A Blender-based channel simulator for FMCW Radar

Yuan Liu, Moein AHMADI, Johann Fuchs, Mohammad Alaee-Kerahroodi, M. R. Bhavani Shankar,
Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, L-1855, Luxembourg
Email: {yuan.liu, moein.ahmadi, johann.fuchs, mohammad.alaee, bhavani.shankar }@uni.lu

***Abstract*—Radar simulation is a promising way to provide data-cube with effectiveness and accuracy for AI-based approaches to radar applications. This paper develops a channel simulator to generate frequency-modulated continuous-wave (FMCW) waveform multiple inputs multiple outputs (MIMO) radar signals. In the proposed simulation framework, an open-source animation tool called Blender is utilized to model the scenarios and render animations. The ray tracing (RT) engine embedded can trace the radar propagation paths, i.e., the distance and signal strength of each path. The beat signal models of time division multiplexing (TDM)-MIMO are adapted to RT outputs. Finally, the environment-based models are simulated to show the validation.**

***Index Terms*—Blender, channel simulation, FMCW radar, indoor pedestrian, ray tracing.**

I. Introduction

Radio-based sensing systems have long and widely been used for various radar applications. A basis for designing and optimizing sensor systems is the knowledge of radar channel characteristics [1]. Conventionally channel models can be obtained by field measurements. However, measurement can be time-consuming and also expensive[2]. The deterministic ray tracing (RT) [3, 4] has been used in wireless communication simulations. Following the trends, recent studies utilize the RT tool embedded in the animation software, e.g., Blender and Optix, for radar channel simulation [5]. The Blender and the OptiX tools can model the dynamic scenarios and then render each frame of the animation, therefore capturing any slight motions of the target.

However, there exist gaps in the state of the art. On the one hand, many rendering-based radar simulation works are for outdoors because they are motivated by the automatic driving industry, e.g., the OptiX-based ones [6] and commercial simulator FEKO [7]. On the other hand, most of them are short of considering dynamic simulation of the targets [5, 6, 8]. In this paper, a Blender-based mmWave MIMO radar simulator is developed for indoor applications built on the free and open-source Blender animation software.

II. Simulation Outline and Signal model

The overall procedure of producing the FMCW radar signals using the RT tool embedded in Blender is illustrated in Fig.1, which mainly consists of four steps, including scenario modeling, image rendering, signal generation, and target estimation.

In scenario modeling, the tool is flexible to define the material properties, geometry size, and moving trajectories of the objects in the scenario by various means, e.g., the animation tools in Blender, add-on custom, and Python scripts via open interfaces. The positions of the light source, camera, and camera angle ranges are in line with the positions of the transmitter (Tx), receiver (Rx) antennas, and antenna patterns, respectively. By the rendering engine chosen, e.g., the Cycles engine, each frame of the dynamic animation is rendered into a 2D picture, with each picture represented by a certain number of pixels, where the distance, strength, and angle of arrival (AoA) of each pixel can be obtained either directly by Blender or calculation. The frame rate and the number of pixels are defined manually. Using the Blender outputs, we can generate FMCW radar signals.

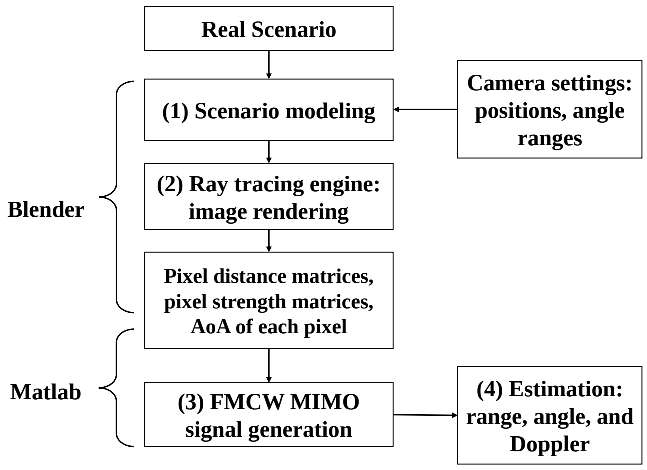

Real Scenario
(1) Scenario modeling
Camera settings: positions, angle ranges
(2) Ray tracing engine: image rendering
Blender
Pixel distance matrices, pixel strength matrices, AoA of each pixel
Matlab
(3) FMCW MIMO signal generation
(4) Estimation: range, angle, and Doppler

Figure 1. Outline of the simulation procedure.

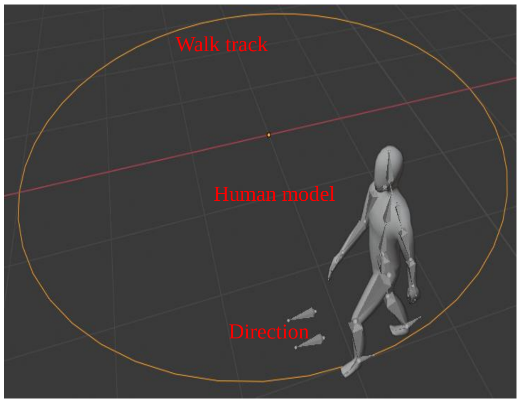
Considering a M-Tx and N-Rx radar array. The beat signal of time division multiplexing (TDM)-MIMO in practice is [9]

$$\mathbf{h}_{T,n}(n_s;l) = \sum_{m=1}^{M} \sigma_{m,n,l} \exp(j2\pi(\frac{2\mu R_{m,n}}{c}\frac{n_s-1}{F_s} - f_D(l-1)T_b)), \tag{1}$$

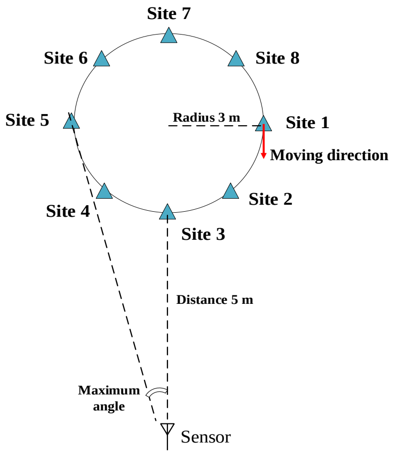
where $m = 1, 2, ..., M$ is the indices of Tx antennas, $n = 1, 2, ..., N$ is the indices of Rx antennas, $l = 1, 2, ..., L$ is the indices of TDM block, where one TDM block contains M orthogonal FMCW chirps, $\sigma_{m,n,l}$ contains the attenuation of the RCS of the target and the propagation channel between the mth Tx and the nth Rx at the lth TDM block, μ is the slope of FMCW chirp, $R_{m,n}$ is the range, c is the propagating velocity of the light, $n_s = 1, 2, ..., N_s$ is the indices of fast time, F_s is the sampling frequency, f_D is the Doppler frequency shift, and T_b is the chirp duration.

In Blender, a 3D scene is rendered into a 2D image, represented by $N_{\text{az}} \times N_{\text{el}}$ pixels. Each pixel can be regarded as one target, with the range and strength information being the rendering outputs of Blender. Based on (1) the beat signal of the nth Rx used in the simulation can be represented by $\mathbf{H}_{T_n} \in \mathbb{C}^{L\times N_s}$, where L and N_s denote the slow time and fast time axis, respectively, and the (l, n_s)-th entry can be expressed as

$$\begin{aligned} H_{T_n\, l,n_s} = \sum_{n_{\text{az}}=1}^{N_{\text{az}}} \sum_{n_{\text{el}}=1}^{N_{el}} \sum_{m=1}^{M} P_{r_{n_{\text{frame}},n_{\text{el}},n_{\text{az}}}} \exp(j2\pi(\\ \frac{2\mu R_{n_{\text{frame}},n_{\text{el}},n_{\text{az}}}}{c}\frac{n_s-1}{F_s} - 2\frac{f_l V_{n_{\text{fame}},n_{\text{el}},n_{az}}}{c}\frac{l-1}{L}T_b \\ + \frac{((m-1)N+n-1)\Delta d}{\lambda}\sin\theta_{n_{\text{el}},n_{\text{az}}})), \end{aligned} \tag{2}$$



(a) Model of a human walks in a circle track.



(b) Illustration of location of the sensor and pedestrian track.

Figure 2. Scenario.

where $n_{\text{az}} = 1, 2, ..., N_{\text{az}}$ is the indices of pixels at the azimuth axis, $n_{\text{el}} = 1, 2, ..., N_{\text{el}}$ is the indices of pixels at the elevation axis, n_{frame} is the indices of Blender frames, $R_{n_{\text{frame}}}$ is the rendering results of range matrix, Blender outputs $R_{n_{\text{frame}}, n_{\text{el}}, n_{\text{az}}}$, $P_{r_{n_{\text{el}}, n_{\text{az}}}}$, $\Theta_{n_{\text{el}}, n_{\text{az}}}$, and $V_{n_{\text{fame}}, n_{\text{el}}, n_{az}}$ denote the range, strength, angle of arrivals and velocity, respectively, $\Delta d = \lambda/2$ is the interval between Rx antennas and λ is the wavelength.

III. Simulation and Discussion

Utilizing a walking scenario as shown in Fig. 2 (a) for simulation. The detailed geometry information is illustrated in Fig. 2 (b). A human walks circularly at a constant speed and the radius of the circle is 3 meters. To observe the obvious change in the velocity, the walking speed may not be realistic in indoor scenarios, where the radius of the circle is 3 meters and the number of animation frames is set to be 100 with a frame rate of 24 Hz. Hence the walking speed along the tangent of the circle is

$$S_{\text{walk}} = \arctan \frac{2 \times 3 \times \pi}{100 \times \dfrac{1}{24}} = 4.52 \quad m/s. \tag{3}$$

In the simulation, the walking direction is not towards the AoA direction, therefore the detected velocity of the main human body is less than the S_{walk}.

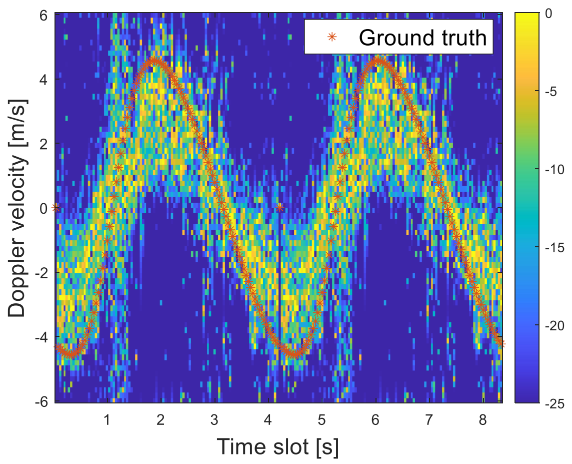


Figure 3. Doppler time plot: the envelope of the Doppler velocity changes like a sinusoidal signal is illustrated, which shows the consistency with the ground truth, i.e., the red stars in the figure.

Utilizing the FFT on the simulated data cube of each CPI, we can obtain the range-Doppler map. Collecting the velocity spectrum with the maximum power of each CPI, the concatenated Doppler-time results are obtained in Fig.3, where the continuous changing of Doppler velocities alone with time is observed in Fig 3.

The mean error of the estimated velocity by simulation and the ground truth is calculated as

$$E_{\text{velo}} = \frac{\Sigma_{i=1}^{N} \mid v_{\text{est}} - v_{\text{tru}} \mid}{N_p} = 1.3 \quad m/s, \tag{4}$$

where N is the number of CPIs, i is the index of the CPI, v_{tru} is the ground truth, and v_{est} is the simulated velocity corresponding to the maximum value of each range-velocity spectral.

Acknowledgment

This work was supported by the Luxembourg National Research Fund (FNR) through the BRIDGES project MASTERS under grant BRIDGES2020/IS/15407066.

References

[1] X. Yin and X. Cheng, *Propagation channel characterization, parameter estimation, and modeling for wireless communications*. John Wiley & Sons, 2016.

[2] Y. Liu, X. Yin, X. Ye, Y. He, and J. Lee, "Embedded propagation graph model for reflection and scattering and its millimeter-wave measurement-based evaluation," *IEEE Open Journal of Antennas and Propagation*, vol. 2, pp. 191–202, 2021.

[3] V. Degli-Esposti, F. Fuschini, E. M. Vitucci, M. Barbiroli, M. Zoli, L. Tian, X. Yin, D. A. Dupleich, R. $Mller$, C. Schneider, and R. S. $Thom$, "Ray-tracing-based mm-wave beamforming assessment," *IEEE Access*, vol. 2, pp. 1314–1325, 2014.

[4] D. He, B. Ai, K. Guan, L. Wang, Z. Zhong, and T. $Krner$, "The design and applications of high-performance ray-tracing simulation platform for 5g and beyond wireless communications: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 10–27, 2019.

[5] M. Ouza, M. Ulrich, and B. Yang, "A simple radar simulation tool for 3d objects based on blender," in *2017 18th International Radar Symposium (IRS)*, 2017, pp. 1–10.

[6] C. Schubler, M. Hoffmann, J. Braunig, I. Ullmann, R. Ebelt, and M. Vossiek, “A realistic radar ray tracing simulator for large mimo-arrays in automotive environments,” *IEEE Journal of Microwaves*, vol. 1, no. 4, pp. 962–974, 2021.

[7] A. F. Applications. Altair only forward. [Online]. Available: https://www.altair.com/

[8] R. F. da Costa, D. d. S. de Medeiros, R. Andrade, O. Saotome, and R. Machado, “General purpose radar simulator based on blender cycles path tracer,” in *XXXVIII Simposio Brasileiro de Telecomunicacoes de Processamento de Sinais (SBrT2020)*, 2020.

[9] R. Amar, M. Alaee-Kerahroodi, P. Babu, and B. S. M. R., “Designing interference-immune doppler-tolerant waveforms for radar systems,” *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–20, 2022.