future internet

MDPI

Article

A Microservice-Based Smart Agriculture System to Detect Animal Intrusion at the Edge

Jinpeng Miao [1,*], Dasari Rajasekhar [2], Shivakant Mishra [1,*], Sanjeet Kumar Nayak [2] and Ramanarayan Yadav [3]

[1] Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA
[2] Department of Computer Science and Engineering, Indian Institute of Information Technology, Design and Manufacturing, Chennai 600127, Tamil Nadu, India; cs22d0003@iiitdm.ac.in (D.R.); sanjeetn@iiitdm.ac.in (S.K.N.)
[3] Department of Electrical and Computer Science Engineering, Institute of Infrastructure Technology Research and Management, Ahmedabad 380026, Gujarat, India; ramnarayan@iitram.ac.in
* Correspondence: jinpeng.miao@colorado.edu (J.M.); shivakaht.mishra@colorado.edu (S.M.)

Abstract: Smart agriculture stands as a promising domain for IoT-enabled technologies, with the potential to elevate crop quality, quantity, and operational efficiency. However, implementing a smart agriculture system encounters challenges such as the high latency and bandwidth consumption linked to cloud computing, Internet disconnections in rural locales, and the imperative of cost efficiency for farmers. Addressing these hurdles, this paper advocates a fog-based smart agriculture infrastructure integrating edge computing and LoRa communication. We tackle farmers' prime concern of animal intrusion by presenting a solution leveraging low-cost PIR sensors, cameras, and computer vision to detect intrusions and predict animal locations using an innovative algorithm. Our system detects intrusions pre-emptively, identifies intruders, forecasts their movements, and promptly alerts farmers. Additionally, we compare our proposed strategy with other approaches and measure their power consumptions, demonstrating significant energy savings afforded by our strategy. Experimental results highlight the effectiveness, energy efficiency, and cost-effectiveness of our system compared to state-of-the-art systems.

Keywords: smart agriculture; animal intrusion detection; LoRa; fog computing

Citation: Miao, J.; Rajasekhar, D.; Mishra, S.; Nayak, S.K.; Yadav, R. A Microservice-Based Smart Agriculture System to Detect Animal Intrusion at the Edge. *Future Internet* **2024**, *16*, 296. https://doi.org/10.3390/fi16080296

Academic Editors: Manuel José Cabral dos Santos Reis and Carlos Serôdio

Received: 23 July 2024
Revised: 12 August 2024
Accepted: 12 August 2024
Published: 16 August 2024

1. Introduction

Smart agriculture harnesses cutting-edge information technology, integrating big data, mobile Internet, cloud computing, and the Internet of Things (IoT) technologies to enable the precise tracking, monitoring, automation, and analysis of agricultural operations. Currently, cloud-based infrastructures are prevalent in supporting various smart agriculture applications and data processing. In this setup, data from smart sensors in agricultural fields are transmitted to the cloud via the Internet, where they are stored and processed for decision-making purposes. However, the utilization of cloud-based infrastructures in smart agriculture comes with two significant limitations, as highlighted in recent research [1]: (i) The transmission of sensor data over the Internet requires continuous connectivity, consuming high bandwidth and resulting in delays, rendering it impractical for rural areas with unstable internet connectivity. (ii) Transmitting large volumes of data from IoT devices to the cloud for storage and processing quickly depletes the energy of battery-powered IoT devices. To overcome these limitations, we propose a LoRa (Long Range)-enabled, fog-based smart agriculture infrastructure. This approach intelligently distributes computational workloads to Raspberry Pi devices, thereby reducing the volume of data transferred to the server and facilitating the delivery of latency-sensitive services in real-time.

Following a survey conducted with farmers to discern the primary challenges amenable to solutions through smart agriculture, animal intrusion emerged as the most pressing con-

cern. Given that farms are typically situated in rural settings, adjacent to natural habitats, the incursion of animals poses a significant threat to farm owners, necessitating prompt action to address the resulting havoc and destruction. Unlike other smart agricultural services such as smart irrigation, crop quality monitoring, and pest control, detecting animal intrusion presents unique challenges due to its inherent uncertainty, lack of control, and unpredictability. Animals may disrupt crops and roam freely across fields at any given moment, leading to substantial losses in production. Consequently, recovering from such damage entails considerable time and financial resources to mitigate the associated costs effectively.

In this paper, building upon our previous research [2], we introduce a cost-effective, energy-efficient integrated fog and LoRa-enabled microservice-based infrastructure designed specifically for smart agriculture. This infrastructure is complemented by an innovative strategy for detecting animal intrusion, which combines passive infrared (PIR) sensors with a rotating camera. Our primary aim is to expedite the detection and localization of animal intrusions for farmers. While a straightforward approach to field monitoring would involve relying solely on cameras without PIR sensors, this method entails the use of a greater number of high-resolution cameras to cover the same area, resulting in higher costs and increased energy consumption compared to PIR sensors. Additionally, employing only cameras necessitates fixing them in a single direction, potentially diminishing the effectiveness of animal identification when an animal approaches the boundary of the camera's field of view. By incorporating a rotating camera, we ensure that the animal is captured at the center of the image, significantly enhancing the success rate of animal identification. Furthermore, our work includes an in-depth analysis of power consumption and a comparison with the camera-only strategy, highlighting the efficiency and efficacy of our proposed approach.

The paper is structured as follows: We commence by illustrating how LoRa's low-power, low-bandwidth, and long-range capabilities revolutionize rural agricultural lands into smart agriculture systems. Subsequently, we delve into the design and implementation of a microservice-based edge server, delivering essential, time-sensitive services to farmers in disconnected Internet environments. To enhance animal intrusion detection, we explore different sensor placement strategies and develop an algorithm to locate invasive animals and predict their future locations. Finally, we meticulously evaluate our system's performance, conducting comparisons with current state-of-the-art frameworks across various metrics such as cost, latency, energy consumption, and distance.

This paper makes the following contributions:

- Adoption of the LoRa protocol effectively addresses the limitations of intermittent Internet connectivity and high latency of cloud-based infrastructure.
- A microservice-based architecture at the edge is used to enable latency-sensitive services to be delivered just in time.
- The proposal of three sensor layouts and an algorithm that accurately predicts the future locations of animals.
- A comprehensive analysis and comparison of the layouts through experiments.
- A comparison of our proposed solution (combination of rotating camera with PIR sensors) with an all-camera strategy in terms of costs and energy.
- A rigorous evaluation and discussion on the accuracy of the algorithm, power consumption, and the practicality of the system.

2. Related Work

With the rise of smart agriculture, numerous systems have emerged. Yet, most grapple with safety hazards, high costs and resource demands, dependency on internet connectivity, or poor performance. Amid the plethora of systems in the literature, our focus lies on the latest intelligent agricultural and animal intrusion detection methods.

Devaraj et al. suggest using the traditional electric fence, which shocks animals that cross the boundary [3]. While effective and easy to install, it requires a consistent and

substantial power supply, along with regular maintenance. In contrast, our system remains unaffected during power outages and, importantly, does not pose risks to animals or people. In [4], authors analyze why traditional methods such as electric fencing are futile in some scenarios and have high costs.

Cameras and computer vision are effective at identifying intruding animals. Some researchers [5–7] use deep learning algorithms to recognize animals captured by the camera at regular intervals. However, fixed interval detection wastes resources and may miss some animals. Yadahalli et al. [4] instead send images to a thin film transistor (TFT) display and use a flash light for better night images, which are more expensive and consume more power. Compared to computer vision, it is also harder for humans to accurately identify animals in images where they make up a small percentage. Instead of capturing images, Thomas et al. [8,9] analyze videos, which is challenging to meet latency requirements and necessitates significantly higher computational power.

Cloud-based infrastructures [4,10,11] are popular in smart agriculture for their powerful computing capabilities. In these systems, data are transmitted over the Internet to the cloud, where the data are stored and processed for decision-making. However, these systems rely on Internet connectivity, which may be unavailable in rural areas, and can result in high latency due to data transmission to the cloud.

The systems proposed from 2017 to 2022 [12–18] that use infrared sensors lack specifics on sensor placement and algorithms. In [5], it fails to achieve better performance. The works presented in [3,19–21] cannot support a large service coverage at a low cost.

In comparison, our proposed system excels at accurately detecting and predicting animal locations while minimizing power consumption and transmission latency, and eliminating dependence on internet connectivity by leveraging LoRa communication protocol.

3. Background

3.1. LoRa and LoRaWAN Protocol

LoRa is an ultra-long-distance wireless transmission technology based on spread spectrum technology [22,23]. Long Range Wide Area Network (LoRaWAN) is a set of communication protocol and system architecture designed for long-distance communication network [23,24]. Lora has a great advantage in handling co-channel interference. It solves the problem of not being able to take into account long distance, anti-interference, and low power consumption at the same time. Compared with other communication technologies, LoRa's ultra-low cost, high sensitivity, ultra-low power consumption, strong anti-interference ability, low bandwidth consumption, and long transmission distance make it ideal for this project.

3.2. IoT Devices

3.2.1. Arduino

A microcontroller-based open source hardware platform. Arduino Mega is Arduino development board based on the ATmega. It is cheap and easy-to-use features make it widely used in practical IoT projects.

3.2.2. Multi-Channel LoRaWAN GPS Concentrator

A high-performance multi-channel transmitter/receiver designed to receive multiple LoRa packets simultaneously. It is intended to provide a robust connection between a central wireless data concentrator and a large number of wireless endpoints over a considerable range of distances.

3.2.3. PIR Sensor

An electronic sensor that measures infrared (IR) light radiating from objects in its field of view. The characteristics of this sensor include the angle of detection (α) and the maximum detectable distance (d) with the detection range calculated as a cone with h as

the diameter of the circle as the base. It is small, cheap, power-efficient, easy to use, and durable. Therefore, it is typically used for security and automatic lighting related purposes.

3.2.4. All-Day Camera

A camera that is able to detect invasive animals both day and night, which requires a built-in motorized IR-cut filter so that it can switch in and out automatically based on light condition. The filter will be turned off with the purpose that only visible light during the daylight, and IR sensitivity during the night with IR LEDs on.

3.3. Fog Computing

3.3.1. Containerization

A software deployment process that bundles the application's code with all the files and libraries the application needs to run on any infrastructure [25]. By virtualizing the operating system kernel, this technology enables user-space software instances to be divided into multiple independent units that run in the kernel as opposed to a single instance. This particular software instance is referred to as a **container**, a software package that provides the complete runtime environment for an application. With containerization, people can create individual packages or containers that can run on all types of devices and operating systems. Containerization is lightweight, portable, scalable, fault-tolerant, agile, and saves hardware resources.

3.3.2. Microservices

A type of software architecture that builds complicated programs using modularity and small functional units that are each focused on a particular responsibility and function. Microservices architecture makes applications easier to scale and faster to develop. Compared to monolithic architecture, microservices architecture is agile, scalable, easy to deploy, technically free, code-reusable, and resilient [26].

4. Proposed System

4.1. System Architecture

The architecture of the proposed microservice-based fog-enabled infrastructure for smart agriculture is shown in Figure 1. It consists of two layers: a sensing layer and fog computing layer, which are linked by cross-layer upstream and downstream communication for data and control information flows [27].

The *sensing layer* is comprised of the sensors and actuators deployed across the agricultural field to periodically sense the physical parameters of interest such as air temperature, air humidity, soil temperature and moisture at various depths, wind speed, and rainfall. To address the challenge of poor Internet connectivity, we have adopted a LoRa and LoRaWAN enabled communication system due to their support for low-power, wide-area networking designed to wirelessly connect limited-energy-operated IoT devices to an edge server at a distance of 1–2 km. The *fog layer* is composed of one or more servers, and provides an administrative control of the entire IoT infrastructure of the agricultural field. It addresses the limitations of intermittent Internet connectivity, high latency and high network bandwidth consumption of cloud-based infrastructure. The fog nodes host latency-sensitive services, including fire detection, sprinkler control, and animal intrusion detection. To enable a flexible architecture that leverages existing container-based support for various machine learning services, we have designed the fog layer as a microservice architecture. This architecture comprises loosely coupled, fine-grained microservices with lightweight protocols. Each microservice performs a specific function, such as data collection, filtering, and processing. For instance, data collectors aggregate data for targeted services, while data filters extract the most relevant information, which is then fed into the respective service.

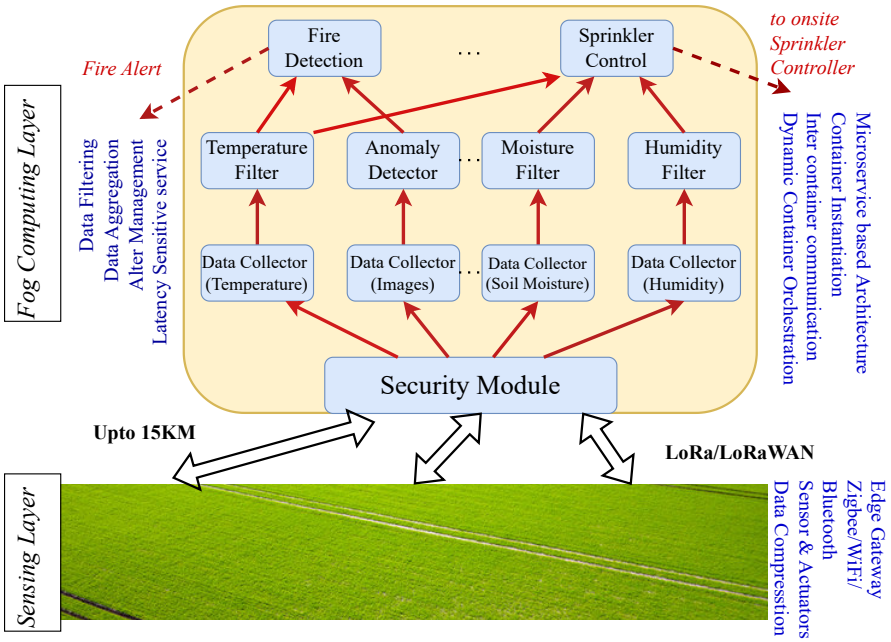

Figure 1. Proposed system architecture.

4.2. Animal Intrusion Detection

In view of the serious problems caused by animal intrusion to farmers, our goal is to automatically detect animal intrusions, identify animals, repel animals with automatic actuations like beep sounds and laser lights, and inform the farmer(s) in a timely manner about the intrusion. This work is performed using two types of sensors: a PIR sensor for detecting any motion in its field of view and an all-day camera sensor attached to the Raspberry Pi for capturing images that will be processed to identify animals. To meet the low-latency requirement, the scheduling mechanism and the prediction algorithm are implemented in the fog layer, while the object detection is carried out on the Raspberry Pi. This is because the low bandwidth of LoRa cannot support the transmission of large-sized images. To achieve a highly flexible architecture that accommodates future additions, we have designed our system to deploy each microservice in its own container, ensuring complete separation and independence. As shown in Figure 2, there are three microservices:

1. The Security module passes the sensor data it receives from authenticated sensors to the appropriate Prediction container.
2. The Prediction container runs localization and prediction algorithms on these data as well as recorded data, and then sends the predicted position at a future time to the camera.
3. The Notification module notifies the farmer via messages once animals are detected.

Corresponding to the process marked with capital letters in Figure 2, the steps are described as follows:

A: Animal movement is detected by PIR sensors and the data are transmitted to the server using LoRa.

B: A container on the edge server predicts the location of the animal at a future time based on input from multiple sensors and sends this location to the Raspberry Pi that operates a camera on the field.

C: The edge server sends a "possible animal invasion" alert to the farmer.

D: The Raspberry Pi instructs the camera to rotate in the direction of the predicted position and take a picture. The Raspberry Pi then runs an animal detection algorithm on this image and sends the results to the edge server.

E: If an animal was identified, actuations are activated immediately to repel it and the edge server sends a reliable alert to the farmer.

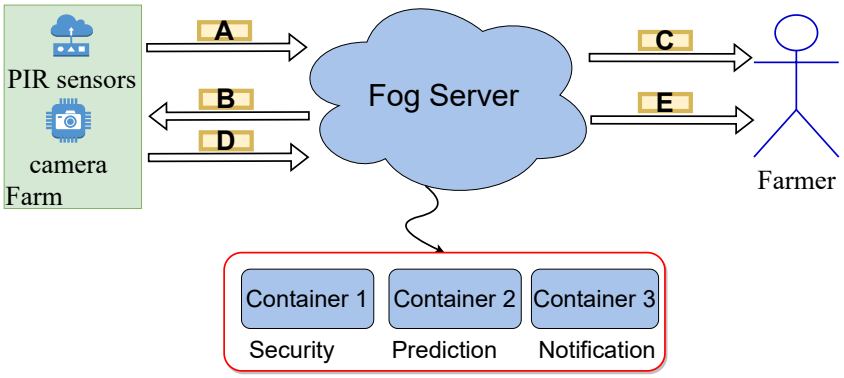

Figure 2. System architecture for animal intrusion detection.

4.3. Sensor Layouts

For animal intrusion detection, a pivotal consideration is the strategic placement of sensors in the field. The objective is to achieve maximum coverage, ensuring thorough data collection, improving prediction accuracy, and accounting for diverse scenarios, all while minimizing the number of sensors. Given the ambiguity surrounding the optimal placement strategy, we put forth and experiment with three plausible layouts tailored for a square-shaped field. The optimal layout certainly depends on the shape of the fields, but the idea remains the same.

We establish four virtual coordinate systems based on the four directions on the farmland (as shown in Figure 3), encompassing the x-axis and y-axis, with the corners serving as the origins of these systems. Our localization and prediction algorithms rely on this coordinate framework. This setup also aims to simplify the process of farmers locating invasive animals. In other words, when sensors detect an animal, the animal's location is regarded as a point (marked with $R1$, $R2$, ... in Figures 4–6) rather than a range, which is convenient for us to design algorithms to predict animals' position. In order to describe the specific location of the animal to the farmer, we define four corners and four sides. Below, we describe the three sensor layouts in detail.

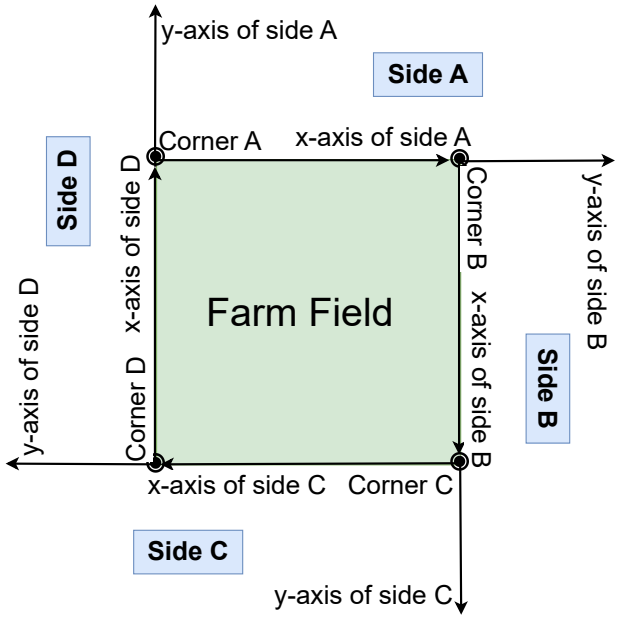

Figure 3. Virtual coordinate systems built upon the farm.

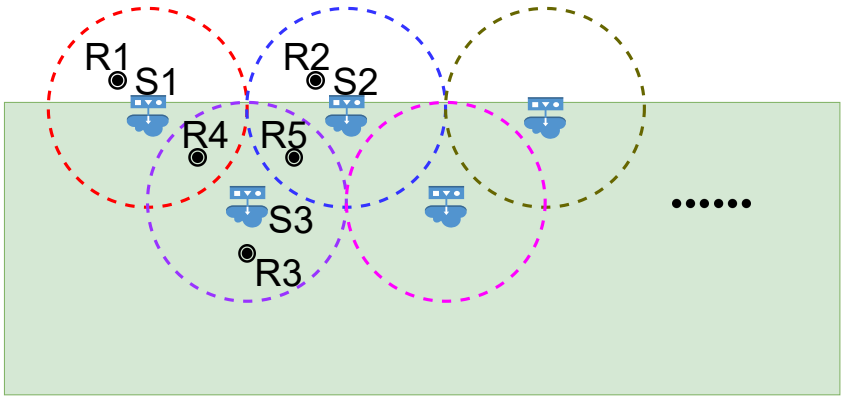

Figure 4. Layout A: vertical placement.

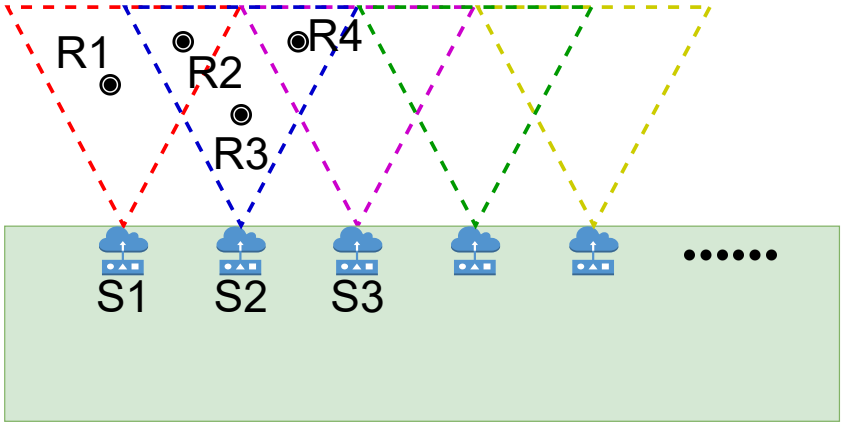

Figure 5. Layout B: horizontal placement.

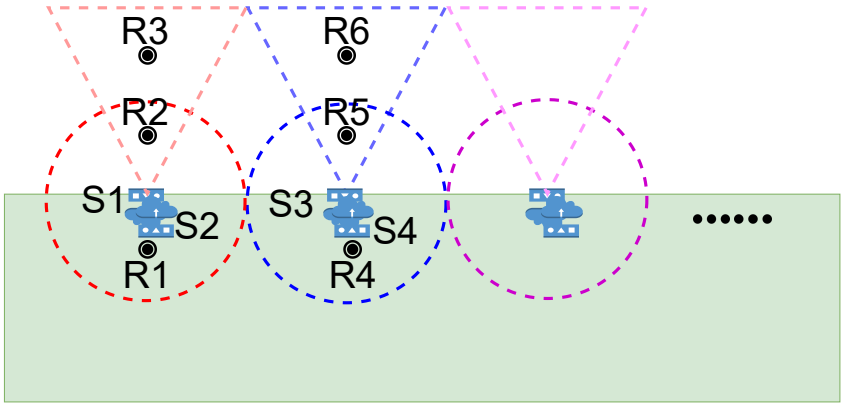

Figure 6. Layout C: hybrid placement.

4.3.1. Layout A—Vertical

We place all sensors at a height of d meters above the ground and project them vertically downward, with the coverage area of each sensor being a circle of diameter h. This produces a coverage area consisting of many circles. As shown in Figure 4, we put two rows of interlocking and overlapping sensors at the boundary of the field. This not only increases the coverage, but the overlapping areas allow for relatively fine-grained segmentation of the area to improve the accuracy of localization and prediction. The increase in budget associated with an additional row of sensors is well worth it compared to more accurately catching animal intrusions and thus preventing damage to the farm. But the shortcoming of this layout is that the farthest detectable location is too close to the farm boundary, only $h/2$, which leads to a greater chance of animal damage to the farm.

4.3.2. Layout B—Horizontal

In contrast to Layout A, all sensors are placed on the ground and horizontally projecting towards the farm's exterior, with each sensor covering an isosceles triangle with h as the base and d as the height, resulting in a coverage area of many triangles. As shown in Figure 5, we put one row of interlocking and overlapping sensors at the boundary. This also has the same advantages as Layout A, i.e., increased coverage and fine-grained area segmentation to improve localization and prediction accuracy. Moreover, it overcomes the limitations of Layout A by extending the farthest detectable distance, thus improving protection.

4.3.3. Layout C—Hybrid

With vertical and horizontal placement, it was natural to explore a hybrid placement. We still place two rows of sensors, one vertically along the boundary and the other projected horizontally outward at the same location, as shown in Figure 6. This layout provides good coverage, fine-grained area segmentation, and a far-reaching detectable location. However, uncovered middle areas can lead to inaccurate or even outrageous predictions. Additionally, this layout has a calculated minimal coverage area.

In addition to these three layouts, we also considered other layouts that required fewer sensors. However, they were ruled out due to their limited coverage, which limits their prediction accuracy, and their short sensing distance to the boundary, which make them unsuitable for fast-moving animals.

4.4. Proposed Algorithm

We propose and deploy an algorithm (as shown in Algorithm 1) on the fog server to predict the future location of intrusive animals based on the previous readings returned by the sensors. Whenever a sensor detects an animal, it sends the data back to the container running the algorithm on the fog server in the format of "#side-#sensor-timestamp". The data received by the container are combined with the previous record to make a prediction. The data here may have two scenarios: one is the data are read back from a non-overlapping coverage area; the other is the animal is in the overlapping area covered by multiple sensors. In this case, there will be multiple sensors with similar timestamps to send back data and the server needs to make the final prediction after receiving data from all these sensors. We use a "tolerance time difference" to define this similar timestamp. In addition, we need to define another threshold as the minimum time interval between animal intrusions, i.e., if the server does not receive new data within the amount of time, the next data received are considered to be a new animal intrusion. It should be determined by the number of animals within the vicinity and their appearance frequency around the field.

We define a mapping of sensor numbers and position coordinates in the algorithm. The server first converts the sensor number in the received data into a coordinate and combines the previous set of coordinates to calculate the distance and direction, and then to calculate the average speed of the animal's movement with the timestamp difference. With the direction and speed, the next position of the animal can be predicted under the assumption that the animal will move in the same direction with the same speed for a short period of time. This period is the sum of the time it takes for the sensor to return data, the time it takes for the algorithm to make the prediction, the time it takes for the instruction to be passed from the server to the Raspberry Pi, and the time it takes for the camera to rotate to point to the predicted position.

The direction and speed of animal movement are not stable, but the constant detection and updating of position information by the sensors, the fast transmission of LoRa, and the high-speed calculation of the system can make the predicted deviation be calibrated quickly and continuously. The field of view of the camera can also provide a certain degree of tolerance. Taken together, our proposed algorithm is expected to effectively and accurately locate and predict the location of animals. Next, we evaluate and verify the adequacy of the algorithm through experiments.

Algorithm 1 Algorithm to Predict Animal Locations

Input: side number *side*, sensor number *sensor*, and timestamp t_{cur}
Output: A coordinate of predicted animal position $\{x_{predict}, y_{predict}\}$

x_{prev} ▷ x value of previous location
y_{prev} ▷ y value of previous location
t_{prev} ▷ Timestamp of previous reading
$time\_threshold$ ▷ Interval to refresh the collected data
$time\_tolerance$ ▷ Interval to define similar timestamp
$latency$ ▷ Time required from detection to camera pointing to the predicted position
$pos\_mapping \leftarrow \{sensor : \{x : y\}\}$
function PREDICT($side$, $sensor$, t_{cur})
    $x_{cur} \leftarrow pos\_mapping[sensor][x]$
    $y_{cur} \leftarrow pos\_mapping[sensor][y]$
    if $t_{cur} - t_{prev} > timing\_threshold$ **then**
        Do nothing
    else if $t_{cur} - t_{prev} \leq time\_tolerance$ **then**
        Wait until all data received
    else
        $dist \leftarrow \sqrt{(x_{cur} - x_{prev})^2 + (y_{cur} - y_{prev})^2}$
        $speed \leftarrow dist/(t_{cur} - t_{prev})$
        $\theta \leftarrow \text{arctangent}(y_{cur} - y_{prev}, x_{cur} - x_{prev})$
        $d \leftarrow latency * speed$
        $x_{predict} \leftarrow x_{cur} + d * math.cos(\theta)$
        $y_{predict} \leftarrow y_{cur} + d * math.sin(\theta)$
        return $x_{predict}, y_{predict}$
    end if
    $x_{prev} \leftarrow x_{cur}$
    $y_{prev} \leftarrow y_{cur}$
    $t_{prev} \leftarrow t_{cur}$
end function
while true **do**
    PREDICT($side$, $sensor$, t_{cur})
end while

5. Evaluation

All experiments presented in this paper using the parameters shown in Table 1. To evaluate our work, we constructed an end-to-end LoRa communication system, deployed the three sensor layouts proposed in Section 4.3, and gathered sensing data by moving along various trajectories. We then implemented our proposed algorithm to analyze the collected data. The tolerance for time difference is set at 0.1 s, and the time threshold is established at 120 s.

Table 1. System configuration.

Component (Manufacturer, City, Country)	Specifications
Arduino Mega (ELEGOO, Shenzhen, China)	256 KB Flash Memory, 8 KB SRAM, 4 KB EEPROM, 16 MHz Clock Speed
Raspberry Pi (Raspberry Pi, Cambridge, UK)	Quad core Cortex-A72 (ARM v8) 64-bit 8 GB RAM, 1.5 GHz Clock Speed
PIR Sensor (WGCD, Shenzhen, China)	Detection range d is 7 m; Detection distance h is 5 m
Camera (Arducam, Shenzhen, China)	Resolution 2592 × 1944, Optical Size 6.35 mm, Focal Length 2.25 mm, FOV 130°(D) 105°(H)

Table 1. *Cont.*

Component (Manufacturer, City, Country)	Specifications
Edge Server (Foxconn, Zhengzhou, China)	3.1 GHz Dual-Core Intel Core i5, 8 GB RAM, 256 GB Disk
LoRa SX1276 (Dragino, Shenzhen, China)	Frequency: 868 MHz, Bandwidth: 7.8 kHz to 500 kHz, Data Rate: 18 bps to 27.5 kbps
PG1302 Concentrator (Dragino, Shenzhen, China)	Tx power up to 27 dBm, Rx sensitivity down to -139 dBm@SF12, BW 125 kHz

5.1. Experiments and Results

5.1.1. Lora Transmission

We built an end node that consists of PIR sensors, one Arduino Mega microcontroller, and LoRa Hat with Antenna. LoRa hat was built using LoRa SX1276 IC Transceiver. To investigate the scheduling capability of the fog node, we connected three end nodes with one LoRa-enabled gateway (Raspberry Pi with PG1302 LoRaWAN Concentrator), which is located 3 km away from the field as shown in Figure 7. The end nodes are scheduled in a round robin fashion by the fog node to avoid the interference of data during simultaneous communication by the three end nodes.

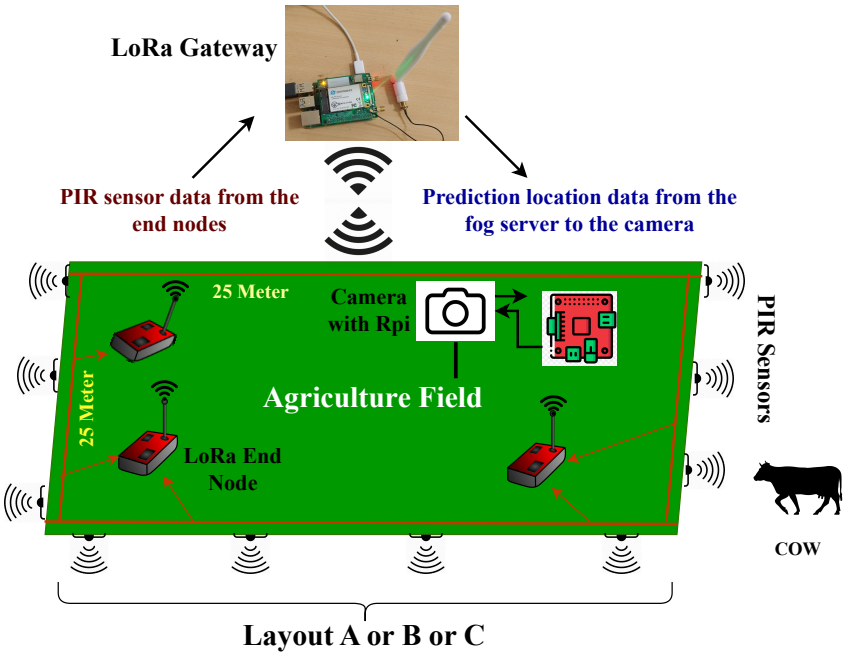

Figure 7. Experimental setup architecture.

5.1.2. Sensor Placement

To detect animals in a 25 by 25 m field, we used PIR sensors. As described in Section 4.3, the layouts of the sensors were accurately deployed. The sensors were fixed to a strip and placed around the perimeter of the field to ensure complete coverage. We used an Arduino ATMega2560 along with a LoRa hat using LoRa SX1276 IC powered by a lithium-ion battery to send data to the Gateway for processing and decision-making.

We changed different speeds, directions, and trajectories to simulate 18 different movements (M1-M18 in Figure 8) of an animal (cow) to evaluate the accuracy and effectiveness of our algorithm. For each of these movements, our system sensed and transmitted PIR sensor values to the edge server for multiple locations depending on which sensors detected movements. For example, Table 2 shows the location values received for Movement M2.

These location data collected from 18 different movements formed the ground truth for our evaluation.

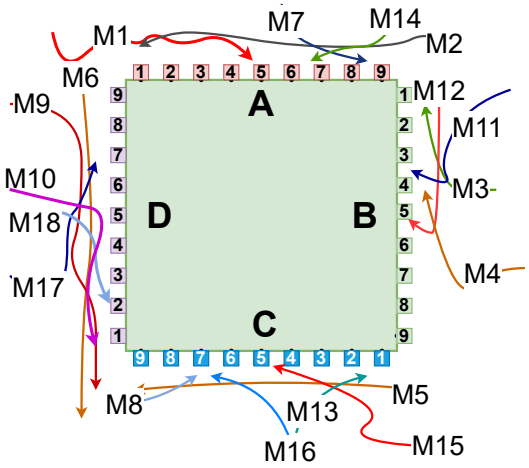

Figure 8. Movement trajectories.

Table 2. M2 location values: side—coordinates (x, y).

	Layout A	Layout B	Layout C
1	A—(20, 1.5)	A—(20, 3.5)	A—(20, 5.5)
2	A—(15, 1.5)	A—(15, 3.5)	A—(15, 5.5)
3	A—(15, 1.5)	A—15, 3.5)	A—(15, 5.5)
4	A—(10, 1.5)	A—(10, 3.5)	A—(10, 5.5)
5	A—(5, 1.5)	A—(5, 3.5)	A—(5, 5.5)

5.1.3. Position Prediction

We used a PC as a fog server in our experiments. It was placed 10 m above the ground to increase the transmission speed with end nodes [27]. The prediction algorithm ran continuously waiting for data. To evaluate our algorithm, we made use of our ground truth data, wherein the container extracted three sets of location data from a movement, used the first two sets of data to predict the location for the time corresponding to the third set of data, and then compared this predicted location with the actual location to assess the accuracy of the prediction. One measure of accuracy we used is the distance offset, which is the distance between the predicted location and the actual location. We measured distance offsets for all movements for which we have at least three location values. For movements such as M2 (Table 2), for which we have more than three location values, we measured distance offset for each triplet of location values resulting in 10 distance offsets measured. Figure 9 shows the average distance offset of each movement.

As we can see, the average distance offset is relatively low (less than 5 m) for most movements and layouts. We observe that Layout B shows relatively small distance offsets in most of the movement tests, although in M1, it had a higher offset in prediction than the other two layouts. However, in M8, Layout B did not have sufficient readings for the algorithm to make predictions due to the presence of some blind triangles near the boundary where the sensor cannot detect the animal once it moves there. Layout C produced the largest distance offsets in most of the tests due to the presence of many blind areas inside the coverage area, which prevented the animal from being detected quickly and continuously, resulting in more inaccurate predictions. Layout A performed moderately and without data loss, which is due to its continuous and extensive coverage area. Based on our experiments and the analysis in Section 2, we recommend Layout B as the optimal sensor deployment method, which can be readily applied to rectangular fields of varying dimensions.

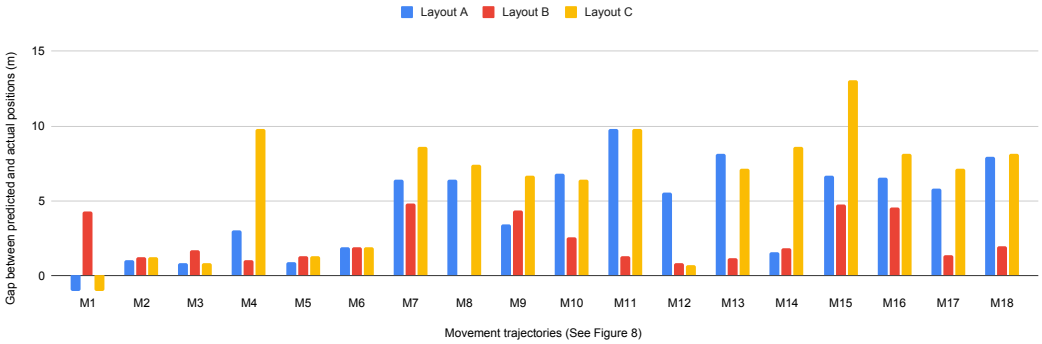

Figure 9. Average distance offset between predicted and actual positions for different types of movements in the three layouts.

5.1.4. Animal Detection

To identify the intrusive animals, we connected a camera to a Raspberry Pi to take pictures of the animals (area where the predicted location is) and identified them using computer vision algorithms. The specifications of the Raspberry Pi are shown in Table 1. We experimented with several popular convolutional neural network (CNN) pre-trained models to test their speed of processing images. We first trained these models on top of a computer and then imported the trained models into the Raspberry Pi. These event-driven models were continuously running on the Raspberry Pi, waiting for images to be taken.

The average detection time (based on 20 runs for each image) is summarized in Table 3. As we can see, these models take 2 to 5 s to identify an animal, with MobileNet having the best performance with an average time of 1.64 s.

Table 3. Animal detection experiment results.

CNN Pre-Trained Model	Latency (s)
VGG16	2.27
ResNet50	3.75
ResNet50V2	3.34
InceptionV3	4.75
MobileNet	1.64
MobileNetV2	2.74
EfficientNetB0	5.07

5.2. Prediction Accuracy

The goal of predicting location is to be able to rotate the camera in a direction where the animal is expected to be. Using the distance offset statistics, we can determine the accuracy of the algorithm by combining the distance between the predicted animal position and the camera placement. As illustrated in Figure 10, the predicted location is represented by point P, and the camera (point C) is positioned within the boundary to point towards the predicted position. The camera has a horizontal field of view of 105 degrees, as shown in Table 1, and the red shading indicates the current range that the camera can cover. If the actual animal position falls within the red shading, represented by point Q, the prediction is considered accurate. Conversely, if the actual animal position falls outside the red shading, represented by point R, the prediction is considered incorrect. Since we only have the distance between the predicted location and the actual location, without knowing their relative positions with respect to the camera, Q could be any place on the red circle which is centered at point P. We make the assumption that the angle formed by the edge PQ and the edge CP at point P is a right angle, so that angle β is the maximum value. In this way, the accuracy of the prediction is the conservative value.

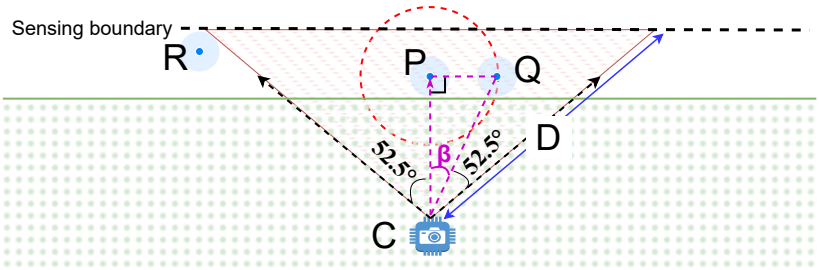

Figure 10. Camera placement.

Based on this validation method, we calculate the prediction accuracy of the three layouts at different distances between the predicted animal position and the camera placement, as shown in the Table 4. The table shows again that Layout B is the best layout solution. For Layout B, a placement distance of 5 m can achieve a very accurate prediction. Notably, Layout C's low accuracy leads to substantial discrepancies between actual and predicted positions for the tested movements. This is attributed to our method of representing areas with central points, which can result in large angles between consecutive line segments. Consequently, the predicted third point may deviate significantly from the actual third point, as observed in this case.

The farther the distance, the wider the coverage, and the higher the accuracy. Nevertheless, we must also consider that increasing the distance results in lower image quality of the animal, which makes animal identification more difficult. We will discuss this further in Section 6.

Table 4. Camera placement and prediction accuracy.

Distance between Camera and Animal (m)	Accuracy (%)		
	Layout A	**Layout B**	**Layout C**
5	66.67%	94.44%	38.89%
10	100%	94.44%	94.44%
15	100%	94.44%	100%

5.3. Imaging Quality and Animal Identification

To fully evaluate the performance of the system, we must also consider the ability to identify the pictured animals. If the animal image is not clear in the picture, it will be difficult to identify. This depends on two factors: the number of pixels that the animal occupies in the image and the pixel requirements of animal recognition algorithms listed in Table 5 [28]. We calculate the number of pixels occupied by different animals at different distances between the camera and the animal based on the camera parameters (as shown in Table 1) and present the results in Table 6.

If the animal is a cow, by comparing these two tables, we can confirm that these widely used models listed can successfully identify animals when the distance between the animal and the camera is 40 m, and we can still use the very effective GoogLeNet and SqueezeNet1_1 when the distance is 80 m. Therefore, to ensure both a large camera coverage to improve the quality of animal imaging and the tolerance of prediction errors, we need to control the camera placement and maximum rotation angle. Specifically, we must ensure that the maximum distance between the camera and the intersection of the coverage boundary and the sensing boundary (i.e., *D* in Figure 10) does not exceed 80 m, with 40 m being the optimal distance. This allows us to adopt MobileNet, which can achieve the best performance shown in Table 3. Applying the same strategy to other animals, we need to make similar adjustments to achieve optimal overall performance. This adjustment should be tailored based on the types of animals most commonly found around the field. For instance, if elephants or bisons are the animals causing crop damage, then placing

cameras a little further from the boundary might be effective. On the other hand, if the animals are deer or goats, we need to reduce the distance between the cameras and the boundary, which will require deploying more cameras.

Table 5. Minimum pixel requirement for CNN models.

Model	GoogLeNet	SqueezeNet1_1	DenseNet201	VGG16/19	MobileNet
Minimum Pixels	15 × 15	17 × 17	29 × 29	32 × 32	32 × 32

Table 6. Pixels animals occupy at different distances.

Distance (m) \ **Animal (Size)**	**Elephant (6 m × 3 m)**	**Bison (3 m × 1.9 m)**	**Cow (2 m × 1.5 m)**	**Deer (1.5 m × 1 m)**	**Goat (0.9 m × 0.6 m)**
10	453 × 298	227 × 189	199 × 151	113 × 99	68 × 60
20	227 × 149	113 × 94	99 × 76	57 × 50	34 × 30
30	151 × 99	76 × 63	66 × 50	38 × 33	23 × 20
40	113 × 75	57 × 47	50 × 38	28 × 25	17 × 15
50	91 × 60	45 × 38	40 × 30	23 × 20	14 × 12
60	76 × 50	38 × 31	33 × 25	19 × 17	11 × 10
70	65 × 43	32 × 27	28 × 22	16 × 14	10 × 9
80	57 × 37	28 × 24	25 × 19	14 × 12	8 × 7

5.4. System Performance and Cost

5.4.1. System Performance

Based on our experiments, we estimate the total time required to achieve animal intrusion detection with the current system configuration (as shown in Table 1), which is summarized in Table 7. It takes approximately 7.11 to 8.61 s for the farmer to receive clear intrusion information, including the predicted location and type of the animal. Kindly note that this duration encompasses both location prediction and animal identification. This is not solely the time it takes to locate the animal when detected by PIR sensors. Furthermore, the farmer will receive consecutive messages to fine-tune the animal's location until the threat is eliminated.

Table 7. System latency.

Step	**Latency (s)**
Transmission of 3 sets of data via LoRa	1
Prediction with proposed algorithm	0.01
Instruction sent to camera via LoRa	1
Camera rotation, image capture and processing	4∼5
Results sent back to fog server via LoRa	1
Alert sent to farmer via LTE	0.1∼0.6 [29]
In total	**7.11∼8.61**

5.4.2. System Cost

To illustrate the expenses incurred during our experiment in the 25 × 25 m^2 field, we have compiled a detailed cost analysis of all the devices used, which is presented in Table 8. With a total cost of USD 285.36, our system is a cost-effective solution for monitoring animal intrusions. As the size of the farm increases, the cost will inevitably rise, but the advantage is that additional expensive equipment, such as a fog server, is not required.

Table 8. System cost.

Device Name	Cost (USD)
Arduino Shield for LoRa	6.12/each × 3 = 18.36
Raspberry Pi 4	95
GPS Concentrator	120
PIR Sensor	0.75/each × 36 = 27
Camera	25
In Total	**285.36**

5.5. Energy Consumption

In terms of energy consumption, we need to conduct measurements in a much larger field (1000 m × 1000 m) to ensure realism and credibility. We assume that the most common animal around the farm field is the cow, and we opt for a 40 m distance between cameras and the sensing boundary so that MobileNet can be used to achieve the best performance. To verify the energy efficiency of our strategy, we compare it with the most common and straightforward approach, which is to use cameras only without PIR sensors. In this comparison, we need to determine the energy consumption difference by only measuring the energy consumption of the unique components of each approach. We also make the assumption that 10 animal intrusion detections are performed each day.

In the experiment, we utilized a power meter capable of directly indicating the energy consumption (measured in milliwatt-hours, mWh) over a specific duration. As depicted in Figure 11, we inserted one end of the power meter into the outlet providing power to the device, and then connected the device to the power meter's outlet. Subsequently, we recorded the time taken to consume 0.001 kWh (1 Wh) of energy for different components.

Figure 11. Power meter connection.

5.5.1. Our Strategy: Rotating Cameras with PIR Sensors

In our proposed strategy, we need to measure energy consumption for two components across various scenarios involving different numbers of animal detections (as detailed in Table 9). The components tested are: (1) One Arduino board, one LoRa hat for the communication with the edge server, along with PIR sensors; (2) One single rotating camera and its connected Raspberry Pi. Here, "number of animal detections" refers to instances where animals intrude into the field and were detected by the PIR sensors and captured by the cameras. The rotation angle was randomly generated for each image captured by the rotating camera.

With the data presented in Table 9, we can calculate the power consumed by both components in one day, as illustrated in Table 10.

Table 9. Time taken to consume 1 Wh for components.

#Detections	Arduino Board, LoRa Hat, and Nine PIR Sensors	Camera and Raspberry Pi
0	166 min 40 s	9 min 18 s
1	166 min 40 s	9 min 15 s
3	164 min 23 s	9 min 5 s
5	164 min 23 s	9 min 2 s
10	160 min	8 min 54 s

Table 10. Energy consumed (*Wh*) by components in one day (24 h).

#Detections	Arduino Board, LoRa Hat, and Nine PIR Sensors	Camera and Raspberry Pi
0	8.64	154.839
1	8.64	154.844
3	8.641	154.862
5	8.641	154.867
10	8.643	154.882

Besides determining the power consumption of each component, we also need to calculate the number of components required to monitor the entire field. As depicted in Figure 12, since the detection distance of PIR sensors is 5 m (as illustrated in Table 1), the capture boundary for the camera is also set to be 5 m away from the farm boundary, aligning with the sensing boundary of the PIR sensors to minimize the number of cameras used. We positioned all cameras along the boundary to ensure coverage of the entire area with the fewest number of cameras, leveraging the maximum distance between points A and B. Since the cameras are rotatable, the angle β can be wider than the field of view as long as the distance between camera and point B is not greater than 40 m.

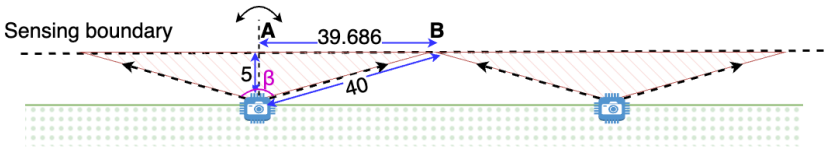

Figure 12. Placement of cameras.

For a field of 1000 by 1000 m, the number of cameras required, C_1, is

$$C_1 = \frac{1000}{39.686 \times 2} \times 4 \approx 52 \quad (1)$$

where 39.686 is the distance between points A and B, as marked in Figure 12. Here, multiplication by four is done to account for the four sides of the farm field.

The number of PIR sensors, S, can be calculated using the following equation:

$$S = \frac{1000}{3.5} \times 4 \approx 1143 \quad (2)$$

where 3.5 is the distance between two adjacent PIR sensors in Layout B. Thus, the number of Arduino boards or LoRa hats needed, A, is

$$A = \frac{S}{9} = \frac{1143}{9} = 127 \tag{3}$$

Finally, the total energy consumption (Wh) can be calculated roughly with

$$P_1 = (((C_1 - 1) \times 154.839 + 1 \times 154.882) + ((A - 1) \times 8.64 + 1 \times 8.643) = 9148.954 \tag{4}$$

5.5.2. All-Camera Strategy

This strategy relies solely on cameras without PIR sensors. With this approach, we must deploy more cameras and fix their positions and directions to ensure the same seamless coverage as the above strategy, as there are no PIR sensors aiding in positioning. And all the cameras must be continuously capturing the field, with image processing running constantly. We measure that it takes 8 min and 2 s to capture and process 36 images, consuming 0.001 kWh of energy (equivalent to 179.253 Wh per day) for each non-rotating camera and its connected Raspberry Pi. So, one clear problem with this strategy is that there is a 13.4 s interval between two captures, which could pose significant issues. Animals intruding the field during these intervals may not be detected.

In addition to maintaining a distance of 40 m between the cameras and the sensing boundary, the angle β also needs to be adjusted to match the field of view of the cameras, as illustrated in Figure 13.

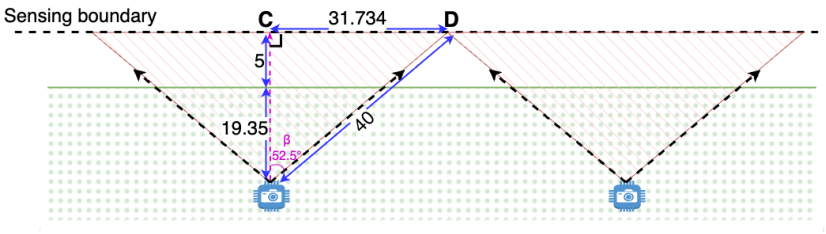

Figure 13. Placement of cameras in an all-camera strategy.

Under this deployment, the number of cameras required, C_2, is

$$C_2 = \frac{1000}{31.734 \times 2} \times 4 \approx 64 \tag{5}$$

where 31.734 is the distance between points C and D, as marked in Figure 13.

Hence, the total energy consumption (Wh) for this strategy is

$$P_2 = (C_2 \times 179.253) = 11{,}472.192 \tag{6}$$

5.5.3. Comparison of Energy Consumption

By leveraging the data measured and calculated above, we can further assess the energy consumption for both strategies over an extended time period, as depicted in Table 11. Our strategy showcases remarkable efficiency gains, conserving 2.323 kWh per day. This translates to substantial long-term savings, amounting to 69.69 kWh per month, 847.895 kWh per year, and an impressive total of 4239.475 kWh over a span of 5 years. Although our approach inherently increases system complexity compared to an all-camera strategy, the significant energy efficiency benefits it offers, coupled with the establishment of a full-time efficient monitoring system, certainly justify this trade-off.

Table 11. Comparison of energy consumption (KWh).

Duration	All-Camera Strategy	Our Strategy	Savings
1 day	11.472	9.149	2.323
1 month (30 days)	344.16	274.47	69.69
1 year (365 days)	4187.28	3339.385	847.895
5 years (1825 days)	20,936.4	16,696.925	4239.475

6. Discussion

6.1. Shape of Farm Field

The evaluation of our system prototype was conducted within the confines of a rectangular field. However, real-world fields exhibit a predominantly irregular shape, necessitating a nuanced approach to the placement of PIR sensors. While the established strategy of aligning sensors along straight boundaries remains consistent and effective, addressing corners emerges as a significant challenge. The diversity in corner shapes demands varied coping mechanisms. Two critical considerations come to the forefront: ensuring an extended detectable distance from the boundary and enhancing seamless coverage. Achieving a longer detectable distance is facilitated by the horizontal projection of PIR sensors outward from the corners. Simultaneously, the goal of enlarging seamless coverage is pursued through the strategic combination of vertically placed sensors or the exclusive use of horizontally placed ones, contingent on the specific field scenarios.

6.2. Other Options for Communication

In addition to the combination of LoRa and cheap camera sensors, alternatives could be to use 4G/5G cellular security cameras or wired solutions. However, 4G/5G cellular security cameras are much more expensive and require a license, making the total cost much higher. Wired solutions are also undesirable since cables would be inconvenient for cultivation and prone to corrosion, making maintenance costly.

6.3. Missing an Animal Intrusion

Based on the results presented in Table 4, it is evident that when an animal is very close to the camera, the success rate of detecting animal intrusion decreases. Naturally, fast-moving animals pose a challenge for our system, as they can quickly move out of the camera's range or get too close, making it difficult to predict their next location. This increases the likelihood of the camera capturing the wrong area. Nevertheless, it is worth noting that fast-moving animals are typically not found in close proximity to farm fields. Lastly, smaller animals that the camera cannot capture clearly are also challenging to detect.

6.4. Multiple Animal Detection and LoRa Transmission Conflict

The proposed system, designed for single animal detection, needs enhancements to handle real-world scenarios with multiple animals invading simultaneously. To address this, advanced computer vision techniques and additional sensors, like Radio-frequency identification (RFID) or acoustic sensors, can be integrated for accurate identification and tracking. Moreover, LoRa transmission conflicts can be resolved by implementing time-division or frequency-division multiple access techniques, collision detection, and retransmission mechanisms, ensuring reliable data transfer and enabling the system to effectively handle multiple animal detections and simultaneous LoRa transmissions.

7. Conclusions

This paper introduces a fog-based smart agriculture system that addresses challenges such as high latency and internet connectivity issues by integrating fog computing with LoRa communication and Raspberry Pi workload distribution. The system employs low-cost PIR sensors and cameras to detect and predict animal intrusion, offering multiple sensor layouts and an algorithm for optimal performance. Additionally, the paper conducts experimental comparisons between these layouts, verifying the algorithm's effectiveness and accuracy. Moreover, this paper includes a comprehensive analysis of power consumption and compares the proposed strategy with a camera-only approach, providing insights into energy efficiency and cost-effectiveness.

Author Contributions: Conceptualization, J.M., D.R., S.M., S.K.N. and R.Y.; Methodology, J.M., D.R., S.M., S.K.N. and R.Y.; Project administration, J.M.; Software, J.M.; Supervision, S.M., S.K.N. and R.Y.; Writing—original draft, J.M. and D.R.; Writing—review and editing, J.M., D.R., S.M., S.K.N. and R.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by TIH-IoT grant number TIH-IoT/2022-07/IC/NSF/SL/NIUC-2022-04/005 and NSF grant number 13012641.

Data Availability Statement: The study does not report any data.

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

References

1. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
2. Miao, J.; Rajasekhar, D.; Mishra, S.; Nayak, S.; Yadav, R. A Fog-based Smart Agriculture System to Detect Animal Intrusion. In Proceedings of the 2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS), Ocean Flower Island, China, 17–21 December 2023.
3. Aiswarya, M.; Banu, E.; Gifta, J.J.; Devaraj, S.A. An Intelligent Agricultural Intrusion Detection and Irrigation Control System Using GSM. *Int. J. Adv. Res. Innov. Discov. Eng. Appl.* **2018**, *3*, 8–15.
4. Yadahalli, S.; Parmar, A.; Deshpande, A. Smart Intrusion Detection System for Crop Protection by using Arduino. In Proceedings of the 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 15–17 July 2020.
5. Radhakrishnan, S.; Ramanathan, R. A Support Vector Machine with Gabor Features for Intrusion Detection in Agriculture Fields. In Proceedings of the 8th International Conference on Advances in Computing and Communication (ICACC-2018) Procedia Computer Science, Kochi, India, 13–15 September 2018.
6. Balakrishna, K.; Mohammed, F.; Ullas, C.R.; Hema, C.M.; Sonakshi, S.K. Application of IOT and machine learning in crop protection against animal intrusion. *Glob. Transit. Proc.* **2021**, *2*, 169–174. [CrossRef]
7. Sabeenian, R.S.; Deivanai, N.; Mythili, B. Wild animals intrusion detection using deep learning techniques. *Int. J. Pharm. Res.* **2020**, *12*, 1053–1058.
8. Thomas, A.K.; Poovizhi, P.; Saravanan, M.; Tharageswari, K. Animal Intrusion Detection using Deep Learning for Agricultural Fields. In Proceedings of the 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 23–25 January 2023.
9. Kiruthika, S.; Sakthi, P.; Sanjay, K.; Vikraman, N.; Premkumar, T.; Yoganantham, R.; Raja, M. Smart Agriculture Land Crop Protection Intrusion Detection Using Artificial Intelligence. *E3S Web Conf.* **2023**, *399*, 04006.
10. Antônio, W.H.; Da Silva, M.; Miani, R.S.; Souza, J.R. A proposal of an animal detection system using machine learning. *Appl. Artif. Intell.* **2019**, *33*, 1093–1106. [CrossRef]
11. Sahana, K. Farm Vigilance: Smart IoT System for Farmland Monitoring and Animal Intrusion Detection using Neural Network. In Proceedings of the 2021 Asian Conference on Innovation in Technology (ASIANCON), Pune, India, 27–29 August 2021.
12. Nikhil, R.; Anisha, B.S.; Kumar, R. Real-Time Monitoring of Agricultural Land with Crop Prediction and Animal Intrusion Prevention using Internet of Things and Machine Learning at Edge. In Proceedings of the 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2–4 July 2020.
13. Venkateshkumar, U.; Anirudh, V.; Khanali, D.; Ezhil, B. Farm Intrusion Detection System using IoT. In Proceedings of the 2022 International Conference on Electronics and Renewable Systems (ICEARS), Tuticorin, India, 16–18 March 2022.
14. Jeevitha, S.; Kumar, S.V. A Study on Sensor Based Animal Intrusion Alert System Using Image Processing Techniques. In Proceedings of the 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 12–14 December 2019.

15. Sharma, P.; Sirisha, C.K.; Gururaj, S.; Padmavathi, C. Neural Network Based Image Classification for Animal Intrusion Detection System. *Int. J. Progress. Res. Sci. Eng.* **2020**, *1*, 1–7.
16. Giordano, S.; Seitanidis, I.; Ojo, M.; Adami, D.; Vignoli, F. IoT solutions for crop protection against wild animal attacks. In Proceedings of the 2018 IEEE International Conference on Environmental Engineering (EE), Milan, Italy, 12–14 March 2018.
17. Vikhram, B.; Revathi, B.; Shanmugapriya, R.; Sowmiya, S.; Pragadeeswaran, G. Animal Detection System in Farm Areas. *Int. J. Adv. Res. Comput. Commun. Eng. (IJARCCE)* **2017**, *6*, 587–591.
18. Mohandass, S.; Sridevi, S.; Sathyabama, R. Animal health monitoring and intrusion detection system based on LORAWAN. *Turk. J. Comput. Math. Educ.* **2021**, *12*, 2397–2403.
19. Geetha, D.; Monisha, S.P.; Oviya, J.; Sonia, G. Human and Animal Movement Detection in Agricultural Fields. *SSRG Int. J. Comput. Sci. Eng.* **2019**, *6*, 15–18.
20. Begum, M.; Janeera, D.A.; Aneesh Kumar, A.G. Internet of Things based Wild Animal Infringement Identification, Diversion and Alert System. In Proceedings of the 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 26–28 February 2020.
21. Saurabh, S.; Milind, J.; Harshita, J.; Mayank, P.; Latif, K.; Amisha, S.; Gautam, A.; Harshal, J.; Jatin S. Self-Intrusion Detection System for Protection of Agricultural Fields Against Wild Animals. *Int. J. Mod. Agric.* **2021**, *10*, 2686–2691.
22. Lora Alliance. Available online: https://www.lora-alliance.org (accessed on 22 July 2024).
23. Alliance, LoRa. A Technical Overview of LoRa and LoRaWAN. White Paper; 20 November 2015. Available online: https://www.academia.edu/31617677/A_technical_overview_of_LoRa_and_LoRaWAN_What_is_it (accessed on 20 July 2024).
24. Lora Alliance. 2017. LoRaWAN 1.1 Specification. October 2017. Available online: https://lora-alliance.org/resource_hub/lorawan-specification-v1-1 (accessed on 22 July 2024).
25. Bernstein, D. Containers and Cloud: From LXC to Docker to Kubernetes. In *IEEE Cloud Computing*; IEEE: Piscataway, NJ, USA, 2014.
26. Chen, R.; Li, S.; Li, Z. From Monolith to Microservices: A Dataflow-Driven Approach. In Proceedings of the 2017 24th Asia-Pacific Software Engineering Conference (APSEC), Nanjing, China, 4–8 December 2017.
27. Mishra, S.; Nayak, S.; Yadav, R. An Energy Efficient LoRa-based Multi-Sensor IoT Network for Smart Agriculture System. In Proceedings of the IEEE Topical Conference on Wireless Sensors and Sensor Networks, (WisNet 2023), Las Vegas, NV, USA, 22–25 January 2023.
28. Models and Pre-Trained Weights. PyTorch. Available online: https://pytorch.org/vision/0.12/models.html#models-and-pre-trained-weights (accessed on 22 July 2024).
29. Ilya, G. *High Performance Browser Networking: What Every Web Developer Should Know about Networking and Web Performance*; O'Reilly Media, Inc.: Newton, MA, USA, 2013.