicitly or implicitly set mobility constraints through location estimates. Second, even if sometimes the nearest neighbor strategy chooses the wrong candidate, it could be offset somehow by our observation and system mobility matrices through Kalman gains, whereas prior systems do not have such correction.

3.2.2 Initial State Pruning. As the nearest neighbor strategy guarantees the number of candidate trajectories is the same as the number of initial states, we look to how to estimate and further reduce this number.

Initial position estimation. To estimate initial positions, we can use trilateration that estimates intersections of circles from multiple antennas. But we can only get a phase reading at a time, which brings multiple concentric circles that would not intersect. To solve this, we employ phase interpolation. Suppose during a short time period (t_a, t_b), we obtain at least two phase readings for each antenna. Then for antenna j, we interpolate raw phases to obtain initial phases as $\hat{\phi}^j_0$ at a common reference time, t_x ($t_a < t^i_x < t_b$). This way, different



(a) Position estimation (b) Velocity estimation

Figure 4: The schematic diagram of (a) position estimation: The centroid of the triangle is the estimated position. (b) velocity estimation: The projection of velocity can be used to estimate the original velocity.

antennas' $\hat{\phi}_0^j$ are synchronized at t_x. Note that raw phases need to be unwrapped before interpolation.

We then translate each interpolated phase to multiple distance candidates similar to z^n, which makes a number of circles for trilateration. In theory, every three circles make an intersection estimation, as shown in Figure 4a. These circles will intersect at a maximum of 6 points. We choose the closest three points and make their centroid as the estimated position. To reduce the number of estimated positions, we assume that the correct triangle should not have a too large perimeter. Therefore, we set a threshold D_{th}, and the centroid will be removed from the candidate set if the perimeter of the associated triangle is greater than the threshold. Note that while using the perimeter threshold is inspired by TurboTrack [28], our novelty lies in how to synchronize asynchronous phase readings and estimate accurate initial positions, which helps fast convergence for Kalman filters.

Initial velocity estimation. Initial velocity has to be estimated as well. The key idea of velocity estimation is recovering the original velocity from projected velocity, as shown in Figure 4b. First, for every antenna, we calculate the projected velocity, which is in the direction of the line connecting the antenna and the tag. We then unwrap the phases and transform the phase difference $\Delta\phi_i$ of antenna i between two successive readings to the projected velocity $|\mathbf{V_i}| = \frac{\Delta\phi_i}{4\pi}\frac{\lambda}{\Delta t}$, where Δt is the time interval between these two readings. At the same time, the projection velocity can be derived from the original velocity as, $|\mathbf{V_i}| = |\mathbf{V}|\cos\angle i$, where $\angle i$ is the angle between the original velocity and the projection velocity $\mathbf{V_i}$. Yet, this relationship is non-linear and thus requires transformation. Let $\mathbf{V_i}$'s unit vector $\mathbf{e_i} = (e_{i,x}, e_{i,y})$ and $\mathbf{V} = (v_x, v_y)$. We can obtain $|\mathbf{V_i}| = \mathbf{e_i}\cdot\mathbf{V} = e_{i,x}v_x + e_{i,y}v_y$. Now, there are only two unknown variables v_x and v_y, which means as long as we can get some more interpolated phases, applying linear least square is sufficient to make accurate velocity estimates.

3.2.3 Optimal Trajectory. Finally, we show how to fast compute the likelihood of pruned trajectories by iteratively propagating beliefs.

Likelihood at initial time. Assume we have interpolated phases $\hat{\phi_{i,0}}$, $1 \le i \le M$. The likelihood at initial time is

$$\mathbf{L}_0 = \prod_{i=1}^{M} P(\theta_\mathbf{0}|\hat{\phi_{i,0}}) \propto \prod_{i=1}^{M} P(\hat{\phi_{i,0}}|\theta_\mathbf{0}) = \prod_{i=1}^{M} \mathcal{F}(d_0^i; z_0^*, R_0), \tag{10}$$

where d_0^i is the distance between antenna i and initial position $(P_{x,0}, P_{y,0})$, z_0^* is the candidate distance closest to $(P_{x,0}, P_{y,0})$, and $\mathcal{F}(x;\mu,R)$ is a Gaussian probability density function with a mean of μ and a variance of R. If we assign weights equally across all interpolated phases, $\mathbf{L}_0$ can be approximated as $\prod_{i=1}^{M}\mathcal{F}(d_0^i; z_0^*, R_0)$ because we normalize all the likelihoods of trajectories at every time snapshot.

Likelihood update at time $k+1$. Following a trajectory, if we have the trajectory's likelihood at time k, its likelihood the time $k+1$ can be derived from the transition and observe equations as

$$\mathbf{l}_{k+1} = \mathbf{l}_k p(\mathbf{z}_{\mathbf{k+1}}^*|\theta_{\mathbf{k+1}})p(\theta_{\mathbf{k+1}}|\theta_\mathbf{k}), \tag{11}$$

$$p(\mathbf{z}_{\mathbf{k+1}}^*|\theta_{\mathbf{k+1}}) = \mathcal{F}(d_{k+1}; \mathbf{z}_{\mathbf{k+1}}^*, \mathbf{R_{k+1}}), \tag{12}$$

$$p(\theta_{\mathbf{k+1}}|\theta_\mathbf{k}) = \mathcal{F}(\theta_{\mathbf{k+1}}; \theta_\mathbf{k}, \mathbf{Q_{k+1}}), \tag{13}$$

where $\mathcal{F}$ is a multidimensional Gaussian probability density function. Similarly, all the likelihood of trajectories are normalized for time $k+1$.

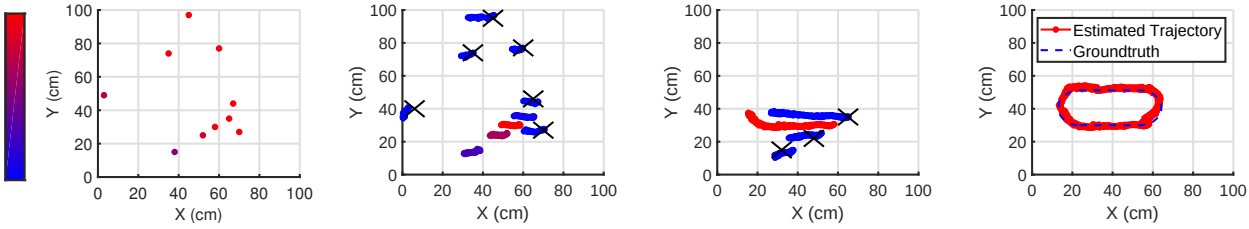
3.3 Putting It All Together

We summarize the major steps of GLAC as follows:

(1) Initially, we interpolate phases and use initial state pruning to obtain a handful of quality initial positions and velocities.
(2) For each candidate trajectory during an update, we use nearest neighbor pruning to select the next distance observation and employ an EKF to estimate hidden mobility states and the likelihood of the trajectory.
(3) We normalize likelihoods among trajectories at every time snapshot and the trajectory of the highest likelihood is chosen as the optimal one.

Figure 5 shows how the optimal trajectory gains likelihood through mobility over time. A few additional points are worth noting:

- **Quick stop.** We observe that after several iterations, most of the likelihoods are concentrated on a few trajectories, while the likelihoods of others are close to zero. This motivates us to further expedite computation by setting up a likelihood threshold l_{th} because it



(a) Initial estimation (t=0) (b) t=1s (c) t=5s (d) t=15s

Figure 5: GLAC working process. We show the trajectories with the top 10 likelihoods in 2D tracking. The trajectories with low likelihoods quickly stop as in (b) and (c). As time goes by, the optimal trajectory wins as it is the most likely one that fits the mobility model.

is almost impossible for the trajectory with a super-low likelihood to be the final winner. Therefore, trajectories that have likelihoods less than l_{th} will be removed and stop updating.

- **Recovery.** Due to various interferences, e.g., severe frequency selective shading, phase readings may have abrupt offsets. To address this, we set a threshold $3\sigma_v$ to discriminate such outliers. In particular, when the velocity change between two adjacent states is above $3\sigma_v$, we will replace this abnormal observation using the predicted state.

4 IMPLEMENTATION

Equipment. We build a prototype using a Thingmagic M6e reader [7] with 4 omnidirectional antennas, which are placed on a plane at (0,0), (0,30), (0,80) and (80,80)[6]. Target objects are attached with multiple kinds of tags, including Alien 9640 [2], ImpinJ Monza 4D [5], and SMARTRAC DogBone [4]. For 2D-tracking, we use a C1-intelligent car, which is remotely controlled through wireless, whereas for 3D-tracking, a VANBOT 4DoF robotic arm [9] and a toy drone are tested, as shown in Figure 10.

The localization software is programmed using Java and Mercury APIs [8], which collect phase readings from the reader to a laptop. Two threads are involved; one for GUI and the other for trajectory updates when a new phase reading comes in. The whole system works in real-time.

Experimental Environment. To obtain ground truth, we use an OptiTrack system [6], which costs $200,000. It is a vision-based camera-array system that can capture 3D-motions with 0.1 mm accuracy. It also has 360 fps and 2.8 ms latency, which qualifies for position ground truth. The experiments are conducted in various indoor environments, including classrooms with tables and chairs, library with benches and bookshelves, and lab environments full of workcounters and computers. Both LoS and NLoS tests are performed. Similar to past works [28, 47], the NLoS scenarios are built with putting separators to block the LoS between the antenna and the RFID tag, and since OptiTrack can only work in LoS, we let the tracking marker remain in LoS for cameras while keeping the tag in NLoS for antennas.

Competition. State-of-the-art competitors includes:

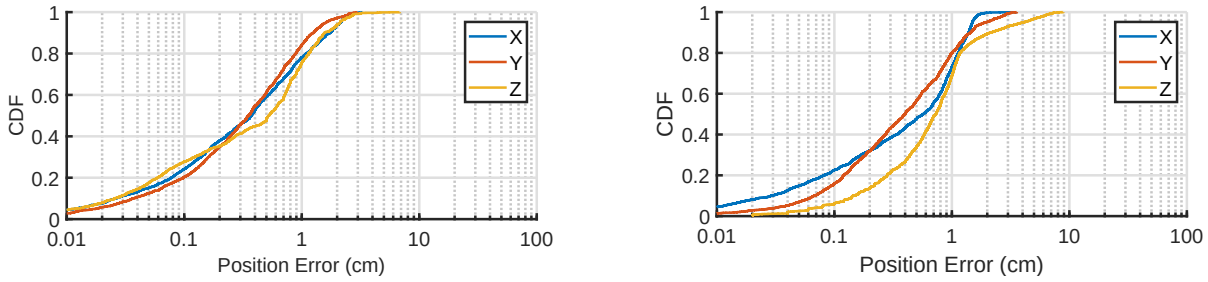
- **TurboTrack** [28] represents an high-precision tracking system achieving sub-centimeter accuracy for 3D. For fair-comparison purposes, TurboTrack here is reproduced with COTS devices, instead of SDRs. Also, one-shot wideband estimation is not implemented as it is not compatible with COTS readers. All the rest modules are faithfully reproduced.
- **RF-IDraw** [47] is a representative of accurate shape-tracking system. But the original proposal requires 8 antennas and 2 readers, which is far more than other competitors. So for fair competition, we adopt a 4-antenna version [40] that keeps all of the essential components and only differs in antenna placement.
- **Tagoram** [53] stands for decimeter-level tracking using COTS RFIDs. It can track only in 2D as its computation overhead is overwhelming for 3D. Our implementation set the resolution to 1000×1000. Note that the known-track version in the original paper is not suitable for robotic tracking and thus not implemented.

5 EVALUATION

5.1 Performance

We first examine GLAC's performance of 3D localization in detail, and then compare it against three state-of-the-art systems with respect to position accuracy, velocity accuracy, and time-efficiency.

[6]This is not mandatory as our solution supports arbitrary placements and the implicit coordinate system is in centimeters.



(a) LoS (b) NLoS

Figure 6: CDFs of GLAC's position errors in x/y/z dimensions for LoS and NLoS scenarios.

5.1.1 3D Accuracy. We evaluate GLAC's 3D tracking accuracy in both LoS and NLoS scenarios. More than 50 experimental trials are conducted, which amounts to over 10,000 phase readings. For each trial, we program the robotic arm to move the target object along both regular and arbitrary tracks. The position error is calculated as the Euclid distance between the estimated location and the ground truth.

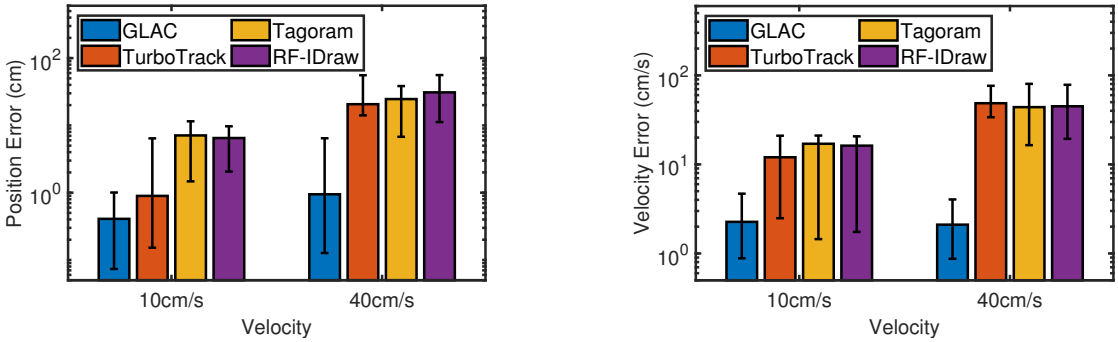
Figure 6 shows CDFs of position errors for both LoS and NLoS cases. We observe that the median accuracies of all x/y/z dimensions are within one centimeter for all scenarios. In particular, the median accuracies of x/y/z dimensions are 0.35 cm, 0.35 cm and 0.52 cm in LoS settings, while counterparts are 0.57 cm, 0.40 cm and 0.73 cm in NLoS settings, which degrade slightly due to lower SNRs. Moreover, the 90th percentiles are within 2cm in both scenarios for each dimension. Those results fully manifest GLAC's ability for high-precision tracking. This is mainly attributed to our HMM framework that removes ambiguities, which lets the optimal trajectory stand out by gaining more and more likelihoods through mobility.

5.1.2 Mobility Comparison. Next, we intend to investigate how GLAC and prior arts perform in different mobilities. The intelligent car is programmed to move along the track at various velocities from 10 cm/s to 50 cm/s. We have conducted 50 experimental trials in 2D for fair comparison, as RF-IDraw and Tagoram only support 2D-tracking.

Figure 7a shows detailed comparisons of position errors under two representative settings: 10 cm/s for slow movement, and 40 cm/s for fast movement. The results show that GLAC is significantly better than all three state-of-the-art systems both in slow-moving and fast-moving cases. Specifically, at a velocity of 10cm/s, the median position accuracies of GLAC, TurboTrack, Tagoram and RF-IDraw are 0.41 cm, 0.90 cm, 7.09cm and 6.09 cm, respectively. When the velocity increases to 40cm/s, GLAC's median position accuracy degrades slightly to 0.95 cm. In contrast, those of TurboTrack, Tagoram and RF-IDraw surge to 20.7 cm, 20.6 cm and 31.0 cm, respectively. In other words, in the slow-moving case, GLAC's median position accuracy is 2.2×, 17.3×, and 14.9× better than TurboTrack, Tagoram, and RF-IDraw, and when it comes to the fast-moving case, those gaps enlarge to 21.8×, 21.7×, and 32.6×.

Besides absolute position, another important metric of mobility is velocity. We compare GLAC's velocity error against three competitors, as shown in Figure 7b. While three competitors do not provide explicit velocity computation, we approximate that as the quotient of the position and time differences between two successive states. Again, the velocity estimation results demonstrate that GLAC is consistently better than prior arts for both dynamic scenarios. In particular, for the slow-moving case, the median velocity accuracy of GLAC is 2.26 cm/s, which is 5.3×, 7.6×, and 7.2× better than TurboTrack, Tagoram, and RF-IDraw, respectively; with the velocity increased to 40 cm/s, the gaps grow to 23.1×, 21×, and 21.4×. Note that although TurboTrack achieves subcentimeter median position accuracy in the slow-moving case, the corresponding velocity error is as large as 12 cm/s, which is far from 2.26 cm/s, the counterpart of GLAC. This reaffirms that accurate position estimates alone do not guarantee the quality of velocity estimates. Two main factors are contributing to the above performance gaps.

- Prior arts do not take care of asynchronous phase readings. When fasting-moving tags meet low-rate phase readings, the position errors inevitably surge like a rocket. In contrast to that, GLAC novelly synchronizes phases from different antennas for initial state estimation and customizes Kalman filters that can take a single phase for each update.
- In addition, those state-of-the-art systems barely pay attention to mobility models and thus can only derive velocity from position estimates that already contain non-negligible errors. As a result, when the tag moves at a high velocity, position errors inevitable cascade



(a) Position error (b) Velocity error

Figure 7: Position and velocity errors of slow-moving (10 cm/s) and fast-moving (40 cm/s) motions for GLAC, TurboTrack, Tagoram, and RF-IDraw. Error bars indicate the 10th and 90th percentiles.

over velocity estimates and future position states. Instead, GLAC models the velocity and position together as the mobility status and update it in an accurate way.

5.1.3 Time-Efficiency Comparison. Other than high precision, real-time is one of our design goals. We evaluate the time efficiency of GLAC together with prior systems. We decouple the total time of deriving one position estimate into communication time for reading phases and computation time for localization.

Table 2 shows the average time cost of over 100 experiment trials. The results clearly show that GLAC is the most time-efficient system as it achieves the overall time gains of 4× over TurboTrack, 72× over Tagoram, and 4× over RF-IDraw. We break these gains down and find that communication is the dominating factor, except for Tagoram. In particular, GLAC is the winner for communication overhead as it only requires a single phase to predict the next location, whereas the other three systems need at least 4 packets [7]. As to computation overhead, GLAC is 5.2× and 24300× better than TurboTrack and Tagoram. This is because TurboTrack uses Particle Filter, which requires a large number of particles to represent the Gaussian distribution, while GLAC employs EKFs that efficiently uses a covariance matrix for Gaussian. Tagoram's huge delay stems from its inefficient exhaustive search, which fits $W \times L$ trajectories and compares their PSNRs to select the optimal one. Note that RF-IDraw is the best computation-efficient system, but after counting communication time in, it is not the winner for the overall time cost. In short, GLAC achieves the best overall time-efficiency among all thanks to shortest reading time brought by single-phase update and decent computation time resulted from our fast inference scheme.

[7]Other work [56] has reported faster reading rates through a set of rate adaption techniques, which we plan to investigate in the future for further performance optimization.

Table 2: Localization time comparison.

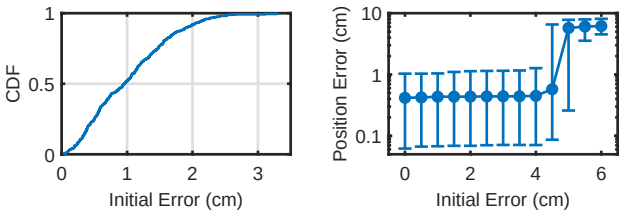
	Reading	Computation	Total
GLAC	28ms	78us	28ms
TurboTrack	111ms/4.0×	408us/5.2×	111ms/4.0×
Tagoram	111ms/4.0×	1896ms/24300×	2007ms/72×
RF-IDraw	111ms/4.0×	3us/0.038×	111ms/4.0×

5.2 Micro Benchmarks

After evaluating the overall performance, we now focus on the performance of constituent modules, which helps understand how GLAC works in detail.

5.2.1 Initial Position Accuracy. First, we examine the accuracy of initial position estimation, which returns a number of estimated initial states, rather than a single one. As the HMM framework is able to remove ambiguities, here we always choose the initial position nearest to the ground truth as the representative and report initial position errors as the distance between this representative and ground truth. This way, the impact of other components is totally removed. As shown in Figure 8a, our initial position estimates are very accurate as the median error is only 0.97 cm and the 90th percentile is 1.92 cm. The main reason for this successful estimation is that we make synchronized phases across antennas through interpolating unwrapped phases.

In addition to that, we conduct experiments to quantify the impact of those initial estimation errors on the trajectory tracking. We randomly inject initial position errors ranging



(a) Initial position accuracy (b) Impact of initial accuracy on localization accuracy

Figure 8: Initial position estimation. (a) plots the CDF of the distance between the nearest initial estimate and the ground truth. (b) shows the impact of initial errors on the trajectory's median position error. Error bars indicate the 10th and 90th percentiles.

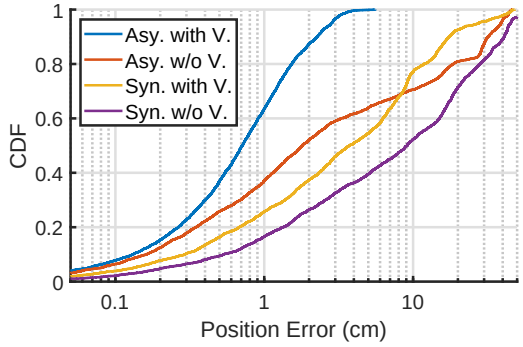


Figure 9: Decomposing mobility gains. This figure plots the CDFs of position errors for different partial implementations of GLAC.

from 0-6 cm based on the ground truth, and estimate corresponding velocities. Then EKFs are used to reconstruct trajectories. The results are shown in Figure 8b. We observe as long as the initial error is less than 4 cm, GLAC can achieve sub-centimeter tracking accuracy. Recall that the 90th percentile of GLAC's initial position error is only 1.92 cm. As a result, our initial position estimates are sufficiently accurate to deliver millimeter-level trajectory accuracy.

Another observation from Figure 8b is that when the initial error is within 4 cm, the median position error remains stable. This is because EKFs converge to the true trajectory after a few iterations, i.e., only a handful of positions may be affected, which do not make too much difference for median errors statistically. One may think that, due to phase ambiguities, an EKF could converge to another trajectory when the initial error is large. Nevertheless, GLAC starts a series of EKFs and employs likelihood evaluation to choose the optimal one, which hardly go wrong with the help of mobility modeling.

5.2.2 Gains of Mobility Modeling. Next, we would like to quantify the effectiveness of our mobility model, namely velocity modelling and ability to dealing with asynchronous phase readings. In particular, we make three variants with partial implementation of the standard GLAC process. The four competitors are as follows.

(1) **Asynchronous handling & velocity modeling.** This is the standard GLAC, which is used as the baseline.
(2) **Asynchronous handling without velocity modeling.** The first variant scheme is to remove velocity modeling, which changes Equation 3 as follows:

$$\theta_{k+1} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \theta_k + \mathbf{s_k}.$$

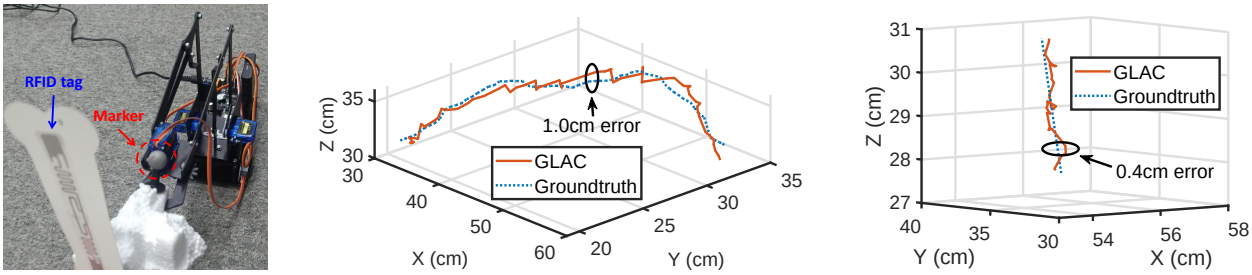
This is built to examine how much the velocity modeling helps localization.
(3) **Synchronous handling with velocity modeling.** The second variant scheme is to investigate the impact of asynchronous handling. So we treat every 4 successive phase readings from 4 antennas as four synchronous phase readings, which is the way widely adopted in most prior arts [28, 47, 53]. Specifically, we change Equation 4 to:

$$\mathbf{z_k} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \theta_k + \mathbf{u_k}.$$

Note that $\mathbf{z_k}$ here is the position estimates obtained by the trilateration similar in initial position estimation, and we pick the one nearest to the predicted position as $\mathbf{z}^*_k$ for fast inference. As to likelihood computation, $p(\mathbf{z}^*_{k+1}|\theta_k)$ is replaced by $\prod_{i=1}^{4} p(\mathbf{z}^*_{i,k+1}|\theta_k)$, where $\mathbf{z}^*_{i,k+1}$ is the candidate distance nearest to the predicted state $\theta_{k+1}{}'$, as we assume the phase readings across 4 antennas are independent.
(4) **Synchronous handling without velocity modeling.** The last scheme is the combination of (2) and (3).

The above four schemes are fed with the same input including various mobilities (from 10 cm/s to 50 cm/s) and output 2D coordinates for simplicity. The CDF results are plotted in Figure 9, and we have the following observations:

- Unsurprisingly, the standard GLAC is the best. In particular, the median position errors of the above four schemes are 0.71 cm, 1.80 cm, 3.73 cm and 9.12 cm.
- Velocity modeling indeed effectively boosts the system performance. In particular, the median position accuracy of scheme 1 is 2.5× better than that of scheme 2. Likewise, scheme 3's median position accuracy is 2.4× better than scheme 4's. The main reason is that



(a) Robotic arm and target tag (b) Rotating (c) Lifting

Figure 10: Robotic arm tracking: (a) shows the robotic arm carrying an item attached with an RFID tag and an OptiTrack Marker. (b) and (c) show the trajectories of GLAC and the ground truth for rotating and lifting operations.

without proper velocity modeling, the transition equation's expressiveness becomes quite limited because it only describes a stationary model where all motions are considered as noise, resulting in huge mobile localization errors.

- Asynchronous handling also constructively improves localization accuracy. Specifically, the median accuracies of asynchronous handling (schemes 1&2) are at least 5× better than synchronous handling counterparts (schemes 3&4). The root cause is that synchronous handling may be a good approximation when frame rates are high, like in [28]; however, it inevitably brings more errors when the reading rate (e.g., 30 reads/s) and high-mobility (e.g., 30 cm/s) happen at the same time.

5.2.3 Gains of Fast Inference. In addition, we intend to evaluate the computation-efficiency brought by our fast inference design. We compare two schemes, GLAC with fast inference and that with naive inference. So, we run two schemes at the same time with the same data, and then check how well they perform at the same time snapshots.

Results are listed in Table 3. We observe that fast inference is significantly better than naive inference in terms of time-efficiency and accuracy. For example, fast inference achieves a median position accuracy of 0.37 cm with 20 ms, whereas naive inference's accuracy is 32.4 cm with the same amount of time. This is mainly due to that fast inference guarantees the number of trajectories (no. of EKFs) does not increase with time. In consequence, it can go further for iteration depth, which means mobility effectively reduces ambiguities. In particular, with 500 ms, fast inference is left with a trajectory of depth 7,8000 (clearly the optimal one), whereas naive inference's iteration depth is only 3, which makes tons of EKFs (98,000) inefficient. Even with 10s, a large number of EKFs with shallow depths only make a median accuracy of 24.1 cm. Hence, deep iteration depth brought by fast inference is the key to real-time high-precision.

Table 3: Time and accuracy for fast inference and naive inference.

Time	Fast inference/naive inference			
	Position error (cm)	Velocity error (cm/s)	# of EKFs	Iteration depth
10ms	41.1/41.1	4.21/4.21	413/413	1/1
15ms	25.0/41.1	3.90/4.21	145/413	8/1
20ms	0.37/32.4	1.52/4.16	52/6036	38/2
500ms	0.36/27.5	1.49/3.88	1/9.8E4	7.8E4/3
10s	0.37/24.1	1.47/3.74	1/1.4E6	4.2E6/4

Another observation is with the help of quick stop and fast inference, the optimal trajectory converges even faster. Recall that without fast inference, the number of candidate trajectories increases exponentially as there are n candidate distance observations with a single phase. And each candidate trajectory requires an EKF for updating. Specifically, the number of trajectories for naive inference is $O(n^\kappa)$, where κ the number of time snapshots, and that of fast inference is $O(n)$. After combined with quick stop, the number of trajectories for fast inference at 500 ms plummets to only 1, which is 400× lower than the number of trajectories at 10 ms.

5.3 Precise Manipulation Showcase

Finally, we show the qualitative performance of GLAC's precise 3D manipulation in Figure 10. During rotating and lifting operations by the robotic arm, GLAC is able to track 3D-movements at sub-centimeter accuracy. Such results clearly demonstrate GLAC's great potential to enable a range of

high-precision robotic applications, such as furniture assembly through multi-robot cooperation, and even robot-assist precise surgeries.

6 RELATED WORK

RF-Localization. There has been much work on RF localization using COTS and SDR devices, including LTE [17, 51], Bluetooth [15, 36], RFID [28, 47, 53] and WiFi [26, 44, 52]. Among these, RFID-localization can be broadly classified into two categories: fingerprint and trigonometry. The fingerprint-based methods are built based on the assumption that each distinct location has its own unique RF-signature [34, 46], while the trigonometry-based methods are generally more accurate as both triangulation [12, 48] and trilateration [28, 29] are derived from fine-grained phase measurements.

While GLAC falls into the trigonometry category and shares similarity with prior arts from a high level, the crucial difference is that we aim to realize real-time 3D localization using only COTS RFID devices. Bearing this in mind, we novelly model mobility using the HMM and employ it to reduce ambiguities without any other extra hardware or assumptions. This is distinct new thinking and hope it will fuel more community interests along this line.

Linear Gaussian Models. It is known that factor analysis [21], mixtures of Gaussian clusters [32], Kalman filter [50], and HMM [14] can be generalized to linear Gaussian modeling [38, 39], which has extensive applications, such as inertial navigation, speech recognition, and stock market forecasting. While all those approaches are trying to solve the general problem of estimating actual states from time linear dynamic observations, there is no generic solution to specific problems. Based on existing fruitful literature of linear Gaussian models, our work is custom-built. In particular, as the standard type of HMM usually considers discrete states, we carefully examine our problem and build our own HMM where ambiguous states at a time are discrete and do not transit, whereas temporal states are continuous and estimated through Kalman filters. We also design a fast inference scheme to choose the optimal trajectory in real-time.

Mobility Model. Mobility is an important metric and has been widely used in wireless networks from many aspects, including network capacity [19, 43], rate adaption [42, 56], throughput control [27], and wireless sensing [49]. Nevertheless, those works are mainly focused on coarse-grained mobility. For example, Blink [56] only detects the tag is moving or not. Tagwatch [27] requires a couple of seconds to derive mobility statues. On the contrary, this work models and derives fine-grained mobility, which is complementary to the above works and thus can be integrated to improve wireless networks, e.g., better network coverage for mobile nodes and higher throughput by being aware of mobility.

7 DISCUSSION & CONCLUSION

Finally, a few points are worth elaborating on,

- **Requiring Motion.** One limitation of GLAC stems from fact we need mobile targets to distinguish ambiguities and pick the right trajectory. For static objects, our initial position estimation can be invoked as many as needed. As shown in evaluations, the 90^{th} percentile error for such is less than 2 cm, which is sufficiently accurate for many purposes. Of course, we can also design a learning algorithm to identify whether the object is moving or not by going through a training process, which is worth further investigation as it would require re-training for various objects.
- **Working Range.** As our system is built upon COTS RFID devices, it inherits the relatively short working range, which is about 7-10 meters. Yet, such a limitation is not fundamental to our design. To break it, we intend to explore beamforming [16], which uses multiple antennas to boost signal strengths arriving at target tags. Other possibilities include semi-passive/active tags [3], and novel backscatter paradigms [55], which can typically work at longer distances, 50-200 meters.
- **Manipulatee Size.** Currently, the size of objects that robots manipulate is limited to commercial tags' sizes. While a typical RFID chip has about a $500\mu m \times 500\mu m$ die size, which is indeed tiny, the dimensions of tag inlays are much larger, e.g., an ImpinJ H47 wet inlay is of $47mm \times 47mm$, and an Alien 9640 inlay is $95mm \times 8mm$. To push such limits, we can either adopt antenna reduction techniques, e.g., Circular Loop Antenna [20], or try RFIDs at much higher frequencies [11] that have much shorter wavelengths.
- **Transferability.** Our key idea that mobility improves accuracy is general and can be extended to other RF systems, such as Bluetooth, WiFi, and LTE. Nevertheless, such adaptations have to take care of RF receivers' form factors, which are usually larger than RFIDs, and identification, which is not natively supported. Other constraints, assumptions, and requirements of those RF systems also pose new challenges, which we are going to investigate in future work.

Overall, we believe that GLAC marks an important step towards precision robot control using COTS RFID systems. It pushes the envelope of real-time RFID localization and tracking to the millimeter-level accuracy without requiring any extra hardware and restrictive mobility for tags and readers. By doing so, it paves the way for fast and wide adoption of cheap and readily available commercial RFIDs in robotic applications demanding high-precision, e.g., welding, assembly, and surgeries.

REFERENCES

[1] 2019. MimioTeach Interactive Whiteboard. (2019). https://mimio.boxlight.com

[2] 2020. Alien 9640 UHF RFID tag. (2020). https://www.alientechnology.com

[3] 2020. Confidex Viking product family. (2020). https://www.confidex.com/smart-industries/confidex-viking/

[4] 2020. DOGBONE with NXP UCODE G2iL. (2020). https://www.smartrac-group.com/files/content/Products_Solutions/PDF/0053_SMARTRAC_DOGBONE.pdf

[5] 2020. MONZA 4D RAIN RFID TAG. (2020). https://www.impinj.com/platform/endpoints/monza-4d

[6] 2020. OptiTrack. (2020). https://www.optitrack.com

[7] 2020. ThingMagic M6e. (2020). https://www.jadaktech.com

[8] 2020. ThingMagic Mercury API. (2020). https://www.jadaktech.com/products/thingmagic-rfid/thingmagic-mercury-api/

[9] 2020. VANBOT. (2020). https://cheng-long.taobao.com

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

[11] Ali Attaran, Rashid Rashidzadeh, and Roberto Muscedere. 2017. Rotman lens combined with wide bandwidth antenna array for 60 GHz RFID applications. *International Journal of Microwave and Wireless Technologies* 9, 1 (2017), 219–225.

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

[13] J. C. Byrn, S. Schluender, C. M. Divino, J. Conrad, B. Gurland, E. Shlasko, and A. Szold. 2007. Three-dimensional imaging improves surgical performance for both novice and experienced operators using the da Vinci Robot System. *The American Journal of Surgery* 193, 4 (2007), 519–522.

[14] O. Cappé, E. Moulines, and T. Rydén. 2006. *Inference in hidden Markov models*. Springer Science & Business Media.

[15] S. S. Chawathe. 2008. Beacon placement for indoor localization using bluetooth. In *IEEE Conference on Intelligent Transportation Systems*.

[16] H. Cox, R. Zeskind, and M. Owen. 1987. Robust adaptive beamforming. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 35, 10 (1987), 1365–1376.

[17] J. A. del Peral-Rosado, J. A. López-Salcedo, G. Seco-Granados, F. Zanier, and M. Crisci. 2012. Achievable localization accuracy of the positioning reference signal of 3GPP LTE. In *IEEE International Conference on Localization and GNSS*.

[18] G. N. DeSouza and A. C. Kak. 2002. Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence* 24, 2 (2002), 237–267.

[19] M. Grossglauser and D. NC. Tse. 2002. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM transactions on networking* 10, 4 (2002), 477–486.

[20] H.-K. R. and J.-M. Woo. 2005. Size Reduction in UHF Band RFID Tag Antenna Based on Circular Loop Antenna. In *18th International Conference on Applied Electromagnetics and Communications*.

[21] H. H. Harman. 1976. *Modern factor analysis*. University of Chicago press.

[22] A. H. Jazwinski. 2007. *Stochastic processes and filtering theory*. Courier Corporation.

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

[24] K. Kamali, A. Joubair, I. A. Bonev, and P. Bigras. 2016. Elasto-geometrical calibration of an industrial robot under multidirectional external loads using a laser tracker. In *Proc. of IEEE ICRA*.

[25] N. Kamamichi, M. Yamakita, K. Asaka, and Z.-W. Luo. 2006. A snake-like swimming robot using IPMC actuator/sensor. In *Proc. of IEEE ICRA*.

[26] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti. 2015. Spotfi: Decimeter Level Localization Using WiFi. In *Proc. of ACM SIGCOMM*.

[27] Q. Lin, L. Yang, H. Jia, C. Duan, and Y. Liu. 2017. Revisiting Reading Rate with Mobility: Rate-Adaptive Reading in COTS RFID Systems. In *Proc. of ACM CoNEXT*.

[28] Z. Luo, Q. Zhang, Y. Ma, M. Singh, and F. Adib. 2019. 3D backscatter localization for fine-grained robotics. In *Proc. of USENIX NSDI*.

[29] Y. Ma, N. Selby, and F. Adib. 2017. Minding the billions: Ultra-wideband localization for deployed rfid tags. In *Proc. of of ACM MobiCom*.

[30] H. Meenink, R. Hendrix, G. Naus, M. Beelen, H. Nijmeijer, M. Steinbuch, E. van Oosterhout, and M. de Smet. 2012. Robot-assisted vitreoretinal surgery. In *Medical Robotics*. Elsevier, 185–209.

[31] P. Michelman. 1998. Precision object manipulation with a multifingered robot hand. *IEEE Transactions on Robotics and Automation* 14, 1 (1998), 105–113.

[32] K. P. Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.

[33] M. P. Murphy and M. Sitti. 2007. Waalbot: An agile small-scale wall-climbing robot utilizing dry elastomer adhesives. *IEEE/ASME transactions on Mechatronics* 12, 3 (2007), 330–338.

[34] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil. 2003. LANDMARC: Indoor Location Sensing Using Active RFID. In *Proc. of IEEE PERCOM*.

[35] T. Probst, K.-K. Maninis, A. Chhatkuli, M. Ourak, E. Vander Poorten, and L. Van Gool. 2017. Automatic tool landmark detection for stereo vision in robot-assisted retinal surgery. *IEEE Robotics and Automation Letters* 3, 1 (2017), 612–619.

[36] A. N. Raghavan, H. Ananthapadmanaban, M. S. Sivamurugan, and B. Ravindran. 2010. Accurate mobile robot localization in indoor environments using bluetooth. In *IEEE ICRA*.

[37] J. Rosen, L. N. Sekhar, D. Glozman, M. Miyasaka, J. Dosher, B. Dellon, K. S. Moe, A. Kim, L. J. Kim, and T. Lendvay. 2017. Roboscope: A flexible and bendable surgical robot for single portal minimally invasive surgery. In *Proc. of IEEE ICRA*.

[38] A-V. I. Rosti and M.J.F. Gales. 2001. Generalised linear Gaussian models. (2001).

[39] S. Roweis and Z. Ghahramani. 1999. A unifying review of linear Gaussian models. *Neural computation* 11, 2 (1999), 305–345.

[40] L. Shangguan and K. J. 2016. Leveraging Electromagnetic Polarization in a Two-Antenna Whiteboard in the Air. In *Proc. of ACM CoNEXT*.

[41] L. Shangguan and K. Jamieson. 2016. The design and implementation of a mobile RFID tag sorting robot. In *Proc. of ACM MobiSys*.

[42] L. Sun, S. Sen, and D. Koutsonikolas. 2014. Bringing mobility-awareness to WLANs using PHY layer information. In *Proc. of ACM CoNEXT*.

[43] S. Toumpis and A. J. Goldsmith. 2004. Large wireless networks under fading, mobility, and delay constraints. In *Proc. of IEEE INFOCOM*.

[44] D. Vasisht, S. Kumar, and D. Katabi. 2016. Decimeter-Level Localization with a Single WiFi Access Point. In *Proc. of USENIX NSDI*.

[45] J. Wang, F. Adib, R. Knepper, D. Katabi, and D. Rus. 2013. RF-compass: Robot object manipulation using RFIDs. In *Proc. of ACM MobiCom*.

[46] J. Wang and D. Katabi. 2013. Dude, Where's My Card? RFID Positioning That Works with Multipath and Non-Line of Sight. In *Proc. of ACM SIGCOMM*.

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

[48] J. Wang, J. Xiong, H. Jiang, X. Chen, and D. Fang. 2016. D-Watch: Embracing "Bad" Multipaths for Device-Free Localization with COTS RFID Devices. In *Proc. of ACM CoNEXT*.

[49] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu. 2015. Understanding and modeling of wifi signal based human activity recognition. In *Proc. of ACM MobiCom*.

[50] G. Welch and G. Bishop. 1995. An introduction to the Kalman filter. (1995).

[51] T. Wigren. 2012. LTE fingerprinting localization with altitude. In *IEEE VTC*.

[52] J. Xiong and K. Jamieson. 2013. ArrayTrack: A Fine-Grained Indoor Location System. In *Proc. of USENIX NSDI*.

[53] L. Yang, Y. Chen, X. Y. Li, C. Xiao, M. Li, and Y. Liu. 2014. Tagoram: real-time tracking of mobile RFID tags to high precision using COTS devices. In *Proc. of ACM MobiCom*.

[54] Z. Yaniv and L. Joskowicz. 2005. Precise robot-assisted guide positioning for distal locking of intramedullary nails. *IEEE transactions on medical imaging* 24, 5 (2005), 624–635.

[55] P. Zhang, D. Bharadia, K. Joshi, and S. Katti. 2016. Hitchhike: Practical backscatter using commodity wifi. In *Proc. of ACM SenSys*.

[56] P. Zhang, J. Gummeson, and D. Ganesan. 2012. Blink: A high throughput link layer for backscatter communication. In *Proc. of ACM MobiSys*.