LANDING OF AN ARDRONE 2.0 QUADCOPTER ON A MOBILE BASE USING FUZZY LOGIC

PATRICK BENAVIDEZ, JOSUE LAMBERT, ALDO JAIMES AND MO JAMSHIDI
Department of Electrical and Computer Engineering
University of Texas at San Antonio
San Antonio
USA
patrick.benavidez@gmail.com, josueilambert@yahoo.com, aldo.jaimes@gmail.com, moj@wacong.org

ABSTRACT—In this paper, a fuzzy control system is presented for an unmanned aerial vehicle (UAV) which provides aerial support for an unmanned ground vehicle (UGV). The UGV acts as a mobile launching pad for the UAV. The UAV provides additional environmental image feedback to the UGV. Our UAV of choice is a Parrot ArDrone 2.0 quadcopter, a small four rotoried aerial vehicle, competent for its agile flight and video feedback capabilities. This paper presents design and simulation of fuzzy logic controllers for performing landing and altitude control. Image processing and Mamdani-type inference are used for converting sensor input into control signals used to control the UAV.

Key Words: Quadrotor, UGV, UAV, cooperative robots, fuzzy logic, ROS, image processing

1. INTRODUCTION

Quadcopters are a class of four-rotored vertical takeoff and landing (VTOL) aerial vehicles. They have been shown to provide stable acrobatic flight as demonstrated in many hobbyist, research and commercial grade products. To provide the flight characteristics that quadcopters are prized for, huge energy consumption is inevitable. For many small low-cost quadcopters, the battery life is typically limited to minutes of flight time due to the weight of the batteries and power draw of the motors. Limited time of use for quadrotors creates a problem for researchers to solve. Researchers are trying to find ways to improve the effectiveness of quadcopter given current battery systems and motor efficiencies. One method many researchers are investigating is limiting the flight time of quadrotors and providing a base station to act as either a landing pad for battery conservation [1], hot swapping [2, 3] or battery recharging purposes [4]. In [2], researchers created a fixed mechanical base station for a quadcopter to proceed with a hot-swap of a battery. In [1] researchers performed landing maneuvers on a mobile base using a low-cost camera sensor to sense infrared Light Emitting Diodes (LEDs) acting as beacons on the mobile base. Researchers from the University of Waterloo, Canada simulated coordinated landing of a quadcopter using nonlinear control methods. With their methods, they designed a joint decentralized controller that
attracts the two linearized systems together via coupled state information [3]. In [4] researchers developed a system where an Adept Mobile Robotics P3AT unmanned ground vehicle (UGV) provided services for mobile landing and target identification for visual inspection by an ArDrone quadcopter [4]. For control of the quadcopter, the researchers utilized classical controllers for controlling the altitude, pitch, roll and yaw. Vision input was used by the researchers on both platforms for navigation and landing control. Actuators onboard the UGV platform performed error correction post-landing by shifting the ArDrone to the optimal landing position. Fuzzy logic controllers for a more comprehensive low-level quadcopter model were presented by Raza in [5]. Learning based model predictive control of a quadcopter was proposed and tested by Bouffard et al. in [6]. Engel et al. [7] presented Simultaneous Localization and Mapping (SLAM) with PID heading control for figure flying. Kendoul et al. [8] presented a non-linear flight controller for landing on a stationary target. Venugopalan et al. [9] presented an autonomous landing controller for an UGV and an unmanned kayak with a 75% success rate. Voos et al. [10] introduced a non-linear tracking and landing controller. Decentralized adaptive stabilization and tracking control for a quadcopter was introduced by Mohammadi, et al. in [11, 12]. Many successful algorithms, methods and programs have been shared with open source software communities. Engel et al. in [13] provided their sample applications of their research as a simulator with ROS/Gazebo support. Disturbances, such as wind, environmental topology and other external actors, are modeled in some quadcopter simulators. Meyer et al. in [14] presented a ROS/Gazebo compatible quadcopter simulator complete with simulated wind disturbances. Some versions of the Gazebo simulator, those newer than that used by Engel et al. have built-in models of wind, sensor noise, and camera noise [15].

In the literature for collaborative UAV/UGV teams, there exists the problem of the UAV losing visual tracking of the UGV. One case where UAVs can lose sight of a UGV is upon approach. When approaching a mobile base, UAVs with a single fixed camera, or a dual (front/bottom) camera configuration as on the ArDrone, can have a significant blind spot due to camera positioning. To combat this problem, researchers have primarily implemented external environmental sensing. Kushleyev et al. [16] employ an expensive environmentally-fixed Vicon motion capture camera system, paired with infrared markers on each robot, that provides 100 Hz robot position updates to a computer. In [17], Achtelik et al. use stereo infrared cameras to accurately track a quadcopter outdoors. Michael et al. [18] forced a static restriction on the mobile base so that the quadcopter could return to a memorized position given that the quadcopter had mapped its environment. The landing pad had to remain static as the quadcopter could not sense the location of the mobile base from above.

In this paper we utilize fuzzy logic controllers to control heading, altitude, approach, and hovering for an ArDrone 2.0 using visual and sonar-based distance feedback. Fuzzy logic is used for the ability to shape the control surfaces generated by rules in our fuzzy inference system. We draw inspiration from the automated UGV/UAV inspection system in [4] for testing our system. Visual feedback is provided by identification of visual tags, similar to the ones used in [4], using open source software to calculate the tag identification match and orientation. Compared to [4], we utilize fuzzy logic controllers instead of classical controllers to simplify the design of quadcopter maneuvers. Also we use a single multi-resolution visual tag that changes by distance, instead of using multiple spatially separated tags. We also do not rely on more than the coordinate systems relative to the cameras. This is a major difference from the work of others that have constructed a fully mapped and controlled test environment to provide high resolution location services. In addition to creating a base set of fuzzy logic controllers for quadcopter landing, this work presents a highly expandable simulation and experimentation setup for collaborative teams of UGV/UAVs.
The rest of the paper is organized as follows: Section 2 details the control problems handled by the fuzzy controllers. Section 3 details the software and hardware experimentation with the controllers. Section 4 provides results from hardware and software tests. Finally, Section 5 provides conclusions on the system and future work with the UAV/UGV system.

2. THE PROPOSED CONTROL SYSTEM

In this section we describe the controller design workflow, the control problems addressed in this work and the control variables used to control the quadcopter. A sample scenario is depicted in Figure 1 which illustrates a general class of problems that a UAV is meant to solve for a UGV. In the figure, the UAV provides aerial support in the form of vision assistance to a UGV that has limited sight. This is the case of searching a cluttered building for a target of interest. In return, the UGV provides landing, recharging, and/or physical services to the UAV which is limited in terms of battery life and/or manipulation capability. We intend to have the UGV provide tracking support to the UAV in the future. This work covers the UAV tracking and landing on the UGV.

2.1 Controller Design Workflow

For the controller design we used a circular workflow process that is depicted in Figure 2. The workflow is as follows. First, the membership functions and/or rules are modified to meet prior knowledge of system response. Next, the response of the system is evaluated using a simulated ArDrone 2.0. These prior steps are repeated until a desirable controller is obtained. Once the controller is sufficient for hardware testing, the controller is applied to the hardware. Based on the hardware test the steps are repeated until the desired hardware system response is obtained.
2.2 Control Tasks

The control tasks described in this section are shown in Figure 3 which presents a depiction of the system controller. Figure 4 shows a depiction of the control tasks.
2.2.1 Altitude Control Task

The altitude control task is to have the quadcopter reach and maintain a desired altitude. On a quadcopter, such as the ArDrone, the control input variable is the throttle and the sensed variable is the altitude. In Figure 5 the variable \( h \) is the desired altitude of quadcopter, the variable \( h_s \) is the sensed altitude from the altitude sensor, and \( F \) is the applied force.
Figure 5. Altitude Control Task

With the ArDrone 2.0, altitude information \( h_S \) can be sensed from the sonar sensor, or be estimated using image processing methods on a known target performed on the bottom-facing QVGA camera. In either implementation, the goal of the task for \( e_h \), the absolute error of sensed and desired altitudes, to equal zero.

2.2.1 Heading Control Task

The heading control task is to control the quadcopter to reach a desired orientation angle with the mobile landing base. To do this, the quadcopter needs to translational velocity along its local y-axis, while rotating along its local z-axis to reduce the orientation angle. Overall, the goal is to have the same orientation on both the UAV and UGV. With the same orientation, the UAV can then approach the UGV for landing operations. To approach the UGV once oriented, the quadcopter must translate along its x-axis to get close enough to use its bottom cameras to detect the landing pad. Figure 6 depicts the heading control task.
In Figure 6, the variable $P_L$ is the desired landing position, $P_{MB}$ ($X_{MB}, Y_{MB}, Z_{MB}$) is the pose relative to the mobile landing pad local frame, and $d$ is the distance between the quadcopter center and the landing point $P_L$ ($X_L, Y_L, Z_L$). Once the target is detected by the bottom camera, either the landing control can be used to lower the craft or hovering control with the mobile base as a reference.

### 2.2.3 Landing Control Task

The landing control task is to control the quadcopter to maintain a minimum orientation angle and positional error on descent towards a visual marker. To do this, the quadcopter needs to translate across both the x and y axes to reduce positional error, rotate along the z-axis to reduce orientation error and while descending along the z-axis to land. Figure 7 depicts the landing control task.

![Landing Control Task Variables](image)

In Figure 7, the variable $h_{MB,LP}$ is the height of the landing pad with respect to the center of gravity of the mobile base, $h_{MB,COG}$ is height of the center of gravity of the mobile base with respect to the ground, $h_S$ is the altitude sensed by the quadcopter, $P_Q$ ($X_Q, Y_Q, Z_Q$) is the pose relative to the quadcopter local frame, and $P_{MB}$ ($X_{MB}, Y_{MB}, Z_{MB}$) is the pose relative to the mobile landing pad local frame and $P_L$ ($X_L, Y_L, Z_L$) is the landing point. The goal of the task is to minimize the respective deviations between axes in $P_L$ and $P_Q$.

### 3. SIMULATION EXPERIMENT STUDIES

#### 3.1 The Applied Software Packages

In development of the proposed fuzzy controllers, several open source software packages, including the Free Fuzzy Logic Library (FFLL) [19], FuzzyLite [20] and jFuzzyLogic [21], were examined for their capabilities, features, and compatibility with the Robot Operating System (ROS). FuzzyLite was chosen as the fuzzy logic software package for these studies. FuzzyLite is an open source Fuzzy Logic Controller library, written in C++, which has a Graphical User Interface (GUI), called QtFuzzyLite, for designing fuzzy logic controllers.

To develop the fuzzy controllers, a combination of ROS [22], ROS Gazebo [23] with the TUM ArDrone Simulator [13], and ROS package ar_track_alvar [24] were used. ROS is a software package that allows for the transport of sensor and control data via "topics" using the publisher/subscriber message passing model. The TUM ArDrone Simulator is a package for ROS...
Gazebo that allows for simulation of the ArDrone in 3D environments. The ROS package ar_track_alvar is used for unique tag identification.

### 3.2 Detailed Controller Design Workflow

The workflow of the designed fuzzy logic controller is listed as follows. First membership functions and/or rules are created/modified in FuzzyLite GUI. Next, the fuzzy logic controller C++ code is exported using the FuzzyLite GUI. The generated code is inserted into the controller node in ROS and compiled. Next, the controller node is ran connected to TUM ArDrone simulator. The controller is evaluated for its readiness for application to the hardware ArDrone. If the test shows unacceptable results, the process is restarted with modifications made to membership functions. If the tests were successful, the ArDrone hardware is tested with the controller. If the hardware test shows unacceptable results the controller is revised based on the observations.

### 3.3 ArDrone Controller and Driver ROS Nodes

ROS provides a framework for information management and passing via various processes that govern device drivers, sensor data, signal processing, and control. Sensor and control data topics are subscribed to by controller, plant and simulator processes.

#### 3.3.1 ROS Package ‘ardrone_autonomy’

The ROS topics listed in Table 1 are inputs to ‘ardrone_autonomy’ which are used to control both the simulated and hardware ArDrone.

**Table 1. ROS topics in ‘ardrone_autonomy’ for control of quadcopter**

<table>
<thead>
<tr>
<th>ROS Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/ardrone/navdata/alt</td>
<td>Estimated altitude in millimeters</td>
</tr>
<tr>
<td>/cmd_vel/twist/linear/x</td>
<td>Control input for local x-axis translational</td>
</tr>
<tr>
<td>/cmd_vel/twist/linear/y</td>
<td>Controls movement along local y-axis</td>
</tr>
<tr>
<td>/cmd_vel/twist/linear/z</td>
<td>Controls movement along local z-axis</td>
</tr>
<tr>
<td>/cmd_vel/twist/angular/z</td>
<td>Controls movement along local z-axis</td>
</tr>
</tbody>
</table>

#### 3.3.2 ROS Package ‘ar_track_alvar’

Visual tags compatible with the ROS software package ‘ar_track_alvar’ are used in our system for identification of the mobile robot. Early testing determined that multiple resolutions are necessary for recognizing visual tags at varying distances from the tags. To provide multiple resolution tags, the authors decided to embed small tags within larger tags. An example of the original tags and the combined multi-resolution visual tag used for our system are depicted in Figure 8.

At greater distances (typically 0.5-1.5 meters) the smaller embedded tags appear to be visual noise to the ‘ar_track_alvar’ software. At closer distances (typically < 0.5 meters) the large tag is not recognized by ‘ar_track_alvar’, where the smaller are then identified and tracked. The following ROS topics are tag recognition inputs to the fuzzy controllers from the package ‘ar_track_alvar’.
3.3.3 ROS Package 'ace_ardrone_fuzzy'

The following fuzzy system state controller actions are performed by the ROS package 'ace_ardrone_fuzzy':
- If not directed to search for landing vehicle, perform mission
- If tag not found, search for tag
- If tag found, apply fuzzy alignment of quadcopter with mobile base
- If tag found, quadcopter aligned and landing directed, apply fuzzy landing controller

This package was created by the authors to maintain the code necessary to simulate and control the quadcopter operations. Implemented in the package are the fuzzy controllers for hovering and landing. Heading control is added into the landing control.

4. SIMULATION AND HARDWARE RESULTS

In this section we describe the developed controllers and results from simulations from design of the altitude and landing controllers. Simulation of hover and landing fuzzy controllers are also presented in this section. Hover and landing were tested separately using Gazebo as a simulation and visualization environment. Simulated altitude data from ‘tum_simulator’ and identified visual tag information from ‘ar_track_alvar’ were used for input to the hovering and descent controllers.
4.1 Hovering Control Simulation

4.1.1 The Developed Fuzzy Inference System

The altitude input variable $h_S$ is used as the ‘sonar_reading’ for input to the controller. The altitude controller is implemented in a total of 5 rules to control the quadcopter over the z-axis. Rules for the altitude controller are listed in Table 3.

<table>
<thead>
<tr>
<th>Input</th>
<th>Input MF</th>
<th>Output</th>
<th>Output MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>sonar_reading</td>
<td>too_low</td>
<td>cmd_vel_gaz</td>
<td>large_increase_velocity</td>
</tr>
<tr>
<td>sonar_reading</td>
<td>a_little_low</td>
<td>cmd_vel_gaz</td>
<td>small_increase_velocity</td>
</tr>
<tr>
<td>sonar_reading</td>
<td>on_target</td>
<td>cmd_vel_gaz</td>
<td>no_change</td>
</tr>
<tr>
<td>sonar_reading</td>
<td>a_little_high</td>
<td>cmd_vel_gaz</td>
<td>small_decrease_velocity</td>
</tr>
<tr>
<td>sonar_reading</td>
<td>too_high</td>
<td>cmd_vel_gaz</td>
<td>large_decrease_velocity</td>
</tr>
</tbody>
</table>

Figure 9 shows the fuzzy controller designed for altitude control during evaluation in FuzzyLite. Figure 9 also shows further details such as control and sensor limits.

4.1.2 Simulation Setup

In the hover control, visual tag information and sonar altitude information are used for controlling the hovering altitude of the quadcopter. Figure 10 shows the configuration of nodes in ROS for performing hovering control. In the figure, the green colored node is the hovering controller node in the ‘ace_ardrone_fuzzy’ package.

As inputs, the node takes a calculated pose relative to the center of the detected landing target and the target identity descriptor (ID) for the target. With the inputs, the node creates a control vector based on the target ID and calculated pose of the target. ROS Gazebo was used for simulating the image streams coming from the ArDrone. Scenes from the simulator were extracted.

Figure 9. Altitude controller fuzzy inference system in QtFuzzyLite
using camera emulating nodes built into the ‘tum_simulator’. With the image streams the ‘ar_track_alvar’ node is able to process the images and provide feedback to the ‘hover_control’ node in the ‘ace_ardrone_fuzzy’ package.

Figure 10. ROS software configuration for software simulation of hover control

4.1.3 Simulation Results
Figure 11 shows a series of images illustrating the hover control for the quadcopter when visual tag information is used for hovering over the mobile base. Figure 11 (a) shows the quadcopter hovering above the mobile base using sonar input. Figure 11 (c) shows the quadcopter hovering above the mobile base with visual tag information available. Observed in the simulation during the hovering operation: the quadcopter maintained a stable altitude with or without the visual target information.

4.2 Landing Control Simulation

4.2.1 The Developed Fuzzy Inference System
The landing controller is implemented in a total of 19 rules to control linear motion in all three axes and rotational motion along the z-axis. Each of the rules were based off of logical reasoning and the quadcopter model in Figure 7. Table 4 lists the rules for orientation about the model z-axis.
(a) Quadcopter hovering over mobile base

(b) Quadcopter front camera view  (c) Quadcopter bottom camera view

Figure 11. Images from Gazebo showing quadcopter hovering before descent

Table 4. Rotational motion along z-axis in robot model of Figure 7

<table>
<thead>
<tr>
<th>Input</th>
<th>Input MF</th>
<th>Output</th>
<th>Output MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>orientation_x</td>
<td>way_left</td>
<td>cmd_vel_z_rot</td>
<td>large_turn_right</td>
</tr>
<tr>
<td>orientation_x</td>
<td>a_little_left</td>
<td>cmd_vel_z_rot</td>
<td>small_turn_right</td>
</tr>
<tr>
<td>orientation_x</td>
<td>on_target</td>
<td>cmd_vel_z_rot</td>
<td>hold_direction</td>
</tr>
<tr>
<td>orientation_x</td>
<td>a_little_right</td>
<td>cmd_vel_z_rot</td>
<td>small_turn_left</td>
</tr>
<tr>
<td>orientation_x</td>
<td>way_right</td>
<td>cmd_vel_z_rot</td>
<td>large_turn_left</td>
</tr>
</tbody>
</table>

Table 5 lists the fuzzy rules for displacement along model z-axis.
Table 5. Translational motion along z-axis in robot model of Figure 7

<table>
<thead>
<tr>
<th>Input</th>
<th>Input MF</th>
<th>Output</th>
<th>Output MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>displacement_z</td>
<td>way_high</td>
<td>cmd_vel_z_linear</td>
<td>large_move_down</td>
</tr>
<tr>
<td>displacement_z</td>
<td>too_high</td>
<td>cmd_vel_z_linear</td>
<td>small_move_down</td>
</tr>
<tr>
<td>displacement_z</td>
<td>centered</td>
<td>cmd_vel_z_linear</td>
<td>do_nothing</td>
</tr>
<tr>
<td>displacement_z</td>
<td>too_low</td>
<td>cmd_vel_z_linear</td>
<td>small_move_up</td>
</tr>
<tr>
<td>displacement_z</td>
<td>way_high</td>
<td>cmd_vel_z_linear</td>
<td>large_move_down</td>
</tr>
</tbody>
</table>

Table 6 lists the fuzzy rules for displacement along model y-axis.

Table 6. Translational motion along y-axis in robot model of Figure 7

<table>
<thead>
<tr>
<th>Input</th>
<th>Input MF</th>
<th>Output</th>
<th>Output MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>displacement_x</td>
<td>way_left</td>
<td>cmd_vel_y_linear</td>
<td>large_move_right</td>
</tr>
<tr>
<td>displacement_x</td>
<td>too_left</td>
<td>cmd_vel_y_linear</td>
<td>small_move_right</td>
</tr>
<tr>
<td>displacement_x</td>
<td>centered</td>
<td>cmd_vel_y_linear</td>
<td>do_nothing</td>
</tr>
<tr>
<td>displacement_x</td>
<td>too_right</td>
<td>cmd_vel_y_linear</td>
<td>small_move_left</td>
</tr>
<tr>
<td>displacement_x</td>
<td>way_right</td>
<td>cmd_vel_y_linear</td>
<td>large_move_left</td>
</tr>
</tbody>
</table>

Table 7 lists the fuzzy rules for displacement along the model x-axis.

Table 7. Translational motion along x-axis in robot model of Figure 7

<table>
<thead>
<tr>
<th>Input</th>
<th>Input MF</th>
<th>Output</th>
<th>Output MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>displacement_y</td>
<td>way_low</td>
<td>cmd_vel_x_linear</td>
<td>large_move_backward</td>
</tr>
<tr>
<td>displacement_y</td>
<td>too_low</td>
<td>cmd_vel_x_linear</td>
<td>small_move_backward</td>
</tr>
<tr>
<td>displacement_y</td>
<td>centered</td>
<td>cmd_vel_x_linear</td>
<td>do_nothing</td>
</tr>
<tr>
<td>displacement_y</td>
<td>too_high</td>
<td>cmd_vel_x_linear</td>
<td>small_move_forward</td>
</tr>
<tr>
<td>displacement_y</td>
<td>way_high</td>
<td>cmd_vel_x_linear</td>
<td>large_move_forward</td>
</tr>
</tbody>
</table>

The fuzzy landing controller is shown below in Figure 12 during evaluation in FuzzyLite. Figure 12 also shows further details such as control and sensor limits. A discussion of the shapes and limits of the membership functions is provided in the simulation results section.
4.2.2 Simulation Setup

For descent of the quadcopter, visual tag information is used exclusively for controlling landing of the quadcopter. Figure 13 shows the configuration of nodes in ROS for performing landing control. In the figure, the green colored node is the landing controller node in the author’s ‘ace_ardrone_fuzzy’ package. As inputs, the node takes a calculated pose relative to the center of the detected landing target and the target identity descriptor (ID) for the target. With the inputs, the node creates a control vector based on the target ID and calculated pose of the target. ROS Gazebo was used for simulating the image streams coming from the ArDrone. Scenes from the simulator were extracted using camera emulating nodes built into the ‘tum_simulator’. With the image streams the ‘ar_track_alvar’ node is able to process the images and provide feedback to the ‘landing_control’ node in the ‘ace_ardrone_fuzzy’ package.
4.2.3 Simulation Results

Final values of control and sensor limits were decided upon by simulation and experimental results. Shapes of membership functions were designed based on the reaction of the quadcopter to various inputs. The output for orientation for example has Gaussian membership functions with centers far outside the limits imposed on the outputs of the fuzzy inference system. This is due to a desired slowed response to high errors in orientation. In simulation of the Figure 14 shows an image from Gazebo where the mobile base is not detected due to being at too high of an altitude to capture the visual tag clearly. Once the simulated quadcopter is at an acceptable altitude for detecting the large visual tag, it is highlighted a blue color using the ‘ar_track_alvar’ ROS software as shown in Figure 15. At this point, coordinates of the tag, relative to the center of the simulated camera lens, are available to the controller. Figure 16 shows the part of descent where the quadcopter simulated camera only has a partial view of the large tag and a full view of the smaller embedded tags. When the smaller tags are in view and identified they are used for identification and tracking of the mobile base for completion of the landing procedure.

4.3 Hardware Experimentation

In the hardware setup for the experiments, a laptop was used for both communications with and control of the ArDrone 2.0 quadcopter. The laptop represents the same computer that will be controlling the UGV in the system. ROS was used to control the quadcopter in the same manner as the simulation studies were performed.
4.3.1 Hardware Setup

In testing of the developed fuzzy logic controllers on the hardware, the configuration in Figure 17 was used. Controllers from 4.1.2 and 4.2.1 were used on the hardware for hovering and landing controls, respectively. Controllers were ran as programs via command line based on the test being run.
4.3.2 Hardware Results

In testing of the physical system, the controllers demonstrated acceptable performance given that the environment was not too complex. Environmental complexity became an issue when the ArDrone exhibited unwanted behavior in its sensed altitude reading being recalibrated during hovering control. The problem was isolated to be a built-in ArDrone manufacturer firmware issue. The author’s suspicion is that when the quadcopter passes close by an object, the sonar transducer used for altitude measurement is recalibrated by the onboard firmware. The result is a negative altitude reported by the ArDrone. This the controller sees as flying below the commanded altitude, therefore forcing the quadcopter to rise. When the quadcopter raises too much, the tendency is for it to hit a ceiling indoors or flyaway outdoors. As a workaround to the problem on the client side, the authors modified in the ‘tum_ardrone’ package to compensate for the ArDrone’s incorrect internal calibration scheme. With the modification in place, the quadcopter can track its altitude correctly. This theory is founded by the fact that the close fly-by sonar issue is easily repeatable and with the fix, the hovering control provides acceptable results in most situations. Flying over tall objects is another problem that the authors will need address later by use of a dynamic flight height based on visual input data rather than sonar-based altitude.

5. CONCLUSIONS

In this work fuzzy controllers were designed for controlling altitude and hovering in place above a target. Simulation based demonstration of the controllers was provided using the combination of open source tools FuzzyLite, ‘ar_track_alvar’, and ROS Gazebo with TUM ArDrone Simulator. Simulation results showed acceptable performance from the altitude and landing controllers. An expandable hardware and simulation environment was described for distributed systems.
On control, we plan to integrate the controllers together using a fuzzy version of a state machine to control state transitions of the system. On the topic of fuzzy control, supervised learning process and data mining are planned to obtain the optimal inputs, rules and membership functions based on prior flight data history. The goal of the learning process is to obtain fuzzy inference systems optimal for flying in various environments and adapt based on available sensor data. Another path the authors will explore is the use of cloud based processing nodes for enhancing the performance of the quadcopter and identification of targets from images with low visible information content.

On visual identification of the mobile base by the quadcopter we plan to further enhance tag detection at greater distances and altitudes using the combination of multiple image processing techniques instead of relying on a single algorithm. The target marker image processor, `ar_track_alvar`, provided a great start to an algorithm for identifying locations of the mobile base. However, the processor failed at greater distances where noise and camera focus became an issue. Incorporation of additional image processing methods based on approximate matching of unfocused images is likely the direction the authors will take to better solve the vision problem.

On the open source community, work presented in this paper will enhance the ArDrone interface software. We plan to provide the solution to the well documented corrupted quadcopter sonar altitude readings and the steps to recreate the problem. We intend to upgrade the core ArDrone interface software provided by Engel et al. to be compatible with the latest version of ROS to provide support for modeling more disturbance types such as wind, noisy camera inputs, noisy accelerometer input.

REFERENCES


ABOUT THE AUTHORS

P. J. Benavidez received the B.Sc. degree in electrical engineering from The University of Texas at San Antonio, USA in 2007, and the M.Sc. degree in electrical engineering the University of Texas at San Antonio (UTSA), USA in 2010. Currently he is a Ph.D. student in the Autonomous Control Engineering (ACE) Laboratory in the electrical and computer engineering program at UTSA. His research interests include robotics, control systems, wireless networks, soft computing, and RGB-D cameras. He has been involved in the organization of the 2013 World Conference on Soft Computing (WCSC) and student conferences at UTSA.
J. I. Lambert will receive the B.Sc. degree in electrical engineering from The University of Texas at San Antonio, USA in 2013. In 2014, he will pursue his Ph.D. at the University of Texas at San Antonio. His research interests include automation and robotics for human interaction, control systems, and power systems.

A. S. Jaimes Betancourt obtained his Licentiate degree in Electronic Engineering at the Universidad Mayor de San Andres (UMSA) in 1999, specializing in Control Systems. From 2000 to 2002 he worked at the Instituto Nacional de Medicina Nuclear (INAMEN) as Electronic Engineer. From 2003 to 2006, he worked at ABB Bolivia as Project Engineer implementing applications of variable speed drives. In 2006, he decided to continue his studies in the United States, and obtained his M.S. degree in Electrical Engineering at the University of Texas at San Antonio (UTSA) in 2009. During this period, he worked as Graduate Research Assistant at the Autonomous Control Engineering Laboratory at UTSA. Currently, he is pursuing his Ph.D. degree on Electrical Engineering at UTSA. His research interests are focused on the field of robotics and control systems, particularly the development of AI algorithms that will enable robots to interact with their environment, and the effective coordination of its tasks.

M. M. Jamshidi (Fellow IEEE, Fellow ASME, A. Fellow-AIAA, Fellow AAAS, A. Fellow TWAS, Fellow NYAS, Member HAE) received BS in EE, Oregon State University, Corvallis, OR, USA in June 1967, the MS and Ph.D. degrees in EE from the University of Illinois at Urbana-Champaign, IL, USA in June 1969 and February 1971, respectively. He holds honorary doctorate degrees from University of Waterloo, Canada, 2004 and Technical University of Crete, Greece, 2004 and Odlar Yourdu University, Baku, Azerbaijan, 1999. Currently, he is the Lutcher Brown Endowed Distinguished Chaired Professor at the University of Texas, San Antonio, TX, USA. He has been an advisor to NASA (including 1st MARS Mission), USAF Research Laboratory, USDOE and EC/EU (Brussels). He has advised over 60 MS and nearly 50 PhD students. He has over 680 technical publications including 68 books (12 text books), research volumes, and edited volumes in English and a few foreign languages. He is the Founding Editor or co-founding editor or Editor-in-Chief of 5 journals including IEEE Control Systems Magazine and the IEEE Systems Journal. He is an Honorary Professor at Deakin University (Australia), Birmingham University (UK), Obuda University (Hungary), and at three Chinese Universities (Nanjing and Xi’an, China). He has received numerous honors and awards, including IEEE Centennial Medal, IEEE Millennium Awards, and the IEEE’s Norbert Weiner Research Achievement Award, among others. He is a member of the University of the Texas
System Chancellor’s Council since 2011. He is currently involved in research on system of systems engineering with emphasis on cloud computing, robotics, UAVs, biological and sustainable energy systems, including smart grids.