sensors

Article

JamCatcher: A Mobile Jammer Localization Scheme for Advanced Metering Infrastructure in Smart Grid

Taimin Zhang, Xiaoyu Ji *, Zhou Zhuang and Wenyuan Xu

School of Electrical Engineering, Zhejiang University, Hangzhou 310027, China; zhangtaimin@zju.edu.cn (T.Z.); zhuangzhou@zju.edu.cn (Z.Z.); wyxu@zju.edu.cn (W.X.)

* Correspondence: xji@zju.edu.cn

Received: 6 January 2019; Accepted: 18 February 2019; Published: 21 February 2019

Abstract: As the core component of the smart grid, advanced metering infrastructure (AMI) is responsible for automated billing, demand response, load forecasting, management, etc. The jamming attack poses a serious threat to the AMI communication networks, especially the neighborhood area network where wireless technologies are widely adopted to connect a tremendous amount of smart meters. An attacker can easily build a jammer using a software-defined radio and jam the wireless communications between smart meters and local controllers, causing failures of on-line monitoring and state estimation. Accurate jammer localization is the first step for defending AMIs against jamming attacks. In this paper, we propose `JamCatcher`, a mobile jammer localization scheme for defending the AMI. Unlike existing jammer localization schemes, which only consider stationary jammers and usually require a high density of anchor nodes, the proposed scheme utilizes a tracker and can localize a mobile jammer with sparse anchor nodes. The time delay of data transmission is also considered, and the jammer localization process is divided into two stages, i.e., far-field chasing stage and near-field capturing stage. Different localization algorithms are developed for each stage. The proposed method has been tested with data from both simulation and real-world experiment. The results demonstrate that `JamCatcher` outperforms existing jammer localization algorithms with a limited number of anchor nodes in the AMI scenario.

Keywords: smart grid; advanced metering infrastructure (AMI); smart meters; jamming; radio interference; localization

1. Introduction

Advanced metering infrastructure (AMI) is the core component of the smart grid [1]. It enables bidirectional communication between utility companies and customers by integrating advanced sensors, smart meters, monitoring systems, computer software/hardware, and data management systems [2]. AMI plays a pivotal role in several critical tasks such as automated billing, demand response, load forecast, and management [3]. Governments around the world are putting effort into developing AMIs. For instance, a total of 65 million smart meters were projected to be installed in the U.S. by 2015, covering more than half of all U.S. households. China has installed over 400 million smart meters through its state-run utilities since launching the program in 2012. The U.K. government plans to roll-out smart meters in every household by 2020.

An AMI typically consists of smart meters in customers' home area, a meter data management system (MDMS) at the control center, and communication networks at different levels of the infrastructure hierarchy that transmit data between smart meters and the MDMS. The tremendous amount of smart meters is connected through the neighborhood area network (NAN). As wireless communication technologies are usually utilized to create wireless multi-hop routes for smart meters, the NAN of AMI is vulnerable to jamming attack. A malicious attacker may use a jammer to interfere with the communication between smart meters (shown in Figure 1) and move in the NAN to affect as many smart meters as possible. The jamming attack could be the first step to launch a variety of crimes in the smart grid. Taking energy theft for example, the criminal could jam the power price signaling in a highly-populated city to manipulate the price and make a profit [4–6]. Furthermore, an adversary can launch a false data injection attack against the smart grid by jamming the wireless transmission between AMI sensors and a remote estimator, and simultaneously modify the system's physical state to produce a misleading operational decision of the smart grid [7–9]. As AMI plays an important role in the smart grid, it is essential to defend AMI from such jamming attacks.

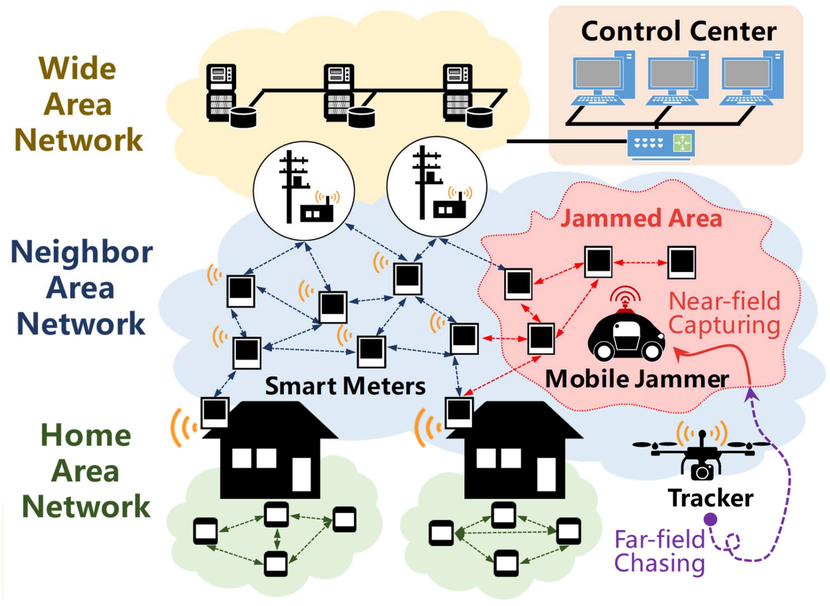


Figure 1. A jamming attack scenario in the neighbor area network of an AMI.

To defend the AMI communication networks against jamming attacks and further catch the jammers, accurate jammer localization is a key step. As industrial unmanned aerial vehicles (UAVs) have been a trend in smart grids, such as the application of high-voltage power lines' inspection and power network damage assessment [10–13], a promising solution is to deploy a tracker (a drone or a mobile robot) in an AMI network to track the jammer without human intervention. The UAV-assisted tracker can move in the area where jamming attacks happened and localize the jammer with the help of affected smart meters. The tracker can collect measurements from smart meters directly, thus achieving real-time localization and tracking. In this paper, we present `JamCatcher`, a mobile jammer localization and tracking scheme for AMI networks. `JamCatcher` utilizes a tracker, e.g., a UAV, to measure the distance between each anchor node (i.e., smart meters) and the jammer with jamming signal strength (JSS) as the input, to finally derive the location of the jammer.

In developing `JamCatcher`, the main challenge is to localize the jammer with a limited number of JSS measurements from anchor nodes. This is due to two reasons, i.e., the sparse distribution of smart meters and the lossy link for data transmission. A sparse distribution means smart meters are far from each other. Taking a typical neighborhood in the U.S. as an example, smart meters are often tens of meters away from each other. As JSS decreases rapidly as a function of the distance, the estimated distance between the anchor node and the jammer becomes more inaccurate as the distance between them increases [14,15]. For instance, the results in [15] show the averaged estimation error increases from less than 1 m to 2.6 m as the distance increases from 10 m to 43 m. Thus, for good accuracy, only measurements from smart meters located within a specific range (referred to as effective range in the rest of the paper) of the jammer should be used, which limits the number of measurements. The lossy link means the links between the smart meters are characterized by high loss rates and instability due to the jamming attack, and the tracker can only receive part of the measurements sent by smart meters located in the effective range. Most existing jammer localization methods are not well applicable to AMI networks since they require anchor nodes to be densely distributed to achieve good accuracy [16–26]. For example, some existing methods utilize geometric characteristics of the jammed area to localize the jammer [18,19,24,25,27], and these methods require a high density of anchor nodes, typically 100 to 300 nodes in 100 m^2. To solve the problem of limited measurements, we take advantage of the jammer's mobility. As the jammer moves in the NAN, it keeps affecting new smart meters. `JamCatcher` utilizes the unscented Kalman filter (UKF) and interacting multiple model (IMM) to estimate the state of the jammer as a weighted average of the model's predicted state and of the new JSS measurements. The main contributions of this paper are presented as follows:

1. As far as we know, `JamCatcher` is the first work to study the jammer localization problem in an AMI communication network. `JamCatcher` provides a mobile jammer localization scheme for anti-jamming strategies in the AMI utilizing UAVs.
2. `JamCatcher` can localize a mobile jammer with a limited number of anchor nodes in a complex environment, while most existed jammer localization methods cannot.
3. Both a real-world experiment and MATLAB simulations are conducted to evaluate `JamCatcher`. Extensive computer simulations are carried out to compare the localization accuracy of `JamCatcher` with existing jammer localization methods.

We organize the remainder of the paper as follows. We discuss the related work in Section 2. We introduce related models and formulate the problem in Section 3. In Section 4, we present our framework for localizing a mobile jammer and give the implementation of our algorithms. In Section 5, we validate the proposed localization algorithms through a real-world experiment and extensive MATLAB simulations. Finally, we conclude in Section 6.

2. Related Work

Jamming attacks can seriously damage the stability of the smart grid. For instance, a malicious attacker can use jamming attack to damage the accuracy of the smart grid's state estimation. Deka et al. [28] introduced an undetectable jamming attack that uses measurement jamming in addition to changing meter readings to create a change in state estimation while remaining undetected by the bad-data detection test. Similarly, Guan et al. [7] introduced a false data injection attack that physically changes the system's state and uses jamming attack to block the wireless transmission channels between sensors and remote estimators, leading to the failure of smart grid's state estimation. Gai et al. [29] introduced an attack strategy that optimally distributed power usage on both spoofing and jamming attacks to increase the adversarial implication to the wireless communications of the smart grid. Some recent research applied the game theory to study jamming attacks against the smart grid's state estimation and developed defense strategies in the smart grid [6,30]. A few works have been focused on studying the possibility of manipulating the power market by jamming the pricing signal [4–6]. Although jamming attacks in the smart grid have drawn much attention, defense strategies are only given theoretically, and so, an anti-jamming scheme aimed at practical use is an urgent need.

Accurate jammer localization plays an essential role in applying various anti-jamming countermeasures [31]. Based on the localization methodology, the existing jammer localization algorithms can be categorized into range-based methods [16,17,20–23,26,32] and range-free methods [18,19,24,25,27]. In the range-based methods, the distance between the jammer and the anchor nodes derived from measurements is utilized to estimate the location of the jammer. Range-free methods usually rely on the geometric characteristics of the jammed area to locate the jammer. Prior research has addressed jammer localization problems in different scenarios, such as wireless local area networks (WLAN) [20–22], wireless sensor networks (WSN) [23–25], and mobile ad-hoc networks (MANET) [26]. Localizing a jammer in an AMI communication network is different from that in other networks due to the large scale and sparse distribution of the anchor nodes. Existing jammer localization methods are not applicable to AMI for three reasons. First, the accuracy of most existing methods relies on a high density of anchor nodes, e.g., range-free methods require typically 100 to 300 nodes in 100 m^2 [18,19]. The performance of existing methods degrades evidently as the node density goes extremely low. Second, existing methods assume a sink node far from the jammed area, which can collect JSS measurements from sufficient anchor nodes. However, this is not practical for localizing a mobile jammer in reality since the time delay can be large. Utilizing a tracker to enter the jammed area and collect real-time measurements is a good way to solve the problem. Third, existing methods are developed for localizing stationary jammers, and they consider the JSS measurements or the jammed area time-invariant and obtain the estimated jammer location through static estimations [16–26], which is not applicable to localizing mobile jammers. The most similar to our work, Wu et al. [33] proposed a radio-based robot localization method with low-density WLAN access points (APs). They installed ten APs in an area with a size of 54 m $\times$ 53 m, and their positioning system only selected two APs from the detected AP set for localization. However, they used an RSS fingerprinting approach, and their method required 236 anchor nodes to measure the AP signal and build a radio map. Building a radio map for jammer localization is unrealistic since the jammer is non-cooperative and its transmitting power is unknown. As far as we knew, we present here the first work to design a mobile jammer localization framework for anti-jamming technology in the AMI communication network utilizing UAVs. We propose localization algorithms that can localize mobile jammers with complex motion and can achieve good accuracy when the anchor nodes are sparsely distributed.

3. Problem Overview

This paper focuses on developing a mobile jammer localization scheme and corresponding localization algorithms for anti-jamming technology in low-density AMI communication networks using mobile trackers. We assume the network is able to identify jamming attacks leveraging the existing jamming detection approaches [34,35]. In this section, we explain the jamming attack scenario in AMI networks and give the assumptions of our work. Then, we formulate the jammer localization problem as a discrete-time state estimation problem.

3.1. Localizing the Mobile Jammer in AMI

The AMI communication network features a hierarchical structure (shown in Figure 1), including home area network (HAN), neighbor area network (NAN), and wide area network (WAN). The HAN is composed of smart devices within a home, and a smart meter acts as a gateway in an HAN. The NAN is composed of smart meters, access points, and data collectors installed in the power supply area. The smart meters form a mesh network using wireless communication technologies such as ZigBee and WiFi. Each access point collects data from smart meters in a specific area and delivers them to the data collector. In case the communication link is in a poor condition, each smart meter can act as a repeater and deliver the data in a multi-hop way. Data collectors deliver the data from access points to the control center through the WAN.

Jamming attacks usually occur in the NAN and interfere with the communication between smart meters. The jammer moves within the network and transmits the jamming signal at a constant power level. When the signal-to-noise ratio (SNR) of a wireless link is lower than a threshold, the two nodes will not be able to communicate. According to [36], the nodes under jamming attack can be divided into three types, i.e., jammed node, boundary node, and unaffected node.

- Unaffected node: Unaffected nodes are nodes that can receive packets from all of their neighbors after jamming is present.
- Jammed node: Jammed nodes are nodes that cannot receive messages from any unaffected nodes.
- Boundary node: Boundary nodes are nodes that can receive packets from a part of, but not all of their neighbors.

Jammed nodes and boundary nodes form a jammed area, where nodes can take good measurements of the jamming signal directly. Different from most existing jammer localization algorithms, which only utilize measurements from boundary nodes, `JamCatcher` makes use of measurements from both boundary nodes and jammed nodes. To be specific, `JamCatcher` divides the jammer localization process into two stages, i.e., far-field chasing and near-field capturing. The far-field chasing stage starts from the time when the tracker is released in the NAN. In this stage, the tracker is unaffected by the jammer (i.e., out of the jammed area) and can receive JSS measurements from all the boundary nodes through multi-hop transmission. The near-field capturing stage starts as the tracker enters the jammed area. In this stage, the tracker itself becomes an affected node. It will only be able to collect JSS measurements from its neighbors through one-hop communication. The neighbor nodes that the tracker can communicate with include boundary nodes and jammed nodes (when the tracker is close to the jammed node and the SNR is above the threshold).

3.2. Related Models

We assume smart meters are able to measure the jamming signal utilizing the method proposed in [17]. Each smart meter is aware of its own position using generalized pattern search (GPS) or some other similar, but less burdensome localization algorithms [37,38]. When the jammer moves in the NAN, affected smart meters can measure the JSS and send the measurements to the tracker. Whether the tracker can receive the messages from smart meters depends on the SNR. In the rest of the paper, observation nodes refer to affected smart meters with which the tracker can communicate. Some related models are given as follows.

The mobile jammer we try to localize has the following characteristics.

Mobile: The jammer moves within a 2D plane with arbitrary speed and towards arbitrary directions. Both the speed and moving directions can not be measured.

Unknown transmit power: The jammer transmits signals at a constant power level and will not change over time. The transmit power level of the jammer is unknown.

Omnidirectional: The jammer is equipped with an omnidirectional antenna.

Based on the characteristics of the AMI network, we examine the smart meters with the following characteristics.

Multi-hop: The smart meters communicate with each other in a multi-hop fashion and form a large-scale wireless mesh network.

Location-aware: Each smart meter is aware of its own location.

Measurement capability: Each smart meter can measure the JSS using methods proposed in [17].

To describe the wireless channel characteristics, we use the log-normal shadowing model, which captures both path loss and shadowing. Let P_0 be the reference power of the source at reference distance d_0. Then, the received signal power P_r at the anchor node in `dBm` at a distance of d can be given as:

$$P_r = P_0 - 10\eta \log_{10}(\frac{d}{d_0}) + X_\sigma \tag{1}$$

where X_σ, caused by shadowing, is a Gaussian zero-mean random variable with standard deviation σ, and η is the path loss exponent and can be obtained through empirical measurements. P_0 depends on the transmit power of the source, and d_0 is typically 1 m. In a free space, η is two, and X_σ is always zero.

3.3. Problem Formulation

The jammer's motion can be described as a non-linear discrete-time controlled process that is governed by the stochastic differential equation:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}) + \mathbf{w}_t \tag{2}$$

where $\mathbf{w}_t$ is the model noise with a zero mean Gaussian distribution, i.e., $\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q})$. $f(\cdot)$ is the transition function that describes the jammer's state change. For example, if the jammer is moving with constant velocity and we use $\mathbf{x}_t = [x_t, y_t, v_{xt}, v_{yt}]^T$ to describe the jammer's state at time t, in which x_t and y_t represent the position coordinates of the jammer and v_{xt} and v_{yt} represent the linear velocity of the jammer, then the motion model will be:

$$\mathbf{x}_t = \begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{x}_{t-1} + \mathbf{w}_t \tag{3}$$

where $\mathbf{x}_t = [x_t, y_t, v_{xt}, v_{yt}]^T$ and the T is the sampling time interval.

The state of the jammer cannot be directly measured, but we can use some sensors to give observations of the jammer. The observation $\mathbf{z}_t$ can be described as:

$$\mathbf{z}_t = h(\mathbf{x}_t) + \mathbf{v}_t \tag{4}$$

where random variable $\mathbf{v}_t$ is the measurement noise with a zero mean Gaussian distribution, i.e., $\mathbf{v}_t \sim \mathcal{N}(0, \mathbf{R})$. $h(\cdot)$ is the observation function, as we assume the smart meters can measure the jamming signal strength of the jammer, and it can be described using Equation (1). For example, assume that at time t, a jammer located at (x_{J_t}, y_{J_t}) starts to transmit at the power level of P_J and that m nodes located at $\{(x_i, y_i)\}_{i \in [1,m]}$ can measure the JSS. Then, $\mathbf{z}_t = [z_t^1, z_t^2, ..., z_t^m]$, where z_t^i is the observation given by node i and can be given as:

$$\begin{aligned} z_t^i &= P_J - 10\eta \log_{10}(d_i) + \mathbf{v}_t \\ d_i &= \sqrt{(x_{J_t} - x_i)^2 + (y_{J_t} - y_i)^2} \end{aligned} \tag{5}$$

Then, the jammer localization problem can be described as estimating the jammer's position (x_{J_t}, y_{J_t}) at time t given the observation sequence $\mathbf{z}_{1:t}$, where $\mathbf{z}_{1:t}$ represents the observations from Time 1 to time t.

4. Design

`JamCatcher` divides the jammer localization process into two stages, i.e., far-field chasing stage and near-field capturing state. The far-field chasing stage is just like the situations in existing research, where measurements can be obtained from sufficient observation nodes. However, the time delay is large in this stage. `JamCatcher` uses the genetic algorithm for finding the best estimation of the jammer's location. Instead of localizing the jammer accurately, far-field stage algorithms aim at driving the tracker as close as possible to the jammer in a short time. In the near-field stage, the tracker can take measurements from limited observation nodes in real time. `JamCatcher` provides a dynamic algorithm and a static algorithm for the near-field capturing stage. The dynamic algorithm takes advantages of the jammer's mobility and gives estimate of the jammer's location based on a dynamic model that describes the jammer's motion and measurements from different time. The static algorithm is a least squares-based method that uses measurements from the current time. It provides a raw estimate of the jammer's state for the dynamic algorithm. An unscented Kalman filter-based data fusion algorithm is used to alleviate the impact of noise and give a refined estimate based on the outputs of the static algorithm and dynamic algorithm.

4.1. Workflow of JamCatcher

`JamCatcher` is a localization scheme for the tracker to cope with the AMI network nodes to localize a mobile jammer in real time. As shown in Figure 2, the workflow of `JamCatcher` can be roughly divided into three steps:

- JSS estimation: As the jammer moves in the AMI network, smart meters within the jammed area become affected nodes. Every smart meter can obtain its state by checking whether it has lost some of its neighbors; if yes, it will label itself as an affected node. Each affected node measures JSS periodically and stores it with a timestamp. Similar to received signal strength (RSS), JSS reflects the distance between the jammer and the affected node. JSS cannot be measured directly since jamming signals are mixed with signals transmitted by regular wireless devices. However, it can be estimated as the average of the ambient noise floor (ANF) [17], which is the measurement of the ambient noise. The tracker is also capable of measuring JSS; thus, the tracker itself can act as an observation node with mobility.

- Data transmission: When the tracker wants to collect JSS measurements, it broadcasts a request. On receiving the request message, each affected node sends a response to the tracker. The response message includes JSS measurement, node location, hop-count, and a timestamp. After receiving the response messages from observation nodes, the tracker judges which state it is in by checking the hop-counts from the messages. Specifically, the tracker is in the far-field chasing stage if most hop-counts are larger than one, which means the tracker has not entered the jammed area. The tracker is in the near-field capturing stage if most hop-counts are one, which means it has entered the jammed area.
- Jammer localization: The tracker chooses among the localization algorithms according to the stage i which it is. In the far-field chasing stage, the tracker is far away from the jammer and is surrounded by unaffected nodes; thus, it can obtain JSS measurements from all the smart meters on the boundary of the jammed area. However, the delay of the measurements is large due to multi-hop transmission. Instead of localizing the jammer accurately, far-field stage algorithms aim at driving the tracker as close as possible to the jammer in a short time. In the near-field capturing stage, the tracker has entered the jammed area and can collect measurements from affected nodes directly, but the number of nodes it can communicate with is limited. The near-field capturing stage can be further divided into two situations, depending on the number of nodes with which the tracker can communicate.

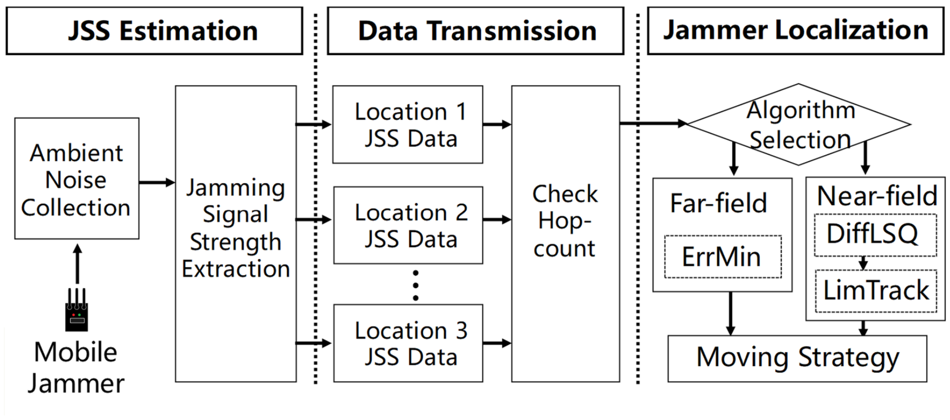


Figure 2. Workflow of JamCatcher. JSS, jamming signal strength; ErrMin, error minimizing.

As the algorithm in the far-field chasing stage is not the focus of this work, we utilized the error-minimizing jammer localization method propose by [17] (`ErrMin`) and improved their method in two aspects. First, we developed an evaluation metric that does not contain transmit power as a parameter and thus reduces the searching space prominently. Secondly, we combined the developed evaluation metric with the genetic algorithm (GA) and proposed an effective jammer localization mechanism. `ErrMin` is a non-real-time jammer localization algorithm for the tracker to move into the jammed area as soon as possible.

The algorithms in the near-field capturing stage cope with the situation when the tracker enters the jammed area and can only communicate with limited observation nodes. A static algorithm (named `DiffLSQ`), as well as a dynamic algorithm (named `LimTrack`) were developed for localizing the jammer in this stage. When the tracker can collect JSS from more than two observation nodes, the static algorithm combined with the dynamic algorithm were used for localization, and the output of the static algorithm was taken as the input of the dynamic algorithm. When the tracker can collect measurements from less than two nodes, only the dynamic algorithm was used. The details of the algorithms are described in the following subsections.

4.2. Far-Field Chasing Algorithm

The far-field chasing algorithm works as follows. For a given set of JSS measurements, it searches through the target area and evaluates each possible location. Based on the JSS measurements, the algorithm gives every candidate location a quantitative evaluation, which indicates how close it is to the true jammer location. Assume a jammer located at (x_J, y_J) starts to transmit at the power level of P_J and m nodes located at $\{(x_i, y_i)\}_{i\in[1,m]}$ become observation nodes. For a given location $(\hat{x}_J, \hat{y}_J)$, each observation node can estimate the jammer's transmit power according to Equation (1):

$$\begin{aligned}\hat{P}_{J_i} &= P_{r_i} + \hat{P}_{L_i}(\hat{d}_i)\\ \hat{P}_{L_i}(\hat{d}_i) &= 10\eta\log_{10}(\hat{d}_i)\\ \hat{d}_i &= \sqrt{(\hat{x}_J - x_i)^2 + (\hat{y}_J - y_i)^2}\end{aligned} \tag{6}$$

For a set of boundary nodes, we can get a set of estimated jammer transmit powers $\{\hat{P}_{J_1}, \hat{P}_{J_2}, ..., \hat{P}_{J_m}\}$. If the chosen location is exactly the jammer's true location and the noise is sufficiently small, we will have $\hat{P}_{J_1} = \hat{P}_{J_2} = ... = \hat{P}_{J_m}$. When the chosen location is away from the true location, the variance of $\{\hat{P}_{J_i}\}_{i\in[1,m]}$ becomes larger than zero. Thus, we use the standard deviation as the evaluation metric and map it to the interval of $[0, 1]$.

$$e_s = exp\left\{-\sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{P}_{J_i} - \hat{\bar{P}}_{J_i})^2}\right\} \tag{7}$$

where $\hat{\bar{P}}_{J_i}$ is the mean of $\{\hat{P}_{J_i}\}_{i\in[1,m]}$. Figure 3 shows the value of the metric at different locations within the rectangular grid of $[-5, 5] \times [-5, 5]$, where eight observation nodes are located over the rectangular grid of $[-3.75, 3.75] \times [-1.25, 1.25]$ with a grid size of 2.5 and the jammer located at $(0, 0)$. We observed that when the number of observation nodes decreased to under four, there was a high possibility that `ErrMin` failed to give the correct estimate since a global minimum did not exist. The performance of `ErrMin` degraded as the measurement noise increased.

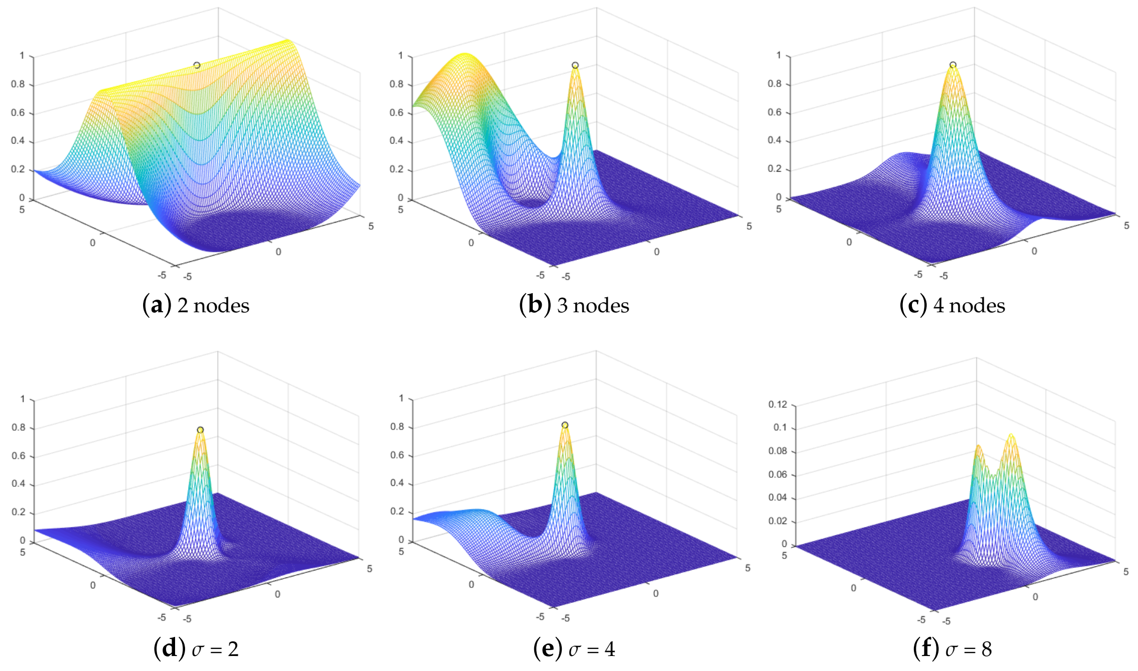


Figure 3. The value of the metric on different locations within the rectangular grid of $[-5, 5] \times [-5, 5]$.

After developing the evaluation metric, finding the best estimation is equal to finding a location that has the highest score on the metric. As a matter of fact, the jammer will always locate inside the group of affected nodes since the jammer we studied was omnidirectional. Thus, the searching space can be reduced to the circle formed by the affected nodes. We adopted GA to search for this location automatically. We mapped the locations in the jammer area into genetic representations and defined the fitness function as Equation (7). After obtaining the estimated position of the jammer, we can estimate the jammer's transmit power P_J as the mean of $\{\hat{P}_{J_i}\}_{i\in[1,m]}$, i.e., $\hat{P}_J = \frac{1}{m}\sum_{i=1}^{m}\hat{P}_{J_i}$.

4.3. Near-Field Capturing Algorithms

We developed a static algorithm and a dynamic algorithm for the near-field capturing process. The static algorithm only takes current JSS measurements as the input and estimates the jammer's location. The dynamic algorithm includes a dynamic model that describes the jammer's motion and an unscented Kalman filter (UKF)-based data fusion algorithm. It starts from an initial estimated jammer state, which is set manually, and gives an estimate using the motion model at the beginning of each time step. Then, it uses JSS measurements of the current time to revise the estimate. Since the estimated jammer location is a weighted average of the predicted location using the motion model and the new JSS measurements, it deals effectively with the uncertainty due to noisy sensor data. However, the dynamic algorithm is sensitive to the initial given state due to the strong nonlinearity of the observation function (see Equation (1)) in our jammer localization scenario. To solve this problem, we used the static algorithm to generate the initial state of the dynamic algorithm. Furthermore, we combined the two algorithms to obtain a refined estimation of the jammer's location when the number of observation nodes was more than two.

Figure 4 shows the workflow of the near-field algorithm. At each time step, the dynamic algorithm gives an estimation based on the last estimated jammer location and the dynamic model. At the same time, the static algorithm gives an estimation based on current JSS measurements. Then, the output of static algorithm, along with the current JSS measurements are used as the inputs of the UKF, and the output is the current estimation of the jammer's location. When the observation nodes are less than two, only the dynamic algorithm is utilized to localize the jammer.

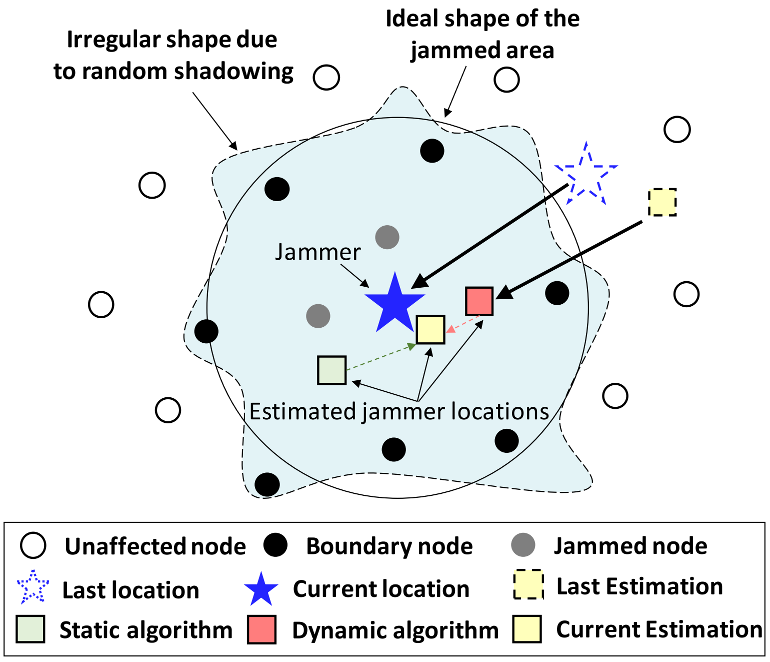


Figure 4. Illustration of near-field capturing algorithms.

4.3.1. Static Algorithm (DiffLSQ)

We propose a least-squares method `DiffLSQ` to localize the jammer when the tracker can collect JSS from more than two affected nodes. Then, based on the log-normal shadowing model, the measured JSS at affected node i located at (x_i, y_i) can be written as:

$$P_{r_i} = P_J - 10\eta \log_{10}(d_i) + X_\sigma \tag{8}$$

where d_i is the distance between affected node i and the jammer. As the transmit power of jammer P_J can be estimated in the far-field chasing stage using `ErrMin`, d_i can be estimated as:

$$\hat{d}_i = 10^{\frac{\hat{P}_J - P_{r_i}}{10\eta}} \tag{9}$$

Therefore, we can obtain the following formula,

$$(\hat{x}_J - x_i)^2 + (\hat{y}_J - y_i)^2 = \hat{d}_i^2 \tag{10}$$

where (x_i, y_i) is the position of affected node i and $(\hat{x}_J, \hat{y}_J)$ is the position of the jammer.

In the above equation, the unknown variables include $\hat{x}_J$ and $\hat{y}_J$. Suppose at time t that the tracker can collect JSS from n nodes; we can have n equations:

$$\begin{aligned} (x_1 - \hat{x}_{J_t})^2 + (y_1 - \hat{y}_{J_t})^2 &= \hat{d}_{1,t}^2 \\ (x_2 - \hat{x}_{J_t})^2 + (y_2 - \hat{y}_{J_t})^2 &= \hat{d}_{2,t}^2 \\ &\vdots \\ (x_n - \hat{x}_{J_t})^2 + (y_n - \hat{y}_{J_t})^2 &= \hat{d}_{n,t}^2 \end{aligned} \tag{11}$$

To avoid solving complicated nonlinear equations, we first linearized the problem by subtracting the nth equation from both sides of the first $n-1$ equations and obtained linear equations:

$$\begin{aligned} &(x_1^2 - x_n^2) - 2(x_1 - x_n)\hat{x}_{J_t} + (y_1^2 - y_n^2) - 2(y_1 - y_n)\hat{y}_{J_t} \\ &\quad = \hat{d}_{1,t}^2 - \hat{d}_{n,t}^2 \\ &(x_2^2 - x_n^2) - 2(x_2 - x_n)\hat{x}_{J_t} + (y_2^2 - y_n^2) - 2(y_2 - y_n)\hat{y}_{J_t} \\ &\quad = \hat{d}_{2,t}^2 - \hat{d}_{n,t}^2 \\ &\vdots \\ &(x_{n-1}^2 - x_n^2) - 2(x_{n-1} - x_n)\hat{x}_{J_t} + (y_{n-1}^2 - y_n^2) - 2(y_{n-1} - y_n)\hat{y}_{J_t} \\ &\quad = \hat{d}_{n-1,t}^2 - \hat{d}_{n,t}^2 \end{aligned} \tag{12}$$

Then, this can be written in the form of $\mathbf{A}\mathbf{x} = \mathbf{b}$ with:

$$\mathbf{A} = \begin{pmatrix} x_1 - x_n & y_1 - y_n \\ \vdots & \vdots \\ x_{n-1} - x_n & y_{n-1} - y_n \end{pmatrix}$$

and:

$$\mathbf{b} = \frac{1}{2} \begin{pmatrix} (x_1^2 - x_n^2) + (y_1^2 - y_n^2) - (\hat{d}_{1,t}^2 - \hat{d}_{n,t}^2) \\ \vdots \\ (x_{n-1}^2 - x_n^2) + (y_{n-1}^2 - y_n^2) - (\hat{d}_{n-1,t}^2 - \hat{d}_{n,t}^2) \end{pmatrix}$$

We can estimate the location of the jammer at time t using the least squares (LSQ) method,

$$\mathbf{x} = [\hat{x}_{J_{t_m}}, \hat{y}_{J_{t_m}}]^T = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{13}$$

From the above, we can infer that in order to solve the problem, we need at least three equations. Because the tracker itself is also able to estimate JSS through ambient noise measurement, the tracker needs to collect JSS from at least two affected nodes.

4.3.2. Dynamic Algorithm (LimTrack)

The dynamic algorithm contains two parts. The first part is a motion model, which describes the jammer's motion. The second part is a UKF based data fusion algorithm that produces an estimate of the state of the jammer as an average of the model's predicted state and of the new measurements using a weighted average.

The jammer's motion is usually complicated in AMI networks. For example, the attacker carrying a jammer on his/her car may change the driving direction at any crossing on the street. Thus, it is not applicable to simply treat the jammer's speed as a constant variable. To model the jammer's motion, we utilized the interacting multiple model (IMM) [39]. The IMM algorithm selects multiple parallel running models and then automatically switches between the models according to the Markov transition probability matrix. As a matter of fact, the jammer's motion is usually bounded by the road it moves along, which means the jammer is either in a linear motion (when the road is straight) or in a turning motion (when there is a turn). Thus, we used a constant velocity (CV) model and a coordinate turn (CT) model to describe the jammer's motion. The CV model is defined as:

$$\mathbf{x}_t = \begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{x}_{t-1} + \mathbf{w}_t \tag{14}$$

where $\mathbf{x}_t = [x_t, y_t, v_{xt}, v_{yt}]^T$ and the T is the sampling time interval. The CT model is:

$$\mathbf{x}_t = \begin{pmatrix} 1 & 0 & \frac{\sin(\omega T)}{\omega} & \frac{\cos(\omega T)-1}{\omega} \\ 0 & 1 & \frac{1-\cos(\omega T)}{\omega} & \frac{\sin(\omega T)}{\omega} \\ 0 & 0 & \cos(\omega T) & \sin(\omega T) \\ 0 & 0 & -\sin(\omega T) & \cos(\omega T) \end{pmatrix} \mathbf{x}_{t-1} + \mathbf{w}_t \tag{15}$$

where ω is the angular velocity (rad/s).

Figure 5 shows the framework of the IMM algorithm. In the input state, elemental filters (models) interact with one another by utilizing probabilistically-weighted sums of the most recent estimates from all elemental filters as their inputs. Then, the weights of elemental filters are updated based on the outputs of each filter. The output is also the probabilistically-weighted sums of the outputs from all elemental filters.

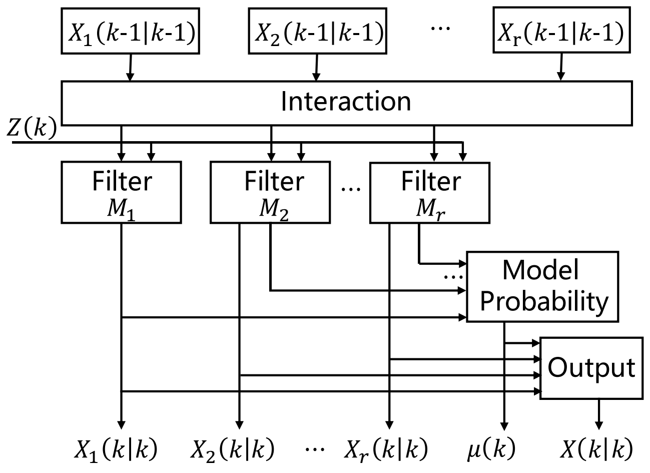


Figure 5. Framework of the interacting multiple model (IMM) algorithm.

The UKF addresses the general problem of estimating the state $\mathbf{x}$ of a non-linear discrete-time controlled process that is governed by the stochastic differential equation given in Equation (2) with an observation $\mathbf{z}$ given in Equation (4). The motion of the jammer is described using the IMM. The initial estimated state of the jammer is obtained from the static algorithm. The observation function in our algorithm contains two parts, i.e., $\mathbf{Z}_k = [\mathbf{z}_t^1, \mathbf{z}_t^2]^T$, where $\mathbf{z}_t^1 = [\hat{x}_t, \hat{y}_t]^T$ is the output of the static algorithm. As the state of the jammer is $\mathbf{x}_t = [x_t, y_t, v_{xt}, v_{yt}]^T$, $\mathbf{z}_t^1$ can be written as:

$$\mathbf{z}_t^1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{x}_t \tag{16}$$

The observation function $\mathbf{z}_t^2$ is a set of JSS measurements from m observation nodes, i.e.,

$$\mathbf{z}_t^2 = [P_{r_1}(\mathbf{x}_t), P_{r_2}(\mathbf{x}_t), ..., P_{r_m}(\mathbf{x}_t)]^T \tag{17}$$

For an observation node located at (x_{oi}, y_{oi}), $P_{r_i}(\mathbf{x}_t)$ is defined as:

$$P_{r_i}(\mathbf{x}_t) = P_J - 10\eta \log_{10}(\sqrt{(x_t - x_{oi})^2 + (y_t - y_{oi})^2}) \tag{18}$$

The UKF uses a deterministic sampling technique called unscented transform to select a minimal set of sample points (i.e., sigma points) around the mean. Each sigma point represents an estimation of the true state and is propagated through the non-linear functions. Then, the new estimation is given as the weighted mean of the sigma points. The details are given as follows:

1. State initiation:

$$\begin{aligned} \hat{\mathbf{x}}_0 &= \mathbb{E}[\mathbf{x}_0] \\ \mathbf{P}_0 &= \mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T] \end{aligned} \tag{19}$$

   where $\hat{\mathbf{x}}_0$ is the initial estimated jammer state (given by `ErrMin`) and $\mathbf{P}_0$ is the initial error covariance matrix.

2. Sigma points' calculation:

$$\begin{cases} \mathcal{X}_{0,t-1} = \hat{\mathbf{x}}_{t-1}, & \omega_0 = \lambda/(L+\lambda) \\ \mathcal{X}_{i,t-1} = \hat{\mathbf{x}}_{t-1} + (\sqrt{(L+\lambda)\mathbf{P}_{t-1}})_i, & \omega_i = 0.5/(L+\lambda) \\ \mathcal{X}_{i+L,t-1} = \hat{\mathbf{x}}_{t-1} - (\sqrt{(L+\lambda)\mathbf{P}_{t-1}})_i, & \omega_{i+L} = 0.5/(L+\lambda) \end{cases} \tag{20}$$

   where $i = 1, 2, ..., L$, L is the dimension of state $\mathbf{x}$, λ is a parameter used to adjust ω, and $\mathbf{P}_t$ is the a posteriori error covariance matrix at time $t-1$.

3. State update:

$$\begin{aligned} &\mathcal{X}_{i,t|t-1} = f(\mathcal{X}_{i,t-1}) \\ &\hat{\mathbf{x}}_{t|t-1} = \sum_{i=0}^{2L} \omega_i \mathcal{X}_{i,t|t-1} \\ &\mathbf{P}_{t|t-1} = \sum_{i=0}^{2L} \omega_i [\mathcal{X}_{i,t|t-1} - \hat{\mathbf{x}}_{t|t-1}][\mathcal{X}_{i,t|t-1} - \hat{\mathbf{x}}_{t|t-1}]^T + \mathbf{Q}_t \\ &\mathcal{Y} = h(\mathcal{X}_{i,t|t-1}) \\ &\hat{\mathbf{y}}_{t|t-1} = \sum_{i=0}^{2L} \omega_i \mathcal{Y}_{i,t|t-1} \end{aligned} \tag{21}$$

4. Measurement update:

$$\begin{aligned}
\mathbf{P}_{yy} &= \sum_{i=0}^{2L} \omega_i [\mathcal{Y}_{i,t|t-1} - \hat{\mathbf{y}}_{t|t-1}][\mathcal{Y}_{i,t|t-1} - \hat{\mathbf{y}}_{t|t-1}]^T + \mathbf{R}_t \\
\mathbf{P}_{xy} &= \sum_{i=0}^{2L} \omega_i [\mathcal{X}_{i,t|t-1} - \hat{\mathbf{x}}_{t|t-1}][\mathcal{Y}_{i,t|t-1} - \hat{\mathbf{y}}_{t|t-1}]^T \\
\mathbf{K}_t &= \mathbf{P}_{xy} \mathbf{P}_{yy}^{-1} \\
\hat{\mathbf{x}}_t &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t (\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}) \\
\mathbf{P}_t &= \mathbf{P}_{t|t-1} - \mathbf{K}_t \mathbf{P}_{yy} \mathbf{K}_t^T
\end{aligned} \tag{22}$$

The output of UKF is an a posteriori state estimate $\hat{\mathbf{x}}_t$ of the jammer and an a posteriori error covariance matrix $\mathbf{P}_t$, which is a measure of the estimated accuracy of the state estimate.

5. Evaluation

We evaluated `JamCatcher` by conducting both a real-world experiment and MATLAB simulations. The data collected from a real-world experiment were used for evaluating both the far-field chasing algorithm and near-field capturing algorithm. To evaluate `JamCatcher` in a larger scale scenario, we conducted extensive MATLAB simulations. As the far-field algorithm is not the focus of this paper, we only evaluated near-field algorithms in the simulations. The performance of `JamCatcher` was compared to some existing jammer localization algorithms. However, most existing algorithms cannot localize the jammer with limited observation nodes (e.g., less than four). Thus, we chose adaptive LSQ (or `ALSQ`) [16] and `ErrMin` [17] for comparison, as they require less observation nodes compared to other existing methods. Note that `ALSQ` and `ErrMin` are not applicable to the near-field capturing stage where observation nodes could be less than two, so we relaxed the conditions and allowed them to take measurements from more observation nodes compared to `JamCatcher` (details are given in the following experiments). The results of `ALSQ` and `ErrMin` were only used as a reference.

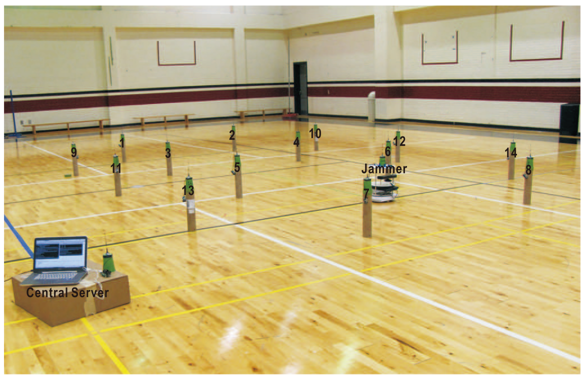
5.1. Real-World Experiments

5.1.1. Data Collection

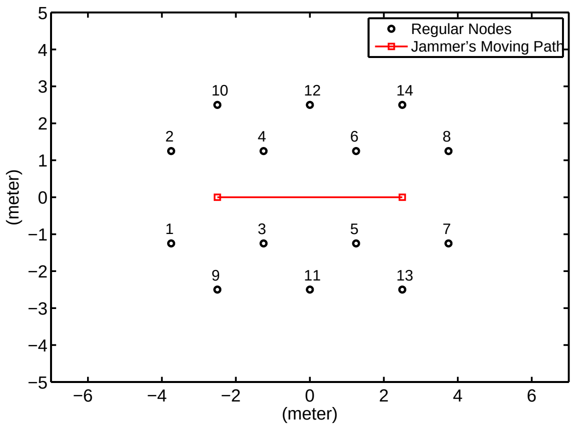
We conducted all experiments in a 20-by-30 m gymnasium. Our experiment involved 14 nodes and a mobile jammer, which were implemented on MicaZ nodes. The deployment layout is illustrated in Figure 6; each node measured the jamming signal strength (JSS) once every second and stored the measured JSS and its timestamp in a local memory. A central server was implemented on a laptop. Connected with a sink node, the server collected the data from all nodes. We mounted one constant jammer on a robot that moved back and forth on a line between $[-2.5, 0]$ and $[2.5, 0]$. In particular, the constant jammer traveled from $[-2.5, 0]$ to $[2.5, 0]$ 11 times and from $[2.5, 0]$ to $[-2.5, 0]$ 11 times. In total, we collected JSS when the jammer was at many different locations in the 22 independent experiments. Additionally, we recorded the true location of the jammer in real time by setting up a surveillance video camera overlooking the entire network field. Given the collected JSS measurements and their timestamps, we estimated the jammer's locations utilizing our proposed localization algorithms.

The collected data were used for both the far-field chasing algorithm and the near-field capturing algorithm. Instead of making a real tracker, we assumed there was a virtual tracker that could collect data from the 14 nodes and localize the jammer using the JSS measurements. For far-field chasing algorithm, all the data collected from 14 nodes were used. We compared the performance of far-field chasing algorithm to `ALSQ` and the original `ErrMin` proposed in [17], which used the simulated annealing (SA) algorithm to estimate the jammer's location. For the near-field capturing algorithm, we manually chose observation nodes from the 14 nodes at different times to create situations when there were less than four nodes. Noting that existing methods failed to estimate the jammer's location when the number of measurements was less than for, for comparison, we fixed the number of observation nodes to eight for the other methods and six at most for `JamCatcher`.

To evaluate the accuracy of our localization process, we show the Euclidean distance between the estimated jammer's location and the true jammer's location. We also used the root mean squared error (RMSE) to analyze the average localization error. We present the cumulation distribution functions (CDF) of the localization error to show the statistical characteristics.



(**a**) Snapshot of the gym room

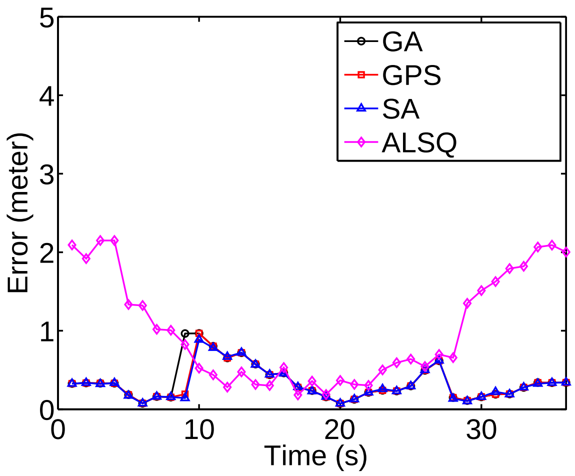


(**b**) Coordinates of the nodes

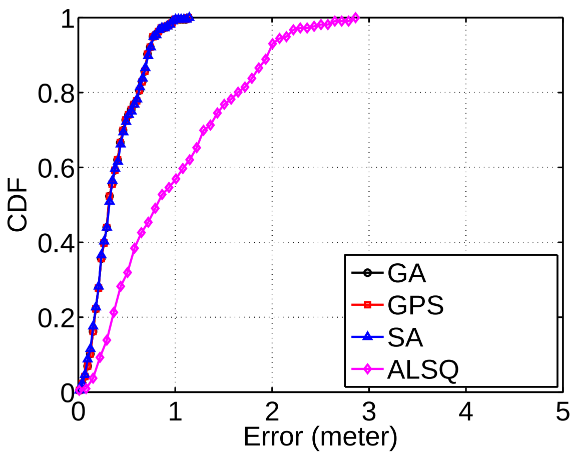
Figure 6. Network deployment in an indoor gym room.

5.1.2. Performance Evaluation

The performance of our error-minimizing-based far-field chasing algorithm is shown in Figure 7, which is compared to the performance of `ALSQ` and the original `ErrMin` (referred to as SA in Figure 7). For our far-field chasing algorithm, we tested two different searching algorithms, i.e., GA and generalized pattern search (GPS). Figure 7a shows localization errors as the constant jammer moved in a straight line between $[-2.5, 0]$ and $[2.5, 0]$. All error-minimizing framework-based algorithms achieved estimation errors less than one meter all the time, while the estimation errors of `ALSQ` were more than 2 m when the mobile jammer was close to $[-2.5, 0]$ and $[2.5, 0]$. Figure 7b depicts the cumulative distribution function of the estimation errors. We observed that the error-minimizing-based method achieved a similar accuracy, and all of them constantly outperformed the `ALSQ` method in terms of RMSE (from 0.65 m to 0.20 m).



(**a**) Average estimation errors vs. time



(**b**) Cumulative distribution function

Figure 7. Performance of the far-field chasing algorithm. GPS, generalized pattern search.

The path loss exponent η is a parameter only related to the environment and can be obtained from empirical measurements. When the value of η used in our algorithm is set to be different from the true value obtained through experiments (i.e., $\eta = 2.11$), the estimated jammer location may deviate from the true location. To evaluate the impact of errors in the path loss exponent, we set η to different values from 2.0 to 3.0, and the result is shown in Table 1. When η was set to be 3.00, which is the most distant value from the true value, the maximum estimation error (RMSE of 0.31 m) occurred. However, compared to the reference estimation error (RMSE of 0.20 m when η was set to 2.11), it can be seen that the deviation in η had a minor influence on the accuracy of our far-field chasing algorithm.

Table 1. Estimation error (meter) comparison of all algorithms with different path loss exponents.

	GA	GPS	SA	ALSQ
$\eta = 2.00$	0.23	0.23	0.23	0.65
$\eta = 2.25$	0.21	0.20	0.20	0.65
$\eta = 2.50$	0.24	0.22	0.21	0.66
$\eta = 2.75$	0.27	0.24	0.25	0.67
$\eta = 3.00$	0.31	0.29	0.29	0.68

The performance of our near-field capturing algorithm was compared to the original `ErrMin` [17]. For `ErrMin`, we used JSS measurements from eight observation nodes, located over the rectangular grid of $[-3.75, 3.75] \times [-1.25, 1.25]$ with a grid size of 2.5 m. For `JamCatcher`, six nodes located on the rectangular grid of $[-2.5, 2.5] \times [-1.25, 1.25]$ were used at the beginning, and after every five seconds, one node was removed from the observation node set manually. The initial state of the near-field algorithm of `JamCatcher` was obtained using the far-field algorithm. Results show that the RMSE of `JamCatcher` was 0.27 m, which outperformed `ErrMin` with an RMSE of 0.31 m, even though the average number of observation nodes of `JamCatcher` was only half as much as `ErrMin`.

To evaluate the influence of the path loss component η on the accuracy of the near-field algorithm, we also changed it from 2.00 to 3.00 and calculated the RMSEs of the estimation errors. Figure 8 shows the results. We can see that the value of the path loss component had a minor influence on our near-field algorithm, which always outperformed the `ErrMin` algorithm. `ErrMin` performed similarly when η was set to be under 2.50 (the true value was 2.11). However, when η was set to be larger than 2.50, we found a dramatic increase in the RMSEs (from an average of 0.4 m when η was under 2.50 to an average of 1.3 m). The reason is that `ErrMin` used searching algorithms to find the location with the highest value on the evaluation metric as the estimated jammer location. As the observation nodes were of limited number and η deviated from the true value, there was a high possibility that other locations had a higher value on the evaluation metric than the true jammer location. In the case of `JamCatcher`, the impact of errors in η was alleviated using the dynamic model of the jammer's motion.

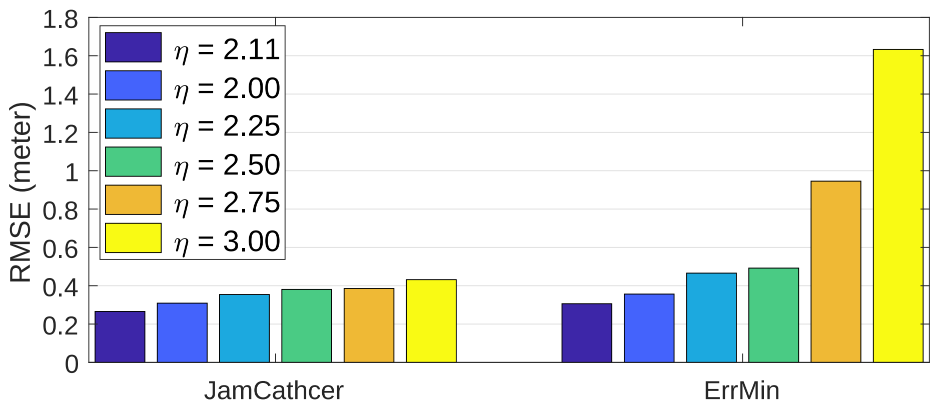


Figure 8. Impact of estimation error in η on the performance of `JamCatcher` and ErrMin.

5.2. MATLAB Simulations

To evaluate `JamCatcher` in a a larger scale scenario, we conducted extensive MATLAB simulations. In the simulations, we focused on the accuracy of the near-field algorithm of `JamCatcher`, which was the main contribution of this paper, and compared the performance of `JamCatcher` to `ALSQ` and `ErrMin`. For existing methods to work, we assumed a system adopting `ALSQ` or `ErrMin` can obtain JSS measurements from all the boundary nodes and ignore the time delay caused by multi-hop transmission. As for `JamCatcher`, we set the maximum communication of the tracker to be 60 m. The tracker only collected JSS measurements from nodes within its communication range instead of using JSS measurements from all the boundary nodes. Thus, the number of observation nodes for `JamCatcher` was always smaller than that for `ALSQ` and `ErrMin` at any time step.

5.2.1. Experimental Setup

The parameters of the simulation are shown in Table 2. We simulated a jammer localization process in a 500-by-500 m square AMI network, and the distance between two neighbor smart meters was 40 m. For simplicity, the smart meters were set on the grid points, e.g., $(0,0)$, $(40,0)$, and $(40,40)$, etc. Note that the node density in our simulation (144 nodes in a 500-by-500 m squared area) was much smaller than that in `ALSQ` [16] or `ErrMin` [17] (200 nodes in a 300-by-300 m squared area). The path loss exponent η of the shadowing model was set as 2.40, and the standard deviation of the random attenuation σ was set to be 2.0 in order to simulate a highly-irregular radio environment. We set the power of ambient noise as -65 dBm, and when the measured JSS on a smart meter was less than -65 dBm, we considered the smart meter an unaffected node. Thus, the jamming range of the jammer was around 80 m. We set an SNR threshold of -3 dBm; when the SNR was under the threshold, the communicating parties would not be able to receive messages from each other.

Table 2. Definition of the parameters in the simulation.

Parameter	Meaning	Value
L	Size of AMI Network	500 m
ND	Neighbor Distance	40 m
P_J	Transmit Power of Jammer	100 mW
P_T	Transmit Power of Smart Meter	50 mW
G	Gain of the Antenna	1
P_N	Power of Ambient Noise	-65 dBm
f_T	Signal Frequency	2.4 GHz
η	Path Loss Exponent	2.40
γ_0	SNR Threshold	-3 dB
d_T	Tracker's Communication Range	60 m
d_J	Jammer's Jamming Range	80 m

The simulation process was done in following steps. The jammer started from the initial location $(150,0)$ and moved at a linear velocity of 5 m/s and a angular velocity of 0.0314 rad/s. The tracker started from $(100,0)$ and kept moving towards the jammer with a linear speed of 6 m/s. At each time step, the mobile jammer created a jammed area within which affected nodes (including jammed nodes and boundary nodes) were located. All the boundary nodes were chosen as observation nodes for `ALSQ` and `ErrMin`. As for `JamCatcher`, we calculated the SNR for each affected node located within the tracker's communication range. If the SNR was larger than the threshold, the affected node was considered an observation node. Each simulation lasted for 200 time steps and was repeated 20 times, then we calculated the RMSEs of each algorithm.

5.2.2. Performance Comparison and Results Analysis

The RMSEs of `JamCatcher`, `ErrMin`, and `ALSQ` were 5.36, 6.50, and 20.5 m, respectively. Figure 9 shows the CDFs of the localization errors of the three methods. We can see that `JamCatcher` achieved similar performance with `ErrMin`, and all of them outperformed `ALSQ` evidently. Note that although `JamCatcher` and `ErrMin` achieved similar performances, the number of observation nodes for `JamCatcher` (average of four nodes) was much less than that of `ALSQ` and `ErrMin` (average of 16 nodes). Thus, our proposed method is more effective and accurate with a limited number of observation nodes.

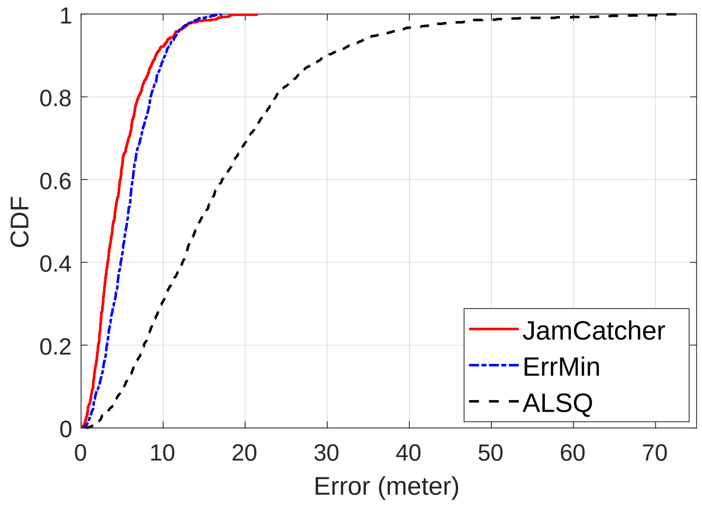


Figure 9. CDF of the localization errors of `JamCatcher`, ErrMin, and `ALSQ`.

5.2.3. Impact of Node Density

To evaluate the impact of node density, we changed the neighbor distance (*ND*) of the smart meters from 10 m to 40 m and calculated the RMSEs of the three algorithm. The results are shown in Figure 10a. When *ND* was set as 10 m, all three methods achieved their best performance, and the RMSEs of the three methods were 2.06, 2.20, and 9.74 m, respectively. As *ND* increased, the number of observation nodes decreased and the performance of each algorithm degraded, respectively. However, in every situation, JamCatcher and `ErrMin` achieved similar performance, and they always outperformed `ALSQ`.

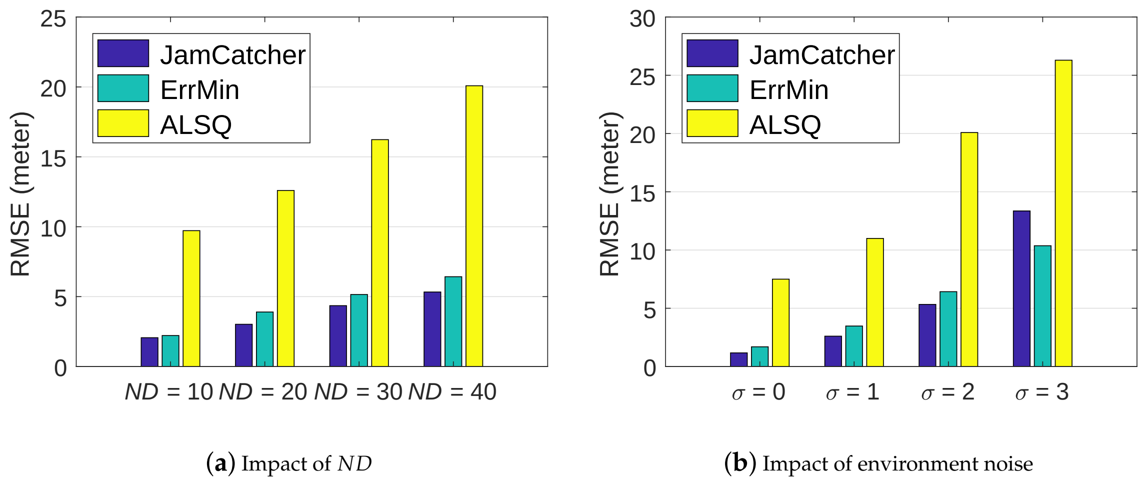


Figure 10. Impact of *ND* and σ on the performance of `JamCatcher`, `ErrMin` and `ALSQ`.

5.2.4. Impact of the Shadowing Effect

We set the value of the standard deviation of the random attenuation σ in the shadowing model from zero to three to evaluate the impact of the shadowing effect. Figure 10b shows the RMSEs of the three methods. The first observation is that when the σ was set as zero, which means no shadowing effect at all, the RMSEs of the three methods were still larger than zero, and the RMSE of `ALSQ` (7.9 m) was much larger than that of `JamCatcher` (0.42 m) and `ErrMin` (0.35 m). The reason is that estimation error of `ALSQ` had a positive correlation with the neighbor distance. As the neighbor distance in our simulation was large, i.e., 40 m, the estimation error of `ALSQ` reached 7.9 m even if there were no noise in the measurements. For `ErrMin`, it utilized a searching algorithm to search for the jammer's location, by increasing the iteration times, the estimation error can converge to zero. As for `JamCatcher`, the estimation error exited due to the model error in our algorithm. The second observation was that as σ's value increased, the RMSEs of the three methods increased, as expected. In any situation, `JamCatcher` and `ErrMin` achieved similar performance, and they all outperformed `ALSQ`.

5.2.5. Tracking Performance in Different Routes

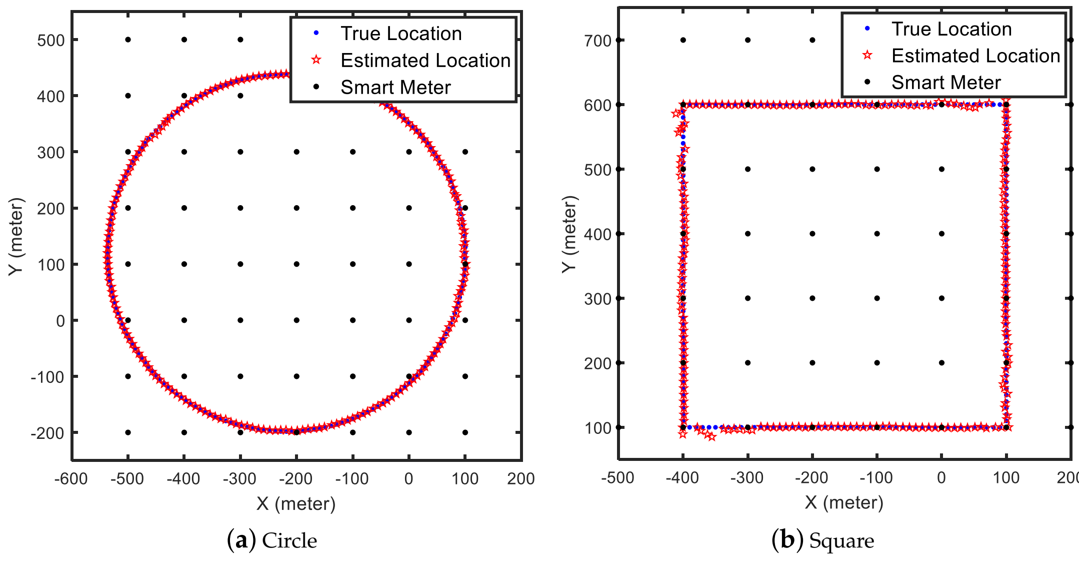
To evaluate the performance of `JamCatcher` in localizing jammers with different mobilities, we designed three moving routes for the jammer to move along (shown in Figure 11a–c). The first route was a circle, and the jammer moved with a constant velocity (10 m per second) and constant angular velocity (1.8 degrees per second). The second route was a square, where the jammer moved at a constant linear velocity (10 m per second) and zero angular velocity at each edge. The third route was a sinusoid, where the jammer moved with a changing linear velocity (maximum speed of 6 m per second) and changing angular velocity (maximum speed of four degrees per second). The neighbor distance ND of smart meters was 100 m. We set the path loss component η in the shadowing model as 2.11 and the standard deviation of the random attenuation σ to be 1.0. The maximum moving speed of the tracker was 20 m/s.

Figure 11d shows the CDFs of localization errors in three routes. The RMSEs of the three routes were 1.3, 4.0, and 4.6 m, respectively. We observed that `JamCatcher` performed better in the situation of Route 1, which was a circle. More than 90% of the estimation error was lower than 2 m. In the square route, the largest estimation error occurred in the four corners due to the sudden change in the direction of the jammer's velocity. Among the three routes, `JamCatcher` performed worst in Route 3, since the speed of the jammer was changing frequently and the dynamic model in `JamCatcher` can only catch this change to some extent. Still, 90% of the estimation error was lower than about 6.5 m.

We changed the parameter of the shadowing model to see the impact of the shadowingeffects on `JamCatcher`, and Table 3 shows the results. In every condition, `JamCatcher` always performed best in Route 1, since the jammer's velocity was stable. With the increase of σ's value, the RMSEs of three routes increased by about 6 m. However, the RMSEs always stayed within 11 percent of the grid size, which was 100 m. When changing η to different values, we found that while the RMSEs of Route 1 and Route 2 were not sensitive to the inaccuracy in η (RMSEs changed by about 3 m), the RMSE of Route 3 changed evidently (from 9.87 m to 20.72 m). This was also caused by the frequent changing of the jammer's speed.

Table 3. RMSE (meters) in the three routes with different noise levels and different path loss exponents.

Route	σ				η				
	0	**1**	**2**	**3**	**2.00**	**2.25**	**2.50**	**2.75**	**3.00**
1	0.66	1.90	3.41	5.38	2.21	2.32	3.17	4.16	5.35
2	4.78	5.43	8.10	10.92	5.82	6.03	6.48	7.12	7.40
3	3.52	4.79	8.28	10.77	9.87	9.94	16.15	19.46	20.72



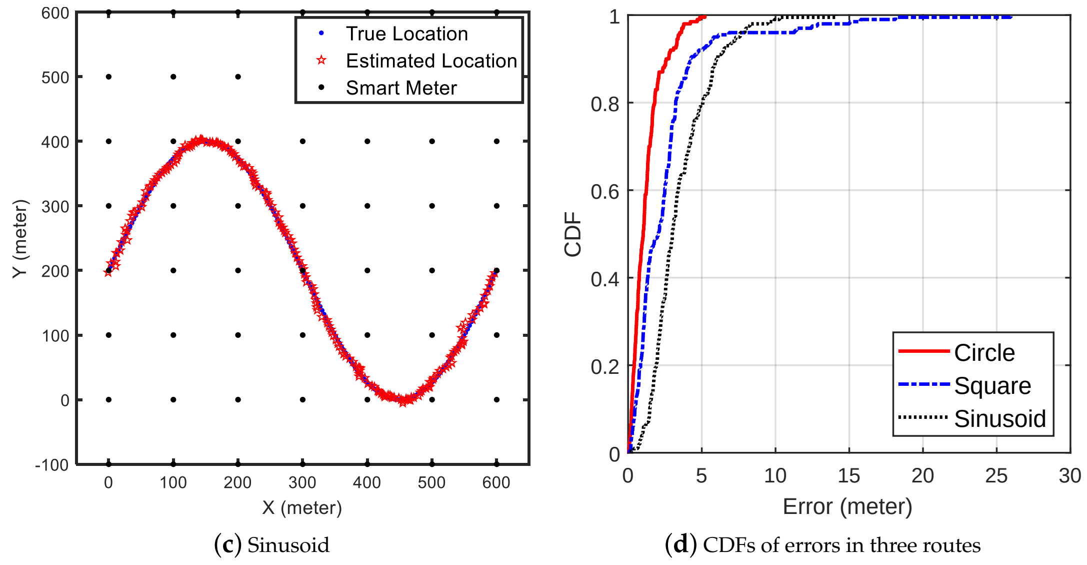
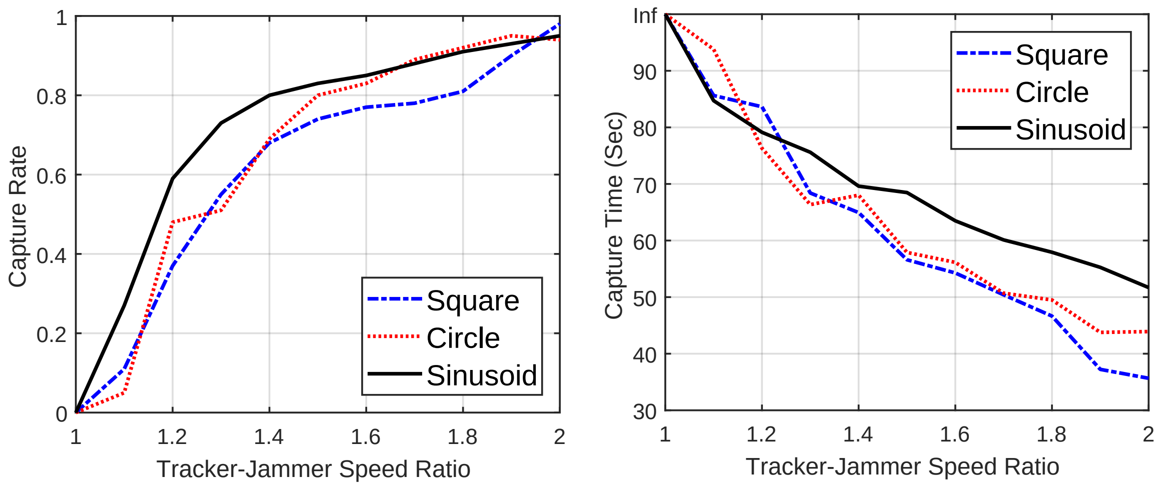


Figure 11. Simulation of three different trajectories the jammer moved along.

5.2.6. Capturing the Mobile Jammer

Based on the simulation setup in the three different routes, we also wanted to know how fast the tracker could catch the jammer and what was the minimum speed for the tracker. To do this, we defined a capture range of 10 m, and if the jammer stayed within a capture range longer than 10 s, we considered it a successful capture event. We defined the capture time as the time that the first capture event happens in the simulation. To find the minimum tracking speed, we set the tracker-jammer speed ratio value from 1.0 to 2.0. For example, in Route 1, the maximum speed of the jammer was 10 m/s; thus, we changed the speed of tracker from 10 m/s to 20 m/s. For each capture simulation, it stopped whether the simulation time reached 100 s or a capture event happened, and we repeated each simulation 100 times. We further defined a capture rate as the proportion of the times capture events happened within the 100 simulations.

The initial positions of the jammer in three routes were $(100, 100)$, $(100, 100)$, and $(0, 200)$, respectively. Since in each case, the tracker always started from $(0, 0)$, the initial distances between the jammer and tracker were 141, 141 and 200 m, respectively. Figure 12 shows how the tracker-jammer speed ratio affected the capture rate (Figure 12a) and average capture time (Figure 12b). When the value of the ratio was 1.0, the tracker was not able to catch the jammer since the distance would not decrease over time. As the speed ratio increased, the capture rate of all three routes increased and the capture time decreased. When the speed ratio was 2.0, the average capture time of the three routes was 35.66 s, 43.91 s, and 34.71 s, respectively. We also observed that the tracker's speed did not have to be twice as much as the jammer's speed; when the speed ratio was larger than 1.5, the capture rate of all three routes exceeded 75%.



(**a**) Impact on the capture time (**b**) Impact on the capture rate

Figure 12. The impact of the tracker-jammer speed ratio.

6. Conclusions

With the development of UAVs, it will be possible to deploy a tracker (a drone or a mobile robot) in an AMI network to defend the smart meters from jamming attacks without human intervention. `JamCatcher` provides this kind of technology with a mobile jammer localization framework. `JamCatcher` divides the jammer localization process in AMI networks into two stages, i.e., the far-field chasing stage and near-field capturing stage, and provides algorithms for the corresponding stages. We evaluated algorithms in `JamCatcher` in both a real-world experiment and MATLAB simulations. The results show that `JamCatcher` outperformed existing jammer localization algorithms and could localize the mobile jammer effectively with limited observation nodes in a complex environment.

Author Contributions: T.Z. and X.J. proposed the idea, performed the experiments and wrote this paper. T.Z. and Z.Z. were involved in the mathematical developments and carried out the computer simulations of this paper. W.X. offered overall guidances on the model building and revised the manuscript.

Funding: This work is supported by China NSFC Grant 61702451, ZJNSF Grant LGG19F020020.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Meng, W.; Wang, X. Distributed Energy Management in Smart Grid with Wind Power and Temporally Coupled Constraints. *IEEE Trans. Ind. Electron.* **2017**, *64*, 6052–6062. [CrossRef]
2. Gungor, V.C.; Sahin, D.; Koçak, T.; Ergüt, S.; Buccella, C.; Cecati, C.; Hancke, G.P. A Survey on Smart Grid Potential Applications and Communication Requirements. *IEEE Trans. Ind. Inform.* **2013**, *9*, 28–42. [CrossRef]
3. Mohassel, R.R.; Fung, A.S.; Mohammadi, F.; Raahemifar, K. A survey on advanced metering infrastructure and its application in Smart Grids. In Proceedings of the IEEE 27th Canadian Conference on Electrical and Computer Engineering, Toronto, ON, Canada, 4–7 May 2014; pp. 1–8.

4. Li, H.; Han, Z. Manipulating the electricity power market via jamming the price signaling in smart grid. In Proceedings of the Global Communications Conference—Workshops (GC Workshops 2011), Houston, TX, USA, 5–9 December 2011; pp. 1168–1172.
5. Liu, Y.; Hu, S.; Ho, T. Leveraging Strategic Detection Techniques for Smart Home Pricing Cyberattacks. *IEEE Trans. Dependable Secur. Comput.* **2016**, *13*, 220–235. [CrossRef]
6. Ma, J.; Liu, Y.; Song, L.; Han, Z. Multiact Dynamic Game Strategy for Jamming Attack in Electricity Market. *IEEE Trans. Smart Grid* **2015**, *6*, 2273–2282. [CrossRef]
7. Guan, Y.; Ge, X. Distributed Attack Detection and Secure Estimation of Networked Cyber-Physical Systems Against False Data Injection Attacks and Jamming Attacks. *IEEE Trans. Signal Inf. Process. Over Netw.* **2018**, *4*, 48–59. [CrossRef]
8. Liu, H.; Chen, Y.; Chuah, M.C.; Yang, J.; Poor, H.V. Enabling Self-Healing Smart Grid Through Jamming Resilient Local Controller Switching. *IEEE Trans. Dependable Secur. Comput.* **2017**, *14*, 377–391. [CrossRef]
9. Anwar, A.; Mahmood, A.N.; Tari, Z. Identification of vulnerable node clusters against false data injection attack in an AMI based Smart Grid. *Inf. Syst.* **2015**, *53*, 201–212. [CrossRef]
10. Zhou, Z.; Zhang, C.; Xu, C.; Xiong, F.; Zhang, Y.; Umer, T. Energy-Efficient Industrial Internet of UAVs for Power Line Inspection in Smart Grid. *IEEE Trans. Ind. Inf.* **2018**, *14*, 2705–2714. [CrossRef]
11. Byambasuren, B.; Kim, D.H.; Oyun-Erdene, M.; Bold, C.; Yura, J. Inspection Robot Based Mobile Sensing and Power Line Tracking for Smart Grid. *Sensors* **2016**, *16*, 250. [CrossRef]
12. Motlagh, N.H.; Taleb, T.; Arouk, O. Low-Altitude Unmanned Aerial Vehicles-Based Internet of Things Services: Comprehensive Survey and Future Perspectives. *IEEE Internet Things J.* **2016**, *3*, 899–922. [CrossRef]
13. Lim, G.J.; Kim, S.J.; Cho, J.; Gong, Y.; Khodaei, A. Multi-UAV Pre-Positioning and Routing for Power Network Damage Assessment. *IEEE Trans. Smart Grid* **2018**, *9*, 3643–3651. [CrossRef]
14. Whitehouse, K.; Karlof, C.; Culler, D.E. A practical evaluation of radio signal strength for ranging-based localization. *Mob. Comput. Commun. Rev.* **2007**, *11*, 41–52. [CrossRef]
15. Blumenthal, J.; Grossmann, R.; Golatowski, F.; Timmermann, D. Weighted Centroid Localization in Zigbee-based Sensor Networks. In Proceedings of the IEEE International Symposium on Intelligent Signal Processing, Alcala De Henares, Spain, 3–5 October 2007; pp. 1–6.
16. Liu, Z.; Liu, H.; Xu, W.; Chen, Y. Exploiting Jamming-Caused Neighbor Changes for Jammer Localization. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 547–555. [CrossRef]
17. Liu, Z.; Liu, H.; Xu, W.; Chen, Y. An Error-Minimizing Framework for Localizing Jammers in Wireless Networks. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 508–517.
18. Liu, H.; Xu, W.; Chen, Y.; Liu, Z. Localizing Jammers in Wireless Networks. In Proceedings of the Seventh Annual IEEE International Conference on Pervasive Computing and Communications—Workshops (PerCom Workshops 2009), Galveston, TX, USA, 9–13 March 2009; pp. 1–6.
19. Liu, H.; Liu, Z.; Chen, Y.; Xu, W. Localizing Multiple Jamming Attackers in Wireless Networks. In Proceedings of the 2011 International Conference on Distributed Computing Systems, Minneapolis, MN, USA, 20–24 June 2011; pp. 517–528.
20. Joshi, K.R.; Hong, S.S.; Katti, S. PinPoint: Localizing Interfering Radios. In Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation, Lombard, IL, USA, 2–5 April 2013; pp. 241–253.
21. Rayanchu, S.K.; Patro, A.; Banerjee, S. Catching Whales and Minnows Using WiFiNet: Deconstructing Non-WiFi Interference Using WiFi Hardware. In Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, San Jose, CA, USA, 25–27 April 2012; pp. 57–70.
22. Kim, K.; Nam, H.; Schulzrinne, H. WiSlow: A Wi-Fi network performance troubleshooting tool for end users. In Proceedings of the 2014 IEEE Conference on Computer Communications, Toronto, ON, Canada, 27 April–2 May 2014; pp. 862–870.
23. Kim, Y.S.; Mokaya, F.; Chen, E.Y.; Tague, P. All your jammers belong to us - Localization of wireless sensors under jamming attack. In Proceedings of the IEEE International Conference on Communications, Ottawa, ON, Canada, 10–15 June 2012; pp. 949–954.
24. Cheng, T.; Li, P.; Zhu, S. An Algorithm for Jammer Localization in Wireless Sensor Networks. In Proceedings of the IEEE 26th International Conference on Advanced Information Networking and Applications, Fukuoka, Japan, 26–29 March 2012; pp. 724–731.

25. Cheng, T.; Li, P.; Zhu, S.; Torrieri, D.J. M-cluster and X-ray: Two methods for multi-jammer localization in wireless sensor networks. *Integr. Comput.-Aided Eng.* **2014**, *21*, 19–34. [CrossRef]
26. Benedictis, A.D.; Koosha, B.; Albanese, M.; Casola, V. A Probabilistic Framework for Distributed Localization of Attackers in MANETs. In Proceedings of the Security and Trust Management—9th International Workshop, Egham, UK, 12–13 September 2013; pp. 49–64.
27. Chen, Y.; Yang, J.; Trappe, W.; Martin, R.P. Detecting and Localizing Identity-Based Attacks in Wireless and Sensor Networks. *IEEE Trans. Veh. Technol.* **2010**, *59*, 2418–2434. [CrossRef]
28. Deka, D.; Baldick, R.; Vishwanath, S. Optimal data attacks on power grids: Leveraging detection & measurement jamming. In Proceedings of the 2015 IEEE International Conference on Smart Grid Communications, Miami, FL, USA, 2–5 November 2015; pp. 392–397.
29. Gai, K.; Qiu, M.; Ming, Z.; Zhao, H.; Qiu, L. Spoofing-Jamming Attack Strategy Using Optimal Power Distributions in Wireless Smart Grid Networks. *IEEE Trans. Smart Grid* **2017**, *8*, 2431–2439. [CrossRef]
30. Li, Y.; Shi, L.; Cheng, P.; Chen, J.; Quevedo, D.E. Jamming Attacks on Remote State Estimation in Cyber-Physical Systems: A Game-Theoretic Approach. *IEEE Trans. Automat. Control* **2015**, *60*, 2831–2836. [CrossRef]
31. Wei, X.; Wang, Q.; Wang, T.; Fan, J. Jammer Localization in Multi-Hop Wireless Network: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2016**, *19*, 765–799. [CrossRef]
32. Pelechrinis, K.; Koutsopoulos, I.; Broustis, I.; Krishnamurthy, S.V. Lightweight Jammer Localization in Wireless Networks: System Design and Implementation. In Proceedings of the Global Communications Conference, Honolulu, HI, USA, 30 November–4 December 2009; pp. 1–6.
33. Wu, B.; Jen, C. Particle-Filter-Based Radio Localization for Mobile Robots in the Environments with Low-Density WLAN APs. *IEEE Trans. Ind. Electron.* **2014**, *61*, 6860–6870. [CrossRef]
34. Wood, A.D.; Stankovic, J.A.; Son, S.H. JAM: A Jammed-Area Mapping Service for Sensor Networks. In Proceedings of the 24th IEEE Real-Time Systems Symposium (RTSS 2003), Cancun, Mexico, 3–5 December 2003; pp. 286–297.
35. Xu, W.; Trappe, W.; Zhang, Y.; Wood, T. The feasibility of launching and detecting jamming attacks in wireless networks. In Proceedings of the 6th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, Urbana-Champaign, IL, USA, 25–27 May 2005; pp. 46–57.
36. Liu, H.; Liu, Z.; Chen, Y.; Xu, W. Determining the position of a jammer using a virtual-force iterative approach. *Wirel. Netw.* **2011**, *17*, 531–547. [CrossRef]
37. Bahl, P.; Padmanabhan, V.N. RADAR: An In-Building RF-Based User Location and Tracking System. In Proceedings of the IEEE INFOCOM 2000, The Conference on Computer Communications, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Reaching the Promised Land of Communications, Tel Aviv, Israel, 26–30 March 2000; pp. 775–784.
38. Chen, Y.; Francisco, J.; Trappe, W.; Martin, R.P. A Practical Approach to Landmark Deployment for Indoor Localization. In Proceedings of the Third Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, Reston, VA, USA, 25–28 September 2006; pp. 365–373.
39. Blom, H.A.; Bar-Shalom, Y. The interacting multiple model algorithm for systems with Markovian switching coefficients. *IEEE Trans. Autom. Control* **1988**, *33*, 780–783. [CrossRef]