Development of Unmanned Aerial Vehicle Navigation and Warehouse Inventory System Based on Reinforcement Learning

Huei-Yung Lin *, Kai-Lun Chang , Hsin-Ying Huang

Posted Date: 23 April 2024

doi: 10.20944/preprints202404.1526.v1

Keywords: UAV; Indoor Positioning; Path Planning; Warehouse Inventory Inspection; Reinforcement Learning

 doi:10.20944/preprints202404.1526.v1

Article

Development of Unmanned Aerial Vehicle Navigation and Warehouse Inventory System Based on Reinforcement Learning

Huei-Yung Lin [1], Kai-Lun Chang [2] and Hsin-Ying Huang [3]

[1] Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei 106, Taiwan; lin@ntut.edu.tw
[2] Department of Electrical Engineering, National Chung Cheng University, Chiayi 621, Taiwan; kailun@godel.ee.ccu.edu.tw
[3] Department of Electrical Engineering, National Chung Cheng University, Chiayi 621, Taiwan; hsinying@godel.ee.ccu.edu.tw
* Correspondence: lin@ntut.edu.tw; Tel.: (optional; include country code; if there are multiple corresponding authors, add author initials) +xx-xxxx-xxx-xxxx (F.L.)

Abstract: In this paper we present the exploration of indoor positioning technologies for UAVs, and navigation techniques for path planning and obstacle avoidance. The objective is to perform warehouse inventory tasks using a drone to search for barcodes or markers to identify the objects. For the indoor positioning techniques, we employ Visual-Inertial Odometry (VIO), Ultra-Wideband (UWB), AprilTag fiducial markers, and Simultaneous Localization and Mapping (SLAM). These algorithms encompass global positioning, local positioning, and pre-mapping positioning, comparing the merits and drawbacks of various techniques and trajectories. In UAV navigation, we combine SLAM-based RTAB-Map indoor mapping and navigation path planning of the ROS for indoor environments. This system enables precise drone positioning indoors and utilizes global and local path planners to generate flight paths that avoid dynamic, static, unknown, and known obstacles, demonstrating high practicality and feasibility. To achieve the warehouse inventory inspection, a reinforcement learning approach is proposed to recognize the markers by adjusting the UAV's viewpoint. We address several main problems in inventory management, including efficiently planning paths while ensuring a certain level of detection rate. Two reinforcement learning techniques, AC (Actor-Critic) and PPO (Proximal Policy Optimization), are implemented based on AprilTag identification. Testing is performed in both simulated and real-world environments, and the effectiveness of the proposed method is validated. Code is available at https://github.com/kellen080/Navigation and https://github.com/kellen080/Indoor_Positioning.

Keywords: UAV; Indoor Positioning; Path Planning; Warehouse Inventory Inspection; Reinforcement Learning

1. Introduction

In recent years, drone related technologies have attracted a widespread attention across various fields. Drones are known for their lightweight, flexibility, low cost, and efficiency, which allow them to perform diverse tasks without the requirements of human intervention. The rapid developments and potential applications of the technologies have captured close interests of academia and industry alike. Several applications of drones include disaster monitoring, environmental surveillance, agriculture, logistics, security monitoring, etc. With the continuous developments of drone technologies, it brings immense potential for applications in many fields. But it is worth noting that most current applications are outdoor-based predominantly due to the limited availability of indoor positioning techniques. As the demand for the indoor usage of drones increases, precise positioning and stable flight control become crucial for promoting indoor drone applications. Thus, in this paper we study and explore the indoor drone positioning and path planning technologies to enhance flight accuracy and control capabilities in indoor environments. This enables more diverse indoor drone applications to meet the demands across different domains. Moreover, an inventory management system based on the development of UAV with reinforcement learning techniques is developed.

The reinforcement learning has been investigated for decades, and proven its effectiveness in playing many video games [1]. Since the operations and strategy are similar to the applications of

 doi:10.20944/preprints202404.1526.v1

UAVs, it is gradually adopted for flight control to perform specific tasks. For the use on warehouse inventory, it is possible to let the UAVs learn how to move with a certain viewing angle and recognize the interested items [2]. With the deployment of UAVs, the warehouse inventory system can then access dangerous or hard-to-reach places. On the other hand, reinforcement learning is a method based on the interaction with environments to obtain feedbacks. The advantage of this machine learning approach is that the input data does not need to be labeled. Despite the requirement on long training time, it has been successfully used in many real-world scenarios. Among the reinforcement learning strategies, this paper adopts AC (Actor-Critic) [3] and PPO (Proximal Policy Optimization) [4] to construct the environmental conditions, action generation, scoring methods, etc. for training. In our application, the design of the environmental conditions is the feature extraction of images. Based on the current situation, the generated actions consist of two types: moving or changing the camera angle. In addition to performing the basic task on tag identification, the scoring will also include other auxiliary methods such as considering the operation cost.

1.1. Indoor Positioning

The indoor positioning refers to the technology that achieves precise location tracking of a mobile device (such as a drone or a smartphone) or people within buildings or other enclosed spaces. Compared to the outdoor positioning technology (such as GPS), indoor positioning techniques face greater challenges due to complex indoor environments, including signal interference and the inability to rely on satellite-based positioning. Typically, a combination of techniques such as visual features, inertial measurement units, and simultaneous localization and mapping (SLAM) is used to achieve redundant and accurate indoor positioning. Current popular approaches include the use of wireless signals, fiducial marker, inertial measurement unit, and simultaneous localization and mapping from environment perception.

Commonly adopted techniques utilizing wireless signals to measure the position and orientation of an object in indoor environments include UWB (Ultra-Wideband) [5], BLE (Bluetooth Low Energy) [6], and Wi-Fi [7]. Usually, these approaches measure the signal-based parameters such as the time of arrival (TOA), time difference of arrival (TDOA), two-way ranging (TWR), or angle of arrival (AOA). Based on the principles of triangulation, signal propagation models, or fingerprint data, the location of objects can be calculated. The advantages of these methods include low power consumption, cost-effectiveness, easy deployment, and maintenance. It has effectively addressed issues related to multi-path effects [8] and signal penetration through obstacles, providing high accuracy and reliability in positioning. These technologies find applications in various fields, including navigation, tracking, security, and smart homes.

Fiducial markers are small and distinctive patterns or symbols used as reference points in computer vision systems. They are placed in an environment to assist the system for determining the position and orientation of objects when the system has difficulty solely relying on the appearance of the objects for localization. Fiducial markers find numerous applications in fields such as robotics, augmented reality, and computer vision, including indoor positioning, robot navigation, and architectural measurements, etc. In general, fiducial markers possess characteristics such as fixed positions, prominent features, and uniqueness. Fiducial markers can present in various forms, including 2D or 3D patterns, or simple colored dots. Among them, the most common types of fiducial markers are 2D barcode-like patterns, such as AprilTag [9], ArUco [10], and ARToolKit [11]. The markers can be scanned or recognized by sensors like cameras and radars, and their known geometric structures and sizes are used to determine their positions and orientations within the scene.

The visual-inertial odometry (VIO) is a positioning technology utilizing both cameras and inertial measurement units (IMU). A camera provides a series of image data, while the IMU measures the acceleration and angular velocity of the device. The key of VIO techniques lies in fusing the data from camera and IMU to eliminate the drift issues commonly encountered when using the visual odometry (VO) alone. Specifically, the VIO fuses the visual information from the camera with the acceleration

 doi:10.20944/preprints202404.1526.v1

and gyroscope data from the IMU to obtain a more accurate camera trajectory, enabling precise and robust tracking of the device's motion over time. The VIO technology offers advantages such as high accuracy and real-time performance, allowing it to operate in real-time in various environments. Consequently, VIO has widespread applications in many fields such as computer vision, robotics, and autonomous navigation. Several commonly adopted VIO techniques include OKVIS [12], MSCKF [13], and ROVIO [14].

Simultaneous Localization and Mapping (SLAM) is a computational technique used in robotics and computer vision. The objective is to construct an environment map and estimate the robot's or sensor's position simultaneously in the unknown environment using information sensed by the moving robot or sensors (such as stereo cameras, radars, and LiDARs). In an unknown space, SLAM techniques enable the robot to detect its own pose in real-time while simultaneously creating a detailed map during the process. Over the years, SLAM has garnered widespread attention and research. Some popular SLAM techniques include RTAB-Map [15], VINS-Mono [16], and ORB-SLAM2 [17]. Implementing a SLAM system is complex and requires consideration of various factors, such as sensor accuracy and environmental uncertainty. Nevertheless, SLAM is crucial for achieving autonomous navigation in mobile robotics, and has significant applications in unmanned aerial vehicles (drones), autonomous vehicles, and robotic systems [18].

1.2. Path Planning

Path planning is generally used in robotics and autonomous navigation to derive a path from a starting point to a destination while considering the obstacles and possible constraints in the environment. Its goal is to determine an optimal trajectory or path that is safe, efficient, and satisfies specific conditions. In the applications, path planning is applied in various domains, such as robotics, autonomous navigation, and unmanned aerial vehicles (drones) [19].

Global path planning typically involves generating maps or representation of the environment. It utilizes search algorithms to find an optimal path in robot's configuration space while avoiding obstacles and adhering to environmental constraints as much as possible. The real-time environmental influences or the robot's actual motion state are not taken into account in general. The global path planning is an offline process. It requires path planning and map building before the robot is set in motion. This would allow the robot to quickly obtain path planning results during task execution and avoid encountering unknown obstacles that may lead to mission failure. In global path planning, the structure and layout of the environment are typically considered as a whole, including the starting and ending positions, map information, constraint conditions, cost functions, and other information. A few commonly used approaches include graph-based methods [20], genetic algorithm-based methods [21], and deep learning-based methods [22]. The output is usually a high-level path or trajectory, followed by a low-level controller or motion planner to execute the planned path.

Local path planning refers to the process of calculating the next action the robot will take based on its current position, orientation, and information about the surrounding environment during its movement. It is used to determine the optimal path or trajectory for a moving object in a small region of the environment, usually around the object's current position, to enable the robot to reach the target point while ensuring safety flight. In practice, global path planning and local path planning are often combined. The global path planning determines the general direction for the movement, while the local path planning continuously adjusts the robot's motion trajectory in real-time during its movement. Several well-known local path planning techniques include dynamic window approach (DWA) [23], artificial potential field [24], and methods based on virtual constraints [25].

1.3. Warehouse Inventory Inspection

Currently, the logistics industry is extremely developed, and a number of solutions have been investigated from warehouse management. Among them, the warehouse inventory is considered a specifically important part of warehouse management. The main task is to confirm the correct quantity

 doi:10.20944/preprints202404.1526.v1

and types of cargos. To ensure the correctness and traceability of quantity, it is necessary to have a timely update on the records of cargos. In conventional methods, stocktaking is generally carried out by manpower, which is a time-consuming, labor-intensive and error-prone task. Nevertheless, the automation solutions have been introduced for the warehousing system in the process of industrial automation to improve efficiency and reduce costs. The techniques for reducing the inventory time include the use of new scanning, sensing technologies, and robot guidance planning, etc.

In [26], Kalinov et al. presented an autonomous heterogeneous robot system for warehouse inventory [26]. The tasks of UGR and UAV are global positioning, navigation and barcode scanning. In addition to regular cameras, the UAV is equipped with a laser scanner for barcode scanning. It keeps flying above the UGR for power supply and data transmission. For real-time barcode detection and reading, convolutional neural networks (CNNs) are commonly adopted [27]. Upon the barcode location with respective to the UAV is known, it can be used to generate a barcode map and optimize the path for inventory investigation [28]. In most applications, the motion path of the UAV can be derived from the traveling salesman problem, and unconfirmed barcodes are used future path planning [29]. Once new barcodes are detected, they will be verified and included to the database, followed by the position update and trajectory optimization of the UAV.

Cristiani et al. presented an inventory management system using mini-drones [30]. Their proposed architecture consists of four parts: intelligent warehouse, one or more drones, ground control station (GCS) and ground charging station (GRS). The intelligent warehouse contained shelves, aisles, and cargo with unique labels. A ground control station was used to control the flight path, as well as transmit the inventory data through WiFi. When performing the scanning task, the UAV moves vertically in a zigzag shape to detect the labels on all shelf units. Yoon et al. presented a technique to scan products and estimate the 3D position of the drone [31]. The barcodes were detected with a trained segmentation model. They have reported good success rates on localization of QR codes. More recently, Rhiat et al. presented the idea of "smart shelf" to optimize inventory management [32]. They employed a mobile robot with gripper for navigation and manipulation. iBeacon and RFID were used identify the items on the shelf.

The application scenario of the proposed inventory management system contains storage boxes placed on a double-layer shelf. Unlike most developments for warehouse inventory [33,34], our target items are placed randomly with AprilTags [35] attached for detection and recognition. Due to the motion path of the UAV, AprilTags might be blocked from certain camera viewpoints. In this paper, we propose a path planning method based on partially observable markers, with the identification for orientation estimation of the camera. The ROIs (regions of interest) are first identified using an object detection network with the consideration of various factors including occlusion, image blur, visible size and 3D pose. In addition, the K-means clustering is adopted to group the feature points, and predict the location of AprilTag. With the implementation based on AprilTag 2 [36], we are able to perform the detection at a high frame rate for far range markers.

2. Localization and Navigation

The proposed method in this work includes two main components, *indoor mapping and localization*, and *path planning with obstacle avoidance*. In indoor mapping and localization, various global and local indoor positioning techniques, including UWB, QVIO, AprilTag, and RTAB-Map SLAM, are used for multi-sensor fusion. The system architecture is shown in Figure 1, with the indoor positioning module in the left. We also demonstrate the indoor localization trajectories, and compare the performance of each sensor. In path planning and obstacle avoidance, the ROS Navigation Stack's global path planner is utilized to find the optimal path, while the local path planner is used to avoid unknown obstacles, as illustrated in the middle of Figure 1. The integration of the RTAB-Map [15], a visual feature-based indoor localization technique, is carried out, reducing costs by using a low-cost camera. Finally, it is integrated with MAVROS, PX4 Autopilot, and QGroundControl, enabling the unmanned aerial vehicle

 doi:10.20944/preprints202404.1526.v1

(UAV) to fly autonomously. As shown in the right of Figure 1, the system can achieve indoor mapping, localization, path planning and obstacle avoidance capabilities.

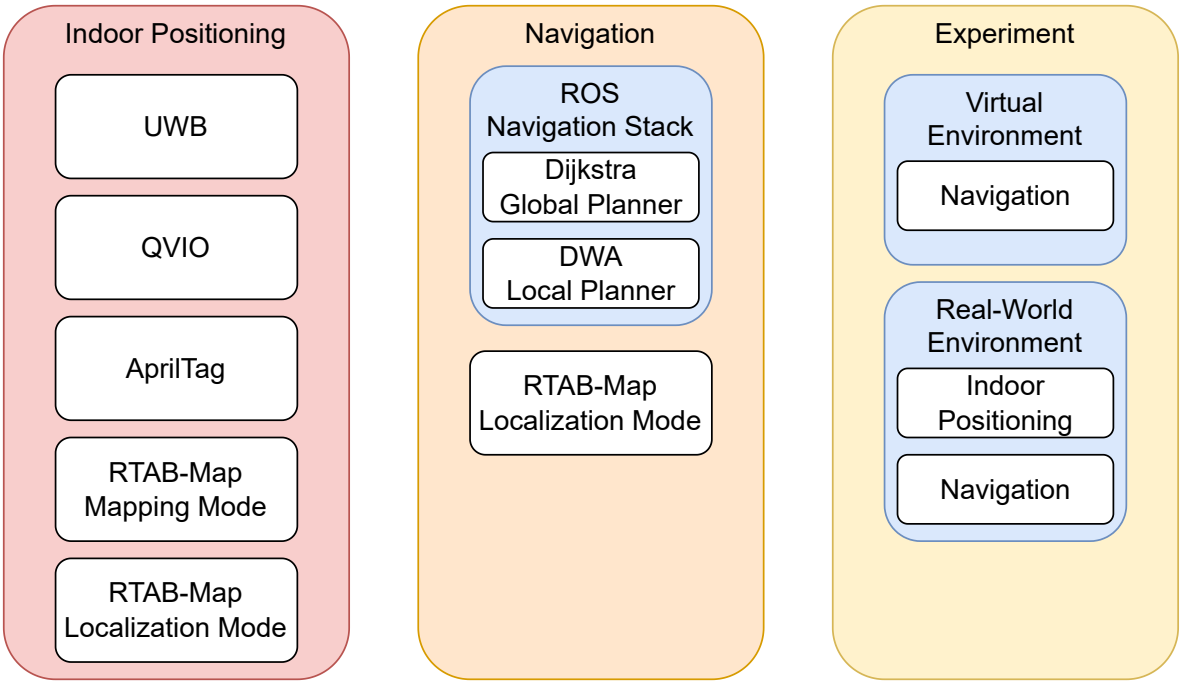


Figure 1. The system architecture of our localization and navigation techniques for UAV.

2.1. Indoor Mapping and Positioning

The experiments of this work are conducted in a laboratory environment with dimensions of 800 cm in length, 700 cm in width, and 250 cm in height. The environment setup includes using items with rich visual features, obstacles used for drone obstacle avoidance testing, placing UWB anchors around the perimeter of the environment, and having AprilTag positioning markers mounted on the ceiling. These settings are used to test and validate the indoor positioning and navigation algorithms. In indoor mapping and localization, many positioning methods are compared, as depicted in Figure 2. After environment setup, the different positioning techniques are activated, and the pose obtained from each sensor is converted into a path. Finally, the paths are displayed on RViz for visualization.

We use the mvVISLAM (machine vision Visual-Inertial SLAM) algorithm, QVIO (Qualcomm VIO), provided by Qualcomm machine vision SDK (mvSDK) as our local positioning method. QVIO employs the Extended Kalman Filter (EKF) to fuse data from the IMU and camera tracking, which result in 6 DOF pose estimation in real-world coordinates. Utilizing a fish-eye tracking camera mounted at a 45-degree downward angle on the front of the drone, visual feature extraction is performed on the captured images. With experiments, it has been observed that even though the images may include the drone's landing gear, the stability of the QVIO (Visual-Inertial Odometry) system is not significantly affected as long as there are sufficient feature points in the images. On the contrary, if lacking feature points, QVIO becomes more susceptible to losing odometry due to the drone's momentary acceleration.

The principles of UWB global positioning technology are similar to the GPS satellite. By deploying a number of known-coordinate positioning anchor points (UWB Anchors) indoors and placing a positioning tag (UWB Tag) on the object to be located, the tag continuously emits pulses at a certain frequency and used to measure the distance to each anchor. Finally, through an algorithm, the current position of the tag is determined.

 doi:10.20944/preprints202404.1526.v1

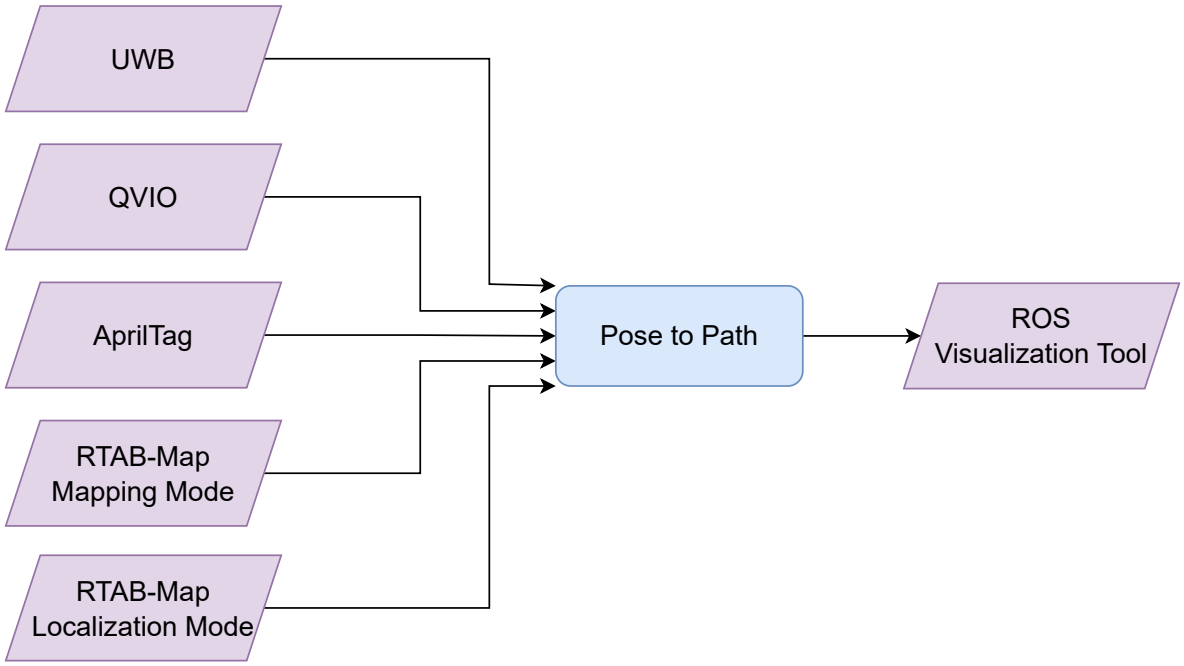


Figure 2. The system flowchart of the proposed technique integration for positioning.

In this work, we use AprilTag fiducial markers of Tag36h11 category for precise global indoor positioning. These AprilTags are placed on the indoor ceiling. ROS wrapper:apriltag_ros provides AprilTag fiducial marker detection algorithm, allowing us to detect the marker ID and its current pose with respective to the camera. It is then published on the ROS platform for further use. We also implement functions for tag bundle detection and tag bundle calibration, enabling the detection of multiple tags in a single or consecutive images.

rtabmap_ros is a ROS wrapper for RTAB-Map, that is a RGB-D SLAM method based on the real-time constraints and loop closure detection. It is capable of generating 3D point cloud maps and creating 2D occupancy grid maps for navigation. RTAB-Map can be used to build a map of the environment by fusing data from RGB-D sensors, and it employs loop closure detection to optimize the map and improve localization accuracy.

2.2. Path Planning with Obstacle Avoidance

In indoor path planning and obstacle avoidance for UAVs, we first construct 2D occupancy grid maps and 3D point cloud maps of the indoor environment using RTAB-Map before the flight. During the flight, the UAVs localize itself in the absence of GPS signals indoors by performing visual odometry based on the current visual input and simultaneously matching it with the 3D point cloud map. This process involves loop closure detection, which helps to eliminate the accumulated errors in visual odometry over time, as shown in the left of Figure 3.

 doi:10.20944/preprints202404.1526.v1

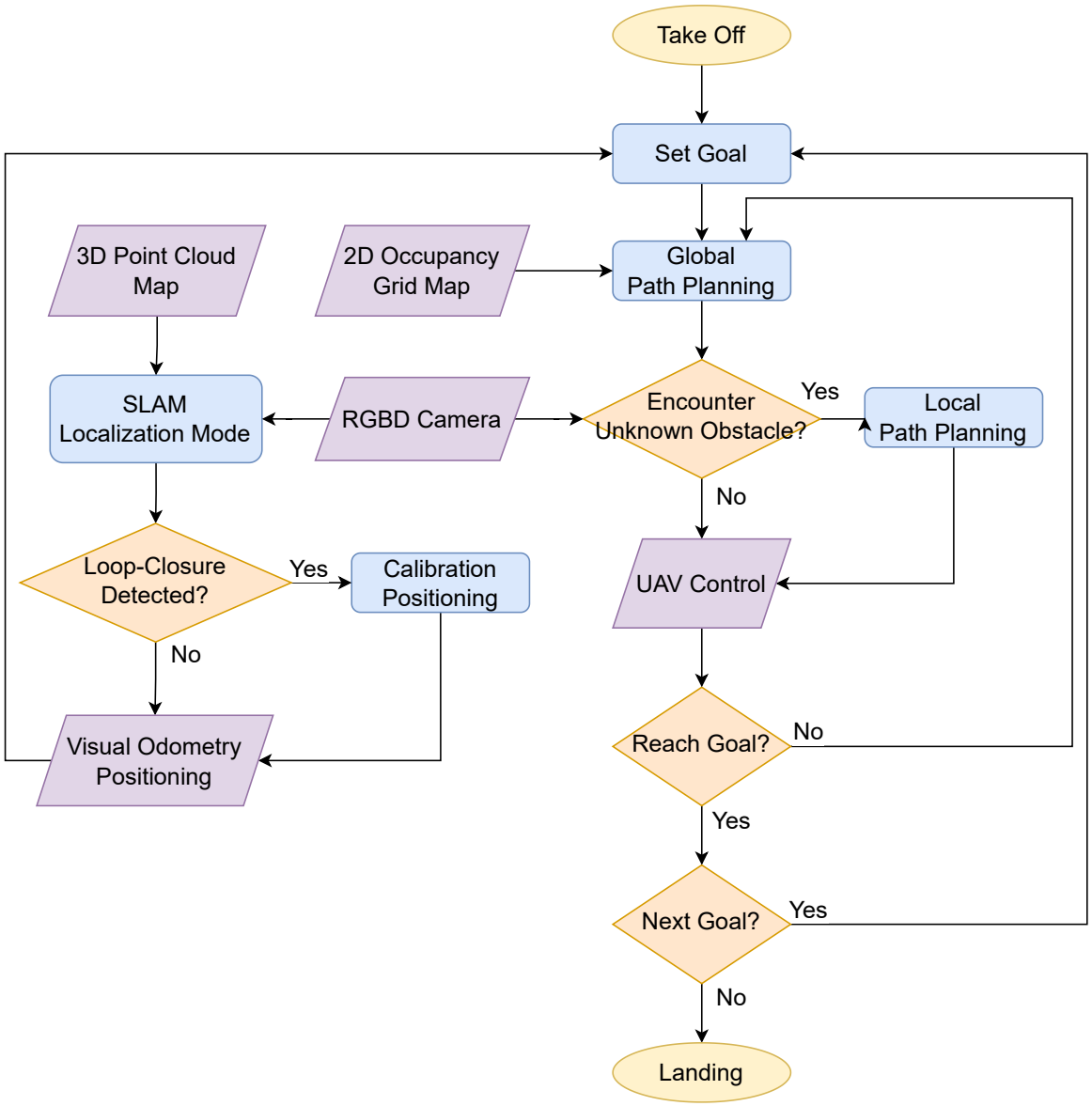


Figure 3. The flowchart of our proposed navigation system.

In path planning and obstacle avoidance for UAVs, we use ROS Navigation Stack as a framework for the entire navigation system. The global path planner derives an optimal path from the current position to the specified destination, by taking into account the known obstacles shown in the 2D occupancy grid map. This helps avoid dead ends that might be local optimal solutions. In addition, the local path planner, with the use of depth camera, enables the UAV to avoid the dynamic and static unknown obstacles which are not marked on the 2D occupancy grid map while flying along the path previously obtained from the global path planner.

Navigation Stack is conceptually straightforward. It takes current pose and external environmental information from odometry and sensors, respectively, and outputs velocity commands to drive the robot. However, it is not possible to apply Navigation Stack to a robot without some prerequisites. Before using Navigation Stack, the robot must be set up on the ROS platform, with a correctly configured *tf* transform tree, and publish sensor data using the appropriate ROS message types. Moreover, Navigation Stack requires the configuration tailored to the shape and dynamics of the robot to fully leverage its capabilities.

In global_planner, we utilize the algorithm based on *navfn* (navigation function) developed in [37]. *navfn* provides a fast interpolation navigation function that can be used to plan global paths for the

 doi:10.20944/preprints202404.1526.v1

robot. The grid-based global path planner assumes a robot operating on a 2D occupancy grid map and uses the Dijkstra algorithm [38] to find the lowest-cost global path from the starting point to the destination within the grid. On the other hand, in local_planner, we employ the algorithm based on dwa_local_planner, developed in [23]. It provides the implementation of Dynamic Window Approach (DWA) technique for local path planning in robot navigation. Given a 2D cost map and the global path to follow, the local planner generates velocity and angular velocity commands and sends them to the robot. The DWA algorithm takes the robot's dynamics and the surrounding environment into account, so to dynamically adjust the robot's velocity and avoid obstacles while following the global path.

3. Warehouse Inventory Inspection

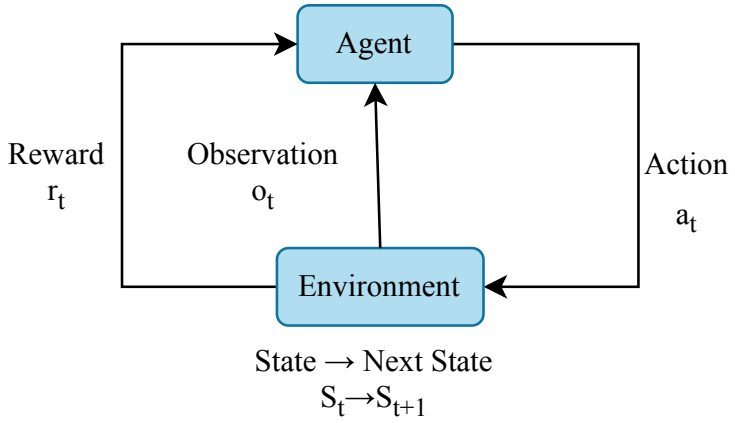
The system architecture of the proposed inventory management technique is illustrated in Figure 6. First, the UAV acquires images from the environment at discrete sampling points with various camera positions and orientations. We collect images in both virtual and real-scene environments as the inputs for training or testing. The virtual environment is built with shelves and aisles for UAV navigation, where all sampling angles are captured by a simulated drone. The camera is mounted on the drone, which flies at a fixed height (4.5m) and records images at a downward angle of 35°. In the real environment, images are taken from fixed positions within a area using a camera mounted on a mobile platform, with no downward perspective. The testing process is similar to the training process, with the distinction that all parameters are fixed, and each step's images are saved.

3.1. Proposed Reinforcement Learning Framework

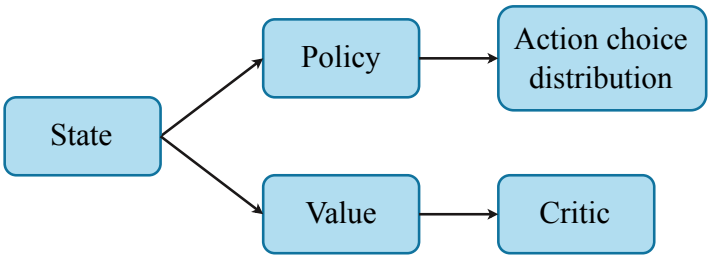
Reinforcement Learning is a learning method based on interacting with the environment to receive feedback. One advantage of this type of machine learning is that the input data do not need to be labeled, but it typically requires longer training time. The algorithms can generally be divided into two types: on-policy and off-policy. In on-policy methods, the actor interacts with the environment and performs learning and updates with the same policy throughout the process. Examples include A3C [39] and DQN [40]. Off-policy methods have separate policies for interacting with the environment and learning. PPO [4] is an example of an off-policy method. In this work we use both framework to construct own environment conditions, action generation and scoring methods for training and testing. The environment conditions are designed for feature extraction from images, and actions are generated based on the current situation, involving movement or changes in camera viewpoints.

AC (Actor-Critic) is an on-policy method, and the basic framework is shown in Figure 4(a). The agent observes the current environment and obtains a set of current information (observation). Then, the agent generates an action to modify the environment and produce the next set of states (next state). At the same time, the environment provides feedback (reward) to the agent based on the actions given by the agent for the next set of states. The feedback is then returned to the agent as the basis for adjusting internal values. The internal structure of the actor is shown in Figure 4(a). After the current state input, it is divided into two parts for processing. First, the current state is processed through the policy network to output a probability distribution of action selection. The current state is then processed through the value network to output an evaluation value (critic). Here, the evaluation value represents the expected cumulative reward in this state. It takes the action provided by the agent as input and outputs the next state. The feedback (reward) for choosing this action is in the current state and whether the data collection for this round completed after internal processing.

 doi:10.20944/preprints202404.1526.v1



(a) The schematic diagram of AC framework.



(b) The internal structure of AC framework.

Figure 4. The schematic diagram and internal structure of AC (Actor-Critic) framework.

Figure 5 illustrates the framework of PPO (Proximal Policy Optimization). It is divided into two parts: Actor (top) and Critic (bottom). In the internal of the actor, there are two policies with the same structure but used for different purposes. One is responsible for interacting with the environment, and the other is responsible for learning and updating. They are separated because one parameter is trained based on the data collected by another parameter. This allows the reuse of collected data to update the parameter multiple times, thereby improving training efficiency. PPO does not calculate the KL divergence in its update but instead restrict its range. In linking action selection with the learning strategy, PPO uses importance sampling to transform between the two distributions. This method enables the data collected after executing a set of action selections to be used for multiple training sessions. In the internal structure of Critic, the loss function is the difference between true and estimated values. The true value is obtained with the next state and feedback as input, while the estimated value is obtained with the current state as input.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 April 2024 doi:10.20944/preprints202404.1526.v1

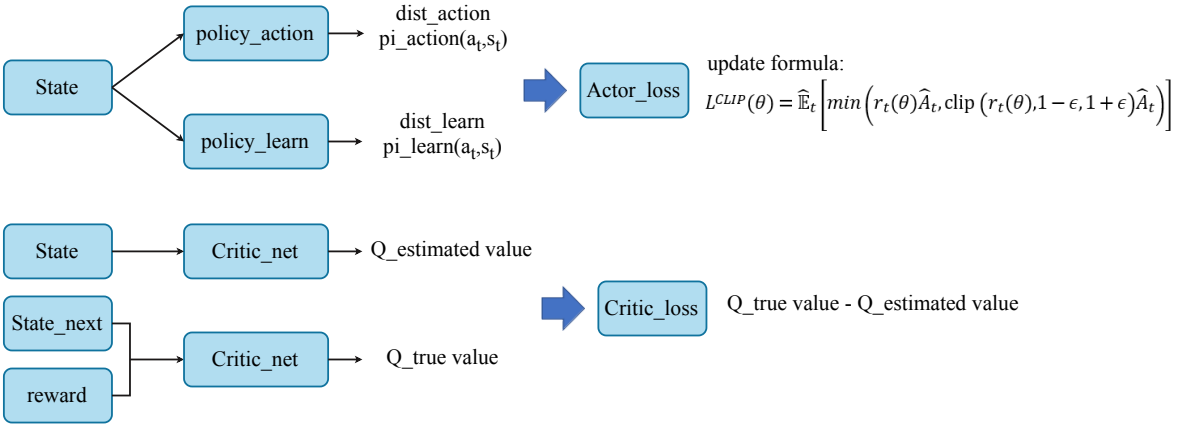
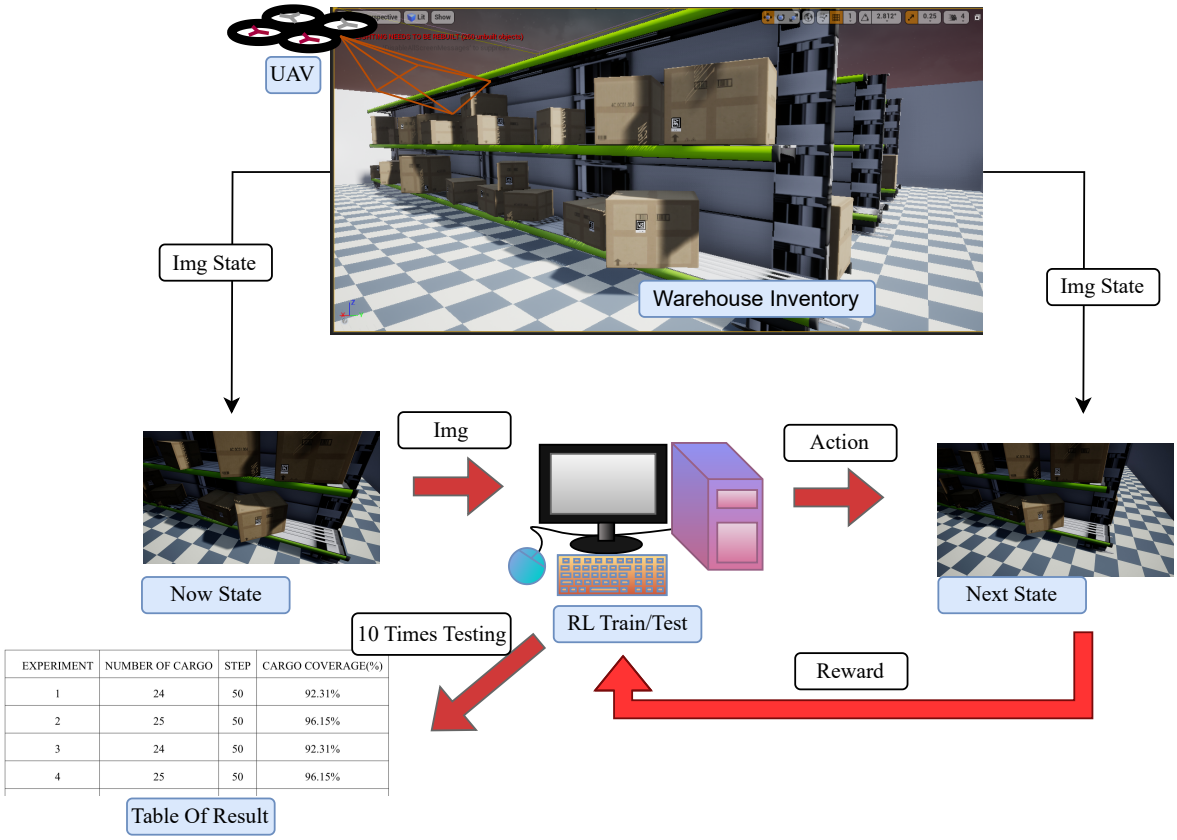


Figure 5. The framework of PPO (Proximal Policy Optimization). It consists of two parts: Actor (top) and Critic (bottom).

Figure 7 depicts the system flowchart of our reinforcement learning training framework. During the training stage, a set of random coordinates, angles, and images are used as input of the initial state. If the amount of data is sufficient, learning and parameter update will be carried out; otherwise, the data collection process will continue. The images are first standardized in size, then followed by feature extraction using ResNet34 [41]. Using the features to learn the policy and critic, a predicted evaluation value and the probability distribution of actions are obtained respectively. We check the action range and output the next state and reward value according to the action selection If the task is complete, network training is performed based on the structure of different methods; otherwise, more data are collected continuously. Finally, after completing all rounds, observe the training results according to the rewards accumulated in different rounds.

 doi:10.20944/preprints202404.1526.v1



EXPERIMENT	NUMBER OF CARGO	STEP	CARGO COVERAGE(%)
1	24	50	92.31%
2	25	50	96.15%
3	24	50	92.31%
4	25	50	96.15%

Figure 6. The architecture of the proposed warehouse inventory management system. The images acquired in a setup environment are used as inputs for training or testing. The data collection is divided into virtual and real scene cases. In the virtual environment, a simulated drone acquires images from various sampling angles within shelves. In the real environment, images are taken from fixed points within a setup area using a camera mounted on a mobile platform. The testing process is identical to training, with the difference being that all parameters are fixed, and each step's images are saved. The testing is conducted ten times, and the results are summarized in Table 4.

Figure 8 illustrates the flowchart of our reinforcement learning architecture. After training is completed, the trained parameters are stored and imported, and use test data as input to test the same structure. Since the input of reinforcement learning does not have label characteristics, there is no standard basis. However, according to our objective, the evaluation method is set here as the detection rate of the quantity of cargo and the time (number of steps) used.

 doi:10.20944/preprints202404.1526.v1

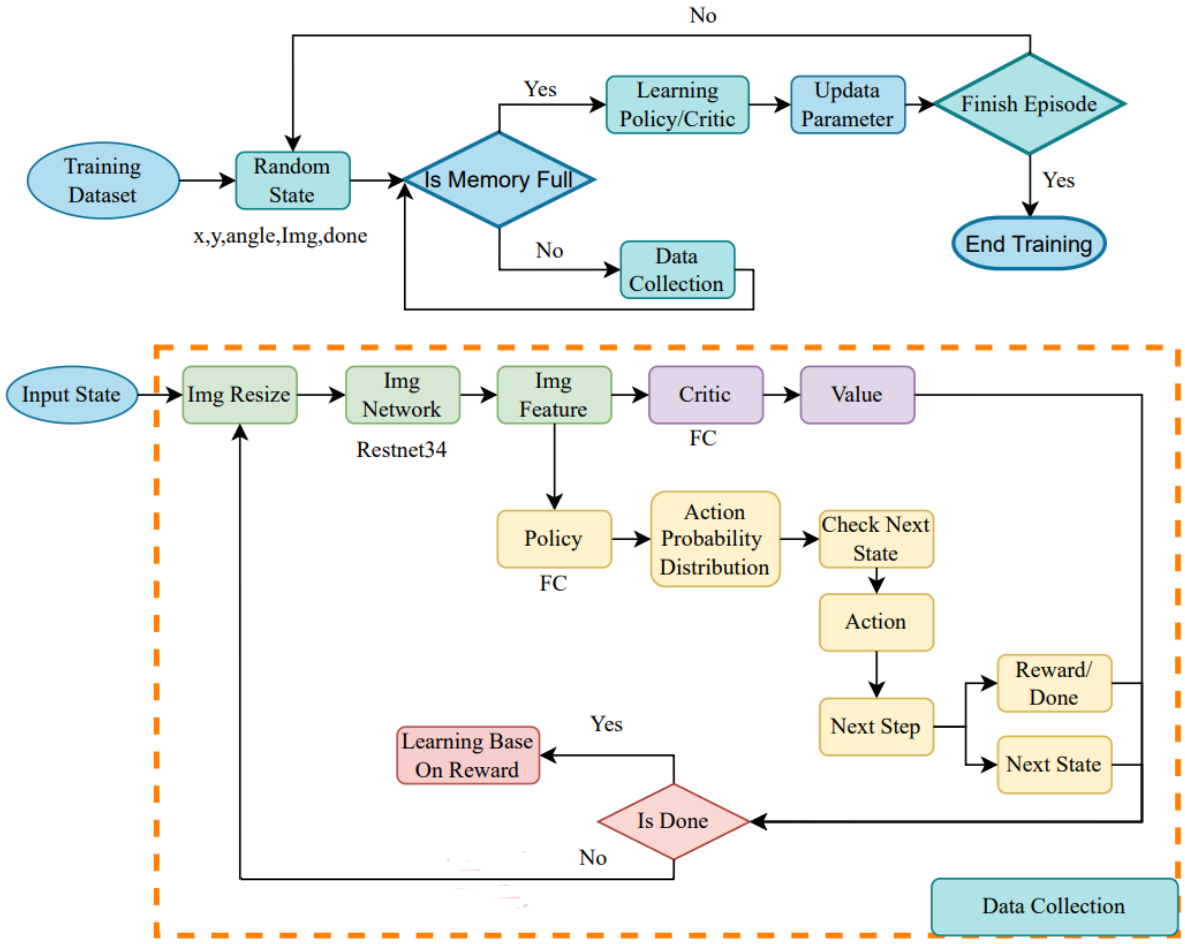


Figure 7. The system flowchart of our reinforcement learning training framework. A set of random coordinates, angles, and images are taken as initial input. The system checks if sufficient data have been collected. If not, data collection continues. If enough data are collected, the learning process begins, and parameters are updated upon completion. Before feature extraction in the input training framework, images are normalized in size. The feature extraction network, ResNet34, is used, and its output is divided into two parts: Policy and Critic. The latter predicts an evaluation value, while the former outputs a probability distribution of actions. Action range checking is performed next, followed by the selection of the next state and reward output according to action selection rules. The system then checks if the task is complete, i.e., if all targets in the scene have been found. If not, a series of continuous state collections continues. Finally, the training results are observed based on the accumulated rewards from each round.

 doi:10.20944/preprints202404.1526.v1

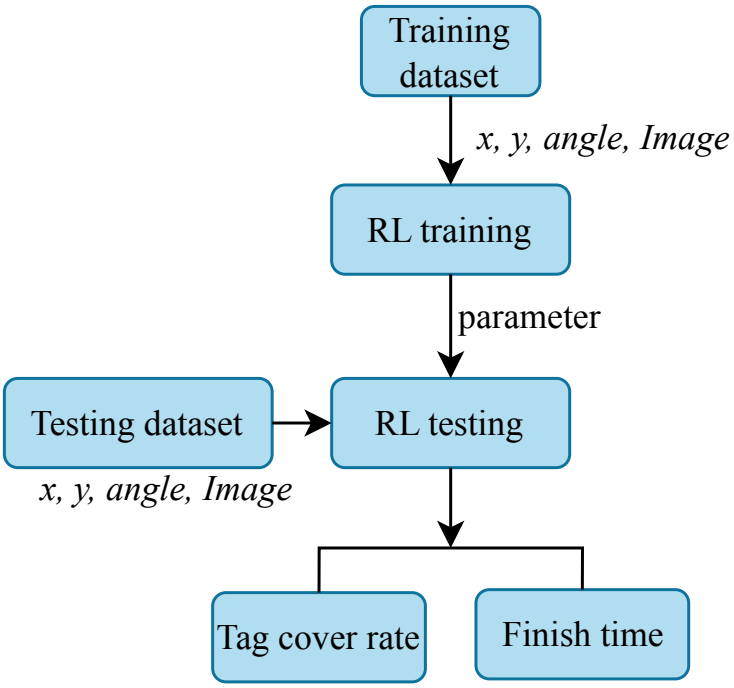


Figure 8. The flowchart of our reinforcement learning architecture for testing. After training is completed, the trained parameters are saved and imported for testing. The testing is conducted using a test dataset as input and follows the same framework. Evaluation is based on the detection rate of objects and the number of steps used.

3.2. *Action Selection*

As shown in Figure 9, an action requires a certain level of continuity, so the choice of the next move contains 'up', 'down', 'left', 'right', 'upper-right', 'upper-left', 'down-right' and 'down-left', with 1 meter displacement from the current position. For the actions on the camera orientation change, we set the range in the horizontal direction from -40° to 40°, with an interval of 10°. Moreover, to imitate how human conduct the inspection task, we include several additional mechanisms to action selection framework. First, the default action is set as move, until no new identifiable AprilTag appears in the images. The action is then adjusted to the camera orientation change. If there are no AprilTags found in the newly acquired images, the actions will change back to *move*.

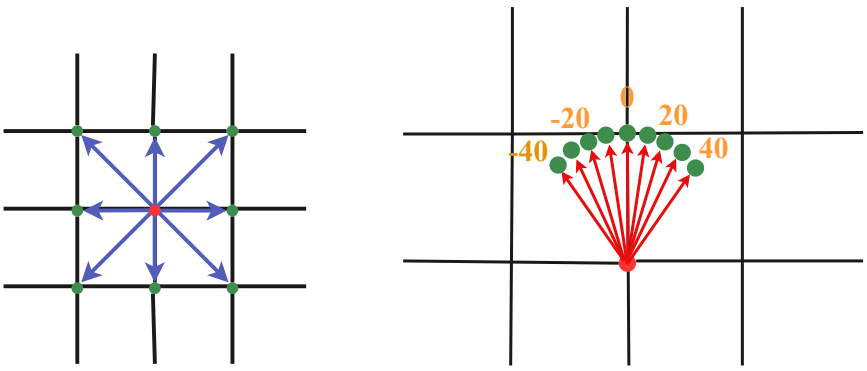


Figure 9. The action movement in translation (left, in front view) and rotation (right, in top view).

In terms of handling boundary conditions for action selection, a greater degree of exploration is usually preferred. Thus, to increase the randomness of action selection, more complex settings are implemented. As illustrated in Table 1, we assume the field range is $n \times n$. If the UAV's current position is on the boundary in the x-direction (say, $x = n$) and the action of increasing x is selected, we

 doi:10.20944/preprints202404.1526.v1

will first check if it is also located on the boundary in the y-direction. If not, the action will be randomly selected from 1, 2, 3, 5, 7. However, if the position is on the boundary in the y-direction (say $y = n$) as well, then only actions 2, 3 or 7 can be selected. In Table 2, we tabulate the relationships between action direction and action number. Nevertheless, the orientation changes are simpler compared to the moving action. All selections are legitimate as long as not the current angle.

Table 1. Handling the boundary condition for action selection.

State	Unavailable Actions	Available Actions
$x = n$	*0, 4, 6*	1, 2, 3, 5, 7
$x = 0$	2, 5, 7	0, 1, 3, 4, 6
$y = n$	1, 4, 5	0, 2, 3, 6, 7
$y = 0$	3, 6, 7	0, *1*, 2, 4, *5*

Table 2. The relationships between action direction and action number.

Action Number	0	1	2	3	4	5	6	7
Action Direction	$+x$	$+y$	$-x$	$-y$	$+x, +y$	$-x, +y$	$+x, -y$	$-x, -y$

3.3. Reward Function

For our reinforcement learning framework, we have designed various scoring methods. Depending on our objectives and the impact on evaluating the situations of action selection before and after, we have assigned different weight distributions for each calculation. The number of tag detection refers to the number of new Apriltags recognized by AprilTag detector in a single frame. If a tag has appeared and detected in the previous image, it will not be counted repeatedly in the current frame. That is, a tag could only be scored once for the reward. Since the tag detection rate is one of scoring criteria, it is expected to see as many recognizable AprilTags as possible in each frame. Thus, it carries the majority of the weight.

In the scoring criteria, completing the search is one of our goals. Therefore, extra points are awarded if the search of all tags in the scene are completed. In addition, even if the predetermined data collection amount is not reached after completing the search, the data collection will be forcibly exited. Afterward, all action selection and interaction with the environment stop, and learning will start. The environment will also be reset, initiating a new round of data collection. For initialization and reset, an empty array is set up to remember the labels seen before. It is cleared every time it is reset. Additionally, a random set of coordinates and angles is generated, representing the initial position. The content of the next state includes new coordinates, angles, images, a flag indicating completion status, selected actions, feedback scores after selecting actions, and a flag indicating the selected rotation angle.

For the entire framework, in addition to aiming to recognizing tags, one of our objectives is to perform the detection as quickly as possible. Thus, in the scoring method, both the number of movements and rotations are taken into account as costs. We expect to find new tags in each step. After each action, it is checked if new tags appear. If there are new tags detected after an action, the action selection counter is reset to zero; otherwise, it is incremented by 1. At this point, the score is deducted based on the counter's value by

$$Total\_Deduction = \sum_{k=1}^{n} \frac{k}{step} \tag{1}$$

where n is the value of the action selection counter, and $step$ is a pre-defined total number of steps that can be taken.

In order to identify areas of interest before the detector recognizes the targets, we propose a directional guidance mechanism. It combines YOLOv5 with the tag detection results, followed by

 doi:10.20944/preprints202404.1526.v1

locating the centers using K-means clustering to guide the next action selection. First, we obtain all potential tag locations using the pre-trained YOLOv5 model. It is compared with the detection derived from AprilTag detector. The overlap will be removed since it is confirmed to be a tag, and the remaining results would be regions of interest for exploration. Prior to K-means clustering, an outlier removal process is carried out to derive the representative cluster center. A score is calculated based on the cluster's center distance to the left-right of the frame. It is then used to direct the UAV's heading changing to the left or right for the next action. This will also prevent getting stuck in a certain place since it always provides a direction where tags are most concentrated.

3.4. Network

In the first half of the network, images are used as input and processed through the Resnet34 network for feature extraction. The output features are then used as input for the subsequent policy network. It consists of fully connected layers. The basic network configuration is as described previously, but there are slight modifications depending on the method used.

3.4.1. PPO (Proximal Policy Optimization)

In PPO, there are two policy networks, both of them with the same architecture but different functions [42,43]. The features obtained from Resnet34 are taken as input. It outputs a standard deviation and a mean value, forming the probability distribution of actions. The next action will be generated from the distribution and then undergo a final check in action selection. After a new action is obtained, it is feedback as input and update the probability distribution.

3.4.2. AC (Actor-Critic)

The features obtained by the previous Resnet34 network are used as input, and the output features are also used as input of the policy network. However, unlike the PPO method, the output here is an action probability curve. The next action is randomly generated from this distribution. Finally, the action selection is checked and the action curve is updated. The critic structure for both methods is the same. This part is relatively straightforward, taking the features obtained from the current state as input. The features are then fed into a critic network composed of fully connected layers. The output of the network provides an estimated value. We compare the estimate with the actual value obtained using the features from the next state into the same critic network to calculate the loss.

3.5. Dataset

Common methods for object detection data collection include sampling using planar and spherical points. As shown Figure 10, let p represent the sampling point, r is the radius, θ is the azimuth angle, and ϕ is the elevation angle. The point p can be obtained from the representative variables. In this work we consider the application for a warehouse environment. Items in the warehouse are typically placed on shelves, so a planar representation is chosen for data collection. We also consider occlusion caused by irregularly arranged goods, and the perspective in rotation is included. UE4 with the Airsim plugin to simulate drones is adopted to create the environment for data collection. The drone's physics engine is turned off during image capture to ensure stability and generality of the training results.

The dataset is generated using original Apriltag36h11 images, and are divided into virtual and real scenes. In the virtual scenes, the sampling is conducted using a spherical viewpoint setup in UE4. We include non-box objects to the scene, and Apriltags come in different sizes, with a total of 340 images. In real scenes, video recording is performed using an Intel Realsense D455, and segmented into images to create the dataset. The dataset includes 16 randomly placed boxes with Apriltags and some areas with square objects made of plastic pieces. There are 100 images annotated in the dataset. For the validation set, images of shelf scenes from the virtual environment are used, with 300 annotated images. The YOLO format is chosen with Apriltags annotated by enclosing in rectangular bounding boxes.

 doi:10.20944/preprints202404.1526.v1

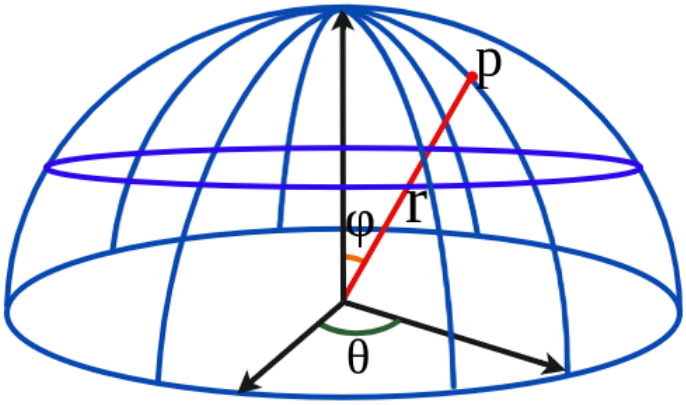


Figure 10. Sampling points are derived from the viewpoints using polar coordinates (r, θ, ϕ).

4. Experiments

The architecture of the proposed navigation system is shown in Figure 11. It contains an UAV equipped with an Intel RealSense D435 depth camera and an UP-Board embedded computer. In addition, there is a WLAN router facilitating information exchange between various devices, and a host computer which is responsible for running the navigation and obstacle avoidance algorithms. We adopt an ModalAI M500 for our experiments. It is a development quadcopter drone featuring the Qualcomm Snapdragon 821 processor. The flight control system of M500 is based on the PX4 Autopilot flight control system, supporting autonomous flight, remote control, and mission automation. It also offers advanced flight modes, such as Altitude Mode for altitude hold, Position Mode for position hold, and Offboard Mode for programmable control.

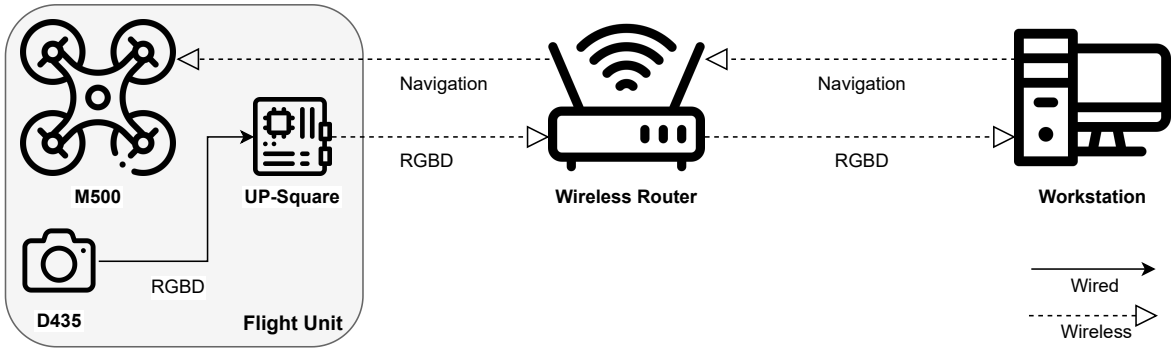


Figure 11. The navigation system architecture.

To conduct navigation experiments, we configured hardware by mounting an Intel RealSense D435 depth camera and an UP-Squared embedded computer on M500. This setup enables the drone to transmit real-time RGB-D images captured by the D435 camera back to the computation computer via wireless local area network (WLAN) using UP-Squared. Because of the inabilities of UP-Squared to successfully run certain navigation algorithms, we also opted for a notebook computer equipped with an Intel i5-8250U 1.6GHz CPU and 8GB RAM. Moreover, a desktop computer with the hardware configuration of an Intel i7-8700 3.2GHz CPU, NVIDIA RTX 2070 8G GPU, and 16GB RAM is used to test computational intensive tasks.

 doi:10.20944/preprints202404.1526.v1

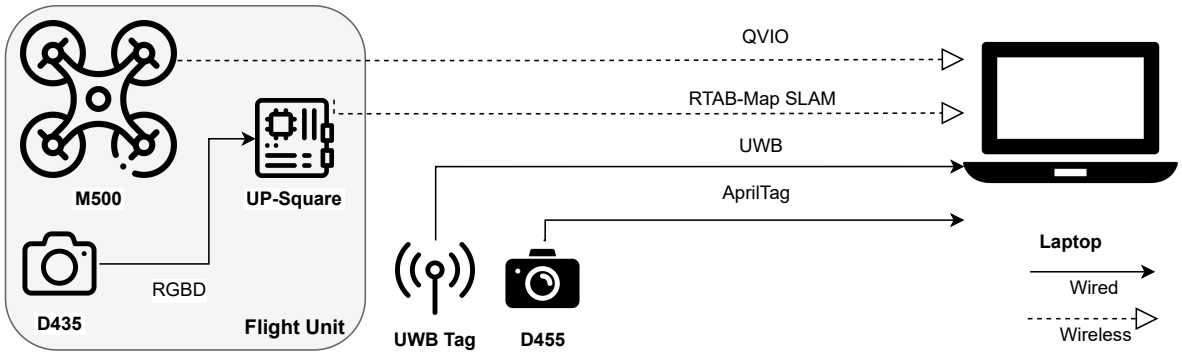


Figure 12. The positioning system architecture.

The architecture of our positioning system is illustrated in Figure 12. A mobile platform is used to carry the UAV along with the UWB Tag, and the D455 depth camera. This setup allows for multiple positioning methods, including AprilTag, QVIO, SLAM, and UWB. The hardware configuration of the mobile platform consists of various components. At the front, there is a VOXL tracking camera placed at a 45-degree downward angle, responsible for QVIO positioning. In addition, a D435 depth camera is mounted in the front for RTAB-Map mapping and positioning. In the rear, there is a D455 camera used for AprilTag detection.

4.1. Indoor Mapping and Positioning

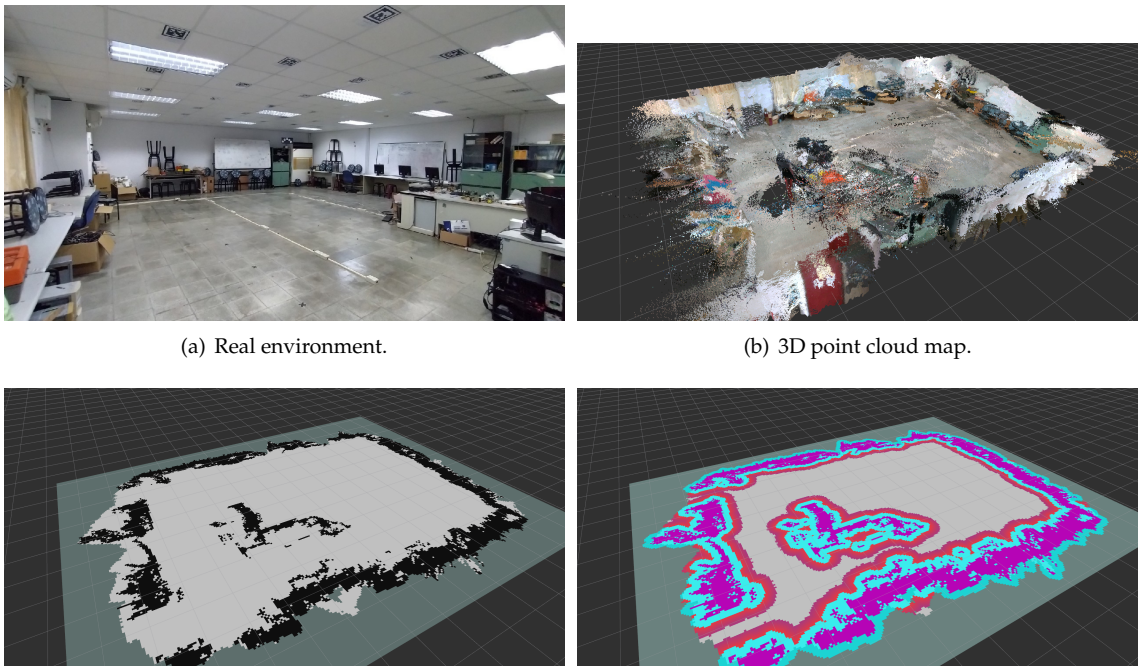
After setting up four anchors, we use the control window to observe the relative positions and coordinates of these anchors. The anchors are placed around the experimental environment, forming a rectangular space within which the tag moves. This setup enables a global indoor positioning technique similar to GPS, allowing for accurate localization within the designated area. Using the mobile platform with the UWB tag mounted on it, indoor positioning experiments are conducted while keeping the height (Z-axis) as a constant. With these experiments, it is observed that although the UWB positioning exhibits larger variations in the Z-axis, the overall X and Y-axis horizontal positioning trajectories do not exhibit drift or divergence over time. Since UWB positioning does not rely on visual cues, it remains accurate even in environments lacking distinctive features (e.g., white-walled rooms) or areas with low lighting conditions. Nevertheless, it is important to be cautious of obstacles as they could impact the accuracy of UWB positioning, depending on their distance and size relative to the anchors and the UWB tag.

Using the QVIO tracking camera positioned at a 45-degree downward angle on the front of the UAV, the area-based indoor positioning technique is employed. Compared to global positioning techniques, the advantage of the area-based positioning lies on achieving more precise localization in a defined space, often resulting in smoother trajectories. However, if the global markers are not available, it is not easy to correct accumulated positioning errors, which might worsen over time due to small inaccuracies. Visual localization heavily relies on extracting feature points from a rich texture scene, which is critical for overall positioning robustness. To address this issue, we place numerous objects in the experimental environment around the surroundings to cover the plain white walls. This can enhance the availability of visual features for QVIO and improve the stability of the visual-based localization.

Utilizing the upward-facing D455 depth camera mounted on top of the mobile platform, we carried out AprilTag detection on the captured RGB images. After a series of tests, we found that continuously detecting AprilTags in the scene can prevent the issues such as incorrect path connections and non-smooth trajectories. The red arrows represent the odometry, and the green path illustrates the motion trajectory. Moreover, when multiple AprilTags (two or more) are detected simultaneously within the same image frame, it results in more accurate and stable pose estimation compared to detecting only a single AprilTag. This multi-detection capability enhances the robustness of the AprilTag-based localization process.

 doi:10.20944/preprints202404.1526.v1

In virtual environments, we utilize Gazebo [44] to simulate the UAV, RGB-D camera, and indoor space for mapping. This simulation is essential for validating the algorithms before the implementation in real-world scenarios. The mapping process includes the generation of a 3D point cloud for visual feature loop closure detection, a 2D occupancy grid for global path planning and obstacle avoidance, and a 2D map showing the safe flying zone for the UAV, as illustrated in Figure 13. We have conducted RTAB-Map localization in real environments with four scenarios. These scenarios include loop closure detection, loop closure not detected, loop closure rejected, and odometry lost.



(a) Real environment. (b) 3D point cloud map.

(c) 2D occupancy grid map. (d) 2D inflation map.

Figure 13. RTAB-Map - Real environment mapping.

- *Scenario 1:* When the drone images detect a loop closure in the previously constructed map, it corrects the accumulated drift error of the visual odometry, achieving global indoor localization.
- *Scenario 2:* When the loop closure is undetected, the drone relies solely on visual odometry and extracts visual features for local indoor localization.
- *Scenario 3:* When loop closure is rejected, it indicates that loop closure is detected but does not exceed the pre-set threshold for acceptance. This typically occurs when the drone's position is correct, but there is some slight deviation in the orientation.
- *Scenario 4:* When odometry is lost, it means the drone's visual odometry is unable to maintain continuity with the previous frame due to significant and rapid image motion in a short period. This often happens if the drone rotates in place, resulting in an instantaneous angular velocity.

Figure 14 shows the indoor localization trajectories obtained from the RTAB-Map SLAM mapping mode and localization mode. The blue, green and red paths represent the trajectory obtained during the initial mapping mode, localization mode and the mapping mode is reactivated during the second localization mode. From the SLAM trajectory comparison graph, we can observe that the latter two trajectories are very similar, and the advantages of pre-constructed maps in the localization mode are not evident. Therefore, the benefits of loop closure detection for correction become more prominent in spacious environments where visual odometry might diverge.

 doi:10.20944/preprints202404.1526.v1

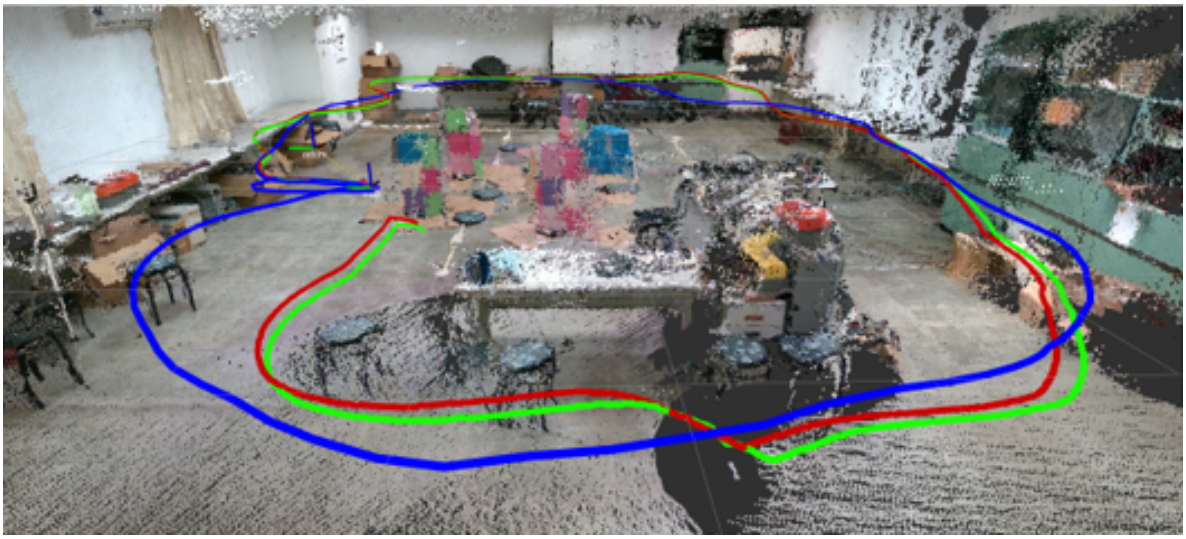


Figure 14. The indoor localization trajectories obtained from the RTAB-Map SLAM mapping and localization modes.

Figure 15 illustrates the characteristics of different localization techniques. The yellow, cyan, red, and blue curves represent ground truth, UWB, QVIO, and AprilTag localization trajectories, respectively. In addition, the magenta path corresponds to the SLAM (Simultaneous Localization and Mapping) mapping mode trajectory, and the green path represents the localization mode trajectory. Despite some noticeable fluctuations in altitude, the UWB trajectory does not diverge significantly due to severe shaking. The performance of QVIO remains in a state of convergence, mainly because of the lack of ground features. The AprilTag trajectory shows excellent stability and closely follows the Ground Truth trajectory. The SLAM localization trajectory is very similar to the mapping trajectory.

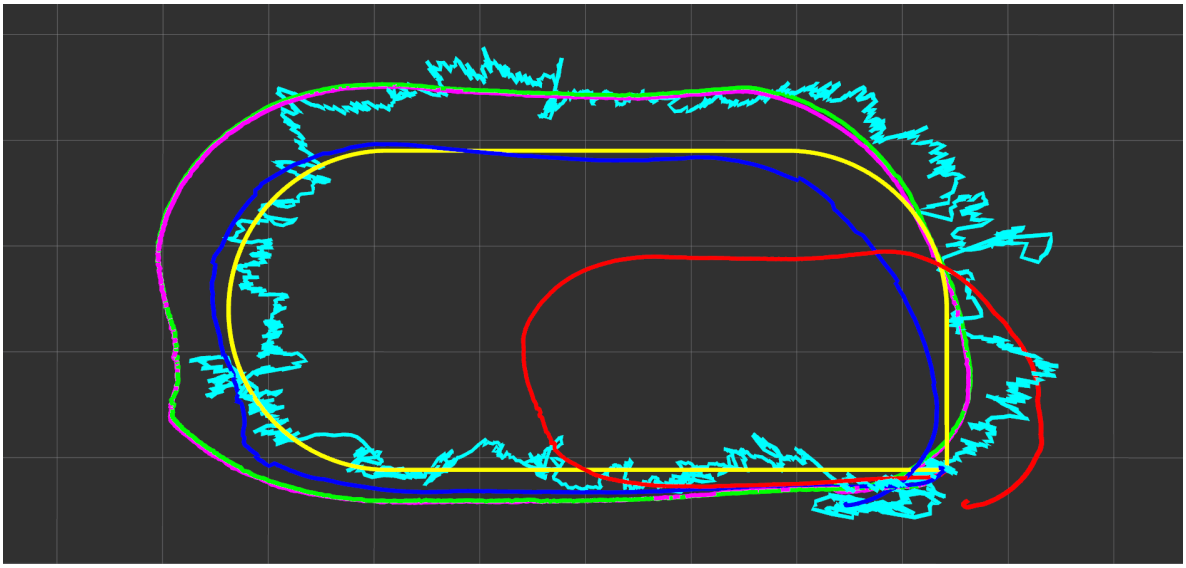


Figure 15. The characteristics of different localization techniques.

4.2. Path Planning with Obstacle Avoidance

The feasibility of navigation and obstacle avoidance algorithms are tested in both virtual and real environments. These consist of the construction of 2D occupancy grid maps with static known and unknown obstacles that are not present during mapping. In addition, we test the algorithm's robustness with dynamic unknown obstacles to validate the performance. In virtual environments, Figure 16 illustrates the integration of Gazebo and RViz. The Gazebo shows the simulation environment, while

 doi:10.20944/preprints202404.1526.v1

the RViz display window shows various elements, including the global path, local path, dynamic obstacles, static obstacles, unknown obstacles, and known obstacles. The red arrow in represents the destination that the drone is heading towards in RViz, the green path is the global path generated by the *navfn* global planner, and the yellow path is the local path generated by the DWA (Dynamic Window Approach) local planner. The light gray, dark gray, black and red grids represent the flyable area, unknown area, known obstacles and obstacles detected by the sensors, respectively. The light blue grid represents the map based on the drone's radius, and the dark blue grid represents the map with an additional safety margin applied after testing.

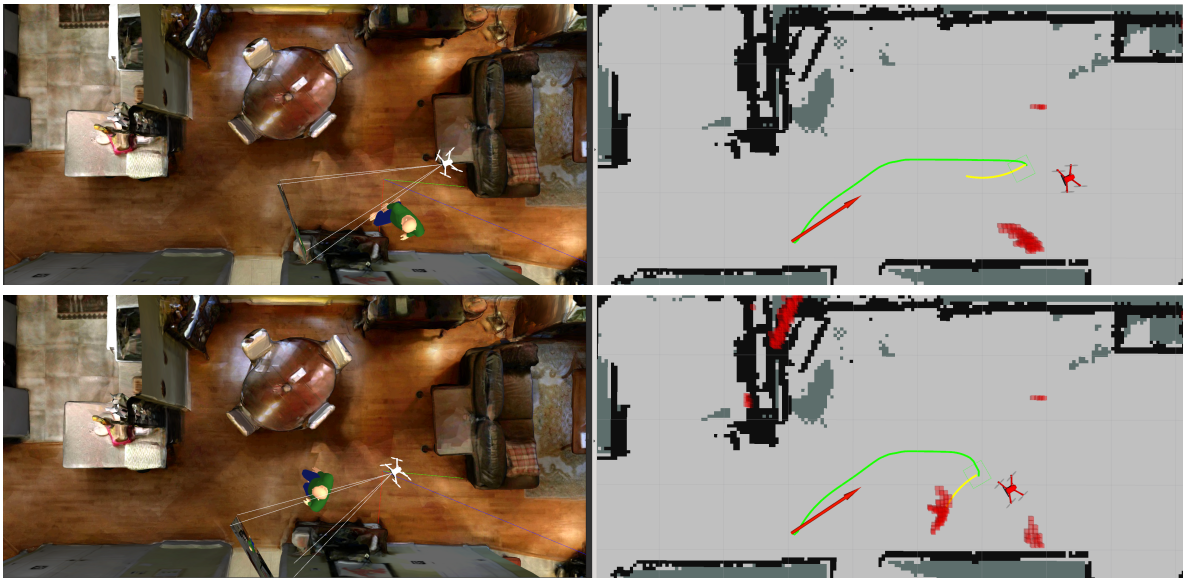


Figure 16. Path planning for dynamic obstacle avoidance - virtual environment.

In real environments, Figure 17 illustrates the integration of RTAB-Map (on the left) and RViz (on the right). The setup allows for testing the effectiveness of the global path planner using static known obstacles. The performance of the local path planner is evaluated using a puzzle made of foam blocks as static unknown obstacles. In addition, the local path planner's real-time responsiveness is tested by having a floor robot carrying the foam blocks as dynamic unknown obstacles.

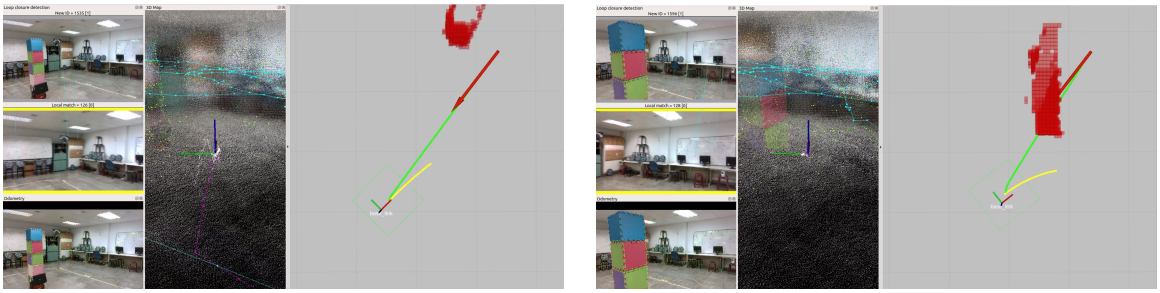


Figure 17. Path planning for dynamic obstacle avoidance - Real-world environment.

4.3. Inventory Inspection

The proposed inventory inspection techniques have been tested on simulation and real-scene environments. In the virtual environment, UE4, the boxes are created with four different sizes, $1 \times 1.33 \times 1$ m^3, $2 \times 1.33 \times 1$ m^3, $2 \times 1.33 \times 1.5$ m^3 and $2 \times 1.33 \times 1$ m^3. The size of Apriltag is 0.21×0.271 m^2. As illustrated in Figure 18, the boxes are placed on a shelf, and the drone equipped with a camera flies at a fixed height (4.5 m). The camera's viewing angle is downward at 35°. We divide the aisle into three sampling point tracks parallel to the shelf. Each track consists of 35 positions, so there are totally 105 sampling viewpoints. The boxes on the shelf are placed randomly, including

 doi:10.20944/preprints202404.1526.v1

different orientation and occlusion. In this paper, we adopt PPO and AC methods. Both of them use Resnet34 for image feature extraction and the same dataset for training. Their architectures are written in PyTorch. PPO runs on Ubuntu18.04 with a Nvidia GeForce RTX 3070Ti, and AC runs on Ubuntu20.04 with a Nvidia GeForce RTX 4090.

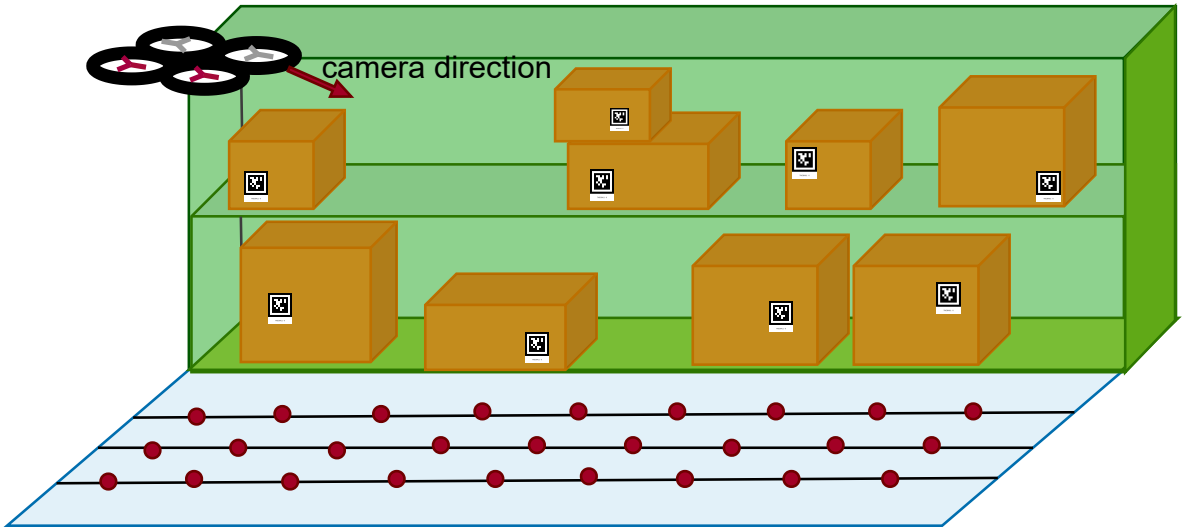


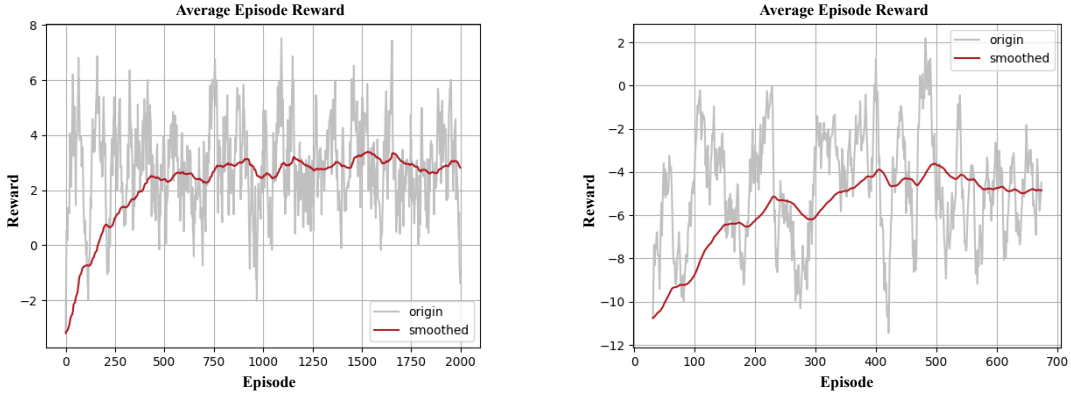
Figure 18. The boxes are placed on a shelf, and the drone equipped with a camera flies at a fixed height.

After multiple rounds of parameter adjustment experiments, we set the maximum number of movements to 50. Both actor and critic have learning rates set as 0.02, and the update frequency for each episode is 20. We set epsilon to 0.2, and the optimizer adopted is Adam. Figure 19(a) shows the training results of PPO, where episode represents how many paths have been taken, and indicating the number of times the environment is reset. In AC training, we set the maximum number of movements to be 100. Both actor and critic have a learning rate set to 0.0002, with gamma set to 0.9. The update frequency for each episode is 24, and there's a learning rate decay of 0.7. The optimizer used is Adam. Figure 19(b) illustrates the training results of AC, where episode represents how many paths have been taken, indicating the number of times the environment is reset.

Table 3. Summary of reinforcement learning training parameters.

	Max Movement	Episode Update	Learning Rate	Episode
PPO	50	20	0.02	0.2
AC	100	24	0.0002	0.2

 doi:10.20944/preprints202404.1526.v1



(a) PPO training result. (b) AC training result.

Figure 19. The reinforcement learning training results.

The testing experiments are conducted using trained parameters to estimate the number of steps taken and the detection rate of boxes. Figure 20 shows the testing results using PPO in a virtual environment with step set to 50. These include steps 1 to 4 and 47 to 50 in the left and right columns, respectively. The blue circles represent the tags newly detected in the frame. Since the previously scanned tags are not marked again, their scores are not included in the new calculation. The text in the upper-left corner shows the cumulative number of tags, with a total of 25 found in this test. 10 simulation tests with 26 boxes are performed in the experiment. The numbers of detected boxes are either 24 or 25, and the average detection rate is reported as 94.62%. Figure 21 shows the testing results using AC in a virtual scene with step set to 80. These include steps 1 to 4 and 77 to 80 in left and right columns, respectively. The same as PPO testing, there are 26 boxes to identify in the scene. With 10 simulation tests, the numbers of detected boxes range from 20 to 24, with an average detection rate of 90.38%.

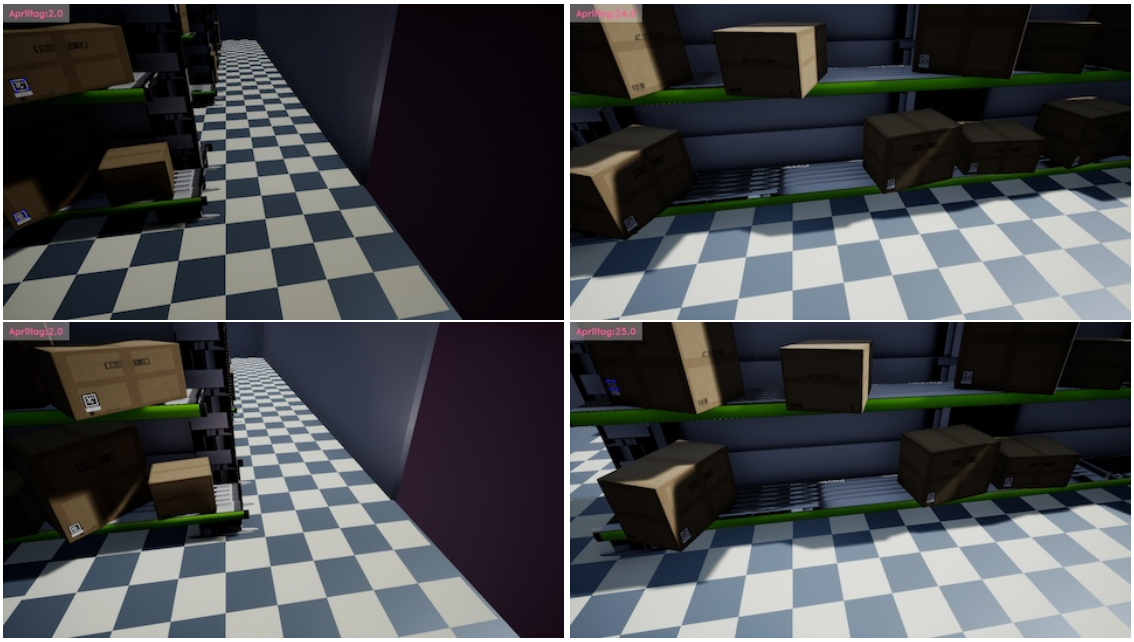


Figure 20. The testing results using PPO in a virtual environment with step set to 50.

 doi:10.20944/preprints202404.1526.v1

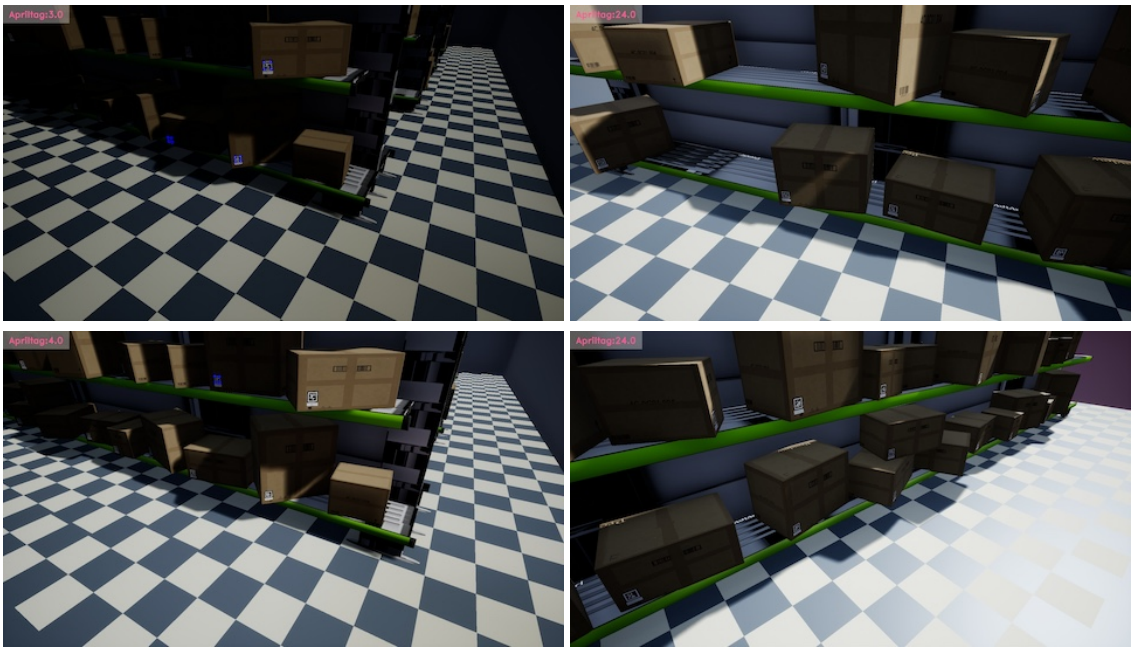


Figure 21. The testing results using AC in a virtual scene with step set to 80.

For real scene experiments, 16 cardboard boxes are placed on a two-tier elevated table. Each box had a unique Apriltag with ID numbers ranging from 0 to 15. Two different box types are used: one was $25 \times 26 \times 35$ cm^3 (Type A), and the other is $18 \times 38 \times 40$ cm^3 (Type B). There are a total of 10 Type A boxes and 6 Type B boxes. The dimension of the Apriltag is 6.7×6.7 cm^2, and the length of the aisle in the scene is approximately 3.6 meters. We utilize a Realsense D455 camera to capture images at a fixed height, and adopt a digital compass to record the orientation. Similar to the virtual environment, the images are sampled at every 30 cm. The aisle is divided into three parallel tracks, so there are totally 3×12 sampling locations. Together with 9 viewpoints each position, result in 324 images for an experiment.

Figure 22 presents the real-world testing results for PPO training (steps 1 – 4 and 10 – 13 in the left and right columns, respectively). In this scene, there are a total of 16 boxes to search for, and the maximum number of steps for each movement is set to 50. Table 4 (left) shows statistics from 10 real-world scenario tests, recording the final number of boxes found, detection rates, and the total number of steps (time) used. It can be seen from the table that in nearly every instance, all the boxes in the scene are found in approximately 20 steps or less. The average number of boxes detected is 15.9, with a detection rate of 99.38%. There are a total of 16 boxes to search for AC training, and the maximum number of steps for each movement is also set to 50. Table 4 (right) provides statistics from 10 real-world tests, recording the number of boxes found, detection rates, and the total number of steps (time) used. It can be observed from the table that in nearly every instance, all boxes in the scene are found. The average number of items detected is 15.7, with a detection rate of 98.13%. However, in terms of the number of steps required, AC needs twice as many steps to complete the task compared to PPO. The average number of steps required was 36.4 steps.

 doi:10.20944/preprints202404.1526.v1

Figure 22. The real-world testing results for PPO training.

Table 4. The statistics from 10 real-world scenario tests, recording the final number of boxes found, detection rates, and the total number of steps (time) used. In the left table of PPO real scene experimental results, all the targets can be identified in 20 steps almost every time. The average number of cargo detected is 15.9, the cargo detection rate is 99.38%, and the average number of steps required is 16.2 steps. In the right table of AC real scene experiment, the results show that almost all the targets can be detected every time. The average number of cargo detected is 15.7, the cargo detection rate is 98.13%, the average number of steps required is 36.4 steps, In terms of the number of steps, the PPO method requires about 2 times to complete the task.

Exp.	No.	Step	Coverage
1	16	19	100%
2	16	11	100%
3	15	50	92.31%
4	16	17	100%
5	16	13	100%
6	16	15	100%
7	16	10	100%
8	16	13	100%
9	16	14	100%
10	16	16	100%
Average	15.9	16.2	99.38%

Exp.	No.	Step	Coverage
1	14	50	87.5%
2	16	40	100%
3	16	19	100%
4	16	37	100%
5	16	44	100%
6	15	50	93.75%
7	16	31	100%
8	16	28	100%
9	16	19	100%
10	16	38	100%
Average	15.7	36.4	98.13%

5. Conclusions and Future Work

This research first integrates multiple indoor localization techniques, including visual-inertial odometry, SLAM, UWB and AprilTag. It compares their stability and accuracy. We also utilize ROS navigation stack, combining PX4 Autopilot, MAVROS, QGroundControl for drone navigation, validating both global path planning and local path planning algorithms. Moreover, RTAB-Map is used for map construction to achieve indoor localization with visual odometry. We replace expensive radar with cost-effective depth cameras, and conduct verification in virtual and real environments, considering various application scenarios, such as known obstacles, unknown obstacles, and dynamic unknown obstacles. Extensive testing and parameter tuning are conducted to validate the algorithm's robustness, stability, and real-time performance. For the warehouse management system using an UAV,

 doi:10.20944/preprints202404.1526.v1

this work implements a path planning technique based on Apriltags using reinforcement learning. The training results in virtual environments have been applied to simulations as well as real-world testing. In the simulated environment, the PPO method achieves an average detection rate of 94.62%, while AC achieves an average detection rate of 90.38%. In the real-world environment, PPO almost consistently finds all targets in about 20 steps, with an average detection rate of 99.38%. AC also finds all boxes in most cases, with an average detection rate of 98.13%. The results demonstrate that the effectiveness of our approach is validated with real-world scenes.

The method used in this paper is based on the ROS Navigation Stack, which currently enables navigation in a two-dimensional space, where the altitude remains fixed. To fully leverage the unmanned aerial vehicle's capabilities for free movement in three-dimensional space in the future work, it is essential to extend the navigation to the Z-axis. This extension involves incorporating path planning for altitude adjustment and obstacle avoidance, enabling the UAV to achieve free movement in three-dimensional space. In addition, this study is conducted in a controlled environment, and practical real-world applications are undoubtedly more complex and unpredictable. Therefore, enhancing robustness is a critical issue. This includes optimizing both global and local path planning trajectories to make the unmanned aerial vehicle more efficient and energy-saving. Strengthening collision avoidance mechanisms and protective strategies in both the UAV's software and hardware is vital to ensure the safety of personnel in the environment. Moreover, developing reliable human recognition systems can further enhance safety measures in the UAV's operation.

Acknowledgments: The support of this work in part by the Ministry of Science and Technology of Taiwan under Grant MOST 106-2221-E-194-004 is gratefully acknowledged.

Conflicts of Interest: Declare conflicts of interest or state "The authors declare no conflict of interest." Authors must identify and declare any personal circumstances or interest that may be perceived as inappropriately influencing the representation or interpretation of reported research results. Any role of the funders in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; or in the decision to publish the results must be declared in this section. If there is no role, please state "The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results".

References

1. de Jesus, J.C.; Kich, V.A.; Kolling, A.H.; Grando, R.B.; Guerra, R.S.; Drews, P.L.J. Depth-CUPRL: Depth-Imaged Contrastive Unsupervised Prioritized Representations in Reinforcement Learning for Mapless Navigation of Unmanned Aerial Vehicles. 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 10579–10586. doi:10.1109/IROS47612.2022.9982161.
2. Moura, A.; Antunes, J.; Dias, A.; Martins, A.; Almeida, J. Graph-SLAM Approach for Indoor UAV Localization in Warehouse Logistics Applications. 2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2021, pp. 4–11. doi:10.1109/ICARSC52212.2021.9429791.
3. Awate, Y.P. Policy-Gradient Based Actor-Critic Algorithms. 2009 WRI Global Congress on Intelligent Systems, 2009, Vol. 3, pp. 505–509. doi:10.1109/GCIS.2009.372.
4. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* **2017**.
5. Xia, J.; Li, S.; Wang, Y.; Jiang, B. Research on uwb/ble-based fusion indoor positioning algorithm and system application. 2021 International Symposium on Computer Technology and Information Science (ISCTIS). IEEE, 2021, pp. 50–54.
6. Xia, J.; Wu, Y.; Du, X. Indoor Positioning Technology Based on the Fusion of UWB and BLE. Security, Privacy, and Anonymity in Computation, Communication, and Storage: SpaCCS 2020 International Workshops, Nanjing, China, December 18-20, 2020, Proceedings 13. Springer, 2021, pp. 209–221.
7. Shang, S.; Wang, L. Overview of WiFi fingerprinting-based indoor positioning. *IET Communications* **2022**, *16*, 725–733.
8. Deng, W.; Li, J.; Tang, Y.; Zhang, X. Low-Complexity Joint Angle of Arrival and Time of Arrival Estimation of Multipath Signal in UWB System. *Sensors* **2023**, *23*, 6363.

9. Krogius, M.; Haggenmiller, A.; Olson, E. Flexible layouts for fiducial tags. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019, pp. 1898–1903.
10. Garrido-Jurado, S.; Muñoz-Salinas, R.; Madrid-Cuevas, F.J.; Marín-Jiménez, M.J. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition* **2014**, *47*, 2280–2292.
11. Kato, H.; Billinghurst, M. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99). IEEE, 1999, pp. 85–94.
12. Leutenegger, S.; Lynen, S.; Bosse, M.; Siegwart, R.; Furgale, P. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research* **2015**, *34*, 314–334.
13. Mourikis, A.I.; Roumeliotis, S.I. A multi-state constraint Kalman filter for vision-aided inertial navigation. Proceedings 2007 IEEE international conference on robotics and automation. IEEE, 2007, pp. 3565–3572.
14. Bloesch, M.; Omari, S.; Hutter, M.; Siegwart, R. Robust visual inertial odometry using a direct EKF-based approach. 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2015, pp. 298–304.
15. Labbé, M.; Michaud, F. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics* **2019**, *36*, 416–446.
16. Qin, T.; Li, P.; Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics* **2018**, *34*, 1004–1020.
17. Mur-Artal, R.; Tardós, J.D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics* **2017**, *33*, 1255–1262.
18. Lin, H.Y.; Tu, K.C.; Li, C.Y. Vaid: An aerial image dataset for vehicle detection and classification. *IEEE Access* **2020**, *8*, 212209–212219.
19. Lin, H.Y.; Peng, X.Z. Autonomous quadrotor navigation with vision based obstacle avoidance and path planning. *IEEE Access* **2021**, *9*, 102450–102459.
20. Ghosh, S.K. *Visibility algorithms in the plane*; Cambridge university press, 2007.
21. Chaari, I.; Koubaa, A.; Bennaceur, H.; Ammar, A.; Alajlan, M.; Youssef, H. Design and performance analysis of global path planning techniques for autonomous mobile robots in grid environments. *International Journal of Advanced Robotic Systems* **2017**, *14*, 1729881416663663.
22. Tsardoulias, E.G.; Iliakopoulou, A.; Kargakos, A.; Petrou, L. A review of global path planning methods for occupancy grid maps regardless of obstacle density. *Journal of Intelligent & Robotic Systems* **2016**, *84*, 829–858.
23. Fox, D.; Burgard, W.; Thrun, S. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine* **1997**, *4*, 23–33.
24. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research* **1986**, *5*, 90–98.
25. Kobayashi, M.; Motoi, N. Local path planning: Dynamic window approach with virtual manipulators considering dynamic obstacles. *IEEE Access* **2022**, *10*, 17018–17029.
26. Kalinov, I.; Petrovsky, A.; Ilin, V.; Pristanskiy, E.; Kurenkov, M.; Ramzhaev, V.; Idrisov, I.; Tsetserukou, D. WareVision: CNN Barcode Detection-Based UAV Trajectory Optimization for Autonomous Warehouse Stocktaking. *IEEE Robotics and Automation Letters* **2020**, *5*, 6647–6653. doi:10.1109/LRA.2020.3010733.
27. Yang, S.Y.; Jan, H.C.; Chen, C.Y.; Wang, M.S. CNN-Based QR Code Reading of Package for Unmanned Aerial Vehicle. *Sensors* **2023**, *23*, 4707.
28. Babu, S.; Markose, S. IoT enabled Robots with QR Code based localization. 2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR). IEEE, 2018, pp. 1–5.
29. Cho, H.; Kim, D.; Park, J.; Roh, K.; Hwang, W. 2D barcode detection using images for drone-assisted inventory management. 2018 15th International Conference on Ubiquitous Robots (UR). IEEE, 2018, pp. 461–465.
30. Cristiani, D.; Bottonelli, F.; Trotta, A.; Di Felice, M. Inventory Management through Mini-Drones: Architecture and Proof-of-Concept Implementation. 2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), 2020, pp. 317–322. doi:10.1109/WoWMoM49955.2020.00060.
31. Yoon, B.; Kim, H.; Youn, G.; Rhee, J. 3D position estimation of drone and object based on QR code segmentation model for inventory management automation. 2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). IEEE, 2021, pp. 223–229.

 doi:10.20944/preprints202404.1526.v1

32. Rhiat, A.; Chalal, L.; Saadane, A. A Smart Warehouse Using Robots and Drone to Optimize Inventory Management. Proceedings of the Future Technologies Conference (FTC) 2021, Volume 1. Springer, 2022, pp. 475–483.
33. Manjrekar, A.; Jha, D.S.; Jagtap, P.; Yadav, V.; others. Warehouse inventory management with cycle counting using drones. Proceedings of the 4th International Conference on Advances in Science & Technology (ICAST2021), 2021.
34. Vamsi, A.M.; Deepalakshmi, P.; Nagaraj, P.; Awasthi, A.; Raj, A. IOT based autonomous inventory management for warehouses. EAI International Conference on Big Data Innovation for Sustainable Cognitive Computing: BDCC 2018. Springer, 2020, pp. 371–376.
35. Kalaitzakis, M.; Cain, B.; Carroll, S.; Ambrosi, A.; Whitehead, C.; Vitzilaios, N. Fiducial markers for pose estimation: Overview, applications and experimental comparison of the artag, apriltag, aruco and stag markers. *Journal of Intelligent & Robotic Systems* **2021**, *101*, 1–26.
36. Wang, J.; Olson, E. AprilTag 2: Efficient and robust fiducial detection. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4193–4198. doi:10.1109/IROS.2016.7759617.
37. Brock, O.; Khatib, O. High-speed navigation using the global dynamic window approach. Proceedings 1999 ieee international conference on robotics and automation (Cat. No. 99CH36288C). IEEE, 1999, Vol. 1, pp. 341–346.
38. Dijkstra, E.W. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*; 2022; pp. 287–290.
39. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. International conference on machine learning. PMLR, 2016, pp. 1928–1937.
40. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* **2013**.
41. Koonce, B.; Koonce, B. ResNet 34. *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization* **2021**, pp. 51–61.
42. Shani, L.; Efroni, Y.; Mannor, S. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. Proceedings of the AAAI Conference on Artificial Intelligence, 2020, Vol. 34, pp. 5668–5675.
43. Zhong, H.; Zhang, T. A theoretical analysis of optimistic proximal policy optimization in linear markov decision processes. *Advances in Neural Information Processing Systems* **2024**, *36*.
44. Koenig, N.; Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. 2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566). IEEE, 2004, Vol. 3, pp. 2149–2154.