Field Evaluation of Path-Planning Algorithms for Autonomous Mobile Robot in Smart Farms

JEONGHYEON PAK\textsuperscript{1,2}, JEONGEUN KIM\textsuperscript{3}, YONGHYUN PARK\textsuperscript{1,2}, and HYOUNG IL SON\textsuperscript{1,2}(Senior Member, IEEE)

\textsuperscript{1}Department of Convergence Biosystems Engineering, Chonnam National University, 77 Yongbong-ro, Gwangju 61186, Republic of Korea
\textsuperscript{2}Interdisciplinary Program in IT-Bio Convergence System, Chonnam National University, Yongbong-ro 77, Gwangju 61186, Republic of Korea
\textsuperscript{3}Hyundai Robotics, Yongin-si 16891, Republic of Korea

Corresponding author: Hyoung Il Son (e-mail: hison@jnu.ac.kr).

This paper was supported by Korea Institute of Planning and Evaluation for Technology in Food, Agriculture and Forestry (IPET) through Smart Farm Innovation Technology development Program, funded by Ministry of Agriculture, Food and Rural Affairs (MAFRA) (421031-04).

ABSTRACT
Path planning is crucial for several applications, including those in industrial facilities, network traffic, computer games, and agriculture. Enabling automated path-planning methods in smart farms is essential to the future development of agricultural technology. Path planning is divided into global and local planners. Global planners are divided into different types and use well-known grid-based and sampling-based algorithms. In this paper, we propose an algorithm suitable for smart farms in combination with simultaneous localization and mapping (SLAM) technology. The characteristics of the grid-based Dijkstra algorithm, the grid-based A* algorithm, the sampling-based rapidly exploring random tree (RRT) algorithm, and the sampling-based RRT* algorithm are discussed, and an algorithm suitable for smart farms is investigated through field tests. We hypothesized path planning for an agricultural harvesting robot, a spraying robot, and an agricultural transport robot, and conducted experiments in environments with static and dynamic obstacles. In addition, the set parameters are validated experimentally. The Shapiro–Wilk test is used to confirm the shape of the normal distribution, and the analysis of variance (ANOVA) and Kruskal–Wallis test are performed to confirm the significance of the experimental results. Smart farms aim to minimize crop damage; thus, it is vital to reach the goal point accurately rather than quickly. Based on the results, we determined that the A* algorithm is suitable for smart farms. The results also open the possibility of reaching the correct destination in the shortest time when working in smart farms.

INDEX TERMS
path planning, navigation, smart farm, autonomous mobile robot, simultaneous localization and mapping

I. INTRODUCTION
Today, agriculture is industrialized, and economies of scale in agricultural production have been achieved. In addition, the integration of digital technology has contributed to the development of innovative farming techniques. In developed countries, agriculture has gone beyond human labor and currently operates in the form of smart farms (Fig. 1). Given the global push for sustainable agriculture, smart farms offer a science-based agricultural alternative that utilizes automated facilities and information and communication technology (ICT) to observe and optimally manage the agricultural environment, to overcome the constraints of time and space. As current smart farms are based on the greenhouse environment, it is possible to control the environment by controlling factors such as temperature and humidity in the greenhouse [1].

Autonomous driving technology improves farmers’ quality of life through simple repetitive tasks and helps reduce labor. This technology could be applied to mobile robots, which could be used to seed [29], harvest [5], [8], [6], spray [4], [10], and transport [32] crops in smart farms based on greenhouse operations.

Several machine vision-based agricultural robots that have greater utility when applying autonomous driving technology have been developed [13]. Mobile robots, such as unmanned ground vehicles (UGVs) and unmanned aerial vehi-
cles (UAVs), are increasingly being used in agriculture, with a major focus on improving farm detection and autonomous navigation [11]. Using robots increase the efficiency and functionality of work in complex agricultural environments [12]. The basic motion of a mobile robot follows a row between crops, detects the end of a row, exits a row, and rotates and enters the next row. Repeating these steps allows the mobile robot to visit each row multiple times [14]. The range of applications of mobile robots in various fields has gradually been expanding. They can solve the aging problem in rural areas by minimizing the need for human labor on farms.

Mapping and localization are important aspects of advanced autonomous navigation systems for mobile robots [3]. The SLAM algorithm is being actively researched in the field of digital agriculture. Most SLAM developments have focused on outdoor mapping for unstructured environments. However, with the advent of smart farms, indoor SLAM has become increasingly important in the agricultural environment. This paper maps a smart farm using a UGV equipped with three-dimensional (3D) LiDAR and compares the performance of a path-planning algorithm.

Navigation problems are considered as path planning, localization, and motion control. Autonomous mobile robots can travel on an optimal path from their starting position to their goal without colliding with obstacles in their work environment through path planning [2]. Path planning can be divided into global and local path-planning methods. Global path planning considers the given map and searches for the entire travel path, essential for setting the optimal path rather than the time required for path calculation. On the other hand, a local path detects obstacles in real time with various sensors (e.g., LiDAR sensors, laser sensors, ultrasonic sensors, infrared sensors, and cameras) and sets the path.

The contribution of this paper is the selection of an algorithm suitable for smart farms by comparing various global planners: 1) We conducted field evaluations in the a special-purpose environment of a real smart farm, and 2) A more feasible study was conducted by adding a path-planning scenario in which static and dynamic obstacles were combined. The analyzed global planner can be applied for various purposes in various environments.

The remainder of this paper is structured as follows: Section 2 describes related work. Section 3 defines problems that require path planning in smart farms and explain the concepts of path-planning algorithms. Section 4 presents the system implemented in this experiment. Section 5 compares path-planning algorithms in field tests with combined static and dynamic obstacles. Section 6 discusses the limitations of the study, Section 7 concludes the paper, and Chapter 8 talks about future work.

II. RELATED WORK

As the interest in robotics increases, SLAM and path-planning algorithms are being actively studied for autonomous driving using mobile robots in atypical environments such as smart farms.

A. SIMULTANEOUS LOCALIZATION AND MAPPING

New technologies such as machine vision and SLAM have tremendous potential in advanced agricultural applications [13]. SLAM is divided into two main methods: visual simultaneous localization and mapping (V-SLAM), which depends on a camera, and light detection and ranging simultaneous localization and mapping (LiDAR-SLAM), which depends on a LiDAR sensor. V-SLAM is still being actively studied [64], [66], [67]. However, monocular cameras cannot provide 3D distance information, and the depth accuracy of stereo and RGB-D cameras is inversely proportional to their detection range, which is about 20 m. In addition, it is difficult to provide reliable localization due to poor performance in tracing feature points in an environment that lacks texture, such as a white wall without objects [65]. On the other hand, the LiDAR sensor not only provides centimeter-level distance information but also has a wide detection range and is unaffected by lighting conditions [33], [34]. LiDAR is more suitable because the outer wall of a smart farm is made of glass or vinyl and is easily affected by light during the day.

Simultaneous localization and mapping (SLAM) has been developed for use in various fields [15], including agriculture. Among LiDAR-based SLAM algorithms, Gmapping [16] and Google’s Cartographer [17] are the most commonly used.
The trajectory obtained from the Google’s Cartographer system was compared with the ground truth, and favorable results were obtained, relevant for indoor robotics applications [18].

### B. PATH-PLANNING ALGORITHMS IN SMART FARMS

The basic algorithm is the basis for the application. The grid-based Dijkstra [44], [58] and A* [53] - [59] algorithms and the sampling-based RRT [62], [63] and RRT* [43], [60], [61] algorithms have been used in many recent studies. Path-planning algorithms are being studied for various applications, such as in surgical robots [50] and autonomous underwater vehicles [51]; a hybrid path-planning algorithm based on membranes [35], [36] is also being studied. In agricultural environments, research in navigation with mobile robots continues as well [7], [9]. To the best of our knowledge, there are not many navigation studies carried out in real, indoor smart farms. Research is mostly carried out in outdoor agricultural environments such as orchards and paddy fields [30], [31] and cannot be applied to indoor agricultural environments. We need to know the basic path-planning algorithm and scale it up for agriculture.

## III. PATH-PLANNING ALGORITHMS

We need to select an appropriate path-planning algorithm considering the special environment of a smart farm. Path-planning algorithms have their individual characteristics. In this section, we identify the reasons for the need for path planning in a smart farm and analyze the characteristics of various path-planning algorithms.

### A. PROBLEM DESCRIPTION

In smart farms, various robots, such as harvesting robots, spraying robots, and transportation robots, are used depending on the purpose. There may be many dynamic or static obstacles in the robot driving path. As only one robot can pass through the internal corridor and gates, for various robots to work, it is necessary to set the path efficiently. The path-planning algorithm searches for the number of paths the robot travels in various cases. The optimum path is then searched according to the purpose of the algorithm. This study evaluates algorithms for setting the optimal path for almost every path between the start and goal points.

### B. PATH-PLANNING ALGORITHMS

Grid-based and sampling-based algorithms are the most widely used global path planners; for example, Dijkstra and A* and the rapidly exploring random tree (RRT) and RRT*, respectively. A grid-based search efficiently expresses obstacles and nonobstacles by dividing the space into grids. Sampling-based path planning is used for finding a path by creating multiple sample points at random and searching point-wise space, without dividing the space into grids. The selected algorithms have many advantages and disadvantages; the results are summarized in Table 1. Obstacles may appear during navigation, and the robot attempts to avoid them [20]; obstacle avoidance is inevitable during navigation, and navigation is the basic operation task for a mobile robot. The path-planning algorithm must determine where the robot is, where it should be going, and how it will interact with its environment (such as static obstacles and dynamic obstacles). Therefore, to proceed with path planning, global and local indices must be considered simultaneously. The local planner mainly responds to unpredictable situations by reflecting the conditions around the robot in real time. The global planner, on the other hand, plans the entire path and takes precedence over the local planner in path planning. If the global planner is set up properly and no visible external action takes place, a small number of calculations from the local planner can allow it to reach the goal point. Both global and local planners are important, but this study focused on the global planner. In this study, the local planner was fixed to the dynamic window approach (DWA) algorithm; thus, only global path planners were compared.

#### 1) Dijkstra algorithm

| Algorithm     | Advantage                                      | Disadvantage                                      |
|---------------|------------------------------------------------|--------------------------------------------------|
| Dijkstra [44], [46], [47] | • Visits all nodes | • No negative weights |
|               | • Shortest distance                        | • Take a long time                                 |
| A* [41], [48], [49] | • Visits some nodes                         | • No negative weights |
|               | • Fast navigation speed                     | • Possibility of path navigation failure          |
|               | • Infinite space exploration                | • Dependency of heuristic function                |
| RRT [37], [39], [52] | • Small amount of calculation               | • Sample dependencies                             |
|               | • Navigates complex environments           | • Unpredictable                                   |
| RRT* [37], [38], [40] | • Optimality                                | • Sample Dependencies                             |
|               | • Uniformity                               | • Slow convergence speed                         |
|               | • Navigates complex environments           | • Unpredictable                                   |

This article has been accepted for publication in IEEE Access. This is the author’s version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3181131
The Dijkstra algorithm [20] is a typical method for determining the shortest path and solving the optimization problem. This is a greedy algorithm, which means that when a decision must be made, it selects the best choice available at the moment. Based on a breadth-first search (BSF), this algorithm searches the path from the starting point to all points; it is a method for exploring widely before exploring deeply. The Dijkstra algorithm was used because it is difficult to express the shortest distance when considering weights.

The Dijkstra algorithm is described in Algorithm 1. The basic logic of the Dijkstra algorithm is to add the shortest point based on the start point and update the shortest distance (Table 1). The Dijkstra algorithm is suitable for use when a single source and all nodes are given. The weights of points not yet connected are infinite. The Dijkstra algorithm is defined as

\[
f(n) = g(n) + h(n)
\]  

(1)

- \( n \) is the current node on the path.
- \( f(n) \) is the cost of the shortest path of all possible paths that can go from the start node \( n_0 \) to the goal node \( G \) via the current node \( n \).
- \( g(n) \) is the cost of the path from the start node to current node \( n \).

2) A* algorithm

The A* algorithm, a grid-based path-planning algorithm, differs from other graph-based search algorithms in that it uses heuristic functions to evaluate the distance to the goal. The A* algorithm is useful when given one source and one object. An efficient algorithm needs a method for determining the fewest possible nodes when searching for the optimal path; choosing a node not on the optimal path is a waste of effort. The A* algorithm assumes that \( f(n) \) can be computed for any node \( n \) [21]; it is described in Algorithm 2.

In this study, we used the Euclidean distance heuristic [22]. The shortest path is established by comparing the sum of the weights and the heuristic function. The A* algorithm aims to find an approximation of the shortest path rather than finding the complete shortest path. In the A* algorithm, \( h(n) \) has the greatest impact on performance (Table 1). Dijkstra is a special case of the A* algorithm with \( h(n) = 0 \). The A* algorithm is defined as

\[
f(n) = g(n) + h(n)
\]

(2)

- \( h(n) \) is a heuristic function that estimates the cost of the lowest path from the current node \( n \) to the goal.

3) RRT algorithm

RRT is a sampling-based path-planning method that randomly generates multiple sample points to find the path. It generates a sufficient number of sample points in space, checks whether they collide with obstacles, and detects the free space efficiently. The key result is that the distribution of the RRT vertices converges toward the sampling distribution, which is usually uniform [23]. However, it is unlikely that the sample will exactly match the goal point; thus, a threshold must be set. RRT allows one to move through a high-dimensional space, which is almost impossible to find with a grid-based search (Table 1).

The configuration consists of a tree made up of nodes and edges. An edge is a line-connecting node. RRT is often used for motion planning because it efficiently creates paths, even in complex environments with various obstacles. Recently, many improved algorithms have been developed to overcome the various shortcomings of RRT. RRT* is a typical example.
Algorithm 3 RRT algorithm

Set up $q_{start}$, $q_{goal}$
List.push($q_{start}$)

while DISTANCE($q_{new}$-$q_{goal}$) > $d_{threshold}$ do
    $q_{rand}$ = RANDOM_NODE()
    $q_{nearest}$ = List($q_{rand}$)
    $q_{new}$ = EXTEND($q_{nearest}$-$q_{rand}$-$\gamma$)
    if $q_{new}$ != OBSTACLE then
        set $q_{nearest}$ to $q_{new}$'s parent
        List.push($q_{new}$)
    end if
end while

Algorithm 4 RRT* algorithm

Set up $q_{start}$, $q_{goal}$
List.push($q_{start}$)

while DISTANCE($q_{new}$-$q_{goal}$) > $d_{threshold}$ do
    $q_{rand}$ = RANDOM_NODE()
    $q_{nearest}$ = List($q_{rand}$)
    $q_{new}$ = EXTEND($q_{nearest}$-$q_{rand}$-$\gamma$)
    if $q_{new}$ != OBSTACLE then
        List.push($q_{new}$)
    end if
    for Neighbors of List do
        $q_{first}$ in List and cost < $q_{nearest}$
        set $q_{first}$ to $q_{new}$'s parent
        if $q_{second}$ in List and cost < $q_{new}$ then
            set $q_{second}$ to $q_{new}$'s parent
        end if
    end for
end while

RRT algorithm is described in Algorithm 3.

4) RRT* algorithm

RRT* seeks to solve the optimality problem based on RRT. It has been proven that the costs obtained using RRT* converge to the optimum solution when the number of samples is increased infinitely (Table 1). The optimal motion planning problem imposes an additional requirement that the resulting feasible path be minimized [24]. The difference from RRT is the reselection of the parent node and reconstruction of the tree, thus removing unused nodes and freeing the search algorithm to focus on improving the solution. RRT* creates a straight path, and the graph is different from the graph of RRT; it is described in Algorithm 4.

IV. EXPERIMENTAL SETUP

The experimental configuration was as follows: 1) create maps of smart farms; 2) set the way points and goal point; and 3) drive along the path. The experiment was conducted in an environment with static and dynamic obstacles. The experiment was conducted in two smart farms, reflecting the various smart farm environments. We briefly mention three experiments: Short-distance path planning with static obstacles (Case 1), long-distance path planning with static obstacles (Case 2), and path planning with dynamic obstacles (Case 3).

A. COMMON CONDITION

For the field tests, the experiment was conducted using the Jackal UGV from Clearpath Robotics, and this platform was controlled by a Jetson TX2. To acquire 3D point cloud data, a LiDAR sensor from Velodyne VLP-16 was placed above the platform as the primary sensor (Fig. 2). The VLP-16 has
TABLE 2. Performance metrics.

| Task          | Item                                                                 | Formula                                                                 |
|---------------|----------------------------------------------------------------------|------------------------------------------------------------------------|
| Navigation    | • Time to reach goal point (s)                                        | $\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$                           |
|               | • Robot position (m) and orientation (rad)                           | $S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})^2$                |
|               | • Localization error                                                | $\sqrt{S^2}$                                                          |
| Path planning | • Success rate (%)                                                   | $(\frac{\text{Number of successful autonomous driving}}{\text{Total number of experiments}}) \times 100$ |

- The localization error is calculated as $(\text{Coordinates of the goal point}) - (\text{Coordinates after autonomous driving end})$.

16 lasers with a 360-degree horizontal field of view and a 30-degree vertical field of view. The hardware architecture is described in Fig. 3(a), and the software architecture is shown in Fig. 3(b). The hardware was implemented using the robot operating system (ROS) Melodic in Ubuntu 18.04.

The experiment for short-distance path planning with static obstacles assumed a harvesting or spraying robot; Jackal goes through gates into the first tomato greenhouse for harvesting or spraying tomatoes. The experiment for long-distance path planning with static obstacles assumed a transportation robot; Jackal loads tomatoes from the first tomato greenhouse and moves to the second Korean melon greenhouse to load Korean melons. The experiment for path planning with dynamic obstacles assumed a supervising robot; Jackal avoids other working robots and enters the greenhouse.

1) Simultaneous localization and mapping

SLAM was required before proceeding with the path-planning algorithm comparison test. SLAM technology uses a 3D cartographer algorithm combined with the A* algorithm. Cartographer is a LiDAR SLAM method that uses inertial measurement unit (IMU) and LiDAR data. IMU data are used to generate an initial estimate of robot motion, while LiDAR data updates the IMU-based estimate through scan matching against the constructed map. Among SLAM algorithms, Cartographer accumulates drift but corrects for drift by closing the loop [25].

The smart farm used 3D SLAM because the outer wall is made of transparent material and allows the passage of light. The map generated using Cartographer before the experiment is shown in Fig. 4 and Fig. 5.

2) Selection of parameters

For a fully fair comparison of the performances of the algorithms being compared, the control parameters were tuned to produce the same magnitude of force for each algorithm. Algorithms are partially random, so the result is not always the same. We fixed the velocity parameters of the algorithms to either 0.5 m/s or 1.0 m/s. As a result, in Fig. 6, the accelerations of the algorithms are different, but they converge to the similar velocity. We finally set the velocity to 0.5 m/s to minimize localization errors such as robot slippage that may occur in the experimental environment.

We set the conversion tolerance to 0.5 as a common condition. Additionally, in the case of the sampling algorithm, the number of samples was set equally to 200,000 or more because many samples are needed to search a wide environment.

3) Performance metric

Algorithm performances were compared based on the following properties: time to reach the goal point [69], localiza-
tion accuracy (robot position and orientation), and success rate (Table 2). For the experimental results of all algorithms, we computed the mean and standard deviation of the time to path the goal point and the position and orientation of the robot. The time to reach the goal point includes the time taken to calculate the path and drive to the destination. The localization accuracy was calculated as the relative distance between the endpoint and the goal point. We used position and orientation coordinates as performance metrics more precisely than in previous studies because location accuracy is important [39], [68]. To the best of our knowledge, few studies have used position and orientation coordinates as performance indicators. The success rate was defined as the number of successful autonomous driving cycles divided by the total number of experiments [70], [71]. In this study, path failure means that the autonomous mobile robot cannot reach the goal point due to internal or external problems. The internal problem refers to the robot falling into an infinite loop if it cannot create a path to the goal point or update the path because it does not know its current location while driving. The external problem refers to when the robot collides with an obstacle. When this occurs, in most cases, the robot cannot reach the goal point.

B. EXPERIMENT 1: PATH PLANNING WITH STATIC OBSTACLES

A static obstacle is a fixed obstacle reflected on the map. The robot creates a map with static obstacles through SLAM and plans a global path.

| Goal point | Case 1 | Case 2 | Case 3 |
|------------|--------|--------|--------|
| Position x (m) | 7.48   | 28.28  | 9.88   |
| Position y (m) | 4.97   | 7.84   | -7.44  |
| Orientation z (rad) | 0.73   | 0.78   | -0.73  |
| Orientation w (rad) | 0.69   | 0.63   | 0.68   |

1) Description of the experimental environment

The algorithm performance was tested at the smart farm (Fig. 1) of the National Academy of Agricultural Sciences in Wangu-gun, Jeollabuk-do, Republic of Korea. This smart farm comprises a control room, loading docks for storing harvested crops, warehouses, and five greenhouses. It is an indoor smart farm, so there is no great challenge in moving the mobile robot, but robot slippage might occur.

2) Scenario

Short-distance path planning with static obstacles (Fig. 7(a)) assumed a harvesting or spraying robot. The experiment was conducted by designating the 10th rail (Table 3) of the fifth tomato greenhouse as the goal point. The robot was to move to the 6th and 10th rails of the fifth tomato greenhouse designated by the operator and harvest crops. Long-distance path planning with static obstacles (Fig. 7(b)) assumed a transportation robot. The experiment was performed by designating the 12th rail (Table 3) of the fourth Korean melon greenhouse as the goal point. According to the path specified by the operator, the crops were to be transported past the 5th rail of the fifth tomato greenhouse and moved to the 3rd and 12th rails of the fourth Korean melon greenhouse.

C. EXPERIMENT 2: PATH PLANNING WITH DYNAMIC OBSTACLES

Dynamic obstacles are unexpected objects that are not reflected on the map. The robot avoids unpredictable obstacles that appear in real time through local path planning.

1) Description of the experimental environment

The algorithm performance was tested at the smart farm (Fig. 8) of the Korea Agricultural Technology Practicalization Agency in Iksan, Jeollabuk-do, Republic of Korea. This is a semi-indoor smart farm with sand and gravel on the floor, which is simply covered with sackcloth. In such an environment, it is difficult for a mobile robot to drive and an odometry error is easy to occur.

2) Scenario

The field test with dynamic obstacles used heterogeneous robots that moved in and out of the greenhouse for work (Fig. 9). The experiment was conducted by designating the 6th rail (Table 3) of the greenhouse as the goal point. According to the path specified by the operator, the robot was to move past the 4th rail of the greenhouse to the 6th rail. Here, when
FIGURE 6. Experimental results of control parameter tuning. (a) Velocity is 0.5 m/s. (b) Velocity is 1.0 m/s.

the robot entered the greenhouse to move to rail 4, it would encounter another robot that had finished its work. Dynamic obstacle avoidance would occur at the point where the paths of the robots intersect.

V. EXPERIMENTAL RESULTS

The results of the field test path planning for the smart farm are presented in Table 5 and Figure 10. In the resulting Table 5, the best values are presented in bold. The experiment investigated which of the grid-based algorithms, Dijkstra, A*, and the sampling-based algorithms RRT and RRT* are suitable for smart farm environments. Grid-based algorithms frequently generate paths closer to obstacles compared to sampling-based algorithms (Fig. 11). They can potentially generate the shortest path, and the search speed is relatively high. Sampling-based algorithms can generate a variety of unpredictable paths and can navigate complex environments. Path planning with dynamic obstacles generates new paths when unpredictable objects are captured by LiDAR based on maps generated using SLAM (Fig. 12).

Based on Table 2, performance metrics are set up for evaluation. We are interested in the time and localization accuracy at which the autonomous mobile robot reaches its goal point when using each algorithm. Through the experimental results, it is possible to understand the characteristics of the algorithm and to check the differences according to the experimental environment.

To evaluate the algorithm proposed in this study, the benchmark function was used, as listed in Table 4. Data were analyzed using the Shapiro–Wilk test [26] to evaluate the normality of the sample distribution. The Shapiro–Wilk test, a test of normality, rejects the null hypothesis of the test of normality at the smallest sample size compared to other tests at all levels of skewness and kurtosis. If the experimental data were in the form of a normal distribution, ANOVA analysis was performed, and the other Kruskal–Wallis test was performed. If the p-value > 0.05, marked with *, the mean of all groups was considered equal, i.e., the difference between the averages of all groups was not large enough to be statistically significant. However, if p-value < 0.05, the difference between the average rankings of some groups is sufficiently large to be statistically significant.

A. EXPERIMENT 1: PATH PLANNING WITH STATIC OBSTACLES

In short-distance path planning with static obstacles, the RRT* algorithm reached the goal point the fastest, followed by the A* algorithm, and the Dijkstra algorithm took the longest time to reach the goal. These results indicate that the Dijkstra algorithm’s search speed is slow because it searches all nodes. The localization accuracy was relatively high in the A* and RRT algorithms. For short-distance path planning with static obstacles, position x was sufficiently valuable to be statistically significant.
For long-distance path planning with static obstacles, the Dijkstra algorithm reached the goal point the fastest. The A* algorithm took a long time to find the long path, indicating that the A* algorithm relies on heuristic functions and is not always faster than the Dijkstra algorithm [42], [43]. In this experiment, the RRT algorithm took the longest time to reach its goal point. The localization accuracy was relatively high in the A* and RRT algorithms. For long-distance path planning with static obstacles, the time to reach the goal point was sufficiently valuable to be statistically significant.

**B. EXPERIMENT 2: PATH PLANNING WITH DYNAMIC OBSTACLE**

In path planning with dynamic obstacles, the A* algorithm reached the goal point the fastest. The localization accuracy was relatively high in the A* and RRT algorithms. If the A* algorithm creates the shortest path according to the heuristic function, it can significantly reduce the drive time compared to other algorithms. Further, the shorter the distance is, the better the search. The A* algorithm has been studied and applied according to its purpose until recently [53] [72]. We discovered experimentally the algorithm suitable for smart farms.

**VI. DISCUSSION**

We considered the autonomous driving problem of smart farms from various perspectives.

**A. PRACTICAL AUTONOMOUS DRIVING FOR SMART FARM**

As there are various movements inside a smart farm, we considered application scenarios of practical autonomous driving technology in a smart farm environment. For the case of repetitive movements (e.g., harvesting operation) in a...
stationary environment, we propose a method for combining a technology that enables tracking a magnetic guide-line and autonomous driving technology that enables movement to the desired location in real time.
FIGURE 11. Grid-based algorithms generate a path close to the obstacle and arrive at the goal point in the relatively shortest path. A sampling-based algorithm that randomly generates nodes creates different paths. (a) Path of A* algorithm. (b) Path of RRT* algorithm.

B. COMPLEMENTED A* ALGORITHM

We devised a hypothesis for addressing the limitations of the A* algorithm. The Bellman–Ford algorithm is negatively weighted. The Dijkstra and A* algorithms are not capable of handling the negative weights that the Bellman–Ford algorithm can handle. The Bellman–Ford algorithm was not considered because this study has no negative weights [27]. Although the fast search speed is an advantage of the A* algorithm, the actual shortest path search has a high probability of failure in a complex and irregular environment [28]. Path planning failures and heuristic-dependent problems were mitigated by applying an inverse algorithm.

C. IMPROVED LOCALIZATION ACCURACY

The localization accuracy can be increased by using landmarks that use Quick Response (QR) codes or AprilTags; this was confirmed in a previous study. Using QR codes as landmarks in the smart farm environment, two landmarks were used over a short distance and three landmarks were used over a long distance. The goal point could not be reached without using landmarks, and using them allowed the goal point to be reached with a small margin of error.

D. EXPERIMENTAL ENVIRONMENT CHARACTERISTICS

Smart farms exist in various forms. We visited a well-maintained indoor smart farm and a semi-indoor smart farm.
that somewhat resembled the outdoors. A well-maintained indoor smart farm should consider slippage and narrow entrances to and from one vehicle. Additionally, in a semi-indoor smart farm, if the floor is similar to a field and the sackcloth on the floor is not fixed well, the localization error becomes more severe. In a variety of environments, both flat and bumpy road surfaces can hit the wheels, accumulating errors and resulting in inaccurate navigation [45].

VII. CONCLUSION

In this study, we evaluated path planning algorithms based on their suitability for smart farms. We compared and analyzed the well-known grid-based Dijkstra and A* algorithms and sampling-based RRT and RRT* algorithms. The three analyses were based on the following: 1) Short-distance path planning with static obstacles 2) Long-distance path planning with static obstacles 3) Path planning with dynamic obstacles. In addition, the reliability was increased by conducting the field experiment under the same conditions through the separately conducted parameter setting experiment.

In special environments such as smart farms, it is essential to minimize damage to crops, so it is more important to accurately reach the target rather than quickly. The A* algorithm showed high localization accuracy in all cases and is more predictable than the sampling-based algorithm. Considering the multi-robot, it showed the best performance in terms of time and location even in an environment with dynamic obstacles. The results of the experiment confirmed that the A* algorithm is suitable for smart farms. Furthermore, the A* algorithm is still frequently studied, and we can apply it in many different ways.

VIII. FUTURE WORK

Path planning is important for scaling to multi-robots in a smart farm. Smart farms of the future will require multiple robots to perform different roles and collaborate for various functions. In a multi-robot environment, different robots become dynamic obstacles, so we must devise better applications for controlling multiple single robots based on...
the current evaluation. The problem of path planning extends to the multi-UGV path planning and coordination problem. Each robot needs to implement SLAM and navigation, further considering planning between multiple robots. We plan to investigate a robot that harvests by placing a robot arm on a platform [6], a robot that controls spraying by attaching a sprayer, and a robot that transports harvested crops to a loading box.

The main contribution of this study is that it identified a basic algorithm suitable for autonomous driving in a smart farm, which has infinite potential for development in a specialized field. This will motivate subsequent studies to improve the overall performance of control systems from a human perspective of the design and application of system parameters.

REFERENCES

[1] Yoon, C., Huh, M., Kang, S. G., Park, J., & Lee, C. (2018, February). Implement smart farm with IoT technology. In 2018 20th International Conference on advanced communication technology (ICACT) (pp. 749-752). IEEE.

[2] Sariff, N., & Buniyamin, N. (2006, June). An overview of autonomous mobile robot path planning algorithms. In 2006 4th student conference on information and communication technology (pp. 309-314). IEEE.

[3] Planner, P. & Astrap, R. (2018). Mapping forests using an unmanned ground vehicle with 3D LiDAR and graph-SLAM. Computers and Electronics in Agriculture, 145, 217-225.

[4] Kim, J., Seol, J., Lee, S., Hong, S. W., & Son, H. I. (2020, May). An intelligent spraying system with deep learning-based semantic segmentation of fruit trees in orchards. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 3923-3929). IEEE.

[5] Zhang, J., Hou, C., Tang, Y., He, Y., Guo, Q., Zhang, Z., & Luo, S. (2019). Computer vision-based localisation of picking points for automatic litchi harvesting applications towards natural scenarios. Biosystems Engineering, 187, 1-20.

[6] Jun, J., Kim, J., Seol, J., Kim, J., & Son, H. I. (2021). Towards an Efficient Tomato Harvesting Robot: 3D Perception, Manipulation, and End-Effectector. IEEE Access, 9, 17631-17640.

[7] Masuzawa, H., Miura, J., & Oishi, S. (2017, December). Development of a mobile robot for harvest support in greenhouse horticulture—Person following and mapping. In 2017 IEEE/SICE International Symposium on System Integration (SII) (pp. 541-546). IEEE.

[8] Kitamura, S., & Oka, K. (2005, July). Recognition and cutting system of sweet pepper for picking robot in greenhouse horticulture. In IEEE International Conference Mechatronics and Automation, 2005 (Vol. 4, pp. 1807-1812). IEEE.

[9] Belforte, G., Deboli, R., Gay, P., Piccarolo, P., & Aimonino, D. R. (2006). Robot design and testing for greenhouse applications. Biosystems Engineering, 95(3), 309-314.

[10] Simmons, P. J., Furukawa, T., & Bulgin, A. (2005, December). Autonomous pesticide spraying robot for use in a greenhouse. In Australian Conference on Robotics and Automation (Vol. 1, No. 9).

[11] Tokekar, P., Vander Hook, J., Mulllar, D., & Isler, V. (2016). Sensor planning for a symbiotic UAV and UGV system for precision agriculture. IEEE Transactions on Robotics, 32(6), 1498-1511.

[12] Kim, J., & Son, H. I. (2020). A voronoi diagram-based workspace partition for weak cooperation of multi-robot system in orchard. IEEE Access, 8, 20676-20686.

[13] Tang, Y., Chen, M., Wang, C., Luo, L., Li, J., Lian, G., & Zou, X. (2020). Recognition and localization methods for vision-based fruit picking robots: A review. Frontiers in Plant Science, 11, 510.

[14] Bergerman, M., Maeta, S. M., Zhang, J., Freitas, G. M., Hamner, B., Singh, S., & Kantor, G. (2015). Robot farmers: Autonomous orchard vehicles help tree fruit production. IEEE Robotics & Automation Magazine, 22(1), 54-63.

[15] Cadena, C., Carloni, L., Carrillo, H., Latifi, Y., Scaramuzza, D., Neira, J., & Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on robotics, 32(6), 1309-1332.

[16] Santos, J. M., Portugal, D., & Rocha, R. P. (2013, October). An evaluation of 2D SLAM techniques available in robot operating system. In 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) (pp. 1-6). IEEE.

[17] Hess, W., Kohler, D., Rapp, H., & Andor, D. (2016, May). Real-time color frame processing in 2D LIDAR SLAM. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 1271-1278). IEEE.

[18] Filonenko, M., & Afanasyev, I. (2018, September). Comparison of various slam systems for mobile robot in an indoor environment. In 2018 International Conference on Intelligent Systems (IS) (pp. 400-407). IEEE.

[19] Grimstad, L., Zakaria, R., Le, T. D., & From, P. J. (2018, October). A voronoi diagram-based workspace partition for weak cooperation of multi-robot system in orchard. IEEE Access, 6, 20676-20686.

[20] Wang, H., Yu, Y., & Yuan, Q. (2011, July). Application of Dijkstra algorithm in robot path-planning. In 2011 second international conference on mechatronics automation and control engineering (pp. 1067-1069). IEEE.

[21] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics, 4(2), 100-107.
[70] Li, Q., Lin, W., Liu, Z., & Prorok, A. (2021). Message-aware graph attention networks for large-scale multi-robot path planning. IEEE Robotics and Automation Letters, 6(3), 5533-5540.

[71] Li, Q., Gama, F., Ribeiro, A., & Prorok, A. (2020, October). Graph neural networks for decentralized multi-robot path planning. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 11785-11792). IEEE.

[72] Candra, A., Budiman, M. A., & Hartanto, K. (2020, July). Dijkstra’s and A-Star in Finding the Shortest Path: a Tutorial. In 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA) (pp. 28-32). IEEE.