Beyond Doppler: Demonstrating Phase-Based Ego-Speed Estimation on Embedded 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 demonstration we introduce a novel approach to ego-speed estimation using a single-chip Commercial-Off-the-Shelf (COTS) millimeter-wave (mmWave) radar. Contrary to previous approaches that are cross-modal learning based and dependent on the computationally expensive nature of DNN's, our proposed approach *RadarTrack* is based on a phase-based approach to speed estimation. Our approach successfully overcomes the drawbacks of traditional ego-speed estimation methods that are doppler-based and static environment dependent. *RadarTrack* is intended to support low-latency execution on embedded systems such that it can be used in real-time applications where efficiency and speed are equally important. We have also created a real-time visualizer which is capable of recording the phase-based ego-speed together with state-of-the-art doppler-based speed estimation and comparatively demonstrate how phase-based ego-speed estimation can better record the speed of an ego-vehicle.**

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

I. Introduction

Accurate motion estimation at slow speeds remains challenging, particularly because conventional sensors such as cameras, IMUs, and LiDARs often lose effectiveness in these conditions. When movements are subtle and visual features are sparse, radar technology offers a significant advantage [1]–[5]. This demonstration primarily focuses on applications that demand precise translational speed estimation in slow-speed scenarios, such as indoor navigation, close-proximity maneuvers, and micro-robotics.

IMU based odometry are widely used for ego-motion estimation across various mobile platforms. However, estimating position from acceleration requires double integration, which accumulates errors over time [6]. In ground vehicles moving at constant speeds, accelerometers struggle to detect subtle changes in acceleration [7], [8]. To overcome these limitations, multimodal odometry systems have been developed, integrating inertial data with complementary sensor inputs, such as visual or ranging information. Among these, visual-inertial odometry (VIO) stands out for its robustness and is widely deployed in devices like smartphones [7], [9], [10]. Nevertheless, VIO can degrade significantly under difficult lighting conditions, such as complete darkness or intense glare, where cameras underperform.

mmWave sensing offers distinct advantages over vision-based systems, particularly its resilience to environmental factors like lighting variations and airborne obscurants. Unlike mechanically scanning radars, single-chip mmWave radars utilize electronic beamforming, making them lightweight and ideal for mobile devices, micro-robots, and wearable technology. However traditional radar-based approaches estimate speed using doppler shifts, where the Discrete Fourier Transform (DFT) constrains measurements to discrete velocity steps. Speeds that do not align with these discrete steps incur estimation errors, especially at very low speeds, placing them in the sub-doppler regime, as encountered in micro-robotics.

Despite its promise, building a reliable indoor odometry system using radar presents several challenges. Radar signals are prone to noise from specular reflections, diffraction, and strong multipath effects. Furthermore, limited hardware antenna counts result in highly sparse point clouds (PCDs) with low angular resolution. This sparse, noisy data makes traditional LiDAR-based techniques like Iterative Closest Point (ICP) [11] largely ineffective when applied directly to mmWave radar data. Additionally, while previous studies [2], [4], [5], [12], [13] have utilized multimodal approaches by combining radar with IMUs or RGB cameras, the true potential of mmWave radar to independently complement these modalities remains unclear. Most existing systems perform well in static environments [1], [4], [12] but struggle when dynamic objects are introduced. Moreover, deep learning-based advances in visual and LiDAR odometry are often computationally intensive, posing a challenge for mobile, wearable, and other resource-constrained platforms.

To implement *RadarTrack*, we first differentiate between static and dynamic objects within the radar's field of view (FoV). This is achieved by analyzing the distribution of radial speed profiles: as the radar moves, stationary objects exhibit uniform motion relative to the sensor, allowing us to apply a distribution-based thresholding method to effectively isolate static points from dynamic ones. Following this, we compute the ego-vehicle's translational speed using a phase-based technique. Instead of relying on doppler shifts [1], [14]—which become unreliable at low speeds and are prone to errors from noisy point cloud (PCD) estimations—our approach leverages subtle phase variations from environmental reflectors. These

variations offer precise measurements of relative movement, enabling robust and accurate speed estimation even in low-speed, sub-doppler conditions.

II. Proposed Framework overview

A. Static-Dynamic Object Classification

We differentiate static and dynamic objects in the radar's field of view (FoV) by analyzing their relative speed profiles. For a stationary radar, the Doppler shift directly provides the radial speed of moving objects. However, when the radar is mounted on a moving platform, all objects appear to have motion relative to it. Stationary objects follow a predictable cosine relationship between their Doppler shift and their angular position, while dynamic objects exhibit deviations from this pattern due to their own motion. We exploit this distinction: by estimating the ego-vehicle speed from each radar point based on its measured radial velocity and comparing it to the expected value, points corresponding to static objects can be identified. Static points cluster around a common ego-velocity estimate, while dynamic points deviate from it. We assume that static points outnumber dynamic ones, which is valid for indoor environments rich in stationary structures like walls.

B. Translational Speed Estimation

Algorithm 1 Range Peak Identification

```
Input: Raw I/Q frames, range values of static objects
Output: Peak indices of static objects
for each frame do
    Perform range-FFT on the frame
    Identify peak indices above the 95^th percentile of SNR
    Consider the indices within ±3 of the static range bins
    Store indices of the range peaks
end for
```

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.

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 based on the 95^{th} percentile of the signal-to-noise (SNR) value. As identified in the previous stage, we select those peak indices closer (within ± 3 range bins) or exactly at the range where static objects are present. Algorithm 1 summarizes the overall flow of range peak selection.

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. 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.

We derive an equation that captures the phase change in static objects induced by the ego-vehicle's motion as a fourth-order polynomial in ego-speed, based on fundamental kinematic laws. By solving the roots of this equation, we achieve precise ego-speed estimation, even for static objects located at oblique angles relative to the radar. More details on the derivation are provided in [15]. 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 speed is a physical quantity. We compute all the roots for a single frame and take the mode. Algorithm 2 summarizes the entire implementation of translational speed estimaiton.

Algorithm 2 Translational Speed Calculation

```
Input: Range bin index, unwrapped phase values φ_i for N chirps
Output: Estimated ego-velocity
for each frame do
    for each chirp do
        Retrieve unwrapped phase values from the selected range bin
        Solve 4^th quadratic equation to find four possible roots
    end for
    Store all roots obtained across N chirps
end for
Identify the common root across chirps (majority root)
Compute the median of the common root
return Median root
```

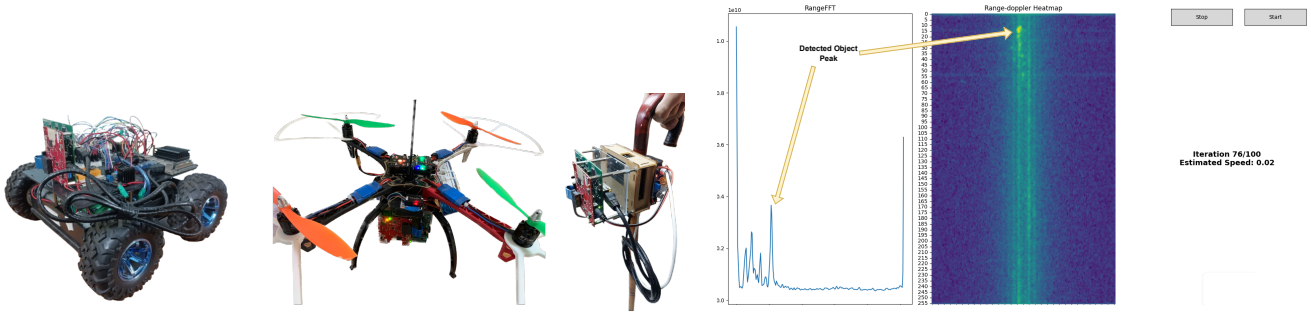
III. Demonstration

This section presents the demonstration of the proposed module and shows how to use it. Primarily we need to have a hardware ego-vehicle setup and the backend software unit as described below:

A. Hardware setup

We can use any of the following ego-vehicle platforms for validating *RadarTrack*: (i) a UGV-based ego-vehicle, (ii) a UAV platform, and (iii) a handheld stick setup, as illustrated in Fig. 1(a). Each platform is equipped with a Jetson Nano (4 GB RAM) serving as the primary processing unit, connected to a DCA1000EVM data capture card paired with an IWR1843BOOST EVM mmWave radar. For demonstration purposes, the radar and data capture setup can be mounted onto any of these platforms to function as the ego-vehicle.

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, providing precise distance measurement. The radial velocity 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, ensuring high temporal resolution. We detailed the configuration parameters in Table I.



(a) (b)

Fig. 1: (a) Radar mounted on different ego-vehicle platforms such as robots, handheld stick and on UAV, (b) Visualization tool to showcase the estimated ego-speed in real-time.

TABLE I: Radar Configuration

Parameters	Value	Parameters	Value
Number of TX Antennas	3	Frame Duration (µs)	72
Number of RX Antennas	4	Idle Time (µs)	7
Frequency Slope (MHz/us)	60.012	Ramp End Time(µs)	65
ADC Samples per Chirp	256	Chirps per Frame	182
Sample Rate (ksps)	4400	Range Resolution (m)	0.041
RX Gain (dB)	30	Max Range (m)	10.99
Chirp Duration (µs)	14	Doppler Resolution (m/s)	0.0496

B. Software Setup

For computation, we employ a Jetson Nano running Ubuntu 18.04, equipped with a quad-core Intel Atom x5-Z8350 CPU and 4 GB of RAM. The Jetson Nano connects to the DCA1000EVM board via a 1 Gb Ethernet link. To configure the mmWave radar attached to the *DCA1000EVM*, we first use Texas Instruments' *mmWave Studio*, a Windows-based tool for radar setup and firmware flashing. After programming the radar, we switch the Ethernet connection from the Windows machine to the Jetson Nano. We have open-sourced our implementation in GitHub [16], which one can clone and generate the ego-speed estimates based on our phase-based approach as well as state-of-the-art doppler based approach (see Fig. 1(b)).

IV. Conclusion

In this demo, we introduced *RadarTrack*, a novel and lightweight phase-based approach for ego-speed estimation using a single-chip COTS mmWave radar. Unlike traditional Doppler-based techniques, *RadarTrack* reliably operates even in low-speed and dynamic environments where sub-doppler effects pose significant challenges. From phase variations in static reflectors, our method achieves robust and accurate speed estimation while remaining computationally efficient and suitable for embedded platforms such as the Jetson Nano.

Acknowledgment

This work has been supported by the Department of Science and Technology (NGP Division) with funding approval number NGP/GS-02/Sandip/IIT-K/WB/2023 (C), dated 09-01-2024. The works of Soham Chakraborty and Soham Tripathy are also supported by CHANAKYA Fellowship Program 2023 from the TIH Foundation for IoT & IoE, IIT Bombay, India.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

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

[9] 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.

[10] 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.

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

[12] A. Kramer, C. Stahoviak, A. Santamaria-Navarro, A.-A. Aghamohammadi, 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] E. Sie, Z. Liu, and D. Vasisht, "Batmobility: Towards flying without seeing for autonomous drones," in *ACM MobiCom*, 2023, pp. 1–16.

[15] A. Sen, S. Chakraborty, S. Tripathy, and S. Chakraborty, "Radar-track: Enhancing ego-vehicle speed estimation with single-chip mmwave radar," *arXiv preprint arXiv:2504.14495*, 2025.

[16] "GitHub - arghasen10/radarTrack_visualizer: RadarTrack: Enhancing Ego-Vehicle Speed Estimation with Single-chip mmWave Radar — github.com," https://github.com/arghasen10/radarTrack_visualizer, [Accessed April 29, 2025].