A Robotic Scout UAV for Mapping Dynamic Environments at Bechtel Corporation

Ahmed Abdelmoamen, Abel Olumide, Adeoluwa Akinwa, and Mohamed Chouikha

Department of Computer Science / Department of Computer Science / Department of Computer Science / SECURE Center of Cybersecurity
Prairie View A&M University
Prairie View, Texas

Abstract

This paper presents the design and implementation of an unmanned aerial vehicle (UAV), which can navigate autonomously in dynamic environments. The goal of the project is to minimize the risks to workers' safety by deploying UAVs to inaccessible places that are frequently found in the oil & gas industry, such as confined pipelines. The autonomous UAV can fly through a series of pipes to generating a 3D map of the flight path. We used light detection and ranging (LIDAR) technology to map the surrounding environment as the UAV flies through the environment. The feedback from the LIDAR sensors is used for real-time autonomous navigation and obstacle avoidance. The route is also logged for subsequent navigation. As a UAV navigates the environment, it records a video of all it sees, which can then be watched by the maintenance engineers. Our approach involves running a simulation using the robotics operating system (ROS) to assert and fine-tune our navigation algorithms before applying them directly to the physical hardware. At this stage, we have successfully implemented the autonomous navigation using LIDAR scanners in the ROS simulation environment. We also implemented an algorithm to manage the battery life of the UAV through which it can use to return home when the battery level drops down to a certain percentage. We expect that this research will help autonomous UAVs to safely navigate new spaces by themselves in different domains such as in industrial maintenance and rescue operations.

1. Introduction

It is becoming increasingly important to use unmanned aerial vehicles (UAVs) in industrial maintenance where human-crewed operations are considered too risky or difficult [26, 27]. For instance, there are several risks associated with work in confined environments such as difficult terrains, an insufficient amount of oxygen for the worker to breathe, the presence of hazardous chemical or biological substances, etc. The versatility of UAVs opens up an opportunity to get access to these confined places, from any angle, regardless of the shape and geometry of the environment [9]. However, the risk of collisions limits the widespread of flying UAVs in unknown environments where no GPS-based maps exist [25].

When the GPS is denied, current navigational devices can become highly inaccurate [31]. In such GPS-denied environments, there is a need for alternative navigation methods for UAVs to generate a map of its surrounding in near-real time, which can be used by the UAV to avoid obstacles while in motion. Various technologies are employed to achieve this autonomous

navigation including light detection and ranging (LiDAR) method [29, 30] in which a laser range scanner onboard the UAV is used to send out laser pulses to nearby objects. The reflected laser rays are then consumed by the rangefinder to determine the distance of each point the rays hit to the UAV. The resulting scan points are combined to form a layout (map) of the surrounding of the UAV. This map, in turn, is updated and the relative location of the UAV in the map is simultaneously tracked using computational algorithms. This method is known as simultaneous localization and mapping (SLAM) [4], which is usually used to localize a robot in an unknown environment while simultaneously mapping it.

In this paper, we used the SLAM method to create an autonomous UAV that would fly through pipelines to identify corrosion or debris. This project is motivated by the challenges faced by Bechtel Cooperation (see https://www.bechtel.com/) in the maintenance of its oil pipeline. It is both expensive and impractical for Bechtel engineers to travel through the length and bends of the pipes to discover obstacles, corrosion, and other problems in the lines. Therefore, there is a need for an intelligent and autonomous UAV that can fly through the confined elbows, pinch points, and long length of the lines to detect such problems. The scout UAV should have the following features: (i) fly through a 48-inch diameter empty dark pipe in vertical and horizontal navigation, and through 90-degree elbows; (ii) fly autonomously until reaching the end of pipeline by finding out the difference between forward and backward navigation and continue moving forward; (iii) return home (i.e., starting point) if the UAV does not make it through obstructions it encounters; and (iv) generate 3D map of the flight path using a laser scanner.

We developed a real-time navigation algorithm that can accurately guide the UAV to navigate through unknown environments. The developed solution is available online at https://github.com/abelmeadows/scoutrobot. We tested the developed algorithm using the robotics operating system (ROS) [7]. Specifically, both the navigation algorithm and environmental maps have been successfully simulated in Gazebo physics engine [3] in the ROS environment. To simulate the pipelines environment, we created similar confined environments that have various obstacles to replicate any unexpected objects in the pipeline. The significance of the simulation result encouraged us to apply the navigation algorithm in the physical hardware using the actual sensor driver libraries for Parrot AR UAV. We also used Bebop2 Ardrone [2], Arduino microprocessor [1], WiFi module, and a web application server built on Python Flask library [6].

The rest of the paper is organized as follows: Section 2 presents related work. Sections 3 and 4 present the design and prototype implementation of the simulation of the proposed UAV system using the Gazebo simulator and the physical hardware using a Parrot AR UAV, respectively. Section 5 concludes the paper.

2. Related Work

UAVs have been employed in a wide range of applications, including industrial inspection, surveying, aerial images and photography, disaster area inspection, package delivery, and rescue operations. Autonomous UAV systems, in particular, are receiving more attention globally as numerous research efforts are undertaking in different domains [32]. This section focuses on the existing work that targets autonomous UAV navigation in indoor environments.

A path planning algorithm is proposed in [31], which enables data communication between the sensors onboard of a UAV for generating a set of way-points that are considered a planned path for the flight to reach a goal point in an indoor environment. These way-points are then translated into navigation commands using a proportional derivative controller. Also, the Hector SLAM method [4] is used to generate a map for the environment using laser scan data. Once the map is created, it is divided into occupancy grids of known cell sizes. At the take-off, the UAV was programmed to navigate to the goal point and maintain its orientation by choosing between left, forward, or right grid cells at any point in time throughout the flight. The flight controller was developed, based on the Newton-Euler equations, for varying the rotational speed of the UAV four rotors to trigger the yaw, roll, and pitch movements.

Another method for autonomous navigation based on visual-based positioning is studied in [35]. The authors used a motion tracking and depth perception camera onboard of the UAV for achieving autonomous navigation in GPS-denied environments. Google's Project Tango tablet was used as a visual sensor tool which has a collection of sensors such as inertial measurement units (IMU) [30] and depth cameras. The tablet was mounted on pelican quadrotors as a carrying body to supply pose estimation. A recognition algorithm was developed to help in avoiding obstacles. The recognized area to the depth camera is mapped as a shaded rectangular, and the unknown region is mapped as a blank oblong. Given that the ability to estimate the position of the UAV relative to the surrounding is a critical factor to autonomous navigation in GPS-denied environments, the depth camera needs to generate waypoints from the captured images of the environment. However, visual-based navigation assumes that there will be an adequate supply of light, which is not very likely available in confined indoor situations such as pipelines.

Autonomous navigation based on geo-registered 3D point cloud is studied in [34]. The authors presented an autonomous navigation and path planning system for UAVs, which combines geo-registered 3D point-clouds and 2D digital maps for vision-based navigation. In particular, the authors merged the outputs of multiple proprioceptive sensors, such as IMU, odometry and barometer sensors, with the 2D Google digital maps to provide accurate and reliable navigation data for UAVs in GPS-denied environments. A neural network object-recognition algorithm is developed to avoid dynamic obstacles, while semantic cloud-points was used to avoid static obstacles. The solution used state vectors (real-time imaging, altitude, azimuth, and world coordinates), which are obtained from the onboard sensors to estimate the position of the UAV. These state vectors can also be obtained from GPS. Given that this research is primarily based on geo-tags obtained from GPS, it would be impracticable to be used in the pipelines where GPS reception is abysmal throughout the entire mission of the UAV.

A UAV navigation solution for confined but partially known indoor environments is presented in [33]. The primary sensors used onboard of the UAV are two scanning laser range finders and an IMU. The authors developed a navigation algorithm that describes the environment by sparse features, including corners and straight lines, where the coordinates of the corner features are pre-known. Given these assumptions, the developed solution showed that two laser scanners could accurately estimate the 3D pose of the UAV. In this paper, we have been able to create a fully autonomous UAV navigation algorithm of completely unknown environments by leveraging on simultaneous localization and mapping method using LiDAR technology.

In [28], He et al. used multi-rotor UAVs to study the inspection rules and methods of wireless transmission towers. The UAVs are used to capture various images from different angles of co-located towers. These images are then processed to draw the transmission lines that need to be inspected. Then the inspection way-points are manually operated, and the photographing position and angle of each way-point from the flight control are recorded through a way-point planning algorithm.

3. Design

Our design of the autonomous UAV system involves running a simulation using the ROS and the Gazebo simulation environment. Besides, we have replicated the navigation algorithm on the physical hardware using a Parrot Bebop2 UAV. Next, we discuss the design of the simulation and physical systems separately.

3.1 Simulation System

LiDAR, also known as active laser scanning, is a remote sensing technology that uses the laser to measure the distance between a moving object and stationary ones. In this project, we used a laser scanner onboard of the UAV that is designed to spin in 360°. It sends invisible laser beams in all directions and catches the reflections from surrounding obstacles while measuring the time it takes for the reflected beam to be received. This way, it measures the distance to surrounding obstacles. Numerous laser measurements are then processed by a computer algorithm to construct a map of such unknown environments.

ROS is an open-source system for controlling robots (e.g., UAVs) by providing APIs and services including hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. ROS has been popularly used to accommodate other third-party applications like Gazebo that provides real physics engine to simulate real-world scenarios of ground robots and UAVs. ROS runs mainly on Unix-based systems such as Ubuntu and Mac OS X.

A node in ROS represents an executable process that controls a single component/sensor in the UAV such as a camera, laser scan, motor, controller, etc. Given that ROS is language-agnostic, nodes are independent and could be written with different programming languages such as Python, C, C++, etc. For instance, a node to get camera feed could be written in Python to communicate with another node that controls the navigation of the UAV written in C++. At the initiation of the simulation, a master node must be allocated to coordinate the communication between all nodes in the simulation environment.

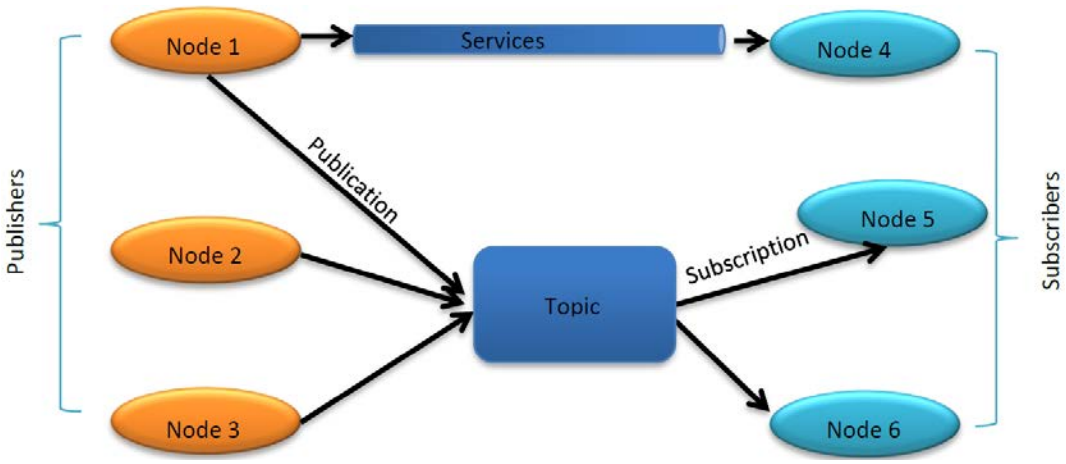


Figure 1. Publisher-subscriber interaction with topics in the robotics operating system

Nodes communicate to each other via a publish-subscribe messaging model over a published topic, which is the information or command to be exchanged between nodes. Each topic has a unique name and message type. As shown in Figure 1, a ROS topic supports many-to-many communication where multiple publishers can publish messages to a single topic and many subscribers can receive messages from a single topic. A topic can be considered as a coordinator between a set of decoupled publisher and subscribers. Moreover, topics can be created or deleted without affecting publishers and subscribers.

Figure 2 shows the simulation system architecture which is divided into three sides: UAV, control and visualization. The figure illustrates the various inter-operations and flow of communication between the components in the three sides. At the UAV side, we used the following sensors onboard of the UAV: LiDAR scanner, camera, and sonar sensors. We then spawned the UAV along with the attached sensors into the Gazebo virtual world.

At the initiation phase of the simulation, we created a new ROS topic and then published the sensed data from the sensors onboard of the UAV, including the point-clouds from the Kinect sensor, scan rays from LiDAR, and odometry (velocity and position) from the sonar sensors, to the newly created topic. The UAV model is also subscribed to the take-off, land, and command velocity (commonly referred to as cmd_vel) topics. The cmd_vel topic uses a Twist message to control the UAV. A Twist message contains a Vector3 type that represents the linear and angular velocities of the UAV. We used the Hector SLAM mapping algorithm [4] to construct a 2D map from the laser scan output. Also, the OctoMap algorithm [5] is used to create a 3D occupancy of the UAV's environment using the point-cloud topic published by the depth camera. Furthermore, the ROS teleop_twist keyboard has been customized to include a take-off and land commands which allow more control on the UAV by publishing to the cmd_vel topic. Finally, we used Rviz [8] as a visualization environment for the 2D map, 3D map, and video feed from the depth camera.

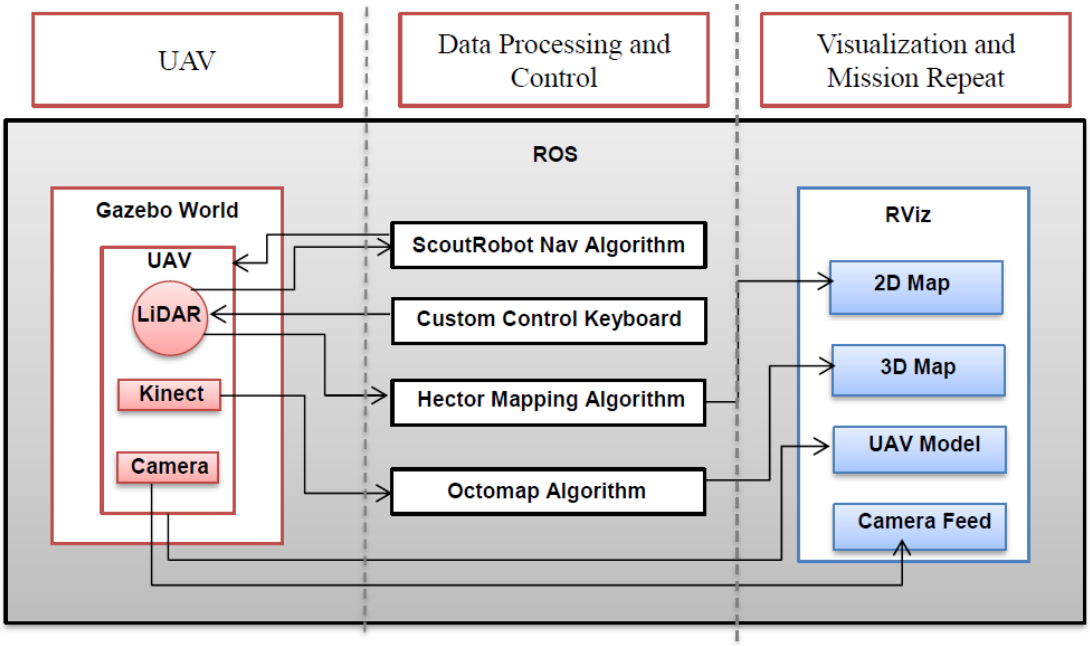


Figure 2. Simulation system architecture

Figure 3 illustrates a flow chart of the developed autonomous navigation algorithm. At take-off, the UAV subscribes to the cmd_vel topic, which provides the movement of the controls in the x, y, z axis. Then, the UAV starts to move forward until it encounters an obstacle at 1.2m distance. When an obstacle is encountered, the minimum values of the laser scans to the left and right directions are calculated. The UAV turns to the path that has a more significant value. If both directions are free, the UAV turns to the right by default until it encounters the next obstacle. If the battery level drops below a certain threshold, the UAV makes a U-turn (i.e., turns 180° relative to the current direction).

3.2 Physical System

Figure 4 shows the physical system architecture, which involves communication between a LiDAR scanner, microcontroller node, WebServer node, and physical Bebop2 UAV. Raw data generated from the physical LiDAR sensor is read over serial interface by an Arduino Mega microcontroller at a rate of about 1,500 data points per second. The data stream has markers delimiting the start of each 360° scan. The microcontroller processes each scan-frame in the data stream to extract the minimum distance at heading zero, minimum, and maximum range at heading -60° and +60° (see Figure 5). The microcontroller formats the extracted data in a JSON form and sends it out over a serial interface to an Arduino MKR WiFi module, which in turn transmits the formatted data to the WebServer node. The WiFi microcontroller and the WebServer node are connected to the drone's WiFi network.

The WebServer node is implemented using Python Flask libraries [6]. This node continually listens for the incoming JSON data from the microcontroller. These feeds are the input to the navigation algorithm, which is deployed on the WebServer node. The output of the navigation algorithm is a decision on which direction the UAV should fly. The WebServer

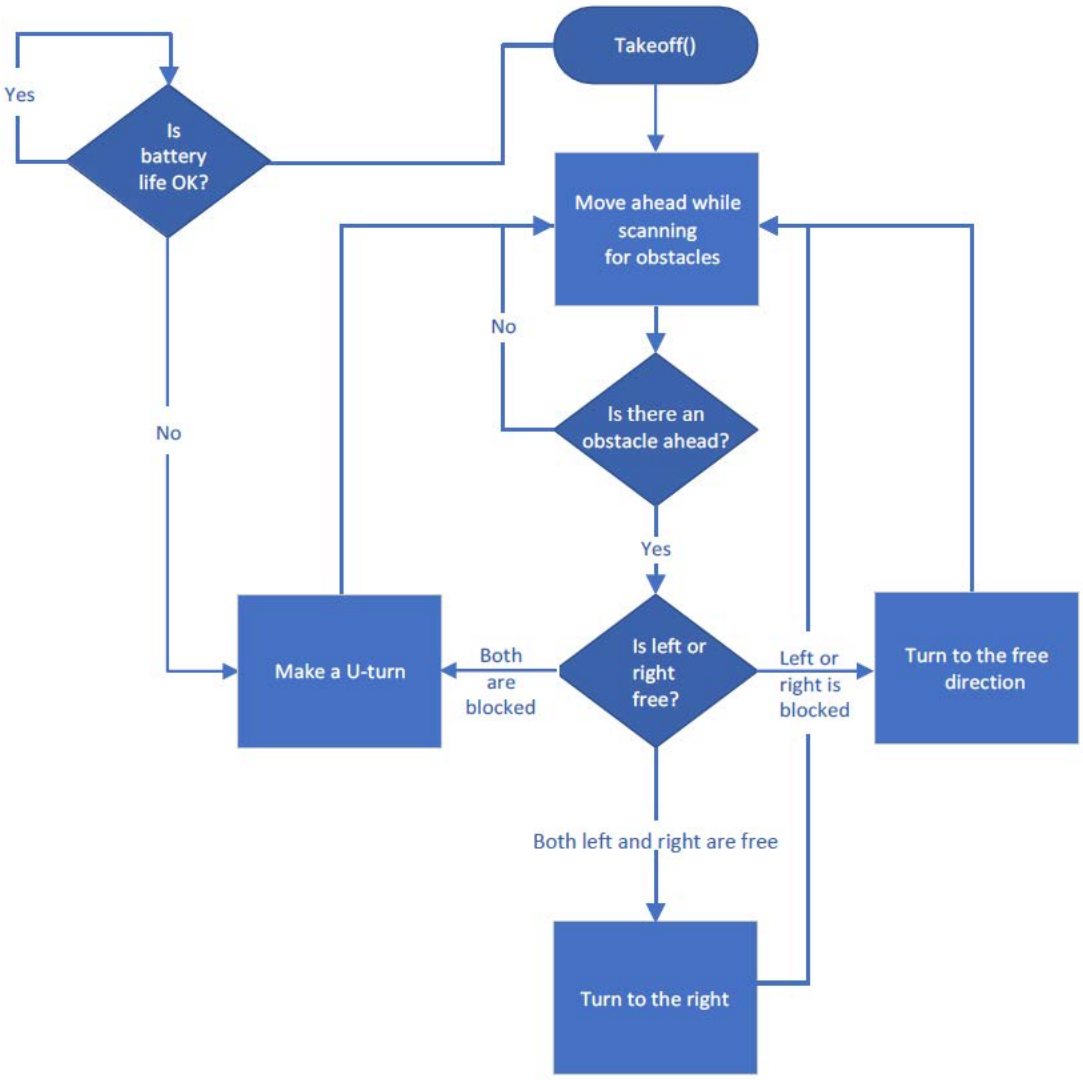


Figure 3. The flow chart for the autonomous navigation algorithm

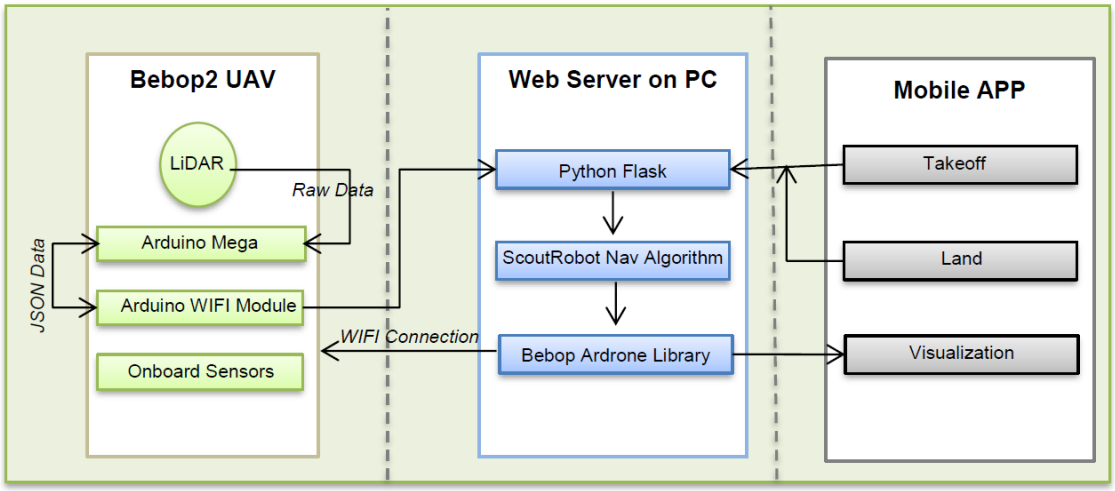


Figure 4. Hardware system architecture

leverages the Ardrone Python libraries to issue the corresponding commands sent to the UAV. We also developed a prototype mobile app which can issue commands to the UAV such as take-off and land commands through the WebServer node. This app can be considered a manual override to the autonomous navigation. In future work, we plan to add more functionalities to this mobile app.

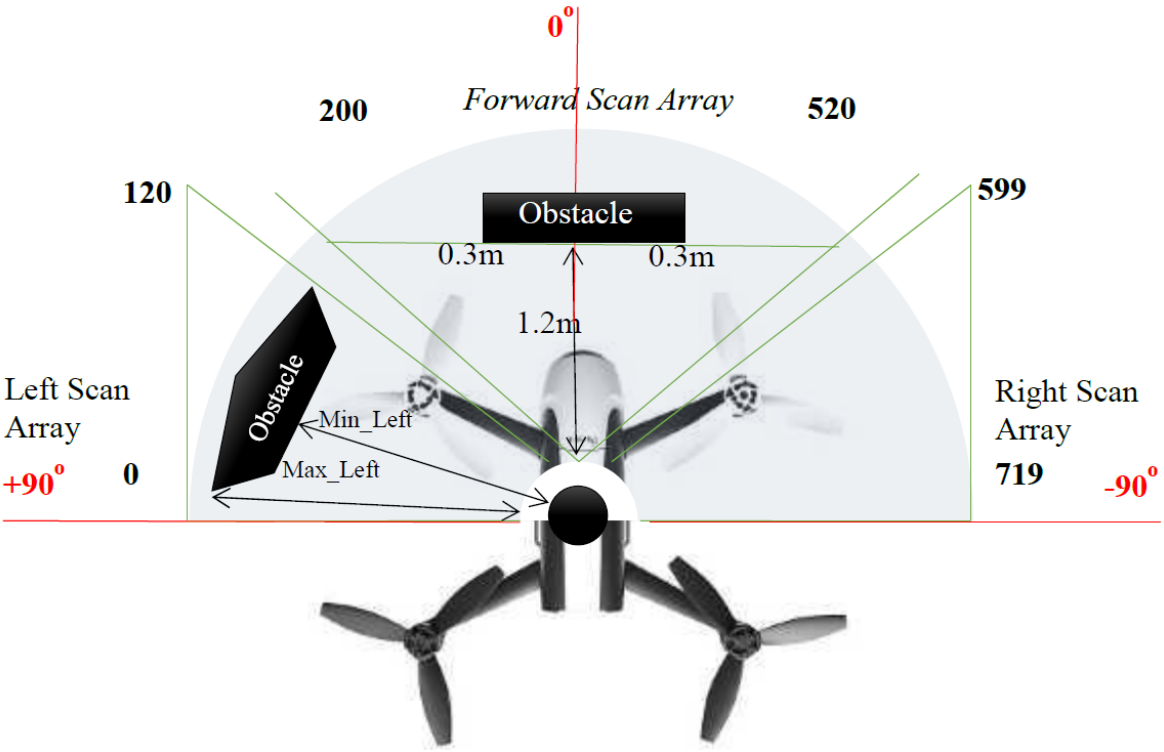


Figure 5 illustrates the navigation plan of the physical UAV system. As shown in the figure, the closest allowable obstacle distance is set to 1.2*m* and an incremental turn value of is set to 15°, which allows for more granular and precise navigation. Three scan regions are defined around the UAV, forward, right, and left. The right corresponds to the 0° to 199° indexes of the scan frames array, the front corresponds to index 200° to 520°, and the left corresponds to index 599° to 719°.

The navigation and obstacle avoidance algorithm is summarized in Figure 6. At the take-off stage, the UAV starts a forward movement (pitch) in the x-plane until it encounters an obstacle at a distance of 1.2 meters ahead. At this point, when it encounters an obstacle, the maximum and minimum distance to obstacles in the right and left regions are compared. If the minimum values (min_left and min_right) are greater than 1.2m (i.e., both directions are free of obstructions within 1.2*m*), the maximum scan values of right and left arrays (max_left and max_right) are then compared and the UAV goes in the direction with the greater value. If both max_right and max_left are equal to infinity, a default right turn direction is selected. If all directions have obstructions within 1.2m, the path of the UAV is set to be blocked. Then, it continues to make incremental turns in the right direction until it finds a free path; otherwise, it makes its way back to the takeoff point.

```
takeoff()

Twist(Lx,Ly,Lz,Ax,Ay,Az) <- (0.5,0,0,0,0,0)
publish forward navigation
subscribe to '/scan'
counter = 0
Twist(Lx,Ly,Lz,Ax,Ay,Az) <- (0,0,0,0,0, 3.14)
turn 180° degrees
counter = 1

if(min_left < 1.2m)
{
     left_range = "free"
}
if(min_right < 1.2m)
{
     right_range = "free"
}
if(front <= 1.2m)
{
 if(max_left==infinity && max_right==infinity)
 {
  Twist(Lx,Ly,Lz,Ax,Ay,Az) <- (0,0,0,0,0,-abs(self.turn_value))
  turn right
 }
 else if(left_range == "free" && right_range == "free")
 {
   if(max_left > max_right)
   {
     Twist(Lx,Ly,Lz,Ax,Ay,Az) <- (0,0,0,0,0,self.turn_value)
     turn left
   }
   else if (max left < max right)
   {
     Twist(Lx,Ly,Lz,Ax,Ay,Az) <- (0,0,0,0,0,-abs(self.turn_value))
     turn right
   }
   else if(left range == "free")
   {
     Twist(Lx,Ly,Lz,Ax,Ay,Az) <- (0,0,0,0,0,self.turn_value)
     turn left
   }
   else if(right range == "free")
   {
      Twist(Lx,Ly,Lz,Ax,Ay,Az) <- (0,0,0,0,0,-abs(self.turn_value))
      turn right
   }
   else
   {
     turn right
   }
}
Shutdown
land()
```

Figure 6. The pseudo code for the autonomous navigation algorithm

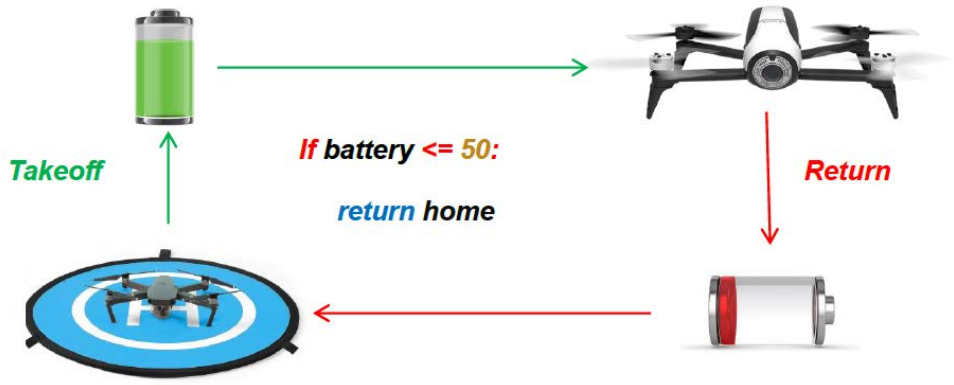


Figure 7. Battery management

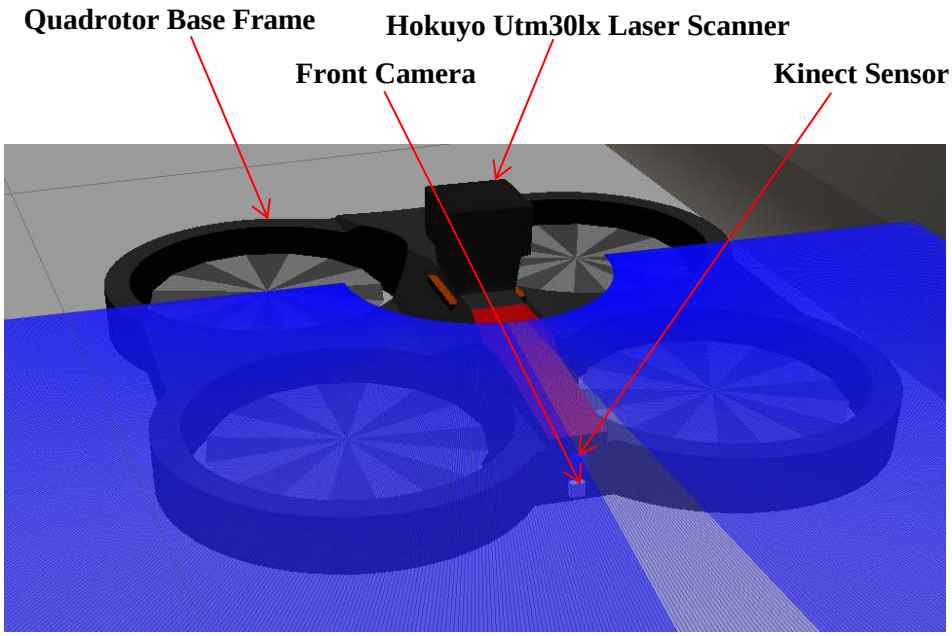
We also monitored the battery life of the UAV during the flight. As shown in Figure 7, the UAV must abort the current mission and return home (i.e., takeoff point) when the battery level drops to a certain threshold. We develop an algorithm to monitor the battery level at both the takeoff and during the flight. The algorithm initiates U-turn navigation on the low battery using the generated map.

4. Implementation

4.1 Simulation System Implementation

Gazebo features a rich test environment with a realistic representation of object mass, stable geometry, collision effects, joint kinematics, daylight effects, and force of gravity. The autonomous navigation algorithm has been deployed on the UAV as it navigates the virtual environments. Several confined environments were created around the UAV to simulate different scenarios. The UAV successfully navigated autonomously through various obstacles in these environments. We also tested a control keyboard that serves as an override to the navigation script for manual intervention in case of emergency. We also constructed both 2D and 3D maps using laser scan integrator and Kinect sensor, respectively, in three distinct Gazebo simulation environments. Also, the video output from the onboard camera can be obtained.

We used the high-quality graphics in Gazebo to provide realistic scenarios for testing our navigation algorithm. This helped us to simulate practical 3D complex indoors and outdoors scenarios using programmatic and graphical interfaces. In the Gazebo environment, we mounted the UAV with a Hokuyo UTM30Lx laser scanner, onboard visual, and Kinect sensor in three distinct environments.

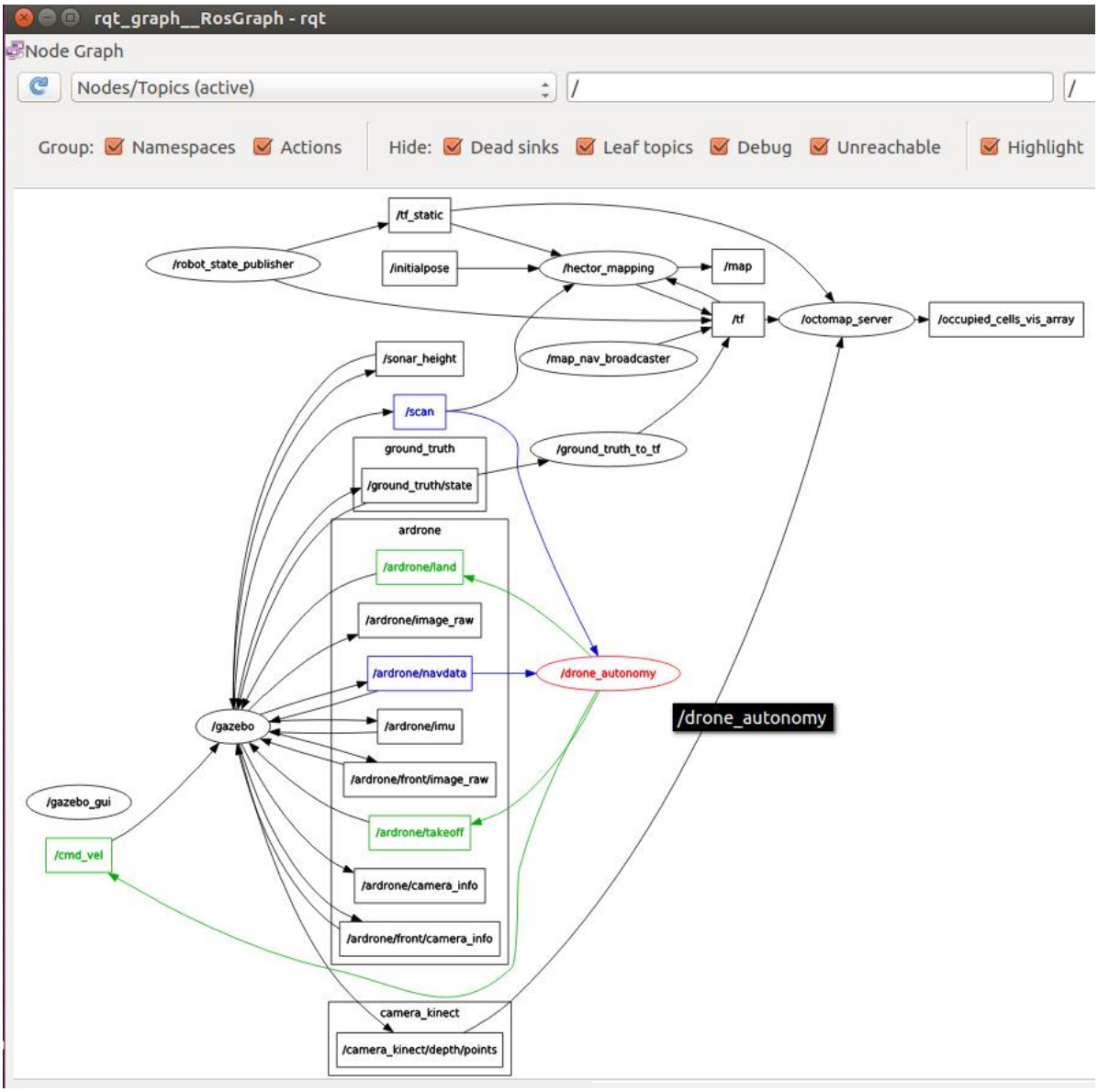


Gazebo presents the physical UAV in real environments as it subscribes and publishes to ROS topics and services. Figure 8 shows the main elements of our simulation including (i) Gazebo World, which represents the real world virtual environment with force of gravity and collision properties; (ii) quadrotor base frame, which represents the main structure of the UAV; (iii) visual front camera, which represents the front image camera mounted as a sensor on the quadrotor (cameras are spawned in gazebo as sensors); (iv) visual bottom camera, which represents the bottom image camera; (v) Kinect sensor, which is a depth sensor that translates the surrounding images to a 3D perception; (vi) Hokuyo UTM30Lx laser scanner, which represents a single axis 180° laser range finder that determines the distance from obstacles to the UAV; and (vii) sonar sensor, which senses the current altitude of the UAV using ultrasonic sound waves.

We used several ROS packages in our simulation including (i) Ardrone_simulator_gazebo7, which contains the Gazebo object and sensor models, quadrotor models, test fly world, and launch files for each object and empty environments; (ii) Scoutrobot-control, which contains the Python ROS node that implements the navigation algorithm; (iii) Teleop_twist_keyboard, which is a generic keyboard control package for ROS that has been modified with some custom commands to manually control the UAV; (v) hector_slam, which contains ROS nodes for implementing the SLAM method by translating the laser scans into a 2D maps; and (iv) Octomap_mapping, which is used to implement the 3D occupancy grid from the point-clouds generated by the depth camera (i.e., Kinect sensor) onboard of the UAV.

The Octomap 3D mapping framework was used in our simulation to partition the three-dimensional space using the Octree data structure, as shown in Figure 9. Octree recursively subdivide a 3D space into eight octants. It provides a C++ mapping library for implementing a 3D occupancy grid, which was suitable for the requirements of this project as Octomap can cumulatively model arbitrary environments without prior assumptions about it. It also allows the generation of large maps without previously knowing the extent of the map.

Figure 8. The UAV simulation model in Gazebo



We used ROS visualization (RViz) for displaying the UAV sensor data and state information generated by ROS nodes. In particular, we visualized the virtual model of the UAV, laser scanner outputs, onboard cameras, Kinect sensors, and results of the mapping algorithms. Also, we used rqt-graph to visualize the communication between all active nodes and topics in the simulation. Figure 9 illustrates the graphical visualization of the ROS computation graph, which is generated from our simulation.

We tested the autonomous navigation algorithm in three different environments. Figure 10 shows the first simulation scenario, which contains a Willow Garage on the right-hand side, and the 2D and 3D map visualization in RViz on the left-hand side. Figure 11 shows the second simulation scenario, which contains a Maze that represents a confined loop environment in Gazebo. In this

scenario, we displayed the UAV's camera feed using RViz. Figure 12 shows the third simulation scenario, which contains a house loop that represents a closed loop outdoor environment in Gazebo. Both the camera feed and 3D map were visualized using RViz.

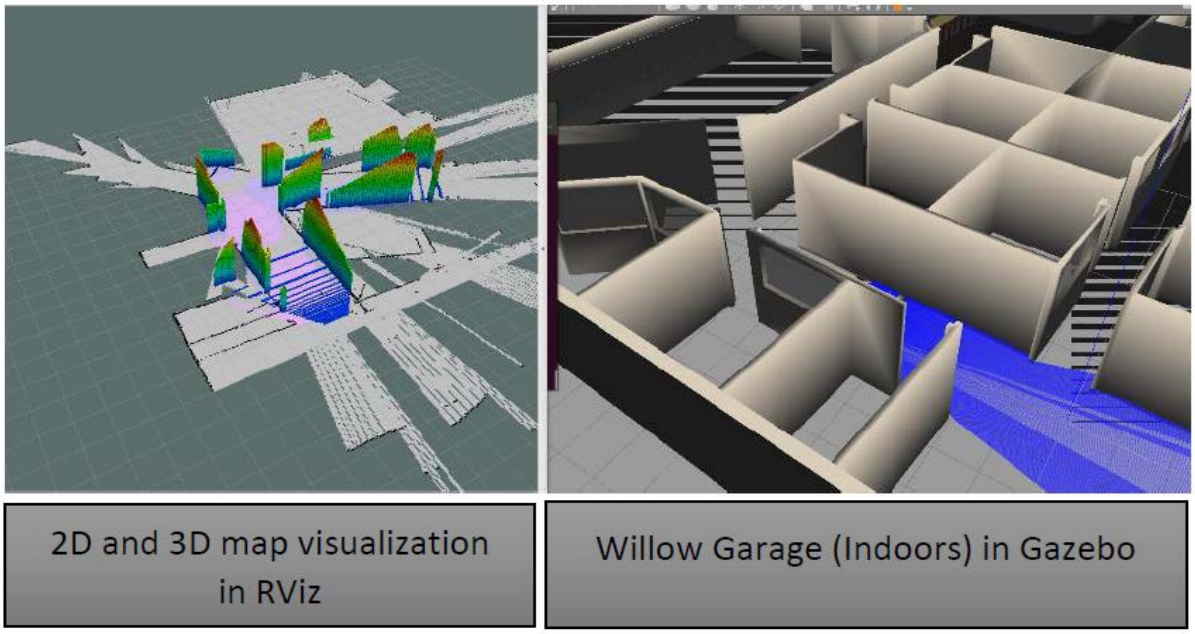
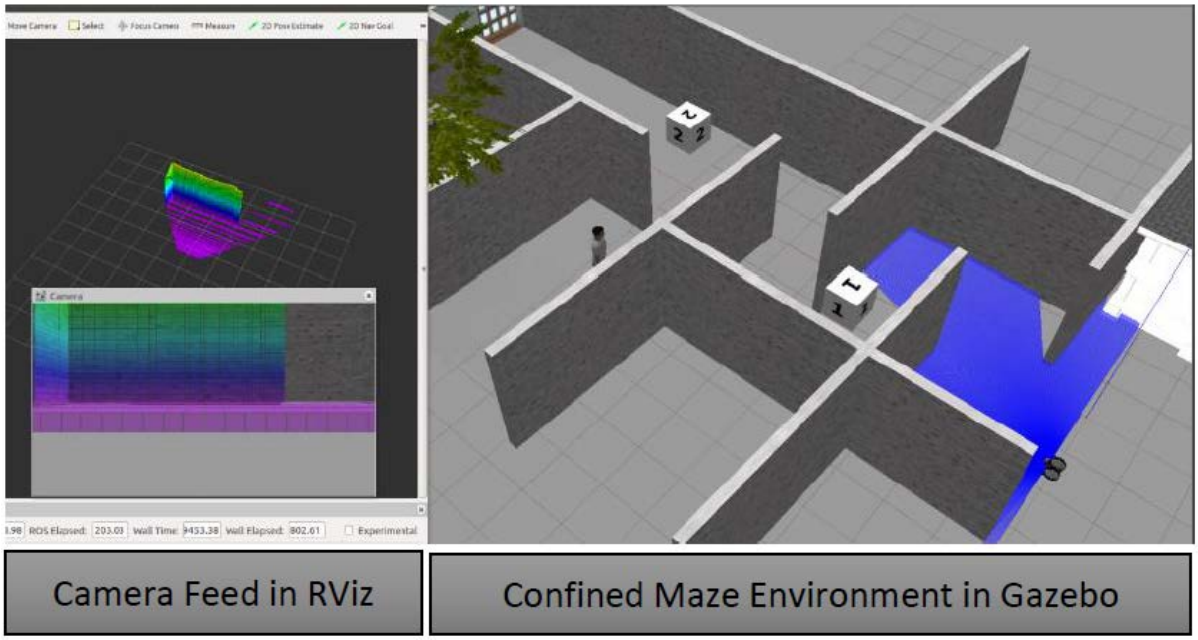


Figure 10. Simulation scenario 1. Willow Garage



4.2 Physical System Implementation

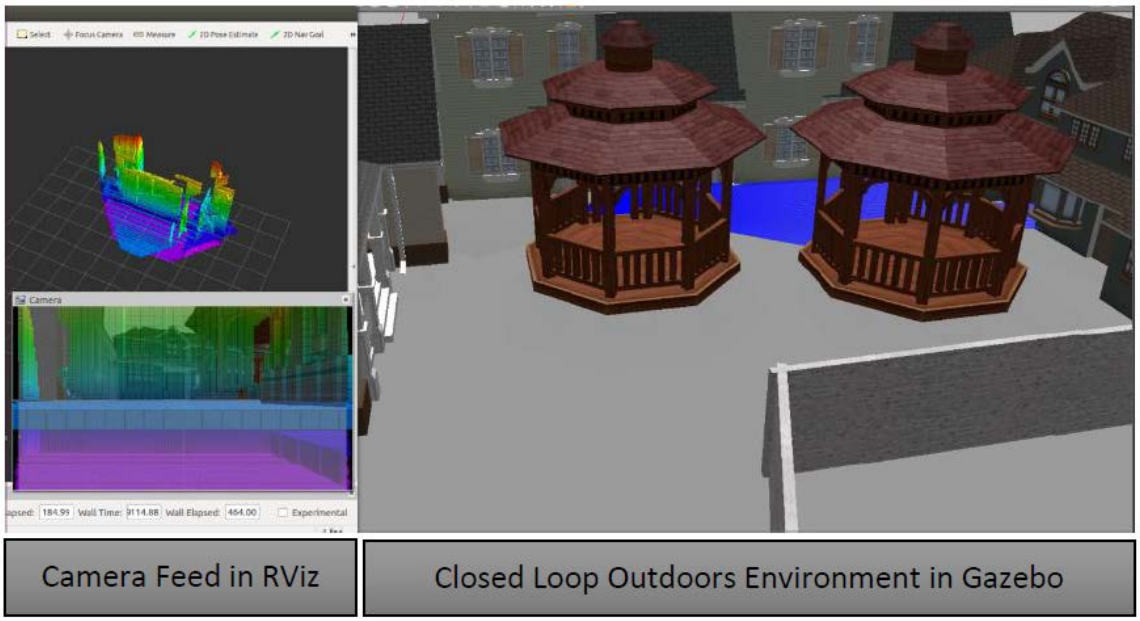


Figure 11. Simulation scenario 2. Maze

We replicated the simulation on the actual physical hardware using a Bebop2 Ardrone, Arduino microprocessor, WiFi module, and a web application server built on the Python Flask library. Figure 13 illustrates the communication between the components of the physical UAV system. We used two RPLIDAR laser scanners running on 3,000rpm one Arduino UNO R3 microprocessor, one Arduino ESP8266 WiFi module, one Parrot Bebop 2 drone, and one laptop, which acts as a ground station. The microprocessor, which is mounted on the drone, collects the LIDAR readings generated from the two scanners. These scan readings are then sent to both the ground station and the drone via the WiFi module. The drone uses the reflected scan data for obstacle avoidance, while the ground station uses them to construct the 3D map of the environment.

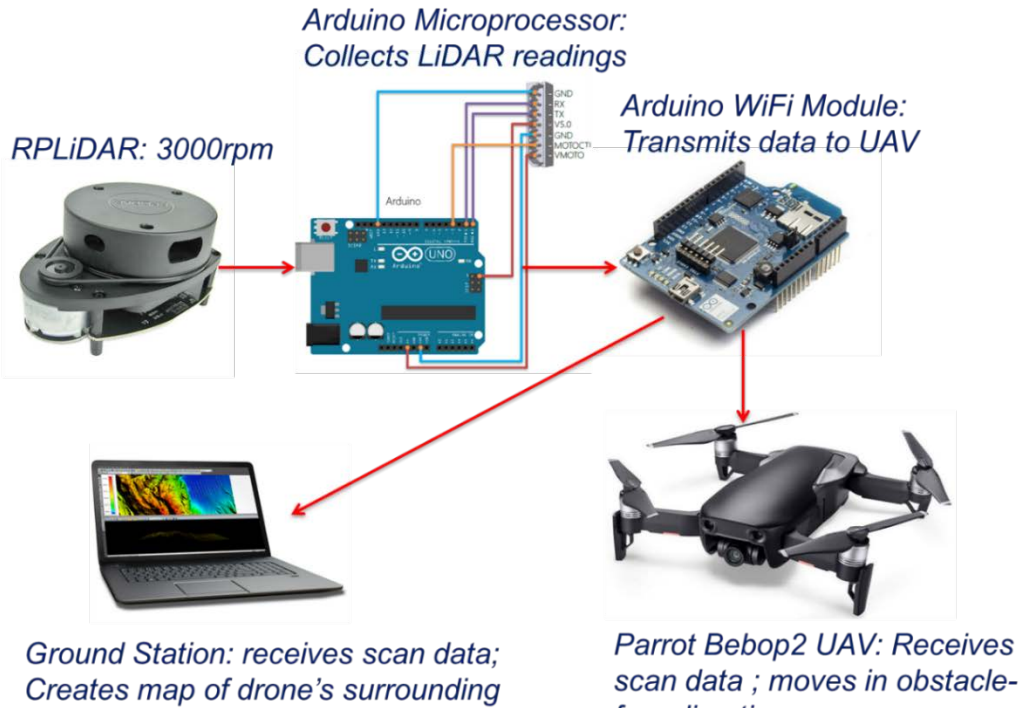


Figure 12. Simulation scenario 3. Gazebo House Loop

Figure 13. The physical system components

As shown in Figure 14, we selected compact hardware to enable the UAV to fly in confined environments with limited space. The Bebop2 Ardrone satisfies the requirements for size, weight, camera image quality, battery life and programmability. This UAV has several helpful features such as a camera, which films in 1080p full HD. Its ability to halt entirely in just four seconds is claimed to be record-breaking. It weighs 500g and offers 25 minutes of flight time. However, it has some limitations such as its low weight-carrying capacity; it can only carry 300g at takeoff. This was a significant drawback in our physical implementation.

Figure 14. The UAV physical system

We selected the RD LiDAR laser scanner because it is one of the lightest weights in the category of 360° rotating omnidirectional laser scanners –it weights around 170*g*. It can scan a 360° environment within a range of 12 meters at 10Hz adaptive scan frequency rate. The dimensions of the scanner are 98.5mm wide and 60mm high.

The RPLIDAR driver library for Arduino Mega requires binding to a hardware serial port, which places more work on the CPU. However, hardware serial ports are non-blocking which consumes CPU cycles only for processing the incoming data. We also needed one serial port for debugging our code and another port for the communication with the WiFi module.

The Arduino MKR1010 WiFi module was used to send the scan data frames to the ground station via an API call. These scans are then converted to pitch, yaw and row control signal to the Bebop UAV. Arduino Mega has no in-built WiFi module to communicate directly to the ground station; thus, we used a separate WiFi module.

We built a web server application to control the UAV using the Python flask library manually. Like the teleop_twist keyboard in ROS, the app serves as a control and override interface for the

UAV in case of any script malfunctions.

The web application continuously listens to data transmission transmitted from the WiFi module onboard of the UAV. The application can run three commands: takeoff, land and shutdown. These commands were implemented as HTTP GET requests. When the address extension is called in a web browser, it runs the function bounded to that address. For instance, http://192.168.42.5:5000/takeoff runs the takeoff function, which sends a takeoff command to the UAV. The application interface can also be accessed by the IP address and port number of the web server without an extension as follows: http://192.168.42.5:5000/.

We run a DHCP network server on the web server which automatically provides and assigns IP addresses to the ground station. The DHCP server also runs a nav command as an HTTP POST request through which it receives the navigation information sent from the Arduino WiFi module in JSON format.

5. Conclusions and Future Work

This paper presented an autonomous UAV system that can safely navigate GPS-denied environments using LiDAR technology while generating a 2D floor plan and 3D maps for such unknown environments. This work has tremendous applications in different domains, such as in mining tunnels, where the shape of tunnels is unknown. Also, it can be used in rescue operations where people are trapped in collapsed tunnels, such as in the case of the twelve Thai footballers trapped in a flooded cave in July 2018.

The implementation of the UAV system was carried out in two phases. In the first phase, we ran a realistic simulation using Gazebo physics engine and ROS. We tested the autonomous navigation and mapping algorithms using multiple virtual worlds in the simulation environment before applying them directly to the physical system. The significance of the simulation results made us more confident of replicating it on the physical hardware using a Bebop2 Ardrone, Arduino microprocessor, WiFi module, and a web application server built on the Python Flask library. We also developed an algorithm to monitor and manage the battery life of the UAV such that it returns home when the battery lie drops down to a certain percentage.

We expect that this research would increase the open-source knowledge base in the area of autonomous UAV navigation by publishing the source codes to the public domain (see https://github.com/abelmeadows/scoutrobot).

In on-going work, we are examining the opportunities for generalizing our approach. Multiple coordinated UAVs can be deployed to create a UAV swarm networking system for completing challenging missions in a more efficient and economical approach.

Also, we plan to develop an approach for sharing sensor data between UAVs with multi-modal sensing requirements. Specifically, we will compose ModeSens [10, 23] with ShareSens [17, 20] to support such capability. ModeSens allows multi-modal sensing requirements of an application to be programmed separately from its function. Programmers can specify different modes for an application, the sensing needs of each mode, and the sensed events that trigger mode transition.

ModeSens monitors for mode transition events, and dynamically adjusts the sensing frequencies to match the current mode's requirements. ShareSens is a mechanism that opportunistically economizes on a collection of sensor data by merging sensing requirements of multiple applications, thereby achieving significant power and energy savings. The composition of ModeSens and ShareSens will be useful for supporting the sensing needs of a wide range of research [13–15, 18, 21, 24] and applications [10–12, 16, 19, 22, 24].

Furthermore, we want to further strengthen our evaluation by measuring the error of map construction. Finally, experiments with more massive datasets are needed to further study the robustness of our approach.

Acknowledgments

Support from Bechtel Corporation is gratefully acknowledged.

References

[1] Arduino, Last accessed on October 16th, 2019.

[2] Bebop ardrone, Last accessed on October 16th, 2019.

[3] Gazebo, Last accessed on October 16th, 2019.

[4] Hector slam, Last accessed on October 16th, 2019.

[5] Octomap, Last accessed on October 16th, 2019.

[6] Python flask, Last accessed on October 16th, 2019.

[7] Robot operating system ROS, Last accessed on October 16th, 2019.

[8] Rviz, Last accessed on October 16th, 2019.

[9] M. Aghaei, F. Grimaccia, C. A. Gonano, and S. Leva. "Innovative automated control system for pv fields inspection and remote control." IEEE Transactions on Industrial Electronics, 62(11):7287–7296, 2015.

[10] A. A. Ahmed. "A modular approach to programming multi-modal sensing applications." In Proceedings of the IEEE International Conference on Cognitive Computing, ICCC '18, pages 91–98, San Francisco, USA, 2018.

[11] A. A. Ahmed. "A model and middleware for composable IoT services." In Proceedings of the International Conference on Internet Computing & IoT, ICOMP '19, pages 108–114, Las Vegas, USA, 2019.

[12] A. A. Ahmed and T. Eze. "An actor-based runtime environment for heterogeneous distributed computing." In Proceedings of the International Conference on Parallel & Distributed Processing, PDPTA '19, pages 37–43, Las Vegas, USA, 2019.

[13] A. A. Ahmed and N. Jamali. "An actor-based approach to coordinating crowdsourced services." International Journal of Services Computing (IJSC), 2(3):43–55, 2014.

[14] A. A. Ahmed and N. Jamali. "Coordinating crowd-sourced services." In Proceedings of MS, pages 92–99, Alaska, USA, 2014.

[15] A. A. Ahmed and N. Jamali. "CSSWare: A middleware for scalable mobile crowdsourced services." In Proceedings of MobiCASE, pages 181–199, Berlin, Germany, 2015.

[16] A. A. Ahmed and N. Jamali. "CSSWare: An actor-based middleware for mobile crowd-sourced services." In Proceedings of the 2015 ACM 12th International Conference on Mobile and Ubiquitous Systems: Computing,

Networking and Services (Mobiquitous '15), pages 287–288, Coimbra, Portugal, 2015.

[17] A. A. Ahmed and N. Jamali. "ShareSens: An approach to optimizing energy consumption of continuous mobile sensing workloads." In Proceedings of the 2015 IEEE International Conference on Mobile Services (MS '15), pages 89–96, New York, USA, 2015.

[18] A. A. Ahmed and N. Jamali. "Supporting resource bounded multitenancy in akka." In Proceedings of the ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity (SPLASH Companion 2016), pages 33–34, 2016.

[19] A. A. Ahmed and N. Jamali. "An actor-based middleware for crowd-sourced services." EAI Transactions on Mobile Communications and Applications, (vol 17):1–15, 2017.

[20] A. A. Ahmed and N. Jamali. "Opportunistic sharing of continuous mobile sensing data for energy and power conservation." IEEE Transactions on Services Computing, (doi:10.1109/TSC.2017.2705685):1–14, 2017.

[21] A. A. Ahmed and N. Jamali. "Approaching actor-level resource control for akka." In Proceedings of the IEEE Workshop on Job Scheduling Strategies for Parallel Processing, JSSPP '18, pages 1–15, Vancouver, Canada, 2018.

[22] A. A. Ahmed and N. Jamali. "A model for representing mobile distributed sensing-based services." In Proceedings of the IEEE International Conference on Services Computing, SCC '18, page 282–286, San Francisco, USA, 2018.

[23] A. A. Ahmed and J. Nadeem. "ModeSens: An approach for multi-modal mobile sensing." In Companion Proceedings of the 2015 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity, SPLASH Companion 2015, pages 40–41, Pittsburgh, PA, USA, 2015.

[24] A. A. Ahmed, D. Wang, and N. Jamali. "Supporting resource control for actor systems in akka." In Proceedings of the International Conference on Distributed Computing Systems (ICDCS 2017), pages 1–4, 2017.

[25] J. Q. Cui, S. Lai, X. Dong, and B. M. Chen. "Autonomous navigation of uav in foliage environment." Journal of Intelligent & Robotic Systems.

[26] [26] R. S. Filho, C. Huang, B. Yu, R. Venkataramana, A. El-Messidi, D. Sharber, J. Westerheide, and N. Alkadi. "Semi-autonomous industrial robotic inspection: Remote methane detection in oilfield." In Proceedings of the IEEE International Conference on Edge Computing (EDGE), pages 17–24, 2018.

[27] S. Gallardo-Saavedra, L. Hernández-Callejo, and O. Duque-Perez. "Image resolution influence in aerial thermographic inspections of photovoltaic plants." IEEE Transactions on Industrial Informatics, 14(12):5678–5686, 2018.

[28] T. He, Y. Zeng, and Z. Hu. "Research of multi-rotor uavs detailed autonomous inspection technology of transmission lines based on route planning." IEEE Access, 7(1):955–965, 2019.

[29] N. Jeong, H. Hwang, and E. T. Matson. "Evaluation of low-cost lidar sensor for application in indoor uav navigation." In Proceedings of the IEEE Sensors Applications Symposium (SAS), pages 1–5, 2018.

[30] R. Li, J. Liu, L. Zhang, and Y. Hang. "Lidar/mems imu integrated navigation (slam) method for a small uav in indoor environments." In Proceedings of the DGON Inertial Sensors and Systems (ISS), pages 1–15, 2014.

[31] A. Nemati, M. Sarim, M. Hashemi, E. Schnipke, S. Reidling, W. Jeffers, J. Meiring, P. Sampathkumar, and M. Kumar. "Autonomous navigation of uav through gpsdenied indoor environment with obstacles." AIAA Infotech at Aerospace, pages 1–7, 2015.

[32] R. Ravi, Y. Lin, T. Shamseldin, M. Elbahnasawy, M. Crawford, and A. Habib. "Implementation of uav-based lidar for high throughput phenotyping." In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS'18), pages 8761–8764, 2018.

[33] F. Wang, K. Wang, S. Lai, S. K. Phang, B. M. Chen, and T. H. Lee. "An efficient uav navigation solution for confined but partially known indoor environments." In Proceedings of the IEEE International Conference on Control Automation (ICCA), pages 1351–1356, 2014.

[34] S. Zhang, S.Wang, C. Li, G. Liu, and Q. Hao. "An integrated uav navigation system based on geo-registered 3d

point cloud." In Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pages 650–655, Daegu, South Korea, 2017.

[35] X. Zhang, Z. Peng, and H. Liu. "Autonomous positioning and navigation for uav in gps denied environments." In Proceedings of the International Micro Air Vehicle Competition and Conference, pages 77–82, Beijing, PR of China, okt 2016.

Biographical Information

Dr. AHMED ABDELMOAMEN AHMED, Ph.D., is an assistant professor in the Department of Computer Science at Prairie View A&M University. He worked in the computer science field for 12 years. His research interests are in the fields of parallel and distributed systems with applications in domains such as IoT, sensor-based services, autonomous robotic navigation, resource coordination for concurrent systems, and efficient mobile sensing.

Mr. ADEL OLUMIDE is a cybersecurity engineer at Visa Inc. headquartered in Foster City, California, USA. He obtained his master's degree in Computer Information Systems from Prairie View A&M University, Texas, USA; and his bachelor's degree in Electrical Engineering from the Federal University of Technology Akure, Nigeria. His research interests are in the areas of robotics, autonomous navigation, systems engineering, cloud computing, network security, and project management.

Mr. ADEOLUWA AKINWA is a system and automation engineer at PayPal Inc. headquartered in San Francisco, California, USA. He holds a master's degree in Computer Science from Prairie View A & M University and majored in Electronics and Electrical Engineering for his bachelor's degree, which he obtained from Obafemi Awolowo University, Ile-Ife, Nigeria. His research interests are in the areas of robotics, autonomous navigation, and system automation.

DR. MOHAMED F. CHOUIKHA, PhD., is a chief scientist and executive director of the SECURE Cybersecurity Center of Excellence at Prairie View A&M University. He is the founding director of the Intelligent Community-Center of Academic Excellence, the founding and first Director of the Center of Applied High Performance Computing, and one of the four founding directors of the Washington Academy of Biomedical Engineering. He has been a consultant to a number of companies in the Washington area and was a board member of Quateams Inc.