CSI-based Indoor Localization via Attention-Augmented Residual Convolutional Neural Network

Bowen Zhang* Student Member, IEEE, Houssem Sifaou Member, IEEE, Geoffrey Ye Li Fellow, IEEE

Abstract

Deep learning has been widely adopted for channel state information (CSI)-fingerprinting indoor localization systems. These systems usually consist of two main parts, i.e., a positioning network that learns the mapping from high-dimensional CSI to physical locations and a tracking system that utilizes historical CSI to reduce the positioning error. This paper presents a new localization system with high accuracy and generality. On the one hand, the receptive field of the existing convolutional neural network (CNN)-based positioning networks is limited, restricting their performance as useful information in CSI is not explored thoroughly. As a solution, we propose a novel attention-augmented Residual CNN to utilize the local information and global context in CSI exhaustively. On the other hand, considering the generality of a tracking system, we decouple the tracking system from the CSI environments so that one tracking system for all environments becomes possible. Specifically, we remodel the tracking problem as a denoising task and solve it with deep trajectory prior. Furthermore, we investigate how the precision difference of inertial measurement units will adversely affect the tracking performance and adopt plug-and-play to solve the precision difference problem. Experiments show the superiority of our methods over existing approaches in performance and generality improvement.

Index Terms

Indoor positioning, tracking, IMU, CSI, deep learning.

This work was sponsored by the donation from Huawei Technologies.

*Corresponding author

Bowen Zhang, Houssem Sifaou and Geoffrey Ye Li are with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, U.K. (e-mail: k.zhang21@imperial.ac.uk; h.sifaou@imperial.ac.uk; geoffrey.li@imperial.ac.uk)
I. INTRODUCTION

In recent decades, accurate positioning is receiving increasing interest as a key enabler for many location-based services, such as navigation, smart robots, and the internet of things. However, the widely-used global positioning system is not applicable in indoor scenarios since the line-of-sight (LOS) propagation between the mobile terminals (MT) and the satellites can be blocked. Therefore, many indoor positioning techniques have been proposed as alternative solutions.

Among these alternatives, channel state information (CSI)-based localization has achieved increasing attention for its simplicity, broad applicability, and reliability [1]–[5]. Since the channel state is determined by the wireless propagation environment around the MT, the CSI-based positioning approach transforms the localization problem into a pattern recognition problem. Specifically, it consists of a data collection stage where CSI measurements are taken at spatially-distributed reference points (RPs), an algorithm development stage where the CSI at different RPs are used to optimize the matching algorithm, and an online estimation stage where the MT location is estimated using the matching algorithm.

Inspired by the great success of deep learning, some works treat the CSI as an image and use convolutional neural network (CNN) to learn the mapping from CSI measurements to two-dimensional terminal coordinates [6]–[9]. After training on the CSI-location pairs collected at the RPs, these CNN-based methods achieve higher positioning accuracy than the traditional probabilistic methods [1].

Although the existing methods have successfully introduced deep CNN into the CSI-based indoor localization problem, we find that using a CNN architecture with a larger receptive field (RF) can improve the performance. Unlike in fully connected networks (FCN), where the value of each pixel depends on the entire input, the value of a pixel in a CNN layer is only affected by the input in a region, which is defined as the RF of that CNN layer [10]. As each pixel in one layer cannot see the input information outside its RF, it is essential to ensure the RF is large enough to cover all the relevant information for accurate decision making. For CSI-based indoor positioning, the requirement for a large RF is significant as we need to leverage the channel response on multiple non-neighbouring frequency bands and antennas to alleviate the influence of environmental noise [11] and fading [1].

Besides the positioning system, a tracking system is usually built to improve the localization
accuracy for moving targets. As the motion patterns in an indoor environment are relatively fixed due to environmental restrictions, the tracking system can use the historical information on the trajectory to enhance the location estimation for the current timestamp [12]. In the context of indoor tracking, a generic approach is to take the results from a positioning system as measurements and track the state of the target, including position, velocity, and acceleration, over time via Bayesian estimators [13]–[15]. Nevertheless, the success of these parametric model-based methods relies on the accuracy of system models [16], which are difficult to estimate in many cases.

To address the above problem, integrating the sensory data from an inertial measurement unit (IMU) into a tracking system is a popular direction [17], [18]. By using built-in IMU, such as acceleration sensor and gyroscope, step length and orientation can be estimated through the pedestrian dead reckoning (PDR) approach [19], with which a more realistic system model can be built.

Some learning-based tracking [12], [20], [21] or IMU-aided tracking methods [22], [23] have recently been proposed for indoor environments. By exploring the correlation of CSI measurements or IMU data along a trajectory, these methods achieve better performance than stationary positioning systems and Bayesian filtering tracking systems.

Despite the performance improvement, the generality of these learning-based tracking systems can be improved. Expressly, most existing tracking methods assume that the tracking network is trained for each indoor environment. At the same time, this is not easy as training a tracking system requires collecting precise positions and CSI/IMU measurements for a continuously moving target at discrete time instances, which are difficult to measure precisely without enough time and workforce [22]. Also, existing works mainly focus on one type of IMU and do not consider the precision difference between different types of IMU. We find it impossible for a single network to handle IMU measurements with varying precision while training a specific model for each type is a huge burden. Therefore, it is necessary for a universal tracking system to be used across indoor environments and compatible for IMU equipment with arbitrary precision.

In this paper, we consider both CSI-based positioning and tracking. For positioning, we will enhance the feature extraction process of the positioning network by increasing its RF. Specifically, we first enlarge the RF by increasing the depth of our network with stacked residual blocks [24]. Then, we extract global context from CSI through an attention-augmented convolutional operation [25]. Besides, we re-model tracking as a denoising task, where the
trajectory generated by a positioning system is a noisy observation of the true trajectory, and explore the inherent properties of indoor trajectories to mitigate the noise. These properties are expressed by a trajectory prior function in our work. As the trajectory prior is independent of the wireless propagation environments and positioning systems, our tracking system has better generality.

Moreover, for IMU-aided tracking systems, we consider the precision difference between different IMUs [26] and adopt the idea of plug-and-play (PnP) [27] to incorporate the IMU data into a learning-based tracking system in a model-based way so that the precision difference problem can be handled by tuning the trade-off parameters in the model.

Our main contributions can be summarized as follows:

1) To improve positioning performance, we develop a novel attention-augmented residual convolutional neural network (AAResCNN) for CSI-based indoor positioning. Experiments on publicly-released datasets show that our network can achieve higher localization accuracy than state-of-the-art (SOTA) CNN-based methods.

2) We propose a universal tracking method by training a set of Gaussian Denoiser based on deep trajectory prior. Simulation results verify the generality of our tracking system for different environments.

3) We use PnP to integrate IMU sensory data into the tracking system. Experiments on simulated datasets show that our methods perform better than learning-based and Bayesian filtering-based methods when the signal-to-noise-ratio (SNR) of IMU sensory data changes.

In the remainder of the paper, related works are reviewed in Section II. In Section III we present the overall architecture and assumptions of our system. Sections IV, V and VI introduce our positioning, tracking, and IMU-aided tracking methods, respectively. The experiments setups and evaluation results are presented in Section VII. Finally, conclusions of our work are shown in Section VIII.

II. RELATED WORKS

Many studies have been conducted for CSI-based indoor localization. This section divides the existing approaches into three categories, i.e., positioning, tracking and IMU-aided tracking, and presents part of representative works for each category. Furthermore, this section describes the difference between our work and the existing ones.
A. CSI-based Positioning

CSI-fingerprinting indoor positioning systems have achieved wide attention in recent years. In 2012, Xiao et al. [1] proposed FIFS using the summation of power across frequency bands in CSI as fingerprints and achieved higher accuracy than traditional received signal strength indicator (RSSI)-based approach. In [28], channel responses on multiple antennas were aggregated for CSI-based indoor positioning. Afterwards, exploring channel information in both frequency and spatial domain becomes the mainstream.

With the development of machine learning, many learning-based approaches have been proposed to improve localization accuracy using CSI fingerprints [2]–[5]. For example, in [2], restricted Boltzmann machines (RBMs) were used to store the CSI-fingerprints as trainable weights for following matching algorithms. Long short-term memory (LSTM) was adopted in [5] to explore the correlation of CSI features over time. X. Wang et al. in [4] constructed a set of CSI images from received packets and used deep CNN for feature extraction and location estimation.

The methods mentioned above all collect multiple received packets for one position to suppress channel fluctuation and noise. However, it is hard to collect multiple packets in a short time when tracking a moving target. Therefore, positioning with a single CSI sample is required. Several CNN-based methods that take a single CSI sample as input have been proposed [6]–[9], [29]. In this paper, we propose a novel CNN architecture that leads to better positioning accuracy.

B. CSI-based Tracking

Bayesian filtering has been widely adopted for tracking tasks. For example, S. Shi et al. in [13] exploited the Kalman filter (KF) to continuously track the trajectory of a moving target using the location results from a CSI-based positioning system. In [14], enhanced Kalman filter (EKF) was applied to recover the trajectory from CSI measurements. Particle filter (PF)-based tracking methods were also proposed in [13], [15].

LSTM-based tracking methods have also been proposed for indoor environments recently. For instance, in [20], [21], continuous CSI measurements of trajectories were first represented as deep features via a CNN network and then the features were fed into an LSTM network for tracking purposes. However, these learning-based tracking methods are designed under a specific environment and their generality is poor. Moreover, training these networks requires collecting the CSI measurements along plenty of continuous trajectories, which is time-consuming. In this
paper, we will design a universal tracking network from a trajectory dataset alone without CSI measurements.

C. IMU-aided Tracking

As shown in [17], [18], [23], fusing the IMU sensor data with wireless signals can help reduce the positioning error. Recently, there have been some works to explore the functionality of IMU data in a learning-based tracking system. For example, A. Xie et al. in [22] used the IMU sensor data to learn the transition function in a state-space model. In [23], the trajectory recovered by the CSI measurements was refined by the trajectory estimated through the PDR approach via back-propagation. Nevertheless, the influence of IMU precision on the tracking system has not been discussed.

III. SYSTEM ARCHITECTURE AND ASSUMPTIONS

A. System Architecture

In this section, we introduce the overall architecture of our localization system, as shown in Fig 1. Suppose a single base station (BS) is providing localization services for the terminals in an indoor environment. When the BS receives a localization request from a terminal in its coverage at time $t$, the BS first measures the CSI from the received packets. Assuming the BS works on a multiple-input multiple-output (MIMO) orthogonal frequency-division multiplexing (OFDM) system with $A$ receive antennas and $S$ sub-carriers, the measured CSI can be represented by a
channel response matrix $I_C^t \in \mathbb{R}^{A \times S \times 2}$, where 2 denotes the real part and imaginary part for the complex channel response, $A$ and $S$ denote the spatial dimension and frequency dimensions of CSI, respectively. $A \times S$ denotes the spatial size of CSI. Furthermore, if the antennas are placed as a uniform linear array (ULA), the channel response on the $a$-th antenna and $s$-th sub-carrier can be expressed as [7],

$$[I_C^t]_{a,s} = \sum_{p=1}^{P} c_p e^{-j2\pi \frac{ad \cos(\phi_p)}{\lambda_c}} e^{-j2\pi f_s \tau_p},$$

(1)

where $P$ denotes the number of distinguishable paths between the MT and the BS; $c_p$, $\phi_p$, and $\tau_p$ denote the complex channel coefficient, the angle of arrival, and the time delay for the $p$-th path, respectively; $d$ denotes the spatial distance between the adjacent antennas, $\lambda_c$ denotes the wavelength of the center frequency, and $f_s$ denotes the frequency of the $s$-th sub-carrier. The BS then estimates the location of the MT using a CSI-based positioning network, as shown in Fig [2] (a),

$$\tilde{l}_t = \text{Position-Net}(I_C^t),$$

(2)

where Position-Net(·) denotes the CSI-based positioning network. Afterwards, the BS transmits the estimated localization results to the MT through the communication link.

When the MT receives the location estimation, $\tilde{l}_t$, from the BS, it will exploit historical trajectory information to reduce the positioning error via a tracking system iteratively [13]. Specifically, it will first search its memory and generate a trajectory that consists of $T$ steps,

$$\tilde{L}_t = \{\tilde{l}_{t-T+1}, \tilde{l}_{t-T+2}, \cdots, \tilde{l}_{t-1}, \tilde{l}_t\},$$

(3)

where $\tilde{L}_t$ denotes the trajectory at time $t$. The trajectory is then refined by a tracking system, as shown in Fig [2] (b),

$$\hat{L}_t = \text{Tracking}(\tilde{L}_t),$$

(4)
where Tracking(·) denotes the tracking system. \( \hat{L}_t = \{\hat{l}_{t-T+1}, \hat{l}_{t-T+2}, \ldots, \hat{l}_{t-1}, \hat{l}_t\} \) denotes the refined trajectory after considering historical trajectory information. At last, the MT uses \( \hat{l}_t \) as its location at time \( t \).

Most of the MTs nowadays are equipped with IMUs, such as accelerometer, gyroscope, and magnetometer [19]. The MTs can utilize the IMU measurements \( \tilde{M}_t = \{\tilde{m}_{t-T+2}, \ldots, \tilde{m}_{t-1}, \tilde{m}_t\} \) for tracking, where \( \tilde{m}_t = [\tilde{r}_t, \tilde{\theta}_t] \) denotes the estimated moving distance and direction from time \( t - 1 \) to time \( t \), respectively [17]. The trajectory refined by the IMU-aided tracking system can be represented as

\[
\bar{L}_t = \text{IMU-Tracking}(\hat{L}_t, \tilde{M}_t),
\]

where IMU-Tracking(·) denotes the IMU-assisted tracking system. \( \bar{L}_t = \{\bar{l}_{t-T+1}, \bar{l}_{t-T+2}, \ldots, \bar{l}_{t-1}, \bar{l}_t\} \) denotes the refined trajectory considering both the positioning network and IMUs, as shown in Fig 2 (c). Similarly, the MT uses \( \bar{l}_t \) as the final result.

**B. Assumptions**

Here, we clarify some important assumptions used in our work. First of all, before putting a positioning network into use, the parameters of the network should be trained on a CSI-location dataset. To construct the dataset, CSI is measured at some random RPs in the indoor environment with known locations, as shown in Fig 1. Also, as the CSI is environment-specific, the data-collection and training process is repeated for each indoor environment, and the positioning network deployed for each environment is different. Secondly, based on the fact that a MT will move from one indoor environment to another, the tracking system should be able to work across the environments. In addition, considering the time-consuming data-collection process for training a tracking system, it would be better that a tracking system trained under one environment can be applied to any other environment directly without retraining. Thirdly, since it is hard to ensure the IMU deployed on each MT has the same precision, the IMU-aided tracking system should be compatible with IMU with arbitrary precision.

**IV. CSI-based Positioning Using Deep CNN**

A large RF means pixels far away from each other in the spatial and frequency dimensions of CSI can be considered simultaneously by the network [10], which can be pretty helpful in exploiting location-related details. For example, it has been investigated in [11] that using CSI
phase difference from different antennas that have a larger spatial distance for angle-of-arrival estimation has a better anti-noise performance. From [1], CSI amplitudes from the subcarriers whose distances are larger than the coherent bandwidth are more effective as fingerprints as these subcarriers are fading independently. Therefore, instead of focusing on the local pixels, we should collect and analyze the clues provided by non-neighbouring pixels.

However, two main factors restrict the RF of the existing CNN-based methods for CSI-based indoor positioning: 1) the number of the convolutional layers of the existing methods is not large enough [7]–[9], [29], while the number of convolutional layers should be increased to enlarge the RF [30]. 2) the RF of a convolutional layer is naturally restricted by its inertial working mechanism, where each pixel in the output features of a convolutional layer is a weighted sum of the pixels in a small local region in the input features, making CNN itself inefficient in extracting the global context from CSI.

Different from the existing works, we propose two ways to increase the RF of our network. On the one hand, we build a very deep CNN with more than 20 convolutional layers by stacking residual blocks (RBs). On the other hand, we use an attention-augmented residual block (AARB) for global context extraction, with which it is easier for the RF to fully fill the whole input. More details about our positioning network are given hereafter.

A. Network Architecture

We first present the overall network architecture of the proposed AAResCNN for CSI-based indoor localization. As shown in Fig. 3, our network mainly consists of four parts: residual blocks (RBs), pooling blocks (PBs), attention-augmented residual blocks (AARBs), and finally a FCN. We denote the input CSI and output predicted location as $I_C$ and $\hat{l}$ respectively. First of all, we use one convolutional layer to extract shallow features $F_0$ from $I_C$,

$$F_0 = Conv(I_C),$$

(6)
where $\text{Conv}(\cdot)$ denotes convolution operation. Considering the large spatial size ($H \times W$) of CSI in MIMO-OFDM systems, we then use a cascade of RBs and PBs to fuse the local information in shallow features $F_0$ and reduce its spatial size gradually. Assuming we have $D$ cascaded RBs and PBs in total, the output of the $d$-th PB can be obtained as

$$F_d = PB_d(RB_d(F_{d-1}))$$

$$= PB_d(RB_d(\cdots PB_1(RB_1(F_0))\cdots)), \quad (7)$$

where $PB_d$ and $RB_d$ denote the $d$-th PB and RB, respectively, and the value of $D$ is set according to the size of CSI input. If the pooling size and stride are set as $(p, q)$ in PBs, the height and width of $F_D$ will be $1/(pD)$ and $1/(qD)$ of those in $F_0$. More details about RB and PB will be given in Section [IV-B] and Section [IV-C] respectively.

After extracting spatially-downsampled features $F_D$, we continue to adopt a stack of RBs to extract local information from $F_D$. Also, to better capture the global context, we substitute RB with AARB after every $z$ RBs. If $z = 1$, we only use AARB after $F_D$. If there are $M$ blocks after the last PB, the $(D + m)$-th output feature maps of our network can be expressed as

$$F_{D+m} = \begin{cases} 
AARB(F_{D+m-1}), & \text{mod}(m - 1, z) = 0, \\
RB(F_{D+m-1}), & \text{mod}(m - 1, z) \neq 0,
\end{cases} \quad (8)$$

where $\text{mod}(\cdot)$ denotes the modulo operation. More details about AARB will be shown in Section [IV-D]. After applying $M$ blocks to $F_D$, we flatten the deep features $F_{D+M}$ to a vector $F_f$,

$$F_f = \text{Flatten}(A_M), \quad (9)$$

Finally, we use a 3-layer FCN with a configuration of $n_1$-$n_2$-$2$ hidden neurons to gradually map $F_f$ from the high-dimensional feature space to 2D coordinates. The predicted location can be obtained by

$$\tilde{l} = FCN(F_f). \quad (10)$$

Suppose there are $I$ CSI-location pairs in the training dataset in total, the overall network is trained to minimize the following loss function,

$$\text{Loss} = \sum_{i=1}^{I} \|\text{Position-Net}(I_C^i) - l^i\|^2 = \sum_{i=1}^{I} \|\tilde{l}^i - l^i\|^2, \quad (11)$$

where $\| \cdot \|^2$ denotes $l_2$-norm and $l^i$ is the true location of the $i$th training sample. $I_C^i$ is the CSI measured at location $l^i$ and $\tilde{l}^i$ is the estimated location using $I_C^i$ as input.
B. Going deeper with residual blocks

To create filters that are responsive to a larger region in the input, increasing the depth of networks has been widely used in image-related tasks [30]. For example, if we use filters of size $k \times k$ for all the convolutional layers in our network, the RF of the $d$-th convolutional layers in our network will be size of $((k - 1)d + 1) \times ((k - 1)d + 1)$, which grows as $d$ increases. Another benefit of using a very deep network is that it can introduce more non-linearities to the network as a convolutional layer is often followed by a non-linear layer. With the growth of non-linearities, our network can model more complex functions, which is essential when dealing with CSI in complicated wireless environments.

However, if we stack convolutional layers in a plain way, we may face a performance degradation problem. That is, as the network goes deeper, the newly-added layers not only fail to bring performance improvements but also raise the training and test errors, which is also reported in [31]. To address the degradation problem, we choose to adopt the residual blocks (RBs) proposed in [24]. We show the detailed architecture of one RB in Fig 4(a). As we can see from Fig 4(a), one residual block consists of two convolution layers, an activation layer along with an identity skip connection. The skip connection provides convenience for the back-propagation of gradients, which helps to stabilize the training phase and reduce the training pressure. Therefore, by stacking RBs, we can build a very deep CNN with a large RF.

Note that in our block-based CNN architecture, the channel dimension of feature maps keeps constant across different layers, different from the channel growth strategies in [9], [29]. Actually, keeping the channel number a small constant can lead to a lighter model. For example, the model parameters used to build a convolutional layer between feature maps with 256 channels can build up to 64 convolutional layers among features with 32 channels.
C. Pooling blocks

Due to the large number of antennas and sub-carriers in a massive MIMO-OFDM system, the spatial and frequency dimensions of CSI can be quite huge; therefore, spatially-downsampling is required. To reduce the feature size, average pooling (AveP) is quite popular in CSI-based indoor positioning networks. After an AveP layer with equal pooling size and stride \((p, q)\), the height and width of input features will be reduced to \(1/p\) and \(1/q\) of the original sizes while the channel number will keep unchanged. However, information loss is unavoidable in this process since pixels in the height and width dimensions are averaged manually and represented by fewer pixels. To mitigate the information loss during the pooling process, we introduce the idea of pooling blocks (PBs) \([32]\). In a PB, we first use a convolutional layer to double the channel number of input features. Next, we apply an AveP to reduce the spatial size. Finally, we adopt another convolutional layer to reduce the channel number to the original size. It is expected the added two convolutional layers will learn to transfer the important information for position estimation from spatial dimension to channel dimension and preserve it through training. The details of a PB are shown in Fig 5. The idea of doubling the channel number before downsampling is also used in \([33]\).

D. Attention-augmented Residual blocks

Although the theoretical receptive field grows as more RBs are stacked, the growth rate of effective receptive field (ERF) is much slower \([10]\). Also, as analyzed in \([10]\), not all pixels in the RF contribute equally to the outputs. In fact, ERF follows Gaussian distribution, which means central pixels in the RF usually have a larger impact. To overcome these drawbacks, attention-mechanisms have been widely adopted \([34]\).

Attention mechanisms have recently been proposed for global context extraction in language models \([35]\) and image-related tasks \([36]\). By introducing attention mechanisms, neural networks
can see the whole input rather than focusing on a local region. In this work, we explore the performance of attention-augmented convolution (AAConv) proposed in [25] for CSI-based indoor localization, and use attention-augmented residual blocks (AARBs) to enhance the performance of positioning networks. Hereafter, we give a detailed description of AAConv and AARBs.

1) Comparison of Conv and AAConv: We first compare the overall architectures of the traditional convolution layer (Conv) and the AAConv in Fig 6. Specifically, in the Conv, each pixel in the output features is a weighted sum of pixels in a local window in the input features and the weights are shared for all output pixels, while all pixels in the input features are considered in the AAConv, and the weights are dynamically generated for each output pixel. By introducing the AAConv, the RF increases from a local window to the whole input, which significantly improves the feature extraction efficiency for the global context.

2) Attention-augmented convolutional layer with a single head: We now describe the detailed architecture of the AAConv with a single head, which is also depicted in Fig 7. Given input features with shape \((H, W, C_{in})\), where \(H\), \(W\), and \(C_{in}\) denotes the height, width and channels of the input feature, respectively, the AAConv first computes three intermediate features called
queries $Q \in \mathcal{R}^{H \times W \times N_q}$, keys $K \in \mathcal{R}^{H \times W \times N_k}$ and values $V \in \mathcal{R}^{H \times W \times N_v}$ through 3 sets of $1 \times 1$ Conv, where $N_q$ equals to $N_k$ all the time.

Next, for each pixel $(h, w)$ in the height and width dimensions of the output features $O \in \mathcal{R}^{H \times W \times N_v}$, which is also denoted as $O_{h,w} \in \mathcal{R}^{N_v}$, an attention weight matrix $w^{O_{h,w}} \in \mathcal{R}^{H \times W}$ is computed from queries $Q$ and keys $K$ through

$$w^{O_{h,w}}_{m,n} = \frac{Q_{h,w} \cdot K_{m,n}}{\sqrt{N_q}} \text{ for } m = 1, \ldots, H; n = 1, \ldots, W$$

(12)

where $w^{O_{h,w}}_{m,n} \in \mathcal{R}$ denotes the pixel $(m, n)$ in the height and width dimensions of attention weight $w^{O_{h,w}}$, $Q_{h,w} \in \mathcal{R}^{N_q}$ denotes the pixel $(h, w)$ in queries $Q$, and $K_{m,n} \in \mathcal{R}^{N_k}$ denotes the pixel $(m, n)$ in keys $K$; $\cdot$ denotes inner product.

The values in the attention weight matrix, $w^{O_{h,w}}$, denote the feature similarity between $Q_{h,w}$ and each pixel in keys $K$, and the weight matrix is then normalized by

$$\hat{w}^{O_{h,w}} = \text{Softmax}_{2D}(w^{O_{h,w}}),$$

(13)

where $\text{Softmax}_{2D} (\cdot)$ denotes applying Softmax function to a matrix. Specifically, for a matrix $w \in \mathcal{R}^{H \times W}$, $[\text{Softmax}_{2D} (w)]_{h,w} = e^{w_{h,w}} / \sum_{m=1}^{H} \sum_{n=1}^{W} e^{w_{m,n}}$ for $h = 1, \ldots, H; w = 1, \ldots, W$, where $w_{h,w}$ denote pixel $(h, w)$ in $w$.

Finally, the normalized attention weight matrix, $\hat{w}^{O_{h,w}}$, is used to reweigh values $V$. The reweighing scheme can be formulated as

$$O_{h,w} = \sum_{m=1}^{H} \sum_{n=1}^{W} \hat{w}^{O_{h,w}}_{m,n} V_{m,n}$$

(14)

where $\hat{w}^{O_{h,w}}_{m,n} \in \mathcal{R}$ denotes pixel $(m, n)$ in weight matrix $\hat{w}^{O_{h,w}}$, $V_{m,n} \in \mathcal{R}^{N_v}$ denotes pixel $(m, n)$ in values $V$.

From (12), (13) and (14), all pixels in the height and width dimensions of input features are considered and the pixels that are more similar to pixel $(h, w)$ in the feature space will contribute more to the generation of $O_{h,w}$.

In the original AAConv proposed in [25], the AAConv with multiple heads is utilized to explore the global context in a more diversified way rather than a single head. However, we notice that using multi-heads will significantly increase the computational cost. Therefore, we only use AAConv with a single head here.
3) Attention with relative positional information: The last thing about the AAConv is the relative positional information, without which the output features are permutation equivariant for the pixel locations in input features \[25\]. To encourage the attention mechanism to consider both the pixel distance and feature similarity, I. Bello et al. in \[25\] incorporate relative positional encodings \[37\] to (12):

\[
\begin{align*}
    w_{m,n}^{O_{h,w}} &= \left( \frac{1}{\sqrt{N_q}} \right) (Q_{h,w} \cdot K_{m,n} + Q_{h,w} \cdot r^H_{m-h} \\
    &\quad + Q_{h,w} \cdot r^W_{n-w}) \quad \text{for } m = 1, \ldots, H; \ n = 1, \ldots, W
\end{align*}
\]  

(15)

where \(m-h\) and \(n-w\) denote the relative height distance and width distance between pixel \((h, w)\) and \((m, n)\), respectively, and \(r^H_{m-h} \in \mathcal{R}^{N_q}\) and \(r^W_{n-w} \in \mathcal{R}^{N_q}\) are learnt positional encodings for relative height \(m-h\) and width \(n-w\), respectively. For an input feature of spatial size \((H, W)\), \(m-h\) takes the values from \(-(H - 1)\) to \((H - 1)\) and \(n-w\) from \(-(W - 1)\) to \((W - 1)\), therefore, the total number of positional encodings for one layer is \(2(H + W) - 2\). The relative positional encodings are shared across heads in one AAConv layer.

4) Attention-augmented residual blocks: After introducing the AAConv, we now give the architecture of the proposed AARB, which is shown in Fig 4(b). As the Conv captures local information while the AAConv extracts global context, we combine the Conv and the AAConv following the idea in \[25\]. Specifically, we replace the first Conv layer in a RB (see Fig 4(a)) with a Conv and an AAConv, and their outputs are concatenated along the channel dimension, which results in an increased number of channels \(N_v + C_{in}\), while the second Conv layer in an AARB will fuse the information in concatenated features and reduce the channel number to \(C_{in}\).

E. Complexity Analysis

We compare the number of parameters and computational complexity of three basic operations in our network, i.e., AveP, Conv, AAConv. Following \[25\], we use floating point operations (FLOPs) as the index of computational cost, which is defined as the number of addition and multiplication used in one operation. Supposing the input features has a shape of \(H \times W \times C_{in}\), the results are shown in Table I. As \(N_q\) and \(N_v\) are usually set as small values, the number of parameters of an AAConv layer is usually smaller than a Conv layer. However, the computational cost of an AAConv layer \((O((HW)^2N_q))\), which is an exponential function of spatial size \(HW\), is larger than a Conv layer \((O(HWk^2C_{in}^2))\), especially when spatial size is large. Therefore, we only use AARB on spatially-downsampled features.
TABLE I

THE NUMBER OF PARAMETERS AND FLOPS OF DIFFERENT OPERATIONS

| Operation | AveP | Conv | AACConv |
|-----------|------|------|---------|
| Params Num | 0    | $\approx k^2 C_n^2$ | $\approx C_n(2N_q + N_v) + 2(H + W)$ |
| Flops     | $H W C_n$ | $\approx 2k^2 H W C_n^2$ | $\approx 2H W [H W (3N_q + N_v) + C_n(2N_q + N_v)]$ |

V. TRACKING SYSTEM VIA DEEP TRAJECTORY PRIOR

In this section, we propose a universal tracking system by exploring the properties of trajectories based on the assumption that some motion patterns are environmentally-invariant.

A. Trajectory Refinement with Deep Prior

Let $\tilde{L} \in \mathcal{R}^{T \times 2}$ denote the predicted positions along a trajectory from a positioning system in consecutive $T$ timestamps. Let $L \in \mathcal{R}^{T \times 2}$ denote the true positions of the same trajectory. The relationship between $\tilde{L}$ and $L$ is

$$\tilde{L} = L + v_p$$

(16)

where $v_p$ is the measurement noise of the positioning system, which is assumed to be additive white Gaussian noise with zero mean. The purpose of trajectory refinement is to recover the clean trajectory $L$ from its noisy measurement $\tilde{L}$. Since this is an ill-posed inverse problem, a regularization term, which is also called a prior function, is required to constrain the solution space. Therefore, the trajectory refinement problem can be reformulated as

$$\hat{L} = \arg \min_L \frac{1}{2} \| \tilde{L} - L \|^2 + \lambda \phi(L),$$

(17)

where $\| \tilde{L} - L \|^2$ denotes the fidelity term that ensures that refined trajectory $\hat{L}$ should be close to the measurement $\tilde{L}$; $\phi(L)$ denotes the trajectory prior function, which enforces the refined trajectory $\hat{L}$ should have some desired properties that a true trajectory $L$ should have; and finally, $\lambda$ denotes the trade-off parameter that balances the effect of fidelity term and the prior term in trajectory refinement, whose value should be tuned during experiments.

According to Bayesian probability, (17) corresponds to denoising $\tilde{L}$ when it has been corrupted by additive Gaussian noise of variance $\sqrt{\lambda}$ with a Gaussian Denoiser, where the trajectory prior $\phi(x)$ is implicitly replaced by a denoiser prior [27]. To address this, we rewrite (17) as

$$\hat{L} = \text{Denoiser}(\tilde{L}, \sqrt{\lambda})$$

(18)
There are two different ways to solve (18), i.e., a model-based optimization method and a discriminative learning method [27]. In this paper, instead of using a hand-crafted prior, we propose to learn the prior with deep learning based methods. To this end, we assume that a trajectory dataset exists, which contains all kinds of trajectories in indoor environments.

**B. Denoising Network for Tracking**

Suppose there are $N_L$ trajectories in the collected dataset $X_L = \{x^i\}_{i=1}^{N_L}$, where $x^i \in \mathbb{R}^{T \times 2}$ denotes one trajectory sample that lasts $T$ timestamps. According to (18), as our aim is to learn a Gaussian denoiser based on a data-driven trajectory prior, we first manually add random $i.i.d.$ Gaussian noise with zero mean and standard deviation equal to $\sqrt{\lambda}$ to trajectory $x^i$,

$$ y^i_t = x^i_t + n^i_t \quad t = 1, \ldots, T $$

where $y^i_t$, $x^i_t$, and $n^i_t$ denote the noisy position, the true position, and the random noise on trajectory $x^i$ at time $t$, respectively. Next, we flatten $y^i$ to be a vector $y^i_f \in \mathbb{R}^{2T}$ and adopt a 3-layer FCN with 128 hidden units to refine $y^i_f$,

$$ \hat{x}^i_f = W_3f(W_2f(W_1y^i_f + b_1) + b_2) + b_3, $$

where $\hat{x}^i_f \in \mathbb{R}^{2T}$ denotes the flattened version of the denoised trajectory, $W_1$, $W_2$, $W_3$ denote the weights in FCN and $b_1$, $b_2$, $b_3$ denote the biases, respectively. $f(\cdot)$ denotes the activation layer. Then, we reshape $\hat{x}^i_f$ to $\hat{x}^i \in \mathbb{R}^{T \times 2}$. Finally, the FCN is trained by,

$$ \min_{\lambda} \frac{1}{N_LT} \sum_{i=1}^{N_L} \sum_{t=1}^{T} \|\hat{x}^i_t - x^i_t\|^2. $$

After training, we can get a Gaussian Denoiser under a specific value of $\lambda$. As the optimal value of $\lambda$ is unknown before deployment, we train a series of Gaussian Denoisers under different
values of $\lambda$ in advance so that we can choose the one with the best performance in the testing phase. During the online testing phase, we take trajectory $\tilde{L}$ recovered by a positioning system as inputs and refine it through the deep tracking network, as shown in (4). For positioning systems with varying accuracy, we only need to fine-tune the value of $\lambda$. We show the training phase and testing phase of the tracking system in Fig 8.

VI. PLUG AND PLAY FOR IMU MEASUREMENTS

In this section, we adopt the idea of PnP [27], [38] to incorporate the IMU measurements with different precisions into an existing tracking system.

Let $\tilde{m}_t = [\tilde{r}_t, \tilde{\theta}_t]$ denote the estimated walking distance and direction from time $t-1$ to time $t$ by the IMU. Suppose the true step length and direction in this period are $r_t$ and $\theta_t$, respectively. We assume the following relationship exists,

$$\tilde{r}_t \cos(\tilde{\theta}_t) = r_t \cos(\theta_t) + v_x, \quad \tilde{r}_t \sin(\tilde{\theta}_t) = r_t \sin(\theta_t) + v_y$$ \hspace{1cm} (22)

where $v_x \sim \mathcal{N}(0, \sigma_{v_x})$ and $v_y \sim \mathcal{N}(0, \sigma_{v_y})$ denote the measurement noise of the IMU on $x$ and $y$ coordinates, respectively. We also define the SNR of IMU measurements as,

$$SNR_{IMU} = 20 \log \frac{r_t \cos(\theta_t)}{\sigma_{v_x}} = 20 \log \frac{r_t \sin(\theta_t)}{\sigma_{v_y}}$$ \hspace{1cm} (23)

Let $\tilde{M} = [\tilde{m}_2, \tilde{m}_3, \cdots, \tilde{m}_T] \in \mathcal{R}^{T-1}$ denote the IMU measurements in consecutive $T$ timestamps. After considering IMU measurements, the trajectory refinement problem becomes

$$\tilde{L} = \arg \min_L f(L, \tilde{M}) + \mu \|\tilde{L} - L\|^2 + \hat{\lambda} \phi(L),$$ \hspace{1cm} (24)

where

$$f(L, \tilde{M}) = \sum_{t=1}^{T-1} \| (l_t - l_{t-1}) - (\tilde{r}_t \cos(\tilde{\theta}_t), \tilde{r}_t \sin(\tilde{\theta}_t)) \|^2,$$ \hspace{1cm} (25)

and $f(L, \tilde{M})$ enforces refined trajectory $\tilde{L}$ to satisfy the motion constraints from IMU measurements; $\|\tilde{L} - L\|^2$ encourages refined trajectory $\tilde{L}$ to be close to estimated trajectory $\tilde{L}$ from the positioning network; $\phi(L)$ denotes refined trajectory $\tilde{L}$ should match the deep trajectory prior. Furthermore, $\mu$ and $\hat{\lambda}$ are the trade-off parameters used to balance the contribution of IMU measurements ($f(L, \tilde{M})$), positioning network ($\|\tilde{L} - L\|^2$), and trajectory prior ($\phi(L)$) in the tracking process. In general, for IMU measurements under different values of SNR, we can fine-tune the values of $\mu$ and $\hat{\lambda}$ to incorporate them correctly.
To solve (24), a new variable $Z$ is introduced to decouple the model-based parts and network-based parts,

$$
\min_{L,Z} f(L, \tilde{M}) + \mu \|\tilde{L} - L\|^2 + \lambda \phi(Z)
$$

\[\text{s.t.} \quad L = Z,\]  \hspace{1cm} (26)

which can now be solved by the alternating direction method of multipliers (ADMM) [39]. Specifically, the $k$-th iteration in solving (26) with the ADMM algorithm can be written as,

$$
L^{k+1} = \arg \min_L f(L, \tilde{M}) + \mu \|\tilde{L} - L\|^2 + \rho/2 \|L - Z^k + p^k\|, \hspace{1cm} (27)
$$

$$
Z^{k+1} = \arg \min_Z \lambda \phi(Z) + \rho/2 \|L^{k+1} - Z + p^k\|, \hspace{1cm} (28)
$$

$$
p^{k+1} = p^k + L^{k+1} - Z^{k+1}, \hspace{1cm} (29)
$$

where $p$ is the scaled dual variable, $\rho$ is the penalty parameter, which does not affect the final result but controls the convergence rate of the ADMM algorithm. The updates of $L$ and $p$ can both be calculated mathematically. For the update of $Z$, following the same idea as in the previous section, we can reformulate (28) as

$$
Z^{k+1} = \text{Denoiser}(L^{k+1} + p^k, \sqrt{\lambda/\rho}) \hspace{1cm} (30)
$$

which means (28) can be solved by using a Gaussian Denoiser trained under the noise level $\sqrt{\lambda/\rho}$. As the value of $\rho$ has little influence on the final output, we can fix the value of $\sqrt{\lambda/\rho}$ used to train the Gaussian Denoiser and fine-tune the value of $\rho$ instead when we fine-tune the value of $\lambda$ to deal with IMU measurements with varying SNR.

VII. EXPERIMENTS

In this section, we will first compare the performance of AAReCNN with SOTA methods [7], [9], [40], [41] on publicly-released CSI-based indoor positioning datasets. And then, we will verify the generality of our tracking system across CSI environments. Finally, we will show the flexibility of our IMU-aided tracking system in dealing with IMU measurements with varying precision.
A. CSI-based Positioning

1) Datasets: To show the performance of our CSI-based indoor positioning network, we conduct experiments on one publicly-released indoor LOS dataset: the KU Leuven Boardroom dataset\(^1\) [41], [42]. The dataset contains CSI and position tags of a massive MIMO-OFDM system measured by the National Instruments 5G Massive MIMO testbed\(^2\). The system has 64 antennas and 100 sub-carriers, therefore, the measured CSI at each position has a shape of \((64, 100, 2)\). During the measurement, the users move along the predefined routes in 4 grids, each spanning a 1.25 m by 1.25 m area. The CSI is collected at 5 mm intervals, resulting in 252004 CSI samples in total. During the experiments, we randomly select 5,000 samples as the test dataset, 5,000 sample as the validation dataset, and we change the number of the training samples from 1,000 to 5,000 to 10,000 to evaluate the influence of RPs’ number on the positioning accuracy. Furthermore, as the antennas of the testbed’s BS can be deployed flexibly, the dataset provides three sets of individual sub-datasets with different antenna topologies, \textit{i.e.}, a uniform rectangular array (URA) of 8 by 8 antennas, a uniform linear array (ULA) of 64 antennas, and a distributed antenna array (DIS). The details of the measurement environment are shown in [41].

2) Implementation Details:

- Details of the network architecture: We first introduce the settings of the network’s overall architecture. Specifically, we set \(D = 2\) and \(p = q = 2\) to reduce the spatial size of deep features by \(1/16\). Then, we set \(M = 7\) and \(z = 1\) to further process the spatially-downsampled features. About the FCN, we set \(n_1 = 64\) and \(n_2 = 32\). The meaning of these parameters can be found in Section [IV-A]. As for the settings in the AACConv layer, we set \(N_q = N_v = 4\). Details about these parameters are discussed in Section [IV-D]. Besides, we use \(5 \times 5\) kernels for all Conv layers, the channel dimension of features is kept as 32 for all blocks, and the activation layer we use is LeakyReLU.

- Details on the training process: We set the batch size as 128 and train the model for 1000 epochs with Adam optimizer. The initial learning rate is set as \(10^{-3}\) and decreases by half for every 200 epochs. We choose the model that performs the best on the validation set as the final model. All experiments run on a single GTX1080Ti GPU, and the codes are implemented by Tensorflow.

\(^1\)https://homes.esat.kuleuven.be/~sdebast/measurements/measurements_boardroom.html
\(^2\)https://www.ni.com/en-gb/innovations/white-papers/14/5g-massive-mimo-testbed–from-theory-to-reality–.html
3) **Comparison with State-of-the-art:** We compare our methods with four SOTA CNN-based methods, whose training processes are set the same as ours for a fair comparison:

- **Sun19 [7]:** In [7], a feature learning module, called CALP, is proposed, and the positioning method is composed of a cascade of five CALP modules followed by a FC layer.

- **Arnold19 [40]:** The method consists of two Conv layers, two pooling layers with a stride \((1, 4)\) and four FC layers with 128 hidden neurons. For the Conv layers, we set the kernel size as \((5, 5)\).

- **Chin20 [9]:** The method has four cascaded gated Conv, Conv, and pooling layers with stride \((1, 4)\), plus a 6-layer FCN, where the hidden neurons have a configuration of 1024-512-256-256-64-2. The method is originally proposed for a system with 924 subcarriers; therefore, it uses a large pooling stride. When we apply it to the Ku Leuven Boardroom datasets, we reduce the pooling stride from \((1, 4)\) to \((1, 2)\). We also enlarge the convolution kernel from \((1, 3)\) to \((5, 5)\).

- **Bast20 [41]:** The authors of [41] are also the releasers of Ku Leuven datasets. Their network contains 13 Conv layers, improved with skip connection and drop-out layers, and three FC layers at the end.

4) **Ablation Investigation:** To show the effectiveness of PBs and AARBs, we also present the performance of our network without PBs or AARBs. Specifically, we first replace the PBs and AARBs in our network with AvePs and RBs, respectively, and denote the resulting network as AAResCNN\_PB0\_AARB0. We then add PBs back and get AAResCNN\_AARB0. By comparing AAResCNN\_PB0\_AARB0 with AAResCNN\_AARB0 and comparing AAResCNN\_AARB0 and AAResCNN, we can prove the effectiveness of PBs and AARBs, respectively.

5) **Performance Evaluation:** Table \[\text{TIII}\] compares the mean-squared error (MSE) between predicted locations \(\hat{l}\) and true locations \(l\) for different positioning networks on the test datasets, which can be expressed as \(\frac{1}{5000} \sum_{i=1}^{5000} \|\hat{l}_i - l_i\|^2\). As we can see from Table \[\text{TIII}\] Chin20 performs the best among the existing methods, while the proposed AAResCNN improves the positioning accuracy of Chin20 by about 10%-48% in different cases, which denotes the superiority of our network in improving the positioning accuracy. In addition, we can see from the ablation investigation that both the PBs and AARBs are effective in decreasing the positioning error.

Fig \[\text{9}\] shows the cumulative distribution function (CDF) of the positioning error using different methods in the boardroom ULA dataset. The results show that AAResCNN has a better localization performance than the SOTA methods.
TABLE II
THE MSE OF DIFFERENT POSITIONING METHODS IN KU LEUVEN BOARDROOM DATASETS. THE BEST RESULTS ARE SHOWN IN BOLD FACE

| Dataset       | Ku Leuven Boardroom | Ku Leuven Boardroom | Ku Leuven Boardroom |
|---------------|---------------------|---------------------|---------------------|
|               | URA (mm)            | ULA (mm)            | DIS (mm)            |
| Training Sample | 1000 | 5000 | 10000 | 1000 | 5000 | 10000 | 1000 | 5000 | 10000 |
| Sun19         | 172.75             | 68.09              | 48.61              | 423.73               | 107.33           | 47.81          | 317.05          | 115.56          | 61.95          |
| Arnold19      | 196.95             | 73.22              | 49.29              | 205.97               | 62.94            | 40.84          | 164.69          | 47.00            | 30.85          |
| Chin20        | 148.53             | 64.49              | 31.80              | 133.03               | 28.41            | 20.35          | 99.00           | 26.23            | 15.21          |
| Bast20        | 206.83             | 120.37             | 80.85              | 275.32               | 89.22            | 42.98          | 286.62          | 94.82            | 44.80          |
| AAResCNN PB0  | 150.65             | 51.70              | 29.64              | 95.96                | 26.86            | 18.04          | 108.32          | 25.35            | 17.13          |
| AAResCNN AARB0 | 147.56           | 45.05              | 27.08              | 79.53                | 26.36            | 17.61          | 81.88           | 23.16            | 16.36          |
| AAResCNN (ours) | 124.63           | 42.86              | 26.15              | 69.54                | 24.72            | 16.21          | 67.26           | 21.97            | 14.17          |

Fig. 9. The CDF of positioning error using different methods in the boardroom ULA dataset. As the number of training samples affects the performance, we consider two situations: (a) \( I = 1000 \) CSI-position training samples are collected; (b) \( I = 10000 \) CSI-position training samples are collected.

Table III shows the model size and the inference time of different positioning networks. The model size is defined as the memory used to store the model and the inference time is the time used to predict the position from one CSI input. Compared with SOTA methods, our method has a slightly longer inference time, which, however, is acceptable considering the performance improvements. Besides, our method has an excellent parameter efficiency. Compared with Arnold19 and Chin20, we only use 40% and 1.28% memory to store our model, respectively.

6) Parameter Analysis: In the above experiments, we set \( M = 7 \), \( z = 1 \) and \( N_q = N_v = 4 \) by default. Here we explain our settings through comparative experiments, where the value of one parameter changes while the others keep constant. The experiments are conducted on the
### TABLE III

**The model size and inference time of different methods**

| methods       | Sun19 | Arnold19 | Chin20 | Bast20 | AAResCNN (ours) |
|---------------|-------|----------|--------|--------|-----------------|
| Size          | 7.7MB | 43.8MB   | 1.38GB | 2.6MB  | 17.7MB          |
| Time          | 0.3ms | 0.2ms    | 0.9ms  | 0.2ms  | 1.1ms           |

### TABLE IV

**The MSE, model size and speed of AAResCNN under varying values of parameters**

| Param   | $M$ | $z$ | $N_q$ ($= N_v$) |
|---------|-----|-----|-----------------|
| value   | 3   | 5   | 7               |
|         | 8   | 1   | 2               |
|         | 3   | 4   | 5               |
| MSE (mm)| 101.8 | 86.3 | 69.5 |
|         | 70.7 | 69.5 | 74.0 |
|         | 77.8 | 80.1 | 77.2 |
|         | 72.4 | 69.5 | 69.7 |
| Size (MB)| 15.0 | 16.4 | 17.7 |
|         | 18.4 | 17.7 | 17.4 |
|         | 17.6 | 17.6 | 17.7 |
|         | 17.8 | 17.6 | 17.8 |
| Time (ms)| 0.6  | 0.9  | 1.1 |
|         | 1.2  | 1.1  | 0.8 |
|         | 0.7  | 0.7  | 0.7 |
|         | 1.0  | 1.0  | 1.1 |
|         | 1.1  | 1.1  | 1.1 |

ULA dataset with $I = 1,000$ training samples. The results are shown in Table IV. As we can see from Table IV, the defaulted settings have the best trade-off between performance and cost. For example, when $M \leq 7$, the MSE falls as the value of $M$ increases, while the MSE stops decreasing when $M \geq 7$. At the same time, the running time and the model size keeps increasing as $M$ grows. Therefore, we set $M = 7$.

### B. Tracking across environments

In this subsection, we will compare the performance of different tracking systems and prove the generality of our method across environments. In our experiments, we first generate 1,000 trajectories inside the measurement areas of the KU Leuven dataset and then we collect the CSI along the trajectories. The resulting CSI-trajectory joint datasets form the test datasets for each environment. We divide the training datasets into two types, *i.e.*, time-efficient and time-costive. For the time-costive type, we generate another $N_L$ trajectories for training and collect the CSI along them. Both the positioning and tracking networks can be trained with these training samples. While for the time-efficient type, we only collect CSI at $I$ discontinuous reference points, which eliminates us from the trouble of generating continuous trajectories and reduces the time cost in the data collection stage. However, a tracking system cannot be designed by the collected CSI-position samples alone due to the lack of a trajectory dataset. One potential way to explore historical trajectory information in this case is to re-use the tracking system designed in other environments whose training data is collected in time-costive type. We assume that
only the training dataset in the boardroom with URA topology adopts the time-costive type and consists of $N_L = 10,000$ training trajectories. We will see if the tracking system designed based on the boardroom URA dataset can help improve the positioning accuracy in other environments whose training data is collected in time-efficient type.

1) Datasets:

- Environmental dataset: we consider 4 environments in total. Besides the Ku Leuven boardroom dataset with three antenna topologies, i.e. ULA, URA, and DIS, we also use the Ku Leuven LOS lab dataset. The lab dataset is similar to the boardroom dataset, but it is collected in a different room with a different wireless environment.
- Trajectory dataset: when we generate trajectories, we consider the following three motion patterns: (a) walking with a constant velocity; (b) walking straight with arbitrary velocity; (c) changing direction at a certain step. The three types arise with equal probability and each trajectory sample consists of $T = 5$ steps.

2) Implementation Details of our Methods and State-of-the-art: In this section, we describe the training and testing details of the compared methods.

- AAResCNN: we train an AAResCNN for each environment using all the available training samples. Specifically, for the boardroom URA dataset, we divide the $N_L T$-step trajectories into $N_L T$ samples and train an AAResCNN, while for the others, we use the $I$ samples for training. During testing, each point along the trajectory is recovered independently using the trained AAResCNNs.
- LSTM-F: we compare with the LSTM-based tracking method proposed in [20], [21]. Following [20], we first train an AAResCNN using all the training samples. And then, we input the CSI to the pre-trained AAResCNN and extract the deep features before the last FC layer. Finally, an LSTM network is used to transfer the deep features of $T$ time steps into locations. During training, we use the AAResCNN trained under boardroom URA dataset to extract deep features and train the LSTM network. During testing, we reuse the LSTM network and extract deep features using the AAResCNN trained for each environment.
- LSTM-P: the method is similar to LSTM-F; the only difference is that we use the predicted locations of the AAResCNN as the inputs of the LSTM network.

[https://homes.esat.kuleuven.be/~sdebast/measurements/measurements_lab.html](https://homes.esat.kuleuven.be/~sdebast/measurements/measurements_lab.html)
TABLE V
THE MSE OF DIFFERENT TRACKING METHODS ACROSS ENVIRONMENTS. THE BEST RESULTS ARE SHOWN IN BOLD FACE

| Dataset         | Boardroom URA (mm) | Boardroom ULA (mm) | Boardroom DIS (mm) | Lab LOS (mm) |
|-----------------|-------------------|-------------------|-------------------|--------------|
|                 | I=500             | I=1000            | I=500             | I=1000       |
|                 |                   |                   |                   | I=5000       |
|                 |                   |                   |                   | I=10000      |
| AAResCNN        | 8.90              | 164.70            | 68.32             | 24.66        |
|                 |                   |                   | 155.41            | 67.05        |
|                 |                   |                   |                   | 21.59        |
|                 |                   |                   |                   | 172.50       |
|                 |                   |                   |                   | 75.46        |
|                 |                   |                   |                   | 46.70        |
| LSTM-F          | 8.30              | 1044.36           | 1111.20           | 1065.30      |
|                 |                   |                   | 998.31            | 1356.26      |
|                 |                   |                   |                   | 1232.88      |
|                 |                   |                   |                   | 1099.45      |
|                 |                   |                   |                   | 1315.23      |
|                 |                   |                   |                   | 1431.48      |
| LSTM-P          | 8.42              | 164.76            | 68.36             | 24.74        |
|                 |                   |                   | 155.62            | 67.11        |
|                 |                   |                   |                   | 21.72        |
|                 |                   |                   |                   | 172.50       |
|                 |                   |                   |                   | 75.51        |
|                 |                   |                   |                   | 46.80        |
| DNN-prior (ours)| 8.99              | 145.00            | 63.71             | 24.17        |
|                 |                   |                   | 130.26            | 60.64        |
|                 |                   |                   |                   | 21.33        |
|                 |                   |                   |                   | 146.86       |
|                 |                   |                   |                   | 68.76        |
|                 |                   |                   |                   | 43.06        |

- DNN-prior (ours): We first train an AAResCNN for each environment. Then we implement our prior-based tracking method using $N_L = 10,000$ trajectories in the boardroom URA dataset, following Section V-B. The Gaussian Denoisers are trained under $\sqrt{\lambda} = \{1, 3, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$, respectively. Finally, the tracking system is used on top of the AAResCNN in each environment by tuning the value of $\lambda$.

Note that we do not compare with Bayesian-filtering-based method here since the MT has arbitrary velocity in our experiments, making it hard to define system models.

3) Performance Evaluation: We compare the performance in Table V. The performance is evaluated by the MSE between the last step of refined trajectory $\hat{L}$ and true trajectory $L$, denoted as $\frac{1}{1000} \sum_{i=1}^{1000} \|\hat{L}_i - L_i\|^2$. As shown in Table V, despite the fact that LSTM-F performs the best in the URA dataset, the method can not be reused for any other environment, as the feature extraction process in each AAResCNN differs significantly. The performance of LSTM-P is not better than AAResCNN in any considered dataset other than the URA dataset, although it can be applied across environments, which means it fails to utilize historical trajectory information effectively when the test environment is different from the training one. On the contrary, the proposed prior-based method works for any environment. It improves the performance of AAResCNN by about 1%-16% depending on the accuracy of AAResCNN except for the boardroom URA dataset, where AAResCNN itself is already highly-accurate and our method becomes ineffective.

C. IMU-aided Tracking With Varying IMU Precision

Here, we will focus on IMU-aided tracking systems and verify the superiority of our plug-and-play based method in dealing with IMU measurements with different precisions. We only
consider the Boardroom ULA dataset and use the same test dataset generated in Section VII-B. As for the IMU measurements, we first calculate the true step length $r_t$ and orientation $\theta_t$ from the trajectories. Then we add noise to them to satisfy different SNR requirements according to (22) and (23). In addition, we collect $I$ CSI-location samples to train the positioning network. We also generate 10,000 trajectories for training purposes and the CSI along the training trajectories is considered to be unknown.

1) Implementation Details of our Methods and State-of-the-art: Besides comparing with the positioning network AAResCNN and the IMU-free tracking method DNN-prior, we also consider the following IMU-aided tracking methods:

- Particle Filter [17]: In [17], a particle filter based method has been proposed to fuse the fingerprinting-based positioning results and IMU measurements. We reimplement this method for comparison.

- DNN-SNR1-100: this method is a data-driven extension of our prior-based tracking approach to IMU-aided tracking. Besides using the noisy locations (training phase) or predicted results from a positioning network (testing phase) as the inputs of the DNN, we also feed the IMU measurements and the corresponding SNR value into the network. The method is trained with IMU measurements from SNR=1 dB to SNR=100 dB simultaneously. We also fine-tune the noise level added during the training phase.

- DNN-SNR1 and DNN-SNR100: Different from DNN-SNR1-100, the methods only consider SNR=1 dB and 100 dB during training, respectively.

- PnP (ours): the details of our PnP-based IMU-aided tracking method are described in Section VI. We use the Gaussian Denoiser that performs the best in Section VII-B under the same testing environment. For IMU measurements with different SNR values, we fine-tune the values of $\mu$ and $\rho$.

2) Performance Evaluation: We compare the performance of different IMU-aided tracking methods in Fig[10] where the MSE vs. SNR is reported. As shown in Fig[10] when the SNR value of IMU measurements is high, all the IMU-aided tracking methods achieve better performance than AAResCNN and DNN-prior. However, these methods perform differently when the SNR value changes. For the methods that assume IMU measurements are highly-reliable during the deployment stage, e.g., particle filter and DNN-SNR100, the performance degrades significantly when the precision of IMU equipment is low in the test phase. While for the method DNN-SNR1, which is trained based on the low SNR value, its performance gets no improvement
Fig. 10. The performance comparison of different IMU-aided tracking methods in the boardroom ULA dataset when the precision of IMU measurements changes. As the number of training samples affects the performance, we consider two situations: (a) $I = 1000$ CSI-position training samples are collected; (b) $I = 5000$ CSI-position training samples are collected.

Fig. 11. Some examples of true trajectories (purple lines), recovered trajectories via positioning network AAResCNN (red lines) and refined trajectories by IMU-aided tracking method PnP (black lines).

when a precise IMU equipment is used. Also, for method DNN-SRN1-100, which takes SNR value as input and considers SNR changes during the training stages, it raises the localization error of DNN-SNR100 in the high SNR range, although it performs better than DNN-SNR100 in the low SNR range. Compared to these methods, our PnP-based IMU-aided tracking method has the best performance in most cases, which signifies its flexibility in dealing with precision difference of IMU.

To show the functionality of IMU measurements in reducing positioning error, we also show some examples of true trajectories, recovered trajectories by positioning network AAResCNN and refined trajectories by IMU-aided tracking method PnP in Fig 11. The examples are extracted under the ULA dataset with $I = 5,000$ training samples. The SNR of IMU measurements is set as 100. As shown in Fig 11 the refined trajectories are closer to the true trajectories.
VIII. CONCLUSION

In this paper, we have investigated CSI-based indoor positioning and tracking for MIMO-OFDM systems. First, we have proposed a new positioning network called AAResCNN, which achieves the lowest positioning error compared with SOTA CNN-based positioning approaches. And then, we improve the generality of tracking and IMU-aided tracking methods. Specifically, we have proposed a novel tracking method based on the deep trajectory prior, which is designed independently from CSI environments and can be used across the wireless environments directly without retraining. We have also proposed a plug-and-play-based IMU-aided tracking method, which can be applied to IMU equipment with arbitrary precision. Numerical results on publicly-released datasets demonstrate that our methods have significantly improved the accuracy of CSI-based indoor positioning and the generality of tracking methods, respectively.

REFERENCES

[1] J. Xiao, K. Wu, Y. Yi, and L. M. Ni, “FIFS: Fine-grained indoor fingerprinting system,” in Proc. 2012 21st Int. Conf. Comput. Commun. Netw. (ICCCN). IEEE, 2012, pp. 1–7.
[2] X. Wang, L. Gao, S. Mao, and S. Pandey, “CSI-based fingerprinting for indoor localization: A deep learning approach,” IEEE Trans. Veh. Technol., vol. 66, no. 1, pp. 763–776, 2016.
[3] Y. Wang, C. Xiu, X. Zhang, and D. Yang, “WiFi indoor localization with CSI fingerprinting-based random forest,” Sens., vol. 18, no. 9, p. 2869, 2018.
[4] X. Wang, X. Wang, and S. Mao, “Deep convolutional neural networks for indoor localization with CSI images,” IEEE Trans. Netw. Sci. Eng., vol. 7, no. 1, pp. 316–327, 2018.
[5] Y. Zhang, C. Qu, and Y. Wang, “An indoor positioning method based on CSI by using features optimization mechanism with LSTM,” IEEE Sens. J., vol. 20, no. 9, pp. 4868–4878, 2020.
[6] J. Vieira, E. Leitinger, M. Sarajlic, X. Li, and F. Tufvesson, “Deep convolutional neural networks for massive MIMO fingerprint-based positioning,” in Proc. IEEE 28th Annu. Int. Symp. Pers. Indoor Mob. Radio Commun., Oct. 2017, pp. 1–6.
[7] X. Sun, C. Wu, X. Gao, and G. Y. Li, “Fingerprint-based localization for massive MIMO-OFDM system with deep convolutional neural networks,” IEEE Trans. Veh. Technol., vol. 68, no. 11, pp. 10 846–10 857, 2019.
[8] G. Cerar, A. Švigelj, M. Mohorčič, C. Fortuna, and T. Javornik, “Improving CSI-based massive MIMO indoor positioning using convolutional neural network,” in Proc. 2021 It. Eur. Conf. Netw. Commun. 6G Summit (EuCNC/6G Summit). IEEE, 2021, pp. 276–281.
[9] W.-L. Chin, C.-C. Hsieh, D. Shiung, and T. Jiang, “Intelligent indoor positioning based on artificial neural networks,” IEEE Netw., vol. 34, no. 6, pp. 164–170, 2020.
[10] W. Luo, Y. Li, R. Urtasun, and R. Zemel, “Understanding the effective receptive field in deep convolutional neural networks,” Adv. Neural Inf. Process. Syst., vol. 29, 2016.
[11] X. Tong, Y. Wan, Q. Li, X. Tian, and X. Wang, “CSI fingerprinting localization with low human efforts,” IEEE/ACM Trans. Netw., vol. 29, no. 1, pp. 372–385, 2020.
[12] S. Bai, M. Yan, Q. Wan, L. He, X. Wang, and J. Li, “DL-RNN: An accurate indoor localization method via double RNNs,” *IEEE Sens. J.*, vol. 20, no. 1, pp. 286–295, 2019.

[13] S. Shi, S. Sigg, L. Chen, and Y. Ji, “Accurate location tracking from CSI-based passive device-free probabilistic fingerprinting,” *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5217–5230, 2018.

[14] J. Choi, “Sensor-aided learning for Wi-Fi positioning with beacon channel state information,” *IEEE Trans. Wirel. Commun.*, 2022.

[15] R. Zhou, M. Tang, Z. Gong, and M. Hao, “FreeTrack: Device-free human tracking with deep neural networks and particle filtering,” *IEEE Syst. J.*, vol. 14, no. 2, pp. 2990–3000, 2019.

[16] Q. Ge, T. Shao, Z. Duan, and C. Wen, “Performance analysis of the kalman filter with mismatched noise covariances,” *IEEE Trans. on Autom. Control*, vol. 61, no. 12, pp. 4014–4019, 2016.

[17] A. Belmonte-Hernández, G. Hernández-Peñaloza, D. M. Gutiérrez, and F. Alvarez, “SWiBluX: Multi-Sensor deep learning fingerprint for precise real-time indoor tracking,” *IEEE Sens. J.*, vol. 19, no. 9, pp. 3473–3486, 2019.

[18] Z. Li, D. B. Acuna, Z. Zhao, J. L. Carrera, and T. Braun, “Fine-grained indoor tracking by fusing inertial sensor and physical layer information in WLANs,” in *Proc. 2016 IEEE Int. Conf. Commun. (ICC)*. IEEE, 2016, pp. 1–7.

[19] N.-H. Ho, P. H. Truong, and G.-M. Jeong, “Step-detection and adaptive step-length estimation for pedestrian dead-reckoning at various walking speeds using a smartphone,” *Sens.*, vol. 16, no. 9, p. 1423, 2016.

[20] M. T. Hoang, B. Yuenv, K. Ren, X. Dong, T. Lu, R. Westendorp, and K. Reddy, “A CNN-LSTM quantifier for single access point CSI indoor localization,” *arXiv prepr. arXiv:2005.06394*, 2020.

[21] Z. Zhang, M. Lee, and S. Choi, “Deep-Learning-Based Wi-Fi indoor positioning system using continuous CSI of trajectories,” *Sens.*, vol. 21, no. 17, p. 5776, 2021.

[22] A. Xie, F. Yin, B. Ai, S. Zhang, and S. Cui, “Learning while tracking: A practical system based on variational gaussian process state-space model and smartphone sensory data,” in *Proc. 2020 IEEE 23rd Int. Conf. on Inf. Fusion (FUSION)*. IEEE, 2020, pp. 1–7.

[23] J. Chen, B. Zhou, S. Bao, X. Liu, Z. Gu, L. Li, Y. Zhao, J. Zhu, and Q. Lia, “A data-driven inertial navigation/bluetooth fusion algorithm for indoor localization,” *IEEE Sens. J.*, 2021.

[24] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, “Enhanced deep residual networks for single image super-resolution,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. workshops*, 2017, pp. 136–144.

[25] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, “Attention augmented convolutional networks,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3286–3295.

[26] W. Yan, L. Bastos, and A. Magalhães, “Performance assessment of the android smartphone’s IMU in a gnss/ins coupled navigation model,” *IEEE Access*, vol. 7, pp. 171 073–171 083, 2019.

[27] K. Zhang, W. Zuo, S. Gu, and L. Zhang, “Learning deep CNN denoiser prior for image restoration,” in *Proc. IEEE Conf. Computer Vis. Pattern Recognit.*, 2017, pp. 3929–3938.

[28] A. Jaffe and M. Wax, “Single-site localization via maximum discrimination multipath fingerprinting,” *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1718–1728, 2014.

[29] C. Wu, X. Yi, W. Wang, L. You, Q. Huang, X. Gao, and Q. Liu, “Learning to localize: A 3d CNN approach to user positioning in massive MIMO-OFDM systems,” *IEEE Trans. on Wirel. Commun.*, vol. 20, no. 7, pp. 4556–4570, 2021.

[30] J. Kim, J. K. Lee, and K. M. Lee, “Accurate image super-resolution using very deep convolutional networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1646–1654.

[31] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[32] G. Lin, A. Milan, C. Shen, and I. Reid, “Refinenet: Multi-path refinement networks for high-resolution semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 1925–1934.
[33] X. Hu, M. A. Naiel, A. Wong, M. Lamm, and P. Fieguth, “RUNet: A robust UNet architecture for image super-resolution,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops, 2019, pp. 0–0.
[34] Z. Lin, M. Li, Z. Zheng, Y. Cheng, and C. Yuan, “Self-Attention ConvLSTM for spatiotemporal prediction,” Proc. AAAI Conf. Artif. Intell., vol. 34, no. 07, pp. 11 531–11 538, Apr. 2020.
[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Adv. Neural Inf. Process. Syst., 2017, pp. 5998–6008.
[36] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” in Int. Conf. Learn. Represent., 2020.
[37] P. Shaw, J. Uszkoreit, and A. Vaswani, “Self-Attention with relative position representations,” in Proc. 2018 Conf. N. Am. Chapter Assoc. Comput. Linguist.: Hum. Lang. Technol., Vol. 2 (Short Pap.), 2018, pp. 464–468.
[38] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, “Plug-and-play priors for model based reconstruction,” in 2013 IEEE Glob. Conf. Signal Inf. Proc. IEEE, 2013, pp. 945–948.
[39] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein et al., “Distributed optimization and statistical learning via the alternating direction method of multipliers,” Found. Trends® in Mach. learn., vol. 3, no. 1, pp. 1–122, 2011.
[40] M. Arnold, J. Hoydis, and S. ten Brink, “Novel massive MIMO channel sounding data applied to deep learning-based indoor positioning,” in Proc. SCC 2019; 12th Int. ITG Conf. Syst., Commun. Coding. VDE, 2019, pp. 1–6.
[41] S. De Bast, A. P. Guevara, and S. Pollin, “CSI-based positioning in massive MIMO systems using convolutional neural networks,” in Proc. 2020 IEEE 91st Veh. Technol. Conf. (VTC2020-Spring). IEEE, 2020, pp. 1–5.
[42] S. De Bast and S. Pollin, “MaMIMO CSI-based positioning using CNNs: Peeking inside the black box,” in Proc. 2020 IEEE Int. Conf. Commun. Workshops (ICC Workshops). IEEE, 2020, pp. 1–6.