Lookahead Ray Tracing Assisted Environmental Awareness in Beam Management: A High-Level Synthesis Approach

Jintong An
Chair of Embedded Systems
TU Dortmund University
Dortmund, Germany
jintong.an@tu-dortmund.de

Selma Saidi
Institute of Computer and Network Engineering
Technical University of Braunschweig
Braunschweig, Germany
saidi@ida.ing.tu-bs.de

Karsten Heimann
Communication Networks Institute
TU Dortmund University
Dortmund, Germany
karsten.heimann@tu-dortmund.de

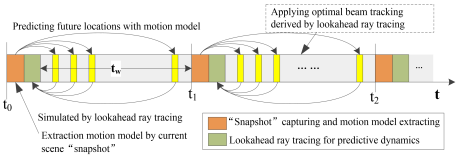
Christian Wietfeld
Communication Networks Institute
TU Dortmund University
Dortmund, Germany
christian.wietfeld@tu-dortmund.de

***Abstract*—In this paper, a high-level synthesis workflow for FPGA-accelerated lookahead ray tracing simulation is proposed to explore the impact of environmental dynamics on future beam management. Corresponding modules have been developed to accelerate massive simulations in lookahead ray tracing: A pipelined structure for computing ray reflection in signal propagation is designed, which is followed by ray validation implemented by parallelized modules. The proposed method is validated in digital twin of a real scene, presenting improved real-time performance handling dynamic environment with lower angular error in beam steering, which shows enhanced capability in environmental awareness for robust beam management.**

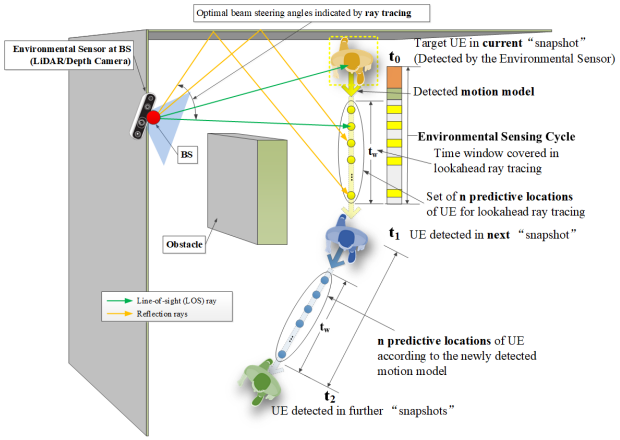
***Keywords*—ray-tracing, FPGA, real-time, environmental awareness**

I. Introduction

In millimeter wave (mmWave) wireless communications for Internet of things (IoT) devices, the dynamics of user equipment (UEs) and obstacles in the environment can result in instability or outage of the directional mmWave links, which is common in logistics application scenarios for IoT. One example of a relevant scenario is the obstruction of the link between the base station (BS) and moving automated guided vehicles (AGV) by rack shelves as obstacles in a warehouse environment (see Section IV). This causes problems with tracking beams and predicting obstacles, which is a topic that has been extensively discussed in the literature. In [1–3], the authors presented methodologies for estimating beam angles of radio frequency (RF) signals through the Kalman filters, with the objective of achieving precise beam tracking for UEs. In [4–6],



(a) Timing of environmental awareness



(b) Example scene with UE moving through obstacle

Fig. 1. Example of lookahead ray tracing in indoor scene

the authors proposed to use radar for beam tracking for human objects in indoor scenarios. Compared to complex processing in radar and RF signals, the use of

images to understand the signal propagation in the environment is more intuitive, allowing the system to clearly identify targets and generate adapted beam management schemes, indicating the concept of **environmental awareness** in this paper. In [7], the authors employed deep learning networks to process images captured by cameras with the objective of identifying dynamic obstacles and UEs. These were then utilized to make predictions regarding beam blockage, achieving up to 88% accuracy. Similarly, in [8], the authors leveraged the image data captured by the camera to anticipate the future movement of obstacles with the assistance of a deep learning network, thereby enhancing the environmental awareness of the target scene. Moreover, the authors of [9] proposed to correlate the received signal strength with the obstacle of UEs detected in the image, which resulted in a better environmental awareness with an 11.5% improvement in tracking accuracy. Similarly, [10–12] also discussed the methods of enhancing environmental awareness for communication systems using scene images. While these approaches result in high pre-

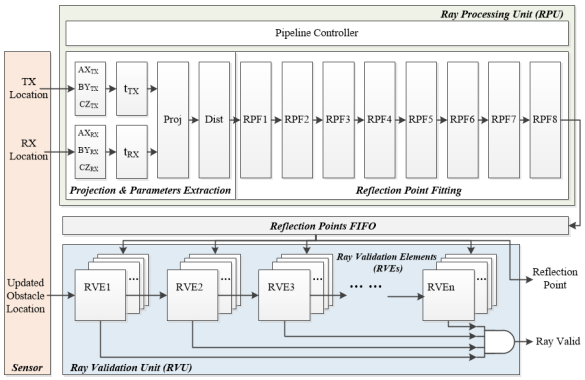


Fig. 2. Module architecture of LRT on FPGA

diction accuracy in beam tracking and/or link blockage, the high training overhead and runtime computational complexity of the neural networks should be further optimized for the **real-time** application targeted in this paper. In order to improve the real-time performance of environmental awareness for dynamic scenes, [13, 14] proposed a real-time simulation of signal propagation with the help of ray-tracing accelerated by FPGAs, which is used to obtain the impact of signal propagation on the current "snapshot" captured by the environmental sensors (e.g., LiDAR or depth camera). Nevertheless, as the environment under study changes over time, a single "snapshot" is no longer sufficient for robust beam management. This necessitates that the ray tracing system explore "future" impact factors to be integrated into the present beamforming decisions, thereby ensuring that the beam tracking scheme adequately matches the time-varying dynamics of the environment. In this paper, model predictive control (MPC) concept, which have gained widespread popularity in the field of robotics, are integrated with real-time ray-tracing methodologies. This integration enables the construction of a novel Lookahead Ray Tracing (LRT) approach, which is proposed in this paper with the objective of enhancing the robustness of beam management through the exploration of predictive dynamics in ray tracing. However, the inte-

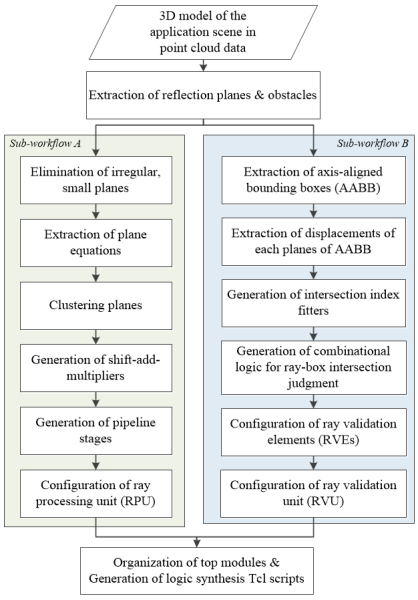


Fig. 3. High-level synthesis workflow

gration of MPC with environmental awareness presents challenges in terms of runtime performance, calling for higher throughput efforts in calculating signal reflection to serve the bursty and bulky ray tracing requests. Consequently, this paper presents an enhanced High-Level Synthesis (HLS) workflow to facilitate the implementation of LRT on FPGAs that is well-customized for the target application scene, thus achieving real-time acceleration of ray tracing by massive simulation requests. The main contributions of our work are as follows:

- In the proposed HLS workflow, an improvement of the module architecture is achieved by pipelining reflection path derivation to provide high throughput capacity for the bursty bulky ray tracing simulation requests in runtime, while the acceleration of ray validation is also realized by hardware modules to meet the real-time performance;
- A lookahead ray tracing approach is proposed in this paper to explore the impact of dynamics within a predefined future time window, which contributes

to appropriate beam steering scheme for better beam tracking against environmental dynamics;

- This approach provides a potential avenue for enhancing the **capacity to configure beam management in response to scene dynamics**, which in this paper is regarded as an improvement in the **robustness of environmental awareness**.

II. Lookahead Ray-Tracing

Environmental awareness allows the dynamics of the application scene to be captured by sensors (such as LiDAR or depth cameras). This provides an efficient way to derive motion models of dynamic objects in the scene, which can be used to predict potential locations of the dynamic objects within the given time window. After testing potential locations in ray tracing simulation, all passing reflection paths in signal propagation are collected, which helps to rationally control beam steering to cover dynamics within the given time window and improve beam tracking robustness, as shown in Fig. 1.

At time t_0, a "snapshot" of current scene is taken. Dynamic objects and their motion models are then detected, which initiates the LRT to simulate the predicted movement with n time steps in the given time window t_w. Subsequently, the ray tracing results can be used to manage the beams, configuring the steering angle and thereby providing reliable beam tracking that accounts for environmental dynamics. As the t_w is coupled with the sensor's sampling rate (e.g., 30 ms for sensor with 30 FPS), extreme cases, such as a sudden change in the movement trend of the target UE, can be neglected, given the assumption that the UE cannot cause significant changes of movement trend in such a short time.

Algorithm 1 Pipelined Reflection Path Derivation

Input: Transmitter and receiver location TX and RX, respectively; Set of reflection planes Ω

Output: Coordinates of reflection point (RP) on the given plane Ω

$t_{TX} \leftarrow \frac{\Omega.A \cdot TX.x + \Omega.B \cdot TX.y + \Omega.C \cdot TX.z + \Omega.D}{\sqrt{\Omega.A^2 + \Omega.B^2 + \Omega.C^2}}$

$t_{RX} \leftarrow \frac{\Omega.A \cdot RX.x + \Omega.B \cdot RX.y + \Omega.C \cdot RX.z + \Omega.D}{\sqrt{\Omega.A^2 + \Omega.B^2 + \Omega.C^2}}$

$\Delta x_{TX} \leftarrow \Omega.A \cdot t_{TX}; \Delta x_{RX} \leftarrow \Omega.A \cdot t_{RX}$

$\Delta y_{TX} \leftarrow \Omega.B \cdot t_{TX}; \Delta y_{RX} \leftarrow \Omega.B \cdot t_{RX}$

$\Delta z_{TX} \leftarrow \Omega.C \cdot t_{TX}; \Delta z_{RX} \leftarrow \Omega.C \cdot t_{RX}$

$M \leftarrow |\Delta x_{TX}| + |\Delta y_{TX}| + |\Delta z_{TX}|$

$N \leftarrow |\Delta x_{RX}| + |\Delta y_{RX}| + |\Delta z_{RX}|$

$L_x \leftarrow RX.x - TX.x + \Omega.A(t_{RX} - t_{TX})$

$L_y \leftarrow RX.y - TX.y + \Omega.B(t_{RX} - t_{TX})$

$L_z \leftarrow RX.z - TX.z + \Omega.C(t_{RX} - t_{TX})$

for Fitting Reflection Point Displacement Iteratively **do**

  $\Delta x_{RP} \leftarrow \frac{M}{M+N} \cdot L_x$

  $\Delta y_{RP} \leftarrow \frac{M}{M+N} \cdot L_y$

  $\Delta z_{RP} \leftarrow \frac{M}{M+N} \cdot L_z$

end for

$RP.x \leftarrow TX.x + \Delta x_{TX} + \Delta x_{RP}$

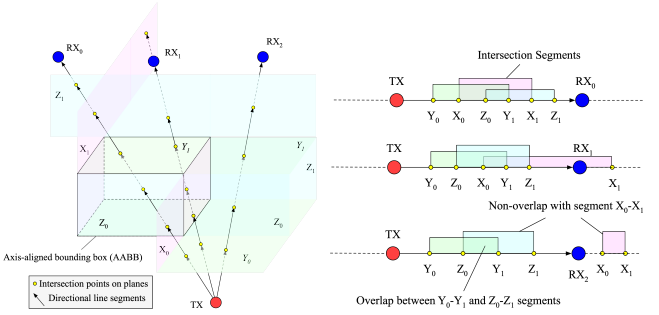
$RP.y \leftarrow TX.y + \Delta y_{TX} + \Delta y_{RP}$

$RP.z \leftarrow TX.z + \Delta z_{TX} + \Delta z_{RP}$

if $RP \in \Omega.boundary$ **then**

  return RP

end if



(a) Three rays are calculated for validation (b) Planes and rays intersect in different orders

Fig. 4. Example of slab-method based ray validation

III. HLS Workflow and Module Architecture

The real-time performance of ray tracing is of paramount importance. However, given that the LRT proposed in this paper necessitates massive ray tracing requests, for which the hardware architecture proposed in [13] and [14] in the form of a matrix of individual processing elements is no longer adequate, a pipelined enhancement in reflection path derivation is presented in this section. The hardware architecture is illustrated in Fig. 2. The location of the transmitter/base station (TX/BS) and receiver/user equipment (RX/UE) to be employed in the ray tracing simulation are obtained by the environmental sensors.

These values are then used to generate a sequence of predicted locations according to the current motion model of RX/UE, which are passed into our proposed pipelined Ray Processing Unit (RPU) for ray tracing simulation: After the projection of the TX and RX on a given plane and the extraction of the essential parameters for the reflection point fitting (completed in four stages), the reflection point on the given plane can be calculated in eight steps. The result is a reflection point on the given reflection plane that corresponds to a first-order reflection path, which are then buffered into a FIFO and forwarded to the Ray Validation Unit (RVU) for validity checking.

Moreover, an enhanced HLS workflow is designed for the purpose of systematically configuring the ray tracing platform implemented on FPGAs, which is intended to make the platform adapt well to the application scene and achieve parallel acceleration in ray tracing for the target scene, as shown in Fig. 3. At the initiation of the proposed HLS workflow, the target scene is captured by depth cameras to obtain 3D point cloud data. The potential reflection planes and obstacles can then be detected. Subsequent to this initialization, two sub-workflows are designed to proceed independently: One addresses the management of signal reflection on the planes as indicated by **sub-workflow A**, while the other handles the issue of obstacles in **sub-workflow B**.

A. Sub-workflow A: processing reflection planes

Sub-workflow A starts with grouping reflection planes into clusters. By clustering and aligning all the planes contributing to signal reflection according to their orientations (normal vectors), the geometric computation can be constrained to a set of mutually independent axes, significantly reducing the granularity of parallel processing elements and their hardware implementations on FPGAs. Next, the module for reflection path derivation in ray tracing is generated in pipelined structure, as shown in Fig. 2 and Algorithm 1.

Algorithm 2 Slab-method based ray validation for single obstacle $RayValidation$

Input: Ray R (segment) to be validated; Given obstacle B is represented by its AABB as $B = \{\Omega_{x0}, \Omega_{x1}, \Omega_{y0}, \Omega_{y1}, \Omega_{z0}, \Omega_{z1}\}$

Output: Validity V of the given ray

$R_X, R_Y.R_Z \leftarrow R$ projected on X, Y and Z-Axis, respectively ▷ Calculate the intersection order index

$\sigma_{x0} \leftarrow \frac{\Omega_{x0} \cdot D}{R_X} \cdot R; \sigma_{x1} \leftarrow \frac{\Omega_{x1} \cdot D}{R_X} \cdot R$

$\sigma_{y0} \leftarrow \frac{\Omega_{y0} \cdot D}{R_Y} \cdot R; \sigma_{y1} \leftarrow \frac{\Omega_{y1} \cdot D}{R_Y} \cdot R$

$\sigma_{z0} \leftarrow \frac{\Omega_{z0} \cdot D}{R_Z} \cdot R; \sigma_{z1} \leftarrow \frac{\Omega_{z1} \cdot D}{R_Z} \cdot R$

$X_{min} \leftarrow min(\sigma_{x0}, \sigma_{x1}); X_{max} \leftarrow max(\sigma_{x0}, \sigma_{x1})$

$Y_{min} \leftarrow min(\sigma_{y0}, \sigma_{y1}); Y_{max} \leftarrow max(\sigma_{y0}, \sigma_{y1})$

$Z_{min} \leftarrow min(\sigma_{z0}, \sigma_{z1}); Z_{max} \leftarrow max(\sigma_{z0}, \sigma_{z1})$

if $(X_{max} < Y_{min}) \vee (X_{max} < Z_{min}) \vee (Y_{max} < X_{min}) \vee (Y_{max} < Z_{min}) \vee (Z_{max} < Y_{min}) \vee (Z_{max} < X_{min})$ **then**

    $V \leftarrow$ **True** ▷ Non-overlap in intersection segments

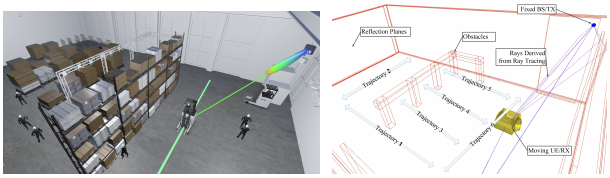
else

    $V \leftarrow V$ **False** ▷ Overlaps in intersection segments

end if

return V

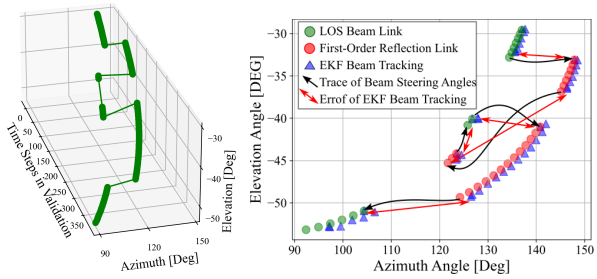
The initial stages consist of calculating geometric parameters for subsequent reflection point (RP) fitting. This includes point projection onto planes and derivation of unsigned point distances, which is outlined in [13] and [14]. Once the essential parameters are extracted from the projection, the RP fitting on the given plane Ω can be performed: The relative displacement of the RP can be calculated along each axis as Δx_{RP},Δy_{RP} and Δz_{RP} according to the law of similar triangles. The overall calculation is achieved in eight pipeline stages, where the depth of the pipeline directly correlates to the precision of the geometric calculation. To save the cost



(a) Scene in digital twin (b) Model used in FPGA

Fig. 5. Experiment environment in digital twin and processed by real-time ray-tracing: moving vehicle imitates dynamics of UE

of DSP and complex logic such as multiplication and division, functional alternatives such as shift-and-add are used to complete the fitting process in multiple stages. Finally, if the coordinates of the reflection point RP are valid, i.e. inside the bounded plane Ω, the following modules can take them for further processing. After the



(a) Optimal beam steering over time (b) Calculated beam steering angles to track the target UE

Fig. 6. Beam tracking by LRT and EKF

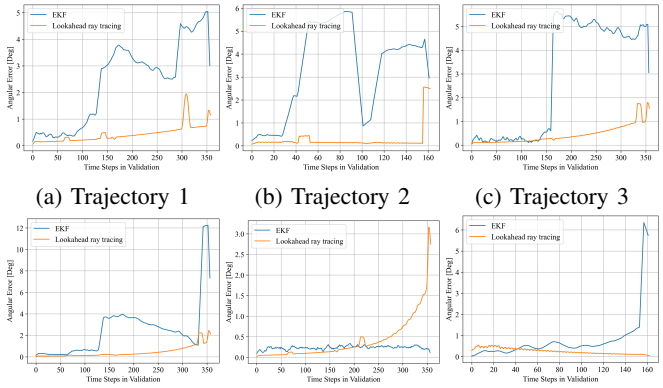
pipelined RPU has been generated to register-transfer level (RTL) modules, the sub-workflow A regarding the processing of reflection planes is finished.

B. Sub-workflow B: processing obstacles

In sub-workflow B, a slab-method-based [15] ray-box intersection checking is introduced to explore the hardware implementation for rapid ray validation. As an example illustrated in Fig. 4, the rays bounded by the transmitter TX with different receivers RX_0, RX_1, RX_2 are evaluated for potential conflicts with the cuboid-shaped obstacle. The obstacle is represented by its Axis-Aligned Bounding Box (AABB). The ray from the TX to various RXs can intersect the

planes of AABB in different order, thus revealing the order of incident and outgoing points along X-, Y- and Z-axes, which is used to check if the ray "hits" the box: The ray is confirmed not to be blocked by the given box as soon as no overlap of the ray segments bounded by the incident and outgoing points along each axis is recognized in the order of intersections, i.e. the ray is not correlated with the corresponding axis; Otherwise the ray-box intersection is approved, as illustrated in Fig. 4.

Therefore, the hardware implementation is accomplished using the same mechanism in the Ray Validation Unit (RVU). As shown in Fig. 2, the RVU is a subsystem that performs ray-box intersection checking on each obstacle for the given ray in a parallel processing manner, in which each validation is performed by a single ray validation element (RVE) regarding each obstacle, as shown in Algorithm 2. The ray to be validated is



(a) Trajectory 1 (b) Trajectory 2 (c) Trajectory 3

(d) Trajectory 4 (e) Trajectory 5 (f) Trajectory 6

Fig. 7. Performance of EKF and LRT in beam tracking

initially projected on X, Y, and Z-axes, followed by the intersection order to be confirmed. As illustrated in Fig. 2, the order of intersection points on the six specified planes of AABB can be determined using the method of similar triangles, indicated here as the order index $\sigma_{x0}..\sigma_{z1}$. The ray segments bounded by the pair of parallel planes along each axis can be used to reveal the overlaps: Overlap detected indicates ray blockage while non-overlap indicates validity. In the sub-workflow B the RVEs are first generated based on each obstacle detected, followed by generating the RVU wrapping all RVEs to provide parallel validation checking for each ray on each obstacle in eight cycles.

IV. Performance and Experiment

This section presents a validation of the proposed LRT approach for efficient beam management, where the target scene is represented by the digital twin of a real warehouse scenario via geometric reconstruction. This scene is further processed by our proposed HLS workflow for the implementation on FPGAs, as shown in Fig. 5 (b) is the simplified model generated for FPGA-based LRT. An automated guided vehicle (AGV) is introduced as the target UE to move in the target scene with a fixed BS, which is controlled on PC via the digital twin. The validation setup and implementation results are shown in Table I.

TABLE I
Validation Setup and Implementation on FPGA

	Results in 13	Results in 15	This paper
FPGA Chip	Intel Cyclone IV E EP4CE115F29C7		
Used Resource	4%-43%	13%	7%-33%
Reflect. Order	1-3		1
Clock Freq.	50 MHz		100 MHz
Elapsed Time	<1ms		<10us
Target Scene	Living Room	Office Room	Warehouse
UE Location	Fixed		Dynamic
Blockage	Dynamic		Fixed
PE Structure	Matrix Form		Pipeline
Ray Validation	Fixed geometric relation		RVU
HLS Overhead	High		Low

The FPGA-based ray tracing module is tested with UE trajectories stimulated by the test bench implemented adjacent to it. The ray tracing results are then collected and compared with the beam tracking results of an extended Kalman filter (EKF) in angular error, which indicates the discrepancy between the computed value and the theoretical optimal beam steering angle at each validation time point. As the UE moves, the beam steering angles change over time, as illustrated in Fig. 6. Six trajectories are tested on the UE, as shown in Fig. 5, with the location sampled every 0.5 m, which is the same as the lookahead time window covers for ray tracing with 10 predictive location points in each.

As the UE enters and exits the shadowing regions caused by obstacles, the beam steering angle suddenly changes and switches to 1^{st}-order reflection paths, causing large error in the EKF beam tracking up to 12 degrees, while the LRT assisted beam management shows much smaller angular error, with a maximum of less than three degrees, as shown in Fig. 7 and 8. It is interesting to note that for trajectory 5, our proposed LRT shows a significant error compared to the EKF result. This is due to the quantization error of the FPGA implementation: As the UE moves away from the BS, the RP located on the back wall exhibits a negligible degree of displacement due to the considerable distance between the BS and the back wall. Consequently, the optimal steering angles corresponding to each time

point are not clearly distinguishable in the hardware implementation since the FPGA modules utilize integer arithmetic instead of floating point arithmetic. However, this can be solved easily by pre-increasing the geometrical parameters by a certain factor (e.g. 1024 times) in the HLS workflow to compensate for arithmetic accuracy of hardware implementation. Moreover, the accuracy of LRT can be enhanced by making adjustments to the time window and the number of predictive locations of the UE within the time window. This is readily managed in FPGA implementations by means of trade-offs with timing and on-chip resource utilization, such as adjusting the depth of the RPU pipeline.

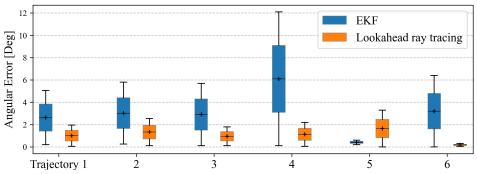


Fig. 8. High-level synthesis workflow

V. Conclusion

In this paper, an enhanced HLS workflow is proposed to facilitate LRT for environmental awareness accelerated by FPGAs. This workflow integrates beam management with model predictive control, thereby handling environmental dynamics for robust beam tracking. To address the challenges posed by bursty, bulky ray tracing requests in LRT, a pipelined structure for computing signal reflection in rays is also proposed, followed by parallelized ray validation modules to achieve rapid processing that supports the real-time beam management. After validation using the digital twin of a real scene, the method proposed in this paper is validated to improve the performance of beam tracking against environmental dynamics, showing application potentials in efficient environmental awareness for robust beam management.

Acknowledgment

This work has received funding by the German Federal Ministry of Education and Research (BMBF) in the course of the 6GEM research hub under the grant number 16KISK038.

References

[1] S. Shaham, M. Kokshoorn, M. Ding, Z. Lin, and M. Shirvanimoghaddam, "Extended kalman filter beam tracking for millimeter wave vehicular communications," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020, pp. 1–6.

[2] H.-L. Song, Y.-c. Ko, J. Cho, and C. Hwang, "Beam tracking algorithm for uav communications using kalman filter," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, 2020, pp. 1101–1104.

[3] G. Liu, F. Zhou, and P. Yu, "Beam tracking based on unscented kalman filter theory in millimeter wave communication systems for iot," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, 2020, pp. 499–504.

[4] W. Chen, H. Yang, X. Bi, R. Zheng, F. Zhang, P. Bao, Z. Chang, X. Ma, and D. Zhang, "Environment-aware multi-person tracking in indoor environments with mmwave radars," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 7, no. 3, pp. 1–29, 2023.

[5] J. Pegoraro and M. Rossi, "Real-time people tracking and identification from sparse mm-wave radar point-clouds," *IEEE Access*, vol. 9, pp. 78 504–78 520, 2021.

[6] J. Pegoraro, D. Solimini, F. Matteo, E. Bashirov, F. Meneghello, and M. Rossi, "Deep learning for accurate indoor human tracking with a mm-wave radar," in *2020 IEEE Radar Conference (RadarConf20)*. IEEE, 2020, pp. 1–6.

[7] G. Charan, M. Alrabeiah, and A. Alkhateeb, "Vision-aided 6g wireless communications: Blockage prediction and proactive handoff," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10 193–10 208, 2021.

[8] G. Charan and A. Alkhateeb, "Computer vision aided blockage prediction in real-world millimeter wave deployments," in *2022 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2022, pp. 1711–1716.

[9] S. Mihara, T. Murakami, A. Yamaguchi, and H. Shinbo, "User equipment tracking for a millimeter wave system using vision and rssi," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 1563–1577, 2022.

[10] T. Murakami, K. Yoshikawa, A. Yamaguchi, and H. Shinbo, "Blockage prediction in an outdoor mm wave environment by machine learning employing a top view image," in *2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2022, pp. 1–6.

[11] M. Al-Quraan, A. Centeno, A. Zoha, M. A. Imran, and L. Mohjazi, "Federated learning for reliable mmwave systems: Vision-aided dynamic blockages prediction," in *2023 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2023, pp. 1–6.

[12] Y. Yang, F. Gao, X. Tao, G. Liu, and C. Pan, "Environment semantics aided wireless communications: A case study of mmwave beam prediction and blockage prediction," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 7, pp. 2025–2040, 2023.

[13] J. An and S. Saidi, "Hls-based approach for embedded real-time ray tracing in wireless communications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 11, pp. 3720–3731, 2024.

[14] J. An, S. Saidi, and L. Simić, "Building hardware accelerators for environmental awareness using ray-tracing," in *2024 Joint European Conference on Networks and Communications / 6G Summit (EuCNC/6G Summit)*, 2024, pp. 1038–1043.

[15] D. Marasinghe, N. Rajatheva, and M. Latva-aho, "Lidar aided human blockage prediction for 6g," in *2021 IEEE Globecom Workshops (GC Wkshps)*, 2021, pp. 1–6.