

Real-Time mmWave Sensing for Robotic Digital Twins

Alberto Martínez Méndez, Alejandro Calvillo-Fernandez, Antonio de la Oliva, and Carlos J. Bernardos

Abstract—In current mobile robotics, perception relies mainly on onboard sensors, whose coverage is inherently local and may leave relevant regions uncovered under partial observability. Existing solutions often address this limitation by adding sensing modalities or deploying richer multisensor infrastructures, which increases system cost, integration effort, and deployment overhead. Motivated by the integrated sensing and communication (ISAC) paradigm, this work presents a real-time system that reuses external millimeter-wave (mmWave) communication hardware to extract channel state information (CSI)-based semantic context and inject it into a robotic digital twin. The resulting twin represents not only the robot state and observed geometry, but also semantically inferred information from regions outside the robot viewpoint. To improve reliability under deployment conditions, we address cross-session variability through multi-session dataset aggregation and support vector machine (SVM)-based classification, and we reduce online semantic flickering through a lightweight temporal confirmation mechanism. Experimental results show that the proposed approach improves cross-session robustness, suppresses transient semantic excursions during runtime, and can support more informed downstream decisions under incomplete visibility.

Index Terms—Digital twin, mmWave sensing, channel state information, semantic sensing, robotic perception, Isaac Sim, ROS2, session shift, real-time inference.

I. INTRODUCTION

Wireless networks are increasingly evolving beyond communication-only infrastructures towards integrated sensing and communication (ISAC) systems, where the same radio resources support both data transfer and environmental awareness [1]. Within this perspective, external wireless sensing becomes attractive as a low-overhead way to extract contextual information from the propagation channel itself, without requiring dedicated sensing hardware. In particular, external millimeter wave (mmWave) sensing [2], [3] enables the use of commercial off-the-shelf radio devices not only as connectivity enablers, but also as complementary semantic sensors.

This capability is especially relevant for mobile robotics. Mobile robots typically rely on onboard sensors such as LiDARs or cameras to perceive and interpret their surroundings [4], [5]. These sensing modalities are fundamental for navigation and environment understanding, but their perception remains inherently local and viewpoint-dependent. During operation, relevant regions may lie outside the field of view, behind occlusions, or in areas not yet explored, so context that is operationally relevant may remain unavailable to the robot.

In this work, we explore this direction by integrating external mmWave channel state information (CSI)-based semantic sensing into a robotic digital twin in Isaac Sim.¹ Rather than treating the digital twin only as a geometric replica of the robot, we use it as a runtime semantic representation in which

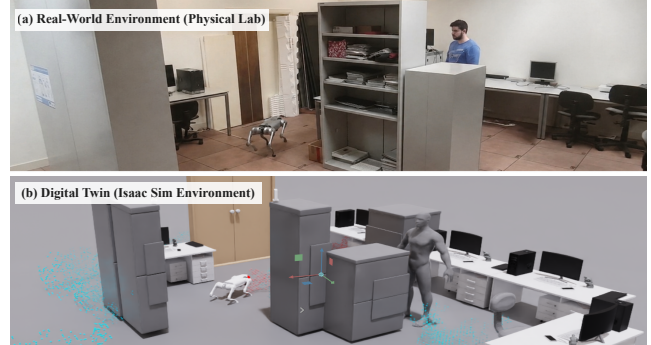


Fig. 1. Conceptual overview of the proposed solution, coupling the real scene with the digital twin.

robot state, observed geometry, and externally inferred entities coexist. The proposed system combines two branches: an external sensing branch that acquires mmWave CSI measurements from commodity MikroTik devices, performs semantic classification, and publishes a shared semantic state; and a robot branch that transfers the real Unitree Go2 pose, leg kinematics, and point-cloud stream into Isaac Sim through Robot Operating System 2 (ROS2). In this way, the digital twin represents not only what the robot directly observes, but also semantically inferred information about externally monitored regions.

Fig. 1 illustrates the intended operational outcome of the proposed approach. The real scene and its digital counterpart are coupled at runtime: the robot state is reflected in the simulator, while externally sensed semantic information is injected into the digital twin to represent entities that may not be directly visible from the robot viewpoint. This allows the digital twin to expose a more informative runtime view of the environment, combining onboard perception with complementary external awareness.

From a deployment perspective, two practical issues must be addressed: CSI-based inference may degrade across sensing sessions, and raw real-time predictions may produce short-lived semantic flickering. To handle these effects, the proposed system adopts a lightweight solution based on multi-session dataset aggregation, support vector machine (SVM)-based classification with radial basis function kernel, and a simple temporal confirmation mechanism for online stabilization.

The main contributions of this paper are stated as follows:

- a complementary perception architecture that integrates external mmWave CSI-based semantic sensing with a robotic digital twin in Isaac Sim;
- a lightweight semantic inference pipeline suitable for real-time deployment with commodity communication hardware;

¹NVIDIA Isaac Sim: <https://developer.nvidia.com/isaac/sim>

- a practical strategy to improve deployment robustness under session shift, based on multi-session dataset aggregation and SVM-based classification;
- a lightweight temporal stabilization mechanism that suppresses short-lived semantic flickering before the inferred state is consumed by the digital twin;
- an experimental evaluation showing that the resulting system provides a stable semantic runtime representation that complements onboard robot perception in partially observable situations.

The remainder of the paper is organized as follows. Section II presents the external mmWave sensing pipeline. Section III describes the proposed system architecture and the runtime convergence in the digital twin. Section IV reports the experimental results. Finally, Section V concludes the paper and outlines future work.

II. RF SENSING PIPELINE

The external sensing branch relies on mmWave CSI measurements acquired from a pair of commercial MikroTik devices running a OpenWrt firmware [6]. Unlike vision-based sensing, this branch does not require a direct camera view of the scene from a fixed external viewpoint. Its purpose is to transform wireless measurements into a compact semantic state that can be consumed by the digital twin at runtime.

A. Measurement model and sample representation

The sensing principle is based on the fact that changes in the environment modify the wireless propagation channel and, reflected in the measured CSI [7]. A standard multipath representation of the channel response is

$$H(f, t) = \sum_{\ell=1}^L \alpha_{\ell}(t) e^{-j2\pi d_{\ell}(t)/\lambda}, \quad (1)$$

where L is the number of relevant propagation paths, $\alpha_{\ell}(t)$ is the complex attenuation of path ℓ , $d_{\ell}(t)$ is its propagation length, and λ is the wavelength. Variations in reflections, path lengths, and occlusions are observable through CSI.

In our implementation, each valid sensing sample is acquired from the MikroTik-based mmWave link as a single measurement record containing timestamp and link metadata, with 32 phase values and 32 magnitude values. After parsing, each sample is represented as $\mathbf{x} = [\phi, \mathbf{m}]$, where ϕ and $\mathbf{m}$ denote the phase and magnitude components respectively.

Although both magnitude and phase encodings can be derived from the parsed CSI record, the experiments reported in this paper use the phase representation, obtained by concatenating ϕ and $\mathbf{m}$. Thus, each input sample contains 64 features: 32 phase values and 32 magnitude values.

B. Semantic classification task

Rather than estimating geometry or reconstructing a dense occupancy map, we formulate external sensing as a supervised multi-class classification problem. Given a parsed CSI sample $\mathbf{x}$, the model predicts a semantic label $\hat{y} \in \mathcal{C}$, where $\mathcal{C}$ denotes the set of semantic states considered in the monitored region.

Now, the system does not require the exact pose, shape, or extent of the detected entity, but only a compact semantic state indicating the condition of the monitored wireless link. This keeps the sensing branch lightweight, reduces annotation effort, and aligns with the way semantic information is later consumed by the digital twin.

C. Training data and classifier

The dataset used in this work was collected using the MikroTik 60 GHz devices running OpenWrt-based firmware. Each sample was stored as an individual text file containing one valid measurement line, and the dataset was organized by semantic class. In total, the dataset contains 3000 samples, evenly distributed across the three classes, with 1000 samples per class. Each sample corresponds to a single CSI measurement acquired by the MikroTik-based sensing link. The training features correspond directly to the parsed raw CSI values, using either magnitude only or phase plus magnitude, depending on the selected encoding.

To map features into semantic classes, we use a SVM with radial basis function kernel [8]. Given an input vector $\mathbf{x} \in \mathbb{R}^d$, the decision function is

$$f(\mathbf{x}) = \sum_{i=1}^M \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b, \quad (2)$$

where $\mathbf{x}_i$ are support vectors, α_i are learned coefficients, y_i are class indicators from the binary subproblems, and b is the bias term. The kernel is

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \quad (3)$$

where γ controls the kernel width. Before classification, all features are standardized using the training statistics, since SVM performance is sensitive to feature scale. In the multi-class setting, the predicted label is obtained as

$$\hat{y} = \arg \max_{c \in \mathcal{C}} s_c(\mathbf{x}), \quad (4)$$

where $s_c(\mathbf{x})$ is the score assigned to class c .

D. Online inference and deployment considerations

At runtime, new CSI measurements are appended to a live stream file and processed online using the same feature extraction procedure employed during training. Each sample produces a raw semantic prediction $\hat{y}_{\text{raw}}^{(t)}$. Since single-sample outputs may fluctuate due to noise or transient channel perturbations, the system applies a lightweight temporal confirmation mechanism before publishing the final semantic state. A label transition is accepted only if the same raw prediction has been observed for K consecutive samples:

$$\hat{y}_{\text{raw}}^{(t-K+1)} = \hat{y}_{\text{raw}}^{(t-K+2)} = \dots = \hat{y}_{\text{raw}}^{(t)}. \quad (5)$$

The resulting stabilized output, denoted as $\hat{y}_{\text{stable}}^{(t)}$, is then written into the shared JSON state, later consumed by Isaac Sim.

A practical challenge in this setting is that CSI distributions are not fully stationary across acquisition sessions.

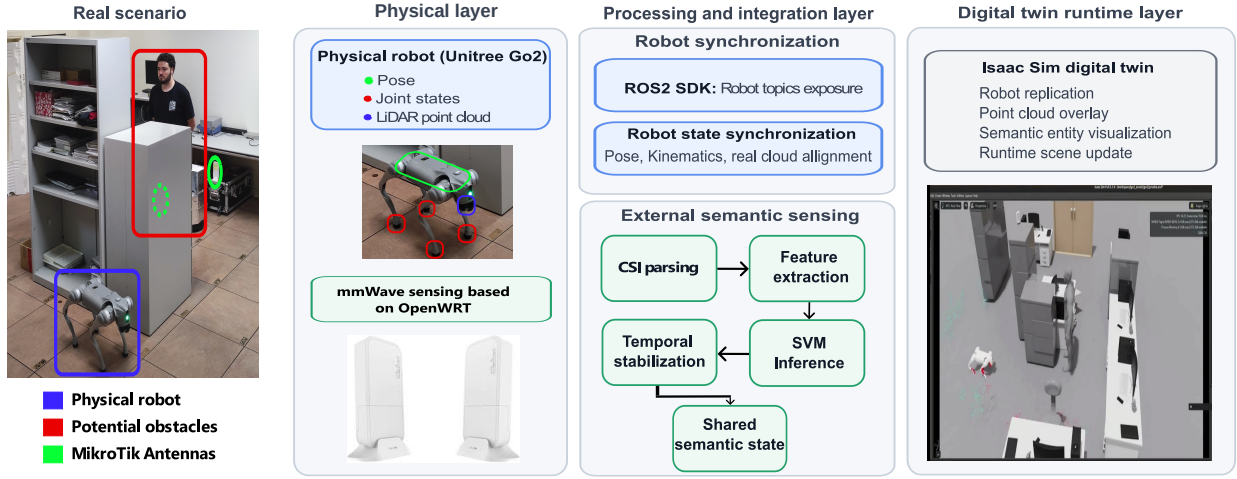


Fig. 2. Three-layer architecture of the proposed system, from the physical scenario to runtime integration in the digital twin.

If $p_{\text{train}}(\mathbf{x}, y)$ and $p_{\text{test}}(\mathbf{x}, y)$ denote the joint distributions at training and deployment time, in practice we often face $p_{\text{train}}(\mathbf{x}, y) \neq p_{\text{test}}(\mathbf{x}, y)$, which degrades generalization. In our case, this effect is caused by variations in link conditions, hardware initialization, environmental clutter, and target placement. For this reason, the final model is trained on aggregated multi-session data, while temporal stabilization reduces sensitivity to short-lived fluctuations during online operation.

III. SYSTEM ARCHITECTURE

The proposed architecture is organized into three layers and two parallel data flows, as illustrated in Fig. 2. On the left, the real scenario shows the physical deployment: a Unitree Go2 robot operates in the scene, while a pair of MikroTik 60 GHz devices forms an AP/STA wireless link across the region to monitor. The devices are placed so that potential external entities perturb the propagation path and their effect can be reflected in the CSI. In this experimental setup, the semantic states considered by the sensing branch correspond to an empty monitored region, a person, or a TurtleBot platform [9]. From this physical layer, two branches are generated in parallel: the robot branch, which provides pose, articulated state, and LiDAR information, and the external sensing branch, which provides mmWave CSI measurements extracted from the wireless link. Both branches are processed independently in the processing and integration layer and finally converge in the digital twin runtime layer, where Isaac Sim combines synchronized robot state, point-cloud information, and externally inferred semantic context into a single representation.

A. External sensing branch

As shown in Fig. 2 (left and center-bottom), the external sensing branch starts from the MikroTik 60 GHz AP/STA pair running a researcher-oriented OpenWrt-based firmware that enables access to CSI measurements from the wireless link [6]. Instead of using dedicated radar hardware, the system reuses a communication radio link as a sensing source: while the link is maintained between the two MikroTik devices, the extracted

CSI captures propagation changes caused by entities located in the monitored region.

During runtime, the acquired CSI measurements are parsed and converted into feature vectors in the processing and integration layer; see Fig. 2 (center-bottom). These features are then fed into a lightweight supervised classifier that outputs a semantic label describing the state of the monitored region. A practical difficulty in this setting is that wireless sensing models are sensitive to environmental and deployment variability: changes in device placement, surrounding clutter, or acquisition conditions can alter CSI patterns and degrade performance across sessions. This generalization challenge is well known in wireless sensing and remains an open issue beyond the scope of the current prototype [10].

For this reason, the current system adopts a pragmatic deployment-oriented strategy; see again Fig. 2 (center-bottom). The classifier is trained with data collected across multiple sessions in order to reduce sensitivity to session-specific artifacts, but the main online safeguard is a lightweight consistency mechanism between consecutive samples. In practice, the selected model is a lightweight SVM with radial basis function (RBF) kernel, offering a good compromise between non-linear separability, low runtime cost, and deployment simplicity. Before publication, the pipeline applies a lightweight temporal confirmation mechanism so that isolated fluctuations do not propagate directly to the runtime state.

The stabilized output is written into a shared runtime state, implemented in the current prototype as a small JSON file; see Fig. 2 (center-bottom). This branch therefore does not attempt to reconstruct full scene geometry from wireless measurements, but instead produces a compact semantic interpretation of the externally monitored region. The effect of this stabilization mechanism under real-time operation is analyzed in Section IV.

B. Robot synchronization branch

In parallel to the external sensing flow, the robot synchronization branch continuously transfers the state of the real

Unitree Go2 into Isaac Sim; see Fig. 2 (top-center). For this purpose, the system uses the public ROS2 integration repository [11], which exposes robot information including pose, articulated state, and LiDAR-derived point-cloud data.

As shown in Fig. 2 (top-center), the robot-side stack first exposes the relevant information through ROS2 topics, after which a Python-based synchronization module running with Isaac Sim consumes these topics and updates the simulated robot accordingly. In the current implementation, this includes the robot base pose, leg joint states, and point-cloud stream, allowing the digital twin to remain aligned with the motion and observed geometry of the physical platform during runtime.

C. Runtime convergence in the digital twin

The final convergence takes place in the digital twin runtime layer, where the outputs of both branches are combined inside Isaac Sim; see Fig. 2 (right). The robot synchronization branch provides the current robot embodiment together with the geometric view of the environment through the point cloud, while the external sensing branch provides the semantic state read periodically from the shared JSON file.

In the current prototype, the semantic output of the external sensing branch is a discrete class prediction, which is periodically read by Isaac Sim from the shared JSON state; see Fig. 2 (right). Inside the simulator, this prediction is mapped to the visibility of semantic entities already placed in the scene. Thus, when the monitored region is classified as containing one of the potential obstacles considered in the real scenario, the corresponding virtual entity is activated; when it is classified as empty, these entities remain hidden. In this way, the digital twin acts as a runtime integration layer where synchronized robot state, observed geometry, and externally inferred semantic context coexist in a unified system representation.

IV. EXPERIMENTAL EVALUATION

This section evaluates three deployment-relevant aspects of the proposed system: robustness under cross-session data shift, temporal consistency during online inference, and the potential impact of the external semantic signal on route selection. This evaluation uses the prototype semantic classes introduced in Section III: *nothing*, *person*, and *turtlebot*. The first assesses whether a classifier that performs well under internal validation remains reliable when evaluated on data collected in a later sensing session. The second assesses whether the proposed stabilization mechanism suppresses short-lived semantic flickering before the inferred state is consumed by the digital twin. The third illustrates how externally inferred obstacle information can modify the path selected by the robot and reduce route cost with respect to a purely local reaction.

To this end, we study three complementary questions. First, we evaluate whether a classifier trained on a single sensing session generalizes to data collected in a later session. Second, we quantify the effect of temporal confirmation on the consistency of the semantic output during real-time operation. Third, we illustrate how the semantic output provided by the proposed

sensing pipeline can be exploited to anticipate a potential obstacle outside the robot’s immediate onboard perception and influence route selection accordingly. For the first question, we compare three confusion matrices: the holdout evaluation of the initial classifier on the initial dataset, the evaluation of that same classifier on a newly collected dataset from a different session, and the holdout evaluation of the classifier trained on the aggregated multi-session dataset. For the second question, we run the real-time inference pipeline 10 times for each configuration over the same live CSI stream, using the same trained model while changing only the temporal confirmation count.

A. Cross-session robustness of the semantic classifier

Fig. 3(a) shows the holdout evaluation of the initial classifier on the initial dataset. The result is nearly perfect: the classes *nothing* and *person* exhibit only one confusion in each direction, while *turtlebot* is classified correctly in all evaluated samples.

At first sight, the result in Fig. 3(a) could suggest that the sensing problem is already solved with a lightweight same-session model. However, this conclusion changes when the classifier is exposed to data collected in a later sensing session. Fig. 3(b) shows the confusion matrix obtained when the same initial classifier is evaluated on newly collected data from a different session, where the performance degrades substantially.

This degradation is structured and not random. The *nothing* class remains comparatively stable, which suggests that the empty-scene condition is easier to preserve across sessions. In contrast, the strongest degradation appears in the boundary between *turtlebot* and *person*. A large fraction of *turtlebot* samples is reassigned to *person*, indicating that the representation learned from the initial session does not remain invariant when acquisition conditions change. This behavior is particularly relevant from a deployment perspective, because it shows that same-session internal validation can substantially overestimate real operating performance in this problem.

To address this issue, we trained a new classifier on the aggregated dataset obtained by combining samples collected across multiple sessions. The corresponding holdout confusion matrix is shown in Fig. 3(c). In this case, the model achieves perfect class separation on the holdout split, correctly recognizing all samples from the three semantic classes. This indicates that the aggregated training set is sufficient to learn a substantially more robust decision boundary while preserving class separability. Instead of increasing model complexity, the adopted strategy enlarges the support of the training distribution so that the classifier becomes more tolerant to session-dependent variations.

B. Temporal stabilization during real-time inference

Even when the classifier is robust on average, isolated raw prediction errors may still appear during live execution. If directly propagated to the digital twin, these short-lived fluctuations produce semantic flickering, causing entities to

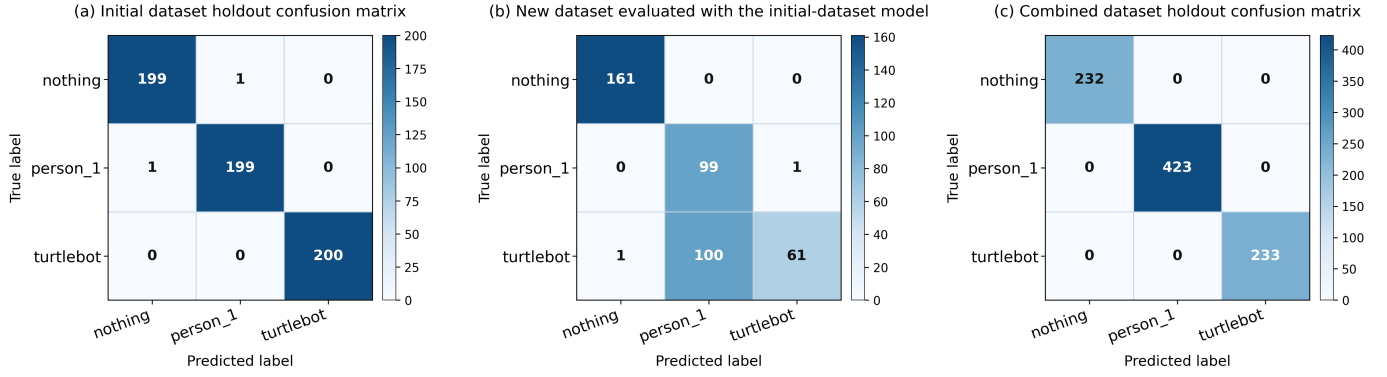


Fig. 3. Confusion matrices illustrating the effect of session shift and multi-session training.

TABLE I
EFFECT OF TEMPORAL STABILIZATION ON REAL-TIME SEMANTIC FLICKERING.

Configuration	Samples	Runs	Excursions	1/2-sample
No stabilization ($C = 1$)	2407	161	88	67/21
With stabilization ($C = 3$)	2407	19	0	0/0

appear and disappear in a way that is inconsistent with the underlying physical situation. This is undesirable in a runtime digital-twin setting, where short-term semantic coherence is often more important than reacting to every individual raw sample.

To quantify this effect, we ran the real-time inference pipeline 10 times for each configuration over the same live CSI stream: one with `confirm_count = 1`, corresponding to no temporal stabilization, and one with `confirm_count = 3`, corresponding to the deployed stabilized configuration. We quantify spurious semantic flickering as *transient label excursions*, defined as isolated one- or two-sample prediction runs enclosed by the same surrounding state. This definition captures the blink-like label changes that are undesirable for runtime semantic integration. Table I summarizes the average results over the 10 runs per configuration: both settings produced the same number of valid predictions, namely 2407 samples per run. Without stabilization, the output sequence exhibited on average 161 state runs and 88 transient label excursions, of which 67 were one-sample excursions and 21 were two-sample excursions. With stabilization enabled, the average number of runs decreased to 19 and the average number of transient excursions dropped to zero.

These results show that temporal confirmation is not merely a cosmetic smoothing layer, but a practical requirement for stable runtime integration. By accepting a semantic transition only after the same label has been predicted in consecutive samples, the system avoids updating the digital twin with isolated classification errors. For the target application, this small confirmation delay is acceptable because semantic consistency over short temporal windows is more important than reacting to every individual prediction.

C. Illustrative impact on route selection

Beyond classification robustness and temporal consistency, the proposed system is intended to provide useful semantic context for downstream robotic decisions. In the illustrative examples of Fig. 4, route selection is performed with the A* algorithm over a grid map in which each block represents an area of 60×60 cm. A* first computes the nominal shortest path using only the known static obstacles. When the external sensing branch reports a potential obstacle, the corresponding grid cell is added to the obstacle map and A* is executed again, producing an alternative route before the robot reaches the affected area. This illustrative example assumes that the potential obstacle lies within the region monitored by the external MikroTik link; optimizing the placement or number of sensing links is outside the scope of this work. In all cases, the black regions denote known obstacles, the red cross marks the robot initial position, the cyan cross marks the destination, and the blue marker denotes a potential obstacle detected through the external sensing infrastructure.

The first scenario (Fig. 4(a)) shows the case in which the robot relies only on its onboard perception. In this situation, the obstacle is detected only after the robot has already committed to its route, forcing a late detour around the central block and yielding a path cost of 16 blocks. Fig. 4(b) considers the same initial and goal positions, but now the proposed framework provides an early warning about the obstacle. In this case, the nominal shortest path, shown as the dashed green route, is discarded before execution, and the robot follows the alternative route shown in orange, reducing the cost to 10 blocks.

Fig. 4(c) and 4(d) extend the same idea to different start-goal configurations. In both scenarios, the dashed green path denotes the nominal shortest route that would be followed if no external obstacle information were available, whereas the orange path denotes the route selected once the externally detected obstacle is taken into account. These additional examples show that the benefit of the proposed sensing framework is not restricted to a single fixed initial and final position, but can also support re-routing decisions under different scene conditions.

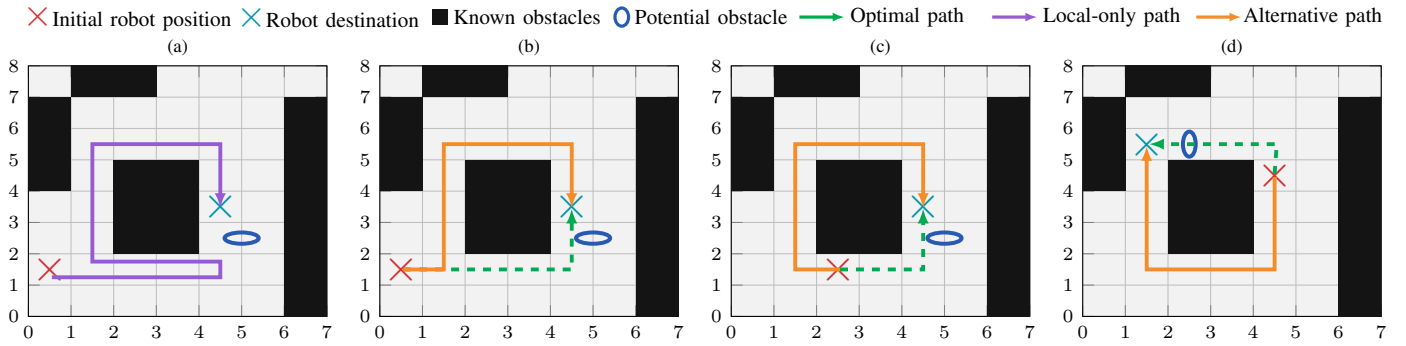


Fig. 4. Illustrative effect of external semantic sensing on route selection.

A video demonstration of the complete system is available online.²

V. CONCLUSION AND FUTURE WORK

This paper has presented a complementary perception system that integrates external mmWave CSI-based semantic sensing into a live robotic digital twin in Isaac Sim. Rather than using the digital twin only as a geometric or kinematic replica of the robot, the proposed approach turns external wireless observations into a compact semantic runtime state that coexists with the synchronized robot embodiment and observed geometry. The results show that this integration can be achieved with a lightweight and deployable architecture, while also highlighting two practical requirements for reliable operation: robustness under session shift and temporal consistency during online inference. In particular, the experiments show that a model validated on a single session may degrade strongly on newly collected data, and that raw per-sample predictions may generate unstable semantic flicker if directly exposed to the simulator. By combining multi-session training with lightweight temporal confirmation, the proposed system produces a more robust and coherent semantic state for runtime digital-twin augmentation.

Overall, the proposed system shows that external mmWave sensing can act as a meaningful source of complementary perception for robotics, augmenting the digital twin with semantic information about regions that are not necessarily visible to the robot at a given instant. Future work will extend this approach along four directions: increasing the number of acquisition sessions and environments to study broader generalization, incorporating additional sensing modalities to move toward multimodal semantic inference, enriching the current coarse semantic classes with more expressive representations, and connecting the inferred semantic state to downstream robotic modules such as navigation, planning, or adaptive behavior selection.

ACKNOWLEDGMENTS

This work has been partially funded by the European Commission Horizon Europe SNS JU MultiX (GA 101192521) Project, and by the Regional Government of Comunidad de Madrid under grant agreement No. TEC-2024COM-360 (Disco6G); and it is part of the Project 6GINSPIRE PID2022-137329OB-C42, funded by MCIN/AEI/10.13039/501100011033/ and TEST-6G, funded by TSI-064100-2023-12. We would like to thanks SLICES-Madrid for their services (<https://www.slices-madrid.eu/funding-acknowledgements/>).

REFERENCES

- [1] F. Liu, Y. Cui, C. Masouros, J. Xu, and L. Hanzo, "Integrated sensing and communications: Toward dual-functional wireless networks for 6g and beyond," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 6, pp. 1728–1767, 2022.
- [2] Z. Wei, W. Chen, S. Ning, W. Lin, N. Li, B. Lian, X. Sun, and J. Zhao, "A survey on wi-fi-based human identification: Scenarios, challenges, and current solutions," *ACM Transactions on Sensor Networks*, vol. 21, no. 1, pp. 1–32, 2025.
- [3] B. van Berlo, A. Elkelany, T. Ozcelebi, and N. Meratnia, "Millimeter wave sensing: A review of application pipelines and building blocks," *IEEE Sensors Journal*, vol. 21, no. 9, pp. 10 332–10 368, 2021.
- [4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.
- [5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.
- [6] P. J. Mateo and J. Widmer, "Mikrotik-researcher-tools: Researcher version of openwrt for wap 60g/lhgg-60ad/wap 60gx3 ap," GitHub repository, 2021, accessed: 2026-03-18.
- [7] P. Picazo-Martinez, C. Barroso-Fernández, A. Calvillo-Fernandez, M. Groshev, C. J. Bernardos, A. de la Oliva, and A. Mourad, "802.11bf multiband passive sensing: Reusing wi-fi signaling for sensing," *IEEE Internet of Things Journal*, vol. 13, no. 6, pp. 10 547–10 563, 2026.
- [8] C. Cortes and V. Vapnik, "Support-vector networks," in *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, ser. COLT '95. New York, NY, USA: ACM, 1995, pp. 273–297.
- [9] ROBOTIS, "Turtlebot3 overview," <https://emanual.robotis.com/docs/en/platform/turtlebot3/>, 2026, accessed: 2026-04-10.
- [10] F. Wang, T. Zhang, W. Xi, H. Ding, G. Wang, D. Zhang, Y. Cui, F. Liu, J. Han, J. Xu, and T. X. Han, "A survey on wi-fi sensing generalizability: Taxonomy, techniques, datasets, and future research prospects," *IEEE Communications Surveys & Tutorials*, 2026.
- [11] N. Abizov and contributors, "go2_ros2_sdk: Unofficial ros2 sdk support for unitree go2 air/pro/edu," GitHub repository, 2024, accessed: 2026-03-18.

²<https://youtu.be/zfRtX-8PC4o>