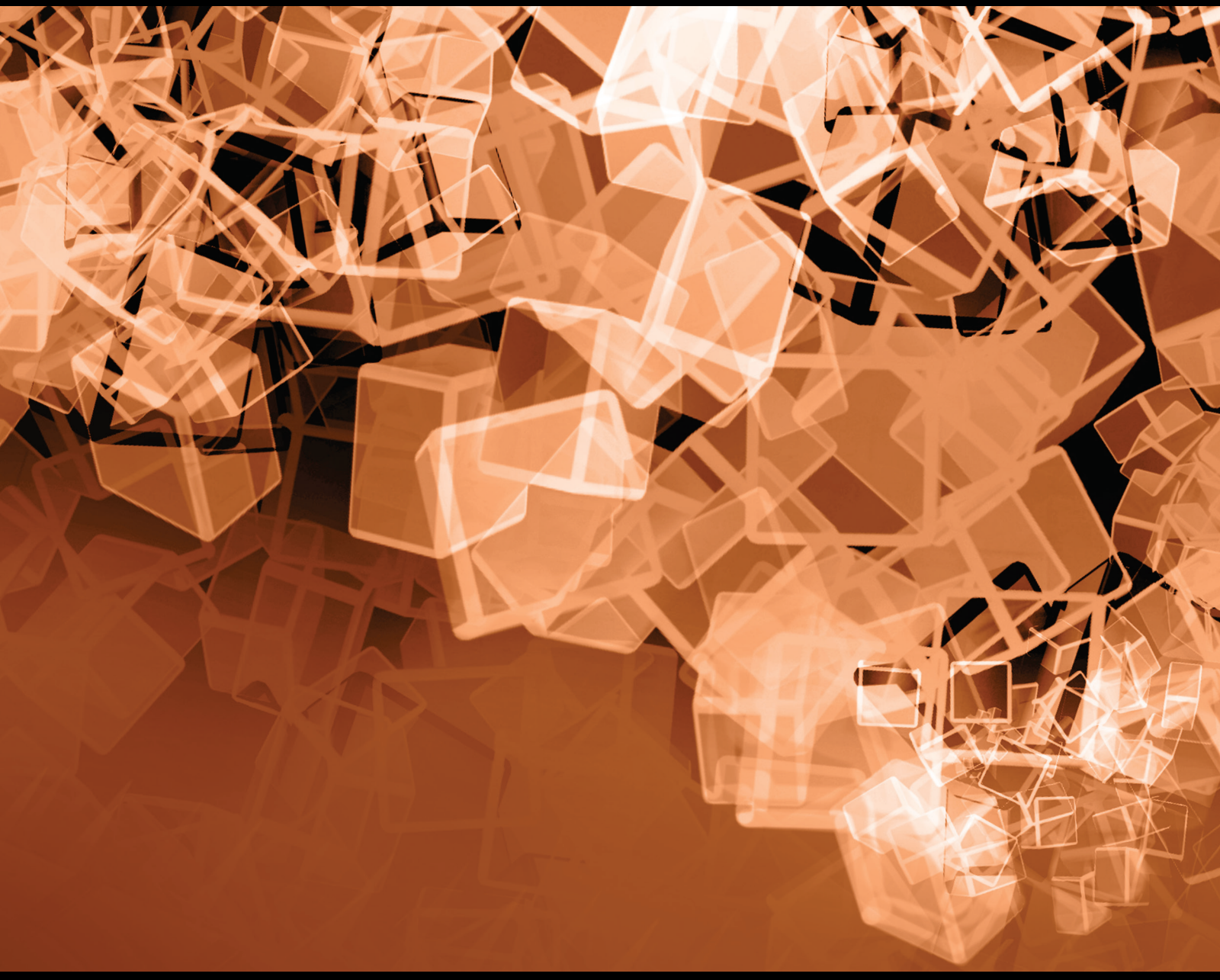
Discrete Dynamics in Nature and Society

Dynamic Modelling and Optimization for Intelligent IoT Networks

Lead Guest Editor: Bo Rong
Guest Editors: Shengjie Xu and Peng Yu



Dynamic Modelling and Optimization for Intelligent IoT Networks

Discrete Dynamics in Nature and Society

Dynamic Modelling and Optimization for Intelligent IoT Networks

Lead Guest Editor: Bo Rong
Guest Editors: Shengjie Xu and Peng Yu

Contents

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 8190688, 11 pages
https://doi.org/10.1155/2022/8190688

Research Article

Machine Learning with Variable Sampling Rate for Traffic Prediction in 6G MEC IoT

Rongqun Peng,[1,2] **Xiuhua Fu**,[1,2] **and Tian Ding**[1,2]

[1]*School of Computer Science and Technology, Shandong University of Technology, Zibo 255000, China*
[2]*State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications), Beijing 100876, China*

Correspondence should be addressed to Rongqun Peng; pengrq@sdut.edu.cn

Received 8 July 2022; Revised 30 September 2022; Accepted 5 October 2022; Published 17 November 2022

Academic Editor: Bo Rong

The high-speed development of mobile broadband networks and IoT applications has brought about massive data transmission and data processing, and severe traffic congestion has adversely affected the fast-growing networks and industries. To better allocate network resources and ensure the smooth operation of communications, predicting network traffic becomes an important tool. We investigate in detail the impact of variable sampling rate on traffic prediction and propose a high-speed traffic prediction method using machine learning and recurrent neural networks. We first investigate a VSR-NLMS adaptive prediction method to perform time series prediction dataset transformation. Then, we propose a VSR-LSTM algorithm for real-time prediction of network traffic. Finally, compared with the traditional traffic prediction algorithm based on fixed sampling rate (FSR-LSTM), we simulate the prediction accuracy of the VSR-LSTM algorithm based on the variable sampling rate proposed. The experiment shows that VSR-LSTM has higher traffic prediction accuracy because its sampling rate varies with the traffic.

1. Introduction

As the global mobile industry moves toward 6G networks, mobile edge computing (MEC)-based network infrastructure has received unprecedented attention to support Internet of Things (IoTs) applications with diverse business needs [1]. To better serve users, the connectivity and intelligence provided by edge computing have tremendous advantages in terms of real-time services, smart living, security, and reliability [2]. At present, edge computing has been applied in smart campuses, video surveillance, industrial IoTs, augmented reality/virtual reality (AR/VR) and other application scenarios, which also proves that MEC-based network infrastructure is effective and fully capable.

The capabilities of edge computing rely on edge servers, which are usually deployed with base stations. It is expected that more IoTs applications will be carried out based on MEC in the future, and the massive data they generate will have great demands on network resources such as bandwidth, computing power, and storage [3, 4]. Therefore, in future, multiple MEC servers will be needed to jointly provide services for different applications. Since MEC servers have different computing processing capabilities and are deployed in a distributed manner, it is critical to offload computing tasks to these heterogeneous MEC servers according to different application requirements. Therefore, reasonable and effective resource allocation mechanisms according to the usage of wireless network resources and MEC server resources will effectively ensure the service requirements of users.

Resource allocation will directly affect the operating cost of MEC-based network infrastructure and the experience of IoTs users. Unreasonable resource allocation will not only increase the operating costs of communication networks but also may lead to serious energy waste. Accurate wireless network traffic prediction can intuitively reflect the changing trend of service requirements, provide an important reference for communication network resource allocation, and is an important guarantee to achieve reasonable and efficient resource allocation and computing task offloading [5, 6].

It has always been a meaningful research topic to analyze the distribution and demand of communication traffic by predicting the wireless network traffic to guide the communication resources' allocation [7]. The previous wireless traffic prediction adopted manual prediction and statistical-based prediction methods, and their limitations are obvious. Manual prediction is inefficient and cannot adjust the network resource allocation in real-time according to the change of service demands. The statistics-based traffic prediction models intelligently utilize certain statistical characteristics, but cannot effectively comprehensively utilize various information that has an important impact on wireless traffic [8]. As machine learning techniques are intensively researched, there have been some works using machine learning and deep learning algorithms for wireless network traffic prediction [9–18].

In the future, artificial intelligence will be one of the native features of the next-generation mobile communication network, namely the sixth-generation (6G). Through AI-based endogenous intelligent design in air interface algorithms, wireless network architecture, and wireless traffic prediction, etc., 6G-based communication network can better achieve network autonomy and intelligence, thus realizing intelligent operation and maintenance management of 6G network infrastructure, including MEC servers and efficient automatic deployment of services [19, 20]. The traffic prediction model based on AI algorithm can automatically mine various features contained in wireless data and comprehensively use these features to accurately predict wireless traffic in real-time. Accurate traffic prediction can directly reflect the spatial and temporal distribution of communication service demands, guide the network resources allocation in different network nodes, and then offload thecomputing tasks in different MEC servers, thereby improving user service experience and enhancing the autonomous and intelligent operation and maintenance of communication networks.

Machine-to-machine (M2M) communications, autonomous driving, and virtual reality are just a few of the new applications that 6G cellular networks are expected to make possible in the next few years. These applications all call for better network latency, capacity, and context awareness. It is critical to make the network aware of traffic demands to achieve these strict standards. The development of an intelligent network necessitates traffic analysis and accurate forecasting of user demand. Knowing user demand ahead of time allows the network to allocate resources more efficiently. The network can manage resource distribution between users who are competing for resources promptly.

The remainder of this paper is organized as follows. Related works closely to our research are listed in Section 2. A traffic prediction model for 6G MEC IoT is described in Section 3. Section 4 focuses on traffic prediction methods with variable sampling rates(VSR) using machine learning and RNN techniques. In this section, the VSR-NLMS adaptive prediction method to perform time series prediction dataset transformation and the VSR-LSTM algorithm for real-time prediction of network traffic are proposed. Simulation and performance analysis are presented in Section 5. Section 6 concludes the full text.

2. Related Works

Network traffic analysis and prediction are fundamental to traffic engineering, network planning, optimization, administration and maintenance, resource allocation, load balancing, etc.. The previous wireless traffic prediction adopted manual prediction and statistical-based prediction methods, which have lots of flaws such as Low efficiency and non-real-time. As machine learning techniques are intensively researched, there have been some works using machine learning and deep learning algorithms for wireless network traffic prediction.

All of [9–14] address network traffic prediction problems in the telecom network. In these research works except [12], LSTM or LSTM variants has been adopted and achieved good prediction performance compared with other algorithms such as ARIMA, SVR, FFNN, RFR, KNNR, etc.In [9–11, 14], the prediction focused on aggregated behavior, e.g., considering traffic volumes observed over a given time interval (normally 5–15 minutes), which is coarse-prediction.

In [12, 13], the author investigates and specializes a set of architectures selected among convolutional, recurrent, and composite neural networks to predict mobile-app traffic at the finest (packet-level/mobile app) granularity. To provide the AI-native services for the 6G vision, the author in [14] proposed a novel edge-native framework to provide an intelligent prognosis model using LSTM-based encoder-decoder for data traffic prediction. The prognosis model was trained on real time-series multivariate data records collected from the edge μ-boxes of a selected testbed network.

In [15–18], the traffic predictions of Internet or campus network, such as ARQ message and ping command, are all fine-grained prediction. Especially in [18], wavelet transform is used to preprocess the data before prediction, transforming the 1-dimensional time series data into 3-dimensional data, which is more conducive to GRU (a LSTM variant) feature extraction and then obtained a better performance than RNN.

A detailed comparison of these related works has been compared according to the aspects of research objects, traffic data granularity, used data sets, and algorithms in this paper, as shown in Table 1.

From the above discussion, it can be seen that compared with other algorithms, LSTM has better performance in predicting time series data such as network traffic. But although not specified in the literature, especially telecommunications network traffic prediction, the traffic data generated using a fixed sampling rate, without considering traffic speed changes, will lead to large complexity of computation or predict the problem of inaccuracy. Aiming at the above problems, a new algorithm, VSR-LSTM, which combines variable sampling rate and LSTM, is proposed in this paper.

In addition, for readers' convenience, all the acronyms in this paper have been collected and listed in Table 2.

3. System Model

As shown in Figure 1, the mobile edge computing system model for machine learning-based traffic prediction is a

Table 1: Related works performing prediction of different research objects and by means of different techniques.

[Ref]	Research object	Fine or coarse	Dataset	Simulator	Techniques
[9]	Telcom network	Coarse (15 min)	GEANTWIDE	Not mentioned	ARIMA SVR LSTM RCLSTM SR-based
[10]	Telcom user traffic and location	Coarse (15 min)	GENAT	Not mentioned	ARIMA SVR LSTM RCLSTM FFNN
[11]	Telcom. Network	Coarse	Operator data	Python	ARIMA FFNN LSTM
[12]	Mobile APP	Fine	MIRAGE-2019	Not mentioned	HMM RFR LR K-NNR
[13]	Mobile APP	Fine	MIRAGE-2019	Not mentioned	RFR MC CNN LSTM GRU
[14]	Mobile 6G network	Coarse (5 min)	Locally obtained	Edge μ-boxes, jupyter notebook	LSTM-based encoder and decoder
[15]	University campus datacenter	Fine	EDU1 dataset	Python, keras	CNN RF DNN
[16]	SDN controller (ONOS)	Fine	Ping ARQ message	Not mentioned	SF LDA SVR
[17]	Internet	Fine	DNS traffic	Python, TensorFlow	BPNN LSTM GRU
[18]	Internet	Fine	User data	MATLAB	RNN

Table 2: Acronyms list in this paper.

Acronyms	Full name
ARIMA	Auto regressive integrated moving average
ARQ	Automatic repeat response
CNN	Convolutional neural network
DNN	Deep neural network
FFNN	Feed forward neural network
FSR	Fixed sampling rate
GRU	Gated recurrent unit
HMM	Hidden markov models
MEC	Mobile edge computing
NRMSE	Normalized root mean square error
KNNR	K-nearest neighbor regressor
LSTM	Long short-term memory
LDA	Linear discriminant analysis
LR	Liner regression
MC	Markov chain
RCLSTM	Random connectivity LSTM
RF	Random forest
RFR	Random forest regressor
RNN	Recurrent neural network
SR-based	Sparse representation-based
SVR	Support vector regression
VSR	Variable sampling rate

three-tier hierarchy consisting of a cloud platform, multiple MEC gateways, and a large number of end-users, including multiple independent IoT networks. Each IoT network serves many end-users (i.e., different end devices). Different terminal types and usage scenarios have different computing, storage, and communication capabilities. For example, smartphones have relatively high computational storage and communication capabilities, and rechargeable batteries have high energy supply capabilities. But some IoT nodes have deficient capacity compared to smartphones, especially many nodes that cannot replace batteries, and their computational storage and communication capabilities are greatly limited. At the same time, various services have different QoS requirements for latency, energy consumption, communication bandwidth, and other indicators, which require different processing methods for different terminal types and different business needs. Generally speaking, services with low computation and power requirements but high latency requirements can be executed on local terminals with strong capabilities, such as smartphones. For IoT nodes with limited computation and power, data can only be transmitted to gateway nodes or edge servers for processing, and for that kind of computationally

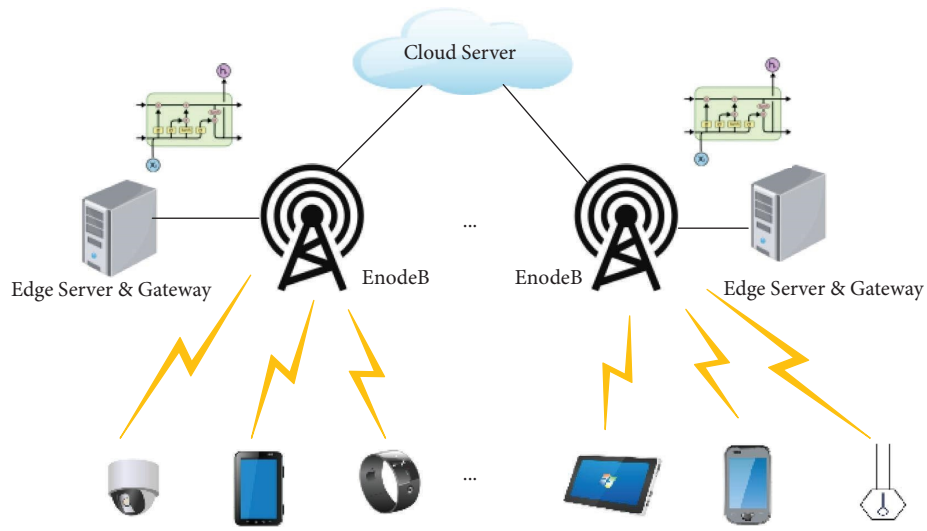


Figure 1: Traffic prediction for 6G MEC IoT.

intensive service, they also need to be offloaded to remote servers with unlimited energy computing power to process them.

In Figure 1, whether IoT end-users offload tasks to edge servers for execution or edge servers offload computationally intensive tasks to remote ends for execution, both require the system to allocate appropriate network bandwidth resources. Although 6G has increased the network speed and bandwidth a lot compared to 5G, the competition for network resources, especially bandwidth resources, still exists, and this resource competition may become more intense in the future, requiring dynamic control of network resources based on user demand. Traffic prediction is of great significance to achieving dynamic resource allocation and is a prerequisite and guarantee for the edge computing server to achieve dynamic resource allocation.

In addition to being able to complete the execution of tasks offloaded to it by users and offload tasks with greater computational demand to the cloud for execution, the edge server in Figure 1 should also have dynamic traffic prediction functions to accurately predict the bandwidth resources required for various offloading tasks and provide a basis for network bandwidth resource allocation. The cloud server mainly completes the computation-intensive tasks offloaded to it by the edge servers.

4. Prediction Methods

4.1. The Significance and Preliminary Solution of Developing Variable Sampling Rate. To avoid confusion in the forecast, if a constant sample rate is to be utilized, it must be double the maximum frequency of the traffic load profile. Since more traffic sample operations likewise generate more traffic prediction operations, a constant sampling rate puts significant computing complexity into the system. As a result, developing a low-sampling-rate solution seems appealing.

The traffic load curve, which has areas with slow and rapid changing movements, served as inspiration for the notion of VSR (variable sampling rate) [21, 22]. We discovered that the main source of traffic load volatility is the timeliness features of mobile user behavior. A notable example of this is the fact that students' real-time traffic on campus is scheduled and determined by the calendar. Then, the real-time network traffic load is also limited by the number of visits and applications. When the number of users fluctuates quickly, the fast-changing zone appears, but when the number of users remains consistent, the slow-changing zone appears. Based on these findings, the best VSR strategy is to sample at a low sampling rate in slowly changing regions and at a high sampling rate in rapidly changing regions. The constant sampling rate approach is often used when considering the entire traffic load distribution, whereas the variable sampling rate strategy is used when the traffic load distribution needs to be sampled and reconstructed independently, a practical scenario where the traffic load varies in speed from region to region. The maximum frequencies in the slow-change and fast-change zones, respectively, are represented by $f^s_{\max}$ and $f^f_{\max}$. From Nyquist's theory, the sampling rate must be twice as fast as $f^s_{\max}$ in the slow-changing region and twice as fast as $f^f_{\max}$ in the fast-changing region. Therefore, the average sampling rate R^{avg}_{VSR}:

$$R^{\text{avg}}_{VSR} = \frac{\left(2f^s_{\max}T_s + 2f^f_{\max}T_f\right)}{\left(T_s + T_f\right)}, \tag{1}$$

of which T_s and T_f represent the total length of time covered by the slow and fast-change zones, respectively. Considering

$$f^s_{\max} < f^f_{\max} = f_{\max}, \tag{2}$$

we have

$$R^{\text{avg}}_{VSR} < 2f_{\max} \tag{3}$$

indicating that the sampling rate of the VSR method is lower than that of the constant sampling rate method.

4.2. VSR-NLMS Adaptive Forecasting Method. As the load profile is unknown at the time of prediction, it is not possible

to classify the load profile into low- and high-speed types. The author in [21] created the VSR-NLMS adaptive forecasting method, which combines the FSR-NLMS (fixed step size-NLSM) predictor with VSR. The sampling rate of time t_n is defined as

$$R_{s(n)} = \frac{1}{\Delta t_{n-1,n}}. \tag{4}$$

in the VSR-NLMS scheme, and it is iteratively updated to show a negative correlation with the target prediction error bound $E_b > 0$. We further have the following constraint

$$R_s^{\min} \leq R_s(n+1) \leq R_s^{\max}, \tag{5}$$

where, $R_s^{\max}$ and $R_s^{\min}$ denote the system's highest and lowest sampling rates, respectively.

As the flow load curve is changing rapidly resulting in large errors, the VSR-NLMS scheme is used here to update the sampling rate in time to ensure accuracy. Based on the subsequent observation, the VSR-NLMS method adjusts the sampling rate. To achieve forecast accuracy, the sampling rate must be raised because the greater the forecast error, the greater the complexity of the traffic load situation. A small prediction error, on the other hand, indicates that the traffic load profile varies slowly, allowing the sampling rate to be adjusted to reduce computing complexity. When deciding on $R_s^{\max}$, we must choose between prediction accuracy and compilation efficiency. Large $R_s^{\max}$ values, obviously, can result in reduced prediction errors, but they also increase the computational complexity of the sampling and prediction method. When we choose $R_s^{\min}$, we infer that the linear predictor's reaction time cannot be too long to handle bursty traffic.

4.3. Dimensional Transformation in Time-Series Prediction. Network traffic prediction is a temporal sequential forecasting technique, the core idea of which is to analyze the nonlinear correlation between the previous data and its historical data at a certain time point [23]. The prediction of values at a future point in time is accomplished based on the outcomes of the modeling analysis. We must use traffic flow time series modeling in this paper. Traffic flow data was converted into multidimensional data, including input feature vectors and model output sample labels in a format to better model LSTM. Scholars now use the "sliding window" approach to convert one-dimensional (1D) data into two-dimensional (2D) data. The following are the primary phases in this method for properly converting 1D time series into 2D machine learning data type.

Step 1: Choose moment T and collect N historical values before moment T and set them as feature vectors. N is the length of the feature vector.

Step 2: Construct the output vector by taking moments $T+1$ to $T+M$ as the label values. M is the number of output variables, which represents the step size of the prediction.

The basic flow chart of the "sliding window" based method is shown in Figure 2.

4.4. RNN and LSTM. The earliest known RNN is Hopfield Network (HN) proposed by Hopfield in 1982 [24]. RNN, as a neural network model that can process time series and structural series data, has been widely used in handwriting recognition, speech discrimination, text translation, and other fields. The data contained in these fields have a common feature. That is to say, the input samples are all continuous sequence data, which can be a piece of text or a piece of speech. There is a great correlation between the preceding and the following, and the length of the data is different, which cannot be accurately divided into separate training samples by traditional neural networks. Compared with the traditional neural network, the RNN model can connect the output at the current moment with the output at the previous moment so that the neural network has the function of "memory". It can record the historical sequence information, continuously reduce the gradient error between the predicted value and the real value in the process of iteration, and finally obtain the optimal model. Normally, any sequence can be obtained by predicting through the RNN model.

However, in the actual training process of the RNN model, it is found that it is still a little insufficient to store the amount of historical information. Since the RNN model stores the corresponding historical information by the number of network layers, the less the number of network layers, the more incomplete the historical information recorded. In addition, the more the number of network layers, the more complex the training process will be, and it is easy to have the phenomenon that the gradient descent speed is fast or even disappears, both of which will lead to poor prediction performance of the model [25].

To solve the above problems, the long short-term memory network (LSTM) model is proposed by Hochreiter et al. in 1997 [26]. Subsequently, the LSTM model quickly made great achievements in speech recognition and machine translation. LSTM, as a variant of RNN, can solve the common problems in the RNN model by changing the structure of the hidden layer. Since the LSTM model is obtained by changing the RNN model, the two models have the same output layer and input layer structure, and the differences are mainly reflected in the structure of the hidden layer.

The structure of the hidden layer of the LSTM model is shown in Figure 3. It can be seen that the hidden layer is mainly composed of three gating units and a memory block (cell) unit, and the three adaptive multiplication gating units are the input gating unit, forgetting gating unit, and the output gating unit, respectively.

The input gating unit I_t can be used to control, where information can be saved to the memory unit at the current moment, and it consists of two network layers with activation functions of sigmoid function σ and tan h function, respectively.

$$\begin{aligned} I_t &= \sigma\left(A_i m_{t-1} + B_i x_t + b_i\right), \\ C_t &= \tan h\left(A_c m_{t-1} + B_c x_t + b_c\right), \end{aligned} \tag{6}$$

where x_t denotes the input vector at moment t, m_{t-1} is the output of the previously hidden layer neuron node, and by

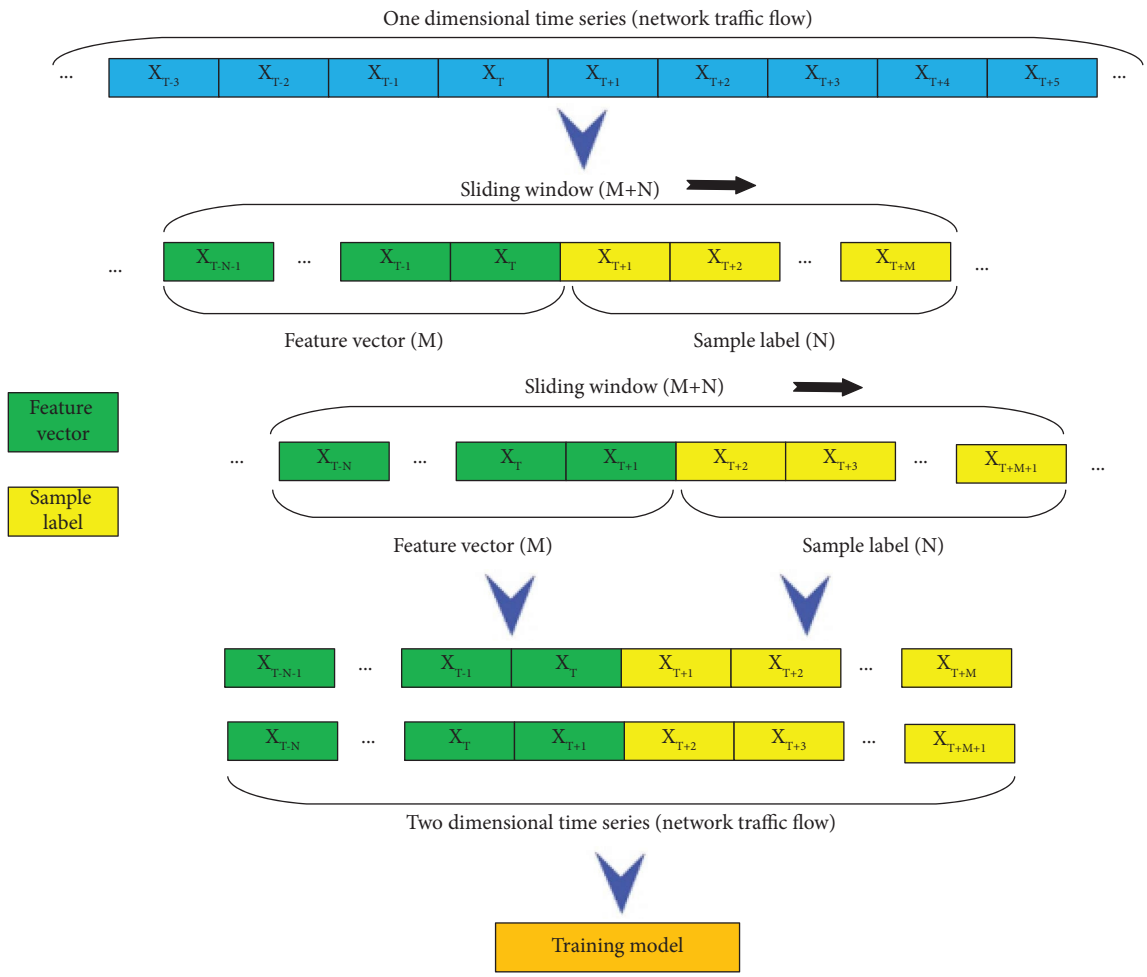


Figure 2: Time series prediction data set transformation.

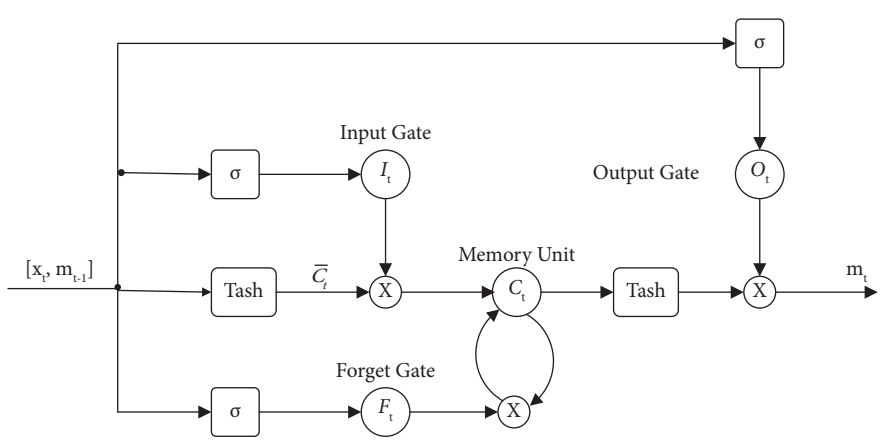


Figure 3: LSTM model hidden layer structure.

using the sigmoid function to activate x_t and m_{t-1}. A_i denotes the weight value corresponding to the output result of the previously hidden layer neuron at the input gate, while B_i is the weight value of input vector x_t and b_i is the bias parameter of the input gating unit at the time of calculation.

$\overline{C_t}$ the candidate cell state, A_c, B_c, and b_c are weight values and the bias parameter of $\overline{C_t}$ respectively.

Next, the result of multiplying the elements corresponding to the two results is used for the update of the memory block cell.

The forgetting gating unit F_t is mainly used to link the state of the memory block unit at the previous moment with the state of the memory block unit at the current moment.

The final result obtained through the forgetting gate

$$F_t = \sigma\left(A_f m_{t-1} + B_f x_t + b_f\right), \tag{7}$$

where A_f denotes the weight value corresponding to the output result of the previously hidden layer neuron at the input forgetting gate. The weight of the information entering the forgotten gating unit through the input gating unit at the current moment is denoted by B_f, and b_f is the bias parameter of the forgotten gating unit at the time of calculation.

The output gating unit is composed of two parts, the current moment input vector combined with the information obtained from short-term memory output results O_t and the output result of the information obtained by combining the input vector with long-term memory at the current moment m_t. The activation functions used are the sigmoid function and the $\tan h$ function, respectively.

$$\begin{aligned} O_t &= \sigma\left(A_\circ m_{t-1} + B_\circ x_t + b_\circ\right), \\ m_t &= O_t \circ \tan h\left(C_t\right), \end{aligned} \tag{8}$$

where, $\circ$ is the element-wise multiplication, which implements the product of the corresponding positional elements of two matrices. As mentioned above, $A_\circ$, $B_\circ$, and $b_\circ$ are weight values and the bias parameter of the O_t respectively.

The state of the memory block unit is determined by both the past moment state and the current moment state, where the past moment state is obtained by multiplying the past moment unit state with the output result of the forgetting gating unit in accordance with the corresponding element, and the current moment state is obtained by multiplying the current moment unit state with the current moment input gating unit in accordance with the corresponding element.

$$C_t = C_{t-1} \circ F_t + I_t \circ \overline{C_t} \tag{9}$$

After the above discussion, we have clearly understood the importance of controlling the sampling rate. The core of this article is to continuously adjust the sampling rate by analyzing the error. The processing method of this article is to apply the LSTM algorithm. How does the LSTM algorithm work in the past? The error value is input, forgotten, and output as a sample, which will be discussed below.

The reason why the LSTM algorithm is different from the traditional RNN algorithm is that it can delete some unnecessary data through the forget gate. Among them, A_i, B_i, and b_i in the input gating unit; A_f, B_f, and b_f in the forgetting gating unit; and A_o, B_o, and b_o in the output gating unit need to be obtained by the LSTM model through training. The training method is as follows: the error at time t is defined in LSTM as ϑ_t and the sum of squares of all node errors in the output layer is represented by L. At this time, we have the following formula

$$\begin{aligned} L &= \frac{1}{2}\sum_{t=1}^{\tau}\left(\hat{y}_t - y_t\right), \\ \vartheta_t &= \frac{\partial L}{\partial m_t}. \end{aligned} \tag{10}$$

Among them, $\hat{y}_t$ is the prediction value, y_t is the true value, and m_t represents the hidden state.

Then, we need to go back in time and use the error of the latter time state to calculate the error of the previous time state.

At state t_n, the error of the input gating unit is

$$\vartheta_{i_t} = \vartheta_t \circ O_t \circ \overline{C}_t \circ i_t \circ \left(1 - i_t\right) \circ \left(1 - \tan h\left(c_t\right)^2\right). \tag{11}$$

The forgetting gating unit error is

$$\vartheta_{f_t} = \vartheta_t \circ o_t \circ c_{t-1} \circ f_t \circ \left(1 - f_t\right) \circ \left(1 - \tan h\left(c_t\right)^2\right). \tag{12}$$

The output gating unit error is

$$\vartheta_{o_t} = \vartheta_t \circ \tan h\left(c_t\right) \circ t \circ \left(1 - o_t\right). \tag{13}$$

The error of the memory block unit is

$$\vartheta_{\overline{c_t}} = \vartheta_t \circ o_t \circ i_t \circ \left(1 - \overline{c_t^2}\right) \circ \left(1 - \tan h\left(c_t\right)^2\right). \tag{14}$$

Using the above four values, the error value of the state t_{n-1} can be calculated as

$$\vartheta_{t-1} = A_o \vartheta_{o_t} + A_f \vartheta_{f_t} + A_i \vartheta_{i_t} + A_{\overline{c}} \vartheta_{\overline{c_t}}. \tag{15}$$

Then, the gradient of the bias term and the weight gradient A, B, and b are derived by the chain rule, and then the gradient is updated by the gradient descent method. To be accurate enough for the calculated A, B, and b values, after training with a sufficient time-span, we use the sampling error of the previous time state E_b as the object to use the LSTM algorithm to predict. When E_b is too high, it means that the sampling accuracy is not enough and we need to increase the sampling rate; when the value E_b is too low, it means that the sampling rate is too high and too many computing resources are wasted. In this case, the sampling rate needs to be reduced.

In addition, it should be pointed out that gated recurrent unit (GRU) is also a variant of RNN, and its structure is similar to LSTM, with one less gate than LSTM, which reduces matrix multiplication and can save a lot of time without sacrificing performance in small dataset scenario. But in the scenario of large datasets, LSTM has better performance than GRU, such as prediction accuracy, recall, AUC, etc. In our study, the dataset obtained from the MEC servers is large, so LSTM with better performance is used for prediction in this paper [27].

4.5. The Proposed VSR-LSTM Model. In this paper, we make full use of the excellent advantage of LSTM in time series prediction, which combines the idea of variable sampling rate, improves the VSR-NLMS algorithm, proposes a network highway traffic prediction model based on LSTM variable sampling rate, named as VSR-LSTM, which contains the following three main parts.

(1) Transform the original data form and divide the training set and test set.

(2) Train the LSTM neural network.

(3) Realize traffic flow prediction and verify the results.

The model's fundamental stages are listed below.

Table 3: Summary of notations in Section 3.

Symbol	Description
T_f	The total length of time covered by the fast-change zones
T_s	The total length of time covered by the slow-change zones
$f^s_{\max}$	Maximum frequencies in the slow-change
$f^f_{\max}$	Maximum frequencies in the fast-change
R^{avg}_{VSR}	Average sampling rate
Rs(n)	The sampling rate of time t_n
E_b	Target prediction error bound
$R^s_{\max}$	The system's highest sampling rates
$R^s_{\min}$	The system's lowest sampling rates
O_t	The output of the model at instant t
I_t	Input gating unit at instant t
F_t	Forgetting gating unit at instant t
C	The internal state
m_t	The hidden state
x_t	Input vector at moment
ϑ_t	The error at time t is defined in LSTM
A_c, A_f, A_o	The weight values of x_t
B_c, B_f, B_o	The weight values of m_{t-1}
b_c, b_f, b_o	The bias parameters
L	The sum of squares of all node errors in the output layer

Step 1: Based on the actual collected traffic flow data, the training set and test set are divided in the ratio of 7 : 3.

Step 2: Based on the basic principle of "sliding window" method, the original one-dimensional traffic data is converted into two-dimensional data.

Step 3: Preprocess the raw data by normalization and other methods.

Step 4: Input the training data into the LSTM network, train the model parameters, and build the prediction model.

Step 5: Input the input vectors of the test set into the trained LSTM neural network and compare the prediction results with the real values to get the error of the model.

The summary of notations in Section 3 is listed in Table 3.

5. Numerical Results

In this section, we implemented the proposed VSR-LSTM and compared it with the traditional fixed sampling rate algorithm FSR-NLSM. Tto facilitate performance comparison, the original fixed sampling step size algorithm FSR-NLMS is also implemented based on LSTM, namely FSR-LSTM.

5.1. Evaluation Setup. We assess the performance of the suggested architecture using a collection of mobile traffic statistics from ten separate MEC servers that we gathered over the course of 1 month. According to Section 4.5, we determine the aggregate cell traffic for each server. We use normalized root mean square error (NRMSE) as a measure of the prediction algorithm's efficacy, which is defined as

$$\text{NRMSE} = \frac{1}{x}\sqrt{\frac{\sum_{t=1}^{N}\left(\tilde{x}_t - x_t\right)^2}{N}}, \tag{16}$$

where $\tilde{x}_t$ and x_t are the predicted value and its corresponding observation at the time t, respectively, and $\overline{x}_t$ is their mean. N is the total number of points. The accuracy of the suggested architecture is compared using the same metric with that found using other prediction algorithms.

The simulation is implemented using Python, and the backend uses TensorFlow and Keras. Table 4 reports the selected hyperparameters. One of the hyperparameters that must be chosen that might influence the trade-off between the amount of time and the prediction accuracy required to train the network is the number of hidden layers, which is fixed at 5. The amount of information that must be maintained and utilized by the network is determined by the connection between the quantity of earlier observed values and the accuracy of the multistep prediction, which is the focus of our attention. The prediction accuracy could be enhanced by adding more layers. We set the total number of epochs to 100 for the same reason. The architecture was trained and validated using three weeks of data. The following results relate to the previous week. We iteratively change the network weights based on the training data using the Adam optimization.

Table 4: Training hyperparameters.

Hyperparameters name	Value
Initial learning rate	0.001
Number of epochs	100
LSTM hidden states	64
LSTM hidden layers	5
Optimization algorithm	Adam
Loss function	MAE

5.2. Results' Analysis. The results of multistep prediction, which involves making predictions for future time instants while delaying the output by a predefined number of timeslots, are then shown. We demonstrate how accuracy suffers when we attempt to forecast traffic statistics for upcoming timesteps. We also look at how the length of the timeslots T and the number of observations that the LSTM network can observe affect the results. These design parameters must be computed since they have an impact on the LSTM network's memory capacity and the amount of traffic data that must be kept for an accurate prediction.

Finally, we compare the proposed algorithm, i.e., VSR-LSTM with a classical time-series network traffic prediction method (FSR-LSTM). For a fair comparison, the same number of hidden layers are used.Figure 4 illustrates the traffic prediction for the same time-span using both methodologies. Using the FSR-NLMS model has lower accuracy because the predictions tend to be closer to the mean of the flow, while VSR-LSTM has a higher flow prediction

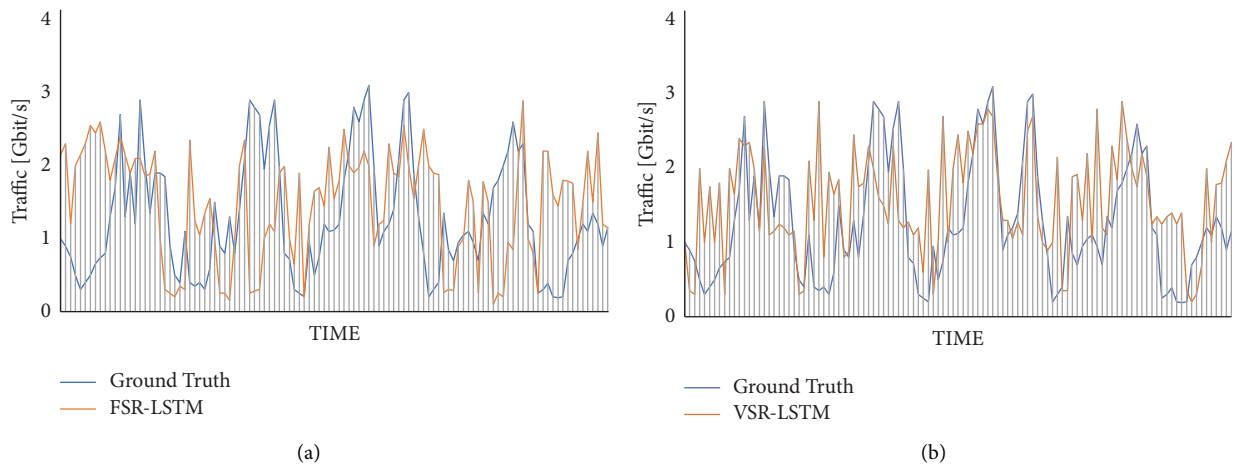


Figure 4: Traffic prediction carve obtained with different models. (a) FSR-LSTM. (b) VSR-LSTM.

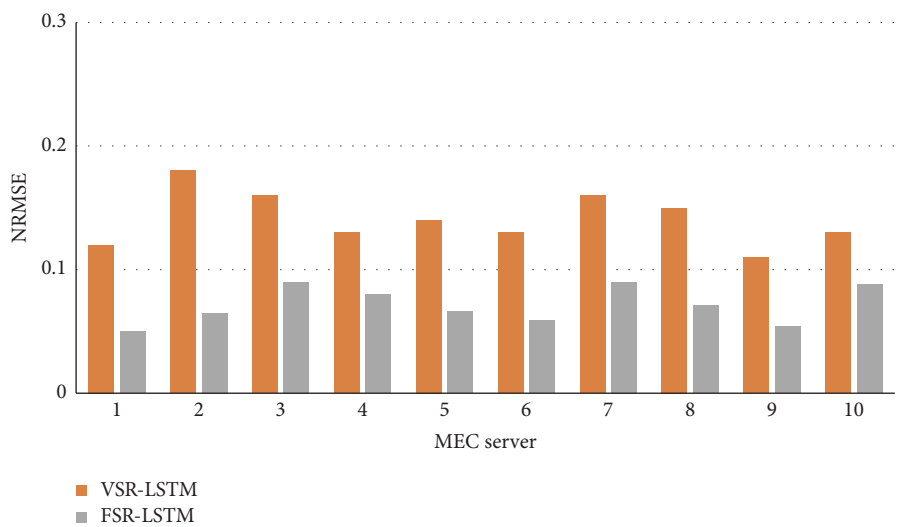


Figure 5: Traffic prediction errors obtained with different models.

accuracy because its sampling speed increases with changes in flow. Furthermore, for the two prediction methods on the 10 flow profiles, we compare their mean errors. As expected, the proposed algorithm obtains less prediction error of the moving flow due to the VSR properties and relative to the FSR-LSTM model, as shown in Figure 5.

The difference in computational complexity between the proposed VSM-LSTM and the baseline FSR-LSTM is mainly due to the difference in sampling rate. Based on the previous analysis in Section 4.1, the average sample rate R_{VSR}^{avg} in VSR-LSTM is lower than the fixed sample rate in FSS-LSTM. For example, if we suppose the $f_{\max}^{f}$ in the traffic load curve is twice the $f_{\max}^{f}$ and also assume $T_s = T_f$, then, according (1), R_{VSR}^{avg} in VSR-LSTM is 25% lower than the fixed sample rate in FSS-LSTM.

6. Conclusions

In this paper, we propose a network traffic prediction method with a variable sampling rate using machine learning and LSTM techniques. In the variable sampling rate case, the sampling rate that determines the accuracy of traffic prediction can change in real time with the dynamic changes of network traffic. Therefore, compared with the traditional traffic prediction methods based on fixed sampling rate, the traffic prediction method proposed in this paper can more accurately reflect the real-time changes of network traffic to further guide the reasonable and effective allocation of network resources.

In the next work, based on the traffic prediction method with variable sampling rate proposed in this paper, we will further investigate the intelligent distributed allocation and management of multidimensional resources for the different demands on resources such as computation, communication, and storage in 6G MEC IoTs networks for different vertical applications in the future.

Furthermore, we should also try to combine our proposed algorithm with Markov chains, hidden Markov models, and other classic prediction theories to improve the prediction accuracy, computational complexity, and other

performances, especially on small granular flow performance, such as a packet, a mobile app, etc. Meanwhile, we also try to extend the application of the novel prediction algorithm to new scenarios such as 6G, digital twin, and metaverse in the future.

Data Availability

The dataset used to support the findings of the study can be obtained from a private company.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Open Foundation of State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) (Funding no. SKLNST-2020-1-10).

References

[1] I. Khan, X. Tao, G. M. S. Rahman, W. U. Rehman, and T. Salam, "Advanced energy-efficient computation offloading using deep reinforcement learning in MTC edge computing," *IEEE Access*, vol. 8, pp. 82867–82875, 2020.

[2] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource allocation with edge computing in IoT networks via machine learning," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3415–3426, April 2020.

[3] M. Merluzzi, P. D. Lorenzo, and S. Barbarossa, "Wireless edge machine learning: resource allocation and trade-offs," *IEEE Access*, vol. 9, pp. 45377–45398, 2021.

[4] J. Han, G. H. Lee, S. Park, and J. K. Choi, "Joint subcarrier and transmission power allocation in OFDMA-based WPT system for mobile edge computing in IoT environment," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 15039–15052, 2022.

[5] Z. Chen, J. Hu, X. Chen, J. Hu, X. Zheng, and G. Min, "Computation offloading and task scheduling for DNN-based applications in cloud-edge computing," *IEEE Access*, vol. 8, pp. 115537–115547, 2020.

[6] P. W. Khan, K. Abbas, H. Shaiba, A. Muthanna, A. Abuarqoub, and M. Khayyat, "Energy efficient computation offloading mechanism in multi-server mobile, edge computing—an integer linear optimization approach," *Electronics*, vol. 9, no. 6, p. 1010, 2020.

[7] I. Lohrasbinasab, A. Shahraki, A. Taherkordi, and A Delia Jurcut, "From statistical- to machine learning based network traffic prediction," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 4, Article ID e4394, 2022.

[8] T. Zhang, L. Feng, P. Yu, S. Guo, W. Li, and X. Qiu, "A handover statistics-based approach for cell outage detection in self-organized heterogeneous networks," in *Proceedings of the 15th IFIP/IEEE International Symposium on Integrated Network and Service Management (IM '17)*, IEEE, Lisbon, Portugal, pp. 628–631, May 2017.

[9] Y. Wang and T. Nakachi, "Prediction of network traffic through light-weight machine learning," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 1919–1933, 2020.

[10] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 114–119, 2019.

[11] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile traffic prediction from raw data using LSTM networks," in *Proceedings of the 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1827–1832.

[12] G. Aceto, G. Bovenzi, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescape, "Characterization and prediction of mobile-app traffic using Markov modeling," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 907–925, 2021.

[13] A. Montieri, G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "Packet-level prediction of mobile-app traffic using multitask deep learning," *Computer Networks*, vol. 200, no. 2021, Article ID 108529, 2021.

[14] S. Zeb, M. A. Rathore, A. Mahmood, S. A. Hassan, J. W. Kim, and M. Gidlund, "Edge intelligence in softwarized 6G: deep learning-enabled network traffic predictions," in *Proceedings of the 2021 IEEE Globecom Workshops (GC Wkshps)*, December 2021.

[15] T. Ko, S. M. Raza, D. T. Binh, M. Kim, and H. Choo, "Network prediction with traffic gradient classification using convolutional neural networks," in *Proceedings of the 2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pp. 1–4, Taichung, Taiwan, January 2020.

[16] J. Kwon, D. Jung, and H. Park, "Traffic data classification using machine learning algorithms in SDN networks," in *Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1031–1033, Jeju, Korea, October 2020.

[17] H. Lu and F. Yang, "Research on network traffic prediction based on long short-term memory neural network," in *Proceedings of the 2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, pp. 1109–1113, Chengdu, China, December 2018.

[18] J. Fan, D. Mu, and Y. Liu, "Research on network traffic prediction model based on neural network," in *Proceedings of the 2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pp. 554–557, Dalian, China, September 2019.

[19] J. Kaur, M. A. Khan, M. Iftikhar, M. Imran, and Q. Emad Ul Haq, "Machine learning techniques for 5G and beyond," *IEEE Access*, vol. 9, pp. 23472–23488, 2021.

[20] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao, and K. Wu, "Artificial-intelligence-Enabled intelligent 6G networks," *IEEE Network*, vol. 34, no. 6, pp. 272–280, November/December 2020.

[21] B. Rong, S. Sun, and M. Kadoch, "Traffic prediction for reliable and resilient video communications over multi-location WMNs," *Journal of Network and Systems Management*, vol. 24, no. 3, pp. 516–533, 2016.

[22] S. Wu, W. Mao, C. Liu, and T Tang, "Dynamic traffic prediction with adaptive sampling for 5G HetNet IoT applications," *Wireless Communications and Mobile Computing*, vol. 2019, pp. 1–11, 2019.

[23] Y. Zhao, "Highway traffic flow prediction based on simple recursive unit network," *China's transportation information*, no. 02, pp. 123–126, 2022.

[24] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities"," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[25] K. Om, S. Boukoros, A. Nugaliyadde et al., "Modelling email traffic workloads with RNN and LSTM models," *Human-centric Computing and Information Sciences*, vol. 10, no. 1, pp. 39–13, 2020.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] S. Yang, X. Yu, and Y. Zhou, "LSTM and GRU neural network performance comparison study: taking yelp review dataset as an example," in *Proceedings of the 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, pp. 98–101, Shanghai, China, June 2020.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 4955498, 11 pages
https://doi.org/10.1155/2022/4955498

Research Article

Edge Intelligence-Based RAN Architecture for 6G Internet of Things

Yang Liu, Qingtian Wang, Haitao Liu, Jiaying Zong, and Fengyi Yang

China Telecom Research Institute, Beijing 102209, China

Correspondence should be addressed to Qingtian Wang; wangqt08@chinatelecom.cn

Received 14 July 2022; Accepted 26 September 2022; Published 15 November 2022

Academic Editor: Bo Rong

Edge Intelligence, which blends Artificial Intelligence (AI) with Radio Access Network (RAN) and edge computing, is recommended as a crucial enabling technology for 6G to accommodate intelligent and efficient applications. In this study, we proposed Edge Intelligent Radio Access Network Architecture (EIRA) by introducing new intelligence modules, which include broadband edge platforms that allow policies to interact with virtualized RAN for various applications. We also developed a Markov chain-based RAN Intelligence Control (RIC) scheduling policy for allocating intelligence elements. Experimental results justified that the virtualized RAN delivers on its performance promises in terms of throughput, latency, and resource utilization.

1. Introduction

The Internet of Things (IoT) contains a bunch of Information sensors, radio frequency identifiers, global positioning devices, infrared sensors, and lasers. To realize the pervasive connection between objects and people, a device must first gather the necessary data, which can include sound, light, heat, electricity, mechanics, chemistry, biology, location, and so on, and then complete the process of intelligently perceiving, identifying, and communicating.

The IoT is an information carrier based on the Internet, traditional telecommunication networks, etc. It enables all common physical objects that can be independently addressed to form an interconnected network. The future IoT will have a deep economical, commercial, and social impact on our lives. The IoT integrates billions of smart devices that can communicate with one another with minimal human intervention. However, the crosscutting nature of IoT systems and the multidisciplinary components involved in the deployment of such systems have introduced new security challenges. Implementing security measures, such as encryption, authentication, access control, and network and application security, for IoT devices and their inherent vulnerabilities is ineffective.

From another point of view, IoT also refers to a network of interconnected computing devices and other endpoints that can exchange data with one another and with other devices and systems over the Internet. Using it, any set of individually addressable everyday physical things can be turned into a distributed system. Economically, commercially, and socially, the future IoT will have profound effects on our lives. The IoT connects billions of devices that may exchange data with one another autonomously, requiring just minimal human oversight. Approximately 50 billion devices have been connected to the Internet by the end of 2020, making IoT one of the fastest-growing sectors in the history of computing.

Cellular network-enabled IoT will make a revolution in our future life. It will pave the way for advanced wireless systems and innovative new services [1]. The widespread implementation of the Internet of Things relies on several enabling technologies, such as 5G and 6G. Recently, the fifth-generation (5G) networks have been globally deployed in practice, while it is still predicted that the 5G may not be able to satisfy the demand of an increasing number of future communications [2, 3]. Researchers from both industry and academia are focusing on the sixth-generation (6G) networks. Compared with 5G, 6G networks not only with the features of high synchronization accuracy, near

100% coverage, and higher bandwidth [4] but also with cloud computing and Artificial Intelligence (AI), especially with the interaction of RAN and computing. Edge Intelligence (EI) is being considered an enabling technology for 6G [5], especially in Radio Access Network (RAN) area, because EI pushes the AI algorithms on the devices or network edge [6]. It is a benefit for the application to be processed with low latency and RAN resources scheduled on the edge.

Regarding the advantages of ultra-low latency and high bandwidth offered by edge computing in the 5G era, edge computing had attracted lots of attention; specifically, edge computing handles the applications by migrating the cloud computing ability into the edge. While at the current 5G, edge computing and network functions are still not merged deeply, this is because edge computing is hard to obtain real-time information from RAN [7]. In addition, the network functions are not beneficial from edge computing to improve the intelligence of RAN in network operation and maintenance. Therefore, edge intelligence has received attention from 6G, and AI has been applied as solutions in the network including radio resource management, mobility management, and orchestration [8]. For example, reinforcement learning is applied in the network to decide on network planning, and federated learning is used to protect the privacy of data. AI not only optimizes the communication resource but also configures the network adaptively and responds quickly [9].

For the vision that EI merges AI into edge computing at the network edge, specifically, EI can be considered as one of the evolution routes for AI natives. On the one hand, EI schedules the computing resource for applications; on the other hand, the resources of RAN can also be orchestrated by EI. For 6 G, networks need to support the new service with more stringent and diverse Quality of Service (QoS) requirements and mass connectivity. Hence, it is necessary to construct EI in future 6G networks.

In this paper, we intend to present the Edge Intelligence-based RAN Architecture (EIRA) towards 6G and build a testbed that takes advantage of both intelligence and virtualization. Specifically, we apply micro-services technologies for orchestrating computing and storage resources. Moreover, virtualization technologies are applied to implement the virtualized RAN. In the experiment, we build the testbed and evaluate the performance of functionalities in the testbed, for virtualized RAN, and intelligence use cases. This paper makes the following contributions:

(i) We present an Edge Intelligent RAN Architecture (EIRA) towards 6G

(ii) We propose extensive and custom edge platforms that embedded AI algorithms to process the applications with various latency and computing resource requirements

(iii) We design a RAN Intelligence Control (RIC) resource scheduling policy between extensive edge platform and custom edge platform to improve the request accepted ratio

(iv) We conduct a testbed in practice and evaluate the performance of network functionalities and intelligence use cases

The remainder of this paper is organized as follows. Section 2 reviews the existing work in the related research field, then Section 3 discusses the architecture of our proposed EIRA. In Section 4, we propose a RIC scheduling policy in the EIRA, followed by experimental results illustrated in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

Edge Intelligence is still in its developing stage; in this section, we first survey edge intelligence and attempt to give the initial vision of edge intelligence. Then, we outline the artificial intelligence algorithms applied in the RAN to enhance performance, i.e., reinforcement learning and federated learning. Last, we investigate the architecture for edge intelligence.

As the most promising solution for 6G, edge intelligence has attracted significant attention, and an increasing number of studies about edge intelligence have been proposed in recent years [10–12]. In June 2020, the University of Oulu released the 6G white paper on edge intelligence [10]. Peltonen et al. discuss the infrastructure and platforms for edge computing and present the seven levels of edge intelligence. They also comprehensively analyze the key enablers for edge intelligence for 6G. Additionally, they outline the future directions of 6G edge intelligence. Liu et al. [11] provide an overview of Multiaccess Edge Computing (MEC) in 5G and IoT and discuss edge intelligence in 5G and IoT. They propose a use case named proximity detection with edge intelligence in an IoT environment. In order to reduce the communication cost, they apply five kinds of neural networks and observe that the Long Short-Term Memory neural network and Gated Recurrent Unit have the best prediction accuracy. Nguyen et al. [12] review the fundamental 6G technologies for IoT and discuss the roles of 6G in IoT applications.

In order to get the potential performance improvement in 5G or the upcoming 6G, some works focus on applying AI techniques in 6G. The study in [13] proposes an intelligent reflective surface-based 6G wireless network infrastructure for energy-efficient and sustainable 6G development. Li et al. [14] propose a deep reinforcement learning approach to optimize the coverage ratio in 6G-based IoT networks. They first present a genetic algorithm to maximize the data coverage ratio and then apply deep reinforcement learning to optimal route policy. The experiments show that the proposed method can reduce the length of the collection path and cost. She et al. [15] develop an architecture enabling device intelligence, edge intelligence, and cloud intelligence to achieve Ultra-Reliable and Low Latency Communication in 6G networks. The authors apply Deep Neural Networks for training and federated learning for improving learning efficiency. Compared with the two algorithms and optimal performance, the experiments show that the proposed deep learning approach shows better performance than the

compared algorithms and is close to optimal. Prathiba et al. [16] propose federated learning for computation offloading and resource management within heterogeneous systems. To save the resource cost and improve available resource utilization, they present the federated Q-Learning.

We forward our focus on the studies about AI-enabled architecture for 6G. Han et al. [17] propose an AI-enabled RAN architecture, and then they design a series of functions to maximize the potential gain. In addition, they implement the transceiver with AI and verify that the constellations designed by AI provide a better error rate than the conventional Quadrature Amplitude Modulation constellations. Yang et al. [18] present a four-tier AI-enabled architecture for 6G to fulfill smart resource management, automatic network adjustment, etc. They also discuss edge intelligence in edge computing and cloud AI in the central cloud. Xu et al. [19] demonstrate the machine-learning-based cyber twin architecture for the 6G-enabled Industrial Internet of Things. They, then present a deep reinforcement learning approach to evaluate systematic trial and error in the cyber twin world. The experiment results show that the proposed system has better performance in terms of computing delay and communication delay.

From the review above, it can be seen that the existing research works on edge intelligence are in infancy; moreover, the AI algorithms are applied as a solution in resource allocation, computation offloading for RAN, and so on. However, the previous studies do not consider establishing an overall architecture for edge intelligence-based RAN and building a testbed to promote 6G-related research. To this end, in the paper, we propose a testbed architecture for 6G IoT and deploy it in practice.

3. System Architecture

In this section, the proposed Edge Intelligent RAN Architecture (EIRA) towards 6G is illustrated, as shown in Figure 1. In EIRA, virtualization and service-based architecture are applied to orchestrate the network elements statelessly. We then introduce the potential deployment plan for EIRA.

EIRA is a four-layer architecture including the Ubiquitous Connection layer, Cloud-Network Resource Pool layer, Edge Intelligence Cloud Platform (PICP) layer, and Applications layer. In particular, the Edge Intelligence Cloud Platform layer is used to achieve intelligence native to inner circulation within the architecture. Especially, the PICP obtains the information from RAN, and then, the PICP analyzes the information and makes the decision to RAN in an intelligent way.

3.1. Ubiquitous Connection Layer. This layer supports multiple access styles including Terrestrial and Nonterrestrial networks. The User Equipment (UE) accesses this layer via a terrestrial style, such as 6G, optical, Wireless Local Access Network (WLAN), or nonterrestrial style like the satellite.

In order to strengthen transmission efficiency and network performance, Cell-Free massive Multiple-Input Multiple-Output (CF mMIMO) is considered in this layer. CF mMIMO removes the concept of cellular or cell and introduces the user-centric design. It is also beneficial for high network connectivity and coverage, huge spectrum, and energy efficiency.

3.2. Cloud-Network Resource Pool Layer. The layer contains heterogeneous resource pools and resource virtualization. A heterogeneous resource pool consists of general and dedicated resources with centralized management. General resources contain common and standardized hardware (i.e., industrial servers based on X86 or ARM CPU) and diversified hardware chips with scalability, including acceleration and clock resource chips. For RAN, high-speed processing and a large number of dedicated resources are required, such as Field Programmable Gate Array (FPGA) for coding and encoding. The clock resources are applied to fulfill synchronization accuracy among network elements and UEs. Dedicated resources (e.g., ASIC chips) provide specialized services for a small number of facilities with large capacities and ultra-high performance requirements.

The resource virtualization part abstracts these resources into virtual resources, i.e., virtualized Computing (vComputing), virtualized Storage (vStorage), virtualized Networking (vNetworking), and virtualized Accelerating (vAccelerating). This part eliminates the difference between general resources and dedicated resources; furthermore, it has a unified view of the heterogeneous resources.

3.3. Edge Intelligent Cloud Platform. On the Edge Intelligent Cloud Platform, the AI models are trained and deployed. Regarding the requirements (i.e., latency and computing abilities) of 6G scenarios, we divide the Edge Intelligent Cloud Platform into a Custom Edge Platform and an Extensive Edge Platform. Specifically, Near Real-Time RAN Intelligence Control (Near-RT RIC) is applied in custom Edge-Platform, and Non Real-Time RAN Intelligence Control (Non-RT RIC) is in Extensive Edge-Platform.

Custom Edge-Platform consists of Service-Based Architecture RAN (SBA RAN), Lightweight Core Nets, Trained AI Model, and Near-RT RIC. Particularly, those components are deployed in this platform as Network Functions. SBA RAN is considered in this platform to realize flexible and rapid deployment of RAN. Lightweight Core Net holds a key role in realizing the full potential of 6G services. To process the application in the intelligent method, the trained AI models are embedded in the Edge Intelligent Cloud Platform. Once the requirements are from UEs, the Near-RT RIC deployed in the Edge Intelligent Cloud Platform chooses the trained AI model to process the application. In particular, Near-RT RIC processes the application within low latency.

The Extensive Edge Platform contains Security Capability, Unified Application Programming Interface (API), Date collector, AI model, and Non-Real Time RAN Intelligent Controller (Non-RT RIC). Security Capability protects the processed, transmitted, and stored information in the EIRA. Functions and the Extensive Edge Platform

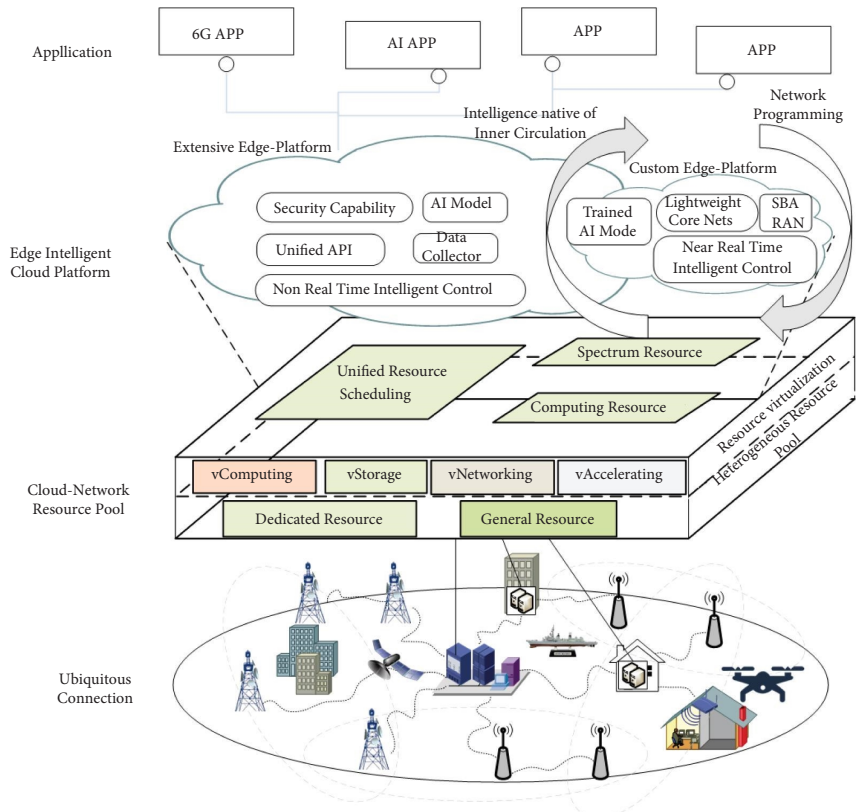


Figure 1: 6G edge intelligent RAN architecture (EIRA).

communicate with each other through a Unified API. Non-RT RIC has features with data collector and model training. The Non-RT RIC collects the network and application data via Unified API, and models are trained in AI models. Moreover, the Non-RT RIC communicates with Near-RT RIC via the Unified API. On one hand, Non-RT RIC sends the trained AI model to the Near-RT RIC; on the other hand, the Near-RT RIC returns feedback to the Non-RT RIC about the updated AI model. Consequently, Near-RT RIC and Non-RT RIC control the RAN with an intelligent method, furthermore making RAN programmable.

3.4. Applications Layer. The future 6G scenarios are deployed in the application layer, and we provide the Unified API for 3rd party to fulfill their application. In Figure 1, we list some applications, such as 6G APP and AI APP.

3.5. Practical Deployment. In order to apply our proposed EIRA into practice, we illustrate the potential deployment plan for EIRA, as shown in Figure 2. As usual, the Extensive Edge Platform could be deployed in the rich computing resource, because it needs to train the AI models. The Custom Edge Platform closes the user to process the application within the requirements of latency. Figure 2 shows the distributed deployment for EIRA, the Extensive Edge Platform in the regional cloud, and Custom Edge Platform in the edge cloud. Especially, an Extensive Edge Platform connects multiple Custom Edge Platforms via wired access style (e.g., optical network). Extensive Edge Platform is able to send the trained AI models to a Custom Edge Platform or multiple Custom Edge Platforms. In addition, Extensive Edge Platform and Custom Edge Platform can also be deployed together at the edge cloud, and it depends on the demand of users.

4. RIC Resource Scheduling Policy

In order to save the storage resource in Custom Edge Platform and improve the acceptance ratio of application requests, in this section, we introduce a scheduling policy-based Markov chain between Non-RT RIC and Near-RT RIC.

4.1. System Model. As shown in Figure 2, we consider the Custom Edge Platform is deployed in the edge cloud and close to the user. The users access the Custom Edge Platform via RAN. The Extensive Edge Platform in the regional cloud connects with multiple Custom Edge Platforms via optical framework. In particular, a Non-RT RIC connects with m Near-RT RICs by a set $\{m = 1, 2, 3, \ldots, M\}$. A Near-RT RIC covers n users denoted as set $\{n = 1, 2, 3, \ldots, N\}$. Let the service requests to a Near-RT RIC be represented by a set $R = \{r_1, r_2, \ldots r_j\}$. Let $r_i = \{\alpha_i, \beta_i, \gamma_i, \theta_i, c\}$ denote a service i request, where α_i is a kind of AI model in xApp that is requested, β_i is the requested process time, γ_i is the access point of the RAN, θ_i is the size of the request, and c_j is the requested computational resource. We assume that only an

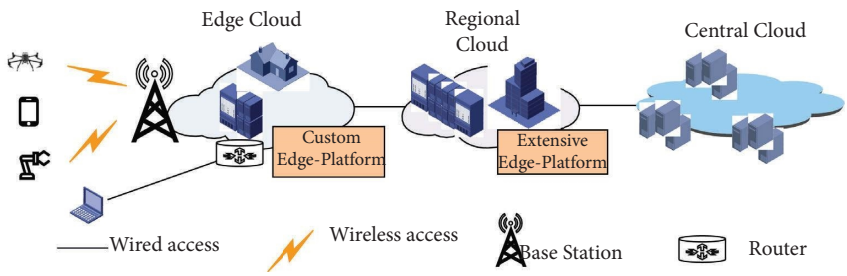


Figure 2: Practical EIRA framework in deployment.

AI model is in an xAPP. In order to maximize the acceptance ratio of the request, we need to predict the AI models of requests in the Near-RT RIC. Therefore, which kinds of xAPP deployed in the Near-RT RIC should be predicted. The acceptance ratio a is calculated as follows:

$$a = \frac{R_a}{R_t}, \tag{1}$$

where R_a is the accepted request and R_t is the total request.

For an accepted request, the processing time should be less than the requested process time β_i. The processing time consists of access time and calculating time.

$$\begin{aligned} &\beta_i \le t_i, 1 < i < j, \\ &t_i = t_{\text{access}} + t_{\text{calculatingtime}}. \end{aligned} \tag{2}$$

The access time is

$$\begin{aligned} t_{\text{access}} &= \frac{\theta_i}{b}, \\ b &= B\log_2(1+\lambda), \end{aligned} \tag{3}$$

where θ_i is the size of the request, B is the bandwidth of the wireless link and λ is the signal-to-inference noise ratio (SNR) of the wireless link.

The calculating time is

$$\begin{aligned} t_{\text{calculatingtime}} &= \frac{c_p}{c_i}, \\ c_p &= c_{\text{total}} - c_{\text{occupied}}, \end{aligned} \tag{4}$$

where c_p is the provided computational resource, c_{total} is the total computational resource that Custom Edge Platform can provide, and c_{occupied} is the occupied computational resource.

Let us consider a Near-RT RIC, the xAPPs set is denoted as $C = \{c_1, \ldots, c_s\}$. We assume that all the xApps in the Near-RT RIC have the same size. Suppose that a Near-RT RIC contains S xApps and a Non-RT RIC contains L AI models. The M xApps are stored in the Near-RT RIC, because of the limited storage resource, $M < S < L$. Let $P = \{p_1, \ldots, p_s\}$ be the set of xAPP popularity for the users, $0 \le p_k \le 1$, $k \in [1, s]$. We count the total acceptance number of requests within a period and record the acceptance requests for xAPP c_k; thereby, the p_k is shown as follows:

$$P_k = \frac{R_{a,k}}{R_{a,S}}, \tag{5}$$

where the number of the acceptance requests for c_k could be defined by $R_{a,k}$ and the total acceptance requests could be defined by $R_{a,S}$.

The xAPPs replacement strategy is the Least Recently Used (LRU). According to the LRU, the number of xAPPs of the Near-RT RIC is regarded as a stack of length $s+1$, where position 1 is at the top of the stack and position s at the bottom of the stack. Whenever a request is accepted, the least requested xApp at the bottom position is replaced and the object xAPP is placed at the top position.

We assume that *t-th* xApp is at position l of the stack. When a request is accepted by the Near-RT RIC, the following three changes are then possible:

(i) l will be moved to the top, which means that the t-th xApp is called to process the request

(ii) l will remain at the same position if the xAPP with position b is called, and $b < l$, $b, l \in [1, s+1]$

l will be moved down by one position if the xAPP with position d is called, and $l < d$, $d, l \in [1, s+1]$

4.2. LRU Model. Let us consider a Markov chain $\{X_{s+1}\}$ with $s+1$ states, as shown in Figure 3. This chain represents the state transition for xApp, where state 1 means that the object is the most frequently called and state s means that it is the least called. The state $s+1$ means that the called xApp is absent. According to reference [20], the $\{X_{s+1}\}$ is irreducible and aperiodic and $X_1, \ldots X_s$ are independent of each other. The probability of *t-th* xAPP at the position l is

$$P^l = p_k, \tag{6}$$

where p_k is the popularity of *t-th* xAPP.

The probability of position l to 1 is

$$\begin{aligned} P^{l \longrightarrow 1} &= p_k, \\ l_t &\in [1, s+1]. \end{aligned} \tag{7}$$

The probability of state l to $l+1$ is equal to the sum of xApp from $l+1$ to s

$$\begin{aligned} P^{l \longrightarrow l+1} &= \sum p_k, \\ l_t &\in [1+1, s]. \end{aligned} \tag{8}$$

Particularly, if $l = s+1$,

$$P^{l \longrightarrow l+1} = 1 - P^l. \tag{9}$$

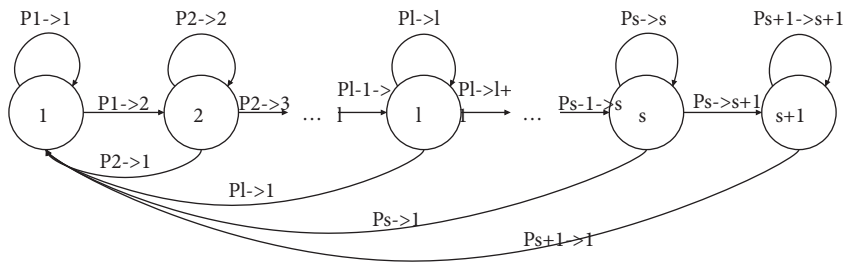


Figure 3: The state transition.

```
Input: requests, S xApps
Output: M xApps
(1) Calculate the acceptance requests by the subject (2)
(2) Calculate the popularity of S xApps by (5)
(3) Calculate the state transition probability by (8)–(11)
(4) Construct the transition matrix P
(5) Find the stationary distribution π by Chapman-Kolmogorov equations
(6) Rank the element π_i increasingly
(7) Output the first M π_i
```

Algorithm 1: RIC resource scheduling policy.

The probability of state l to $l-1$ is equal to the sum of xApp from l to l.

$$P^{l\longrightarrow l} = \sum p_k, \quad l_k \in [1, l-1], \quad l \in [2, s]. \tag{10}$$

We construct the transition matrix P, where the row and column of the matrix are $s+1$ and $s+1$, as follows:

$$P = \begin{bmatrix} p_{11} & p_{12} & 0 & 0 & \dots & 0 \\ p_{21} & p_{22} & p_{23} & 0 & \dots & 0 \\ p_{31} & 0 & p_{33} & p_{34} & \dots & 0 \\ & \vdots & & & & \\ p_{l1} & \dots & p_{ll} & p_{ll+1} & \dots & 0 \\ & \vdots & & & & \\ p_{s+1\,1} & 0 & 0 & \dots & 0 & p_{s+1s+1} \end{bmatrix}. \tag{11}$$

According to Chapman–Kolmogorov, we have

$$\pi_i = \sum_{j}^{s+1} \pi_j P_{j,i} 1 \le i \le s+1, \tag{12}$$

$$\sum_{i=1}^{s+1} \pi_i = 1. \tag{13}$$

Then, the stationary distribution π can solve (12) and (13). For the Near-RT RIC, we calculate the stationary of xApps and then sort the π_i increasingly. At last, we deploy M xApps in the Near-RT RIC with high values of M π_i. The pseudo is given in Algorithm 1.

5. Experimental Results

5.1. Experimental Environment. This section builds an experimental testbed of the proposed Edge Intelligence-based RAN architecture. We evaluate the overhead of network performance with real-time kernel latency and CPU resource usage. Figure 4 shows the testbed hardware devices. Since it is the initial period to discuss the 6G RAN design, the virtualized RAN is established based on 5G NR and 5GC, which can be later updated to 6G RAN. The performance of network latency and throughput is verified within EIRA. In order to evaluate the proposed RIC resource scheduling policy, we consider five application scenarios and set the value of $M = 2$. After this, we analyze the performance of face detection on the platform.

The testbed consists of an all-in-one server, a switch, and an RRU, and the virtualized BBU (vBBU)/lightweight virtualized 5 GC (v5GC) are deployed on the edge intelligence platform in the all-in-one server as virtualized network functions (VNFs). Two same commercial terminals Huawei Mate30 are used. In the test, terminals access the testbed via a 5G NR air interface, and the RRU connects with a switch via optical hybrid cable; moreover, the switch connects with an all-in-one server via optical fiber. The transmission configuration of RAN, the physical configuration of an all-in-one server, and the configuration of the commercial terminal are displayed in Tables 1–3, respectively.

The test is executed in uplink (UL) with the frequency band 3.5 GHz, the bandwidth 100 MHz, and the transmission mode of Time Division Duplex (TDD). Particularly, the frame structure adopted is DSUUU (1D3U) with a 2.5-millisecond single period to achieve the aim of large-capacity transmission for UL, and the number of UL layers is 2. In

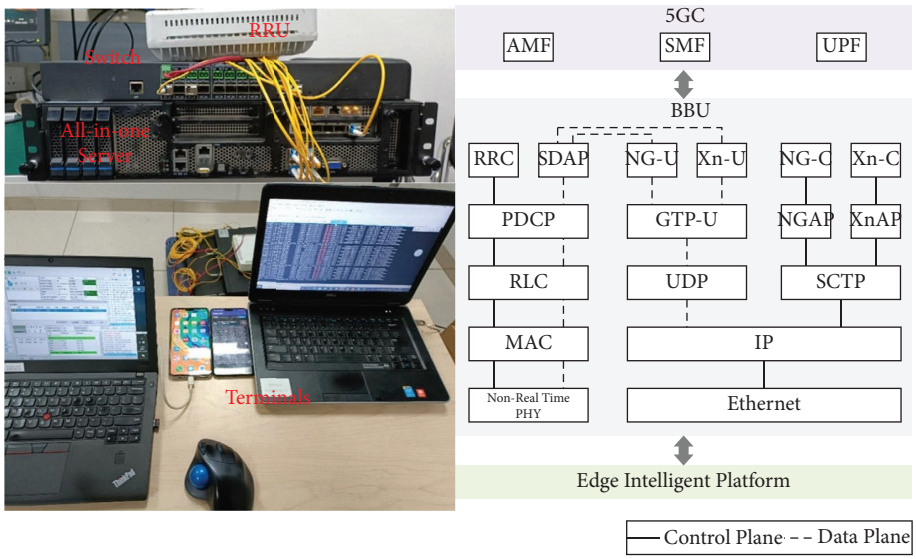


Figure 4: EIRA hardware testbed with a 5G NR protocol stack.

Table 1: Transmission configurations of RAN.

Parameters	Values
Carrier frequency	3.5 GHz
System bandwidth	100 MHz
Transmission mode	TDD
Frame structure	2.5 ms single period DSUUU
Transmission/receiving antennas	2/2
Number of layers for UL	2

Table 2: Physical configurations of the all-in-one server.

Parameters	Configurations
Hardware	Standard 2U server
CPU	2 Xeon(R) silver 4216 CPUs @2.1 GHz 64 RAM
Operating system	CentOS real-time kernel system
Deployed applications	5G vBBU, v5GC (AMF + SMF + UPF), iperf, face detection

addition, the all-in-one server in this experimental environment is a standard 2U server, which has 2 Xeon(R) Silver 4216 CPUs, and two cards of accelerators for channel encoding/decoding and fronthaul processing separately are applied. For the operation system, the CentOS real-time kernel operating system is used to meet RAN real-time requirements. The applications deployed on the all-in-one server with container include iPerf and face detection, with the same deployment modes of vBBU and v5GC and the testbed supports for scaling of the container.

5.2. Evaluation of Network Functionalities. In the experiments, the Key Performance Indicator of real-time kernel latency is used to evaluate the performance of vBBU, which is because real-time kernel latency can reflect the jitter for threaded interrupts, and it can show the processing performance for vBBU. To satisfy the real-time requirement of vBBU, eight CPU threads for parallel computing are used, and the statistics of real-time kernel latency are shown in Figure 5. The duration time for statistics is two minutes, and the statistics results of each thread are basically below 13 us, which satisfies the maximum latency of 20 us in the standard of O-RAN [21].

Table 3: Configurations for the commercial terminal.

Parameters	Configurations
CPU	8-0043ore 2 × Cortex-A76 @ 2.86 GHz + 2 × Cortex-A76 @ 2.09 GHz + 4 × Cortex-A55 @1.86 GHz
GPU	16-Core Mali-G76
NPU (neural network processing unit)	Dual-core NPU
Operating system	EMUI10
Storage	8 GB RAM + 128 GB ROM

The all-in-one server consists of two 16-core CPUs (i.e., 32 physical cores) to deploy an edge intelligence-based platform, vBBU, v5GC, and other components. Hyper-threading technology is applied to generate 64 logical cores. Table 4 lists the usage of CPU when the vBBU is unloaded (i.e., UE access the network and no service is performed) and the vBBU is fully loaded (i.e., UE access the network and uplink service is performed), respectively, and the results are an average of 20 runs. The CPU resource occupied by vBBU at full load is higher than that in the case of no-load because the physical layer is required for packet processing and scheduling at full load. While the resource occupation of v5GC for no-load and full-load is more or less the same, since DPDK is applied for fast packet processing, that is, DPDK continuously occupies the corresponding CPU resources. Due to the fewer number of test users, the resources occupied by v5GC are the same in the two cases. When the

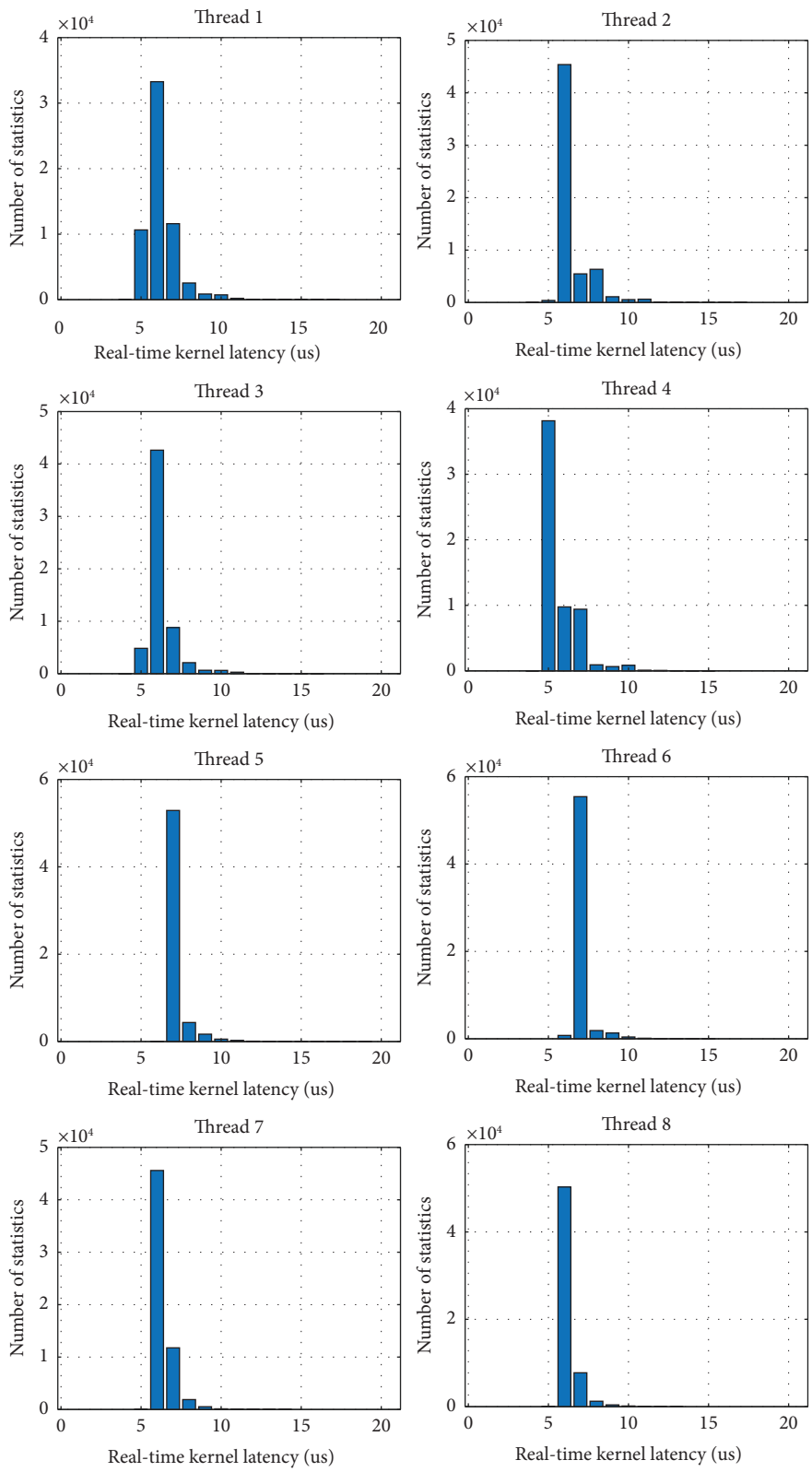


Figure 5: Statistical performance of real-time kernel latency for eight CPU threads.

Table 4: Statistics of CPU resource usage.

Number of CPU cores occupied	vBBU	v5GC	Platform and other components	Total
No-load vBBU	3.90	4.53	2.92	11.35
Full-load vBBU	6.59	4.53	3.47	14.59

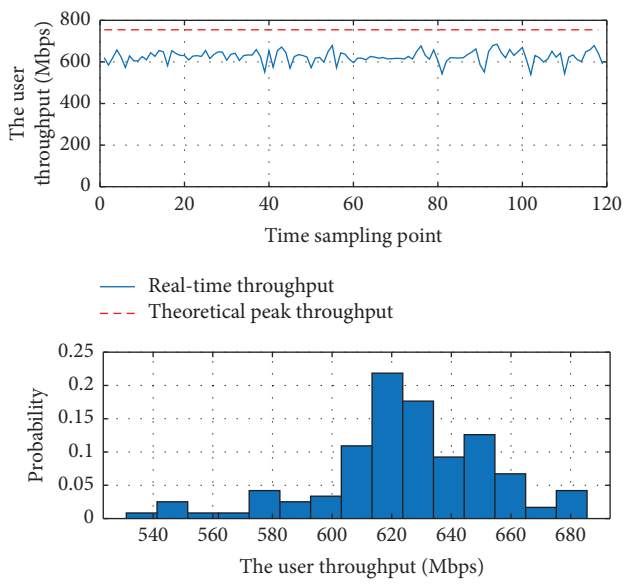


Figure 6: UL rate sampling and probability distribution of single users.

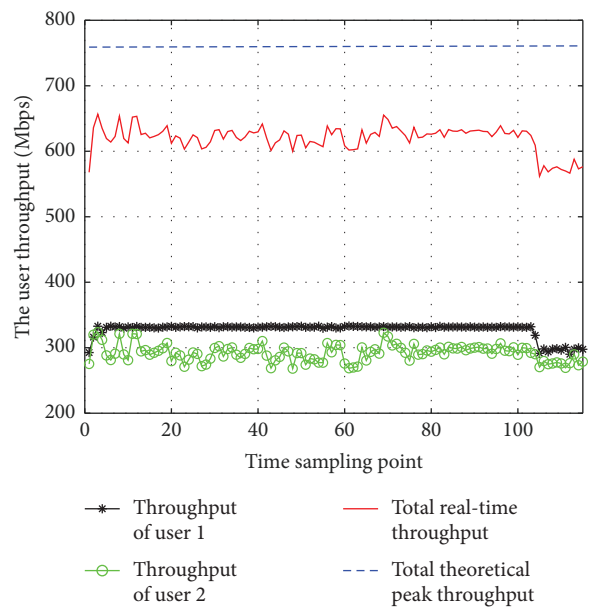


Figure 7: UL rate sampling of two users.

vBBU is fully loaded, the CPU resource occupied by the other components is higher than that when the vBBU is not loaded, owing to the increased resource occupied by vBBU containers and control plane components.

Figure 6 shows the sampling curve and probability distribution of the uplink rate of a single user. We can see the values in rate are mainly among 620–660 Mbps, and the average rate is about 624.54 Mbps. In addition, the theoretical peak rate is 760 Mbps based on our configurations, and the actual maximum peak rate during the testing is 684.95 Mbps, which can be 90% of the theoretical peak rate. The theoretical peak value formulation is calculated as follows [22]:

$$\text{data rate} = 10^{-6} * \sum_{j=1}^{J} \left\{ v_{\text{Layers}}^{(j)} * Q_m^{(j)} * R_{\max} * \frac{N_{PRB}^{BW(j),\mu} * 12}{T_s^{\mu}} * \left(1 - OH^{(j)}\right) \right\}. \tag{14}$$

The sampling and probability distribution of the uplink rates with the two users are depicted in Figures 7 and 8, respectively. The peak rate of the first user is 333.19 Mbps, and the average rate is 327.61 Mbps. While the peak rate of the second user is 323.73 Mbps, and the average rate is 292.04 Mbps. The sum of peak rates for the two users is 656.92 Mbps. It can be seen that the peak rate of two users is lower than that of a single user, which is because the overhead of two users is greater than that of a single user. We note the rate of user 2 is lower than that of user 1 and the throughput of user 2 has a fluctuation from time sampling point 0 to 100 because there is more interference in the test environment for user 2.

Table 5 shows the test results of Round Trip Time (RTT) between the terminal and the vBBU. In Radio Resource Control (RRC) connected state, we use the Internet Control

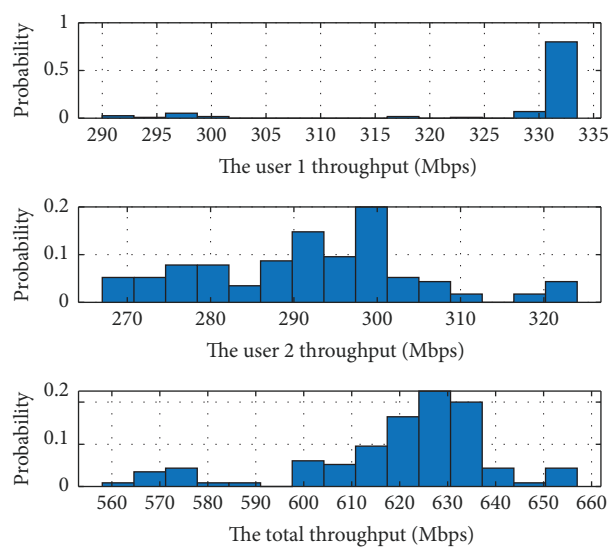


Figure 8: UL rate probability distribution of two users.

Table 5: Statistics of user plane RTT.

RTT (ms)	Ping 32 bytes	Ping 1500 bytes
Minimum value	6.677	6.355
Maximum value	10.428	13.276
Average value	7.988	7.993
The success rate of the ping packet	100%	100%

Message Protocol (ICMP) with the command PING to test the RTT. The packet size is 32 bytes and 1500 bytes, respectively, and we run 100 times continuously. The results reveal that the average delay of 1500-byte packets is slightly

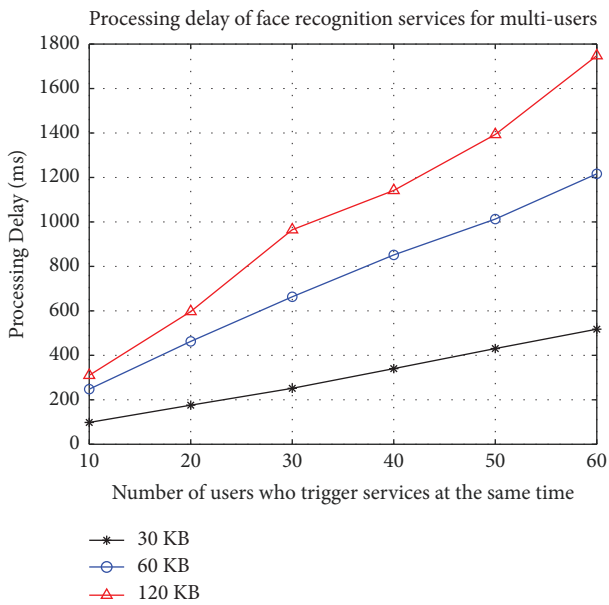


Figure 9: The processing time for intelligent application.

larger than that of 32-byte packets because the testbed processes short packets faster than a long-size packet.

5.3. Evaluation of Intelligence Case. In order to evaluate the performance of an intelligent case, a face detection application is deployed in the testbed in the form of a Pod. It is considered using OpenCV [23], and we use three images with the size of $1024 * 768$ pixels as input data from terminals for face detection. The volumes of the three images are 30 kB, 60 kB, and 120 kB. In Figure 9, the X-axis is the number of images that the testbed received from terminals. The Y-axis is the processing time for the images. The black line is for the processing time of 30 kB, the blue line is for 60 kB, and the red line is for 120 kB. Figure 9 shows that the processing time increases from 300 ms when the number of services is 10 to 1770 ms when the number of services is 60 at the volume of 120 KB. We can see that the trend of processing time is increasing, and the larger images require more processing time.

6. Conclusions

This paper presents the edge intelligence-based RAN architecture towards 6G with a lab environment testbed for EIRA provided to make it reliable, efficient, and smart. In EIRA, virtualized network functions are deployed and the intelligence modules are implanted in extensive Edge Platform and Custom Edge Platform respectively, interacting with virtualized RAN for different applications to build RAN open and programmable. Specifically, we propose an approach for allocating resources between the extensive Edge-Platform and Custom Edge Platform. Our proposed work is unique in the sense that it is one of the initial efforts to build an edge intelligent RAN architecture with a lab-scale testbed supporting all key features.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by National Key R&D Program of China under Grant 2020YFB1806700. The authors wish to thank Intel for advice on testbed design and Comba Telecom Systems Holdings Limited for experimental support.

References

[1] Z. Wang, R. Liu, Q. Liu, J. S. Thompson, and M. Kadoch, "Energy-efficient data collection and device positioning in UAV-assisted IoT," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1122–1139, 2020.

[2] L. Chen, F. R. Yu, H. Ji, B. Rong, X. Li, and V. C. M. Leung, "Green full-duplex self-backhaul and energy harvesting small cell networks with massive MIMO," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3709–3724, 2016.

[3] A. Nessa, M. Kadoch, and B. Rong, "Fountain coded cooperative communications for LTE-A connected heterogeneous M2M network," *IEEE Access*, vol. 4, pp. 5280–5292, 2016.

[4] X. H. You, C.-X. Wang, J. Huang et al., "Towards 6G wireless communication networks: vision, enabling technologies, and new paradigm shifts," *Science China Information Sciences*, vol. 64, no. 1, Article ID 110301, 2021.

[5] E. Peltonen, M. Bennis, M. Capobianco et al., " 6G white paper on edge intelligence," *Artificial Intelligence, arXiv preprint arXiv:2004.14850*, 2020.

[6] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: applications, trends, technologies, and open research problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, 2020.

[7] N. Chen, B. Rong, X. Zhang, and M. Kadoch, "Scalable and flexible massive MIMO precoding for 5G H-cran," *IEEE Wireless Communications*, vol. 24, no. 1, pp. 46–52, 2017.

[8] R. Li, Z. Zhao, X. Zhou et al., "Intelligent 5G: when cellular networks meet artificial intelligence," *IEEE Wireless Communications*, vol. 24, no. 5, pp. 175–183, 2017.

[9] C. X. Wang, M. D. Renzo, S. Stanczak, S. Wang, and E. G Larsson, "Artificial intelligence enabled wireless networking for 5G and beyond: recent advances and future challenges," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 16–23, 2020.

[10] E. Peltonen, M. Bennis, M. Capobianco et al., "6G white paper on edge intelligence," 2020, https://arxiv.org/abs/2004.14850.

[11] Y. Liu, M. Peng, G. Shou, Y. Chen, and S Chen, "Toward edge intelligence: multiaccess edge computing for 5G and Internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6722–6747, 2020.

[12] D. C. Nguyen, M. Ding, P. N. Pathirana, S. Aruna, J. Li, and D. Niyato, "6G Internet of Things: a comprehensive survey," *IEEE Internet of Things Journal*, vol. 9, 2021.

[13] Q. Liu, S. Sun, B. Rong, and M. Kadoch, "Intelligent reflective surface based 6G communications for sustainable energy

infrastructure," *IEEE Wireless Communications*, vol. 28, no. 6, pp. 49–55, 2021.

[14] T. Li, W. Liu, Z. Zeng, and N. X. Neal, "DRLR: a deep reinforcement learning based recruitment scheme for massive data collections in 6G-based IoT networks," *IEEE Internet of Things Journal*, vol. 9, 2021.

[15] C. She, R. Dong, Z. Gu et al., "Deep learning for ultra-reliable and low-latency communications in 6G networks," *IEEE Network*, vol. 34, no. 5, pp. 219–225, 2020.

[16] S. B. Prathiba, G. Raja, S. Anbalagan, and K. Dev, "Federated learning empowered computation offloading and resource management in 6G-V2X," *IEEE Transactions on Network Science and Engineering*, vol. 9, 2021.

[17] S. Han, T. Xie, I. Chih-Lin et al., "Artificial-intelligence-enabled air interface for 6G: solutions, challenges, and standardization impacts," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 73–79, 2020.

[18] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao, and K. Wu, "Artificial-intelligence-enabled intelligent 6G networks," *IEEE Network*, vol. 34, no. 6, pp. 272–280, 2020.

[19] H. Xu, J. Wu, J. Li, and X Lin, "Deep-reinforcement-learning-based cybertwin architecture for 6G IIoT: an integrated design of control, communication, and computing," *IEEE Internet of Things Journal*, vol. 8, no. 22, Article ID 16337, 2021.

[20] H. Ben-Ammar, Y. Hadjadj-Aoul, G. Rubino, and A.-C. Soraya, "A versatile Markov chain model for the performance analysis of CCN caching systems," in *Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, Abu Dhabi, United Arab Emirates, December 2018.

[21] O-Ran, "Cloud Architecture and deployment scenarios for O-RAN virtualized," 2021, https://www.o-ran.org/specifications.

[22] 3GPP TS, *User Equipment (UE) Radio Access Capabilities*, https://www.3gpp.org/ftp/Specs/archive/38_series/38.306, 2021.

[23] Willowgarage, "Open-source computer vision library (OpenCV)," 2022, http://https//www.willowgarage.%20com/pages/software/opencv.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 3898277, 13 pages
https://doi.org/10.1155/2022/3898277

Research Article

Research on Information Fusion of Computer Vision and Radar Signals in UAV Target Identification

Yuan Wei,[1,2] **Tao Hong**,[1,2,3] **and Chaoqun Fang**[1,2]

[1]*School of Electronic and Information Engineering, Beihang University, Beijing 100191, China*
[2]*Shenzhen Institute of Beihang University, Shenzhen 518063, China*
[3]*Yunnan Innovation Institute·BUAA, Kunming 650233, China*

Correspondence should be addressed to Chaoqun Fang; fangchaoqun@buaa.edu.cn

Received 18 March 2022; Revised 20 May 2022; Accepted 30 May 2022; Published 19 July 2022

Academic Editor: Bo Rong

As one of the crucial sensing methods, multisensor fusion recognition aids the Internet of Things (IoT) in connecting things through ubiquitous perceptual terminals. The small size, sluggish flying speed, low flight altitude, and low electromagnetic intensity of unmanned aerial vehicles (UAVs) have put enormous strain on air traffic management and airspace security. It is urgent to achieve effective UAV target detection. The radio monitoring method, acoustic detection scheme, computer vision, and radar signal detection are commonly used technologies in this field. The radio monitoring approach has low accuracy, the acoustic detection strategy has a limited detection range, computer vision is limited by weather conditions, and the radar signals at low altitudes are influenced by ground clutter. To address these issues, this paper proposes an information fusion strategy based on two levels of fusion: data-level fusion and decision-level fusion. In this strategy, Computer vision and radar signals complement each other to improve the detection accuracy. For each level, the method of information fusion is introduced in detail. Furthermore, the effectiveness of the method has been demonstrated by a series of comprehensive experiments. The results show that the accuracy of the fusion method is improved, and the proposed method can still work even when the single method loses function.

1. Introduction

The rapid development of the types and quantities of UAVs [1] has broadened their range of applications from military [2–4] to civilian (e.g., smart agriculture [5, 6], smart city [7], and surroundings [8]). These show that UAVs will have a significant impact on future production and lifestyle. Furthermore, certain UAVs may be illegally used which may disturb the normal airspace order [9]. These issues will put pressure on air traffic management and even threaten airspace security. Faced with these problems, it is crucial to employ adequate detection methods to accurately detect and track UAVs. UAV target detection refers to the functions of positioning, detecting, warning, or classifying through related technologies. By achieving this goal, researchers can obtain more precise information on UAVs and provide support for the following activities, such as path planning, obstacle avoidance, and equipment maintenance. Computer vision, capture remote sensing, and radar detection are the three main approaches in this field [9]. In this paper, we focus on the combined use of computer vision and radar detection.

The identification of UAVs based on computer vision mainly analyses the images captured by cameras or snapshotted from video streams. These digital images may contain different kinds of UAVs, which will be judged by the corresponding algorithms. Benefitting from easy image acquisition, simple data processing, and low equipment cost and weight, this method has attracted lots of attention and has been used commercially, such as CASIA [10]. In [11], the author proposed the CenterNet method for drone target detection, applied TensorRT to accelerate and split the network model, and proposed a method for location tracking using multiple cameras. But this method also has

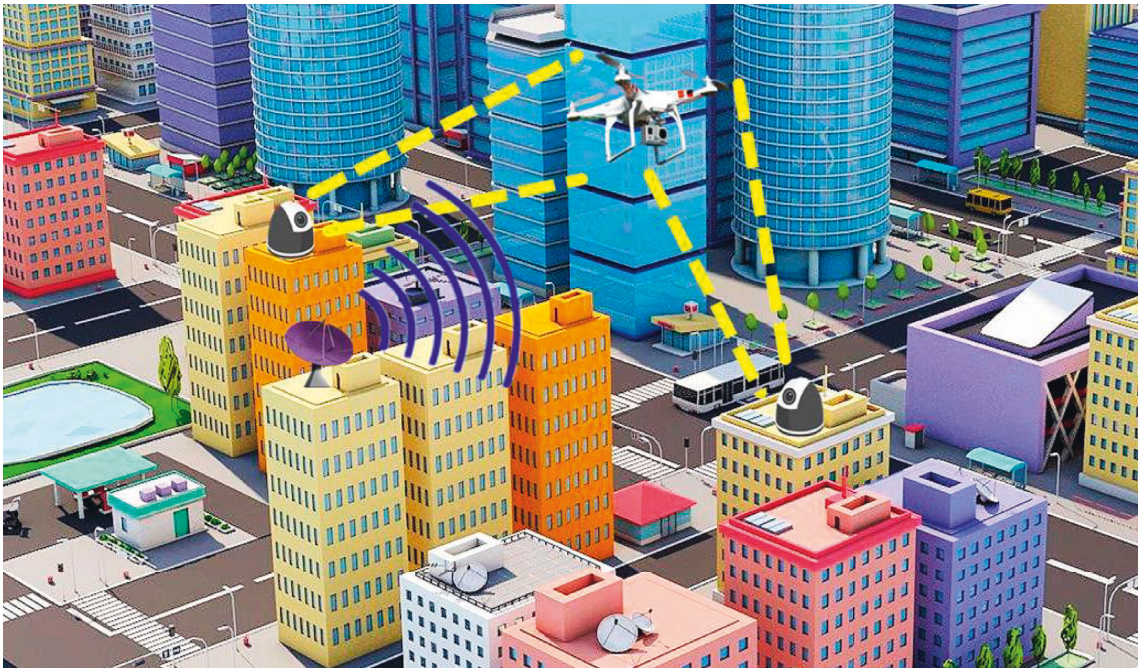


Figure 1: The scene of tracking UAVs with radars and cameras.

some limitations in this special field. For example, the method based on computer vision cannot work properly during the night.

Radar is another commonly used equipment for detecting UAVs. By processing the reflection of electromagnetic waves, the direction, distance, position, and speed of UAVs can be determined. Electromagnetic waves will never be affected by light and can penetrate clouds [12]. Therefore, radar can easily detect high-altitude targets. All-round airspace surveillance has been provided by radars with the continuous development of radar technology and application of passive radar [13]. In [14], micro-Doppler characteristics are used to identify the UAVs and the cepstrum method is used to extract the number and speed information of the UAV rotor. However, as an application of electromagnetic waves, radar signals are easily interfered by other electromagnetic waves [15]. At the same time, the radar may be unable to detect low-altitude targets. As a result, the radar approach is not omnipotent. There will be a detecting-blind area in some cases.

To overcome the shortcomings of computer vision and radar signals in detection, we propose a set of information fusion methods based on combining computer vision and radar signals to overcome the limitation of using a single method and maximizing the detectable range of the UAVs. In the process of information fusion, we separately fuse the two levels of the information, namely data and decision. By implementing the fusion of the two levels, we examined the efficiency of the information fusion methods and compared the improvement of detection accuracy with a single method.

The main contributions of this paper are as follows:

(1) According to the two levels of information fusion, information fusion models suitable for computer vision and radar signals are given, respectively. For each model, the modules in it are proposed in detail to prove its feasibility in theory. The methods used in the model can be replaced to make the model extensible.

(2) For each model, an information fusion implementation method using specific methods is designed. In order to prove the effectiveness of this method, comprehensive experiments were performed to compare the detection accuracy before and after fusion. The experimental results showed that the two specific implementation methods are feasible and effective.

The arrangement of the remaining sections in this paper is as follows: Section 2 discusses the related work of computer vision and radars in UAV detection, and discusses the existing information fusion methods. Section 3 introduces the proposed method and gives a specific implementation method of the data fusion model and the decision fusion model. Section 4 conducts comprehensive experiments, implements two methods, and analyses them separately. Finally, the conclusion of this paper is given in Section 5.

Figure 1 depicts a certain scene of identifying UAVs. In the city, radars and cameras on rooftops maintain monitoring the airspace. When an alien UAV enters the monitoring area, it will be detected by both cameras and radars. Then, the system can identify the UAV by fusing data collected by cameras and radars. Because both cameras and radars are used, UAV identification is more efficient and accurate than either method alone.

2. Related Work

2.1. Computer Vision. Artificial intelligence (AI), particularly deep learning, has offered significant technical assistance for computer vision. To achieve the objective of detection, cameras are used to acquire images, and the corresponding algorithms are used to determine whether there are specific targets. This technology has been used widely in facial recognition, medical diagnosis, UAV detection, and other sectors, with positive results. The following methods are commonly used in UAV detection.

2.1.1. Convolutional Neural Network (CNN). CNN is widely used in UAV detection because convolutional kernels can effectively extract target features from images. In order to solve the detection of UAVs in a video sequence, Aker and Kalkan [16] proposed an end-to-end model based on CNN. Simultaneously, they also proposed an algorithm based on a background subtracted to solve the problem of insufficient data in the training model. In order to identify UAVs accurately and determine their types and flight modes, Allahham et al. [17] proposed a new detection method. This method used multichannel-dimensional CNN and achieved good results in the DroneRF dataset. Reducing background interference can improve the detection accuracy; therefore, Zhang et al. [18] used Mask R-CNN to eliminate the invalid area in the UAV detection process and used the attention mechanism to detect the targets.

2.1.2. YOLO (You Only Look Once). Unlike other methods, the YOLO method only requires one recognition procedure. Hu et al. [19] improved YOLO v3 to make it more suitable for detecting small targets, for example, UAVs. Owing to the unique advantages of YOLO, how to achieve real-time detection has become the focus of UAV detection [20, 21].

Other computer vision methods are also being applied in this field, such as boosting [22], fuzzy clustering [23], and multiple neural networks (MNNs) [24]. These methods have unique characteristics and play an important role in detecting UAVs.

2.2. Radar Methods. Radar plays a significant role in the detection of UAVs due to its intrinsic properties. From military to civilian applications, this technology began to evolve toward ease of use and low cost. The radar technology used in this field mainly includes digital array radar [14], multi-input-multioutput (MIMO) radar [25], continuous wave radar [26], synthetic aperture radar (SAR), and inverse synthetic aperture radar (ISAR).

SAR can penetrate clouds, smoke, and fog, and produce high-resolution images [27–29], which can reduce the impact of weather conditions in detection. In order to detect suspicious UAVs and reduce the cost, Park et al. [30] designed a set of systems based on low-cost SAR. The systems had the characteristics of autonomy and mobility, and performed well in the tests.

During the evolution of SAR, ISAR occurs and plays an important role in the detection of long-range targets, because it can provide high-resolution imaging. Pieraccini et al. [31] examined the radar cross section (RCS) of tiny UAVs and employed ISAR for 2-dimensional (2D) and 3-dimensional (3D) imaging, and the experimental results were good. Authors in [32] proposed a method of introducing Bayesian statistics into ISAR to solve the problem of a small RCS. The efficiency of this strategy was confirmed by simulation, which used posterior probability density to determine the imaging results.

2.3. Information Fusion Methods. Some studies examined multiple types of fusion procedures in order to enhance accuracy. In the research of Kim et al. [33], new images are synthesized for UAV detection; this method combined the time-domain and frequency-domain information of the micro-Doppler signature (MDS), and these images were the data set of classification. Training in the CNN could improve the accuracy by more than 5%, which proved the effectiveness of this merging method. Joshi et al. [34] reviewed 112 articles, all of them fused optical and radar remote sensing data, which is of great significance to the research of this paper. These studies have been applied in the field of land, and many studies showed that the effect of using the fusion method was better than using a single method. At the same time, for the traditional classification algorithms, the most commonly used method was to fuse before classification, with pixels as input. This review discussed the related articles from multiple perspectives and fully explained the application status of the fusion method.

3. Methods to Identify UAVs by means of Information Fusion

The redundancy design of the system helps to improve system performance and robustness. This work introduces the information fusion method based on computer vision and radar signals to improve the detectable range of UAVs. A two-level information fusion system including data fusion and decision fusion is designed in this work. On one hand, the UAV's position namely the coordinate is the primary target in the data fusion part. On the other hand, decision fusion aims to fuse the unique feature of the UAV. Figure 2 presents the UAV identification system based on information fusion.

3.1. Data Fusion

3.1.1. Digital Data Processing. When a 3D object is photographed by an optical camera, its image will be turned into a 2D image, but its relative position in the picture remains unchanged. An image is composed of many pixel points. Pixel coordinates can be used to express an object's position while determining its location. However, it is necessary to select an appropriate coordinate system. The image coordinate system, camera coordinate system, and world coordinate system are the three types of coordinate systems used

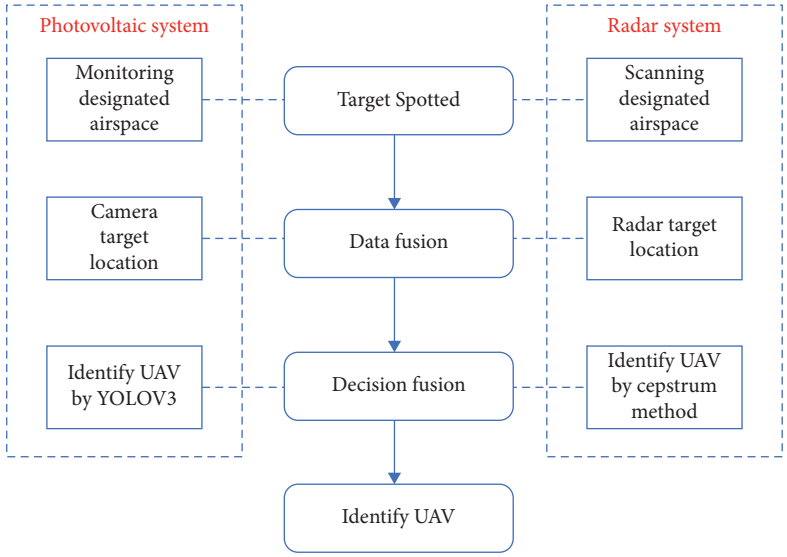


Figure 2: UAV identification system based on information fusion.

in the photographs. Among them, the image coordinate system is a 2D coordinate system, and the other two coordinate systems are 3D coordinate systems. In this paper, the camera captures images of the UAVs directly, so the camera coordinate system is the best option for calibrating the UAV's position. In the processing of digital data, the position of an UAV is marked with 2D coordinates.

3.1.2. Radar Data Processing. Broadband radar can identify the target's direction and distance via echo. Through the micro-Doppler effect of the UAV, the position of the target can be obtained, and then, according to the orientation, other characteristics such as the height of the target can also be obtained. Therefore, compared with the camera, the radar can achieve height measurement; therefore, the target's position obtained in this way will be 3D information.

3.1.3. Data Fusion Method. In order to better detect the same UAV targets, the combination of optical images and radar positioning can be used. If the coordinates of the two UAVs are the same after being transformed into the same coordinate system, it can be determined that this is the only UAV, so as to realize the detection of UAVs at the data level. It is actually a perspective projection problem to transform the 3D coordinates of the object obtained by the radar into the 2D coordinates of the images taken by the camera. This problem can be solved in three steps as follows:

(1) Determine the Projection Plane. The 3D coordinates of the object obtained by the radar are based on the radar being the origin of the reference system, so the observation point of the optical camera often does not coincide with the position of the origin of the coordinates in the radar. As shown in Figure 3, determine the reference point A (a, b, c), the observer coordinate S (x_0, y_0, z_0), take any reference direction point B (d, e, f), and then, set a sight distance λ to determine the projection plane HPFK. In this case, a projection plane equation can be determined as follows:

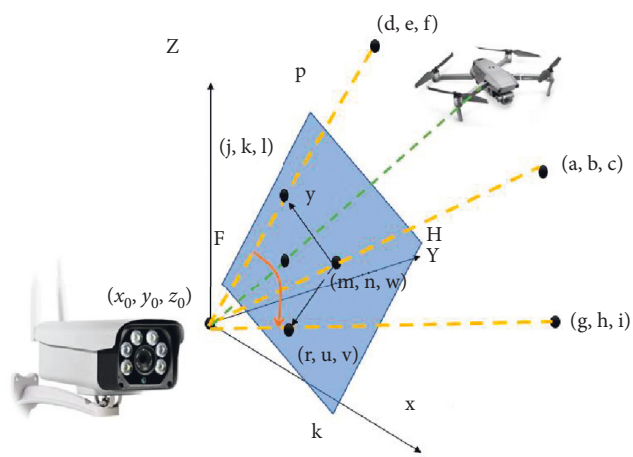


Figure 3: The coordinate conversion of UAV.

$$A'x + B'y + C'z + D' = 0. \tag{1}$$

A', B', and C' are calculated by

$$(A', B, C') = (a - x_0, b - y_0, c - z_0). \tag{2}$$

D' is determined by

$$\lambda = \left| \frac{A'x_0 + B'y_0 + C'z_0 + D'}{\sqrt{A'^2 + B'^2 + C'^2}} \right| = 0. \tag{3}$$

(2) Determine the Projection Plane Coordinate System. After the reference point A is given, the line between the observer and the reference point is the normal vector of the projection plane. The equation of this straight line I is as follows:

$$\frac{X - x_0}{A} = \frac{Y - y_0}{B} = \frac{c - z_0}{C} = k_1. \tag{4}$$

The straight-line equation and the projected plane equation can be combined to find the intersection point D (m, n, w); this point is set as the origin of the projection plane. In the same way, the intersection point E (j, k, l) of the reference direction point with the observer line and the projection plane can be obtained. These two 3D points are two relative 2D coordinate points on the projection plane, and the vector $(j - m, k - n, l - w)$ is set to the positive y-axis direction on the projection plane.

Using the straight line I as the axis, rotate the reference direction point around this axis (viewed by the position of the observer) clockwise by 90°, so that there will be no uncertainty (i.e., no 2 points will be generated), and a new coordinate point C will be obtained. Similarly, connect this point with the observer and find the intersection point F (r, u, v) of the connection line with the projection plane so the vector $(r - m, u - n, v - w)$ is in the positive x-axis direction on the projection plane.

(3) Conversion of 3D Coordinates to 2D Coordinates. When determining the coordinates of the target point, the observer and the target are directly connected, and the intersection point with the projection plane is the projection point of the target on the 2D plane. The origin, the x-axis positive direction, and the y-axis positive direction have been determined on the projection plane before, so that the projection coordinates of the target point on the 2D plane can be obtained.

Figure 4 shows how we convert the coordinates of the drone in the picture to three-dimensional coordinates of the real world. Starting with radar coordinates $(0, 0, 0)$, the coordinate of the camera is (x_1, y_1, z_1). The maximum horizontal and vertical viewing angles of the camera are α and β, respectively. The length and height of the photos are l_1, l_2. Set up a cartesian coordinate system to locate the coordinates of the points on the photo. The coordinates of the upper left corner are $(0, 0)$ and the coordinates of the lower right corner are (l_1, l_2). The coordinates of the UAV in the picture are (m_1, n_1). The coordinates of the UAV in the cartesian coordinate system with the centre of the picture as the origin are $(m_1 - l_1/2, l_2/2 - n_1)$. α_1 and β_1 represent the angle at which the drone is deflected relative to the direction of the camera.

$$\alpha_1 = \arctan\ \theta \frac{m_1 - (l_1/2)}{(l_1/2)/\tan\ \alpha},$$
$$\beta_1 = \arctan\ \theta \frac{(l_2/2) - n_1}{(l_2/2)/\tan\ \beta}. \tag{5}$$

The direction vector of camera erection is $(\sin\ \theta\ \cos\ \varphi, \sin\theta \sin\varphi, \cos\theta)$. Then, the direction vector of the UAV relative to the camera is

$$v = ((\sin\ \theta - \beta_1)\cos(\varphi - \alpha_1),\ \sin(\varphi - \alpha_1)\sin\ \varphi,\ \cos(\theta - \beta_1)). \tag{6}$$

The connection between the camera and UAV can be expressed as $(x_1, y_1, z_1) + k * v$, where k is a parameter.

Similarly, another parametric equation of the connection between the camera and UAV can be worked out, and the intersection point of the two straight lines is the UAV coordinate.

3.2. Decision Fusion

3.2.1. Digital Decision Processing. After YOLO was put forward [35, 36], it has been widely used because it can detect the target in real time. When it came to YOLOv3 [37], the accuracy of detection for small targets has been improved significantly. For UAV detection, the YOLOv3 algorithm is more suitable. First of all, real-time detection can meet the detection requirements of UAVs in fast flight: different from static objects or slow-moving objects (such as pedestrians and ships), the requirements for time are not very strict, but the UAV's speed may be faster; hence, the requirements for time accuracy will naturally improve. Secondly, UAVs may have a high flying altitude, which may be the very small targets in the camera. The feature extraction effect using the CNN and other methods may not be obvious. However, YOLOv3 rebuilt the neural network structure and reconstructed the loss function, focusing more on the detection of small targets, which is also suitable for UAV detection. Based on the abovementioned reasons, YOLOv3 is used to process the optical images, and the detection results are obtained before the decision fusion. Figure 5 depicts the workflow of YOLOv3.

Simple Online and Real-time tracking (SORT) is a simple and efficient tracking method based on the Kalman filter and Hungarian matching algorithm. The main shortcoming of the SORT algorithm is that the association metric it uses is valid only when the uncertainty of state estimation is low. Otherwise, tracking will fail when the target is covered.

On the basis of the SORT algorithm, the DeepSORT algorithm combines the motion information and the appearance information of the target as the association metric. In this way, the DeepSORT algorithm can track the occluded target.

The DeepSORT algorithm uses the results of the detector to initialize the tracker, and sets a counter for each tracker. The counter is accumulated after Kalman filtering. When the prediction result matches the detection result, the counter is set to zero. If no appropriate detection result is matched within a period of time, the tracker will be deleted.

The DeepSORT algorithm combines motion information and appearance information to match the prediction box and tracking box by using the Hungarian algorithm. For motion information, the algorithm uses Mahalanobis distance to describe the connection degree of prediction results and the detection results. When the target movement information uncertainty is low, the Mahalanobis distance is a suitable correlation factor. However, when the target is blocked or the lens view is shaken, only the Mahalanobis distance correlation will lead to a target identity switch, so appearance information should be considered. The Mahalanobis distance can provide reliable target location information in a short-term prediction, and the cosine similarity of the appearance feature can be used to recover target

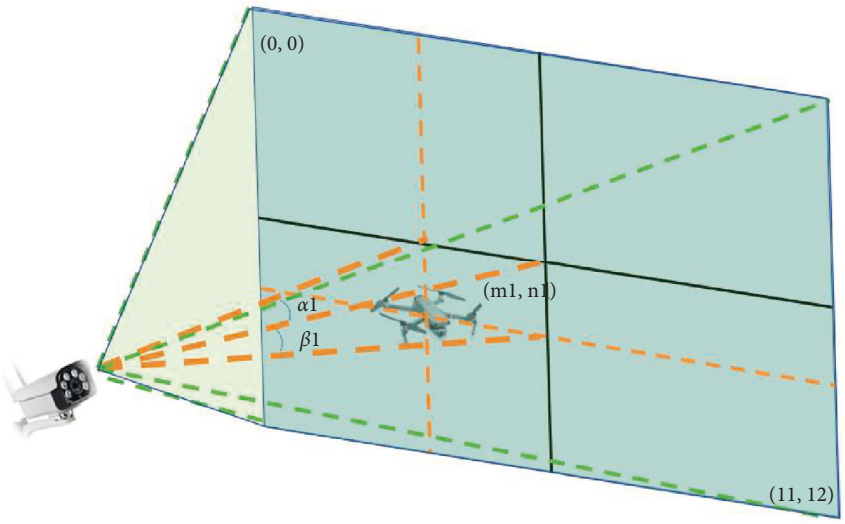


Figure 4: Locating the UAV using the photovoltaic system.

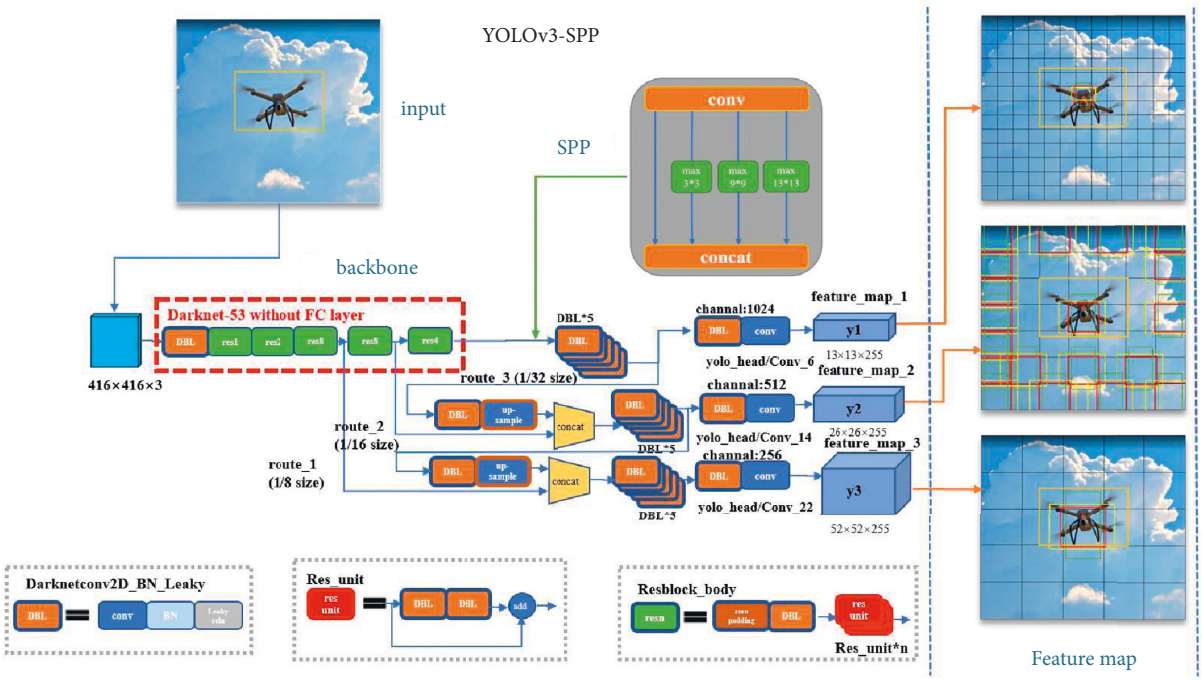


Figure 5: Workflow of the YOLOv3 algorithm.

identification (ID) when the target reappears. Using linear weighting, the two methods complement each other. Figure 6 depicts the workflow of the DeepSORT algorithm.

3.2.2. Radar Decision Processing. Chen [38] found that the micro-Doppler phenomenon caused by micromotion can also be observed in the microwave radar system, proposed the mathematical expression of the micro-Doppler effect, and believed that it has a potential application value in target feature extraction. In addition to its own main direction movement, a target may also have other mechanical movements. These additional mechanical movements will cause the frequency modulation of radar echo, resulting in the micro-Doppler effect.

The influence of different kinds of objects' mechanical motion on radar echo is quite different (such as the vibration of birds' wings and the vibration of UAVs' rotors). Therefore, the use of micro-Doppler features has a good effect in distinguishing different types of objects. At the same time,

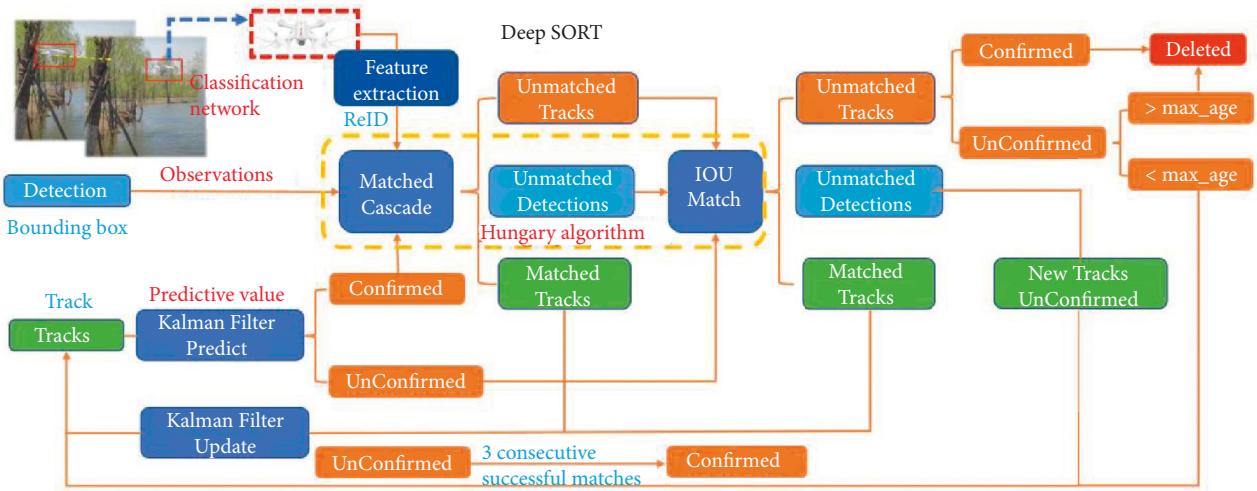


Figure 6: Workflow of the DeepSORT algorithm.

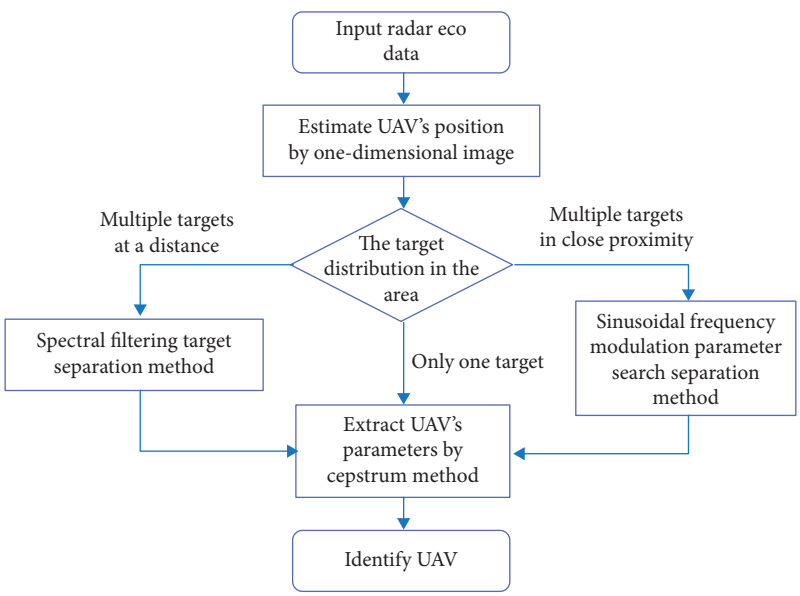


Figure 7: Identification of UAV by the radar system.

the micro-Doppler effect of UAVs will also be greatly different due to the speed, number, length, and other factors of the rotors; therefore, it also plays an important role in identifying different types of UAVs.

For micro-Doppler feature extraction, the Fourier method and time-frequency analysis method are mainly used. The frequency information related to the time cannot be obtained by the Fourier method, so it is not the mainstream method of micro-Doppler feature extraction. For time-frequency analysis, the main methods include short-time Fourier transform [39], generalized S-transform [40], and Gabor transform [41–43]. Gabor transform is a short-time Fourier transform with a Gaussian window. It has no cross-term and faster operation speed also has obvious time-frequency characteristics, so it is suitable for extracting micro-Doppler features of UAVs. This is also the reason why this method is chosen in this paper.

We initially estimate the position of the UAV using a one-dimensional image after receiving the radar echo of the UAV target. Then, based on the target's distribution, we choose a suitable approach for separating the UAV target. Finally, to fulfill the goal of UAV radar system identification, we employ the cepstrum approach to extract the properties of the UAV Figure 1 shows the workflow of identifying UAVs by radar system.

3.2.3. Decision Fusion Methods. When multiple methods are used to detect the targets, all the detection results can be fused in the decision-making stage, which is an effective method to transform the weak classifiers into the strong ones. This is also an important idea of the boosting algorithm [44].

X for an unknown target of a certain category, T_i for the ith detection method, $P_i(X)$ for the probability vector output

by T_i, $U_i(X)$ for the recall rate of T_i, and $V_i(X)$ for the accuracy rate of T_i. $P_i(X) = \{P_{i,1}(X), \ldots, P_{i,j}(X), \ldots, P_{i,M}(X)\}$, $U_i(X) = \{U_{i,1}(X), \cdots, U_{i,j}(X), \cdots, U_{i,M}(X)\}$, and $V_i(X) = \{V_{i,1}(X), \ldots, V_{i,j}(X), \ldots, V_{i,M}(X)\}$, where $i = 1, 2, \ldots, N$, N represents the number of detection methods to be fused; $j = 1, 2, \ldots, M$, M represents the number of target categories. The following rules were used for decision fusion.

Maximum rule:

$$F_{\max} = \{j \mid \quad F_j = \max\{P_{i,j}\},\ i = 1, \ldots, N;\ j = 1, \ldots, M;\}. \tag{7}$$

Minimum rule:

$$F_{\min} = \{j \mid \quad F_j = \min\{P_{i,j}\},\ i = 1, \ldots, N;\ j = 1, \ldots, M;\}. \tag{8}$$

Mean rule:

$$F_{\text{mean}} = \left\{ j \mid \quad F_j = \frac{\sum_{i=1}^{N} P_{i,j}}{N},\ i = 1, \ldots, N;\ j = 1, \ldots, M; \right\}. \tag{9}$$

Product rule:

$$F_{\text{prod}} = \left\{ j \mid \quad F_j = \prod_{i=1}^{N} P_{i,j},\ i = 1, \ldots, N;\ j = 1, \ldots, M; \right\}. \tag{10}$$

Recall rule:

$$F_{rec} = \left\{ j \mid \quad F_j = \sum_{i=1}^{N} P_i(X) \cdot U_i(X),\ i = 1, \ldots, N;\ j = 1, \ldots, M; \right\}. \tag{11}$$

Accuracy rule:

$$F_{acc} = \left\{ j \mid \quad F_j = \sum_{i=1}^{N} P_i(X) \cdot V_i(X),\ i = 1, \ldots, N;\ j = 1, \ldots, M; \right\}. \tag{12}$$

Figure 2 depicts the decision fusion model. The model shows that the two approaches are independent before a single detection result is obtained. Multiple detection methods will not interfere with each other because of this independent processing, and using both picture and radar forms can ensure the most data diversity. Because the detection methods utilized can be altered, the model has scalability. The data properties reflect this decision-level fusion as well. The usage of images mainly extracts features from targets such as lines and textures of objects, and judges whether the targets exist. The use of micro-Doppler signals mainly extracts time-frequency information, and analyses the change of frequency over a period of time to infer the target's status. Therefore, before the decision fusion, it can also be seen as the interaction of different features. These features with large differences can jointly complete the task of target detection.

4. Results and Discussion

The previous sections explain how the system works. In the actual work, the identification results may be affected by external conditions. Therefore, this paragraph will use the measured results to verify the accuracy of the system's identification of UAVs.

4.1. Introduction to the Experimental System. Considering the actual operation scenario of the system, the experiment was carried out outdoors. The test equipment include an antenna, vector network analyzer, high-speed camera, turntable, computer, one single rotor UAV and one quadrotor UAV, signal amplifier, and power supply. The device was connected as shown in Figure 8.

4.2. Outdoor Test. Figure 9 shows the experimental system.

We conducted the outdoor experiments in three different scenarios: one with good light and a short detecting distance, one with poor light and a short detecting distance, and one with good light and a long detecting distance. The reason why we chose these three scenarios was to check whether the information fusion method works well when the single method cannot work well in a bad situation. We know that the computer vision method fails to identify UAVs when the light is poor. As to the radar method, UAVs are small targets and their echo signals are weak. It is hard to extract useful information of targets when the targets are far enough. However, the common rotors of UAVs are composite materials and their echoes are even weaker than those made of metal. So, the radar may be unable to detect UAVs when they are far beyond the detection range. In our experiments, we choose 4 meters as the short detecting distance and 8 meters as the long detecting distance.

The outdoor experiment ran three sets of tests on single-rotor and quadrotor UAVs for every scenario. After the UAVs are launched, the turntable is activated so that the radar system and camera system can scan the UAVs. The radar system and camera system will collect data and transmit it to the computer for processing. The processed data of the images taken by the camera are shown in Figure 10.

4.3. Experimental Results and Discussion. It can be seen from the experimental results that the identification system confirms that the radar system and the camera system identify the same target by determining the target coordinates first. Then, the system will compare the identification accuracy of the two systems. Finally, the identification system will output the result of the system with higher identification accuracy in different environments, which improves the accuracy by a maximum of 9.5%. At the same time, the robustness of the identification work is guaranteed (Table 1).

The single identifying method whether the computer vision method or radar works well when the light is good enough and the detecting distance is short. However, the

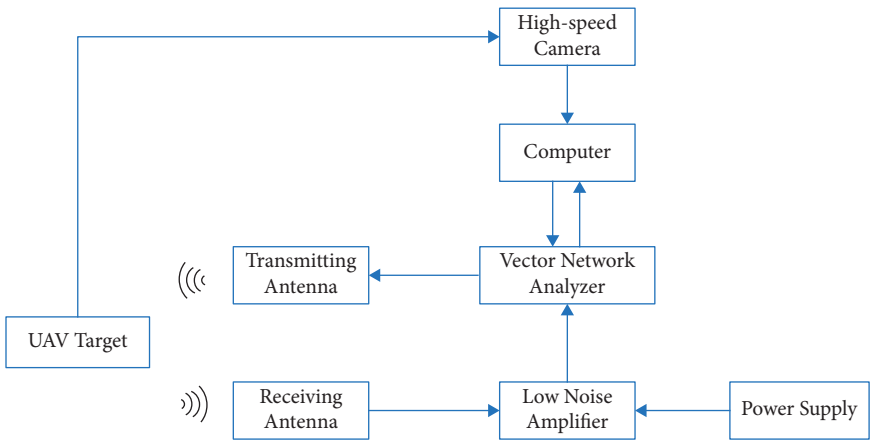


Figure 8: The experimental system.

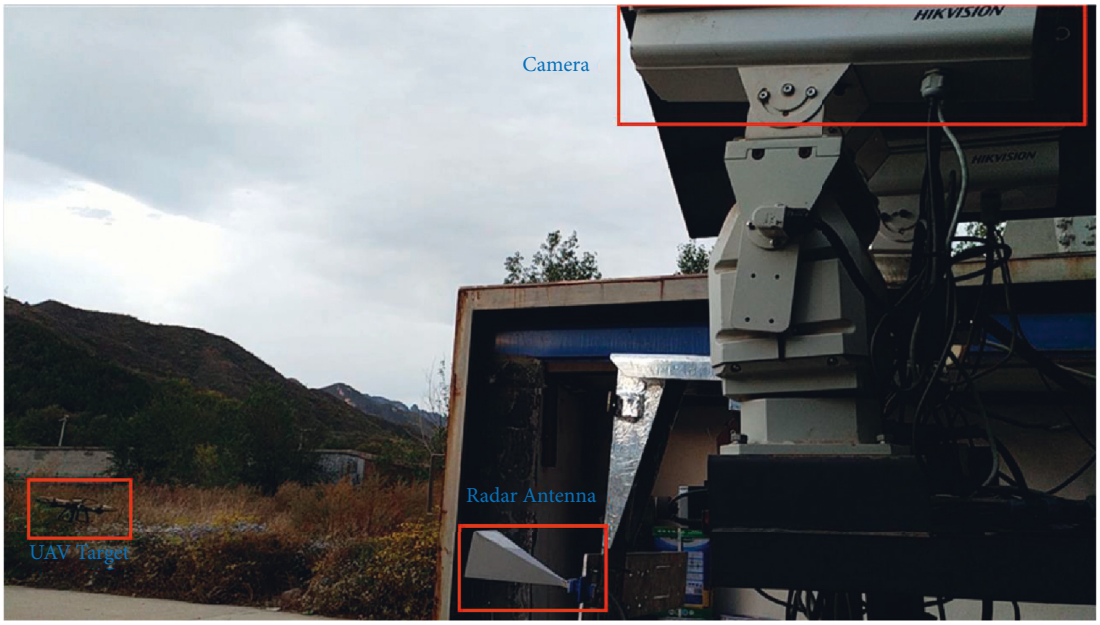


Figure 9: The experimental environment.

Figure 10: Identification of UAVs by the photovoltaic system.

Table 1: Experiment 1: identification of UAVs by the information fusion model in a good light and short distance scenario.

Group number	1		2	
UAV type	Helicopter	4-rotor drone	Helicopter	4-rotor drone
Coordinate in the image	(745, 559)	(1197, 581)	(775, 886)	(1153, 567)
Recognition accuracy by YOLOv3	0.96	0.90	0.85	0.84
Coordinate by radar	(2.5, 3.3, 2.9)	(−3.7, 3.1, 2.1)	(2.9, 3.4, 2.6)	(−3.5, 3.2, 1.9)
Transformed coordinate	(743, 562)	(1199, 577)	(775, 887)	(1150, 566)
Recognition accuracy by radar	0.91	0.86	0.93	0.92
Recognition accuracy by the proposed system	0.96	0.90	0.93	0.92
Improved accuracy	5.5%	4.7%	9.4%	9.5%

Table 2: Experiment 1: identification of UAVs by the information fusion model in a poor light and short distance scenario.

Group number	1		2	
UAV type	Helicopter	4-rotor drone	Helicopter	4-rotor drone
Coordinate in the image	Failed	(897, 521)	(675685)	Failed
Recognition accuracy by YOLOv3	Failed	0.09(almost failed)	0.05(almost failed)	Failed
Coordinate by radar	(2.2, 3.1, 2.8)	(−3.9, 2.9, 2.1)	(2.5, 3.1, 2.7)	(−3.8, 2.6, 1.8)
Transformed coordinate	(622, 419)	(899, 527)	(677, 687)	(915, 515)
Recognition accuracy by radar	0.95	0.90	0.87	0.82
Recognition accuracy by the proposed system	0.95	0.90	0.87	0.82

Table 3: Experiment 1: identification of UAVs by the information fusion model in a good light and long-distance scenario.

Group number	1		2	
UAV type	Helicopter	4-rotor drone	Helicopter	4-rotor drone
Coordinate in the image	(445, 659)	(785, 672)	(445, 706)	(825, 667)
Recognition accuracy by YOLOv3	0.86	0.79	0.85	0.74
Coordinate by radar	(4.2, 3.9, 4.9)	Failed	Failed	Failed
Transformed coordinate	(433, 672)	Failed	Failed	Failed
Recognition accuracy by radar	0.11(almost failed)	Failed	Failed	Failed
Recognition accuracy by the proposed system	0.86	0.79	0.85	0.74

detecting accuracy of the single method will never be higher than the proposed information fusion method because the maximum rule is chosen for decision-making in the proposed method. The detecting performance is significant and the robustness of the identification work will be guaranteed in spite of a higher cost. This is more practical.

In this scenario, images of UAVs from the camera are clear enough, and it is easy to identify UAVs and track their location in the images. The detecting distance is also critical for radar detecting of UAV targets. Because the UAVs are made of composite materials to lose weight, the radar echo is much weaker than those made of metal. The micro-Doppler signature is produced by the rotors on the UAVs. Usually, the rotors are tiny compared with the whole UAV body. Under current technology conditions, a short detecting distance is necessary to ensure the extraction of the micro-Doppler signature. Or else, radars cannot receive a strong enough echo of UAVs, and the extraction of the micro-Doppler signature will never be accomplished.

In Table 2, it is obvious that the recognition function of the computer vision method is out of operation in a poor light scenario. The cameras fail to capture images in dark environments. Therefore, the recognition could never be carried out. The computer vision method fails to detect UAVs in this kind of scenario. It is dangerous for airspace surveillance. However, the radar can still work well in dark environments. The proposed information fusion method still works well because it is composed of the radar system. Additional equipment contribute to more robust performance.

Table 3 shows the weakness of the recognition performance of radar when the UAVs are far from radars. The radar echo is weak when the distance is long. The longer the distance, the weaker the echo, especially when the UAVs are made of composite materials such as plastic and carbon fibre. However, the computer vision method can still work in this scenario, though the images of UAVs are smaller in the picture captured by cameras; the robustness of the identification work is guaranteed as well.

From experiments 2 and 3, we can conclude that the proposed method can always guarantee the detecting function no matter which part loses function. It is more clear from Figure 11. We equipped additional devices to obtain a more robust system function regardless of extra costs because, in airspace surveillance, the detecting performance is more critical.

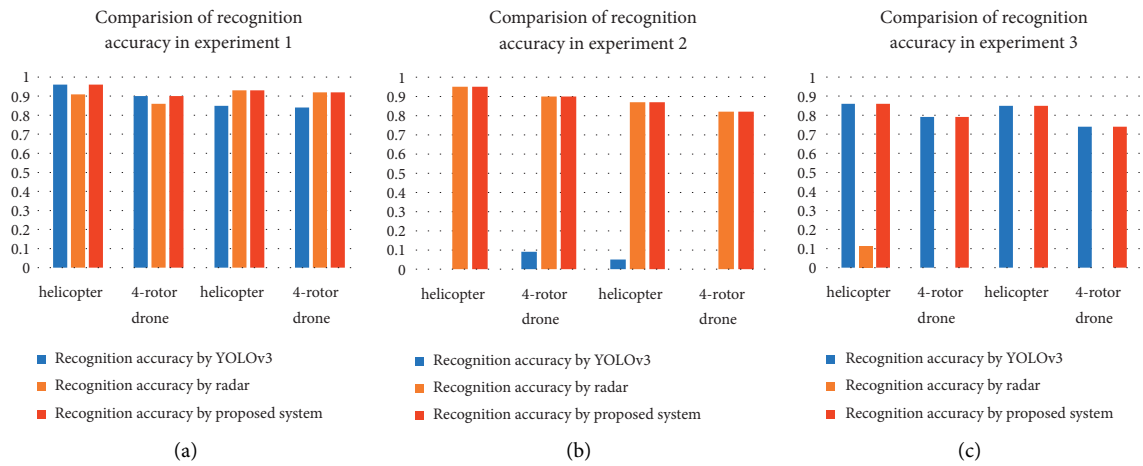


Figure 11: Comparison of recognition accuracy in 3 experiments. (a) Experiment 1. (b) Experiment 2. (c) Experiment 3.

5. Conclusions

In this work, we presented a UAV target identification method based on information fusion of computer vision and radar signals. The system uses coordinates to confirm that the radar system and camera system are identifying the same target. Then, the system will compare the identification results of both single systems to give the final identification result. The comprehensive experiments verified that the system can identify a single-rotor UAV and quad-rotor UAV. And it is superior than the single method. It is a worthy choice to obtain a more robust system function regardless of extra costs.

In the future, we will replace YOLOv3 with the latest YOLOv5 which will provide better performance. After some improvements to YOLOv5, we will also try to construct a set of UAV optical image real-time detecting systems with a hardware platform. Due to a shortage of time, the UAV data set is insufficient. Situations in the real world are more complicated. Therefore, more UAVs, seasonal elements, and targets similar to UAVs (birds and kites) can all be added to continue to improve UAV data sets. More decision-making algorithms are also in the plan.

Data Availability

The data used to support the findings of this study are included within the article.

Disclosure

A preprint has previously been published [45].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work by Chaoqun Fang was supported by the Central Guidance on Local Science and Technology Development Special Fund of Shenzhen City under Project no. 2021Szvup079. The work by Tao Hong was supported by the National Natural Science Foundation of China under Grant no. 61827901.

References

[1] C. F. Liew, D. DeLatte, N. Takeishi, and T. Yairi, "Recent developments in aerial robotics: A survey and prototypes overview," 2017, https://arxiv.org/abs/1711.10085.

[2] M. A. Ma'sum, M. K. Arrofi, G. Jati et al., "Simulation of intelligent unmanned aerial vehicle (UAV) for military surveillance," in *Proceedings of the International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 161–166, Sanur Bali, Indonesia, March 2013.

[3] V. Roberge, M. Tarbouchi, and G. Labonte, "Fast genetic algorithm path planner for fixed-wing military UAV using GPU," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 5, pp. 2105–2117, 2018.

[4] D. Orfanus, E. P. de Freitas, and F. Eliassen, "Self-organization as a supporting paradigm for military UAV relay networks," *IEEE Communications Letters*, vol. 20, no. 4, pp. 804–807, 2016.

[5] W. Andrew, C. Greatwood, and T. Burghardt, "Aerial animal biometrics: individual friesian cattle recovery and visual identification via an autonomous UAV with onboard deep inference," *CoRR*, vol. 1907, Article ID 05310, 2019.

[6] I. Yano, W. Santiago, J. Alves, L. Mota, and B. Teruel, "Choosing classifier for weed identification in sugarcane fields through images taken by UAV," *Bulgarian Journal of Agricultural Science*, vol. 23, pp. 491–497, 06 2017.

[7] H. Menouar, I. Guvenc, K. Akkaya, A. S. Uluagac, A. Kadri, and A. Tuncer, "UAV-enabled intelligent transportation systems for the smart city: applications and challenges," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 22–28, 2017.

[8] J. Dandois, M. Olano, and E. Ellis, "Optimal altitude, overlap, and weather conditions for computer vision UAV estimates of forest structure," *Remote Sensing*, vol. 7, no. 10, pp. 13895–13920, 2015.

[9] Y. Wei, T. Hong, and M. Kadoch, "Improved kalman filter variants for UAV tracking with radar motion models," *Electronics*, vol. 9, no. 5, p. 768, 2020.

[10] "Casia-unlocking Your Drones," 2020, https://www.irisonboard.com/casia/.

[11] L. Tao, T. Hong, and Y. Guo, "Drone identification based on CenterNet-TensorRT," *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, vol. 2020, Article ID 9379645, 2020.

[12] "Advantages and disadvantages of radar," 2011, http://physicsa5.pbworks.com/w/page/38521145/Advantages%20and%20Disadvantages%20of%20Radar.

[13] M. Edrich and A. Schroeder, "Multiband multistatic passive radar system for airspace surveillance: a step towards mature pcl implementations," in *Proceedings of the International Conference on Radar*, pp. 218–223, Adelaide, SA, Australia, November 2013.

[14] J. Zhao, X. Fu, Z. Yang, and F. Xu, "Radar-assisted UAV detection and identification based on 5G in the Internet of things," *Wireless Communications and Mobile Computing*, vol. 2019, pp. 1–12, Article ID 2850263, 2019.

[15] A. Aldowesh, T. Alnuaim, and A. Alzogaiby, "Slow-moving micro UAV detection with a small scale digital array radar," in *Proceedings of the IEEE Radar Conference (RadarConf)*, pp. 1–5, Boston, MA, USA, Aprile 2019.

[16] C. Aker and S. Kalkan, "Using deep networks for drone detection," in *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, Lecce, Italy, October 2017.

[17] M. S. Allahham, T. Khattab, and A. Mohamed, "Deep learning for rfbased drone detection and identification: a multi-channel 1-d convolutional neural networks approach," in *Proceedings of the IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pp. 112–117, Doha, Qatar, May 2020.

[18] J. Zhang, Q. Zhang, and C. Shi, "An unmanned aerial vehicle detection algorithm based on semantic segmentation and visual attention mechanism," in *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence CSAI*, vol. 18, May 2018, Article ID 309313.

[19] Y. Hu, X. Wu, G. Zheng, and X. Liu, "Object detection of UAV for antiUAV based on improved yolo v3," in *Proceedings of the Chinese Control Conference (CCC)*, pp. 8386–8390, Guangzhou, China, October 2019.

[20] S. Hassan, T. Rahim, and S. Shin, *Real time UAV Detection Based on Deep Learning Network*, in *Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC*, pp. 630–632, Jeju, Korea (South), December 2019.

[21] N. Tijtgat, W. Van Ranst, T. Goedeme, B. Volckaert, and F. De Turck, "Embedded real-time object detection for a UAV warning system," in *Proceedings of the The IEEE International Conference on Computer Vision (ICCV) Workshops*, Venice, Italy, January 2017.

[22] S. Bjrklund, "Target detection and classification of small drones by boosting on radar micro-Doppler," in *Proceedings of the 15th European Radar Conference (EuRAD)*, pp. 182–185, Madrid, Spain, November 2018.

[23] M. M. Ferdaus, S. G. Anavatti, M. A. Garratt, and M. Pratama, "Fuzzy clustering based nonlinear system identification and controller development of pixhawk based quadcopter," in *Proceedings of the Ninth International Conference on Advanced Computational Intelligence (ICACI)*, pp. 223–230, Doha, Qatar, July 2017.

[24] V. Anavatti and S. Anavatti, "Real-time system identification of unmanned aerial vehicles: a multi-network approach," *Journal of Computers*, vol. 3, no. 7, p. 07, 2008.

[25] J. Klare, O. Biallawons, and D. Cerutti-Maori, "UAV detection with mimo radar," in *Proceedings of the 18th International Radar Symposium (IRS)*, pp. 1–8, Prague, Czech Republic, August 2017.

[26] C. Liang, N. Cao, X. Lu, and Y. Ye, "UAV detection using continuous wave radar," in *Proceedings of the IEEE International Conference on Information Communication and Signal Processing (ICICSP)*, pp. 1–5, Singapore, November 2018.

[27] M. Caris, S. Stanko, S. Palm, R. Sommer, and N. Pohl, "Synthetic Aperture Radar at Millimeter Wavelength for UAV Surveillance Applications," in *Proceedings of the IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pp. 349–352, Turin, Italy, November 2015.

[28] P. Kaniewski, C. Lenik, W. Susek, and P. Serafin, "Airborne radar terrain imaging system," in *Proceedings of the 16th International Radar Symposium (IRS)*, pp. 248–253, Dresden, Germany, July 2015.

[29] L. Lou, K. Tang, B. Chen et al., "A 253mw/channel 4tx/4rx Pulsed Chirping Phased-Array Radar Trx in 65nm Cmos for X-Band Synthetic-Aperture Radar Imaging," in *Proceedings of the IEEE International Solid - State Circuits Conference - (ISSCC)*, pp. 160–162, San Francisco, CA, USA, March 2018.

[30] S. Park, Y. Kim, E. T. Matson, and A. H. Smith, "Accessible synthetic aperture radar system for autonomous vehicle sensing," in *Proceedings of the IEEE Sensors Applications Symposium (SAS)*, pp. 1–6, Catania, Italy, May 2016.

[31] M. Pieraccini, L. Miccinesi, and N. Rojhani, "Rcs measurements and isar images of small UAVs," *IEEE Aerospace and Electronic Systems Magazine*, vol. 32, no. 9, pp. 28–32, 2017.

[32] J. Xu, M. Liu, F. Zhao, K. Cheng, and L. Yang, "Statistical isar imagery for low-altitude and small-size UAV based on sparse bayesian learning," in *Proceedings of the 6th Asia Pacific Conference on Synthetic Aperture Radar (APSAR)*, pp. 1–6, Xiamen, China, March 2019.

[33] B. K. Kim, H. S. Kang, and S. O. Park, "Drone classification using convolutional neural networks with merged Doppler images," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 1, pp. 38–42, 2017.

[34] N. Joshi, M. Baumann, A. Ehammer et al., "A review of the application of optical and radar remote sensing data fusion to land use mapping and monitoring," *Remote Sensing*, vol. 8, no. 1, p. 70, 01 2016.

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, Las Vegas, NV, USA, July 2016.

[36] J. Redmon and A. Farhadi, "Yolo9000," *Better, faster, stronger*, vol. 12, 2016.

[37] J. Redmon and A. Farhadi, "Yolov3," *An incremental improvement*, vol. 04, 2018.

[38] V. C. Chen, "Micro-Doppler effect of micromotion dynamics: a review," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5102, pp. 240–249, 2003.

[39] E. Swiercz, "Time-frequency transform used in radar Doppler tomography," in *Proceedings of the 15th International Radar Symposium (IRS)*, pp. 1–5, Gdansk, Poland, August 2014.

[40] Z. Sun, J. Wang, C. Yuan, Y. Bi, and H. Xiang, "Parameter estimation of walking human based on micro-Doppler," in *Proceedings of the 12th International Conference on Signal Processing (ICSP)*, Hangzhou, China, pp. 1934–1937, 2014.

[41] A. R. Persico, C. Clemente, C. Ilioudis, D. Gaglione, J. Cao, and J. Soraghan, "Micro-Doppler based recognition of ballistic targets using 2d gabor filters," in *Proceedings of the*

Sensor Signal Processing for Defence (SSPD), pp. 1–5, Edinburgh, UK, October 2015.

[42] J. Lei and C. Lu, "Target classification based on micro-Doppler signatures," in *Proceedings of the IEEE International Radar Conference*, pp. 179–183, Arlington, VA, June 2005.

[43] F. H. C. Tivive, S. L. Phung, and A. Bouzerdoum, "Classification of micro-Doppler signatures of human motions using log-Gabor filters," *IET Radar, Sonar & Navigation*, vol. 9, no. 9, pp. 1188–1195, 2015.

[44] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: bagging, boosting, and variants," *Machine Learning*, vol. 36, pp. 105–139, 1999.

[45] Y. Wei, T. Hong, and C. Fang, *UAV Target Identification Based on Information Fusion of Computer Vision and Radar Signals,* Research Square, Beijing, China, 2022.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 3398071, 7 pages
https://doi.org/10.1155/2022/3398071

Research Article

Firefly Optimization-Based Cooperative Localization Algorithm for Intelligent IoT

Cheng peng Liu,[1] Bin Xia,[2] and Liye Zhang[2]

[1]*National Key Lab of Science and Technology on Communications, University of Electronic Science and Technology of China, Chengdu 611731, China*
[2]*School of Computer Science and Technology, Shandong University of Technology, Shandong 255000, China*

Correspondence should be addressed to Bin Xia; xiabin@sdut.edu.cn

Received 12 April 2022; Revised 29 April 2022; Accepted 2 May 2022; Published 6 June 2022

Academic Editor: Bo Rong

To achieve effective connection between each node of the Internet of Things (IoT), there is a great demand of precise positioning. The traditional firefly localization algorithm only considers part of the ranging information, and the positioning effect is not good. Therefore, an improved algorithm is proposed to take advantage of the ranging information between unknown nodes to achieve cooperative location. First, all ranging information is considered to construct a new objective function. The firefly optimization approach is then utilized to discover the best solution based on the new goal function. The starting location created by the random technique has a significant impact on the new algorithm's localization performance, thus the traditional firefly localization method's initial position is employed to increase cooperative localization performance. The simulation results demonstrate that the novel approach outperforms the classic firefly technique in terms of placement accuracy.

1. Introduction

Due to advancements in wireless communication technology over the previous several decades, the Internet of Things (IoT) is now extensively employed in buildings and residences [1]. A typical Internet of Things system consists of low-power, low-cost devices that connect with one another through the Internet. The Internet of Things' primary purpose is to guarantee that all devices, including sensors, smartphones, wearable sensors, tablets, and transportation systems, can connect with one another via a standard interface. This enables machine-to-machine communication without the need for human intervention [2], potentially reducing manual labour costs. One of the most crucial difficulties in IoT is precise positioning. Consequently, wireless localization, or the extraction of an object's geolocation information, has received a lot of attention. A Global Positioning System module may offer position information to a wireless device, but it is expensive and energy-intensive. As a result, GPS may not be suitable for all IoT applications.

The Internet of Things (IOT) industry is growing at an astonishing rate, with the application of location-based services gradually permeating people's lives. For example, in a smart factory, the Internet of Things collects key data of the equipment in the factory in real time through sensors, completes data storage, analysis, and other processing, timely and effectively handles equipment failures, and ensures the normal operation of equipment. The IoT management platform manages equipment in a unified manner, allocates and uses equipment reasonably, arranges production plans, and improves the overall operational efficiency of smart factories. In smart agriculture [3, 4], the Internet of Things collects data such as temperature, then deals with these data, and combines precise positioning information to achieve smart operation of agricultural production. Precise positioning is the key to IoT location-based service applications.

Node positioning requires ranging information. As shown in Figure 1, the dotted line represents ranging information from unknown nodes to anchor nodes, and the solid line represents ranging information between unknown

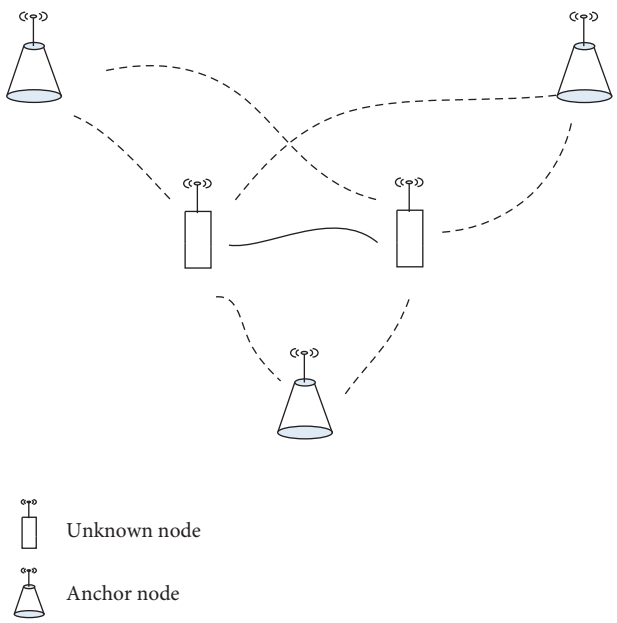


Figure 1: The positioning panorama for IoT network.

nodes. According to ranging information provided by the ranging methods such as arrival time [5, 6] or received signal strength [7–9], a positioning equation system is established, and the equation system is solved to obtain the position of the node. Common solution methods include particle swarm algorithm [10–12] and firefly algorithm [13–15]. The firefly algorithm has the advantages of few parameters to be adjusted, is easy to implement, has simple structure, etc., and it is widely used. The goal function in the traditional firefly localization method only comprises ranging information from unknown nodes to anchor nodes and does not fully use the ranging information between unknown nodes, resulting in limited positioning accuracy increase. In view of this, we propose a firefly-based optimization method. To increase the position accuracy, the cooperative algorithm may make full use of all range information.

In addition to our proposed algorithm, there are other cooperative algorithms. Reference [16] searches for the optimal solution along the gradient descent direction; reference [17] proposes a statistical manifold (SM) method, which uses the Riemann metric tensor to correct the given gradient optimization direction to find the optimal solution. They all use the gradient information to find the appropriate solution through iterative calculation, but the establishment of the iterative coefficient matrix is very complicated. In the literature [18, 19], the colocalization problem needs to be transformed by a semidefinite programming (SDP) method. It can only be solved for convex optimization problems. However, this conversion process is more complicated. An upgraded cuckoo search (CS) localization algorithm [20] was presented that can exploit all of the distance information available. In relation to the SDP and SM methods, both our algorithm and the upgraded CS algorithm have relatively simpler operation and do not need to establish a convex optimization problem or construct an iterative coefficient matrix.

The reminder of the paper is organized as follows. Section 2 introduces the traditional firefly localization algorithm and shows how to use the firefly algorithm to obtain the estimated location of unknown nodes. Section 3 explains that the traditional firefly localization algorithm has limited room for improvement in accurate localization. Therefore, we propose a firefly-based cooperative algorithm and describe the colocalization algorithm in detail. In Section 4, we use MATLAB to conduct simulations and show the influence of ranging error, initial position, and search step size on the positioning error. Finally, Section 5 concludes the paper.

2. Traditional Firefly Localization Algorithm

It is assumed that in the positioning area, the coordinates of the anchor node are $(u_k, v_k)k \in \{1, 2, \ldots, \mathrm{K}\}$; the coordinates of the unknown node are (x, y). According to the ranging information between the unknown node and all anchor nodes, the established positioning equations are

$$\begin{cases} d_1 = \sqrt{(x-u_1)^2 + (y-v_1)^2} + e_1, \\ \vdots \\ d_k = \sqrt{(x-u_k)^2 + (y-v_k)^2} + e_k \\ \vdots \\ d_{\mathrm{K}} = \sqrt{(x-u_{\mathrm{K}})^2 + (y-v_{\mathrm{K}})^2} + e_{\mathrm{K}}, \end{cases}, \tag{1}$$

where $\sqrt{(x-u_k)^2 + (y-v_k)^2}$ and d_k represent the actual distance and the measured distance from the k-th anchor node (u_k, v_k) to the unknown node (x, y), respectively, and e_k represents the ranging error. Randomly initializing the firefly position in the positioning area, the position of the n-th firefly can be expressed as $\widehat{\mathrm{X}}_n = (x_n, y_n)^T$, the objective function defined according to formula (1) is

$$f(\widehat{\mathrm{X}}_n) = \frac{1}{\mathrm{K}} \sum_{k=1}^{\mathrm{K}} \left(\sqrt{(x_n-u_k)^2 + (y_n-v_k)^2} - d_k \right)^2. \tag{2}$$

Through the objective function given by formula (2), the estimated position of the unknown node can be obtained by using the firefly algorithm. Algorithm 1 gives the specific flow of the traditional firefly algorithm.

3. Cooperative Localization Algorithms with Firefly Optimization

In the traditional algorithm, the objective function only considers the distance between the anchor node and the unknown node and does not consider others. The use of ranging information is insufficient, resulting in very limited space for improving the positioning accuracy. In view of this, a firefly-based cooperative algorithm is proposed. First, instead of selecting a random position as the initial position, the initial position is determined by the traditional firefly positioning method; then, all ranging information is used for precise positioning.

Suppose there is K anchor node and I unknown node randomly distributed in the positioning area, where the

```
Input:
d_k the measured distance from the k-th anchor node
(u_k, v_k) to the unknown node
Output:
(x̂, ŷ) the estimated position of the unknown node
Randomly initialize the fireflies;
While maximum iteration is not reached do
Calculate the objective value of the firefly using formula (2)
    Calculate the relative brightness of fireflies;
    Calculate the attraction of fireflies;
    Update the position
end
```

Algorithm 1: Traditional firefly algorithm. Specific flow of the traditional firefly algorithm.

position of the anchor node is expressed as (x_k, y_k), $k \in (1, \ldots, \mathrm{K})$, and the position of the unknown node is expressed as $(x_i, y_i), i \in (1, \ldots, \mathrm{I})$. The positioning equation established according to the ranging information between all the nodes and unknown node as

$$\begin{cases} d_{1,1} = \sqrt{(x_1 - u_1)^2 + (y_1 - v_1)^2} + e_{1,1} \\ \vdots \\ d_{i,k} = \sqrt{(x_i - u_k)^2 + (y_i - v_k)^2} + e_{i,k} \\ \vdots \\ d_{\mathrm{I},\mathrm{K}} = \sqrt{(x_\mathrm{I} - u_\mathrm{K})^2 + (y_\mathrm{I} - v_\mathrm{K})^2} + e_{\mathrm{I},\mathrm{K}} \\ d'_{1,2} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + e'_{1,2} \\ \vdots \\ d'_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + e'_{(\mathrm{I}-1),\mathrm{I}}, i < j \\ \vdots \\ d'_{(\mathrm{I}-1),\mathrm{I}} = \sqrt{(x_{(\mathrm{I}-1)} - x_\mathrm{I})^2 + (y_{(\mathrm{I}-1)} - y_\mathrm{I})^2} + e'_{(\mathrm{I}-1),\mathrm{I}}, \end{cases} \tag{3}$$

where $\sqrt{(x_i - u_k)^2 + (y_i - v_k)^2}$ and $d_{i,k}$ represent the real distance and the measured distance from the anchor node to the unknown node, respectively; $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ and $d'_{i,j}$ represent the real distance and measured distance from the i-th unknown node to the j-th unknown node, respectively. $e_{i,k}$ and $e'_{i,j}$ represent the ranging error. The cooperative process based on firefly optimization is as follows:

(1) $i = 1$

(2) Calculate the initial position of the i-th unknown node $(\hat{x}_i, \hat{y}_i)$ by using the traditional firefly localization algorithm.

(3) Determine whether the initial position calculation of all unknown nodes is completed; if not, $i = i + 1$, go to step 2; otherwise, go to the next step.

(4) Suppose there is N firefly, $X_n^{(l)} = (x_{n,x_1}^{(l)}, y_{n,y_1}^{(l)}, \ldots, x_{n,x_1}^{(l)}, y_{n,y_1}^{(l)})^T$ indicates the position of the first firefly, and

$$\begin{cases} x_{n,x_i}^{(l)} = \hat{x}_i + r_1 \\ y_{n,y_i}^{(l)} = \hat{y}_i + r_2 \end{cases}, i = 1, 2, \ldots, \mathrm{I}, \tag{4}$$

where the random numbers r_1 and r_2 both obey the zero mean normal distribution with variance $\sigma^2 = 1$, and l represents the number of searches.

(5) The objective function defined according to formula (3) is

$$f\left(X_n^{(l)}\right) = \frac{1}{\mathrm{K}_2} \sum_{i=1}^{\mathrm{I}} \sum_{k=1}^{\mathrm{K}} \left(\sqrt{\left(x_{n,x_i}^{(l)} - u_k\right)^2 + \left(y_{n,y_i}^{(l)} - v_k\right)^2} - d_{i,k} \right)^2 + \frac{1}{\mathrm{K}_2} \sum_{i=1}^{\mathrm{I}-1} \sum_{\substack{j=1 \\ j>i}}^{\mathrm{I}} \left(\sqrt{\left(x_{n,x_i}^{(l)} - x_{n,x_j}^{(l)}\right)^2 + \left(y_{n,y_i}^{(l)} - y_{n,y_j}^{(l)}\right)^2} - d'_{i,j} \right)^2. \tag{5}$$

Among them, $\mathrm{K}_2 = C_\mathrm{I}^1 C_\mathrm{K}^1 + C_\mathrm{I}^2$ is the total number of measured distances between all nodes. Calculate the objective function value corresponding to each firefly according to formula (5).

(6) Calculate the relative brightness of fireflies with reference to the following formula:

$$I = I_0 e^{-\gamma r_{mn}^2}, \tag{6}$$

where r_{mn} indicates the distance between fireflies, I_0 indicates the strongest brightness, and γ indicates the absorption coefficient. The relative brightness can help you determine the direction of firefly movement.

```
Input:
d_{i,k} the measured distance from the k-th anchor node to the i-th unknown node,
d'_{i,j} the measured distance from the i-th unknown node to the j-th unknown node
Output:
(x̂_i, y_i), i ∈ (1, ..., I) the estimated position of the i-th unknown node
i = 1;
while all the unknown nodes' positions have not been calculated do
calculate the initial position of the i-th unknown node using the firefly algorithm;
i = i + 1;
end
randomly initialize the fireflies;
while maximum iteration is not reached do
calculate the objective value of the firefly using formula (5)
calculate the relative brightness of fireflies using formula (6);
calculate the attraction of fireflies using formula (7);
update the position;
end
```

Algorithm 2: Cooperative localization algorithm based on firefly optimization. Specific flow of the cooperative localization algorithm.

(7) Calculate the attraction of fireflies with reference to the following formula:

$$\beta = \beta_0 e^{-\gamma r_{mn}^2}, \tag{7}$$

where β_0 represents the greatest attraction.

(8) Update the position with reference to the following formula:

$$X_n^{l+1} = X_n^l + \beta\left(X_m^l - X_n^l\right) + \alpha \times \varepsilon, \tag{8}$$

where X_m^l is the position of the m-th firefly, α is the search step, and the random number ε obeys a uniform distribution on the interval $(-0.5, 0.5)$.

Determine whether the maximum number of searches has been reached. If the search is completed, the output result is the optimal position of the unknown node. Otherwise, jump to Step 5 and continue the search. Algorithm 2 gives the specific flow of the cooperative localization based on firefly optimization, and notation of variables is given in Table 1 for clarity.

Table 1: Notation of variables.

Notation	Items
$d_{i,k}$, $d'_{i,j}$	The ranging distance
(u_k, v_k)	The position of the k-th anchor node
(x_i, y_i)	The position of the i-th unknown node
$e_{i,k}$, $e'_{i,j}$	The ranging error
X_n^l	The position of the n-th firefly
r_{mn}	The distance between fireflies
β	The attraction of the firefly
α	The search step

4. Simulation Study

In simulation experiments, the main parameters of the traditional firefly positioning algorithm are the fireflies' number $N = 20$, the searches' number $L_1 = 100$, and the search step $\alpha = 0.25$. The main parameters of the colocation based on firefly optimization are the fireflies' number $N = 20$, the searches' number $L_2 = 100$, and the search step $\alpha = 0.05$. A rectangular positioning area is 12m × 12m in which all nodes are randomly distributed. The range inaccuracy is supposed to have a variance of σ^2 and a zero-mean Gaussian distribution. The positioning error is defined as

$$\frac{\sum_{i=1}^{\mathrm{I}} \sqrt{\left(\widehat{x}_i - x_i\right)^2 + \left(\widehat{y}_i - y_i\right)^2}}{\mathrm{I}}, \tag{9}$$

where $(\widehat{x}_i, \widehat{y}_i)$ represents the position of the i-th node.

4.1. Influence of Ranging Error on Positioning Error. As shown in Figure 2, with the reduction of the ranging error, the localization mistake of the localization algorithm also decreases, but the localization error of the new algorithm is significantly lower than that of the traditional firefly localization algorithm. This is because the new algorithm considers not just the ranging information between the anchor node and the unknown node but also the ranging information among the unknown nodes, which increases placement accuracy. According to Table 2, the improvement percentage of positioning performance under different conditions is different, but the improvement percentage is more than 31%. Simultaneously, the improvement effect is better as the number of unknown nodes increases.

An overview of these methods' performance comparison is provided by Table 3 in terms of ranging information usage, complexity, operation procedure, and positioning accuracy. All methods possess their drawbacks and advantages, which make them more appropriate for specific uses.

4.2. Influence of Initial Position on Positioning Error. As shown in Figure 3, when the colocalization algorithm based on firefly optimization uses a randomly generated initial

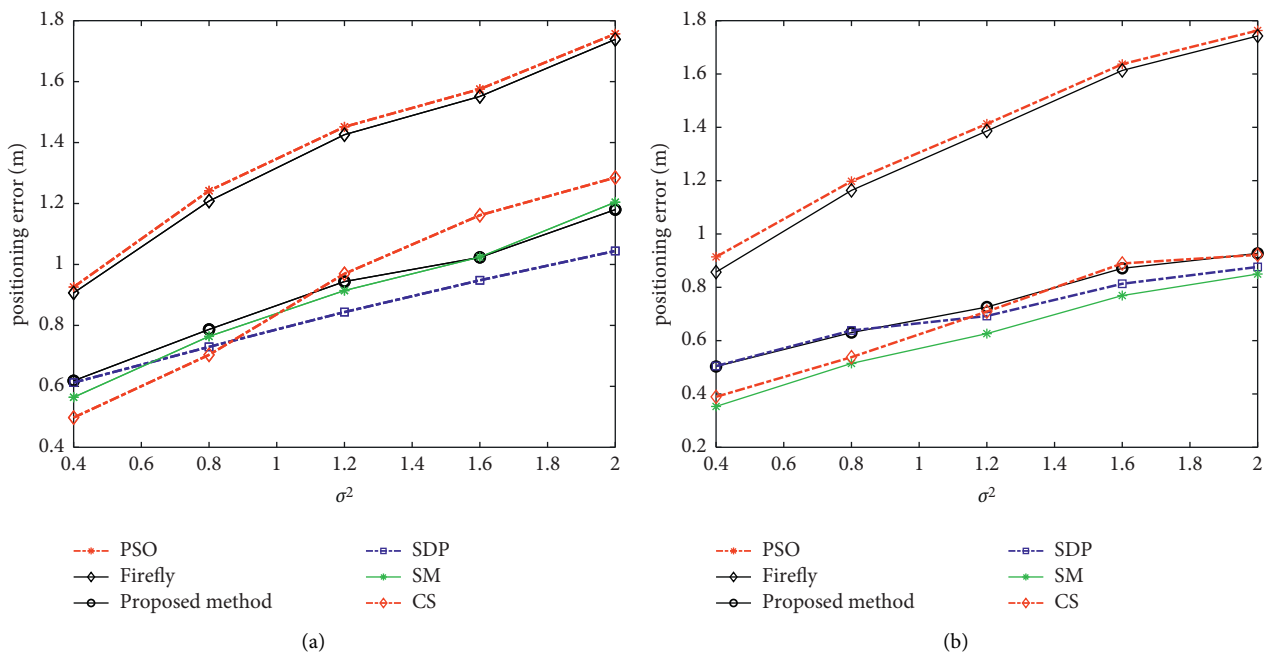


Figure 2: The variation of the positioning error with the variance of the ranging error. (a) I = 8, $K = 4$; (b) I = 16, $K = 4$.

Table 2: Improvement percentage of the positioning performance.

Ranging error	Positioning error/m, I = 8		Improvement percentage	Positioning error/m, I = 16		Improvement percentage
	Traditional firefly	Proposed method		Traditional firefly	Proposed method	
0.4	0.9067	0.6183	31.8	0.8570	0.503	41.3
0.8	1.2080	0.7868	34.9	1.1635	0.6303	45.8
1.2	1.4259	0.9438	33.8	1.3863	0.726	47.6
1.6	1.5514	1.0232	34	1.6133	0.8712	46
2	1.7385	1.1799	32.1	1.7426	0.9272	46.8

Table 3: Performance comparison of positioning methods.

Method	Ranging information usage	Complexity	Operation procedure	Positioning accuracy
PSO, firefly	Part	Low	Low	Low
Proposed method, upgraded CS	ALL	High	Middle	High
SDP, SM	ALL	High	High	High

value, its localization error is larger than that of the traditional firefly algorithm. The result shows that the initial value generated by the random method seriously affects the localization of the new algorithm performance, and the initial position provided by the traditional firefly algorithm significantly improves the positioning accuracy of the new algorithm.

4.3. Influence of Search Step Size on Positioning Error. The size of the search step is crucial for balancing the algorithm's global and local search capabilities. A bigger step size is advantageous for increasing population variety, searching for areas with more ideal values, improving global search capability, and avoiding entering the premature convergence position. A smaller step size can improve the local search ability, which is conducive to refined search and improves the accuracy of the positioning result. As shown in Figure 4(b), the optimal value range of the search step size is 0.04–0.05.

4.4. Complexity Analysis. The time complexity used for traditional firefly positioning is approximately $O(L_1 * N^2 * K * I)$, where the number of anchor nodes is K, the number of unknown nodes is I, the maximum number of searches is L_1, and the number of fireflies is N. The time complexity of firefly colocation is $O(L_1 * N^2 * K * I) + O(L_2 * N^2 * I^2)$,

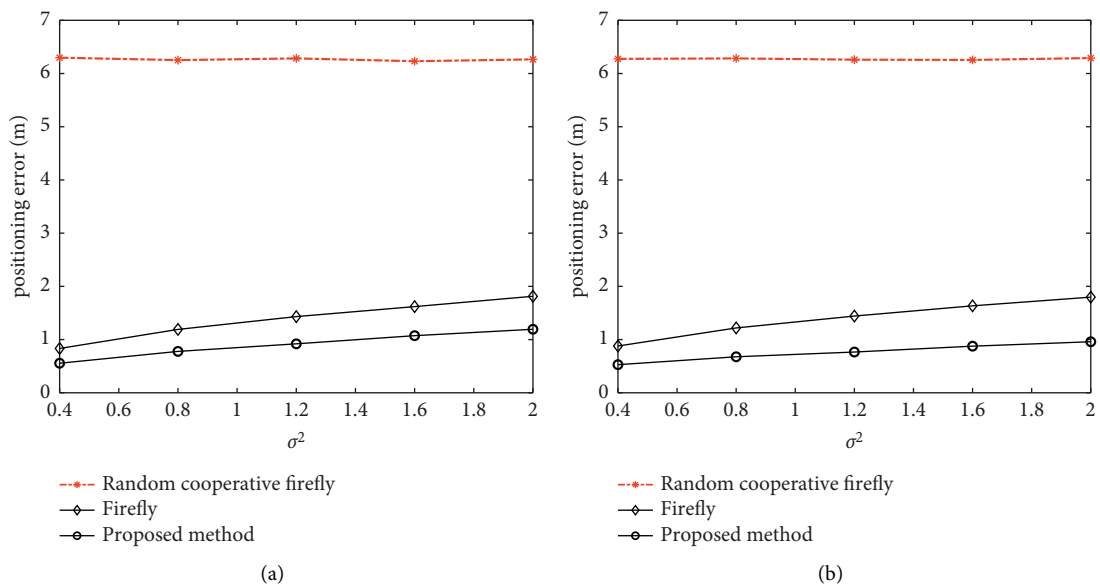


Figure 3: The effect of initial value on positioning accuracy. (a) I = 8, $K = 4$; (b) I = 16, $K = 4$.

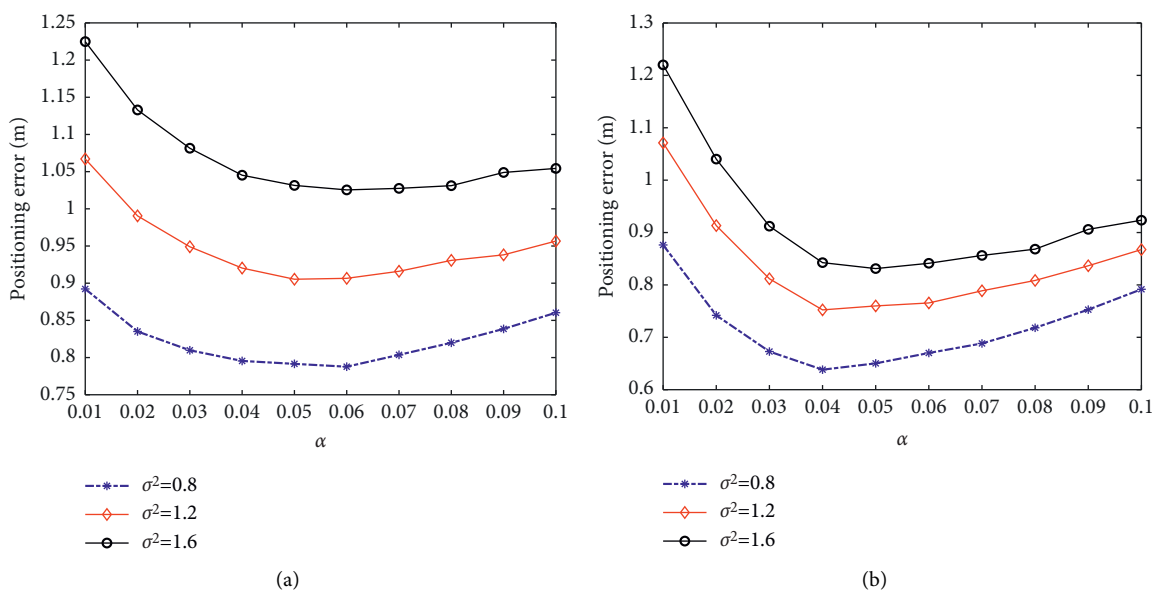


Figure 4: Influence of different search step size on positioning error. (a) I = 8, $K = 4$; (b) I = 16, $K = 4$.

where L_2 is the maximum number of searches. In contrast to the placement method, the new technique searches for unknown nodes using range information, which leads to an increase $O(L * N^2 * I^2)$ in time complexity, but brings the benefits of significantly improved positioning accuracy.

5. Conclusion

This work has enhanced the localization performance of standard firefly technique by making full use of all range information to accomplish cooperative location. The algorithm creates an objective function with range information between unknown nodes, which it then solves using the firefly optimization process. The simulation results justify that our proposed approach may lessen range error's impact. In the future, this research can also be applied to the Internet of Things, which rapidly expands a wide range of applications by connecting everything, moving towards a smart future world. It is very helpful for the precise positioning of various devices in the Internet of Things. The upcoming 5G-based IoT has many complex application scenarios that require more efficient location services, so this research has far-reaching implications for the Internet of Things.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62001272.

References

[1] Y. Wu, H. N. Dai, H. Wang, Z. Xiong, and S. Guo, "A survey of intelligent network slicing management for industrial IoT: integrated approaches for smart transportation, smart energy, and smart factory," *IEEE Communications Surveys & Tutorials*, pp. 1–38, 2022.

[2] W. Yu, Y. Liu, T. Dillon, W. Rahayu, and F. Mostafa, "An integrated framework for health state monitoring in a smart factory employing IoT and big data techniques," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 2443–2454, 2022.

[3] P. K. Kashyap, S. Kumar, A. Jaiswal, M. Prasad, and A. H. Gandomi, "Towards precision agriculture: IoT-enabled intelligent irrigation systems using deep learning neural network," *IEEE Sensors Journal*, vol. 21, no. 16, Article ID 17479, 2021.

[4] S. Silva, D. Duarte, and A. Valente, "Augmented intelligent distributed sensing system model for precision agriculture," in *Proceedings of the Telecoms Conference*, Leiria, Portugal, February, 2021.

[5] Z. Liu, R. Chen, F. Ye, G. Guo, Z. Li, and L. Qian, "Improved TOA estimation method for acoustic ranging in a reverberant environment," *IEEE Sensors Journal*, 2020.

[6] S. Zeb, "On TOA-based ranging over mmWave 5G for indoor industrial IoT networks," in *Proceedings of the IEEE Globecom Workshops*, pp. 1–6, Taipei, Taiwan, December, 2020.

[7] H. Li, Z. Huang, H. Sun, X. Wang, and J. Qi, "A dynamic adaptive indoor ranging model based on RSSI," in *Proceedings of the 39th Chinese Control Conference*, pp. 2850–2855, Shenyang, China, July, 2020.

[8] J. Luomala and I. Hakala, "Analysis and evaluation of adaptive RSSI-based ranging in outdoor wireless sensor Networks," *Ad Hoc Networks*, vol. 87, pp. 100–112, 2019.

[9] Y. C. Lin, C. C. Sun, and K. T. Huang, "RSSI measurement with channel model estimating for IoT wide range localization using LoRa communication," in *Proceedings of the International Symposium on Intelligent Signal Processing and Communication Systems*, pp. 1-2, Taipei, Taiwan, December, 2019.

[10] S. P. Singh and S. C. Sharma, "Implementation of a PSO based improved localization algorithm for wireless sensor networks," *IETE Journal of Research*, vol. 65, no. 4, pp. 502–514, 2019.

[11] Z. Hao, X. Li, and Y. Ding, "An improved PSO algorithm for node localization in indoor long-narrow confined space," in *Proceedings of the 13th IEEE Conference on Industrial Electronics and Applications*, pp. 1841–1846, Wuhan, China, June, 2018.

[12] S. P. Singh and S. C. Sharma, "A PSO based improved localization algorithm for wireless sensor network," *Wireless Personal Communications*, vol. 98, no. 1, pp. 487–503, 2018.

[13] R. Harikrishnan, V. Jawahar Senthil Kumar, and P. Sridevi Ponmalar, "Firefly algorithm approach for localization in wireless sensor networks," in *Proceedings of the3rd International Conference on Advanced Computing*, pp. 209–214, Kaohsiung, Taiwan, November, 2015.

[14] T. Mao, S. Van-Oanh, S. Chin-Shiuh, and N. Trong-The, "Parallel firefly algorithm for localization algorithm in wireless sensor network," in *Proceedings of the 2015 International Conference on Robot, Vision and Signal Processing*, pp. 300–305, Kaohsiung, Taiwan, November, 2015.

[15] W. Peng, S. Shaojing, Z. Zhen, and G. Xiaojun, "Time difference of arrival (TDOA) localization combining weighted least squares and firefly algorithm," *Sensors*, vol. 19, no. 11, pp. 1–14, 2019.

[16] M. NaraghiPour and G. C. RojasSensor, "Network localization via distributed randomized gradient descent," in *Proceedings of the IEEE Military Communications Conference*, pp. 1714–1719, Diego, CA, USA, November2013.

[17] B. Xia, W. Yuan, N. Xie, and C. Li, "A novel statistical manifold algorithm for position estimation," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1513–1518, 2019.

[18] S. Kim and M. Kojima, "Semidefinite programming relaxations for sensor network localization," in *Proceedings of the IEEE International Symposium on Computer-Aided Control System Design*, pp. 19–23, Yokohama, Japan, September, 2010.

[19] Yanhu, S. Zhuo, and R. Fan, "A semidefinite programming algorithm for improving noisy sensor positions using accurate inter-sensor range measurements," in *Proceedings of theInternational Conference on Image Processing and Machine Vision*, pp. 124–129, Hong Kong China, May, 2020.

[20] X. Qin, B. Xia, T. Ding, and L. Zhao, "An improved Cuckoo search localization algorithm for UWB sensor networks," *Wireless Networks*, vol. 27, no. 1, pp. 527–535, 2021.