RadNet: A Testbed for mmWave Radar Networks

Enver Bashirov
University of Padova
Padova, Italy
enver.bashirov@phd.unipd.it

Marco Canil
University of Padova
Padova, Italy
marco.canil@phd.unipd.it

Michele Rossi
University of Padova
Padova, Italy
michele.rossi@unipd.it

ABSTRACT

Human sensing with millimeter waves (mmWaves) is rapidly gaining momentum. In particular, mmWave radars are becoming the technology of choice in applications like contactless vital signs monitoring, people tracking, or activity recognition, when preserving the users privacy is a concern. However, single mmWave radar sensors have limited range (up to 6-8 m) and are affected by occlusions. For this reason, covering medium to large indoor spaces requires the deployment of multiple radar devices, i.e., *radar networks*. Because of the complexity of reflections produced by people moving in real life environments, the development and validation of algorithms for mmWave radar networks can only be fulfilled through extensive experimental campaigns. In this work, we present RadNet, the first experimental testbed for the easy deployment and testing of radar network algorithms. We describe its architecture and functioning and we show experimental results of a multi-radar people tracking algorithm implemented on the RadNet experimental platform.

CCS CONCEPTS

• **General and reference** → **Experimentation**; • **Hardware** → *Analysis and design of emerging devices and systems*; • **Networks** → Network experimentation.

KEYWORDS

mmWave, radar, radar network, edge computing, testbed

ACM Reference Format:
Enver Bashirov, Marco Canil, and Michele Rossi. 2022. RadNet: A Testbed for mmWave Radar Networks. In *Emerging Topics in Wireless (EmergingWireless '22), December 9, 2022, Roma, Italy*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3565474.3569068

1 INTRODUCTION AND RELATED WORKS

MmWave radars are emerging as valid alternatives to cameras for the pervasive monitoring of environments, as they combine highly accurate sensing capabilities with appealing properties of the mmWave spectrum such as insensitivity to extreme light conditions and to the presence of dust, smoke, or rain [1]. Furthermore, mmWave sensing preserves the users privacy, as no image of the subjects is captured [1]. Applications range from the estimation of human vital signs [2], to people tracking and identification [3], contact tracing [4], activity recognition [5], and many others. However, commercial mmWave radars have limited range (up to $6-8$ m) [6] and are impaired by occlusions [3], which may represent a significant drawback in crowded spaces or in the presence of cluttered furniture. For this reason, the deployment of multiple radars, i.e., *radar networks*, looks appealing to accurately monitor medium to large areas. Furthermore, the adoption of multiple sensors paves the way to a number of new sensing possibilities involving the fusion of radars data. While the first works regarding radar networks begin to appear [7, 8], in the literature there are still no experimental testbeds for the fast deployment and testing of radar network algorithms. Also, simulations have shown to be inapt for the development of human sensing algorithms, as reflections generated by humans moving in real life environments are often too complex to be accurately modeled. In [9], the authors propose a ray tracing model for radar signals propagation in radar networks, but they neither consider human targets nor the specific case of mmWaves reflections. In [10], a simulator to estimate the radar cross section (RCS) of people from computer animated human models is presented. The results are compared with measurements obtained from a real 2.4 GHz radar, but only providing a qualitative analysis. Also, the authors use a primitive-based approach which works well only at low frequencies, as proven in [11], where a similar simulator is proposed. In [12], a RCS of pedestrian simulator is proposed and evaluated using a 77 GHz radar achieving acceptable results in the estimation of range-time, doppler-time, and range-doppler maps. However, all these works are tested with only one person performing simple movements (e.g., walking on a straight line) and single-input single-output radars, while at least a linear array of antennas is required for most applications. Furthermore, these simulators fail to consider some of the crucial aspects of radar measurements in real life such as the presence of furniture or walls, introducing noise, clutter, and ghost targets; the presence of multiple people, producing additional reflections and occlusions; typical real life movements, that produce a wide variety of reflections that are usually not generated by simple movements. In conclusion, currently available simulators are not suitable to develop mmWave radar sensing algorithms that will work in practice. For this reason, experimental testbeds are fundamentally important.

The main contributions of this paper are:

(1) the design of RadNet, an experimental testbed for the easy deployment and testing of mmWave radar networks;
(2) the implementation of RadNet using commercial edge computers and mmWave radars;
(3) the implementation of a multi-radar people tracking algorithm, to demonstrate RadNet capabilities.

The remainder of this paper is organized as follows. Section 2 presents the RadNet architecture, its design/working principles

EmergingWireless '22, December 9, 2022, Roma, Italy

ACM ISBN 978-1-4503-9934-0/22/12…$15.00
https://doi.org/10.1145/3565474.3569068

and features. Section 3 formulates the multi-radar people tracking problem and provides the required preliminaries, while Section 4 illustrates the obtained results. Finally, concluding remarks are given in Section 5.

2 RADNET DESCRIPTION

2.1 Architecture

RadNet adopts a *server-client* architecture [13]. Each radar is directly connected to an *edge* computer (a client node) that is capable of performing local processing and each edge computer is connected via a network connection to a *central* node (a server). The central node controls edge nodes, and the edge nodes serve as distributed computational resources to process radars data. However, users do not communicate directly with neither the central node nor the edge nodes. The user interface is provided by an additional *driver* node, which is a transparent entity designed to act as an intermediary for passing control messages to the central node. So, all control actions are provided via the driver node. Also, the driver node can be instantiated and terminated from any machine connected to the network, allowing a flexible control of the system, see Fig. 1.

2.2 System Control

RadNet works by tasks that are selected by the user through the driver node and assigned to the edge nodes by the central node.

2.2.1 Control Messages. The management of tasks is carried out through a message-passing interface. The aim of the system is to distribute the workload among the edge nodes, while having a centralized unit managing each of them. The connection between nodes is established and maintained by an internal exchange of messages, which is detailed later in this section.

Currently, available driver commands are:

- *ping*: to measure the latency between central and edge nodes;
- *speed*: to measure the communication speed between central node and an edge node;
- *timesync*: to verify the time synchronization between central node and an edge node;
- *open*/*close*: to enable/disable edge connections to the system;
- *start* and *stop*: to start and stop radar processing at the edges.

Note that additional parameters can be specified for each command, providing more flexibility. Also, the framework allows extending the functionalities provided by the commands and adding new ones.

As previously mentioned, the system works by exchange of messages between the nodes. The driver node only communicates with the central node and the central node distributes the messages to the edge nodes. An overview of the overall system is presented in Fig. 1. The control unit manages the system and consists of central and driver nodes. The edge cluster consists of edge nodes providing distributed processing capabilities.

Both the edge nodes and the central node send and receive messages. A message consists of a header, an action, and data, if available. After the retrieval of a message, the receiver performs a certain task and responds with another message that can contain some results related to the current task or a simple acknowledgement. Communication between nodes is always kept alive by checking messages and responding.

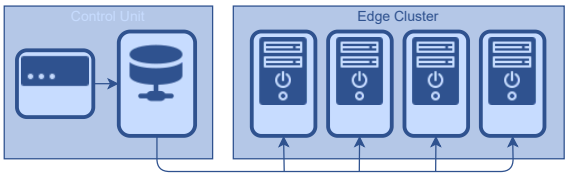

Figure 1: RadNet system overview.

Fig. 2 shows a typical exchange of messages from the system's startup, which will be thoroughly described next. At the beginning, the central node is initialized. Then, an edge tries to connect to the central node by sending a *join* message. Upon receiving the *join* message, the central node checks the message header and meta data which contains sender's information such as the IP address and the name of the edge node. Then, the central node proceeds to register the edge and stores its information so that, later on, it will be able to communicate with it. If the registration is completed successfully, a *welcome* response is sent to acknowledge the edge about the success of the join operation. At this point, the edge node becomes idle and communicates this via an *idle* message. During the idle period, the edge node stays in receive mode and waits for the assignment of a new action by the central node. In the example, a *start* message is sent to the driver and forwarded to an edge device. This command tells the edge to start some radar processing (e.g., to start performing people tracking). Upon receiving *start* message, the edge starts its local process and sends back a *starting* confirmation message, which is received from the central node. Then, as long as the task is not stopped, the central node keeps sending *continue* messages to the edge node, which replies with *processing* messages. Note that *processing* messages sent by an edge node may also contain some data which comes from its processing operations, such as, for example, tracking information or raw point cloud data from the radar, depending on the required task. When a *stop* command is issued by the driver, it is forwarded to the intended edge node by the central node and the latter stops its processing operations and acknowledges the reception of the command by answering, in turn, with a *stop* confirmation message. Afterwards, the edge node goes into an idle state.

2.2.2 Parallelization and Data Management. Since applications based on radar networks often have to work in an online fashion, delays and latencies must be avoided or reduced as much as possible. For this purpose, RadNet is developed using multi-threading. All system modules which can work independently are deployed in a separate thread. This allows communication tasks and radar processes to continue in parallel, without interfering with one another. Also, data management is implemented in parallel to other activities. At an edge node, after retrieving and processing data from the radar, these data are placed in a temporary buffer. During processing, the communication module keeps checking the temporary buffer and, as soon as some data are available, they are sent to the central node. Similarly, at the central node, once data are received, a separate thread to save them is started, so that the system can keep working in parallel. Besides being sent, data can be stored locally at the edges or at the central node, or they can be consumed on-the-fly, without being saved.

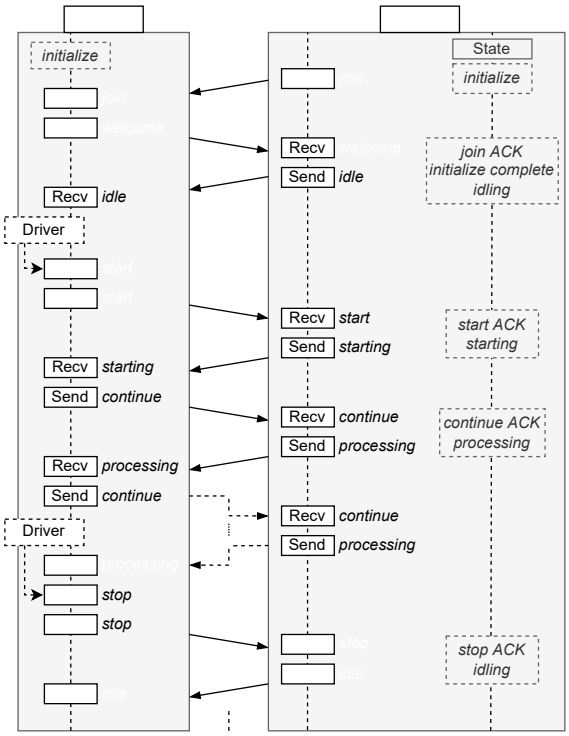

Figure 2: Example of communication between central and edge nodes. Send, Recv and ACK annotations in the figure correspond to sending, receiving and acknowledgement events, respectively.

2.3 Implementation Details

The backbone of RadNet framework is written in Python using socket programming and multi-threading. The communication between the nodes in RadNet is carried out via Transmission Control Protocol and Internet Protocol (TCP/IP) [14] as TCP allows data sequencing, guarantees the delivery of the transmissions and provides error checking. Thus, it is reliable and allows a simpler system implementation. In the future, we are also planning to allow users to choose between TCP and User Datagram Protocol (UDP) to send the data, as the latter may provide a faster data stream in certain situations. A dedicated (user defined) error correction protocol for data and commands may have to be implemented in this case. As already mentioned, RadNet has been designed to be modular, so that new functionalities can be added easily.

3 APPLICATION EXAMPLE

In order to test RadNet on a real application, we implemented a multi-radar multi-target tracking system on it and we used RadNet to evaluate its performance.

In the system, each radar is connected via USB to an *edge computer* (corresponding to a RadNet edge node) that performs a local target tracking, while all edge computers are connected via Ethernet to a *Fusion Center* (FC, that is instantiated on the central node). The FC receives tracking information from the edges at regular time intervals and uses them to produce a central unified fused tracking. For the fusion, the radars relative position and orientation is required to refer all data to the same reference system. Here, we assume to have them and, in practice, we retrieve them from ground truth (GT) data. Methods for their automatic estimation can be found in [7, 15].

3.1 Preliminaries on mmWave indoor radar

A Multiple-Input Multiple-Output (MIMO) Frequency-Modulated Continuous Wave (FMCW) radar jointly estimates the distance, the radial velocity, and the angular position of the targets with respect to the radar device [16]. To sense the environment, sequences of linear chirp signals with bandwidth B are sent by the radar. A full sequence is named "radar frame" and is repeated every T seconds, which is the period.

3.1.1 Distance, velocity and angle estimation. The targets distance, r, and velocity, v, are computed from the frequency shift induced by the delay of the reflections, usually applying discrete Fourier transform (DFT) processing. The range resolution of a FMCW radar is related to its bandwidth B by $\Delta r = c/(2B)$, where c is the speed of light. Thus, with a bandwidth of $2-4$ GHz, mmWave devices reach a resolution of a few centimeters [4, 3]. Moreover, a 2D array of multiple receiving antennas makes it possible to obtain the angle-of-arrival (AoA) of the reflections along the azimuth (θ) and elevation (ϕ) dimensions, by exploiting the phase shifts across different antenna elements. The azimuthal resolution depends on the number of antennas N as $\Delta\theta = \lambda/(Nd\cos\theta)$, where d represents the spacing between the antennas.

3.1.2 Radar point-clouds. The presence of a human in the environment generates a large number of reflections detected by the radar. These produce a set of points, termed *radar point-cloud*, that can be transformed into the 3-dimensional Cartesian space using the information about the distance, the azimuth, and the elevation angle. Each point is described by vector $[x, y, z]^T$, including its spatial coordinates x, y, z obtained transforming r, θ and ϕ.

3.1.3 People tracking. The common approach to people tracking from mmWave radar point-clouds [6, 17, 3] includes *(i)* a detection phase via density-based clustering algorithms (e.g., DBSCAN [18]) to separate the reflections from multiple subjects, and *(ii)* applying Kalman filtering (KF) techniques [19] on each cluster centroid to track the movement trajectory of each subject in space. In this work, we estimate the subjects' trajectories in the (x, y) horizontal plane. The *state* of each subject at time k is defined as $\mathbf{x}_k = [x_k, y_k, \dot{x}_k, \dot{y}_k]^T$, where $\dot{x}_k$ and $\dot{y}_k$ are the velocity components along the two axes. Also, we assume that the state evolves as $\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1}$, where the transition matrix $\mathbf{F}$ represents a constant-velocity model [20]. Considering radar sensor s and target n, we denote by $\hat{\mathbf{x}}_k^s(n)$ and $\mathbf{C}_k^s(n)$, respectively, the KF state estimate and the corresponding 4×4 error *covariance matrix* computed at time k. A track $\mathcal{T}_k^s(n) = \{\hat{\mathbf{x}}_k^s(n), \mathbf{C}_k^s(n)\}$ from sensor s is a pair composed of a state estimate and its covariance obtained at time k. $\mathbf{P}_k^s(n) = (\mathbf{C}_k^s(n))^{-1}$ denotes the *precision matrix* corresponding to the inverse of the covariance matrix $\mathbf{C}_k^s(n)$. In the following, we will use subscript "$k|m$" to denote the same quantities computed at time k using observations up to time $m \leq k$.

3.2 Data Fusion

The FC operates in a time-slotted fashion: it considers subsequent time windows and, inside each window, it only processes the information produced during that particular time interval. The behavior of the track fusion algorithm differs in case it has to combine two sensor tracks (SS fusion) or one sensor track and a so-called central track (SC fusion), as we now explain. Considering a specific subject n, if the FC is currently not maintaining a track for it, but one or more radars are, a new central track has to be initialized based on the received information from the local sensors. Two cases may happen: *(i)* if n is tracked by one sensor only, then, the corresponding central track is initialized using its state and covariance; *(ii)* if n is tracked by more than one sensor, let us say, sensor 1 and sensor 2, then, their tracks are fused into a single central track (SS fusion). In case *(ii)*, the fusion equations are

$$\mathbf{C}^c_{0|0}(n) = \left[\mathbf{P}^1_0(n) + \mathbf{P}^2_0(n)\right]^{-1}, \tag{1}$$

$$\hat{\mathbf{x}}^c_{0|0}(n) = \mathbf{C}^c_{0|0}(n)\left[\mathbf{P}^1_0(n)\hat{\mathbf{x}}^1_0(n) + \mathbf{P}^2_0(n)\hat{\mathbf{x}}^2_0(n)\right], \tag{2}$$

where, here and in the rest of the paragraph, superscript c denotes quantities that refer to the FC. This fusion rule is typically used when the errors at the sensors are uncorrelated or the correlation can be neglected [21]. Note that, in case more than two radars are tracking the same target, the above process is sequentially repeated, firstly using sensors 1 and 2 to produce a central track, and then fusing the resulting track with the information from sensor 3 and so on until the information from all sensors (radars) is used.

If, instead, the FC has already initialized the track for a subject, the fusion has to be performed between the central track and a sensor track corresponding to the same subject. Denote by $\mathcal{T}^c_m$ and $\mathcal{T}^s_m$ the set of tracks maintained by the FC and by sensor s, respectively, at the central time step m. Upon receiving the local information $\mathcal{T}^s_m$, the FC runs a track-to-track association algorithm to find pairs of corresponding tracks $\{\mathcal{T}^c_m(n), \mathcal{T}^s_m(n)\}$. Once such pairs are available, each central track is updated with its corresponding sensor track using the *information decorrelation* method [22, 21] as follows

$$\mathbf{C}^c_{m|m}(n) = \left[\mathbf{P}^c_{m|m-1}(n) + \mathbf{P}^s_m(n) - \bar{\mathbf{P}}^s(n)\right]^{-1}, \tag{3}$$

$$\begin{aligned}\hat{\mathbf{x}}^c_{m|m}(n) = \mathbf{C}^c_{m|m}(n)\Big[&\mathbf{P}^c_{m|m-1}(n)\hat{\mathbf{x}}^c_{m|m-1}(n) + \\ &+\mathbf{P}^s_m(n)\hat{\mathbf{x}}^s_m(n) - \bar{\mathbf{P}}^s(n)\bar{\mathbf{x}}^s(n)\Big],\end{aligned} \tag{4}$$

where $\bar{\mathbf{P}}^s(n)$ and $\bar{\mathbf{x}}^s(n)$ are the last communicated precision matrix and state estimate of track n from sensor s to the FC. Information decorrelation copes with the problem of correlated tracks between the FC and the radar sensors by removing the most recently received information about target n, as, otherwise, this would be accounted for twice by just summing $\mathbf{P}^c_{m|m-1}(u)$ and $\mathbf{P}^s_m(n)$. For more details about the fusion algorithm, including those regarding the time-slotted information processing and the tracks association, please refer to [8].

4 EXPERIMENTAL RESULTS

We tested RadNet using 4× IWR1843BOOST Texas Instruments mmWave radars[1] operating in the 77-81 GHz band in real-time at a frame rate of $1/T = 15$ Hz with a Field of View (FoV) of ±60° and ±15° over the azimuth and elevation planes, respectively. Each radar was connected via USB to an edge computer. As edge computers, we used 1× Jetson TX2 DevKit[2], 1× Jetson Nano DevKit[3], and 2× Jetson Xavier NX DevKit[4].

Table 1: Summary of tracking algorithm performance

Median error	1 radar	2 radars	3 radars	4 radars
1T-MOTA	96%	97%	96%	94%
1T-MOTP [m]	0.08	0.12	0.15	0.15
2T-MOTA	74%	77%	85%	86%
2T-MOTP [m]	0.10	0.29	0.26	0.22

4.1 Measurement setup and Dataset

To assess the performance of the people tracking algorithm implemented using RadNet, we conducted tests in a 7×4 m research laboratory (see Fig. 3a) equipped with a motion tracking system featuring 10 cameras. This provides the GT 3D localization of a set of markers placed on the radars and on the subjects with millimeter-level accuracy. We acquired 5 sequences with 1 target (1T) and 30 sequences with 2 targets (2T). Each sequence is 40 seconds long. People either moved according to predefined trajectories or moved freely, as depicted in Fig. 3. For single target sequences, only free movement was considered. Subjects were tracked by 4 radars simultaneously, deployed in 2 different setups. Then, tracking information have been fused in different ways to evaluate distinct scenarios. In particular, all possible combinations of 1 to 4 radars have been considered for the fusion, resulting in $\binom{4}{1} + \binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 15$ different experiments per sequence. Thus, a total of $35 \times 15 = 525$ experiments have been analyzed.

4.2 Evaluation metrics

To assess the tracking performance, we adopt the *Multiple Object Tracking Performance Accuracy* (MOTA) metric, which accounts for the number of misses, false positives, and switches in the object detections, and the *Multiple Object Tracking Performance Precision* (MOTP) metric, which represents the mean position error by considering only correctly tracked objects. More details about these metrics can be found in [23].

4.3 Tracking performance

Tab. 1 summarizes the median tracking performance, across all the configurations, see Fig. 3. Columns identify the various fusion cases: when only 1 radar is used, i.e., there is no fusion; when i radars are used, $i > 1$, it means that their tracks have been fused and the results refer to the fused track. Considering the 1T case, there is almost no difference between using a single radar or multiple radars. This result was expected, since the tracking of a single person in an empty room is a relatively simple task, as no occlusions can occur to impact the quality of radar tracks. A slight worsening of MOTA and MOTP is observed as the number of radars increases due to the fact that single radar target state estimations are not perfect and

[1] https://www.ti.com/tool/IWR1843BOOST

[2] https://developer.nvidia.com/embedded/buy/jetson-tx2-developer-kit

[3] https://developer.nvidia.com/embedded/buy/jetson-nano-devkit

[4] https://developer.nvidia.com/embedded/jetson-xavier-nx-devkit

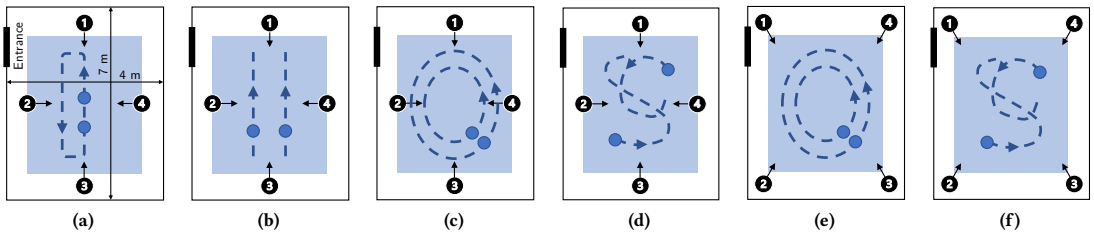

Figure 3: Illustration of the setups and trajectories used to test RadNet. The numbered dots represent the radar devices, with the arrow identifying their pointing directions. The blue dots represent the subjects, while the dashed blue lines represent subjects' trajectories. Fig. 3d and Fig. 3f represent free movement.

during the fusion process this can result in a small increment in the estimation error (the estimation noise is also fused). This is more evident in the MOTP values that go from 0.08 m to 0.15 m, which is still a small error. One may argue that, given these results, in the 1T case the use of only one radar is to be preferred with respect to more radars. This is reasonable for the very simple case of one target moving in a quite restricted area. If, instead, the target moves in a larger environment, the use of multiple radars is required to be able to track it in the entire space, as single sensors range is limited to a few meters. Considering the 2T case, it is possible to observe a clear increase in the MOTA as more radars are used for the tracking, going from 74% with 1 radar to 86% with 4 radars, providing an improvement of 12%. Situations with 2 targets are way more challenging, as they can occlude each other, possibly severely limiting the tracking capabilities of a single radar device. MOTP values remains always within a median value of 0.26 m, which is in line with state-of-the-art results [4].

5 CONCLUSIONS

In this paper we presented RadNet, the first testbed to facilitate the deployment and testing of algorithms for mmWave *radar networks*. We described its architecture and working principles, and implemented a first version of it using commercial mmWave radars and edge computers. As an example, we also deployed a distributed multi-radar multi-target people tracking algorithm, showing the improvement that exploiting multiple radars can bring, in terms of tracking accuracy (+12%).

ACKNOWLEDGMENTS

This work received support from the European Commission's Horizon 2020 Framework Programme under the Marie Skłodowska-Curie Action MINTS (GA no. 861222). We thank Anish Shastri for his help in the realization of the experiments with RadNet.

REFERENCES

[1] Syed Aziz Shah and Francesco Fioranelli. 2019. Rf sensing technologies for assisted daily living in healthcare: a comprehensive review. *IEEE Aerosp. Electron. Syst. Mag.*, 34, 11, 26–44.

[2] Giacomo Paterniani, Daria Sgreccia, Alessandro Davoli, Giorgio Guerzoni, Pasquale Di Viesti, Anna Chiara Valenti, Marco Vitolo, Giorgio Matteo Vitetta, and Giuseppe Boriani. 2022. Radar-based Monitoring of Vital Signs: A Tutorial Overview, (Mar. 2022). https://www.techrxiv.org/articles/preprint/Radar-based_Monitoring_of_Vital_Signs_A_Tutorial_Overview/19212918.

[3] Jacopo Pegoraro and Michele Rossi. 2021. Real-time people tracking and identification from sparse mm-wave radar point-clouds. *IEEE Access 9*, 78504–78520.

[4] Marco Canil, Jacopo Pegoraro, and Michele Rossi. 2022. Millitrace-ir: contact tracing and temperature screening via mmwave and infrared sensing. *IEEE J. Sel. Top*, 16, 2, 208–223.

[5] Akash Deep Singh, Sandeep Singh Sandha, Luis Garcia, and Mani Srivastava. 2019. Radhar: human activity recognition from point clouds generated through a millimeter-wave radar. In *Proc. ACM mmNets*. Los Cabos, Mexico, 51–56.

[6] Peijun Zhao, Chris Xiaoxuan Lu, Jianan Wang, Changhao Chen, Wei Wang, Niki Trigoni, and Andrew Markham. 2019. Mid: tracking and identifying people with millimeter wave radar. In *Proc. of IEEE DCOSS*. Santorini, Greece, 33–40.

[7] Anish Shastri, Marco Canil, Jacopo Pegoraro, Paolo Casari, and Michele Rossi. 2022. Mmscale: self-calibration of mmwave radar networks from human movement trajectories. In *Proc. of IEEE RadarConf22*. New York, USA, 1–6.

[8] Jacopo Pegoraro, Marco Canil, Anish Shastri, Paolo Casari, and Michele Rossi. 2022. Oracle: occlusion-resilient and self-calibrating mmwave radar network for people tracking. (2022). https://arxiv.org/abs/2208.14199.

[9] Demetrio Gubelli, Oleg A. Krasnov, and Olexander Yarovyi. 2013. Ray-tracing simulator for radar signals propagation in radar networks. In *Proc. of EuRAD*. Nuremberg, Germany, 73–76.

[10] Shobha Sundar Ram and Hao Ling. 2008. Simulation of human microdopplers using computer animation data. In *Proc. of IEEE RadarConf*. Rome, Italy, 1–6.

[11] Akash Deep Singh, Shobha Sundar Ram, and Shelly Vishwakarma. 2018. Simulation of the radar cross-section of dynamic human motions using virtual reality data and ray tracing. In *Proc. of IEEE RadarConf*. Oklahoma City, USA, 1555–1560.

[12] Yoshana Deep, Patrick Held, Shobha Sundar Ram, Dagmar Steinhauser, Anshu Gupta, Frank Gruson, Andreas Koch, and Anirban Roy. 2020. Radar cross-sections of pedestrians at automotive radar frequencies using ray tracing and point scatterer modelling. *IET Radar, Sonar Navig.*, 14, (June 2020), 833–844(11), 6, (June 2020).

[13] Alex Berson. 1996. *Client/Server Architecture (2nd Ed.)* McGraw-Hill, Inc., USA. ISBN: 0070056641.

[14] Behrouz A. Forouzan and Sophia Chung Fegan. 2002. *TCP/IP Protocol Suite*. (2nd ed.). McGraw-Hill Higher Education. ISBN: 0072460601.

[15] Anish Shastri, Neharika Valecha, Enver Bashirov, Harsh Tataria, Michael Lentmaier, Fredrik Tufvesson, Michele Rossi, and Paolo Casari. 2022. A review of millimeter wave device-based localization and device-free sensing technologies and applications. *IEEE Commun. Surveys Tuts.*, 24, 3, 1708–1749.

[16] Sujeet Milind Patole, Murat Torlak, Dan Wang, and Murtaza Ali. 2017. Automotive radars: a review of signal processing techniques. *IEEE Signal Process. Mag.*, 34, 2, 22–35.

[17] Zhen Meng, Song Fu, Jie Yan, Hongyuan Liang, Anfu Zhou, Shilin Zhu, Huadong Ma, Jianhua Liu, and Ning Yang. 2020. Gait recognition for co-existing multiple people using millimeter wave sensing. In number 01. New York, USA, (Apr. 2020), 849–856.

[18] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of AAAI KDD-96*. Portland, Oregon, 226–231.

[19] R. E. Kalman. 1960. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Eng.*, 82, 1, (Mar. 1960), 35–45.

[20] Thomas Wagner, Reinhard Feger, and Andreas Stelzer. 2017. Radar signal processing for jointly estimating tracks and micro-doppler signatures. *IEEE Access 5*, 1220–1238.

[21] Chee-Yee Chong, S. Mori, W.H. Barker, and Kuo-Chu Chang. 2000. Architectures and algorithms for track association and fusion. *IEEE Aerosp. Electron. Syst. Mag.*, 15, 1, 5–13.

[22] K.C. Chang, R.K. Saha, and Y. Bar-Shalom. 1997. On optimal track-to-track fusion. *IEEE Trans. Aerosp. Electron. Syst.*, 33, 4, 1271–1276.

[23] Keni Bernardin, Alexander Elbs, and Rainer Stiefelhagen. 2006. Multiple object tracking performance metrics and evaluation in a smart room environment. In *Proc. of IEEE Int. Wkshp. on Vis. Surveillance* number 91. Vol. 90.