



# Radarize: Enhancing Radar SLAM with Generalizable Doppler-Based Odometry

Emerson Sie  
University of Illinois Urbana-Champaign

Heyu Guo  
Peking University

Xinyu Wu  
University of Illinois Urbana-Champaign

Deepak Vasisht  
University of Illinois Urbana-Champaign

<https://radarize.github.io>

## ABSTRACT

Millimeter-wave (mmWave) radar is increasingly being considered as an alternative to optical sensors for robotic primitives like simultaneous localization and mapping (SLAM). While mmWave radar overcomes some limitations of optical sensors, such as occlusions, poor lighting conditions, and privacy concerns, it also faces unique challenges, such as missed obstacles due to specular reflections or fake objects due to multipath. To address these challenges, we propose Radarize, a self-contained SLAM pipeline that uses only a commodity single-chip mmWave radar. Our radar-native approach uses techniques such as Doppler shift-based odometry and multipath artifact suppression to improve performance. We evaluate our method on a large dataset of 146 trajectories spanning 4 buildings and mounted on 3 different platforms, totaling approximately 4.7 Km of travel distance. Our results show that our method outperforms state-of-the-art radar and radar-inertial approaches by approximately 5x in terms of odometry and 8x in terms of end-to-end SLAM, as measured by absolute trajectory error (ATE), without the need for additional sensors such as IMUs or wheel encoders.

## CCS CONCEPTS

• **Hardware** → **Signal processing systems**; • **Computer systems organization** → **Sensors and actuators**; **Robotics**; • **Computing methodologies** → **Machine learning**.

## KEYWORDS

Radar, SLAM, Doppler Shift, Wireless Sensing, Machine Learning

## ACM Reference Format:

Emerson Sie, Xinyu Wu, Heyu Guo, and Deepak Vasisht. 2024. Radarize: Enhancing Radar SLAM with Generalizable Doppler-Based Odometry. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3643832.3661871>



This work is licensed under a Creative Commons Attribution International 4.0 License. MOBISYS '24, June 3–7, 2024, Minato-ku, Tokyo, Japan  
© 2024 Copyright is held by the owner/author(s).  
ACM ISBN 979-8-4007-0581-6/24/06.  
<https://doi.org/10.1145/3643832.3661871>

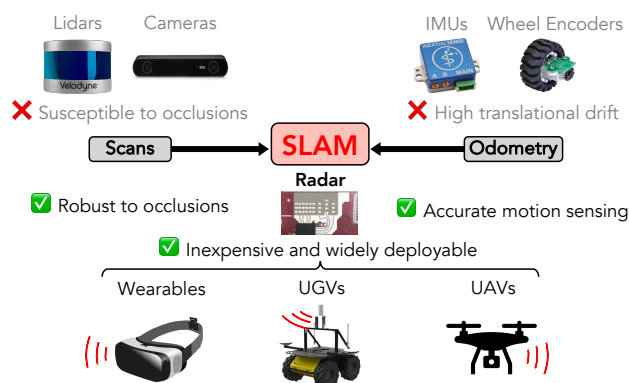


FIGURE 1: Key advantages of radar over conventional SLAM sensors. Radarize leverages both environmental sensing and motion sensing capabilities of radar to perform SLAM.

## 1 INTRODUCTION

Simultaneous localization and mapping (SLAM) is a core requisite for many robotics applications. Using SLAM, a robot can simultaneously construct a map of its surrounding environment and localize itself within this map. Traditional SLAM relies on optical sensors (i.e. cameras and LIDAR). However vision-based SLAM suffers from failures in low lighting conditions and raises privacy concerns during indoor use, while LIDAR-based SLAM is thwarted by common obscurants like fog and smoke.

mmWave radars avoid many of these challenges. They can operate in low lighting and occluded settings, e.g., in agricultural robots or search and rescue robots, mmWave radars can continue to operate despite being occluded by dirt and smoke. For indoor robots such as vacuum cleaning robots, radars can be easily hidden behind a facade to not appear prying to users. Moreover, radar has lower resolution than optical sensors, increasing privacy-friendliness. As a result, there is an increasing interest in developing SLAM techniques using frequency-modulated carrier wave (FMCW) mmWave radar [33, 43].

### 1.1 Challenges for Radar-Based SLAM

Current techniques for mmWave radar-based SLAM are inadequate due to three key challenges:

(i) **Degenerate Scan Matches:** Past work like RadarHD [43] uses scan matching techniques to infer a radar's motion. This technique is inherited from optical SLAM pipelines that compare two frames from a camera or LIDAR to estimate the robot's self-motion. We observe that relying on scan matching can induce failure modes for

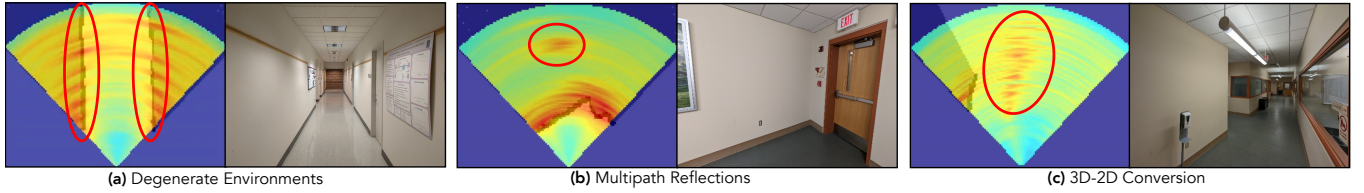


FIGURE 2: **Common sources of artifacts in indoor environments.** Each image of a scene is paired with a top-down radar heatmap superimposed on a depth camera point cloud. (a) When moving down hallways, featureless flat walls present ambiguities during scan matching. (b) Multipath reflections can cause objects to appear behind surfaces. (c) Limited elevation resolution on most sensors induces artifacts from floors/ceilings.

radar-based SLAM similar to those of LIDAR-based SLAM [10, 32]. Specifically, in corridor and hallway-like environments, the lack of distinct geometric features (e.g., due to flat walls) leads to degenerate cases where one frame can be matched with several other frames along the corridor, leading to large errors. To illustrate this, consider Fig. 2(a), which depicts a common hallway environment. We can see that the radar heatmap consists of reflectors along two parallel flat walls (circled in red). If the observer moves down the hallway (i.e. forward) the heatmap after motion would be similar to the heatmap captured just before the motion took place. Hence, during scan matching, the result would be as if no motion occurred.

**(ii) Inadequacy of Inertial Sensors:** A common approach to deal with degenerate environments is to use additional sensors such as accelerometers (e.g., milliMap [33], milliEgo [34]). This approach carries several disadvantages. First, accelerometers suffer from integrative errors because estimating position from acceleration requires double integration. Second, accelerometers fare badly in scenarios when the robot moves at a near constant velocity which is common in the case of ground vehicles [28]. In such scenarios, the acceleration is nearly unobservable due to being dominated by noise [28, 35, 50, 51]. Finally it is well known that, in practice, effectively using multiple sensors in conjunction requires laborious precise calibration [51]. Such factory calibration procedures often necessitate expensive bespoke infrastructure (i.e. synchronized clocks, precision turntables, robotic manipulators, motion tracking systems, fiducial targets [29, 35, 49]), rendering such procedures out of reach for low-cost and self-assembled sensor suites in the field [16, 51]. Furthermore, this calibration may need to be re-performed periodically as the operating conditions of the robot may change over time due to factors such as wear and tear [9, 51].

**(iii) Artifacts due to Multipath:** Radar-based SLAM is prone to two types of artifacts in the generated maps. (a) Multipath effect: radio signal reflections from every day objects cause false images to appear in maps produced by radar-based SLAM approaches. For example in Fig. 2(b), a phantom caused by reflections is seen behind a cul-de-sac. Recent work in radar-based SLAM has not addressed multipath effects [33, 43]. (b) 3D-to-2D conversion artifacts: since SLAM focuses on creating a 2D map of the 3D world, past work largely utilizes azimuthal beamforming techniques on single chip mmWave radars. This approach leads to floor and ceiling reflections being mapped to artificial obstacles in the generated map, as shown in Fig. 2(c). A long ceiling lamp appears in the heatmap even as the radar is pointed horizontally.

## 1.2 Radarize– Radar-Native SLAM

We present Radarize, a novel FMCW radar-based SLAM system that counters these challenges. In designing Radarize, we take a radar-native approach, i.e., we use properties unique to radio signals that have been underexplored in past work. Notably, Radarize is self-contained — it performs SLAM using only a simple off-the-shelf single-chip radar [20] and does not rely on additional sensors or custom hardware. Radarize is built on the following technical contributions.

**Doppler-based Translation Estimation:** Scan-matching algorithms and inertial sensors ignore a key inherent advantage of radio signals – doppler shift. As a robot explores the environment, the radar signal undergoes doppler shift directly proportional to its velocity. Since doppler shift directly measures velocity, it can accurately estimate translation even in repetitive environments. Furthermore, velocity-to-distance estimates experience lower integration errors and drift compared to accelerometers. Therefore, Radarize utilizes doppler shift as its primary translation estimate.

Using doppler shift to estimate translation is challenging in indoor settings. In outdoor settings, doppler shift is commonly used to estimate velocities, e.g., to detect speeding cars. Unlike outdoor scenarios that have few big reflectors, indoor settings suffer from a clutter of small reflectors. When the radar moves indoors, it observes reflections from multiple reflectors, each with a different relative velocity. The angle between the reflector and the radar determines the doppler shift experienced by the radar. As such, a radar must infer its translation by comparing reflectors from different directions. Radarize creates doppler-azimuth heatmaps from raw radar signals for this objective, i.e., it represents radar reflections as a function of doppler shifts along different angles. In this doppler-azimuth representation, doppler shift across different angles exhibits a unique signature that captures a radar’s self-velocity and heading direction.

**Correlation-based Rotation Estimation:** We observe that while doppler shift is accurate for translational motion, it does not capture rotational motion accurately. Therefore, we need to find an alternative mechanism to estimate rotation of the radar. To estimate rotational motion, we follow a two-step approach. First, Radarize performs a coarse grained rotation estimation using two time-shifted radar frames. Specifically, it compares the rotational shift between two range-azimuth heatmaps that demonstrate the reflected signal intensity at different locations in space. We design a neural network with a unique data augmentation procedure and a cyclic consistency loss to accurately estimate the rotation. While this process is similar to scan matching algorithms discussed above, rotational motion is far less likely to suffer from the degenerate scan

matching cases (unlike translational motions). In the second stage, Radarize relies on the SLAM pipeline discussed below to fine tune these estimates in conjunction with the map of the environment and the translational motion estimated using doppler shift.

**Artifact Rejection:** To address multipath, Radarize includes a smart multipath rejection scheme. It analyzes the range-azimuth output of the radar, and rejects all but the first reflection, i.e., corresponding to the shortest distance, along each direction. Since multipath reflections travel longer distances, this approach suppresses multipath effect. In addition, to counter reflections from floors and ceilings, we introduce an antenna array pre-processing step to reduce reflections from high or low elevations, and a post-processing step to reject such false objects from 2D maps.

We implemented Radarize using PyTorch [42] and evaluated it on a large dataset of 146 trajectories collected across 4 different buildings and mounted on 3 different platforms totalling approximately 4.7 Km of travel distance. In terms of odometry, we find that it outperforms competing radar-inertial and neural-inertial approaches by 5 $\times$  in terms of absolute trajectory error (ATE). In terms of end-to-end SLAM, we find that this value increases up to 8 $\times$ . Additionally, our qualitative results (Fig. 15) show that Radarize can enable very long loop closures up to 80m of total travel distance.

### 1.3 Summary of Contributions

Our key contributions are:

- We present Radarize— a novel SLAM pipeline designed for use with small commodity radar sensors. Radarize exploits the properties of FMCW radar signals to perform radar-based SLAM without need for auxiliary sensors such as IMUs. Yet, Radarize exceeds the performance of radar-based methods leveraging IMUs.
- We identify effective representations of FMCW radar signals for odometry and mapping tasks. We identify sources of artifacts in such representations, and describe techniques for ameliorating them. We also show how data augmentation techniques can enable learning-based techniques to efficiently utilize such representations.
- We show via a large scale and comprehensive evaluation that Radarize generalizes across platforms and environments. It works as well when deployed on humans (e.g., as a wearable) as it does on wheeled ground robots, and can map buildings of diverse architectural styles.
- We contribute a new dataset for radar-driven SLAM spanning three platforms, four buildings, and 4.7 Km of moving distance<sup>1</sup>.

## 2 BACKGROUND

**Radar-based Sensing:** mmWave-based radars have increasingly become accessible and mainstream over the last decade [8, 31]. These devices transmit low power radio signals and capture the reflections of these signals from the surrounding environment. By analyzing these reflections, the radar estimates the distance, angle, and doppler shift corresponding to each reflector in the surrounding.

<sup>1</sup>Demos, code, and data are available at <https://radarize.github.io>

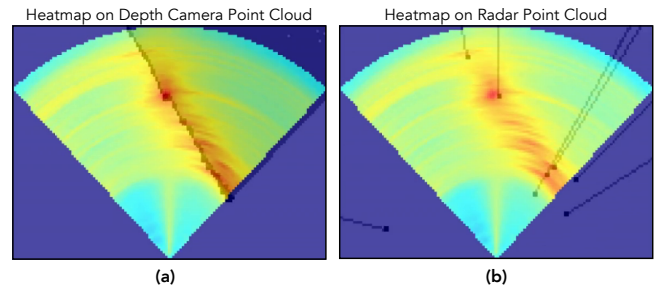


FIGURE 3: The range-azimuth heatmap captures the reflections along different distances and angles. The heatmap is mapped to a cartesian plane for easy visualization. The intensity of reflection varies from blue (low) to red (high). Left. Dense heatmap overlaid on depth camera point cloud. Right. Same heatmap overlaid on sparse radar point cloud. We trace the shadow behind each point for visibility.

**Distance:** A typical radar sensor relies on Frequency Modulated Carrier Wave (FMCW) to measure distance. FMCW signals comprise of chirps, a specific radio signal where the frequency of the signal varies linearly with time. When a reflection is captured by the radar, it is a delayed copy of the chirp and hence lags in frequency. By analyzing the difference in the frequency between transmitted and received chirps, a radar can compute the time delay between them. The time delay corresponds to distance because radio waves travel at the speed of light. The range resolution increases with the bandwidth of the chirp.

**Angle:** A radar sensor relies on multiple transmit/receive antennas to measure the angle of each reflection. The phase of the received (or transmitted) signal at different antennas depends on the angle of arrival (or departure). By estimating the phase difference across antennas, a radar can identify the corresponding angles. The antenna separation is determined by the signal wavelength, and the antenna span defines the angular resolution (higher span offers better resolution).

**Doppler Shift:** A radar can capture multiple reflections over time to estimate the doppler shift corresponding to each reflector in the surrounding. The doppler shift corresponds to the velocity of the reflector. The resolution increases with the total time of signal capture. For more concrete mathematical discussions on radar processing, we refer the reader to [21].

We will use two different representations of the radar data in this paper. First, we use doppler-azimuth heatmaps to denote a 2-dimensional image where each pixel corresponds to a fixed doppler and angle value. The color of the pixel denotes the intensity of reflection emerging from that doppler and angle value. Similarly, we use range-azimuth heatmaps to demonstrate the reflections emerging at particular range and angle values. For this heatmap, the reflectors at different doppler shifts are squished into a single point.

**Simultaneous Localization and Mapping (SLAM):** In SLAM, a moving agent, such as a robot, creates a map of the environment and estimates its trajectory simultaneously. SLAM relies on two types of inputs: scans (e.g., LIDAR scans) and odometry inputs (e.g., inertial sensors that provide relative motion of the agent between scans). The scans of the environment can come from different sensor types such as LIDARs [46, 47], cameras [2, 38, 39, 44], acoustic sensors [6,

7]. These scans are typically represented in the form of probabilistic occupancy grids, where the probability value corresponding to each grid point indicates the probability of presence of an obstacle at that grid point.

We rely on the Cartographer [14] framework for 2D-SLAM. Cartographer was originally designed for LIDAR-based SLAM. It uses a small sequence of scans to create local submaps. Within each submap, it matches the current scan to the submap to estimate the pose of the agent. The odometry input serves as a prior for the agent's pose, and isn't necessarily required since scan matching can produce odometry estimates, especially when using dense optical sensors like LIDARs. As the agent walks the environment, the drift in its pose increases over time. To reduce such drift, Cartographer leverages global optimizations using constraints like loop closures, e.g., if a robot revisits the same location, Cartographer can use this scan to correct for accumulated drift. We refer the reader to [14] for a detailed discussion.

### 3 RADARIZE DESIGN

Before we delve deeper into Radarize, we present a set of goals and choices that motivate our design:

**(i) Streamlined Hardware Requirements:** There is a body of past work that relies on large expensive scanning radars [3, 15, 37, 41] for radar applications. Such radars rely on mechanical motors to spin them and scan their surroundings. We do not consider such radars in this work because they can't support lightweight, compact robots. Instead, we focus on the emerging context of single-chip commodity off-the-shelf (COTS) radars [17, 18, 20]. These radar sensors are inexpensive and small enough to be integrated into mobile devices and small robots. We recognize that more complex radar designs are possible, but we aim for widespread availability, low cost, and ease of use. This choice is consistent with recent work in radar-based SLAM [33, 43, 48].

**(ii) Radar-only Design:** Instead of relying on auxiliary sensors like wheel encoders or IMUs, we opt to use radar's intrinsic properties as the source of odometry. This makes sense for two reasons. First, we observe that the intrinsic properties of radar make IMUs redundant for 2D SLAM. Radar measures doppler shift as a first order primitive. Doppler shift corresponds to velocity and as such, is less prone to drift. Hence, the use of doppler shift can effectively supersede accelerometers. Similarly, rotation can be observed from the shifting of reflectors in the environment over time. Second, adding more sensors adds unnecessary complexity and overhead, often necessitating nontrivial calibration procedures and/or synchronization procedures [52, 54]. By using only a single sensor, we sidestep such concerns.

**(iii) Dense over Sparse Methods:** Radar SLAM methods can roughly be categorized into two categories [1] (a) dense (direct) methods that rely on dense heatmaps, and (b) sparse (indirect) methods that process radar signals into a sparser representation (i.e. keypoints, point clouds) before being used. We show a comparison of a radar heatmap v.s. a radar point cloud in Fig. 3. As shown, sparse representations lose potentially useful information but offer the advantage of smaller computational complexity and better noise/artifact resilience. In contrast, dense methods preserve all the

information at the cost of potential artifacts. We choose the latter approach, opting to capitalize on the benefits of dense methods while introducing new methods to deal with artifacts.

#### 3.1 Design Outline

An overview of Radarize is shown in Fig. 4. Radarize consists of three parts. First, a preprocessing module (shown in beige) converts radar I/Q frames into heatmaps. The tracking module (shown in blue) identifies translation (Sec. 3.2) and rotation (Sec. 3.3) from doppler-azimuth and range-azimuth heatmaps respectively. The mapping module (shown in green) first suppresses vertical reflections (like floors and ceiling), then uses a lightweight segmentation model (i.e. UNet) to segment reflectors on range-azimuth heatmaps (Sec. 3.4). The segmentation model is followed by a de-echoing process that performs multipath suppression. Finally, the outputs of the mapping and tracking modules are input to the SLAM backend, which outputs real-time global map and trajectory estimates.

We tailor our design for the Texas Instruments IWR 1843 radar [20] shown in Fig. 5(left). This radar has three transmit antennas and four receive antennas. One of the transmit antennas is offset in height. We can transmit using each of the transmit antennas to create a virtual antenna array configuration shown in Fig. 5(right). We choose this radar because it has good azimuthal angular resolution (8 virtual antennas in the horizontal plane) and is representative of the capabilities of most small radars.

#### 3.2 Estimating Translation with Doppler-Azimuth Heatmaps

Our first goal is to track the radar's egomotion in the horizontal plane. The radar's egomotion consists of 3 degrees of freedom (two translational and one rotational). Radarize factorizes the task of egomotion estimation into translation estimation and rotation estimation. We describe the process of estimating translational motion using a radar's doppler shift below.

Consider the scenario in Fig. 6(a), which depicts a horizontally-pointed forward-facing radar. Suppose the radar undergoes translational motion with speed  $v$  along direction  $\theta$  (red arrow from the radar). This will induce relative motion from various reflectors in the environment. Suppose there is a reflector along direction  $\phi$  (yellow circle). The resultant relative radial velocity (yellow arrow) will be  $v \cos(\theta + \phi)$ , which in turn induces a doppler shift  $\frac{v \cos(\theta + \phi)}{c} f$ , where  $f$  is the radar signal frequency.

Our goal is to estimate  $v$  and  $\theta$  for the robot, given the observed reflections. It is challenging to estimate both  $v$  and  $\theta$  from a single measurement of doppler shift. However, indoor environments have multiple reflectors. Reflectors are even more abundant at mmWave frequencies, where even smooth surfaces such as walls act as a set of reflectors due to the dispersion effect (see Fig. 2(a)). While smooth surfaces are often considered a challenge for scan matching (leading to degenerate cases), Radarize can use their abundance of reflections as an advantage. Specifically, given a set of reflectors at different directions, the doppler shift exhibits a unique signature that identifies both  $v$  and  $\theta$ . These signatures present themselves clearly in doppler-azimuth heatmaps, where the doppler shift is plotted as a function of angle  $\phi$ .



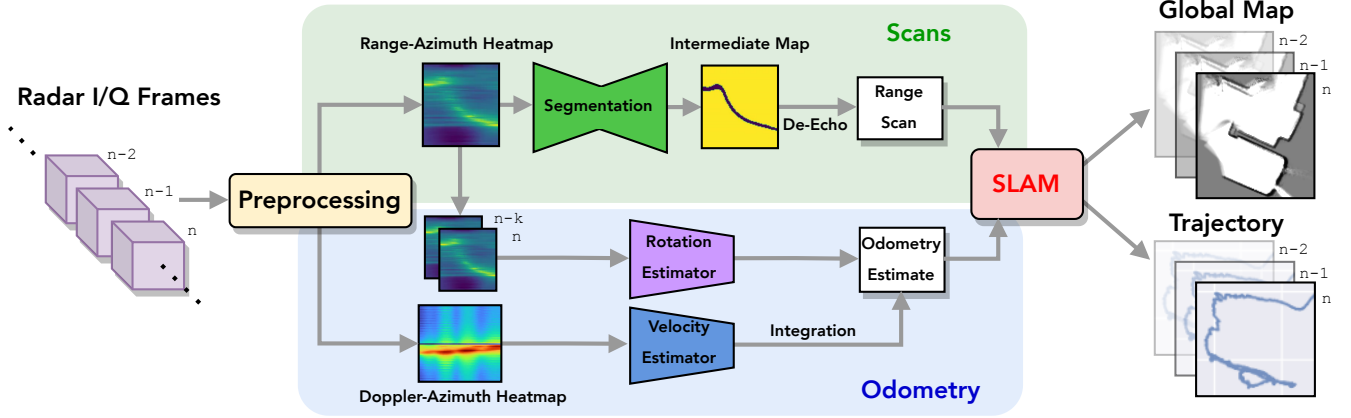


FIGURE 4: Radarize Overview. *Left.* Radar frames are processed into heatmaps. *Top.* The mapping module converts range-azimuth heatmaps into range scans. *Bottom.* The tracking module (a) regresses relative rotation from successive range-azimuth heatmaps and (b) regresses velocities from doppler-azimuth heatmaps and integrates them into odometry estimates. *Right.* An optimization-based SLAM backend outputs real-time map and trajectory estimates.

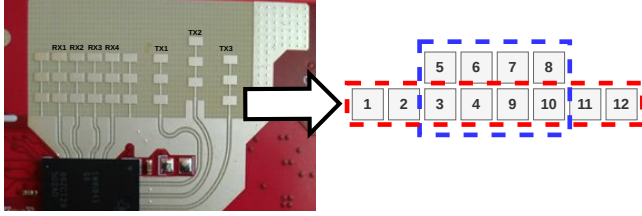


FIGURE 5: *Left.* Physical IWR1843 antenna array. *Right.* Corresponding virtual antenna array. Azimuth-only heatmaps are derived from subarray enclosed in red. Elevation-aware heatmaps are derived from the subarray enclosed in blue.

To illustrate these signatures, we simulate the antenna array depicted in Fig. 5(red) moving in three velocities (forward at 0.30 m/s, right at 0.50 m/s, forward + right at 0.50 m/s) in a reflector-rich environment, then use beamforming and doppler fast-fourier transform (FFT) to plot the resultant doppler-azimuth heatmaps. The resulting plots are shown in Fig. 6(b-d). For Fig. 6(b),  $\theta = 0$  and hence, the doppler signature follows the function  $v \cos \phi$  and peaks at  $\phi = 0$  with an amplitude of  $v$  (as expected). Similar signatures can be observed for  $\theta = \frac{\pi}{2}$  and  $\theta = \frac{\pi}{4}$  in Fig. 6(c) and (d) respectively. We notice that an intuitive visual structure emerges for different values of  $\phi$ . The degree of forward (c.f. backward) motion results in increasing upward (c.f. downward) displacement of all the heatmap peaks. The degree of right (c.f. left) motion results in the peaks sloping increasingly positively (c.f. negatively).

In practice, the heatmaps are sparser and discontinuous, but continue to exhibit similar visual signatures in the doppler-azimuth plots. This interpretable structure motivates us to train a visual model that predicts  $v$  and  $\theta$  from such doppler-azimuth heatmaps. The details of the model are presented in Sec. 4. Once we have a model for the velocity, we derive the inter-frame translation by multiplying with the known inter-frame arrival time.

### 3.3 Estimating Rotation with Range-Azimuth Heatmaps

Using a single-chip radar for rotation estimation is an underexplored and challenging problem. Observe that the doppler-based estimates do not accurately capture rotational motion. This is because during rotational motion, the radial velocity of the reflectors with respect to the radar is zero. Therefore, we cannot rely on our doppler-based translation estimation approach for this task.

Radarize’s rotation estimation approach relies on range-azimuth heatmaps generated by the radar. Consider the radar in Fig. 6(a) undergoing pure rotational motion e.g. rotating clockwise (c.f. counter-clockwise). Then it is expected that the range-azimuth heatmap (see example in Fig. 7) after rotation will undergo a coherent linear transformation, i.e., all reflectors will shift left (c.f. right) as compared to the version of the range-azimuth heatmap before rotation. This is because under pure rotation, each reflector’s range to the radar remains constant, but their bearing angles change by the exact same amount. Moreover, their shift along the azimuth axis is directly proportional to the degree of the radar’s rotation. We plan to use this intuition to estimate the rotational motion.

However, precisely matching reflectors across different frames is a challenging task due to three reasons. First, the low resolution of heatmaps generated by low-cost radars makes reflector identification challenging. Second, when translation motion is mixed with rotation, reflectors in range-azimuth heatmaps undergo various degrees of warping. Finally, some reflectors are only visible at certain angles and do not appear in some heatmaps.

We notice that this task is similar to feature matching in the vision domain. Given the success of machine learning in visual feature matching, Radarize trains a visual model (details in Sec. 4). Given two successive range-azimuth heatmaps, Radarize’s model estimates the relative rotation between them by measuring the extent of coherent left or right-translation (ignoring some potential incoherent warping due to translational motion) by making appropriate visual correspondences between the heatmaps.

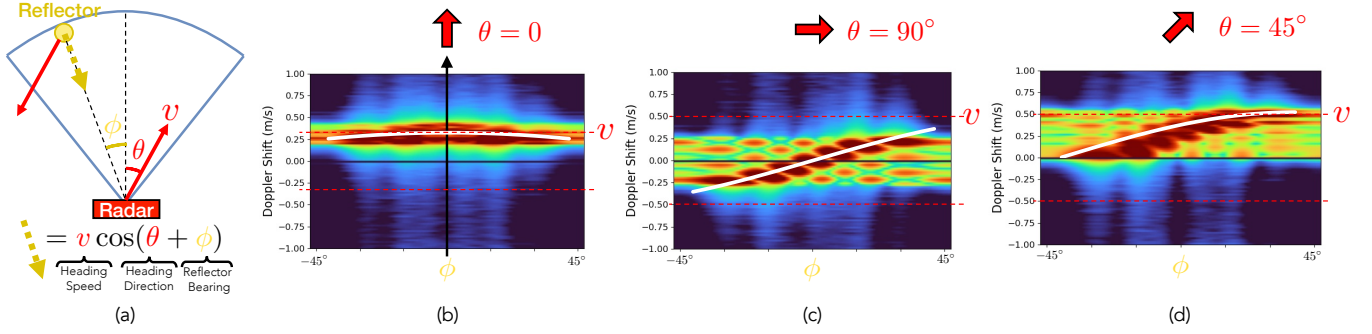


FIGURE 6: Estimating 2D translational motion from doppler-azimuth heatmaps. (a) A forward facing radar "squeezes" reflectors in the environment onto the horizontal plane (blue) and measures their velocity and bearing angle  $\phi$ . (b) Doppler-azimuth heatmap when moving forward at 0.3 m/s. (c) Doppler-azimuth heatmap when moving right at 0.5 m/s. (d) Doppler-azimuth heatmap when moving diagonally (forward + right) at 0.5 m/s. For (b-d), theoretical values are overlaid on the heatmaps as white lines.

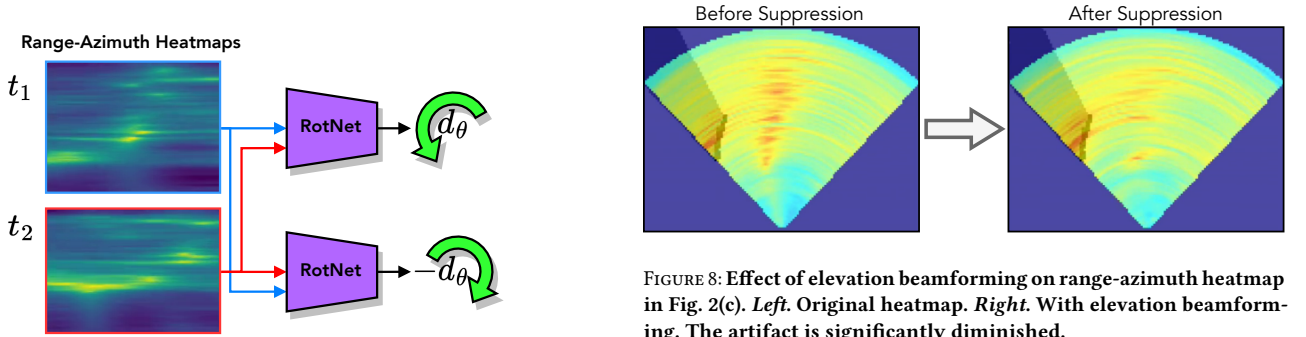


FIGURE 7: Cyclic consistency loss used to train rotation model. The radar undergoes right rotation (and some translation) from  $t_1$  to  $t_2$ , causing the heatmap at  $t_2$  to be left-shifted (modulo some warping).

We propose two data augmentation strategies to make the model generalize to various rotation and translation speeds. First, we vary the time spacing between two successive range-azimuth frames, i.e., we vary  $t_2 - t_1$  in Fig. 7 to expand the range of shifting and warping within the dataset. This helps us augment our training dataset with different velocities. Second, we adopt a cyclic-consistency loss due to the fact that any rotation from  $t_1$  to  $t_2$  implies the same rotation in reverse from  $t_2$  to  $t_1$ , as depicted in Fig. 7. We adopt similar strategy at inference time – we obtain two estimates of rotation by feeding the heatmaps in forward and reverse orders. We average these estimates for the final output.

### 3.4 Generating Local Maps

Our next goal is to generate local maps of the environment surrounding the robot. Multiple local maps are combined together to form the global map using our SLAM backend. As discussed in Sec. 1.1, the key challenge for accurate mapping is the artifacts due to 3D-2D conversion artifacts (Fig. 2(c)) and multipath effects (Fig. 2(b)).

**Artifact Rejection:** Consider the antenna array in Fig. 5 which consists of two virtual arrays – the 1D azimuth-only antenna array (red) and the 2D elevation-aware antenna array (blue). Deriving a range-azimuth heatmap from the former has a high azimuthal angular resolution. However, it is not elevation-aware, and thus is prone to artifacts from floors and ceilings (Fig. 8 left). We use the

2D virtual array to suppress such artifacts. Specifically, we create an elevation-aware range-azimuth heatmap by setting the beamforming weights such that the radar points straight ahead in the elevation dimension. This elevation-aware heatmap encapsulates elevation information at the cost of azimuthal angular resolution. We can see an example of such artifact suppression in the heatmap in Fig. 8 (right). Our observation is that legitimate in-plane reflectors are consistent across both heatmaps, while ceiling and floor reflectors are suppressed in the elevation-aware heatmaps.

We aim to have Radarize's mapping module benefit from this observation. Therefore, for each incoming radar frame, we compute both the pure-azimuthal heatmap as well as the elevation-aware heatmap. We then feed both heatmaps to a segmentation model. The model can leverage the former heatmap to output a high resolution map, and leveraging comparisons between the two to weed out 3D to 2D conversion artifacts.

**Echo Suppression:** Another source of errors is the presence of phantoms due to multipath propagation (Fig. 2(b)). We propose a simple heuristic to diminish the effect of such artifacts – we remove all but the first reflection along each direction from the output of the segmentation module. Intuitively, the output of the segmentation module is a pixel-wise binary classification of the reflectors within a scene. This can be interpreted as a 2D occupancy grid of points, some of which are behind others from the radar's point of view. Due to the presence of phantoms in radar heatmaps, we should place most of our confidence only on the first reflector encountered along any particular bearing angle.

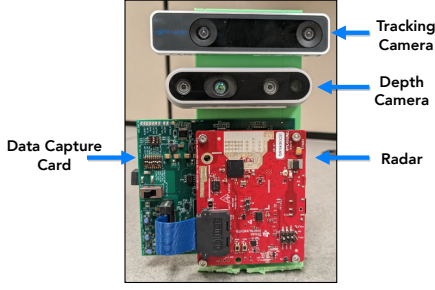


FIGURE 9: Sensor suite used for data collection.

### 3.5 Integrating Odometry and Mapping

We feed the odometry outputs from our tracking module and the sequence of scans from our mapping module to the state-of-the-art optimization-based SLAM backend (Cartographer [14]). As a production-grade SLAM system, Cartographer can continuously merge these inputs to generate a map of the environment and the trajectory of the device in real-time, even on computationally lightweight devices.

## 4 IMPLEMENTATION

### 4.1 Hardware Design

**Sensors:** We use the hardware setup shown in Fig. 9. It consists of an Intel T265 tracking camera [23], Intel D435i stereo depth camera [22], an Texas Instruments IWR1843 radar [20], and a DCA1000 data capture card [19] attached to a 3D printed fixed frame. We use the tracking and depth cameras solely to obtain the pseudo-ground truth trajectories and depth maps respectively. The data capture card is used to capture unprocessed I/Q samples from the radar rather than preprocessed sparse point clouds.

In total, this setup allows us to capture a sequence of colocated depth images from a stereo depth camera, accelerometer and gyroscope readings from its built-in IMU, sparse point clouds from the radar, frames of radar I/Q samples from the data capture board, as well as pseudo-ground truth poses as obtained from a state-of-the-art visual odometry sensor. *Note that during deployment, only the data capture card and radar components are necessary.*

**Parameters:** We tune the parameters of the sensors to be appropriate for indoor settings. We set the chirp parameters of the radar such that the maximum unambiguous range is 4.284m and the range resolution is 4cm. We adjust the maximum depth of the depth camera to match that of the radar. Since the depth camera’s horizontal field of view is limited to 88 degrees ( $[-44^\circ, 44^\circ]$  deviation from boresight), we compute range-azimuth heatmaps within the same range (with an angular bin spacing of  $1^\circ$ ). As a result, the size of our range-azimuth heatmaps is  $N_r = 96$  by  $N_\theta^r = 88$ .

Additionally, we adjust the inter-chirp interval to be  $516\mu\text{s}$ . This yields  $v_{\max} = \frac{\lambda}{4 * T_c} = 1.89 \text{ m/s}$ , which is a reasonable upper bound on human walking speed. For our doppler-azimuth heatmaps, we find that  $N_c = 96$  and  $N_\theta^c = 180$  (with  $1^\circ$  angular bin spacing) to be a good value. This gives us a velocity resolution of  $v_{\text{res}} = \frac{\lambda}{2 * N_c * T_c} = 0.04 \text{ m/s}$ . Finally, the sensors operate at different frequencies (i.e. 200 Hz for the IMU, 100 Hz for the camera). We set the radar frame rate to be 30 Hz and synchronize all sensor readings to the radar.

FIGURE 10: *Left.* Moving sensors by hand. *Center.* Sensors mounted on a cart. *Right.* Sensors mounted on robot.

Platform	Building				Total
	A	B	C	D	
Handheld	15	9	12	11	47
Cart	10	11	15	9	45
Roomba	9	15	15	15	54
<b>Total</b>	34	35	42	35	146

TABLE 1: Breakdown of the trajectories in our dataset.

### 4.2 Dataset Collection

We collect a large and diverse dataset of trajectories spanning several floors across four different campus buildings A, B, C, and D using the set of sensors depicted in Fig. 9. During data collection, we ensure that there are no other moving objects in the environment. This is consistent with past work in SLAM, which classically assumes a static environment.

**Buildings:** The buildings were chosen to maximize the diversity of architectural styles and construction materials. Building A is a modern construction featuring many glass and metallic surfaces. Building B is the most dated construction and primarily consists of wood and drywall. Building C comprises mainly of brick and metallic surfaces. Building D is an intermediary between building B and C. We choose random floors in each building to randomize the layout.

**Platforms:** In addition to diversity of environments, we also aim for a diversity of platforms. We move the sensors in one of three ways as shown in Fig. 10: (a) *Hand-held:* We instruct a set of volunteers to carry the sensors by hand while walking around the building. The volunteers are encouraged to encompass the full range of 2D top-down motions – for example, they can choose to move backwards or sideways through a narrow hallway. This leads to a challenging set of trajectories featuring sudden accelerations and fast rotations. (b) *Cart-driven:* We mount the sensors onto a cart and push it around the building. The trajectories are more constrained and consist mostly of constant linear velocity motion and slow rotations. Across both aforementioned cases, the set of linear speeds range up to a brisk walking pace of 1.6 m/s. (c) *Robot-driven:* We mount the sensors onto a small ground robot (the iRobot Create3 [24]) and use teleoperation to drive the robot around the building. The robot has a maximum linear velocity of 0.5 m/s and holds the sensors significantly closer to the ground than the previous cases.

**Dataset Summary:** We collect 146 trajectories, with roughly even division across movement types and buildings. Each trajectory consists of around a minute of capture time. The mean travel distance across all handheld and cart-driven trajectories is over 50m. Owing to the relatively slow speed of the ground robot, the mean

travel distance for the robot trajectories is around 20m. In total, our dataset consists of roughly 226,800 data samples and 4680m of travel distance. Table 1 contains a summary of the dataset.

**Caveat on Adverse Conditions:** We do not collect data in adverse conditions (e.g. in occluded environments or low-light environments) for two reasons. First, such conditions would make the optical sensors (i.e. depth and tracking camera) fail and produce junk values, depriving us of the pseudo ground-truth for evaluation. Second, numerous prior works have already shown the robustness of radar to both occluded environments and low-light environments. Specifically, [43] immersed a radar within a smoke chamber and found no difference in the output, and [48] reports identical performance of radar in both light and dark conditions.

### 4.3 Model Architecture & Training

**Translation Model:** We use a CNN based on ResNet18 [12]. We modify the input to accept a single-channel doppler-azimuth heatmap of size  $N_c \times N_\theta^c$  and modify the output layer to output a two-dimensional velocity estimate. To train the model, we use ground truth velocity estimates as obtained from our tracking camera as supervision. We use the standard L2 (mean-squared error) loss for training. Our optimizer is Adam [25] with parameters  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ . We use a learning rate of  $1e-3$  and a batch size of 128. For data augmentation, we apply random vertical flipping on the doppler-azimuth heatmap images. Specifically, for a data sample consisting of a doppler-azimuth heatmap image and a velocity label  $v$ , we vertically flip the doppler-azimuth heatmap and assign the label  $-v$ .

**Rotation Model:** We again use a CNN based on ResNet18 [12]. We modify the input to accept a two-channel image of size  $N_r \times N_\theta^r$  consisting of stacked range-azimuth heatmaps i.e. a range-azimuth heatmap observed at current time,  $H_n$ , with a previous observation  $H_{n-k}$ , where  $k > 1$ . The output is a single value  $\hat{\Delta\theta}_{n-k}^n$  corresponding to the rotation estimate. To train our model, we use the pseudo-ground truth rotation estimates from the tracking camera as supervision. We use a standard L2 (mean-squared error) loss with cyclic consistency constraint given by

$$L_{rot}(H_{n-k}, H_n) = \|\Delta\theta_{n-k}^n - \hat{\Delta\theta}_{n-k}^n\|_2^2 + \|\Delta\theta_{n-k}^{n-k} - \hat{\Delta\theta}_{n-k}^{n-k}\|_2^2 \quad (1)$$

Our optimizer is Adam [25] with parameters  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ . We use a learning rate of  $1e-3$  and a batch size of 128. We leverage two kinds of data augmentation. First, we design a horizontal flipping data augmentation. Specifically, for an incoming data sample (which consists of a stacked  $H_n$  and  $H_{n-k}$  and label  $\Delta\theta_{n-k}^n$ ) we randomly apply horizontal flipping on the stacked heatmap image and set the label of the sample to be  $-\Delta\theta_{n-k}^n$  accordingly. Secondly, we vary  $k$  randomly for each training sample to lie within a small range – we find empirically that 1-5 frames is a good value.

**Segmentation Model:** We choose UNet [45] as a backbone as it is fast and achieves state-of-the-art results on segmentation tasks. The input to the model is a two-channel image of size  $N_r \times N_\theta^r$ . This is obtained by stacking the pure-azimuthal and elevation-aware range-azimuth heatmaps. The output of the model is a 2 channel range-azimuth image of occupancy logits of the same size, which we convert into probabilities using the softmax function. Thresholding

the output converts it into a 2D grid of discrete points. For the ground truth, we use the point clouds given by the stereo depth camera. For training, we use the Dice loss with Adam [25]. Our optimizer is Adam [25] with parameters  $\beta_1 = 0.8$ ,  $\beta_2 = 0.9$ . We use a learning rate of  $1e-4$  and a batch size of 48. For data augmentation, we use random horizontal flipping on the range-azimuth heatmaps and depth ground-truth.

## 5 EVALUATION

We evaluate our method by contrasting it with several representative state-of-the-art baselines (or combinations thereof):

- **RadarHD** [43] is a radar super-resolution method. It leverages an asymmetric UNet architecture to upsample range-azimuth heatmaps into high angular resolution point clouds. The point clouds are subsequently utilized for downstream localization and mapping tasks.
- **milliEgo** [34] is an radar-inertial odometry estimator based on sparse point clouds. It leverages a cross-attention mechanism to merge odometry estimates from point cloud observations with auxiliary IMU data.
- **RNIN** [5] is a neural-inertial navigation model that regresses IMU readings into odometry estimates. These estimates are designed to be integrated into various downstream state estimation pipelines.

We use the publicly available implementation and pretrained weights for the baselines. For fairness, we also fine-tune the models on our own dataset.

**Metrics:** To compute the relevant evaluation metrics we use the evo package [11], widely used to benchmark the performance of odometry and SLAM algorithms. We consider two main metrics: Absolute Trajectory Error (ATE) and Relative Error (RE) for both position and rotation (heading). While ATE captures errors over the entire trajectory, RE captures relative error over smaller trajectory chunks. We refer the reader to [53] for an in-depth explanation of these metrics.

### 5.1 Experimental Setup

Our experiments are designed to answer the following:

- How well does Radarize's compare with IMU-aided and/or super-resolution based methods?
- How well does Radarize generalize across different environments and platforms?

To answer the first question, we split our dataset into 5 random training/validation/testing splits in a ratio of 70/10/20 and aggregate the results across all splits. To answer the second question, we consider train/test splits across different buildings and platforms types.

### 5.2 Odometry Performance

First, we evaluate our odometry estimation pipeline (Sec. 3). We summarize our results in Table 2. Our experiments show that our odometry estimation method is superior to the baselines by a wide margin, in spite of not leveraging any IMU information. We present a detailed breakdown below.



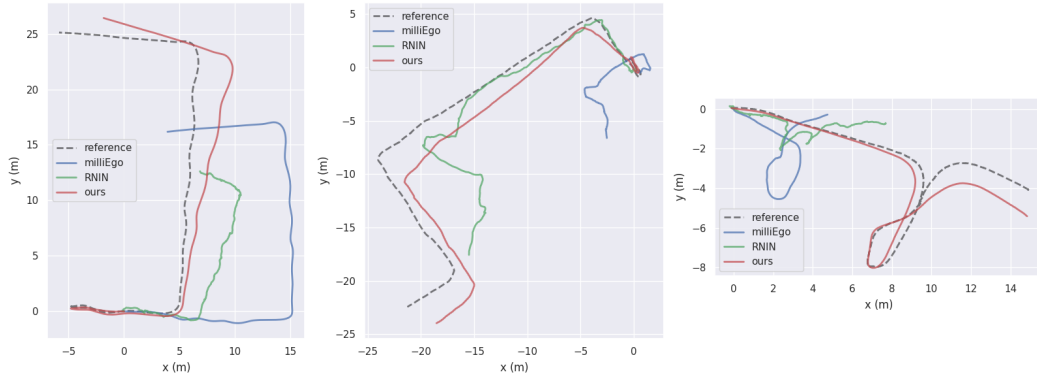


FIGURE 11: Radarize’s odometry achieves correct scale. We observe IMU-based methods are able to reconstruct the shape of long and linear trajectories, but struggle with absolute scale.

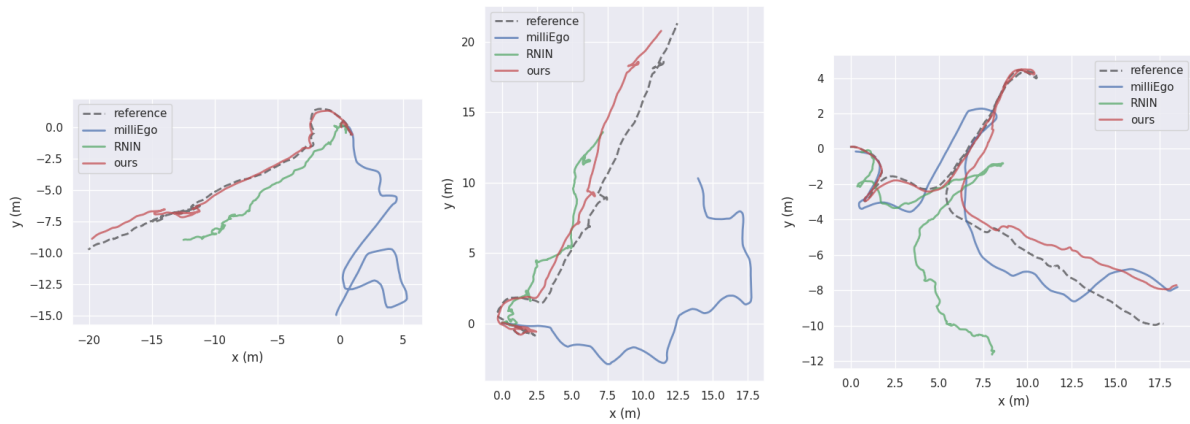


FIGURE 12: Radarize’s odometry is flexible. Our method excels at the handheld trajectories which feature a wide range of motions, whereas the other methods become brittle.

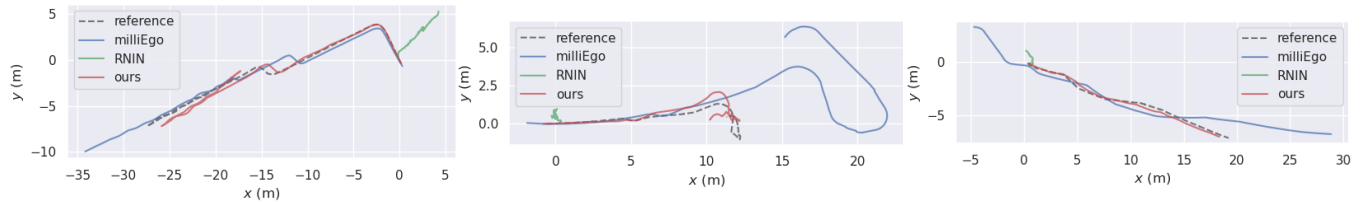


FIGURE 13: Radarize’s odometry is robust. We compare performance on OOD trajectories by training on the handheld trajectories and testing on wheeled ones. Methods leveraging IMUs struggle, whereas Radarize retains similar performance.

**Achieving Accurate Scale:** In general, we observe that IMU-based methods can reconstruct the shape of long trajectories, but struggle with its scale. This is substantiated by our cart-driven experiments, which consists primarily of forward motion down long hallways (Fig. 11). This is due to the IMU-based methods having to perform double integration for translation but only single integration for heading. As such, IMU-based methods perform better on heading direction estimation than on translation estimation. By contrast, Radarize almost always achieves the correct scale because it directly estimates velocity and needs only single integration for translation.

**Accommodating Diverse Motions:** Recall that in our dataset, we collect handheld trajectories with more unrestrained motion than

when placing the sensors on wheels (i.e. the cart or robot). We find that milliEgo works reasonably well when exposed to cart or robot motion which consists mainly of forward and backward movement down long hallways, but fails when exposed to sideways movement or fast rotations (Fig. 12). This is likely due to the sparse and fickle nature of radar point clouds.

**Robustness to Out-of-Distribution (OOD) Motions:** To what extent is Radarize’s odometry pipeline invariant to platform-specific features? To answer this question, we train all odometry methods on the handheld trajectories in our dataset and test on the cart-driven and robot-driven parts of our dataset. As shown in Table 3, Radarize is far more robust to shifts in platforms than other

Method	ATE		RE	
	Pos (m)	Rot (rad)	Pos (m)	Rot (rad)
milliEgo [34]	4.100	1.054	0.059	1.018
RNIN [5]	2.993	0.542	0.018	0.524
<b>Radarize (Ours)</b>	<b>0.805</b>	<b>0.366</b>	<b>0.014</b>	<b>0.355</b>

TABLE 2: Comparing odometry performance. Results are averaged across 5 random dataset splits. Lower is better.

Method	ATE		RE	
	Pos (m)	Rot (rad)	Pos (m)	Rot (rad)
milliEgo [34]	3.977	0.817	0.050	0.807
RNIN [5]	5.122	1.927	0.019	1.904
<b>Radarize (Ours)</b>	<b>1.121</b>	<b>0.565</b>	<b>0.013</b>	<b>0.559</b>

TABLE 3: Testing odometry generalization to OOD motion. Each method was trained on the handheld trajectories and tested on the cart and robot trajectories.

methods. This is due to IMU-aided methods leveraging and thus overfitting to platform-specific side-channel information (e.g., walking gaits, floor tile spacing, imperfections in wheels, etc. [4, 5, 13]). As expected, RNIN has the worst OOD performance with milliEgo performing slightly better due to access to radar data (Fig. 13). Radarize outperforms the baselines due to not leveraging such side channel information.

### 5.3 SLAM Performance

Next, we evaluate the quality of trajectories and maps obtained after feeding odometry and scans from the baselines and our method into the SLAM backend [14]. The results are summarized in Table 4. We note that the performance of the scan matching only approach [43] is the worst, followed by methods leveraging odometry sources. This is due to the prevalence of degenerate environments in indoor environments such as long hallways where repeating features like flat walls confuse scan matching algorithms. By contrast, Radarize’s performance stems from a combination of its accurate odometry and scans, which the pose graph backend can optimize to yield high quality trajectories and consistent maps.

**Map Quality:** We show qualitative examples of Radarize’s local map fragments in Fig. 14. We find that in spite of having more points per scan, RadarHD does not add any improvement over Radarize. This is due to the excess points in RadarHD contributing noise (e.g., due to not suppressing 3D to 2D conversion artifacts), which impedes the SLAM backend’s ability to optimize and coherently combine scans into a consistent map.

**Loop Closures:** The lack of compounding errors due to high-quality local mapping means that Radarize enables long range loop closures, such as those in excess of 50m. We show examples in Fig. 15, where the SLAM backend successfully closes long range loops.

**Generalization Across Environments:** We test whether Radarize can successfully generalize across environments. We choose a subset of the 3 most distinctive buildings in our dataset (A,B,C). We test on trajectories in one building, and train on trajectories in the remaining buildings. The results are shown in Table 5. We observe a small but minor performance decrease, which suggests that Radarize learns mostly environment-invariant features.

### 5.4 Computational Requirements

Finally, we quantify Radarize’s computational requirements. We deploy Radarize on two different host devices (a desktop computer and single-board edge computer). The desktop computer is equipped with a 12-core AMD Ryzen 9 3900X processor, 32 GB RAM, and an NVIDIA RTX 3090 GPU. The Jetson AGX Orin [40] is a small form-factor computer geared towards mobile edge machine learning applications such as robotics. It is equipped with a 12-core Arm Cortex-A78AE CPU, 32 GB RAM, and an 2048-core NVIDIA Ampere architecture GPU with 64 Tensor Cores.

We report the running time of each of Radarize’s components in Table 6. To determine the total pipeline runtime, we sum along the longest (bottleneck) path. We obtain a total runtime of 17.06 ms (desktop) and 31.22 ms (AGX Orin). This translates to an update rate of 59 Hz and 32 Hz, which corresponds to super-realtime and realtime performance respectively. We conclude that Radarize is computationally efficient enough to be deployed as a realtime system even on edge compute devices such as the AGX Orin.

## 6 RELATED WORK

**Methods Using Large Radars:** Large spinning radars have been used to perform localization and mapping on vehicles in outdoor environments. Recent work in this space includes [3], RADARODO [37], Pharaoh [41], RadarSLAM [15]. As GPS is available in outdoor environments, some of these approaches fuse their estimates using GPS [30]. Radarize differs along two axes. First, these methods are suited for larger and more expensive radars with considerably greater resolution. Second, these works do not address the unique challenges of using radar within indoor environments such as multipath effects and phantom reflections.

**Methods Using Commodity Hardware:** As large radars are impractical indoors and on resource constrained edge devices, there has been recent interest in leveraging small low cost radars and commodity-grade IMUs. Works involving fusion of radar and IMU data includes [26] which relies on classical methods and milliEgo [34] which leverages deep learning. We note there is also a body of work focusing purely on IMU-based odometry, such as RONIN [13] RNIN-VIO [5], which could be fused with radar in a loosely-coupled framework. Unlike these works, Radarize develops purely radar-based odometry by relying on doppler shift, and does not require inertial sensors. Finally, there are works also using doppler shift for egomotion estimation such as [27] and BatMobility[48]. BatMobility leverages doppler shift and machine learning to estimate ground-parallel translational motion of an autonomous UAV. [27] uses doppler shift to estimate translation and line-fitting with RANSAC to estimate rotation within a hallway environment. In contrast to these methods, Radarize does not require strong apriori assumptions about the environment (e.g. existence of a flat ground surface in BatMobility, or a hallway environment with parallel flat walls as in [27]).

**Methods Based on Super-Resolution:** Radar is noted for its low resolution when compared with its optical counterpart (i.e. lidar). Several works attempt to solve this issue through deep-learning based super resolution techniques. milliMap [33] applies GANs to densify top-down projections of sparse radar point clouds.

Method		ATE		RE	
Odometry	Scans	Position (m)	Rotation (rad)	Position (m)	Rotation (rad)
—	RadarHD [43]	4.833	0.751	0.045	0.726
milliEgo [34]	RadarHD [43]	3.900	0.808	0.022	0.780
RNIN [5]	RadarHD [43]	3.299	0.541	0.020	0.522
Radarize (Ours)	RadarHD [43]	0.658	0.154	0.018	0.149
<b>Radarize (Ours)</b>	<b>Radarize (Ours)</b>	<b>0.606</b>	<b>0.116</b>	<b>0.017</b>	<b>0.113</b>

TABLE 4: Comparing SLAM performance. Results are averaged across 5 random dataset splits. Lower is better.

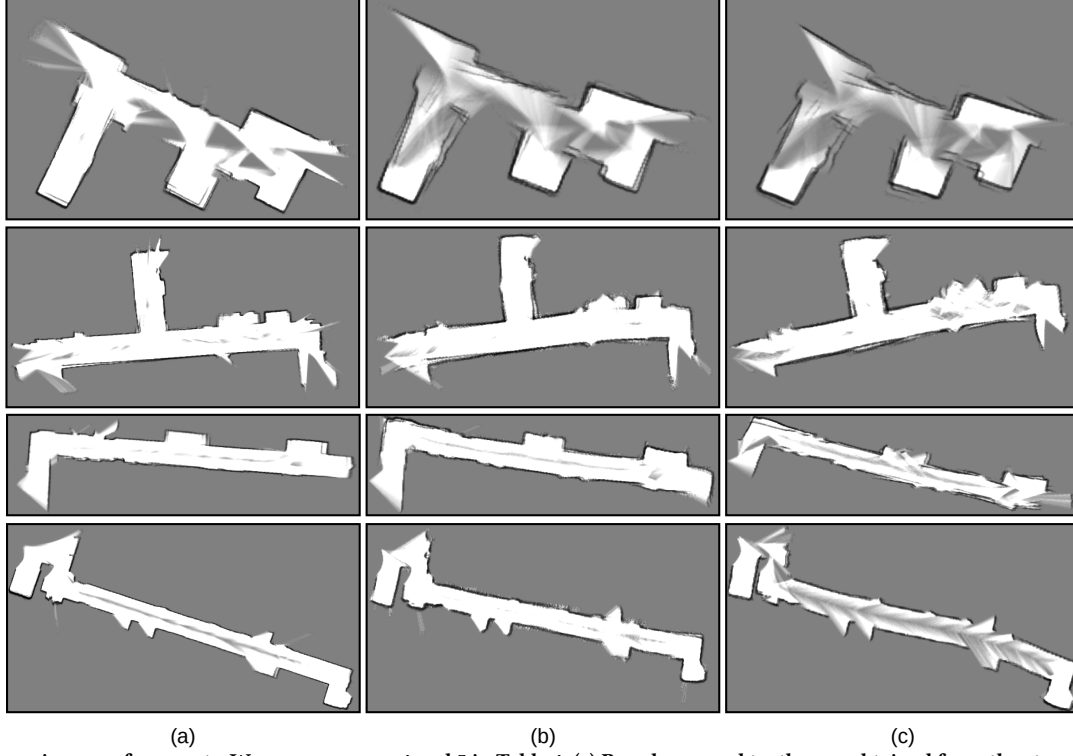


FIGURE 14: Comparing map fragments. We compare rows 4 and 5 in Table 4. (a) Pseudo-ground truth map obtained from the stereo depth camera. (b) Radarize map reconstruction. (c) RadarHD map reconstruction. Radarize enables better mapping via artifact and multipath suppression.

Building	ATE		RE	
	Pos (m)	Rot (rad)	Pos (m)	Rot (rad)
A	0.635	0.102	0.018	0.100
B	0.696	0.179	0.016	0.175
C	0.492	0.105	0.016	0.104
<b>Mean</b>	<b>0.608</b>	<b>0.129</b>	<b>0.017</b>	<b>0.126</b>

TABLE 5: Testing Radarize’s generalization across environments (tested on one building and trained on the rest).

Device	Mean Runtime (ms)					Total
	Preproc.	Trans.	Rot.	Map	SLAM	
Desktop	9.53	2.23	6.60	3.11	0.93	17.06
AGX Orin	11.76	7.57	14.78	9.43	4.68	31.22

TABLE 6: Mean runtime of each stage of Radarize’s pipeline on different devices. The total runtime is computed along the longest path.

RadarHD [43] applies an asymmetric UNet segmentation to upsample range-azimuth heatmaps across the angular dimension. Unlike these works, we identify 3D to 2D conversion from ceilings

and floors as a common source of mapping artifacts in indoor environments, and address this using our preprocessing algorithm which takes into account elevation. Finally, while milliMap relies on explicit ground truth inputs for odometry and RadarHD infers them using scan matching on radar point clouds, Radarize relies on doppler-based odometry.

## 7 CONCLUDING DISCUSSION

We present Radarize, a radar-native SLAM pipeline using only a small commodity mmWave radar. Radarize exploits the properties of small FMCW radars in indoor environments to simultaneously perform accurate odometry and mapping. Radarize is flexible enough to be mounted on different platforms, such as headsets or handsets mounted on humans or robots close to the ground. We conclude with a discussion of limitations and future work:

**Range Limits:** We use a commodity-grade stereo depth camera with an ideal operating range of less than 5m. As such, we set the radar’s maximum range to match that of the depth camera.

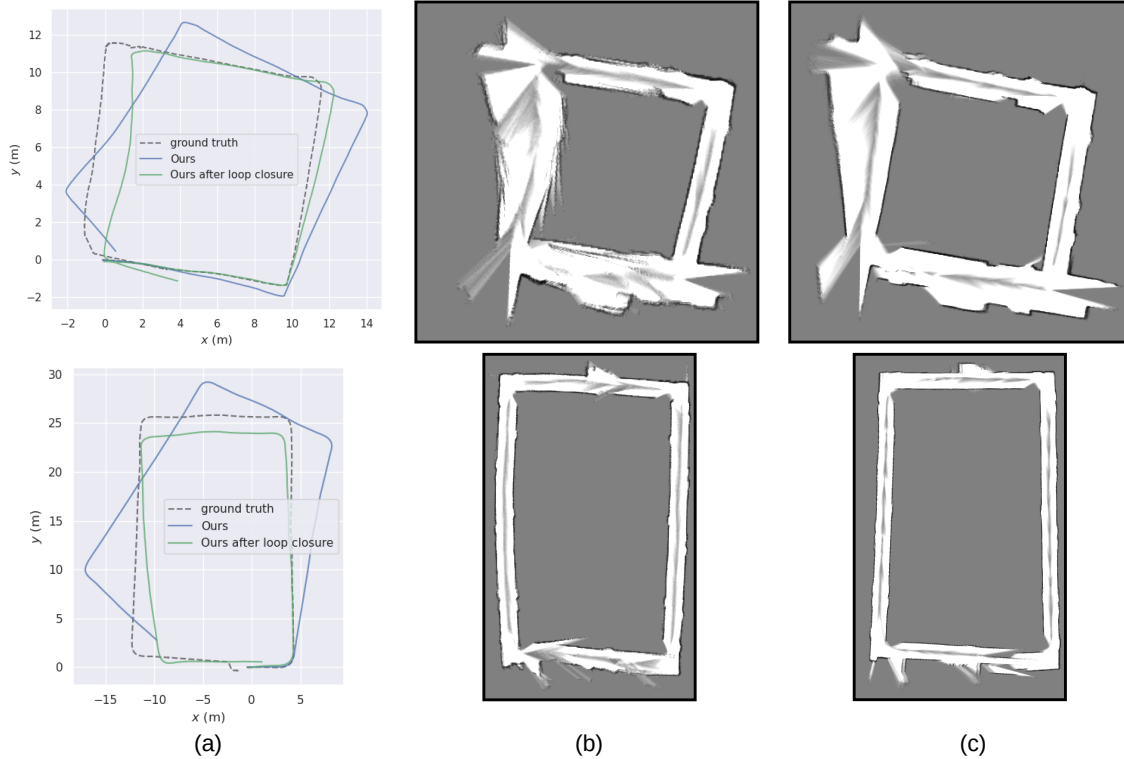


FIGURE 15: Radarize enables large-scale consistent maps. Radarize’s odometry has low drift over long distances, enabling long-range loop closures. (a) Trajectory estimate before and after loop closure. (b) Estimated map. (c) Pseudo ground-truth from depth + tracking camera.

Although this suffices for most indoor environments, it might be prudent to extend the range of radar for operation in larger spaces or in outdoor scenarios. This will require retraining Radarize’s models using ground truth from a more expensive lidar sensor.

**Speed Limits:** As explained in Sec. 4, our dataset is collected by humans carrying a sensor either by hand, on a cart, or on a slow moving ground robot. Hence, the maximum linear speed in our dataset is 1.6 m/s. We set our radar parameters such that the maximum unambiguous velocity corresponds to around the same limit. We believe this is a reasonable speed for most ground robots and even humans. However, if a larger speed limit is necessary, it is possible to increase the maximum unambiguous velocity of the radar by reconfiguring the chirp parameters.

**Dynamic Objects:** As mentioned in Sec. 4, we do not train or test our system in the presence of a dynamic objects in the environment. Similar to how the performance of camera or lidar based SLAM deteriorates in dynamic environments [36], the presence of moving objects within the environment is likely to adversely affect the performance of the system. We leave the resolution of such issues to future work.

## ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd Wen Hu for their insightful comments and suggestions on improving this paper. This work was supported in part by NSF RINGS Award 2148583.

## A ARTIFACT APPENDIX

### A.1 Abstract

This artifact contains the code and dataset for the paper Radarize: Enhancing Radar SLAM with Generalizable Doppler-Based Odometry (MobiSys’24). It includes the pretrained models and experimental configurations to reproduce the main results in the paper. Additionally, it includes a large dataset of 146 trajectories spanning 4 different buildings and collected on 3 different platforms. We hope this proves useful for further research into radar-based SLAM methods.

### A.2 Artifact check-list (meta-information)

- **Run-time environment:** Ubuntu 20.04 machine with sudo access, conda environment with Python 3.8+.
- **Hardware:** x86\_64 CPU, CUDA  $\geq 11$ , 3-capable GPU, 32 GB RAM
- **Run-time state:** Small random deviation in the output may result due to non-determinism in the SLAM backend optimization.
- **Metrics:** Absolute Trajectory Error (ATE), Relative Error (RE).
- **Output:** Result tables for odometry and SLAM.
- **Experiments:** Run top-level bash scripts.
- **How much disk space required (approximately)?:** 1 TB
- **How much time is needed to prepare workflow (approximately)?:** Approximately 1 hour.
- **How much time is needed to complete experiments (approximately)?:** If doing evaluation only: approximately 8 hours. If training from scratch: approximately 1 day.
- **Publicly available?:** Yes
- **Workflow framework used?:** yacs



- **Archived (provide DOI)?:** [10.5281/zenodo.11093859](https://doi.org/10.5281/zenodo.11093859)

### A.3 Description

**A.3.1 How to access.** The code is available in a GitHub repository linked at <https://radarize.github.io>. The data is available at DOI [10.5281/zenodo.11093859](https://doi.org/10.5281/zenodo.11093859). It contains the following .zip files:

- eval.zip contains pretrained models and baseline outputs.
- dataset.zip contains the 146 trajectories in rosbag format.

**A.3.2 Hardware dependencies.**

- x86\_64 workstation.
- 32 GB RAM.
- 1 TB of free storage.
- CUDA >=11.3-capable GPU.

**A.3.3 Software dependencies.**

- Ubuntu 20.04.
- ROS Noetic.
- git.
- conda.
- imagemagick.

### A.4 Installation

- (1) Install all software dependencies (e.g. ROS Noetic, git, conda, imagemagick) via apt-get.
- (2) Clone the GitHub repository into a folder e.g. ~/radarize\_ae/.
- (3) Download eval.zip and dataset.zip from [10.5281/zenodo.11093859](https://doi.org/10.5281/zenodo.11093859). Place these inside radarize\_ae/.
- (4) cd radarize\_ae/ and unzip the zip files i.e.
 

```
unzip eval.zip
unzip dataset.zip
```
- (5) Install conda environment with
 

```
conda env create -f env.yaml
```
- (6) Source environment with conda env activate radarize\_ae. Then run
 

```
pip install -e .
```
- (7) Create a catkin workspace e.g. ~/catkin\_ws. Install cartographer\_ros into this workspace. Under radarize\_ae/cartographer/, copy the contents of configuration\_files/ and launch/ into ~/catkin\_ws/install\_isolated/share/cartographer\_ros/.
- (8) Source the cartographer\_ros environment with
 

```
source ~/catkin_ws/install_isolated/setup.bash
```
- (9) After sourcing both the conda environment and catkin\_ws concurrently, you should now be able to run all scripts under radarize\_ae/.

### A.5 Experiment workflow

- To generate the main results in the paper, run run\_eval.sh.
- To train models from scratch, run run.sh.

### A.6 Evaluation and expected results

First, run the evaluation top-level ./run\_eval.sh. Then to generate the tables, we provide two scripts to run:

- odom\_eval.sh will write an output to odom\_result.txt. It should approximately match Table 2.
- slam\_eval.sh will write an output to slam\_result.txt. It should approximately match Table 4.

**Note:** Minor deviation from these numbers might occur due to the non-determinism of the backend SLAM optimization.

### REFERENCES

- [1] ABU-ALRUB, N. J., AND RAWASHDEH, N. A. Radar odometry for autonomous ground vehicles: A survey of methods and datasets. Publisher: arXiv Version Number: 1.
- [2] CAMPOS, C., ELVIRA, R., RODRIGUEZ, J. J. G., M. MONTIEL, J. M., AND D. TARDOS, J. ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM. 1874–1890.
- [3] CEN, S. H., AND NEWMAN, P. Precise ego-motion estimation with millimeter-wave radar under diverse and challenging conditions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), IEEE, pp. 1–8.
- [4] CHEN, C., LU, X., MARKHAM, A., AND TRIGONI, N. IONet: Learning to cure the curse of drift in inertial odometry. Publisher: arXiv Version Number: 1.
- [5] CHEN, D., WANG, N., XU, R., XIE, W., BAO, H., AND ZHANG, G. RNIN-VIO: Robust neural inertial navigation aided visual-inertial odometry in challenging scenes. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (2021), IEEE, pp. 275–283.
- [6] DOKMANIC, I., DAUDET, L., AND VETTERLI, M. From acoustic room reconstruction to slam. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), IEEE, pp. 6345–6349.
- [7] EVERS, C., AND NAYLOR, P. A. Acoustic SLAM. 1484–1498.
- [8] FELIC, G. K., EVANS, R., DUONG, H. T., LE, H. V., LI, J., AND SKAFIDAS, E. Single-chip millimeter wave radar. *Microwave J* 58 (2015), 108–116.
- [9] FENG, Z., LI, J., ZHANG, L., AND CHEN, C. Online spatial and temporal calibration for monocular direct visual-inertial odometry. 2273.
- [10] FILIP, I., PYO, J., LEE, M., AND JOE, H. LiDAR SLAM with a wheel encoder in a featureless tunnel environment. 1002.
- [11] GRUPP, M. evo: Python package for the evaluation of odometry and slam. <https://github.com/MichaelGrupp/evo>, 2017.
- [12] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.
- [13] HERATH, S., YAN, H., AND FURUKAWA, Y. RoNIN: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), IEEE, pp. 3146–3152.
- [14] HESS, W., KOHLER, D., RAPP, H., AND ANDOR, D. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)* (2016), pp. 1271–1278.
- [15] HONG, Z., PETILLOT, Y., AND WANG, S. RadarSLAM: Radar based large-scale SLAM in all weathers. Publisher: arXiv Version Number: 1.
- [16] HUANG, W., LIU, H., AND WAN, W. Online initialization and extrinsic spatial-temporal calibration for monocular visual-inertial odometry. Publisher: arXiv Version Number: 1.
- [17] INSTRUMENTS, T. AWR2944 evaluation module for automotive second-generation, 76-ghz to 81-ghz, high-performance soc. <https://www.ti.com/tool/AWR2944EVM>.
- [18] INSTRUMENTS, T. IWR6843 intelligent mmwave sensor standard antenna plug-in module. <https://www.ti.com/tool/IWR6843ISK>.
- [19] INSTRUMENTS, T. Real-time data-capture adapter for radar sensing evaluation module. <https://www.ti.com/tool/DCA1000EVM>.
- [20] INSTRUMENTS, T. Single-chip 76-GHz to 81-GHz industrial radar sensor integrating DSP, MCU and radar accelerator. <https://www.ti.com/product/IWR1843>.
- [21] INSTRUMENTS, T. The fundamentals of millimeter wave radar sensors. <https://www.tij.co.jp/lit/wp/spyy005a/spyy005a.pdf>, 2020.
- [22] INTEL. Depth Camera D435. <https://www.intelrealsense.com/depth-camera-d435/>.
- [23] INTEL. Intel® realSense™ tracking camera t265. <https://www.intelrealsense.com/tracking-camera-t265/>.
- [24] IROBOT. Create 3 Robot. <https://edu.irobot.com/what-we-offer/create3>.
- [25] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. Publisher: arXiv Version Number: 9.
- [26] KRAMER, A., STAHOVIK, C., SANTAMARIA-NAVARRO, A., AGHA-MOHAMMADI, A.-A., AND HECKMAN, C. Radar-inertial ego-velocity estimation for visually degraded environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), IEEE, pp. 5739–5746.
- [27] KWON, S.-Y., KWAK, S., KIM, J., AND LEE, S. Radar Sensor-Based Ego-Motion Estimation and Indoor Environment Mapping. *IEEE Sensors Journal* 23, 14 (July 2023), 16020–16031.

- [28] LEE, W., ECKENHOFF, K., YANG, Y., GENEVA, P., AND HUANG, G. Visual-inertial wheel odometry with online calibration. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 4559–4566.
- [29] LI, Y., YANG, S., XIU, X., AND MIAO, Z. A spatiotemporal calibration algorithm for IMU–LiDAR navigation system based on similarity of motion trajectories. 7637.
- [30] LIANG, Y., MULLER, S., SCHWENDNER, D., ROLLE, D., GANESCH, D., AND SCHAFFER, I. A scalable framework for robust vehicle state estimation with a fusion of a low-cost IMU, the GNSS, radar, a camera and lidar. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), IEEE, pp. 1661–1668.
- [31] LIEN, J., GILLIAN, N., KARAGOZLER, M. E., AMIHOOD, P., SCHWESIG, C., OLSON, E., RAJA, H., AND POUPYREV, I. Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–19.
- [32] LIM, H., KIM, D., KIM, B., AND MYUNG, H. AdaLIO: Robust adaptive LiDAR-inertial odometry in degenerate indoor environments. Publisher: arXiv Version Number: 1.
- [33] LU, C. X., ROSA, S., ZHAO, P., WANG, B., CHEN, C., STANKOVIC, J. A., TRIGONI, N., AND MARKHAM, A. See through smoke: Robust indoor mapping with low-cost mmwave radar. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services* (New York, NY, USA, 2020), MobiSys '20, Association for Computing Machinery, p. 14–27.
- [34] LU, C. X., SAPUTRA, M. R. U., ZHAO, P., ALMALIOGLU, Y., DE GUSMAO, P. P. B., CHEN, C., SUN, K., TRIGONI, N., AND MARKHAM, A. milliEgo: single-chip mmWave radar aided egomotion estimation via deep sensor fusion. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems* (2020), ACM, pp. 109–122.
- [35] LV, J., ZUO, X., HU, K., XU, J., HUANG, G., AND LIU, Y. Observability-aware intrinsic and extrinsic calibration of LiDAR-IMU systems. 3734–3753.
- [36] MINODA, K., SCHILLING, F., WÜEST, V., FLOREANO, D., AND YAIRI, T. VIODE: A simulated dataset to address the challenges of visual-inertial odometry in dynamic environments. 1343–1350.
- [37] MONACO, C. D., AND BRENNAN, S. N. RADARODO: Ego-motion estimation from doppler and spatial data in RADAR images. 475–484. Conference Name: IEEE Transactions on Intelligent Vehicles.
- [38] MUR-ARTAL, R., MONTIEL, J. M. M., AND TARDOS, J. D. ORB-SLAM: A versatile and accurate monocular SLAM system. 1147–1163.
- [39] MUR-ARTAL, R., AND TARDOS, J. D. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-d cameras. 1255–1262.
- [40] NVIDIA. Jetson Orin for Next-Gen Robotics. <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/>.
- [41] PARK, Y. S., SHIN, Y.-S., AND KIM, A. PhaRaO: Direct radar odometry using phase correlation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 2617–2623. ISSN: 2577-087X.
- [42] PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., ANTIGA, L., DESMAISON, A., KÖPF, A., YANG, E., DeVITO, Z., RAISON, M., TEJANI, A., CHILAMKURTHY, S., STEINER, B., FANG, L., BAI, J., AND CHINTALA, S. PyTorch: An imperative style, high-performance deep learning library. Publisher: arXiv Version Number: 1.
- [43] PRABHAKARA, A., JIN, T., DAS, A., BHATT, G., KUMARI, L., SOLTANAGHAEI, E., BILMES, J., KUMAR, S., AND ROWE, A. High resolution point clouds from mmwave radar. arXiv <https://arxiv.org/abs/2206.09273>, 2022.
- [44] QIN, T., LI, P., AND SHEN, S. VINS-mono: A robust and versatile monocular visual-inertial state estimator. 1004–1020.
- [45] RONNEBERGER, O., FISCHER, P., AND BROX, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., vol. 9351. Springer International Publishing, 2015, pp. 234–241. Series Title: Lecture Notes in Computer Science.
- [46] SHAN, T., AND ENGLT, B. LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), IEEE, pp. 4758–4765.
- [47] SHAN, T., ENGLT, B., MEYERS, D., WANG, W., RATTI, C., AND RUS, D. LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), IEEE, pp. 5135–5142.
- [48] SIE, E., LIU, Z., AND VASISHT, D. Batmobility: Towards flying without seeing for autonomous drones. In *ACM International Conference on Mobile Computing (MobiCom)* (2023).
- [49] XIAO, Y., RUAN, X., CHAI, J., ZHANG, X., AND ZHU, X. Online IMU self-calibration for visual-inertial systems. 1624.
- [50] YANG, N., VON STUMBERG, L., WANG, R., AND CREMERS, D. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. Publisher: arXiv Version Number: 2.
- [51] YANG, Y., GENEVA, P., ZUO, X., AND HUANG, G. Online IMU intrinsic calibration: Is it necessary? In *Robotics: Science and Systems XVI*, Robotics: Science and Systems Foundation.
- [52] YANG, Y., GENEVA, P., ZUO, X., AND HUANG, G. Online self-calibration for visual-inertial navigation systems: Models, analysis and degeneracy. Publisher: arXiv Version Number: 3.
- [53] ZHANG, Z., AND SCARAMUZZA, D. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 7244–7251.
- [54] ZHU, F., REN, Y., AND ZHANG, F. Robust real-time LiDAR-inertial initialization. type: article.