radarTrack: Enhancing Ego-Vehicle Speed Estimation with Single-chip mmWave Radar

Argha Sen[††], Soham Chakraborty[†], Soham Tripathy[‡], Sandip Chakraborty[§]
Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur
Email: [††]arghasen10@gmail.com, [†]sohamc1909@gmail.com, [‡]sohamtripathy2001@gmail.com, [§]sandipchkraborty@gmail.com

***Abstract*—In this work, we introduce *radarTrack*, an innovative ego-speed estimation framework utilizing a single-chip millimeter-wave (mmWave) radar to deliver robust speed estimation for mobile platforms. Unlike previous methods that depend on cross-modal learning and computationally intensive Deep Neural Networks (DNNs), *radarTrack* utilizes a novel phase-based speed estimation approach. This method effectively overcomes the limitations of conventional ego-speed estimation approaches which rely on doppler measurements and static surrondings. *radarTrack* is designed for low-latency operation on embedded platforms, making it suitable for real-time applications where speed and efficiency are critical. Our key contributions include the introduction of a novel phase-based speed estimation technique solely based on signal processing and the implementation of a real-time prototype validated through extensive real-world evaluations. By providing a reliable and lightweight solution for ego-speed estimation, *radarTrack* holds significant potential for a wide range of applications, including micro-robotics, augmented reality, and autonomous navigation.**

***Index Terms*—Ego-Speed Estimation, mmWave Sensing, Phase-based Speed Estimation**

I. Introduction

Comprehending the movements of mobile agents, whether navigating through distant planets or mixed reality environments, is fundamental for adequate perception and interaction. Ego-motion estimation is crucial, as it does not depend on pre-existing maps or environmental infrastructure. Instead, it derives the agent's position and orientation from the data collected during movement. Accurate odometry is also crucial for developing maps, especially when using techniques like *Simultaneous Localization and Mapping* (SLAM) to dynamically build and update a model of the environment.

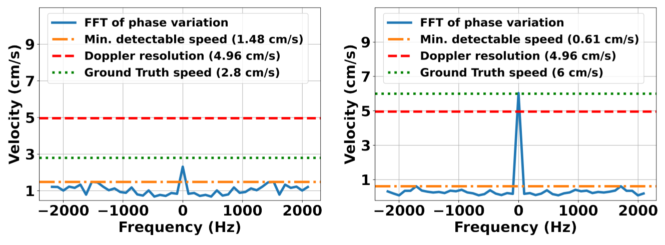
Due to their affordability and widespread availability, MEMS inertial sensors (IMUs) are widely utilized for ego-motion estimation across various mobile platforms. Estimating position from acceleration involves double integration, leading to errors [1]. Accelerometers struggle to detect acceleration in ground vehicles moving at constant speeds due to subtle changes in acceleration [2], [3]. Moreover, IMU often requires precise calibration, which is challenging and labor-intensive. This calibration often requires specialized and costly equipment, such as synchronized clocks, precision turntables, and motion tracking systems. Multimodal odometry systems have been developed to address these limitations, integrating inertial data with additional sensor inputs, such as visual or ranging information. Among these, visual-inertial odometry (VIO) stands out for its robustness and is commonly found in devices like mobile phones, offering a more reliable solution for motion estimation [2], [4], [5]. However, VIO performance can suffer or even fail in difficult lighting conditions, such as in darkness where RGB cameras are ineffective or in intense illumination where depth cameras may encounter glare. Similar visibility challenges affect LiDAR-inertial odometry (LIO), especially in the presence of airborne particles such as dust, fog, or smoke. Moreover, LiDARs tend to be bulky, heavy, and costly, making them more suitable for high-end robotics than micro-robots or wearables.

The first three authors have equal contributions.

mmWave based sensing offers distinct advantages over vision-based systems, particularly due to its robustness against environmental factors like scene illumination and airborne obscurants. Unlike mechanically scanning radars, single-chip mmWave radars utilize electronic beamforming, making them lightweight and ideal for micro-robots, mobile devices, and wearable technology. This technology is already being used for motion sensing in smartphones like Google Pixel [6] or obstacle detection in commercial drones [7], [8].

Traditional radar-based approaches estimate speed using doppler shifts, constrained by the Discrete Fourier Transform (DFT) to discrete steps of $\frac{\lambda}{2N_cT_c}$, where T_c is the chirp duration and N_c is the number of chirps transmitted by the radar in a single frame [9]. Speeds not matching these steps incur errors, especially at very low speeds ($v < \frac{\lambda}{2N_cT_c}$), placing them in the sub-doppler regime, as seen in microrobots.

To illustrate this limitation, we conducted a comparative study by mounting a radar on an ego-vehicle and placing a static object in front of it. By moving the ego-vehicle directly towards the static object, the radar captured phase variations from the reflected signals, and by applying FFT to these phase changes and scaling the results using the relation $v = \frac{\lambda}{4\pi}\frac{d\Phi}{dt}$, we obtained phase-based speed estimates [9]. Fig. 1 demonstrates the effectiveness of phase-based speed estimation compared to traditional doppler-based methods under different speed conditions. The blue line represents the FFT of phase variations from a static object in the radar's field of view (FoV), where the peak corresponds to the actual speed of the ego-vehicle. The noise level indicates the minimum achievable



(a) (b)

Fig. 1: Comparison of doppler-based and phase-based speed estimation for a radar-mounted ego-vehicle moving at (a) 2.8 cm/s, and (b) 6 cm/s. The noise floor indicates the minimum achievable speed detection limit.

speed detection limit for the phase-based approach. As shown in Fig. 1(a), when the ego-vehicle operates in the sub-doppler regime, the doppler-based method will fail to resolve the motion accurately, whereas the phase-based approach successfully estimates the speed. In contrast, when the ego-vehicle speed exceeds the doppler resolution threshold, as shown in Fig. 1(b), although the doppler-based method can estimate better, it still has resolution error, which is not the case for the phase-based approach. This highlights the advantage of phase-based speed estimation in scenarios where traditional doppler-based methods suffer from resolution constraints.

Creating a reliable indoor odometry system based on this concept faces several challenges. Firstly, radar returns are susceptible to noise from specular reflections, diffraction, and significant multipath effects. Also, due to hardware constraints on the number of antennas it results in highly sparse pointclouds (PCDs) with limited angular resolution. This low-quality data makes conventional LiDAR-based methods, such as *Iterative Closest Point* (ICP) [10], almost ineffective when applied directly to mmWave radar data. Secondly, while some previous works [11]–[15] have employed a multimodal approach by fusing radar data with other sensors, like IMUs and RGB cameras, the potential of mmWave radar to complement these modalities remains uncertain. Additionally, existing works perform well in environments with only static objects [12], [15], [16] but fail to perform adequately in the presence of dynamic objects. Moreover, incorporating recent advances in DNNs for visual or LiDAR odometry poses challenges due to the significant computational load, which may limit their use in mobile, wearable, and other resource-constrained devices. Our research explores methods to make mmWave radar more computationally efficient, focusing on leveraging only the phase values of the reflected signals. In our previous work [17], we demonstrated that phase-based ego-speed estimation outperforms traditional approaches. However, this method is constrained by the requirement of a static environment and is effective only when a stationary object is directly near the boresight angle of the radar's FoV. However, estimating the absolute ego-speed from phase variations of objects at oblique angles is not trivial.

Motivated by these gaps of traditional ego-speed estimation, in this work, we propose *radarTrack*, a novel ego-speed estimation framework that relies solely on the phase returns of mmWave radar. We have developed a fourth-order expression of these phase patterns in terms of the ego-speed and showcase how estimating the roots of this kinematics-based equation can achieve accurate speed estimation. By leveraging a purely signal processing-based pipeline, *radarTrack* enables us to achieve low latency in estimating ego-speed on embedded platforms compared to its DNN counterparts. The contributions are summarized as follows:

1) **Kinematics-based analytic equation for phase variation:** We derive an equation to represent the phase change in objects induced by the ego-vehicle's motion as a fourth-order equation in terms of ego-speed, based on kinematics laws. By solving the roots of this equation we have precise ego-speed estimation even for static objects positioned at oblique angles in front of the radar.
2) **Lightweight, radar-based ego-motion estimation across multiple platforms:** We present a novel ego-speed estimation framework implemented on three platforms- an Unmanned Ground Vehicle (UGV), a drone, and a hand-held stick- using a single commercial-off-the-shelf (COTS) mmWave radar. Our method eliminates the need for traditional, bulky setups with multiple sensors.
3) **Dynamic and static object segmentation:** Our method introduces an innovative framework that combines doppler-based speed profiling to effectively segment dynamic and static objects in cluttered environments, helping us to robustly estimate ego-speed in highly dynamic environments.
4) **Real-time edge processing for efficient ego-speed estimation:** Our lightweight signal processing enables real-time performance without relying on deep learning models. With a latency of only ≈ 0.29 s ($\approx 85\%$ lesser than its closest baseline, Radarize [16]) on a low-compute Jetson Nano, our method is ideal for mobile applications.

II. Related Work

We have classified the previous works on ego-motion estimation into two categories: (i) methods based on multi-modal fusion and (ii) methods that are purely radar-based.

Multi-Modal Sensor Fusion: Multi-modal sensor fusion combines data from various sensors, employing techniques from probabilistic fusion to deep learning-based attention mechanisms. Traditional methods like probabilistic fusion, fuzzy reasoning, and hybrid fusion have advanced with neural networks, allowing for data fusion at multiple levels—from raw data to final decisions [13]. Recent advancements incorporate self-attention mechanisms to dynamically evaluate input or feature importance, improving ego-motion estimation or other downstream applications on human activity recognition. Data-driven deep odometry models have significantly advanced, allowing them to effectively model complex physical processes and disturbances through end-to-end learning. These models have been employed across various sensors, including RGB cameras [18], depth cameras [19], [20], IMUs [21], [22],

TABLE I: Comparison of SOTA approaches with *radarTrack*.

SOTA Approaches	Dynamic	Radar Only	Doppler based	PCD based	Phase based
Milli-RIO [13]	×	× (IMU)	×	✓	×
Kramer et al [12]	×	× (IMU)	✓	×	×
Radarize [16]	×	✓	✓	×	×
EmoRI [23]	✓	✓	✓	✓	×
milliEgo [11]	×	× (IMU)	×	✓	×
Huang et al [14]	✓	× (IMU)	✓	✓	×
Cen et al [15]	×	✓	×	✓	×
4D iRIOM [24]	✓	× (IMU)	×	✓	×
radarTrack	✓	✓	×	✓	✓

and LiDARs [10]. MilliEgo [11] proposes a trajectory estimation technique using either co-located multi-radar setups, radar-inertial data, or radar-inertial-visual data. It uses DNNs to regress out six degrees of freedom (DoF) pose estimates, incorporating a mixed attention multi-modal sensor fusion module to leverage the complementary nature of different sensors (mmWave, IMU, Camera, etc.). While it achieves promising results, it faces challenges with large drifts in handheld scenarios and imperfect loop closures. Additionally, the DL component leads to high inference times. The main advantage of deep odometry models is their ability to bypass extensive calibration, making them accessible to non-experts.

Purely Radar-Based Methods: Unlike these methods which rely on a combination of sensors, several recent works like [15], [16], [23] focus solely on radar-based odometry using point association or doppler shift For example, Radarize [16] utilizes doppler-azimuthal heatmaps for translation estimation and range-azimuthal heatmaps for rotation estimation. The system processes consecutive heatmaps through neural networks to determine both the translation and rotation of the device, incorporating advanced techniques like echo suppression for multipath rejection. Despite achieving competitive accuracy, the method's reliance on neural networks and high computational requirements limit its real-time capabilities, especially in dynamic environments with numerous moving objects. In [17] authors have proposed ego-speed estimation using raw phase values of the mmWave radar, however the proposed system can only work with a single static object and when the ego-vehicle is moving in a straight line towards the static object. Also, the estimated speed is the relative radial speed of the static object. [23] focuses on improving ego-speed and trajectory estimation in environments with dynamic objects by employing a hybrid FFT-MUSIC algorithm to reduce angle estimation errors and selectively process data from static points to mitigate the influence of dynamic objects. However, it still faces challenges in highly dynamic environments and cannot work in sub-doppler resolutions. [15] employs a three-step process: extracting landmarks, associating points through scan matching in two rounds, and determining the rigid body transformation that best aligns the PCDs. The method incorporates advanced noise and multipath rejection techniques, leveraging the shape consistency of PCDs across frames. While effective in various scenarios, this approach requires careful handling of high-frequency noise and multipath reflections and may face limitations in environments with cluttered objects or significant measurement errors.

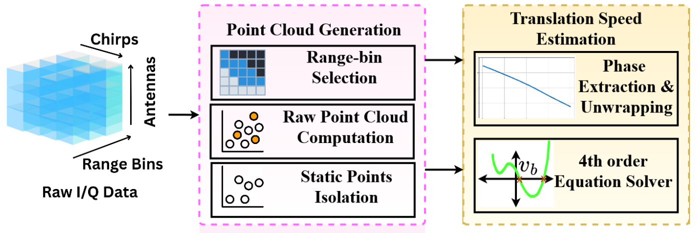

Chirps
Antennas
Range Bins
Raw I/Q Data
Point Cloud Generation
Range-bin Selection
Raw Point Cloud Computation
Static Points Isolation
Translation Speed Estimation
Phase Extraction & Unwrapping
v_b
4th order Equation Solver

Fig. 2: System pipeline of *radarTrack*.

Also, works like [25], [26] deal with the inherent sparsity of the radar compared to optical sensors like LiDAR. [25] employs GANs to densify sparse radar PCDs, while RadarHD [26] uses an asymmetric UNet for radar heatmap upsampling. To the best of our knowledge, *radarTrack* is the only dynamic, radar-only approach that utilizes PCD and phase-based features. In contrast, other SOTA methods rely on IMU, Doppler, or partial radar features (see TABLE I).

III. System Design

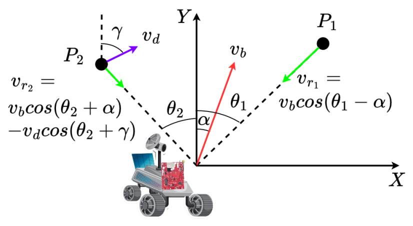
Before we dive into the details of *radarTrack*, let's outline the objectives and decisions that guided our design choices.

A. *Design Choices*

Traditional radar applications often depend on large, costly scanning radars that rotate and scan their environment using mechanical motors. We exclude these radars from our study as they are unsuitable for lightweight, compact robots. Instead, we focus on single-chip COTS radars. Unlike previous approaches that rely on additional sensors such as wheel encoders, IMUs, or LiDARs [11]–[14], [25], our method leverages the intrinsic properties of radar for odometry. This choice is justified as IMU-based speed estimation leads to integration error accumulation, and radar's inherent capabilities make IMUs redundant for 2D SLAM. Most existing ego-motion estimation methods are designed for environments with static objects within the FoV [12], [16]. However, in real-world scenarios, dynamic objects are commonly present. Our design addresses this challenge by incorporating dynamic object handling into our ego-speed estimation framework. Most previous works rely on mmWave radar data combined with DNNs to estimate translational speed [11], [16], [27]. However, these methods can be computationally intensive and challenging to implement on edge devices for real-time computing. Our approach demonstrates how signal processing and mathematical approaches can be effectively used to achieve precise ego-speed estimation.

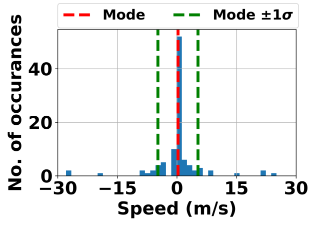
B. *Design Outline*

As shown in Fig. 2, our method is designed with two main components that work together to estimate the speed of an ego-vehicle using radar data. Each component addresses a specific aspect of motion estimation, leveraging different properties of radar sensing. The components are as follows:

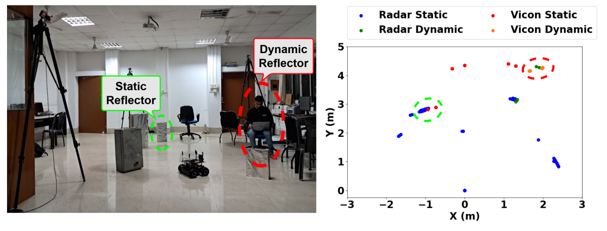


Y
X
P_1
P_2
γ
v_d
v_b
$v_{r_2} =$
$v_b cos(\theta_2 + \alpha)$
$-v_d cos(\theta_2 + \gamma)$
$v_{r_1} =$
$v_b cos(\theta_1 - \alpha)$
θ_2
θ_1
α

(a)



(b)



(c)

Fig. 3: (a) For all static points, the radial component (green arrow) of the relative speed which varies sinusoidally, (b) Distribution of ego-vehicle speed estimates, (c) A typical dynamic scene (left) and classification output (right).

Differentiating between static and dynamic objects: The initial step involves distinguishing between static and dynamic objects within the FoV by analyzing the radial speed profiles of individual objects. When a radar sensor moves, all stationary targets appear to move in the opposite direction from the sensor's perspective. To effectively separate static from dynamic objects, we employ distribution-based thresholding on the scaled radial velocities of all points in the radar PCD. Since static points have a similar distribution in the radial speed profiles relative to the radar thus, it effectively segregates the static objects from dynamic ones.

Translational Speed Computation: We use a novel phase-based approach to compute the translational speed of the ego-vehicle. Unlike previous methods that rely on doppler shift [16], [27], which may not be effective at low speeds (sub-doppler resolution) and can be erroneous due to PCD estimation, our approach utilizes the subtle phase changes in reflectors within the environment. These phase variations provide precise relative movement information of the vehicle, allowing accurate speed estimation in slow-speed applications.

C. Segregating Static Objects from Dynamic Objects

We differentiate between static and dynamic objects in the radar's FoV by analyzing the relative speed profiles of individual points detected by the radar. For a stationary radar, the doppler shift directly gives the radial speed of moving objects. But when the radar is in motion, all targets within its FoV appear to move relative to the radar. For stationary objects, the doppler shifts follow a cosine relation with the angle between the radar and the object. However, for dynamic objects, the radial speed has components from both the radar velocity and the object's own velocity and, therefore, does not follow the direct cosine relation. We exploit this property to segregate static and dynamic objects.

The radar return for a frame can be expressed as $\mathcal{P} = \{\mathbf{p}_i = (r_i, \theta_i, \phi_i, v_{ri})\}_{i=1}^{N}$, where the i^{th} point in the radar PCD is characterized by its range r_i, azimuth angle θ_i, and elevation angle ϕ_i, in the radar's frame of reference, and v_{ri} is the radial speed of the point, relative to the radar. Consider Fig. 3(a), where a radar-mounted ego-vehicle is moving in an environment with a static (P_1) and a dynamic (P_2) object in the FoV of the radar. The instantaneous velocity of the ego-vehicle is v_b, making an angle α with the radar pointing direction. The dynamic object, P_2, moves with a velocity v_d, making an angle γ with the radar pointing direction. The radial speeds of the static and dynamic points can be given by $v_{r1} = v_b cos(\theta_1 - \alpha)$ and $v_{r2} = v_b cos(\theta_2 + \alpha) - v_d cos(\theta_2 + \gamma)$, respectively. Notably, all static points will have similar radial velocity profiles $v_{ri} = v_b cos(\theta_i - \alpha)$. However, dynamic points will have different velocity profiles due to the contribution from their velocity component, $v_{di} cos(\theta_i + \gamma_i)$. We define the ego-vehicle speed estimate from the i^{th} point, $\hat{v}_{bi}$ as $\frac{v_{ri}}{cos(\theta_i - \alpha)}$. We observe that for static objects, $\hat{v}_{bi}$ will be theoretically equal to ego-vehicle speed, v_b, whereas dynamic objects will not follow this relationship. In practice, $\hat{v}_{bi}$'s do not exactly equal v_b due to the radar's noise and doppler and angular resolutions. The static points are identified as,

$$S = \{\mathbf{p}_i \in \mathcal{P} \mid \hat{v}_{bi} \in (M_0 - k\sigma, M_0 + k\sigma)\} \tag{1}$$

where $M_0 = \text{mode}_i(\hat{v}_{bi})$, σ is the standard deviation, and k is a multiplying factor which signifies the spread of the static points. Higher errors in doppler and angular estimation lead to higher values of k. We have chosen k as 1. Figure 3(b) shows the distribution of ego-velocity estimates for a frame. Estimates corresponding to dynamic points lie away from the peak corresponding to static points. We assume static points outnumber dynamic points, which is fair since indoor scenarios contain static reflectors like walls, windows, and furniture with high radar cross-sections.

To find α, consider two static points $\mathbf{p}_i$ and $\mathbf{p}_j$. The ratio of their radial velocities can be expressed as $\frac{v_{ri}}{v_{rj}} = \frac{cos(\theta_i - \alpha)}{cos(\theta_j - \alpha)}$. We obtain an estimate of α as,

$$\hat{\alpha}_{i,j} = \frac{v_{rj} cos(\theta_i) - v_{ri} cos(\theta_j)}{v_{ri} sin(\theta_j) - v_{rj} sin(\theta_i)} \tag{2}$$

With the assumption that static points are in the majority, we can estimate α as $\hat{\alpha} = \text{mode}_{\substack{i,j \\ i \neq j}} (\hat{\alpha}_{i,j})$.

D. Estimating Translational Changes with Phase Unwrapping

Once we have the static points, we estimate the translational speed of the ego-vehicle. For that purpose, we first need to precisely track the presence of the objects in the FoV.

1) Range bin selection: The range-FFT on the radar's raw I/Q frames outputs the positions and phase variations of the objects present in the FoV of the radar. Initially, we identify the peak index in the range-FFT for a given frame and chirp, which corresponds to the object's distance from the radar

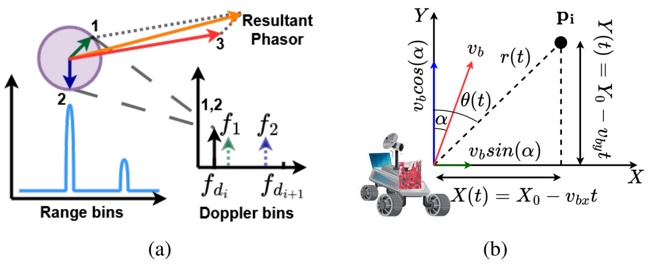


Fig. 4: (a) Speeds falling between two doppler bins are assigned to the lower bin due to FFT; resultant of two phasors is shown, (b) A typical scenario where a static object is present in front of a moving radar.

based on the 95th percentile of the signal-to-noise (SNR) value. As identified in the previous stage, we select those peak indices closer (within $\pm$ 3 range bins) or exactly at the range where static objects are present. Once we select the range bins corresponding to the static objects, our next objective is to analyze the phase variation in those bins over time.

2) Phase Unwrapping of Raw Phase Values: The raw phase values obtained from the radar at a particular range bin where a static object is present may experience discontinuities due to the periodic nature of the phase representation. To ensure accurate phase measurements for speed estimation, we perform phase unwrapping. This process involves identifying and correcting phase jumps greater than π radians, providing a continuous phase representation over time. Specifically, if a phase difference between consecutive measurements exceeds π, a correction is applied by subtracting multiples of 2π to eliminate the discontinuity. The unwrapped phase can be expressed as: $\Phi_{\text{unwrapped}}(t) = \Phi(t) + 2\pi n$, where n is an integer that adjusts the phase to remove discontinuities. Due to phase unwrapping, we can only estimate velocities without ambiguity when the consecutive change in the phase values, $\Delta\Phi$, is less than 2π. Thus, unwrapping the phase values leads to a maximum measurable speed of up to 7.75 m/s.

3) Computing Translational Speed: As shown in Fig.3(a), the previous works on ego-speed estimation use the doppler shifts from the radar PCD and compute the ego-speed as $v_b = \frac{{v_r}^i}{cos\theta^i}$, where ${v_r}^i$ and θ^i are the radial speed and angle-of-arrival of i^{th} PCD respectively. As PCD estimation is sparse due to poor angular resolution ($\approx$ 15°), it can easily corrupt the final translational speed computation. On the other hand, the doppler shift ${v_r}^i$ is computed by doppler-FFT on the number of chirp axis of the radar raw I/Q data, so if the phase change in the chirp axis is lower than the doppler resolution then FFT-based doppler speed computation won't detect the speed accurately. Thus, both these approaches together contribute to an erroneous ego-speed estimation.

Our idea for ego-speed estimation is derived from the basic physics of the phase values. As shown in Fig. 4(b), at time t within a frame $(0 < t < T)$, we have the static object at a distance of $r(t)$, given by $r(t)^2 = X(t)^2 + Y(t)^2$. As seen previously, the ego-speed vector has magnitude v_b and makes an angle α with the radar pointing direction. On differentiating with respect to time (t) and assuming ego-speed, v_b remains constant during the frame time, T_f (which is 150 ms in our case) we have,

$$\begin{aligned} r(t)\frac{dr(t)}{dt} &= (X_0 - v_b cos(\alpha)t)(-v_b cos(\alpha)) \\ &\quad +(Y_0 - v_b sin(\alpha)t)(-v_b sin(\alpha)) \end{aligned} \tag{3}$$

On integrating Eq. (3) with respect to time from $t = 0$ to t and with respect to $r(t)$ from r_0 to r_t we have,

$$r_t = \sqrt{r_0^2 - 2v_b t K(\alpha) - v_b^2 t^2} \tag{4}$$

where $K(\alpha) = Y_0 cos(\alpha) + X_0 sin(\alpha)$. When the ego-vehicle is in motion, the distance $r(t)$ changes with time, leading to a corresponding phase change $\Phi(t) = \Phi_0 + \frac{4\pi r}{\lambda}$. Therefore, the rate of change of r_t at multiple chirps coming within a single frame can be expressed as $\frac{dr_t}{dt} = c\frac{d\Phi(t)}{dt}$, where c is a constant. After differentiating both sides of Eq. (4) with respect to t, squaring and simplifying, we have,

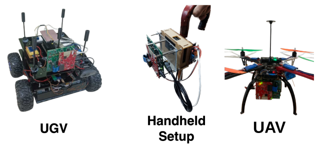
$$a(t)\, v_b^4 + b(t,\alpha)\, v_b^3 + c(t,\alpha)\, v_b^2 + d(t,\alpha)\, v_b + e(t) = 0 \tag{5}$$

where $a(.), b(.), c(.), d(.)$ and $e(.)$ are coefficients given by,

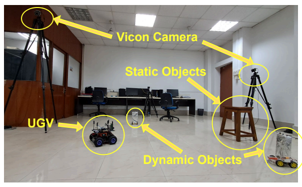
$$\begin{aligned} &a(t) = t^2, \qquad\qquad\qquad b(t,\alpha) = -2K(\alpha)t \\ &c(t,\alpha) = K(\alpha)^2 - t^2c^2\Theta^2(t) \\ &d(t,\alpha) = 2tK(\alpha)c^2\Theta^2(t), \quad e(t) = -r_0^2c^2\Theta^2(t) \end{aligned}$$

where, $\Theta(t) = \frac{d\Phi}{dt}(t)$. Eq. (5) represents a 4th-order equation in ego-speed, v_b, with four possible roots. For each static point, j from 1 to N, we directly obtain $\Phi_{i,j}$ at a time instant t_i, where i goes from 1 to N_c (number of transmitted chirps). For each static point in a frame, we have four roots of the equation for each chirp. Thus, we have $N \times N_c \times 4$ such roots in a single frame. One root must be common across all the chirps, as v_b is a physical quantity. We compute all the roots for a single frame and take the median to find the common root, which occurs most frequently. This approach helps compute the translational speed even below doppler resolution as it depends purely on the phase changes of the selected static objects.

Although this approach is effective when a static object is the only one in its range bin. When multiple objects share the same range bin, their phasor representation forms a resultant phasor that deviates from Eq. (4), imposing a restrictive condition on selecting static points. This constraint can be relaxed by noting that the resultant of two phasors, with amplitudes $A_1 >> A_2$, closely follows the dominant phasor, as shown in Fig.4(a) by phasors 1 and 3. For example, let's say we have n objects in a single range bin, whose phasors can be computed with doppler-FFT having amplitudes $A_1, A_2, ..., A_n$. We consider this range bin for further computation only if a single phasor is dominant over the other phasors $(A_1 >> A_2 >> ... >> A_n)$. Thus overall, the advantage of our method over naive doppler-based processing is demonstrated using Fig.4(a). For an object located in the



(a)



(b)

Fig. 5: (a) Different ego-vehicle settings, (b) deployment setup of *radarTrack*.

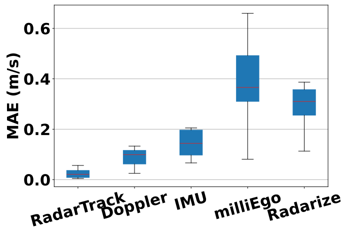


Fig. 6: MAE with *radarTrack* compared to baselines.

indicated range bin, phasor 1 (green) and 2 (blue) show two possible radial speeds (f_1 and f_2), both of which lie between the i and $(i+1)^{th}$ doppler bins. Doppler-FFT places both speeds in the i^{th} bin, whereas our method can accurately find and distinguish them.

IV. Implementation

We have implemented *radarTrack* over three different setups, as discussed next.

A. Hardware setup

Platforms: We use a Texas Instruments IWR1843 radar [28], and a DCA1000 data capture card to receive raw I/Q samples. We evaluate our method on three platforms: (i) *UGV ego-vehicle*: We mount our radar on a remotely operated ground robot, which navigates along various trajectories. (ii) *UAV*: We utilize a low-flying drone with a *Pixhawk 2.4.8 flight controller* to collect radar data under different speeds. (iii) *Handheld*: For the handheld setup, we mount the radar on a 95 cm long handheld stick carried around by volunteers in predefined trajectories. The hardware setups are shown in Fig. 5(a). In each case, for computing, we use a Jetson Nano running Ubuntu 18.04, featuring a quad-core CPU and 4 GB of RAM. We also use a 6-axis IMU, *MPU6050* for logging the acceleration and gyroscope values.

Radar Configuration: The radar works in a frequency range of 77-81 GHz. We configure the radar to a range resolution of 0.0429 meters, and the radial speed resolution is set to 0.0496 m/s. The system operates at 10 frames per second, with 182 chirps per frame and 256 ADC samples per chirp.

B. Ground Truth Speed Estimation

To establish the accuracy of our speed measurements, we utilized a Vicon Vero tracker (v1.3X) motion capture system, which provides high-precision, real-time tracking of the position and orientation of the ego-vehicle. By comparing the speed estimates from the radar with the ground truth measurements from the Vicon system, we could evaluate the accuracy and reliability of our speed estimation algorithm.

C. Baselines and Evaluation Metrics

We choose the following baselines for comparison of our method: (1) Radarize [16], which uses a ResNet18-based DNN architecture for translational speed estimation taking only radar doppler-azimuth heatmaps as its input, (ii) Doppler based approach [9], which computes doppler-FFT on the raw I/Q data and estimates the speed from the peak doppler bins, (ii) IMU-based odometry [29, Sec. 6.1.1] and (iii) MilliEgo [11] model which fuses both mmWave range-angle heatmaps along with IMU. It predicts the position coordinates of the ego-vehicle, and we modify it to predict the ego-speed instead. We use the Vicon system to capture the trajectory of our ego-vehicle. From this recorded trajectory, we calculate the ground truth speed of the ego-vehicle. We have considered mean absolute error as the primary evaluation metric.

V. Evaluation

This section provides a thorough evaluation under different combinations of reflectors and ego-vehicle setups.

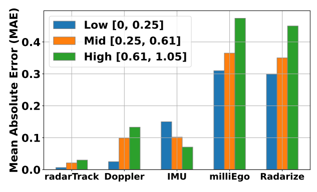
A. Overall Performance

Fig. 6 shows the distribution of MAE across different methods. As can be seen, the median of MAE with *radarTrack* is ≈ 2 cm/s. Fig. 7(a) presents the MAE for various methods at different speed levels (Low, Mid, High). The speed ranges are defined as follows: **Low** $<$ 0.25 m/s, 0.25 m/s $<$ **Mid** $<$ 0.61 m/s, and 0.61 m/s **<** **High** $<$ 1.05 m/s. Our method consistently achieves the lowest error (approx 5% of the base speed, on average), outperforming the closest competitor, the doppler-based approach, by up to 4x in accuracy.

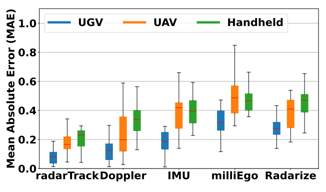
As the vehicle's speed increases, we notice that the error grows, likely due to the increasing noise in the phase component of the reflected signals. The doppler-FFT technique also tends to worsen speed estimation at higher speeds as the inherent phase errors can accumulate more errors in the doppler-FFT. Also, at low speeds, the doppler-based method struggles to capture small-scale movements that fall below the doppler resolution, leading to higher errors when compared to our approach. Interestingly, the IMU-based odometry performs better at high speeds compared to low speeds. This discrepancy is likely caused by time drift in the IMU data during lower-speed scenarios, where longer collection durations introduce drift errors. milliEgo [11] performs the worst, suffering from limitations in both IMU and mmWave modalities. On the other hand, radarize [16] also performs poorly during dynamic scenarios when multiple objects are moving in the radar's FoV.

B. Performance with Different Ego-Vehicle Settings

We evaluated the performance of *radarTrack* on three different ego-vehicle platforms: UGV, handheld device, and UAV, as demonstrated in Fig. 5(a). Fig. 7(b) presents the results for

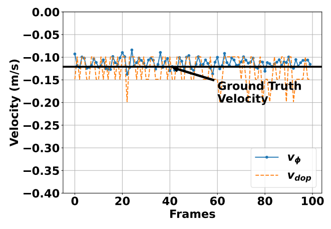


(a)

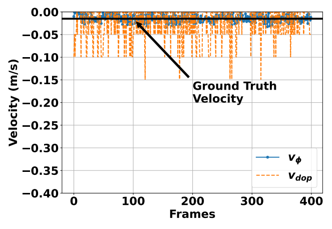


(b)

Fig. 7: MAE across (a) different speeds, (b) different ego-vehicle platforms.



(a)



(b)

Fig. 8: Comparative study of the doppler-based vs phase-based speed computation: (a) Ego-vehicle moving at medium speed (12 cm/s) and (b) lower speed (1.5 cm/s) (Note: $^{-}$ve speed represents the ego-vehicle is approaching a target).

radarTrack across the different ego-vehicle platforms, comparing its performance against other baselines. For the UGV platform with a baseline speed of 0.25 m/s, *radarTrack* achieves a MAE of approximately 0.02 m/s, performing consistently well due to the minimal noise caused by translational movement on the ground. In contrast, the UAV platform (baseline speed of 0.21 m/s) presents more challenges, as vibrational noise and translational motion in the vertical axis introduce significant noise into the speed estimates. Consequently, the MAE for the UAV setup is around 0.15 m/s, which is notably higher due to the difficulty in compensating for the UAV's variable altitude and dynamic movement. Lastly, the handheld setup (baseline speed of 0.35 m/s) exhibits an intermediate level of noise, primarily due to human subjects repeatedly turning the stick left and right, generating noise in the speed estimates, leading to an MAE of approximately 0.16 m/s.

C. Performance Improvement with Phase-Based Approach at Different Speeds

State-of-the-art methods for ego-speed estimation, such as Radarize [16] and MilliEgo [11], primarily rely on range-doppler processing, which involves applying an FFT to the range data across multiple chirps. However, this approach results in a fixed doppler resolution, meaning speeds that fall between the resolution steps cannot be accurately estimated.

In contrast, using raw phase data enables tracking much finer movements due to the relationship $\phi = \frac{4\pi d}{\lambda}$, where λ is the wavelength, typically on the order of millimeters (≈ 4 mm). So by leveraging the phase variations, for example with a phase change of just $0.057°$, minute changes in range (as small as $\Delta r \approx 0.63 \mu m$) can be detected [30, Sec. II-B].

In Fig. 8, we present a qualitative comparison between speed estimates from the phase-based method and those obtained from the range-doppler heatmap, a conventional state-of-the-art approach. The speed estimates from the range-doppler heatmap are notably noisier at lower speeds (e.g., 1.5 cm/s) because the true speed lies within the doppler resolution limit (4 cm/s). Even at higher speeds, the phase-based method delivers more accurate speed estimates. This is because, at high speeds, the range-doppler FFT can introduce errors as the range bin also shifts within a single frame, leading to incorrect phase values when tracked at the frame level. In contrast, *radarTrack* operates at a chirp-level resolution for phase-based speed estimation, enabling more accurate and robust performance across a range of speeds.

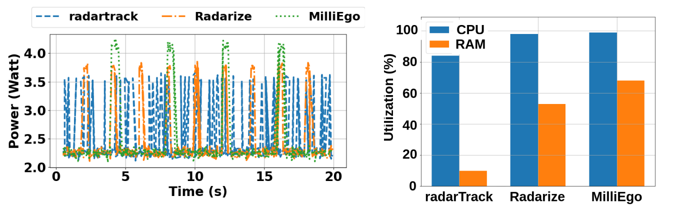
D. Performance with different combinations of static and dynamic reflectors

We evaluate the performance of *radarTrack* under various combinations of static and dynamic reflectors (see Fig. 5(b)): (i) 3 static objects, (ii) 2 static and 1 dynamic object, (iii) 2 static and 2 dynamic objects, (iv) 1 static and 1 dynamic object, and (v) 2 dynamic objects.

TABLE II: MAE for Different Combinations of Static and Dynamic Reflectors

Configuration	MAE (m/s)
3 static objects	**0.018**
2 static objects, 1 dynamic object	0.025
2 static objects, 2 dynamic objects	0.037
1 static object, 1 dynamic object	0.045
2 dynamic objects	0.054

The static objects in our setup include typical household items like chairs and metal objects, representing surfaces with different reflective properties. For dynamic objects, we utilize a small robot car with an aluminum foil-coated box (to emulate high reflectivity to generate maximal interference with our method) and human subjects to capture different motion and reflective behaviors. The accuracy of *radarTrack* tends to decrease as the number of high-reflective dynamic objects increases. This is primarily due to the additional phase noises introduced by the movement of dynamic objects, which complicates the inference process. However, despite the increasing complexity, *radarTrack* consistently maintained a reasonable level of performance, effectively handling the dynamic environments. The results (see Table II) highlight that introducing dynamic objects leads to some degradation in accuracy, but the overall system remains robust. Notably, with three static objects, the system achieves its highest accuracy since the reflections from the static objects remain stable. Conversely, with two dynamic objects, performance dips slightly due to the increased unpredictability and motion interference in the radar FoV. Notably, the ego-vehicle does not observe any static objects initially in this case, thus having a high estimation error initially (approx 0.2 m/s); however, as it moves towards the wall, it considers the wall as a static object and reduces the error significantly (approx 0.01 m/s).



(a) (b)

Fig. 9: Power & computing resource consumption at the edge.

E. Resource and Power Consumption

We analyze the power consumption and resource utilization of *radarTrack* in comparison to Radarize and MilliEgo, evaluated on a Jetson Nano equipped with 4 GB RAM and a quad-core CPU. As shown in Fig. 9(a), *radarTrack* demonstrates a significantly lower and more stable power consumption profile. The baseline power for *radarTrack* remains around 2.25 W, with occasional spikes reaching 2.56 W during inference. In contrast, the DNN-based methods, Radarize and MilliEgo, exhibit higher peaks, reaching up to 2.75 W and 3.12 W, respectively. The continuous spikes in power for these DNN-based methods reflect their higher computational load and energy demands, making them less suitable for lightweight, low-power applications. As shown in Fig. 9(b), *radarTrack* utilizes only 84% of CPU and 10% of RAM, significantly lower than the DNN-based approaches consuming 99% of CPU usage, with Radarize using 53% and MilliEgo consuming up to 68% of the available 4 GB memory. The latency of *radarTrack* is approximately 0.29 s, significantly lower than that of Radarize (3 s) and MilliEgo (5 s).

VI. Conclusion

In conclusion, we propose *radarTrack*, a novel odometry framework that leverages a single-chip mmWave radar for robust ego-motion estimation in challenging environments. Unlike existing methods, *radarTrack* relies solely on mmWave radar data, overcoming limitations such as noise, sparsity, and slow speed operation. Our approach, based on phase-based speed estimation, offers low latency and high reliability, making it suitable for mobile, wearable, and resource-constrained devices. The proposed approach can be extended for real-time navigation support for micro-robots, autonomous maneuvers, and assistive movements through augmented reality, which we keep as the future extension of this work.

References

[1] S. Shen, M. Gowda, and R. Roy Choudhury, "Closing the gaps in inertial motion tracking," in *ACM MobiCom*, 2018, pp. 429–444.

[2] W. Lee, K. Eckenhoff, Y. Yang, P. Geneva, and G. Huang, "Visual-inertial-wheel odometry with online calibration," in *IEEE/RSJ IROS*, 2020, pp. 4559–4566.

[3] Y. Yang, P. Geneva, X. Zuo, and G. Huang, "Online imu intrinsic calibration: Is it necessary?" *2020 Robotics: Science and Systems*, 2020.

[4] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem," in *AAAI*, vol. 31, no. 1, 2017.

[5] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE T-RO*, vol. 34, no. 4, pp. 1004–1020, 2018.

[6] J. Lien, N. Gillian, M. E. Karagozler, P. Amihood, C. Schwesig, E. Olson, H. Raja, and I. Poupyrev, "Soli: Ubiquitous gesture sensing with millimeter wave radar," *ACM TOG*, vol. 35, no. 4, pp. 1–19, 2016.

[7] H. Yu, F. Zhang, P. Huang, C. Wang, and L. Yuanhao, "Autonomous obstacle avoidance for uav based on fusion of radar and monocular camera," in *IEEE/RSJ IROS*, 2020, pp. 5954–5961.

[8] K. Deng, D. Zhao, Q. Han, S. Wang, Z. Zhang, A. Zhou, and H. Ma, "Geryon: Edge assisted real-time and robust object detection on drones via mmwave radar and camera fusion," *ACM IMWUT*, vol. 6, no. 3, pp. 1–27, 2022.

[9] S. Rao, "Introduction to mmwave sensing: Fmcw radars," *Texas Instruments (TI) mmWave Training Series*, pp. 1–11, 2017.

[10] Y. Lin and H. Caesar, "Icp-flow: Lidar scene flow estimation with ICP," in *IEEE/CVF CVPR*, 2024, pp. 15 501–15 511.

[11] C. X. Lu, M. R. U. Saputra, P. Zhao, Y. Almalioglu, P. P. De Gusmao, C. Chen, K. Sun, N. Trigoni, and A. Markham, "milliego: single-chip mmwave radar aided egomotion estimation via deep sensor fusion," in *ACM SenSys*, 2020, pp. 109–122.

[12] A. Kramer, C. Stahoviak, A. Santamaria-Navarro, A.-A. Agha-Mohammadi, and C. Heckman, "Radar-inertial ego-velocity estimation for visually degraded environments," in *IEEE ICRA*, 2020, pp. 5739–5746.

[13] Y. Almalioglu, M. Turan, C. X. Lu, N. Trigoni, and A. Markham, "Milli-rio: Ego-motion estimation with low-cost millimetre-wave radar," *IEEE Sensors Journal*, vol. 21, no. 3, pp. 3314–3323, 2020.

[14] Q. Huang, Y. Liang, Z. Qiao, S. Shen, and H. Yin, "Less is more: Physical-enhanced radar-inertial odometry," in *IEEE ICRA*, 2024.

[15] S. H. Cen and P. Newman, "Precise ego-motion estimation with millimeter-wave radar under diverse and challenging conditions," in *IEEE ICRA*, 2018, pp. 6045–6052.

[16] E. Sie, X. Wu, H. Guo, and D. Vasisht, "Radarize: Enhancing radar slam with generalizable doppler-based odometry," in *ACM MobiSys*, 2024, pp. 331–344.

[17] A. Sen, S. Chakraborty, S. Tripathy, and S. Chakraborty, "Poster: Dynamic ego-velocity estimation using moving mmwave radar: A phase-based approach," in *ACM MobiSys Posters*, 2024, pp. 692–693.

[18] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *IEEE ICRA*, 2017, pp. 2043–2050.

[19] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, "Demon: Depth and motion network for learning monocular stereo," in *IEEE CVPR*, 2017, pp. 5038–5047.

[20] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," in *ECCV*, 2018, pp. 817–833.

[21] C. Chen, X. Lu, A. Markham, and N. Trigoni, "Ionet: Learning to cure the curse of drift in inertial odometry," in *AAAI*, vol. 32, no. 1, 2018.

[22] C. Chen, C. X. Lu, J. Wahlström, A. Markham, and N. Trigoni, "Deep neural network based inertial odometry using low-cost inertial measurement units," *IEEE TMC*, vol. 20, no. 4, pp. 1351–1364, 2019.

[23] R. Ding, H. Jin, J. Ding, X. Wang, G. Fan, F. Zhu, X. Tian, and L. Kong, "Push the limit of single-chip mmwave radar-based egomotion estimation with moving objects in fov," in *ACM SenSys*, 2023, pp. 417–430.

[24] Y. Zhuang, B. Wang, J. Huai, and M. Li, "4d iriom: 4d imaging radar inertial odometry and mapping," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3246–3253, 2023.

[25] C. X. Lu, S. Rosa, P. Zhao, B. Wang, C. Chen, J. A. Stankovic, N. Trigoni, and A. Markham, "See through smoke: robust indoor mapping with low-cost mmwave radar," in *ACM MobiSys*, 2020, pp. 14–27.

[26] A. Prabhakara, T. Jin, A. Das, G. Bhatt, L. Kumari, E. Soltanaghai, J. Bilmes, S. Kumar, and A. Rowe, "High resolution point clouds from mmwave radar," in *IEEE ICRA*, 2023, pp. 4135–4142.

[27] E. Sie, Z. Liu, and D. Vasisht, "Batmobility: Towards flying without seeing for autonomous drones," in *ACM MobiCom*, 2023, pp. 1–16.

[28] "Iwr1843boost," https://www.ti.com/tool/IWR1843BOOST, Aug. 2019, accessed March 17, 2025.

[29] T. Lei, A. A. Mohamed, and C. Claudel, "An imu-based traffic and road condition monitoring system," *HardwareX*, vol. 4, p. e00045, 2018.

[30] S. Basak and M. Gowda, "mmspy: Spying phone calls using mmwave radars," in *IEEE SP*, 2022, pp. 1211–1228.