CASTER: A Computer-Vision-Assisted Wireless Channel Simulator for Gesture Recognition

ZHENYU REN [1], GUOLIANG LI [2,3] (Graduate Student Member, IEEE), CHENQING JI [1] (Graduate Student Member, IEEE), CHAO YU[1], SHUAI WANG [4] (Senior Member, IEEE), AND RUI WANG [1] (Member, IEEE)

[1]Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen 518055, China

[2]State Key Laboratory of Internet of Things for Smart City, University of Macau, Macau, China

[3]Department of Computer and Information Science, University of Macau, Macau, China

[4]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China

CORRESPONDING AUTHOR: R. WANG (e-mail: wang.r@sustech.edu.cn)

ABSTRACT In this paper, a computer-vision-assisted simulation method is proposed to address the issue of training dataset acquisition for wireless hand gesture recognition. In the existing literature, in order to classify gestures via the wireless channel estimation, massive training samples should be measured in a consistent environment, consuming significant efforts. In the proposed CASTER simulator, however, the training dataset can be simulated via existing videos. Particularly, in the channel simulation, a gesture is represented by a sequence of snapshots, and the channel impulse response of each snapshot is calculated via tracing the rays scattered off a primitive-based hand model. Moreover, CASTER simulator relies on the existing video clips to extract the motion data of gestures. Thus, the massive measurements of wireless channel can be eliminated. The experiments first demonstrate an 83.0% average recognition accuracy of simulation-to-reality inference in recognizing 5 categories of gestures. Moreover, this accuracy can be boosted to 96.5% via the method of transfer learning.

INDEX TERMS Wireless hand gesture recognition, channel model, simulation-to-reality inference.

I. INTRODUCTION

SENSING is becoming one of the core services of the next-generation wireless systems. There have been a significant number of works on wireless sensing, particularly the machine-learning-based human motion recognition (HMR), via channel state information (CSI) [1], [2], [3], [4] or passive architecture [5], [6], [7]. In most of these works, a significant number of labeled wireless signals should be collected and processed for the training of motion recognition models, which might be infeasible in many applications. In this paper, we would like to show that it is possible to generate the above training dataset for hand gesture recognition via channel simulation, instead of real measurement.

In fact, there have been a number of works on the extension of sophisticated communication channel models,

such that the effects of sensing target on the channel impulse response are incorporated. Hence, the channel simulation based on these models might be used for motion recognition. For instance, the Data-Driven Hybrid Channel (DAHC) model of IEEE 802.11bf specification [8], [9] divided wireless channel into two parts: the target-unrelated components and the target-related components. The existing methods of communication channel modeling can be applied on the former; whereas the primitive-based human body model [10] was utilized to compute the latter. A similar channel model was also used in [11] for the optimization of communication and sensing performance. The *WiGig Tools* [12], developed by National Institute of Standards and Technology (NIST), enriched existing quasi-deterministic channel ray-tracers with supplementary target-related rays (T-Rays), such that the consistent effects of human motion

could be included. Moreover, the methods for simulating radar echo signals off the human body were proposed in [13], [14]. All the above works relied on the primitive-based human body model [10], [15], where the hand was modeled as a single ellipsoid. Thus, these methods cannot model fine-grained hand gestures.

In order to facilitate the machine-learning-based HMR with the above channel models, diversified motion data are required to drive the primitive-based human body model in the channel simulation. Depth cameras and wearable sensors were used in [13], [14] to obtain sufficient body motion data for channel simulation. Nevertheless, to the best of our knowledge, there is no study on the capture of hand gestures for channel simulation. Moreover, it is unknown if conventional monocular cameras, instead of depth cameras, could obtain the motion data with adequate accuracy in the applications of wireless HMR. Note that the monocular cameras are of lower cost, and it is much more convenient to obtain hand gesture video clips of monocular cameras from online sources.

In this paper, we would like to shed some light on the above issues by proposing a Computer-vision-Assisted wireless channel SimulaTor for gEsture Recognition, namely CASTER. The proposed CASTER simulator is composed of channel generator and video gesture catcher. In the channel generator, the target hand is modeled with 21 primitives, and the channel impulse response is calculated by tracing the rays scattered off all the primitives. Based on the hand model, a gesture is represented by a sequence of snapshots, and the channel impulse responses for all the snapshots can be obtained respectively. In the video gesture catcher, trajectories of 21 primitives in one gesture can be captured from videos of a conventional monocular camera. Thus, the catcher provides an efficient way to retrieve motion data for the channel generator. In order to demonstrate the high fidelity of the proposed CASTER simulator, we use the simulated dataset of channel impulse responses to train a gesture recognition model and use a passive sensing system [7] to measure the real channel for model testing. It is shown that an 83.0% average recognition accuracy of simulation-to-reality inference can be achieved by recognizing 5 categories of gestures. Moreover, this accuracy can be boosted to 96.5% via the method of transfer learning [16], where the gesture recognition model trained via a simulated dataset is further fine-tuned with a small amount of unlabeled real measurements according to the adversarial discriminative domain adaptation (ADDA) method in [17]. The main advantages of the proposed CASTER simulator are summarized below:

- Conventional measurement of training dataset for wireless HMR is replaced by channel simulation and gesture video recognition, saving the significant cost of real experiments.
- In the proposed CASTER simulator, the locations of the signal transmitter, sensing receiver, target hand, and scattering clusters can be adjusted freely to adapt to heterogeneous sensing scenarios.

As a result, the proposed CASTER simulator has the potential to customize the gesture recognition models for heterogeneous scenarios without real measurements.

The remainder of this paper is organized as follows. The simulator framework is elaborated in Section II. The channel generator is presented in Section III, and the video gesture catcher is presented in Section IV. The performance of the CASTER simulator is evaluated in Section V. Finally, the conclusion is drawn in Section VI.

In this paper, we use the following notations: non-bold letters are used to denote scalar values, bold lowercase letters (e.g., $\mathbf{a}$) are used to denote column vectors, bold uppercase letters (e.g., $\mathbf{A}$) are used to denote matrices, $|\mathbf{a}|$ and $\mathbf{a}^T$ denote the L2-norm and transpose of vector $\mathbf{a}$.

II. SIMULATOR FRAMEWORK

The proposed CASTER simulator is developed with the primitive-based hand model. In order to extract high-fidelity channel impulse responses from existing videos, the CASTER simulator is composed of the channel generator and video gesture catcher. The former generates a sequence of channel impulse response snapshots given arbitrary hand gestures and arbitrary locations of the transmitter and receiver. The latter captures the parameters of real hand motions from existing videos as the former's input. As a result, the CASTER simulator is able to provide datasets for the training of the hand gesture recognition model without real channel measurement.

As depicted in Fig. 1, the locations of the transmitter, receiver and the target hand can be arbitrary in the channel generator. A gesture is represented as a sequence of snapshots, with an interval of Δt_s seconds. In each snapshot, the channel is assumed to be quasi-static, and the channel impulse response is calculated via the primitive-based method [10]. Particularly, the hand is modeled via 21 keypoints (joints) and 21 ellipsoids (primitives) connecting the keypoints. The non-line-of-sight (NLoS) channel components via the hand can be approximated by the 21 rays respectively scattered off the centers of all primitives. Hence, the channel impulse response of one snapshot can be obtained by aggregating all the rays from the transmitter to the receiver, including the line-of-sight (LoS) ray, the NLoS ones scattered off the target hand, and the others scattered at the environment.

As a remark notice, the 21-keypoint hand model is widely recognized in the fields of computer vision and biomedical engineering [18]. The renowned hand models, such as *openpose* [19], *mediaipipe* [20], and *MANO* [21], are all based on this 21-keypoint representation. It could provide the same degrees of freedom in describing the complex hand and finger motions as explained in [18]: a human hand consists of 21 joints, yielding 27 degrees of freedom, which are the same as the 21-keypoint hand model.

Moreover, the proposed video gesture catcher first extracts the 3-dimensional (3D) coordinates of hand keypoints from each video frame in a local hand world coordination system
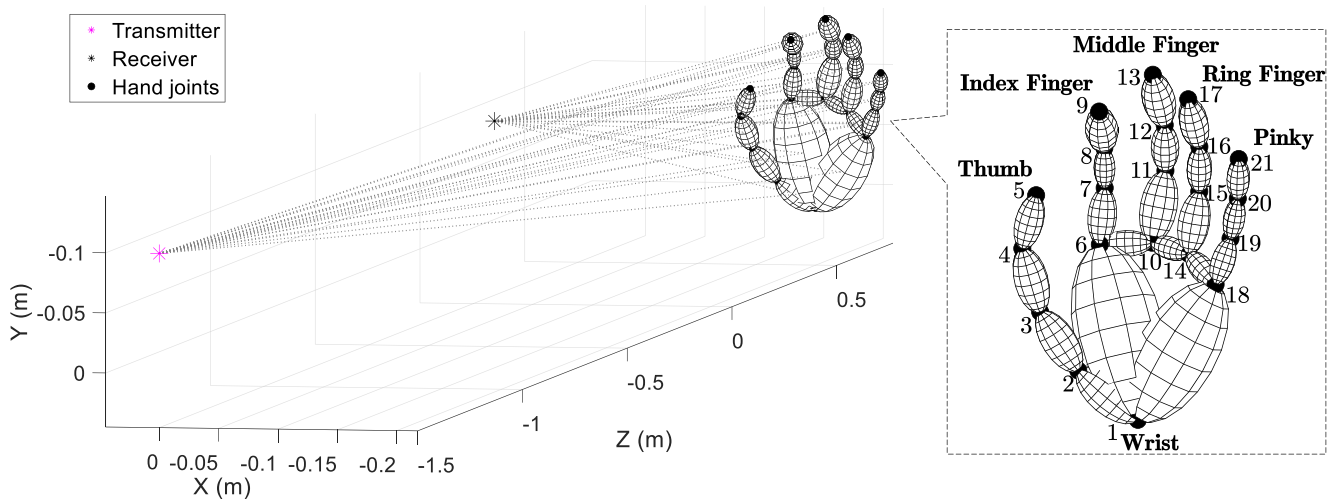
FIGURE 1. Illustration of primitive-based hand model and channel simulation scenario.

via machine learning technique, converts the trajectories of the keypoints from the local hand world coordinate system to a global camera coordinate system and then eliminates the fake hops and jitters of trajectories via low-pass filtering. Finally, since the interval between two video frames, denoted as Δt_v, is usually much larger than Δt_s, an interpolation is necessary to fill a sufficient number of snapshots between two video frames. As a remark, the video clips for the gesture catcher can be recorded in arbitrary environment as long as the desired hand gestures can be identified by the gesture catcher. Hence, they could be obtained from massive online sources.

III. CHANNEL GENERATOR

Without loss of generality, the generation of channel impulse response for the t-th snapshot ($\forall t$) is elaborated in this section. As shown in Fig. 1, the rays from the transmitter to the receiver can be categorized into two parts: target-unrelated components and target-related components. The former refers to the LoS ray and the NLoS rays scattered at the static environment, and the latter refers to the NLoS rays scattered off the target hand. Particularly, let $h(\tau, t)$ and $u(\tau, t)$ be the overall channel impulse response and target-related channel impulse response of the t-th snapshot, $v(\tau)$ be time-invariant target-unrelated channel impulse response. Following the channel model in [9], we have

$$h(\tau, t) = u(\tau, t) + v(\tau), \tag{1}$$

where the generation of $u(\tau, t)$ and $v(\tau)$ is elaborated in the following parts respectively.

A. TARGET-RELATED CHANNEL COMPONENTS

Let $\mathbf{p}_t$ and $\mathbf{p}_r$ be the coordinates of the transmitter and the receiver respectively, $\mathbf{p}_i(t)$ and $\mathbf{p}_j(t)$ be the coordinates of the two joints associated with the n-th primitive in the t-th snapshot ($\forall n, t$). Hence, the center of the n-th primitive is $\mathbf{p}_n^c(t) = [\mathbf{p}_i(t) + \mathbf{p}_j(t)]/2$. As previously mentioned, each

primitive is modeled as an ellipsoid, the length of the axis connecting the two joints is denoted as $2l_n(t)$, where

$$l_n(t) = |\mathbf{p}_i(t) - \mathbf{p}_j(t)|/2. \tag{2}$$

Moreover, the lengths of the other two axes are identical, denoted as $2r_n(t)$. Usually, $r_n(t) < l_n(t)$, and we choose $r_n(t) = l_n(t)/2$. Hence, we shall refer to the axis connecting the two joints as the long axis of the ellipsoid. As a remark note that the primitive size (r_n and l_n) varies slightly over time due to the non-rigid nature of human motion.

Let $R_t^n(t) = |\mathbf{p}_t - \mathbf{p}_n^c(t)|$ be the distance between the transmitter and the n-th primitive center, $R_r^n(t) = |\mathbf{p}_r - \mathbf{p}_n^c(t)|$ be the distance between the receiver and the n-th primitive center, $G_t^n(t)$ and $G_r^n(t)$ be the transmit and receive antenna gains at the directions of incident ray $\mathbf{p}_t - \mathbf{p}_n^c(t)$ and scattered ray $\mathbf{p}_n^c(t) - \mathbf{p}_r$, $\sigma_n(t)$ be the bistatic radar cross section (RCS) of the n-th primitive, c be the speed of light, f_c and λ be the carrier frequency and wavelength respectively. The response of the path scattered off the n-th primitive can be expressed as
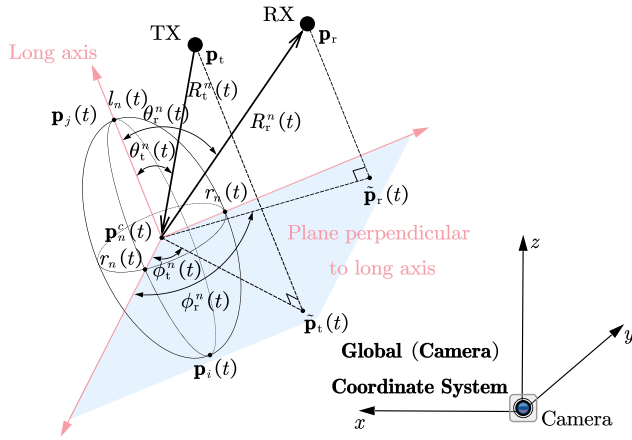
$$u_n(\tau, t) = \lambda \sqrt{\frac{\sigma_n(t) G_t^n(t) G_r^n(t)}{(4\pi)^3 \left(R_t^n(t) R_r^n(t)\right)^2}} e^{-j\phi_n(t)} \delta(\tau - \tau_n(t)), \tag{3}$$

where $\delta(a)$ is the impulse function, whose value is 1 when $a = 0$ and 0 otherwise, while $\tau_n(t) = [R_t^n(t) + R_r^n(t)]/c$ and $\phi_n(t) = 2\pi f_c \tau_n(t)$ measure the delay and phase shift.

Moreover, the calculation of the bistatic RCS $\sigma_n(t)$ follows the method in [22], [23]. As depicted in Fig. 2, let $\theta_t^n(t)$ and $\theta_r^n(t)$ represent the incident and scattered elevation angles respectively, $\phi_t^n(t)$ and $\phi_r^n(t)$ represent the incident and scattered azimuth angles respectively, $\mathbf{v}_n(t) = [\mathbf{p}_i(t) - \mathbf{p}_j(t)]/(2l_n(t))$ represent the normalized vector along the long axis, we have

$$\theta_t^n(t) = \arccos\left(\left(\mathbf{p}_n^c(t) - \mathbf{p}_t\right)^T \mathbf{v}_n(t)/R_t^n(t)\right), \tag{4}$$

$$\theta_r^n(t) = \arccos\left(\left(\mathbf{p}_n^c(t) - \mathbf{p}_r\right)^T \mathbf{v}_n(t)/R_r^n(t)\right), \tag{5}$$

FIGURE 2. Bistatic RCS estimation for the *n*-th primitive.

and

$$|\phi_{\text{r}}^n(t) - \phi_{\text{t}}^n(t)| = \arccos\left(\frac{\left(\mathbf{p}_n^{\text{c}}(t) - \tilde{\mathbf{p}}_{\text{t}}(t)\right)^T\left(\mathbf{p}_n^{\text{c}}(t) - \tilde{\mathbf{p}}_{\text{r}}(t)\right)}{|\mathbf{p}_n^{\text{c}}(t) - \tilde{\mathbf{p}}_{\text{t}}(t)||\mathbf{p}_n^{\text{c}}(t)) - \tilde{\mathbf{p}}_{\text{r}}(t)|}\right),$$

(6)

where

$$\tilde{\mathbf{p}}_{\text{t}}(t) = \mathbf{p}_{\text{t}} - \mathbf{v}_n(t)\left(\mathbf{p}_{\text{t}} - \mathbf{p}_n^{\text{c}}(t)\right)^T \mathbf{v}_n(t)$$

and

$$\tilde{\mathbf{p}}_{\text{r}}(t) = \mathbf{p}_{\text{r}} - \mathbf{v}_n(t)\left(\mathbf{p}_{\text{r}} - \mathbf{p}_n^{\text{c}}(t)\right)^T \mathbf{v}_n(t)$$

denotes the projection of the transmitter and receiver's locations on the plane containing the center of the *n*-th ellipsoid and perpendicular to its long axis in the *t*-th snapshot. As a result, the bistatic RCS $\sigma_n(t)$ of *n*-th ellipsoid in the *t*-th snapshot is given by (7), shown at the bottom of the page.

Aggregating the NLoS rays scattered off all the primitives, the target-related channel impulse response can be written as

$$u(\tau, t) = \sum_{n=1}^{21} u_n(\tau, t).$$

(8)

B. TARGET-UNRELATED CHANNEL COMPONENTS

CASTER simulator models the environment by K static scatterers. Let the RCS, transmit and receive antenna gains and the distance of the k-th NLoS ray be σ_k, G_{t}^k, G_{r}^k, R_{t}^k, and R_{r}^k, respectively. The NLoS components of target-unrelated channel impulse response can be written as

$$v_{\text{NLoS}}(\tau) = \sum_{k=1}^{K} \lambda\sqrt{\frac{\sigma_k G_{\text{t}}^k G_{\text{r}}^k}{(4\pi)^3\left(R_{\text{t}}^k R_{\text{r}}^k\right)^2}}e^{-\text{j}\phi_k}\delta(\tau - \tau_k),$$

(9)

where $\tau_k = (R_{\text{t}}^k + R_{\text{r}}^k)/c$ and $\phi_k = 2\pi f_{\text{c}}\tau_k$.

Moreover, let transmit and receive antenna gains at the direction of LoS path be $G_{\text{t, LoS}}$ and $G_{\text{r, LoS}}$, distance between transmitter and receiver be R_{LoS}, the LoS component of target-unrelated channel is modeled via the following free space model:

$$v_{\text{LoS}}(\tau) = \frac{\lambda\sqrt{G_{\text{t, LoS}}G_{\text{r, LoS}}}}{4\pi R_{\text{LoS}}}e^{-\text{j}\phi_{\text{LoS}}}\delta(\tau - \tau_{\text{LoS}}),$$

(10)

where $\tau_{\text{LoS}} = R_{\text{LoS}}/c$ and $\phi_{\text{LoS}} = 2\pi f_{\text{c}}\tau_{\text{LoS}}$. As a result, according to [9], the target-unrelated channel impulse response can be written as

$$v(\tau) = v_{\text{LoS}}(\tau) + v_{\text{NLoS}}(\tau).$$

(11)

IV. VIDEO GESTURE CATCHER

As mentioned in the previous section, the motion of the target hand is characterized by the trajectories of the 21 keypoints in a sequence of snapshots, denoted as $\mathbf{p}_i(t)$, $i = 1, 2, \ldots, 21$. We leverage the tool of *Mediapipe* [20] to extract the keypoint trajectories from videos of monocular cameras, where two issues in the conversion are addressed in this section. The *Mediapipe* could localize the positions of the keypoints in each video frame. The positions are represented in the coordinate system with the origin at the hand center, namely hand world coordinate system. However, it is difficult to calculate the Doppler frequency with such coordinate system, as the hand center is moving. Hence, we first transfer the coordinates to a unified coordinate system by solving the Perspective-n-Point (PnP) problem [24], where the fake hops on the trajectories are smoothed. Moreover, because there are usually 30 video frames per second, which is not sufficient for estimating the Doppler frequencies of gesture. For example, the typical Doppler frequencies of gestures on the 60 GHz signals are around 800 Hz (assuming a maximum radial velocity of 4 meters per second), which requests 1600 snapshots per second at least. Hence, interpolation is introduced such that the channel impulse response can be generated with a shorter interval.

A. CONVERSION OF COORDINATE SYSTEMS

For the elaboration convenience, we first introduce the following three coordinate systems. The two-dimensional (2D) pixel coordinate system in the unit of pixels is used to identify the positions of hand keypoints in each video frame. The origin of the pixel coordinate system is usually at the upper left corner of each frame, as shown in Fig. 3. The three-dimensional (3D) hand world coordinate system in the unit of meters measures the positions of hand keypoints in the real world with respect to the hand center. Moreover,

$$\sigma_n(t) = \frac{4\pi r_n^4(t)l_n^2(t)\left[\left(1 + \cos\theta_{\text{t}}^n(t)\cos\theta_{\text{r}}^n(t)\right)\cos\left(\phi_{\text{r}}^n(t) - \phi_{\text{t}}^n(t)\right) + \sin\theta_{\text{t}}^n(t)\sin\theta_{\text{r}}^n(t)\right]^2}{\left[r_n^2(t)\left(\sin^2\theta_{\text{t}}^n(t) + \sin^2\theta_{\text{r}}^n(t) + 2\sin\theta_{\text{t}}^n(t)\sin\theta_{\text{r}}^n(t)\cos(\phi_{\text{r}}^n(t) - \phi_{\text{t}}^n(t))\right) + l_n^2(t)\left(\cos\theta_{\text{t}}^n(t) + \cos\theta_{\text{r}}^n(t)\right)^2\right]^2}.$$

(7)

FIGURE 3. Illustration of three coordinate systems.

(a) Pixel coordinate system   (b) Hand world coordinate system   (c) Camera coordinate system
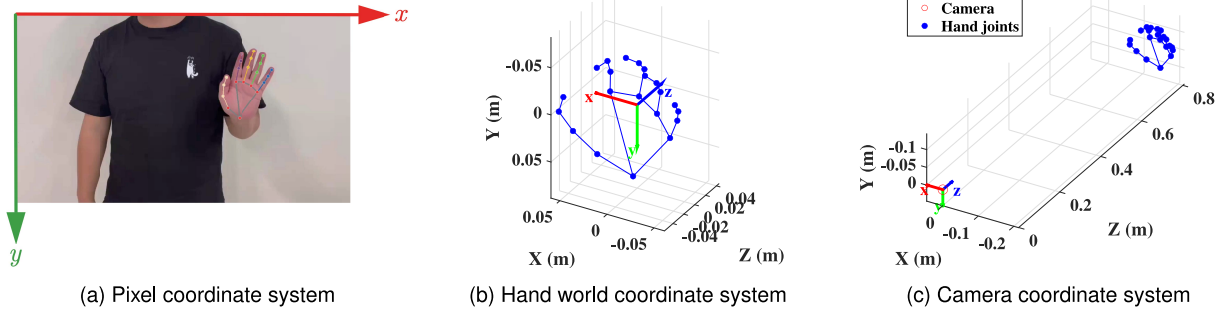
the 3D camera coordinate system in the unit of meters measures the positions of hand keypoints with respect to the static camera lens, which captures the videos. The *Mediapipe* is able to identify the 21 keypoints, localize them in the first two coordinate systems. Because the hand center is usually in motion and the camera is static, the trajectories in the camera coordinate system instead of in the hand world coordinate system, could be used to calculate the Doppler frequencies. Thus, the coordinates of hand keypoints $\mathbf{p}_i(t)$, $i = 1, 2, \ldots, 21$, transmitter $\mathbf{p}_t$ and receiver $\mathbf{p}_r$, defined in the previous section should be measured in the camera coordinate system. The above three coordinate systems are illustrated in Fig. 3, as referenced.

Define the coordinates of the i-th keypoint ($i = 1, 2, \ldots, 21$) in the pixel, hand world and camera coordinate systems as (u_i, v_i), $(x_i^{\mathrm{w}}, y_i^{\mathrm{w}}, z_i^{\mathrm{w}})$, and (x_i, y_i, z_i), respectively, where the snapshot index t is ignored in this section for the simplicity of elaboration. Let f be the focal length in the unit of pixels, (c_x, c_y) be the coordinates of image center in the pixel coordinate system, we define the camera intrinsic matrix $\mathbf{A}$ as

$$\mathbf{A} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{12}$$

Hence, the relation between the 2D pixel and 3D camera coordinate systems can be expressed as

$$z_i[u_i \ v_i \ 1]^T = \mathbf{A}[x_i \ y_i \ z_i]^T. \tag{13}$$

Let $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ and $\mathbf{t}$ be the rotation matrix and translation vector from hand world coordinate system to camera coordinate system, we define the camera extrinsic matrix $\mathbf{T}$ and perspective projection matrix Π as follows:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}, \tag{14}$$

$$\Pi = \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{1}_{3 \times 1} \end{bmatrix}, \tag{15}$$

where $\mathbf{I}_{3 \times 3}$ denotes a 3×3 identity matrix, $\mathbf{0}_{1 \times 3}$ and $\mathbf{1}_{3 \times 1}$ are the three-dimensional row and column vectors with all 0 and 1 entries respectively. According to [24], the relations between the hand world and camera coordinate systems are given by

$$[x_i \ y_i \ z_i \ 1]^T = \mathbf{T}[x_i^{\mathrm{w}} \ y_i^{\mathrm{w}} \ z_i^{\mathrm{w}} \ 1]^T. \tag{16}$$

As a result, the relation between the hand world and the pixel coordinate system could be described as

$$z_i[u_i \ v_i \ 1]^T = \mathbf{A}\Pi\mathbf{T}[x_i^{\mathrm{w}} \ y_i^{\mathrm{w}} \ z_i^{\mathrm{w}} \ 1]^T. \tag{17}$$

For the elaboration convenience, we denote the projection from the hand world coordinate system to the pixel coordinate system as the following function $\mathcal{P}$:

$$
\begin{aligned}
[u_i \ v_i]^T &= \mathcal{P}\left([x_i^{\mathrm{w}} \ y_i^{\mathrm{w}} \ z_i^{\mathrm{w}}]^T, \mathbf{R}, \mathbf{t}, \mathbf{A}\right) \\
&= \frac{1}{z_i}[\mathbf{I}_{2 \times 2} \ \mathbf{0}_{2 \times 1}]\mathbf{A}\underbrace{\left(\mathbf{R}[x_i^{\mathrm{w}} \ y_i^{\mathrm{w}} \ z_i^{\mathrm{w}}]^T + \mathbf{t}\right)}_{=[x_i \ y_i \ z_i]^T}.
\end{aligned} \tag{18}
$$

The *Mediapipe* could provide the coordinates (u_i, v_i) and $(x_i^{\mathrm{w}}, y_i^{\mathrm{w}}, z_i^{\mathrm{w}})$ of all the keypoints ($i = 1, 2, \ldots, 21$) in each video frame. Hence, their coordinates in the camera coordinate system can be calculated with the knowledge of the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$.

In fact, the parameters in the camera intrinsic matrix $\mathbf{A}$ can be measured in advance, the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ can be estimated via (18) for $i = 1, 2, \ldots, 21$. Particularly, given the coordinates of the 21 keypoints in the pixel and hand world coordinate systems, the detection of the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ can be formulated as follows.

$$\min_{\mathbf{R}, \mathbf{t}} \sum_{i=1}^{21} \left| (u_i, v_i) - \mathcal{P}\left([x_i^{\mathrm{w}} \ y_i^{\mathrm{w}} \ z_i^{\mathrm{w}}]^T, \mathbf{R}, \mathbf{t}, \mathbf{A}\right) \right|^2,$$
$$\text{s.t. } \mathbf{R}(\mathbf{R})^T = \mathbf{I}_{3 \times 3}, \ \det(\mathbf{R}) = 1, \tag{19}$$

where $\det(.)$ represents the determinant of a matrix.

The above problem is referred to as the Perspective-n-Point (PnP) problem [24]. It can be solved via the *cv2.solvePnP* function from the popular computer vision library *OpenCV* [25], where the Levenberg-Marquardt optimization method [26] is adopted.

B. MOTION SMOOTHING AND SNAPSHOT INTERPOLATION

Because of the errors of keypoint detection with *Mediapipe*, there might be fake hops or jitters in the detected trajectories of keypoints, which do not exist actually. This will lead to
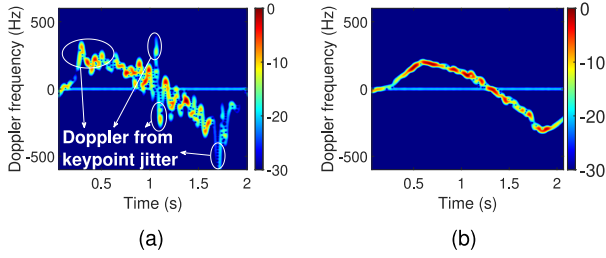
FIGURE 4. Comparison of simulated spectrograms via CASTER. (a) before one-euro filter smoothing; (b) after one-euro filter smoothing.

the false alarm of high Doppler frequencies (as depicted in Fig. 4). In order to generate a high-fidelity dataset for gesture recognition model training, a low-pass filter, namely one-euro filter [27], is proposed to smooth both trajectories and velocities, followed by snapshot interpolation between neighboring video frames.

Let $\mathbf{q}_{i,k} = [x_{i,k} \; y_{i,k} \; z_{i,k}]^T$ and $\hat{\mathbf{q}}_{i,k} = [\hat{x}_{i,k} \; \hat{y}_{i,k} \; \hat{z}_{i,k}]^T$ be the positions of the i-th keypoint in the k-th frame before and after the low-pass filtering respectively, $\dot{\mathbf{q}}_{i,k} = [\dot{x}_{i,k} \; \dot{y}_{i,k} \; \dot{z}_{i,k}]^T$ and $\hat{\dot{\mathbf{q}}}_{i,k} = [\hat{\dot{x}}_{i,k} \; \hat{\dot{y}}_{i,k} \; \hat{\dot{z}}_{i,k}]^T$ be the estimated velocities of the i-th keypoint in the k-th frame before and after the low-pass filtering respectively. Initializing $\hat{\mathbf{q}}_{i,1}$ with $\mathbf{q}_{i,1}$, the trajectory smoothing for the i-th keypoint in the k-th frame is given by

$$\hat{o}_{i,k} = \alpha_{i,k} o_{i,k} + \left(1 - \alpha_{i,k}\right)\hat{o}_{i,k-1}, \quad \forall i, k \geq 2 \qquad (20)$$

where the notation o represents the dimensions of x, y and z, respectively, and

$$\alpha_{i,k} = \frac{1}{1 + \frac{1}{2\pi \Delta t_{\mathrm{v}}\left(f_{c_{\min}} + \beta|\hat{\dot{o}}_{i,k}|\right)}}$$

is the smoothing factor, Δt_{v} is the video frame interval, $f_{c_{\min}}$ is the minimum cutoff frequency, β is the speed coefficient of update. Moreover, the velocity in the above equation can be calculated as

$$\hat{\dot{o}}_{i,k} = \gamma \dot{o}_{i,k} + (1 - \gamma)\hat{\dot{o}}_{i,k-1}, \quad \forall i, k \geq 2 \qquad (21)$$

where $\dot{o}_{i,k} = (o_{i,k} - \hat{o}_{i,k-1})/\Delta t_{\mathrm{v}}$, $\hat{\dot{o}}_{i,1}$ is initialized with 0, γ is the fixed smoothing factor.

The overall smoothing procedure via one-euro filter is illustrated in Alg. 1. In fact, the smoothing of the i-th keypoint's velocity $\hat{\dot{o}}_{i,k}$ and trajectory $\hat{o}_{i,k}$ in the k-th frame is conducted by repeating two first-order low-pass filters (21) and (20) to the position and velocity of the i-th keypoint. This procedure effectively eliminates false hops or jitters in the detected keypoint trajectories while preserving the motion features. An example of the smoothing result is shown in Fig. 4.

Finally, we adopt the cubic spline interpolation method [28] to insert $\Delta t_{\mathrm{v}}/\Delta t_{\mathrm{s}} - 1$ positions of the i-th keypoint ($\forall i$) between every two neighboring frames (say $\hat{\mathbf{q}}_{i,k}$ and $\hat{\mathbf{q}}_{i,k+1}$, $\forall k$), and denote the position of the i-th keypoint in the t-th snapshot as $\mathbf{p}_i(t)$.

Algorithm 1 One-Euro Low-Path Filter for Keypoint Trajectory Smoothing

1: **Input:**
- $\{\mathbf{q}_{i,k} = [x_{i,k} \; y_{i,k} \; z_{i,k}]^T | i \in \{1, \ldots, 21\}, k \in \{1, \ldots, K\}\}$, where $\mathbf{q}_{i,k}$ denotes the location of the i-th keypoint in the k-th frame.
- $f_{c_{\min}}$: Minimum cutoff frequency for position.
- β: Speed coefficient.
- γ: Smoothing factor for velocity.
- Δt_{v}: Video frame interval.

2: **Output:**
- $\{\hat{\mathbf{q}}_{i,k} = [\hat{x}_{i,k} \; \hat{y}_{i,k} \; \hat{z}_{i,k}]^T | i \in \{1, \ldots, 21\}, k \in \{1, \ldots, K\}\}$: where $\hat{\mathbf{q}}_{i,k}$ denotes the location of the i-th keypoint in the k-th frame after smoothing.

3: **for** $k \leftarrow 2$ to K **do**   ▷ Iteration over frames.
4:     **for** $i \leftarrow 1$ to 21 **do**   ▷ Iteration over keypoints.
5:         **for** o represents the dimensions of x, y, and z respectively **do**
6:             $\hat{o}_{i,1} \leftarrow o_{i,1}, \; \hat{\dot{o}}_{i,1} \leftarrow 0$
7:             $\dot{o}_{i,k} = (o_{i,k} - \hat{o}_{i,k-1})/\Delta t_{\mathrm{v}}$
8:             $\hat{\dot{o}}_{i,k} = \gamma \dot{o}_{i,k} + (1-\gamma)\hat{\dot{o}}_{i,k-1}$ ▷ Equation (21): smooth velocity.
9:             $\alpha_{i,k} = \frac{1}{1 + \frac{1}{2\pi \Delta t_{\mathrm{v}}(f_{c_{\min}} + \beta|\hat{\dot{o}}_{i,k}|)}}$ ▷ Update smoothing factor for position.
10:             $\hat{o}_{i,k} = \alpha_{i,k} o_{i,k} + (1-\alpha_{i,k})\hat{o}_{i,k-1}$ ▷ Equation (20): smooth position.
11:         **end for**
12:         $\hat{\mathbf{q}}_{i,k} = [\hat{x}_{i,k} \; \hat{y}_{i,k} \; \hat{z}_{i,k}]^T$
13:     **end for**
14: **end for**
15: **return** $\{\hat{\mathbf{q}}_{i,k} = [\hat{x}_{i,k} \; \hat{y}_{i,k} \; \hat{z}_{i,k}]^T | i \in \{1, \ldots, 21\}, k \in \{1, \ldots, K\}\}$

V. EVALUATION OF CASTER SIMULATOR

In this section, the high fidelity of the CASTER simulator in the applications of gesture recognition is demonstrated. Specifically, the generation of gesture datasets via CASTER simulator and real measurement is first elaborated. Then, the recognition performance via the above two datasets is discussed.

A. SIMULATION AND EXPERIMENTAL DATASETS

In order to verify the quality of the dataset generated by CASTER simulator, 500 clips of videos on 5 gestures, including "Pushing and Pulling", "Beckoning", "Rubbing Fingers", "Plugging" (slicing forward with all fingers together), and "Scaling" (spreading thumb, index finger, middle finger) were recorded using a normal monocular camera at a rate of 30 frames per second (fps). The motion data for hand model is then extracted via the video gesture catcher.

On the other hand, in the channel generator, the locations of transmitter, receiver and target hand
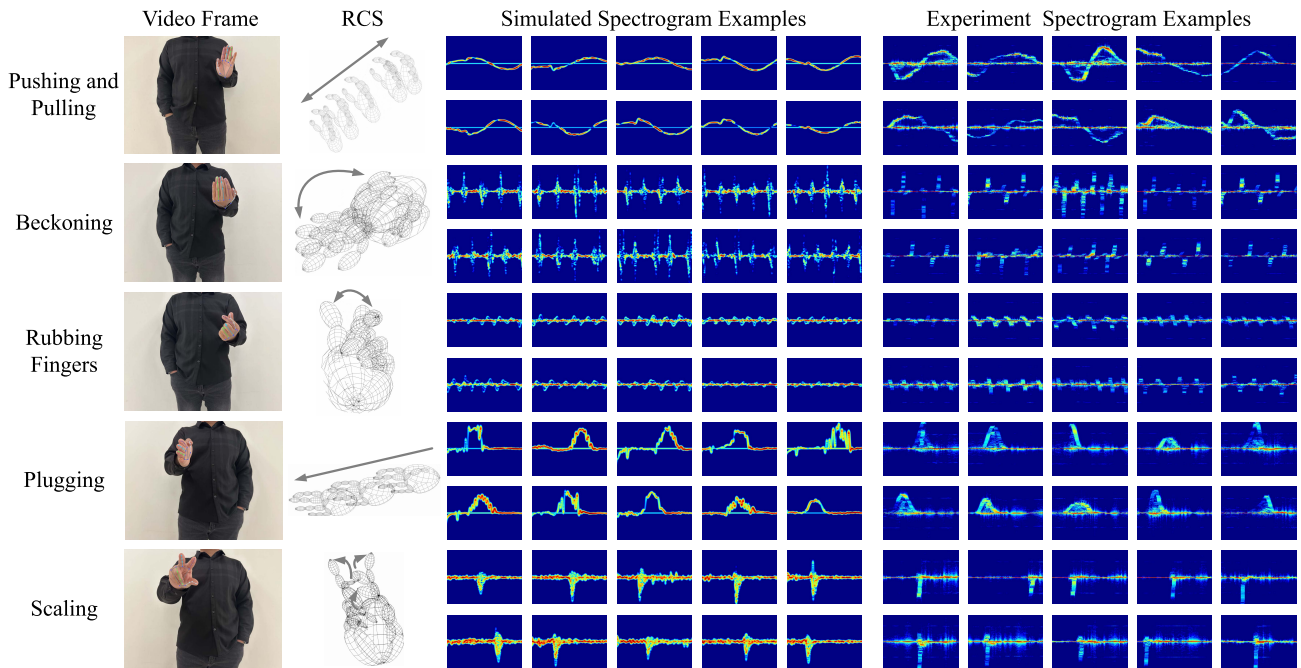
FIGURE 5. Illustration of the simulated and experimental dataset from CASTER, where some examples of spectrogram are plotted.

center are $[0m, -0.1m, -1.5m]$, $[0.2m, -0.1m, 0.1m]$, and $[0m, 0m, 0.4 \sim 0.8m]$, respectively. Moreover, in order to model the target-unrelated channel, K static RCSs are randomly generated from a normal distribution with a mean value of $0.005\,m^2$ and a standard deviation of $0.001\,m^2$. These RCSs are associated with scatterers that are randomly located within a $2\,m \times 2\,m \times 2\,m$ cuboid centered at the receiver. The positions of these scatterers are used to calculate the associated parameters G_t^k, G_r^k, R_t^k, and R_r^k.

Thus, 100 sequences of channel impulse responses for each gesture are obtained via the proposed CASTER simulator with a sampling rate of 2000 snapshots per second. Then, one spectrogram, illustrating the Doppler frequency versus time, is calculated for each video clip (each sequence of channel impulse responses) by applying the short-time Fourier transform (STFT) with a window of 0.125 seconds (250 snapshots). As a result, a simulated dataset of 500 spectrograms for the recognition of 5 gestures is obtained as illustrated in Fig. 5.

In order to measure the real Doppler spectrum of gestures, an integrated passive sensing and communication system working on millimeter wave (mmWave) band is developed as in our previous work [7]. As illustrated in Fig. 6, at the transmitter, an NI USRP-2954R [29] is utilized to generate an intermediate frequency (IF) signal at 500 MHz, which is subsequently up-converted to 60 GHz and transmitted using a Sivers 60 GHz phased array [30]. At the receiver, two phased arrays are connected to a single USRP device to receive signals from the reference and surveillance channels, respectively. The transmit mmWave signal is modulated via orthogonal frequency-division multiplexing (OFDM). The
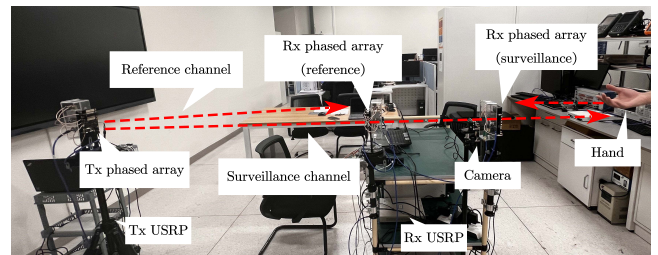

FIGURE 6. Facilities and scenario of experiment.

carrier frequency is 60.48 GHz and the signal bandwidth is 5 MHz.

In the experiment, the locations of the transmitter and receiver are consistent with those in the simulator. 100 trials are measured for each gesture via the passive sensing system. Following the signal processing in [7], the spectrogram of hand gestures can be computed through the cross-ambiguity function (CAF). As a result, an experimental dataset with 100 spectrograms per gesture is obtained, as illustrated in Fig. 5.

B. PERFORMANCE OF GESTURE RECOGNITION

First of all, it can be observed from Fig. 7 that the spectrograms from real experiment and CASTER simulator exhibit similar time-Doppler patterns. To further demonstrate the high fidelity of the proposed simulator in the applications of gesture recognition, the following six training and testing schemes are adopted with the same image recognition model named ResNet18 [31]:
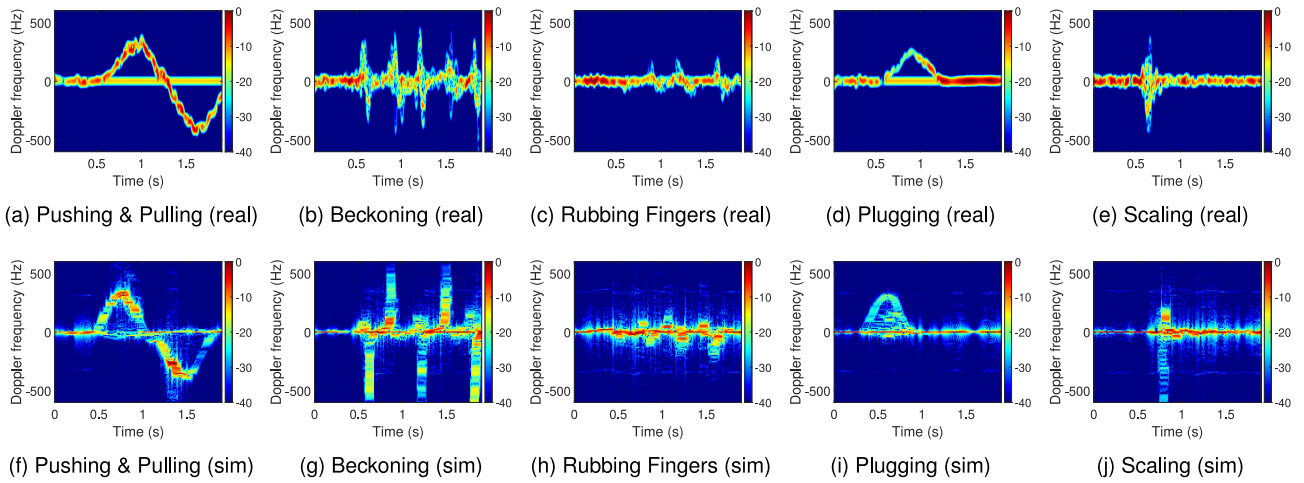
(a) Pushing & Pulling (real)   (b) Beckoning (real)   (c) Rubbing Fingers (real)   (d) Plugging (real)   (e) Scaling (real)

(f) Pushing & Pulling (sim)   (g) Beckoning (sim)   (h) Rubbing Fingers (sim)   (i) Plugging (sim)   (j) Scaling (sim)

FIGURE 7. Spectrogram comparison of 5 gestures generated by CASTER (first row) and the experiment (second row).

FIGURE 8. Gesture recognition accuracy of the 6 training and testing schemes.

- *Scheme* 1*:* The training set consists of 60 simulated spectrograms for each gesture, and the test set consists of 40 measured ones for each gesture;
- *Scheme* 2*:* The training set consists of 50 simulated spectrograms and 10 measured ones for each gesture, and the test set consists of 40 measured ones for each gesture;
- *Scheme* 3*:* The training set consists of 40 simulated spectrograms and 20 measured ones for each gesture, and the test set consists of 40 measured ones for each gesture;
- *Scheme* 4*:* The training set consists of 30 simulated spectrograms and 30 measured ones for each gesture, and the test set consists of 40 measured ones for each gesture;
- *Scheme* 5*:* The training set consists of 60 measured spectrograms for each gesture, and the test set consists of 40 measured ones for each gesture;
- *Scheme* 6*:* The training set consists of 60 simulated spectrograms for each gesture, and the test set consists of 40 simulated ones for each gesture.

The overall results of the gesture recognition are shown in Fig. 8, and the confusion charts of the 6 schemes are shown in Fig. 9 respectively. It can be observed that an accuracy of 83.0% (Scheme 1) can be achieved if the simulated dataset is used for training and the experimental dataset is used

for testing. On the other hand, there is still roughly 16.0% and 17.0% performance loss compared with the Scheme 5 and 6, indicating that the difference between simulated and experimental datasets is not negligible. One method to mitigate such difference is to mix some experimental samples into the simulated dataset. It can be observed from the results of Scheme 2, 3, 4 that mixing some experimental samples could significantly improve the testing accuracy. Moreover, it can be observed that the enhanced recognition accuracy converges to 98.5% for Schemes 3 and 4. However, this is still 0.5% lower than the accuracy achieved with Scheme 5. This difference indicates the inherent feature distinctions between simulated and experimental datasets.

Furthermore, it is apparent from the Fig. 9(a) that gesture recognition for "beckoning" and "plugging" are not sufficiently accurate. The recognition accuracy is 70% and 75% respectively. Moreover, 22.5% confusion probability exists between the gestures of "beckoning" and "pushing and pulling", indicating that some of the simulation samples of "beckoning" are similar to the experimental samples of "pushing and pulling".

To qualitatively support the aforementioned observations, we applied dimensionality reduction techniques to the extracted features (network output before entering the fully-connected layer classifier) for the entire simulation and experimental datasets using the ResNet18 model trained by Scheme 1. Specifically, t-distributed Stochastic Neighbor Embedding (t-SNE) [32] and Principal Component Analysis (PCA) [33] were employed to visualize and analyze the high-dimensional features (512 dimensions for ResNet18) of the dataset, as illustrated in Fig. 10. It could be observed that although 83.0% gesture recognition accuracy is achieved, the distributions of the features of different gestures are not sufficiently separated. Moreover, the simulated and experimental features for the gesture "Scaling" are not well aligned, indicating the inherent feature distinctions between simulated and experimental datasets. These could be regarded as the limitation of the proposed simulator, since the
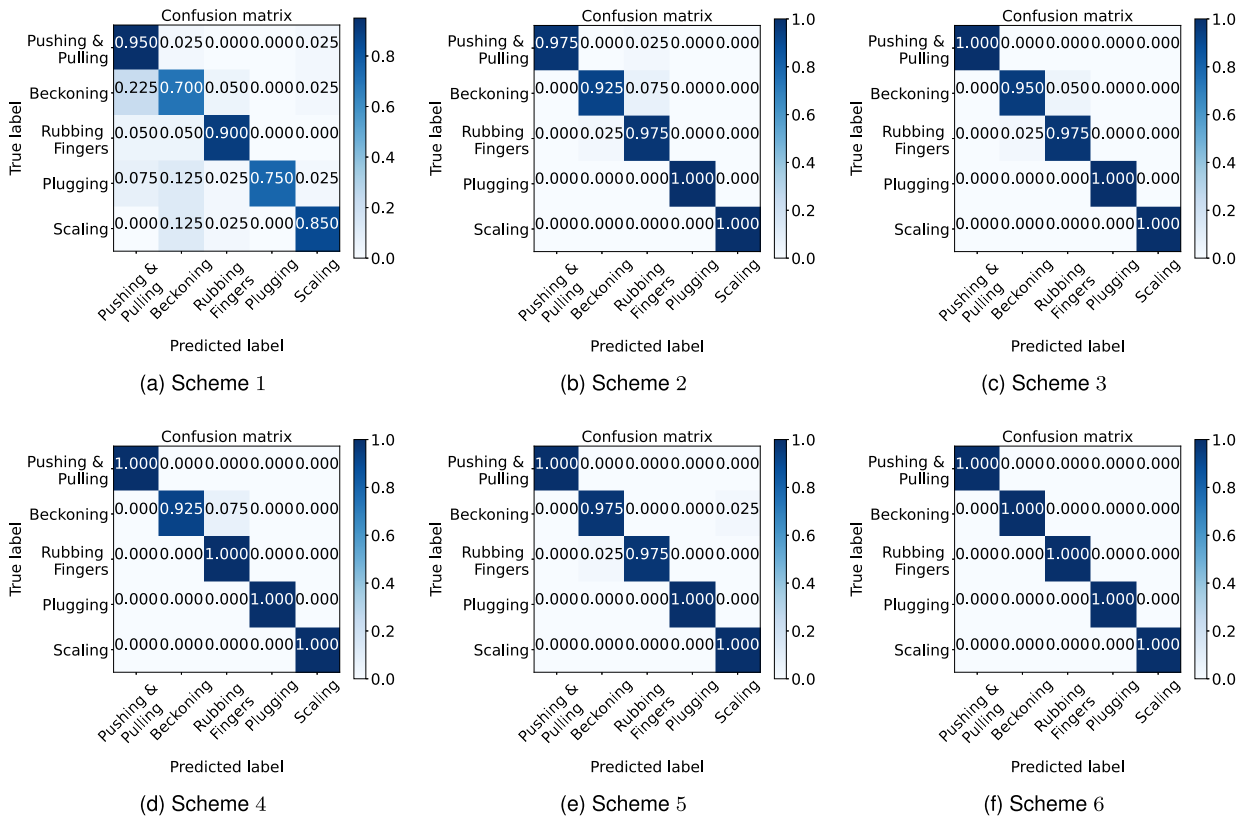
FIGURE 9. Confusion charts of the 6 training and testing schemes.
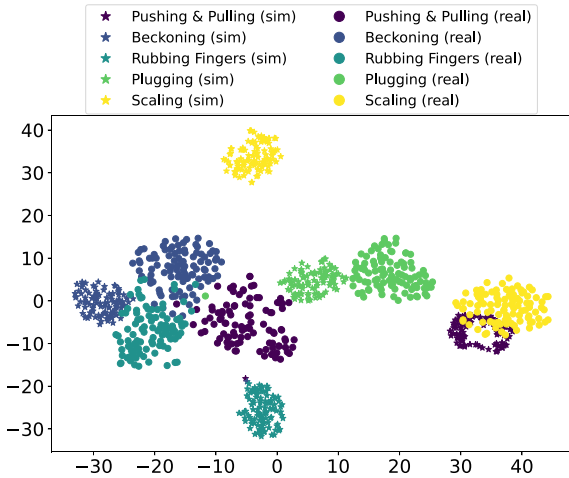

FIGURE 10. t-SNE visualization for the feature spaces of simulation and experimental datasets. Five gesture categories are distinguished by different colors, with simulation sample features denoted by star shapes and experimental sample features represented by solid circle shapes.

real-world channel is more complex than the simulated one due to the impacts of multipath and the non-ideal hardware.

C. IMPROVEMENT VIA TRANSFER LEARNING

The transfer learning technique [16] is applied in this part to relieve the above issue of feature distinction. In this context, the simulated dataset is referred to as the source domain, and the experimental dataset as the target domain. The adversarial discriminative domain adaptation (ADDA) [17] is adopted to align the feature distributions of the source and target domains. The ResNet18 model trained by Scheme 1 in the previous part, serves as the source domain gesture recognition model, and the target domain gesture recognition model is initialized with the same architecture and parameters. Then, additional 50 unlabeled experimental samples are added to the simulated dataset for Scheme 1. This is used to train a domain discriminator to distinguish the source and target domain features and fine-tune the feature extractor part of the target model alternatively, such that the source feature representation is mimicked. The details of the ADDA method can be found in [17]. The confusion chart of the testing result and t-SNE visualization of the feature spaces in simulation and experimental datasets after ADDA are shown in Fig. 11. The recognition accuracy is boosted to 96.5% and the feature spaces of simulation and experimental datasets are well aligned. This result indicates that the feature distinctions between simulated and experimental datasets can be mitigated significantly by transfer learning.

It could be observed that the gesture recognition accuracy after the transfer learning (96.5%) is still lower than that of Scheme 2 (97.5%). This is mainly because in the transfer learning of Scheme 1, the 50 measured data samples are all unlabeled, while Scheme 2 uses labeled experimental and simulated samples. In fact, without the recognition of
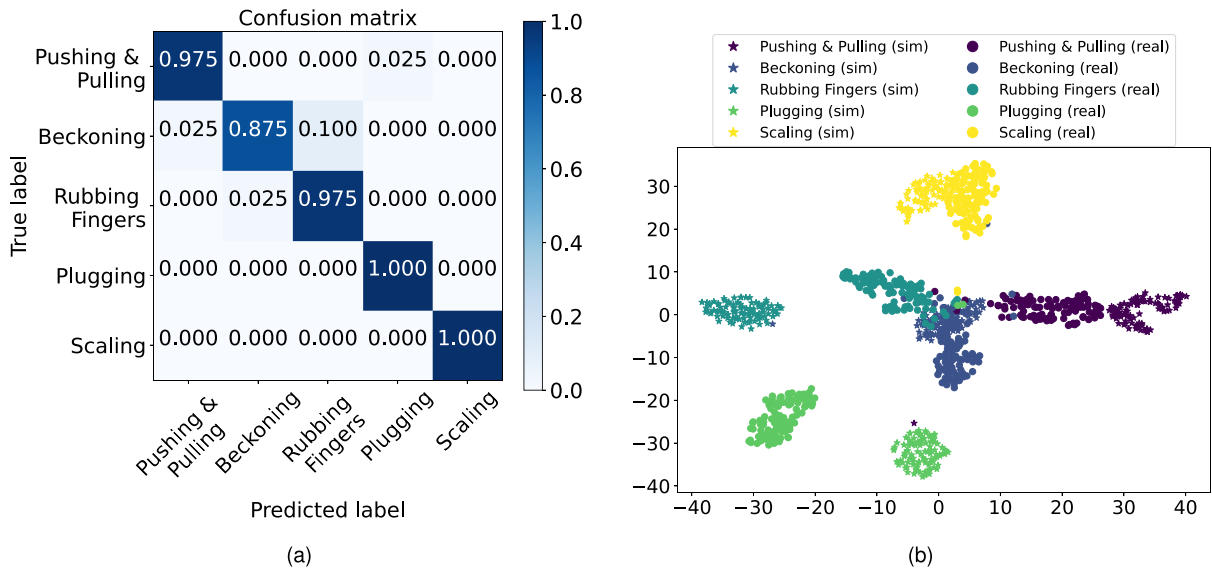
FIGURE 11. Gesture recognition result after ADDA. (a) confusion chart of simulation-to-reality inference; (b) t-SNE visualization of the feature spaces in simulation and experimental datasets.

experimental samples, the unsupervised transfer learning of Scheme 1 would be more practical than Scheme 2, which trains the model with labeled simulated and experimental samples.

VI. CONCLUSION

In this paper, a computer-vision assisted wireless channel simulator, namely CASTER simulator, is proposed to generate high-fidelity dataset for hand gesture recognition. In the simulator, the target hand is modeled by 21 ellipsoid primitives, and the ray-tracing method is adopted to calculate the channel impulse responses. Moreover, a video gesture catcher is proposed to capture real motion data of gestures. In the experiments with 5 different gestures, both real dataset via experiment and simulated dataset via CASTER simulator are obtained. An accuracy of 83.0% can be achieved in simulation-to-reality inference, i.e., using simulated and experimental datasets in model training and inference respectively. Moreover, this accuracy can be boosted to 96.5% by transfer learning, i.e., fine-tuning the gesture recognition model with a few unlabeled real data.

REFERENCES

[1] Y. Ma, G. Zhou, and S. Wang, "WiFi sensing with channel state information: A survey," *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–36, Jun. 2019.

[2] J. Liu, H. Liu, Y. Chen, Y. Wang, and C. Wang, "Wireless sensing for human activity: A survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1629–1645, 3rd Quart., 2020.

[3] Y. Zhang et al., "Widar3.0: Zero-effort cross-domain gesture recognition with Wi-Fi," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 8671–8688, Nov. 2021.

[4] J. W. Smith, S. Thiagarajan, R. Willis, Y. Makris, and M. Torlak, "Improved static hand gesture classification on deep convolutional neural networks using novel sterile training technique," *IEEE Access*, vol. 9, pp. 10893–10902, 2021.

[5] W. Li, R. J. Piechocki, K. Woodbridge, C. Tang, and K. Chetty, "Passive WiFi radar for human sensing using a stand-alone access point," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 3, pp. 1986–1998, Mar. 2021.

[6] H. Sun, L. G. Chia, and S. G. Razul, "Through-wall human sensing with WiFi passive radar," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 4, pp. 2135–2148, Aug. 2021.

[7] J. Li, C. Yu, Y. Luo, Y. Sun, and R. Wang, "Passive motion detection via mmWave communication system," in *Proc. IEEE 95th Veh. Technol. Conf.*, 2022, pp. 1–6.

[8] R. Du et al., "An overview on IEEE 802.11 bf: WLAN sensing," 2023, *arXiv:2310.17661.*

[9] M. Zhang et al., "Channel models for WLAN sensing systems," IEEE, Piscataway, NJ, USA, documents 802.11, Sep. 2021. [Online]. Available: https://mentor.ieee.org/802.11/documents?isdcn=Meihong

[10] G. Li, S. Wang, J. Li, R. Wang, X. Peng, and T. X. Han, "Wireless sensing with deep spectrogram network and primitive based autoregressive hybrid channel model," in *Proc. IEEE 22nd Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2021, pp. 481–485.

[11] G. Li et al., "Integrated sensing and communication from learning perspective: An SDP3 approach," *IEEE Internet Things J.*, vol. 11, no. 4, pp. 5589–5603, Feb. 2024.

[12] "WiGig tools." Accessed: May 17, 2024. [Online]. Available: https://github.com/wigig-tools

[13] S. Vishwakarma, W. Li, C. Tang, K. Woodbridge, R. Adve, and K. Chetty, "SimHumalator: An open-source end-to-end radar simulator for human activity recognition," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 37, no. 3, pp. 6–22, Mar. 2021.

[14] B. Erol, C. Karabacak, S. Z. Gürbüz, and A. C. Gürbüz, "Simulation of human micro-doppler signatures with Kinect sensor," in *Proc. IEEE Radar Conf.*, 2014, pp. 863–868.

[15] R. Boulic, N. M. Thalmann, and D. Thalmann, "A global human walking model with real-time kinematic personification," *Vis. Comput.*, vol. 6, pp. 344–358, Nov. 1990.

[16] F. Zhuang et al., "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021.

[17] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2962–2971.

[18] N. Wheatland, Y. Wang, H. Song, M. Neff, V. Zordan, and S. Jörg, "State of the art in hand and finger modeling and animation," *Comput. Graph. Forum*, vol. 34, no. 2, pp. 735–760, 2015.

[19] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime multi-person 2D pose estimation using part affinity fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 172–186, Jan. 2021.

[20] F. Zhang et al., "MediaPipe hands: On-device real-time hand tracking," 2020, *arXiv:2006.10214*.

[21] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 1–17, Nov. 2017.

[22] E. F. Knott, J. F. Schaeffer, and M. T. Tulley, *Radar Cross Section*. Raleigh, NC, USA: SciTech Publ., 2004.

[23] K. D. Trott, "Stationary phase derivation for RCS of an ellipsoid," *IEEE Antennas Wireless Propag. Lett.*, vol. 6, pp. 240–243, 2007.

[24] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for augmented reality: A hands-on survey," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 12, pp. 2633–2651, Dec. 2016.

[25] "OpenCV: Perspective-n-point (PnP) pose computation." Accessed: May 17, 2024. [Online]. Available: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html

[26] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quart. Appl. Math.*, vol. 2, no. 2, pp. 164–168, 1944.

[27] G. Casiez, N. Roussel, and D. Vogel, "1 € filter: A simple speed-based low-pass filter for noisy input in interactive systems," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2012, pp. 2527–2530.

[28] C. De Boor and C. De Boor, *A Practical Guide to Splines*, vol. 27. New York, NY, USA: Springer, 1978,

[29] (Nat. Instrum., Austin, TX, USA). *Usrp-2954*. Accessed: May 17, 2024. [Online]. Available: https://www.ni.com/en-us/shop/model/usrp-2954.html

[30] (Sivers Semicond., Sweden, U.K.). *Evk 06002/00*. Accessed: May 17, 2024. [Online]. Available: https://www.siversima.com/product/evk-06002-00/

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[32] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, Nov. 2008.

[33] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

CHENQING JI (Graduate Student Member, IEEE) received the B.Eng. degree from the Southern University of Science and Technology, Shenzhen, China, in 2023, where he is currently pursuing the M.Eng. degree with the Department of Electronic and Electrical Engineering. His research interests include wireless sensing, channel modeling, and passive mmWave radar system.

CHAO YU received the B.S. degree from the Qingdao University of Technology, Qingdao, China, in 2019, and the M.E. degree from the Southern University of Science and Technology, Shenzhen, China, in 2023, where he is currently a Research Assistant. His research interests include integrated sensing and communication and passive sensing.

SHUAI WANG (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees from the Beijing University of Posts and Telecommunications in 2011 and 2014, respectively, and the Ph.D. degree from the University of Hong Kong in 2018. He is currently an Associate Professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. He has published more than 80 papers in top journals and conferences. His research interests lie at the intersection of autonomous systems and wireless communications. He received the Best Paper Awards from IEEE ICC in 2020, IEEE ComSoc SPCC Committee in 2021, and IEEE ICDCSW in 2023. He is currently an Associate Editor of IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY.

ZHENYU REN received the B.Eng. degree from the Southern University of Science and Technology, Shenzhen, China, in 2022, where he is currently pursuing the M.Eng. degree in electronic and electrical engineering. His research interests include wireless sensing and channel modeling.
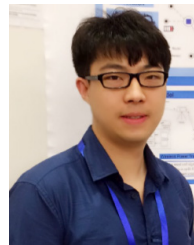

GUOLIANG LI (Graduate Student Member, IEEE) received the B.Eng. and M.Eng. degrees (summa cum laude) from the Southern University of Science and Technology, Shenzhen, in 2020 and 2023, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer and Information Science, University of Macau, Macau, China. His research interests include autonomous driving, mobile robotics, and sensor simulation. He was the recipient of the Best Student Paper Award from the IEEE ICDCS'2023 Workshop on SocialMeta.

RUI WANG (Member, IEEE) received the B.S. degree from the University of Science and Technology of China in 2004, and the Ph.D. degree in wireless communications from The Hong Kong University of Science and Technology in 2008.

From 2009 to 2012, he was a Senior Research Engineer with Huawei Technologies Company Ltd. Since 2012, he has been an Associate Professor with the Southern University of Science and Technology of China. He has research experience in both academia and industry. He has authored over 100 papers in top-level IEEE journals and flagship international conferences, especially in the area of wireless radio resource optimization and interference management. He has contributed over 20 U.S. patent applications and over 50 Chinese patent applications (30 of them have been granted).