IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Ground-level Mapping and Navigating for Agriculture based on IoT and Computer Vision

Wei Zhao [1], Xuan Wang [2], Bozhao Qi [2], Troy Runge [1]

[1]Department of Biological System Engineering, University of Wisconsin-Madison, Madison, WI 53706, USA

[2]Electrical and Computer Engineering, University of Wisconsin–Madison, Madison, WI, USA

Corresponding author: Troy Runge: Tel/fax: (608)8903143 E-mail: trunge@wisc.edu.

ABSTRACT Autonomous agricultural systems are a promising solution to bridge the gap between labor shortage for agriculture tasks and the continuing needs for increasing productivity in agriculture. Automated mapping and navigation system will be a cornerstone of most autonomous agricultural system. Accordingly, we propose a ground-level mapping and navigating system based on computer vision technology (Mesh Simultaneous Localization and Mapping algorithm, Mesh-SLAM) and Internet of Things (IoT), to generate a 3D farm map on both the edge side and cloud. The innovation of this system includes three layers as sub-systems that are 1) ground-level robot vehicles' layer for conducting frames collection only with a monocular camera, 2) edge node layer for image feature data edge computing and communication, and 3) cloud layer for general management and deep computing. High efficiency and speed of mapping stage are enabled by making the robot vehicles directly stream continuous frames to their corresponding edge node. Then each edge node, that coordinate a certain range of robots, applies a new Mesh-SLAM frame by frame, whose core is reconstructing the features map by a mesh-based algorithm with scalable units and reduce the feature data size by a filtering algorithm. Additionally, the cloud-computing allows comprehensive arrangement and heavily deep computing. The system is scalable to larger-scale fields and more complex environment by taking advantage of dynamically distributing the computation power to edges. Our evaluation indicates that: 1) this Mesh-SLAM algorithm outperforms in mapping and localization precision, accuracy, and yield prediction error (resolution at centimeter); and 2) The scalability and flexibility of the IoT architecture make the system modularized, easy adding/removing new functional modules or IoT sensors. We conclude the trade-off between cost and performance widely augments the feasibility and practical implementation of this system in real farms.

INDEX TERMS Mesh-SLAM, IoT, Intelligent agriculture, productive agriculture

I. INTRODUCTION

In recent decades agriculture systems have faced both agriculture labor shortage due to the nature of the work and the increasing requirement of productivity. The trend is expected to continue with climate change and increasing population further adding stress to these systems. The U.S. National Agricultural Statistics has revealed the number of farms and ranches has decreased to 2.04 million by 3% from 2012 to 2017, and the land for agriculture has decreased from 2% from 900.2 to 914.5 million acres [1]. This decrease has occurred while the United States population increased by 11.2 million from 2012 to 2017 [2], with similarly challenges globally. The situation of global food demand is even worse than it is in the US [3]. A potential food crisis will happen during the 21st century which could damage international agriculture. While a plateau between the food demand and supply also be possible when ingesting novel and high technologies. Other stresses on agricultural production such as drought, political issues, or the recent COVID-19 outbreak, can cause worldwide intermittent shortage of farm products. Agronomic producers also face growing concerns of the high cost of management, limited ability of crop monitoring, pressures to minimize environmental impact.

A potential solution to mitigate some of the issues are autonomous agricultural systems. These systems are hoped to reduce labor issues for the most dangerous and tedious agronomic tasks, improve efficiency, and reduce environmental impacts through better utilization of crop

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3043662, IEEE Access

inputs. Mapping and localization are key technologies for enabling autonomous navigation systems which in turn will enable autonomous agricultural system.

An agricultural system robot navigation system is implemented by path planning on maps created that monitors both terrain, crops, and other objects. Crop monitoring is also essential to allow robots to distinguish between crops and weeds, monitor plant health, and determine crop maturity. Computer vision on inexpensive visual sensors provides strong support for both local navigation and crop monitoring. However, there are certain related technical challenges in rural fields including data transmission with high bandwidth and high speed, system scalability in different sizes of land, mapping and localization accuracy, updating and maintenance, etc. Rapid advancements in computer vision, mapping, and the Internet of Things (IoT) have provided some solutions as follows.

Computer vision-based in agriculture: Computer vision methods have been highly involved in automated plant monitoring approaches, with representative approaches summarized in TABLE I. More recent approaches have utilized ground-level image data over overhead distant images from satellites or UAVs.

TABLE I
SOME OF THE REPRESENTATIVE APPROACHES

Temporal categories	Approach	Strength	Weakness
Early approach	Satellite imagery [4]	Large landscape coverage	Costly, low special and temporal resolution
	Unmanned Aerial Vehicles (UAVs) [5]	The capability of collecting big data of high special and temporal resolution	Traditional SfM and MVS failed to process dynamic scenes, e.g. growing plants.
Recent approaches	Inexpensive image sensors	Scanning plants and make estimations with computer vision techniques	
	Multi-View Stereo (MVS) [6]	The capability of getting condensed and fine-grained 3D reconstruction	

3D-Mapping for farms: Geometrically mapping between scenarios with changing visual features is a significant step of 3D-Map reconstruction. This data association has been recently utilized in other studies including developing a technique to map varying scenarios by key visual features in different seasons [7]; work done to provide mapping with high robustness under illumination and seasonal variation using scene recognition and localization [8], [9]; a spatial-temporal map that was highly robust to season variations [10]; and a LIDAR system that was adopted to obtain the information in a vertical direction [11]. But these approaches are highly dependent on the prior information of the plant shape, which constrains them from a wide application.

Smart-Farming with IoT: Farm data are increasing since the data collection techniques have been developed, which leads to farming concepts that are more data-driven and data-enabled. This new concept of Smart Farming [12] is the outcome of the rapid development of the Internet of Things (IoT) and cloud computing services. Smart Farming is surpassing precision agriculture because it is depending on both the location and data, improved by environment consciousness, and prompted by real-time instances [13]. It is vital to enhance the spatial farmland surveillance capacity to enlarge the agriculture productivity. Hsu et. al. [14] presented a creative IoT agriculture platform leveraging cloud and fog computing. With the help of fog and cloud, the proposed system can be applied to large-area data collection and analysis, allowing farmland with limited network information resources to be integrated and automated with agricultural monitoring automation, and other related analysis in large areas. Existing work also outlines the challenges and constraints when deploying the IoT in the domain of food and agriculture [15], [16]. Plant monitoring is a key step in navigation where a robot is guided safely and autonomously even in an unknown environment. Thus, robot vehicles should have precise information about their position and be connected with the other robotics via IoT architecture. Muangprathub et. al. monitored temperature, humidity, and soil moisture over a large area using wireless sensor networks. Based on collected information, the system was able to develop the best watering strategy for each plant [17]. Other than plant monitoring, IoT sensors can also help dairy industries for animal health monitoring and analysis [18], [19]. Moreover, smart farming solutions can protect environmentally sensitive areas threatened by damage from cattle herds.

Being scalable of the spatial range of the agriculture applications, e.g. large farmlands, is the key step to achieve high agricultural efficiency. A 3D reconstruction-based navigation, where a robot is localized and guided autonomously with secure even in an unknown scenario, is the significant step of plant monitoring. So, robot vehicles should have precious information about their position and be connected with the other robotics via IoT architecture.

Proposed mapping algorithm: This paper describes a vision-based mapping algorithm involving edge computing to overcome the difficulties faced by the current methods as is shown in Fig. 1. It is precise, inexpensive, and mobile-robot-friendly in agriculture scenarios. It leverages high computation-force edge nodes that supply the Wi-Fi access points (APs) to users and provides computation power to localization algorithms. This scheme works out the problem that a user device doesn't have sufficient computation power to do visual-based tasks. Also, it solves the problem that a centralized server fails to support large quantities of concurrent robots. The edge nodes are managed by a cloud server. And this design has two advantages. One is that confidential information could be filtered by edge nodes before uploading. Another one is that multiple nodes could be regulated by the cloud server to implement navigation among multiple nodes. We propose the weakness of SLAM

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3043662, IEEE Access

can be solved to become robust to environment variations by deep network methods.

1) A precise and expandable image-based mapping and navigation algorithm which contains edge nodes and a cloud server.

2) A mesh-based method to convert a SLAM map into a proper coordinate for implementation convenience.

3) Flexibility on any plant ground shape but not limited to row-shaped plants. The map could adapt to any shaped area.

4) An IoT framework to keep sufficient frame rate data are sent to edge node using UDP protocol. Reduce the feature data size by applying Each Node layer image processes before transmitting data to cloud server.

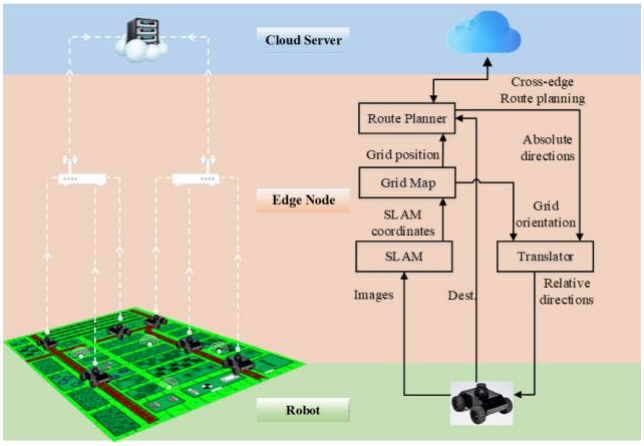


FIGURE 1. **The IoT architecture of Cloud-Edge-Robot**

II. THEORETICAL BACKGROUND

A. SLAM

SLAM is a method to reconstruct the environment in three dimensions and track the movement of the sensor in the environment. The sensors could be inertial measurement unit (IMU), RGB cameras, LIDAR, or GPS. Visual SLAM (vSLAM) only relies on visual data, e.g. photos and depth, and has been a hot topic for a while. It requires three inputs: monocular, RGBD, and stereo, with the solution performed in one of two ways. The first is a feature-based solution, where the inconsistency of image features in sequential image streams is used to recognize camera movement, for example, Mono SLAM [20], PTAM [21] and ORB-SLAM [22]. ORB-SLAM is the most recent technique with reported 1% error of map dimensions. A second solution is a direct method which takes all the images as a unity, as described in DTAM [23] and LSD-SLAM [24]. The SLAM methods are ideal for applications that use smart devices. DTAM requires GPU to become real-time, ORB-SLAM and LSD-SLAM require CPU. PTAM could be real-time on mobile phones if the map is not large [25]. In agricultural systems the outdoor navigation is for a larger area, with the paper looking at a minimum field size of 2.7 acres. Thus, the state-of-art SLAM methods would be taxing to the CPU and the device power supply.

B. IoT- Edge Computing

In edge computing, the tasks are run at the edge of the network [26], which differs from previous systems where the computation work is done by centralized cloud servers. But after the evolution in IoT techniques, the data size has increased tremendously and data transmission and processing have become more challenging. If the computation is finished at the edges and data is kept locally, there are less delay, higher throughput and more confidential [27]. In Para-Drop [28], Wi-Fi routers are treated as edge nodes that directly communicate with users. However, there are very few edge computing applications through a lot of research effort has been reported [29]. One example is using edge computing to performing video streaming [30], and another example is applying edge computing to process big datasets on smart electronic grids [31]. This paper would be another in-detailed contribution to this area.

C. Research Deficiency and Challenge

Considering the precise and performance of the mapping algorithm, the computer vision method and cameras are the best options. A mono-camera tracking algorithm could achieve real-time and accurate performance on a normal laptop with no GPU. For instance, ORB-SLAM reports an error of 1% [32]. Also, the camera orientation is built in the SLAM output. This avoids the difficulty of sensor binding. So, SLAM fits properly in the farm localization application.

However, there are several unavoidable challenges to utilize SLAM into agricultural navigation:

1) The CPUs of smart devices are not as strong as laptops, which could easily fail from the SLAM computation load.

2) Power consumption of smart devices.

3) Robot vehicles do not have enough storage for the maps which are usually larger than thousands of megabytes.

4) Download times and battery life of robot vehicles.

If the SLAM is performed on a centralized cloud server, the computing power and the network bandwidth is challenged if tremendous concurrent robots are doing the image streaming request. Also, the SLAM is designed with a static environment, so it would be hard since the plants change visually along with time.

III. Methods

A. Initialization

SLAM map: Typically, localization and mapping are processed concurrently in SLAM. Here, pre-constructed maps are used to re-localize the camera as shown in Fig. 2. These maps are built by higher-accuracy devices, like stereo cameras, and they are uploaded to edge nodes. With the assumption that switching between edge nodes is performed in the low-level mechanism when the device is roaming, and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3043662, IEEE Access

the switching is opaque to users, the map scope allocated to an edge node is designed to be bigger compared with the designed region to guarantee a seamless edge node switching in the situation that the next edge node is not connected yet while the user is already outside of the map coverage of the original node.

Meshing information: The alignment between the SLAM map and the mesh map is implemented by mapping the horizontal and vertical unit vectors of the mesh algorithm into the SLAM coordinates. The horizontal and vertical axis correspond to the column and the row respectively, while their product is pointing in the upward normal direction. By deducting the SLAM result of one camera location, and together with m and n, the number of columns and rows in the mesh, are easily obtained.

Destinations: In the process of constructing the map or performing re-location afterward, the information of the destinations and their corresponding places in SLAM is recorded by a pair, as shown in Fig. 3-a. is the SLAM coordinates and is a unique string key. The pair will be transformed into where is the corresponding mesh coordinator. These pairs are stored in a hash table for lookup in the future.

B. mn-Scaled Meshing

SLAM algorithms typically are used for localization and motion tracking, thus this use would be limited in navigational maps. Only the keyframes and the features with large intensity gradients in three-dimensional Euclidean space are recorded in a SLAM map [33]. The real-world scale can't be reflected by the map built by a monocular camera, but prior information of the landscape, like barriers, feasible routes, and plants are stored for real-life mapping. In our implementation, seamless inter-edge navigation was provided between neighboring edge nodes by the navigation map [34], which is shown in Fig. 2.

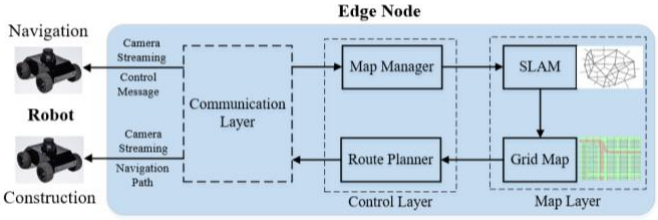


FIGURE 2. The Structure of the Edge Node Layer

A mesh map expanding the horizontal farm landplane is especially proposed to serve the requirements as shown in Fig. 3.

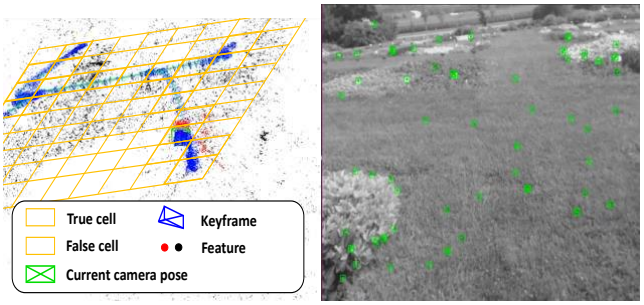


FIGURE 3. a) A demonstration of mn-scaled meshing with SLAM map and b) real-time farm view

This is implemented by an mn-scaled meshing algorithm with each unit of the mesh taking a boolean value to show whether the corresponding area is accessible for robots. Keyframes from SLAM are used to complete the matrix. Besides, this mesh algorithm is based on three assumptions:

1) The land is continuous with no terrace.

2) The keyframes are captured at a vertical height range with small variance.

3) There should be no less than one keyframe captured for an area accessible to robots.

The meshing procedure includes surface matching onto the keyframe coordinates, projecting the coordinates onto a mesh map, checking if each mesh has got at least one keyframe being projected onto, and setting the mesh value to true if yes. Details are described in Algorithm 1 with an example in Fig. 3-b using monocular ORBSLAM2 [35]. A redundant intermediate output is a normal vector of the mapped plane, and it could be avoided by applying a mesh map that labels the accessibility of areas. Thus, this method provides convenience to route planning.

In this research, we reduce the feature data size by a filtering algorithm - Mesh-SLAM. Mesh-SLAM only keep the key features and corresponding mesh map. In this case, around 60% - 80% feature data is discarded depending on feature density in different frames. This Mesh-SLAM is designed for specific situation – large area with significantly similar or redundant features, like most agriculture scenarios. Finally, we are able to balance the trade-off between excessive feature data of a large farm and bandwidth constrain by either hardware or Wi-Fi Communications protocol.

C. Mesh Projection

The step of projecting SLAM coordinates to mesh coordinates involves projecting both the position and orientation as shown in Fig. 4.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3043662, IEEE Access

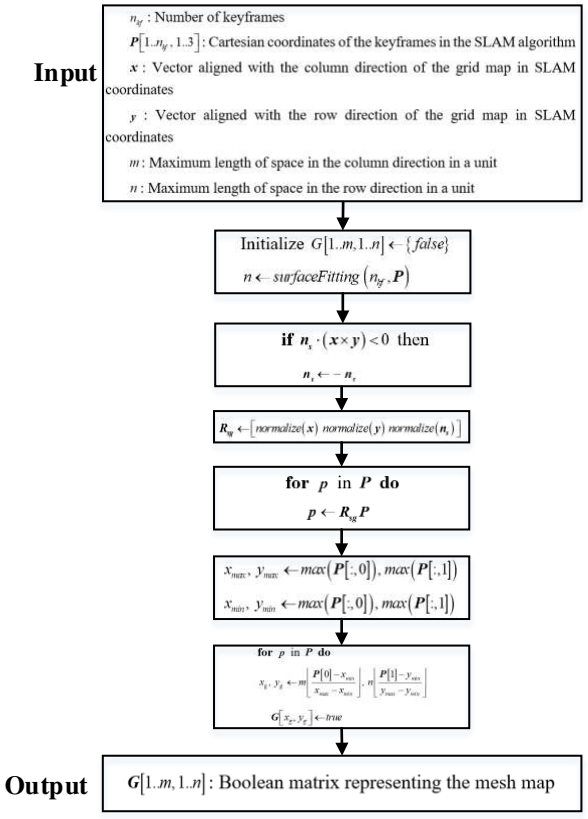


FIGURE 4. mn-Scaled Meshing algorithm

The projection method for projecting position and keyframes onto the mesh coordinate is identical, shown in Fig. 4. The cell for a SLAM coordinate p is (x_g, y_g).

Projecting SLAM orientation onto the mesh map is a necessary step to provide accurate navigation. The three-dimensional orientation is converted onto a planer map. Since each unit in the mesh is neighbored by 8 units, the neighboring units are numbered from 0 to 7. The forward vector of the mesh map is $v_g = R_{sg} v$ given v is the forward vector of the camera of SLAM, as described in Fig. 5. Correspondingly, the oritation is calculated by the projection of the direction and coordinate. After we built a Mesh-map using the algorithm described in Fig. 5, a boolean matrix of the viable cells/positions is created. Since the ratio between real-world scale and the size of each unit in the Mash-map is known, we developed a route planning algorithm to address the undirected weighted graph problem. Fig. 5 illustrate the details of the algorithm. Dijkstra or Bellman-Ford algorithm is well fit in this shortest path route planning problem with OD (origin - destination).

Input:
$C_i[1..n_c^{(i)}]$, $\forall i \in \{1..N\}$: Connection cells of each grid map
$h_i[1..N]$, $\forall i \in \{1..N\}$: Actual cell height of the grid map
$w_i[1..N]$, $\forall i \in \{1..N\}$: Actual cell width of the grid map
$\Phi_c[c_\alpha^{(i)}, c_\beta^{(j)}]$: Whether $c_\alpha^{(i)}$ and $c_\beta^{(j)}$ of grid maps i and j are adjacent to each other
$W_i[c_\alpha, c_\beta]$: Shortest path length between c_α and c_β in grid map i
Output:
$G(V, E)$: Graph representing the graph of the connection cells and the weights between them

Initialize sets V, $E \leftarrow \{\}$
Initialize $W[e]$ such that $e \in E$
for $i \in \{1..N\}$ **do**
  for $j \in \{1..n_c^{(i)}\}$ **do**
    V.insert($C_i[j]$)
  end for
  for distinct j, $k \in \{1..n_c^{(i)}\}^2$ **do**
    $e(C_i[j], C_i[k]) \leftarrow W_i[C_i[j], C_i[k]]$
    E.insert($e(C_i[j], C_i[k])$)
  end for
end for
for distinct i, $j \in \{1..N\}^2$ **do**
  for α, $\beta \in \{1..n_c^{(i)}\} \times \{1..n_c^{(j)}\}$ **do**
    if $\Phi_c[C_i[\alpha], C_j[\beta]]$ **then**
      $e(C_i[\alpha], C_j[\beta]) \leftarrow \frac{h[i]+h[j]+w[i]+w[j]}{4}$
      E.insert($e(C_i[\alpha], C_j[\beta])$)
    end if
  end for
end for
Construct graph $G(V, E)$

FIGURE 5. Route planning algorithm based on the Mesh-map

IV. System Design and Implementation

A. Test Location

The case study is conducted at the West Madison Agricultural Research Station (Madison, WI, USA). The Research Station has 34 kinds of plants. Fig. 6 shows a bird view of the station layout and Fig. 7 is an abstract map showing the testing areas. Fig. 8 gives a direct view for the real experiment scenario from the camera.

FIGURE 6. Map of the testing area.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3043662, IEEE Access

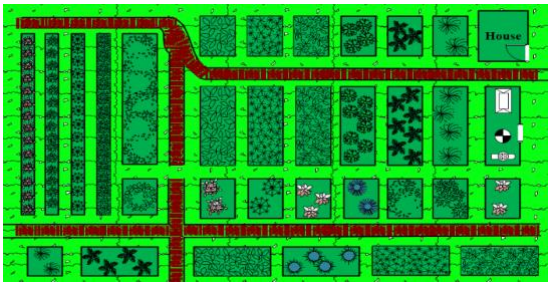


FIGURE 7. **Abstract map of the testing area**

B. Experiment instrument and Data Collection

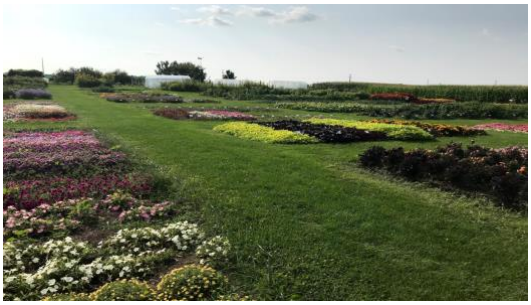


FIGURE 8. **Experiment farm real view**

We collected the data across summer because one of the research questions is how the plants' shape change may affect the mapping and localization results. The frequency of the experiments is performed in accordance to the plants' growth rate. The sample collection date is shown in TABLE II.

We summarized the total number of data capture and time duration for each capture in the following table. It took around 6.5 hours creating the map for the first time and then the time needed for updating maps became less and less. As the farm changes, the system only needs to update corresponding features or missing features when necessary, so it needs less time than creating the map. In other words, the system doesn't need a significant amount of time to maintain the map.

TABLE II.
DATA COLLECTION TIME AND CONTEXT

Sample	Date Measured	Time cost(hour)	Context
1	4/5/2018	6-7	Create the map
2	15/5/2018	4-5	Update the map
3	2/6/2018	3-4	Update the map
4	15/6/2018	2-3	Update the map
5	8/7/2018	2-3	Update the map

C. System Design

The mapping system is designed with three components from bottom to up: robot vehicle, edge nodes, and a cloud server. In Fig. 1, a cloud server controls the edge nodes, and the presented area is managed by edge nodes. The robot vehicle is controlled by the edge node.

Our designed cloud services can be easily migrated to other cloud servers, e.g., AWS, Digital Ocean, and so on. For the edge node, we use an edge computing box, the box is designed based on the Docker container technology, which means it can be run on any device that supports Docker. Hence, we can update and upgrade edge nodes whenever needed. In terms of system software, there are three major modules, including data collection, data transmission, and data analysis. Different modules are responsible for different tasks, each module can be changed or updated without interrupting other modules. What's more, each module (also sub-modules with a module) can be updated over the air as long as the internet connection is good.

Robot vehicles: Only two tasks are designed for robot vehicles and no complex computation is involved. One of the tasks is to transfer the message of destination and its surrounding scenarios captured by its camera to the "region-manager" edge node. The other task is to get the feedback from the edge node with guidance to the destination.

We use a commWercial-off-the-shelf USB camera for video collection on robot vehicles, it is a plug-and-play design. There are no specific requirements on the robot vehicle, so any form of mobile vehicle with a good mileage can be the robot vehicle in our system.

The robot needs the map of the farm for the first time building the SLAM mesh map. After that, it uses a controllable itinerary plan to update the map when necessary. The purpose of having an itinerary plan to make sure all the paths in the farm have been covered. If the system collects enough features to build the SLAM mesh map, then the process is finished. Otherwise, the system will find out which points on the map don't have enough features, and design an itinerary plan to collect data until we have enough features to build the mesh map.

Edge nodes: The procedure of processing a navigation task by edge nodes is shown in Fig. 2. Once the edge node is activated, a robot vehicle sends a navigation request and a sequence of frames with surrounding environment to the connected edge node. Each image is processed by SLAM to calibrate the coordinate and projected onto a mesh map. The mesh location is utilized for navigation. A cloud server is required if the destination is not in the "managing region" of the connected edge node. The last step was to send back the planned route to robots.

Cloud server: The cloud server provides two services: global navigation and map maintenance. The global navigation task was synchronized with the edge node request, but the map maintenance has to be asynchronous because the growing status change of the plants was involved which requires a high computation tracking algorithm and appropriate hardware to process such tasks (e.g., GPU).

Existing cloud-based applications execute computation tasks on the centralized cloud servers. All the data need to be uploaded to the cloud for further analysis. With the increasing number of IoT devices, a large volume of data is produced every second and it is hard to upload and process such data on the cloud server. Edge computing is a newly proposed concept that computation is done at the edge of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3043662, IEEE Access

network, close to where the data is being generated. Raw data can be processed locally and only update necessary data to the cloud. Hence, network bandwidth can be saved and applications can have a better response time. To sum up, we focus on the scalability and flexibility when designing the system from both hardware and software aspects. We believe our proposed design make it easy to adapt most future needs.

D. System Implementation

A SLAM implementation, ORB-SLAM2 [36], was chosen to build this system due to its advantage of engineering convenience. ORB-SLAM2 uses the Boost Serialization library [37] to save and read maps. Other monocular SLAM implementations are also applicable in this system if properly adjusted. The details of the measurement for each section are described in the corresponding paragraphs.

In this work, we choose an RGB camera as the monocular camera to collect data from the farm and conduct various experiments. More specifically, we choose the Logitech-C922x-Pro USB camera in this work with frame rate of 15fps. Logitech C-series USB cameras are widely chosen as monocular cameras when building prototype systems for SLAM and ROS [34], [38]. Also, the H.264 encoder in this camera offers high resolution and frame rate with a reasonable price compared to general RGB cameras.

Besides, we decided to use monocular cameras for the proposed system based on the following reasons. First, a monocular camera is cheaper than other types of cameras like stereo and RGB-D cameras. As the system is mainly used in outdoor environments, it is possible that the system needs to work under extreme weather conditions. So, we can easily replace any broken parts for the system with a lower cost. Second, stereo and RGB-D usually require more computing and energy resources to process collected data. Farming robots have limited computing resources and battery capacities. Moreover, since this camera is used in an outdoor environment which limits the RGB-D camera because the lighting condition is not structured light. Last, machine vision cameras and CCD cameras are overwhelming in terms of cost and performance for this system, since our robot working scope is for mapping and navigation instead of preciously operation. Hence, it is important to use energy efficient hardware and develop energy and computational efficient software. As discussed in our previous work [35], current monocular-based SLAM systems are not suitable to run on portable devices for our purpose.

TABLE III.
SYSTEM IMPLEMENTATION HARDWARE CONFIGURATION

IoT	Item	Description	Item	Description
Cloud Server	CPU	Intel Core i7 8700K	Bandwidth	~500Mbps
	GPU	NVIDIA RTX 1080ti	Network	Wi-Fi *IEEE 802.11ac*, UDP
	RAM	64GB	System	Ubuntu16.04, x86_64
Edge Node	CPU	Hex core ARMv8 64-	Bandwidth	~500Mbps
	GPU	256-core Pascal GPU	Network	Wi-Fi *IEEE 802.11ac*, UDP
	RAM	8GB	System	Ubuntu16.04, x86_64

In terms of computing hardware of this IoT system, different hardware configurations were chosen for Cloud server and Edge nodes as shown in Table III. Since we focus on leveraging the advantages of edge computing platforms to design a system that can work in a large area. The Cloud server is sufficient enough in terms of GPU and RAM in this test scenario. Meanwhile, the bandwidth and CPU could be the restriction if the system need to scale up. Taking advantage of this system, simply upgrading the hardware will solve it that is discussed in Section V.

Our system costs depend on the size of the farmland and also the service/application scenarios. Since this system is easily scalable, people can calculate cost according to this brief explanation of the cost for each component. For the cloud server, public cloud services such as AWS and Azure, the cost is mainly based on the usage and we have student discount. In this case, we set up our own server with ~$3200. And each single edge node is ~$400. While this price varies depending on the network card, storage and other related hardware configurations when each node covers a larger area with higher frame rate. The total number of edge nodes needed also depends on the farm size.

Fault Tolerance: The accuracy requirements for each farm and crop are different. In general, the requirement of map accuracy is much lower when the crops are at their early stages. A 60cm map resolution should be enough as the crop will not flourish at early stages. However, as the crop grows, we may need a map with a 30cm resolution. The map can be updated as the crop grows, and the resolution can be updated eventually. This will reduce the time needed for map creation and updating, and also reduce the computing overhead, which makes the system scale up more easily.

Throughput: Since the bandwidth between the local area network (LAN) and the cloud server could probably be the bottleneck for the number of clients to scale up. Hence the traditional client-cloud server paradigm is not suitable for our purpose, that's also why we proposed the client-edge-cloud architecture in this work. In our design, each robot is configured to stream captured images (640*480) at a frame rate of 6. The robot first streams images to the edge and then to the cloud. With the help of the edge, we can achieve a higher throughput than traditional setups as images can be preprocessed at the edge, which could significantly reduce the size of data that is needed to be uploaded to the cloud.

Load Testing: We have conducted an experiment, with smartphone mimicking the robot streaming, to study the system performances under different loads. As shown in Fig. 12, we have studied how the system performs when there are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3043662, IEEE Access

multiple edge nodes and different numbers of users. Our evaluation results show that the more working robots (more than 16), the lower CPU usage, and the CPU usage of the centralized server setting decreases more rapidly. The CPU usages of both edge node settings increase at first and decrease when there are more than 26 working robots. Hence, we could achieve a good performance of 2 edge nodes. If the system needs to support more robots, we could add more edge nodes to the system.

Reliability: As mentioned previously, the map needs to be updated as the crop grows or farm changes. Updating the map periodically can significantly improve the reliability and robustness of the system. We draw a figure to illustrates how to maintain the map in Fig. 9.

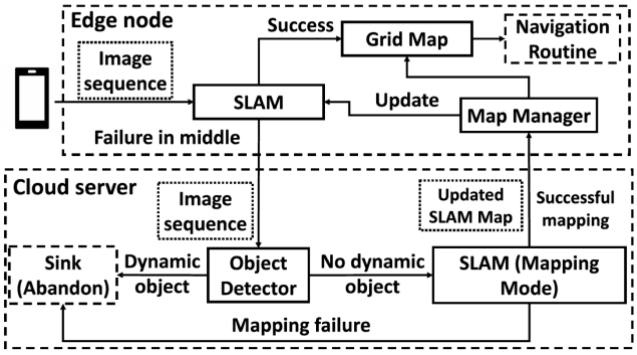


FIGURE 9. The flowchart of map maintenance

V. Results and Analysis

A. Accuracy

The accuracy performance of the mapping between SLAM and the Mesh is an important criterion in this system. This accuracy is evaluated under various scenarios and mesh densities.

Experiment Setup: The testing field is described in the experiment section. During the process of calibrating the SLAM map onto the mesh map, the cameras were set to the space boundary at which location keyframes were captured. And this step ensures the edges of the space, the dimension, and the direction of the mesh map are consistent.

TABLE IV.
RESULTS FROM THE ACCURACY EXPERIMENT

Cell length (approximate) (cm)	30	60
Localization success frequency (%)	84.7	89.3
RMSE (cm)	19.5	0
Maximal error (cm)	36.9	0
Orientation accuracy (%)	100	100

Note: Orientation includes 8 directions separated by 45 degrees.

This effort ensures an accurate calibration and precise output. The ORBSLAM2 has a farm accuracy below 5cm in the monocular mode [39]. The validity of a location within a mesh relies on if a keyframe is captured in that mesh unit. Therefore, the mesh unit size should not be too small to guarantee a keyframe is associated especially when the density of keyframes is unknown. A small-scale preliminary experiment was designed with the mesh unit dimension set to be 30 cm, which is empirically safe for this particular application. The final experiment sets the mesh unit with 30 cm and 60 cm respectively as shown in Fig. 10.

The mesh is perfectly square due to the aspect ratio of the ground, so the actual dimension is considered. Both experiments (30 cm and 60 cm) are conducted with the same route. An edge node is set up and connected with the robot. The OpenCV camera calibration model [40], [41] is used to set up the camera. Each experiment was repeated three times, and in each execution, the robot moves towards the next station along the path (red line shown in Fig. 10), and the edge node computes the mesh coordinates and the ground truth. Fig. 11 shows the real-time path merging results when building the map. Correspondingly, these merged paths could possibly be assigned into different grids when we scale cell size.

Result: Table IV shows the results under both settings. The accuracy is calculated based on the localization success frequency. It is the ratio of the recordings of a correct localization. Another measurement is the accurate localization frequency, which is the proportion of the localization records over the correct localization records. The difference between the computed mesh coordinates and the ground truth is calculated by the Root-Mean-Square (RMSE) metric, which is shown below. Accurate localization is defined if the error is acceptable depending on plant size and farm scale.

$$\mathrm{RMSE}=\sqrt{\frac{\sum_{i=1}^{m}\left(\left(h\left(x_i-x_i^{(g)}\right)\right)^2+\left(w\left(y_i-y_i^{(g)}\right)\right)^2\right)}{m}} \quad (1)$$

Here, the width and height of a unit of mesh are w and h. The coordinates of the server computation result are (x_i, y_i), while $(x_i^{(g)}, y_i^{(g)})$ shows the ground truth. m is the quantity of successful localization records. A maximal error happens if it is measured from the center of the localization output to the actual location using Euclidean distance.

The table shows 84.7% correct localization for the 30 cm group, and with 89.3% correct for the 60 cm group. The RMSA is 19.5 cm and 0 cm for 30 cm and 60 cm group respectively. The maximum error is 36.9 cm in the 30 cm group and 0 for the 60 cm group. The calculated orientation has been verified that they both are constant with the ground truth.

Analysis: This experiment was carried out with two goals: to show the accuracy of the algorithm, and to understand the sensitivity of the Mesh map parameters. The results show the localization success frequencies of both groups are similar. Hence, SLAM is the only key to decide the success of localization. A unit in the mash position could always be calculated if the given SLAM could provide the SLAM map

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3043662, IEEE Access

coordinates. Thus, localization failures are mostly caused by SLAM, which is further discussed in the next section.

In terms of accuracy, the 60 cm group shows the results are matching with the ground truth better. Even with the 30 cm dimension, the maximum error is located at the neighbor of the actual value. There are two possible reasons for the error: SLAM localization failures, and errors of mapping projection between SLAM and mesh. In the aspect of direction, the results are 100% correct for both groups in the 8-direction system. Overall, the experiment shows the algorithm obtains a high accuracy in localization with a 60 cm unit and high performance with 8 directions.

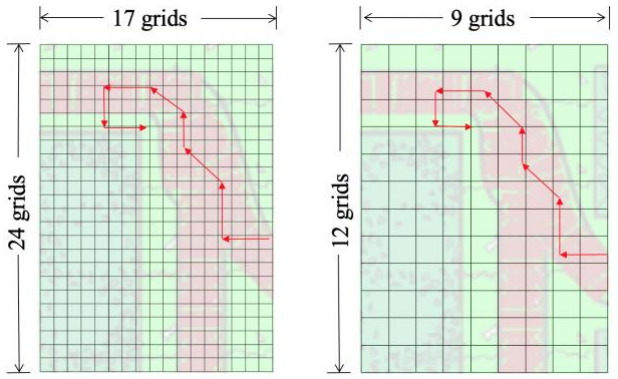


FIGURE 10. **The configuration of the accuracy experiment with a unit of 30cm-cells (left) and 60cm-cells (right)**

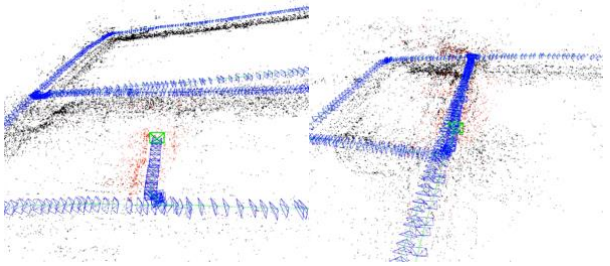


FIGURE 11. **Building the SLAM map including (left) the map before path merging and (right) after path merging**

B. IoT - Scalability and Feasibility for Farm

The capacity of the edge computing framework was evaluated by testing the time gap between neighboring responses on the robot side when the volume of the simultaneous requests from the robots is enlarged step-by-step. This experiment was constructed on a centralized cloud server with one or two edge nodes. The results are shown in Fig. 12.

The cumulative distribution function of time intervals between successive responses from the server and the edge node(s) with different numbers of concurrent users is shown in Fig. 12.

These time intervals can be treated as user waiting time. To calculate the user waiting time, each response's timestamp is subtracted by the previous one. When there are two working edge nodes, the configuration yields much smaller waiting time than other settings. In general, the system can gain more advantages when more edge nodes are available. What's more, the waiting time is smaller than that under the centralized settings when only a single edge node is available. With the number of users increasing from 26 to 36, the centralized service has a significant deterioration. For about 14.8% of cases, a robot needs to wait for at least 2 seconds and even needs to wait for more than 3 seconds for around 5% of cases. If there is one edge node available, 2-second waiting time appears in 8.7% of cases and 1.6% for ~3-second waiting time. Less than 0.5% of cases experienced a more than 2 second waiting time when there are two edge nodes available. Fig. 13 summarizes the minimal, maximal and average CPU usages. CPU usages for 6-robot group is about 193.7%, 201.2% and 124.3% for the single, double edge node and central server respectively. When there are 16 robots, the CPU usages reach to the highest (266.5% and 263.8%) for both edge and centralized settings. The more working robots (more than 16), the lower CPU usage, and the CPU usage of the centralized server setting decreases more rapidly. The CPU usages of both edge node settings increase at first and decrease when there are more than 26 working robots.

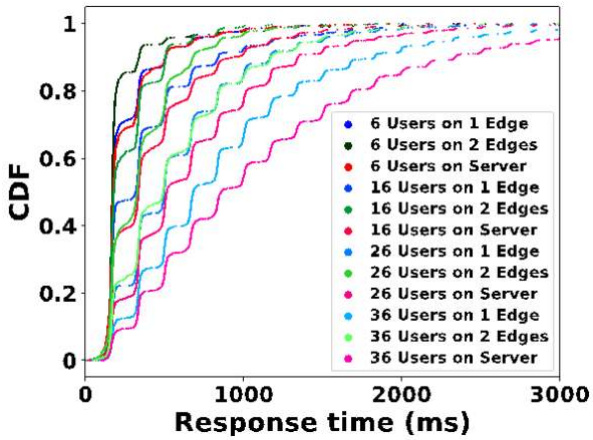


FIGURE 12. **The CDF of the time intervals between responses. Note: User means a working robot**

Results: Here, a user means a working robot vehicle. The time gap can be treated as the user waiting time as well. It is the time interval between the timestamps of each neighboring response. For each experimented user volume, the user waiting time produced by two nodes is much smaller than that of other settings. It suggests that the more the nodes were used, the better the performance, with the time gap of using one node smaller than that with the centralized cloud setting. When the quantity of the robots grows from 10 to 26, the centralized service setting demonstrates a significant performance decrease, wherein 14.8% of cases a robot would have to wait for no less than two seconds, and the cases to stand by for longer than three seconds is 5.0%. For comparison, chances are 8.7% and 1.6% for one-node setting, and the probability is less than 0.5% for the two-node

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3043662, IEEE Access

setting. The CPU usage distribution concerning the number of robots is shown in Fig.13. With 6 robots, the CPU usage is 193.7% for the one-node setting, 201.2% for the two-node setting, and 124.3% for the central cloud server setting. And it is 266.5% and 263.8% for the one-node and centralized cloud server setting with 16 robots. The CPU usage of the one-node and centralized cloud server setting decreases as the robots' number increases.

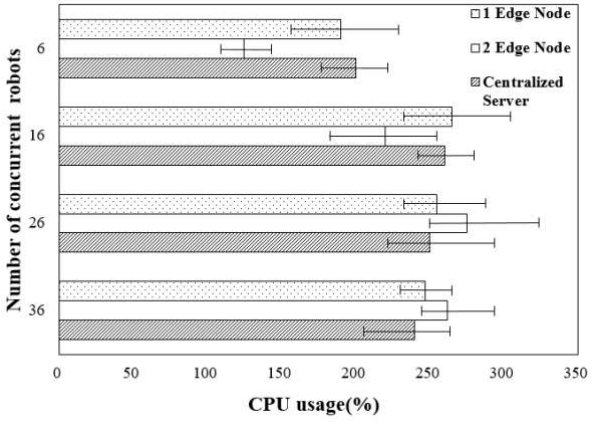


FIGURE 13. CPU usages in each experiment configuration

Analysis: A longer robot waiting time corresponds to worse performance of remote control and processing time. The experiment shows a user would have a higher chance to wait for more than 5 seconds to get the following response if the volume of concurrent working robots increases. There are two possible reasons for this downgraded performance: 1) low computation power, 2) missing packets in the network. However, the experiment shows that the CPU usage decreases when the robot volume increases to 16 and above, which is contradictory. This suggests the longer waiting time is caused by missing packets that deliver images and requests when the robot volume is large. This also explains why the centralized cloud server setting performs worse than edge settings. All users send out UDP packets at about the same frequency. So, if the number of concurrent users becomes larger, the possibility that a communication backlog happens becomes higher. This results in more requests less and less responsive, and thus longer user waiting time. The poor performance of the centralized cloud server setting is because the centralized setting is designed with an extra hop in the connection between the Wi-Fi router and the user, leading to a higher probability of missing packets and longer traveling distances.

The experiment also reveals that under a two-node setting, the system service area could be enlarged under the same request-response criteria. This conclusion relies on the premise that the number of robots connected to each node is equal, as the performance could become worse if the distribution is not even. However, even in the worst case scenario, where all robots are connected to a single node, the performance of the one-node setting is still higher than that of a centralized server setting, though only by a small amount. Thus we conclude that the edge computing scheme proposed in this paper is capable of providing a better concurrency than the traditional centralized cloud server setting.

VI. Conclusion

In this paper, a mapping algorithm and a vision-based farm navigation scheme have been proposed. A cloud-assisted architecture was utilized to disperse the computation load and network communication between multiple edge nodes and a single cloud server. Additionally, a mesh map was presented which avoids the prior information of the testing land. The experiment shows 1) The maximum of localization error is 60 cm, which is among the top performance with other systems 2) This scheme allows larger capacity than the centralized server setting 3) The map could be more frequently updated with different scenarios by taking advantage of this IoT architecture's clever network distribution.

VII. Future Work

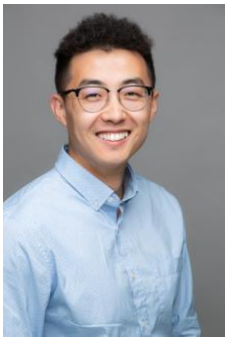
The utilization of SLAM and the assumption of the planar Mesh-map are based on the premise that the testing land is planar. Thus, the adoption of SLAM to better serve the navigation problem could still be improved. Sensors could be added to detect other context information to boost the performance. For instance, an IMU sensor could capture the erect dimension features of the ground and thus the unfavorable paths could be calculated, and the robot could avoid these inappropriate routes. These and other features are imagined making SLAM a viable option for automated mapping and navigation systems to enable autonomous agricultural systems.

REFERENCES

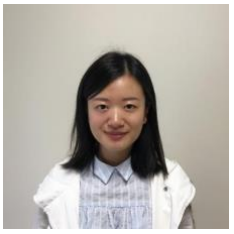
[1] USDA National Agricultural Statistics Service, 2017 Census of Agriculture. Complete data available at www.nass.usda.gov/AgCensus.
[2] U.S. Census Bureau (2020). U.S. and World Population Clock. Retrieved from https://www.census.gov/popclock/
[3] H. Godfray et al., "The future of the global food system," *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 2010, vol.365, no. 1554, pp. 2769-2777.
[4] F. Rembold et al., "Using low resolution satellite imagery for yield prediction and yield anomaly detection," *Remote Sensing*, 2013, vol. 5, no. 4, 1704–1733.
[5] K. Zainuddin et al., "Verification test on ability to use low-cost UAV for quantifying tree height," in *Proceedings of the 2016 IEEE 12th International Colloquium on Signal Processing & Its Applications, Malacca City*, Malaysia, 2016, pp. 317-321.
[6] Furukawa, Y., Ponce, J., 2010. Accurate, dense, and robust multi-view stereopsis. IEEE Trans. Pattern Anal. Machine Intell. 32(8), 1362–1376.

[7] S. Griffith, and C. Pradalier, "Reprojection flow for image registration across seasons," in *Proceedings of the 27th British Machine Vision Conference*, York, UK, 2016.
[8] T. Naseer, B., Suger, M. Ruhnke, and W. Burgard, "Vision-based markov localization across large perceptual changes," in *Proceedings of the European Conf. on Mobile Robots*, Lincoln, UK, 2015.
[9] S. Lowry et al., "Visual place recognition: A survey," *IEEE Trans. Robotics.* 2016, Vol. 32, no. 1, pp. 1–19.
[10] C. Beall, and F. Dellaert, "Appearance-based localization across seasons in a metric map," in *Proceedings of the IROS Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, Chicago, USA, 2014.
[11] M. Stein, S. Bargoti, and J. Underwood, "Image based mango fruit detection, localisation and yield estimation using multiple view geometry," Sensors, 2016, vol. 16, no. 11, pp.1424-8220.
[12] H. Sundmaeker, C. Verdouw, and S. Wolfert, "Digitising the Industry-Internet of Things connecting physical, digital and virtual worlds," in *Proceedings of the Internet of food and farm 2020*, Vermesan, O., and P. Friess, Ed. 2016, pp. 129-151.
[13] A. Kaloxylos, A. Groumas, and V. Sarris, "A cloud-based Farm Management System: Architecture and implementation," *Computers and Electronics in Agriculture*, 2014, vol.100, pp. 168-179.
[14] H. Tse-Chuan, H. Yang, Y. Chung, and C. Hsu, "A Creative IoT agriculture platform for cloud fog computing," *Sustainable Computing: Informatics and Systems*, 2018, pp. 2210-5379.
[15] B. Christopher et al., "IoT in agriculture: Designing a Europe-wide large-scale pilot," *IEEE communications magazine*, 2017, vol. 55, no. 9, pp. 26-33.
[16] T. Kerry et al., "Farming the web of things," *IEEE Intelligent Systems*, 2013, vol. 28, no. 6, pp. 12-19.
[17] M. Jirapond et al., "IoT and agriculture data analysis for smart farm," *Computers and electronics in agriculture*, 2019, vol. 156, pp. 467-474.
[18] T. Mohit, J. Byabazaire, N. Jalodia, A. Davy, C. Olariu, and P. Malone, "Machine learning based fog computing assisted data-driven approach for early lameness detection in dairy cattle," *Computers and Electronics in Agriculture*, 2020, vol. 171, pp. 0168-1699.
[19] T. Mohit et al., "SmartHerd management: A microservices-based fog computing–assisted IoT platform towards data-driven smart dairy farming," *Software: Practice and Experience*, 2019, vol. 49, no. 7, pp. 1055-1078.
[20] A. J. Davison, I. D. Reid, N. D. Molton, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2007, no. 6 pp. 1052-1067.
[21] K. Georg, and M. David, "Parallel Tracking and Mapping for Small AR Workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.
[22] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, 2015, vol. 31, no. 5, pp. 1147-1163.
[23] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 2320-2327.
[24] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proceedings of the European Conference on Computer Vision*, Springer, Cham, 2014, pp. 834-849.
[25] G. Klein, and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proceedings of the 2007. 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225-234.
[26] W. Shi, J. Cao, and Q. Zhang, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, 2016, vol. 3 no. 5, pp. 637-646.
[27] T. Taleb, S. Dutta, and A. Ksentini, "Mobile edge computing potential in making cities smarter," *IEEE Communications Magazine*, 2017, vol. 55, no. 3, pp. 38-43.
[28] P. Liu, D.Willis, and S. Banerjee, "ParaDrop: Enabling Lightweight Multi-tenancy at the Network's Extreme Edge Edge Computing," in *Proceedings of the* 2016 IEEE/ACM Symposium on Edge Computing (SEC), Washington, DC, USA, 2016.
[29] H. H. Pang, and K. L. Tan, "Authenticating query results in edge computing," in *Proceedings of the 20th International Conference on Data Engineering*, Boston, MA, 2004.
[30] S. Wang, X. Zhang, and Y. Zhang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, 2017, vol. 5, pp. 6757-6779.
[31] N. Kumar, S. Zeadally, and J. J. P. C. Rodrigues, "Vehicular delay-tolerant networks for smart grid data management using mobile edge computing," *IEEE Communications Magazine*, 2016, vol. 54, no. 10, pp. 60-66.
[32] A. Huletski, D. Kartashov, and K. Krinkin, "Evaluation of the modern visual SLAM methods," in *Proceedings of the 2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference*, St. Petersburg, Russia, 2015, pp.19-25.
[33] G. Klein, and D.Murray, "Improving the agility of keyframe-based SLAM," in *European Conference on Computer Vision. Springer*, Berlin, Heidelberg, 2008, pp. 802-815.
[34] L. Kang, W. Zhao, B. Qi, and S. Banerjee, "Augmenting self-driving with remote control: Challenges and directions, " in *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*. pp, 19-24.
[35] W. Zhao et al., "Vivid: Augmenting Vision-Based Indoor Navigation System with Edge Computing," IEEE Access, 2020, vol. 8, pp. 42909-42923.
[36] W. Zhao et al. "Real-time vehicle motion detection and motion altering for connected vehicle: Algorithm design and practical applications," Sensors, 2019, vol. 19, no. 19, pp. 4108.
[37] R. Ramey, "Boost serialization library," URL www. boost. org/doc/libs/release/libs/serialization, 2008.
[38] B. Qi et al., "DrivAid: Augmenting driving analytics with multi-modal information," in *Proceedings of the 2018 IEEE Vehicular Networking Conference*, 2018, pp. 1-8.
[39] M. Voisin-Denoual, "Monocular Visual Odometry for Underwater Navigation: An examination of the performance of two methods," 2018.
[40] W. Yin, Y. LUO, and S. LI, "Camera calibration based on OpenCV," *Computer Engineering and Design*, 2007, pp.1-063.
[41] X. Wang et al., "Temporal Frame Sub-Sampling for Video Object Tracking," *Journal of Signal Processing Systems*, 2019, pp.1-13

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3043662, IEEE Access



WEI ZHAO received M.S. degree in Computer Science and his joint Ph.D. degree in Traffic Engineering from the WTU and University of Wisconsin – Madison. He also finished his second Ph.D. defense majoring in Biological Systems Engineering from University of Wisconsin – Madison. He has published over 20 journal and conference papers and 3 patents. He works on Intelligent and Precision Agriculture, Robotics AI & Sensors, IoT and Edge Analytics, Autonomous Vehicles.



XUAN WANG received the B.S. degree in Electrical Engineering from Northwestern Polytechnical University, School of Electronics and Information, Xi'an, China, in 2012. She received the M.S. and Ph.D. degree in Electrical Engineering from the University of Wisconsin – Madison, WI, in 2016 and 2019. From 2014-2019, she was a Research Assistant at the Department of Electrical and Computer Engineering in the University of Wisconsin – Madison. She is currently a Data Scientist. Her research interest was focused on computer vision and signal processing, including image processing, video object tracking, video-based human activity analysis, and big data analysis.

BOZHAO QI is a research assistant in the Department of Computer Sciences and a PhD student in the Department of Electrical and Computer Engineering at the University of Wisconsin-Madison. His research interests lie in the fields of mobile computing, mobile health, context awareness and ubiquitous computing. Qi received his BS degree in Electrical Engineering and Computer Science from Case Western Reserve University. He has worked on several vehicular-related projects during his Ph.D. study. The topics of projects cover sensing vehicle dynamics, transit analytics, human mobilities and so on. Recently, he is currently working on the driving behavior detection and evaluation.

TROY RUNGE is the Patrick Walsh and Noreen Warren Endowed Professor and Chair in Biological Systems Engineering of College of Agricultural & Life Sciences. Troy Runge's research focuses on biorefinery systems that create the most value from biomass feedstocks and make both renewable fuels and materials. He is investigating diverse biomass processes to produce fiber for paper and sugar for ethanol by retrofitting pulp mills. His research is heavily applied and utilizes collaborations with Wisconsin companies. Ultimately, the research could improve processes for biomass aggregation, storage and transportation.