

USING DISTANCE SENSORS TO PERFORM COLLISION AVOIDANCE MANEUVERES ON UAV APPLICATIONS

A. Raimundo ^{a b}, D. Peres ^{a b}, N. Santos ^{a b}, P. Sebastião ^{a b}, N. Souto ^{a b}

^a ISCTE-Instituto Universitário de Lisboa, DCTI, Avenida das Forças Armadas, 1649-026 Lisboa

^b Instituto de Telecomunicações, Av. Rovisco Pais 1, 1049-001 Lisboa

aslo@iscte.pt, drbps@iscte.pt, nmass@iscte.pt; pedro.sebastiao@iscte.pt; nuno.souto@iscte.pt;

KEY WORDS: UAV, object collision avoidance, distance sensing, LiDAR, 3D environment, drone, 3D vehicle.

ABSTRACT:

The Unmanned Aerial Vehicles (UAV) and its applications are growing for both civilian and military purposes. The operability of an UAV proved that some tasks and operations can be done easily and at a good cost-efficiency ratio. Nowadays, an UAV can perform autonomous missions. It is very useful to certain UAV applications, such as meteorology, vigilance systems, agriculture, environment mapping and search and rescue operations. One of the biggest problems that an UAV faces is the possibility of collision with other objects in the flight area. To avoid this, an algorithm was developed and implemented in order to prevent UAV collision with other objects. "Sense and Avoid" algorithm was developed as a system for UAVs to avoid objects in collision course. This algorithm uses a Light Detection and Ranging (LiDAR), to detect objects facing the UAV in mid-flights. This light sensor is connected to an on-board hardware, Pixhawk's flight controller, which interfaces its communications with another hardware: Raspberry Pi. Communications between Ground Control Station and UAV are made via Wi-Fi or cellular third or fourth generation (3G/4G). Some tests were made in order to evaluate the "Sense and Avoid" algorithm's overall performance. These tests were done in two different environments: A 3D simulated environment and a real outdoor environment. Both modes worked successfully on a simulated 3D environment, and "Brake" mode on a real outdoor, proving its concepts.

1. INTRODUCTION

Humans are always searching for a better solution to resolve day-to-day problems. There is always something that needs to be more practical, efficient, affordable or even easier to interact with human beings. Due to that fact, there was a necessity of creating something that resolve those problems in a way that can guarantee same operational functions as human operators or even new ways to do the same thing with less effort. To accomplish that, Unmanned Aerial Vehicles (UAVs) were developed to preform actions that were difficult for human beings to execute. They are different from Manned Aerial Vehicles. UAVs are remotely controlled and Manned Aerial Vehicles are locally controlled. UAVs are characterized by their size, weight and maneuver. They are very small and light-weighted when compared to real manned aerial vehicles, such as commercial travel planes or even civil aircrafts. Their initial purpose was to serve the military for executing scout and surveillance missions, and even to preform aerial attacks in Gulf War and more recent, Iraq (Cocaud, 2007). Nowadays, UAVs are used to perform several tasks that require fast and efficient methods, such as analyzing crash sites, firefighting, search and rescue processes and so on (Scherer, 2015). There are various types of UAVs, each one with its limitations, functionalities and purposes. In order to control those UAVs, it's necessary to have a ground control station (GCS), where communication is established between UAV and GCS, via wireless connection. There are several ways to communicate wirelessly, which can be via Radio, Satellite or Mobile Networks (Murilhas, 2015; Saraiva, 2015). They can perform autonomous actions, given by manned remote control, such as executing missions that are global positioning system (GPS) - guided through waypoints marked in a map. In a mission, user controller orders UAV to flight through pre-defined GPS coordinates. However, like every other manned control processes, error could occur. Consequently, there is a necessity

to develop intelligent systems that can mitigate, or even avoid completely human-caused problems. An UAV, in order to make decisions by itself, has to be prepared to take emergency actions. Data indicators can trigger an emergency state status such as fast altitude drop, communication failure, low battery and risk of collision (for object collision avoidance).

2. UNMANNED AERIAL VEHICLE

UAVs are aircrafts controlled by a manned ground control station, which can send and receive commands between GCS and UAVs with the absence of a present pilot on an UAV. The UAVs have a wide range of applications. Companies have developed hardware components in order to make an UAV fly with maximum stability and performance. Software projects were made to communicate with hardware components, in order to give a user-friendly interface, to read data and easily communicate with an UAV.

There are some different structural types for UAVs, each one has its way of taking off, landing, aerodynamics and usability:

Fixed-wing: These UAVs have some benefits that doesn't exist in any other UAV types. They benefit from their designed two-winged aerodynamics. Their wings help the wind flow through the UAV. Because of that, they can achieve greater air speeds and glide without losing too much height. This leads to a much less power consumption and that allows much more autonomy for long range flights. Fixed-wing UAVs lift and land horizontally, and they can't hover in a certain position like other UAVs. Due to increased range, these structural type of UAV is most used by the military (Scherer, 2015).

Multi-rotor: Unlike fixed-wing UAVs, these multi-rotor UAVs are capable of hovering, standing still in the air. They do not have wings. Their behavior is similar to a helicopter, but with more propellers. The most common multi-rotor UAVs are: Quadcopters, Hexacopters and Octacopters. They lift off and land

vertically. Compared with fixed-wing UAVs, they fly at lower speeds but have more action control. For being a multi-rotor UAVs, they consume much more power and have a shorter autonomy (Scherer, 2015).

2.1 Companion Hardware

In order to control hardware components, read data and send commands, an UAV needs to be equipped with an on-flight board control. That component is described as a flight controller. To provide autonomous tasks, extended wireless communication features and on-board situation control, an UAV needs additional hardware component: a mini-computer. Both flight controller and mini-computer are connected internally through serial connection and they exchange data between them. These components are categorized as Companion Hardware.

2.2 Raspberry Pi

Raspberry Pi is a credit-card sized micro-computer, developed in 2012 at the University of Cambridge's Computer Laboratory. Runs Linux as operating system with the option to have a graphical user interface, and provide some useful inputs, such as USB ports, HDMI port, RJ45 port, audio port and GPIO (General Purpose Input / Output) connectors. These GPIO connectors allows developers to attach other hardware components, such as sensors, motors, leds and displays.

Raspberry Pi can be used for a variety of things, such as a personal computer, a web server, or a device controller where the setup steps are simple. The operating system is downloaded to an SD card, then user configures some minor variables, such as operating system language, keyboard layout and drive setup management. After that, additional packages, such as compilers and libraries, can be added (Brock, Bruce, & Cameron, 2013). It's a powerful computer, and when compared to desktop increased-sized solutions, it's relatively cheap. This companion hardware it's really useful because of its size, price and included functionalities (Saraiva, 2015).

2.2.1 Flight controller: Some UAV flight controllers use hardware configurations based on existing projects and 3DR Pixhawk is one of them. ArduPilot is a full-featured open source project, licensed by GNU General Public License version 3 (GPLv3) that supports "from conventional airplanes, multirotors, and helicopters, to boats and even submarines" (Ardupilot Dev Team, n.d.). ArduPilot supports several types of UAV's such as fixed wings, multi-rotors and Unmanned Ground Vehicles (UGVs). Each type is configured in a single firmware file, which is installed on the Pixhawk main processor. This flight control board has some embedded sensors to provide useful data to Ground Control Station (GCS): a gyroscope, an accelerometer, a barometer and a magnetometer (Saraiva, 2015). It has the option to add a Global Position System (GPS) sensor to provide location data, such as latitude, longitude, altitude and speed. To perform guided missions, a GPS module is required in order to specify waypoints. This autopilot is fully programmable and can have First-Person View (FPV) camera gimbal support and control, Radio Controlled (RC) channel inputs and other sensors. The built-in hardware failsafe uses a separate circuit to transfer control from the RC system to the autopilot and back again. This prevents crashes by safely land on the ground (Bin & Justice, 2009). In this work, a 3DR Pixhawk flight controller was used with ArduPilot available firmware. In order to transfer data to Raspberry Pi, Pixhawk uses an USB connection.

2.3 Control and communication

In order to control an UAV, it's necessary to have intermediary components between UAV itself and piloting user. Such components are essential for piloting because they determine the behavior of an UAV.

2.3.1 Radio Controller: Piloting an UAVs can be quite a challenge for novice pilots, because an UAV can fly in various 3D space directions: left and right, to the front and back, up and down. The simplest and fastest way to start controlling an UAV is using a radio controller. A radio receiver on the UAV is needed to receive radio frequency (RF) data. This data is transmitted by a radio transmitter, controlled by the user, to control UAV's attitude.

2.3.2 Ground Control Station: A Ground Control Station (GCS) is an essential UAV monitoring and controlling tool. It provides user relevant flight data, read by flight controller sensors. Can be used to track down UAV location by reading GPS data, perform autonomous tasks, calibrate sensors, pre-flight tests, and so on. It's a required tool for ground operation tasks, used before, during and after UAV flights. There are various Ground Control Stations software available for Windows and Linux operating system environments, such as Mission Planner, APM Planner and QGroundControl. In order to send and receive data between UAVs and GCS, both must have telemetry equipment, such as Wi-Fi or Radio Frequency transmitters and receivers. This data is encrypted and sent by a common message type via MAVLink protocol.

2.3.3 MAVLink message protocol: MAVLink is a header-only message protocol that uses group of messages to transmit data between the UAV and GCS. It is designed to be reliable, fast and safe against transmission errors. It was first released in 2009 by Lorenz Meier under the LGPL license (Murilhas, 2015; Scherer, 2015). Each message is byte-encrypted with sensor related content, which can be interpreted by Ground Station Control or by Raspberry Pi, which will serve as a message intermediary between Pixhawk and GCS. Commands like take off, raise or decrease altitude (throttle increase or decrease, respectively) are sent by GCS and interpreted by Pixhawk to perform desired action.

2.4 Distance sensing

Distance sensing can be useful for many UAV applications, such as ground altitude measurement, UAV terrain shape following, object detection and environment mapping. The use of distance sensors on an UAV extends its features for user determined purposes. There are various types of distance sensors, and each one has its features, restrictions and limitations.

2.4.1 Light Detection and Ranging (LiDAR): A Light Detection and Ranging (LiDAR) sensor is a measurement technology, which is based on a precise measurement of the time delay between the transmission of a pulsed optical laser light signal and its reception (Duh, n.d.). This allows UAV to produce environment mapping, by analyzing objects that are close or far from UAV current position. This sensor produces two types of light signals: Reference signal fed from the transmitter and a received signal reflected from the target. The time delay between these two signals is calculated through a signal processing method, known as correlation. These correlation process accurately calculates the time delay, which is translated into distance based on the speed-of-light constant (“LiDAR - Technology and System Hardware Overview,” n.d.). Each LiDAR for each application works at different electromagnetic wavelengths. Meteorology LiDARs usually work on ultraviolet (250 nm) and infrared (1500 nm to 2000 nm) wavelengths. For terrestrial mapping, at near-infrared wavelengths (Duh, n.d.). For pulsed LiDAR lasers, to obtain object distances and range resolutions, the equations are (1) and (2) are used:

$$R = c \frac{t}{2} \quad (1)$$

where R means the distance, c the speed of light value (~299,792,458 m/s) and t the time delay between transmitting and receiving the light pulse (in ns), and

$$\Delta R = c \frac{\Delta t}{2} \quad (2)$$

where ΔR is the range resolution, c the speed of light and Δt resolution of time measurement (in ns) (Duh, n.d.). Range resolution is not the same thing as accuracy. Accuracy is the difference of measured distances between values read by LiDAR and real and true distance values. For example, if LiDAR measures a distance of 1.01 m from an object or surface and the real distance value is 1 m, the LiDAR reading accuracy is 1 cm (Value read - Real value = LiDAR accuracy). Range resolution is, for example, on environment mapping, a LiDAR sensor can generate a graph called “point cloud”. And each point means a read distance. Range resolution is the difference between points on a point cloud graph and for a terrain elevation model, “it’s the accuracy of the elevation”, or by other words, how far is the model measurements are from the real terrain elevation (Schmid, Hadley, & Wijekoon, 2011).

3. SENSE AND AVOID ALGORITHM

The Sense and Avoid algorithm was developed to achieve this article’s main goal: an autonomous avoidance system for UAVs in object collision course to use in search and rescue operations. This algorithm was designed to work fully on outdoor environments only. It was developed in Java programming language, for any operating system that supports Java VM and any flight controller that supports MAVLink messages and ArduPilot ArduCopter (for multirotors) firmware. In order to generate MAVLink messages in a way that MAVProxy could interpret as commands, there was the need of a MAVLink interpreter that uses Java as same programming language as “Sense and Avoid” algorithm. A Java library named MAVLink Java was used and it was developed by GitHub’s username Ghelle (Ghelle, n.d.). It is an essential tool in order to read messages generated from Pixhawk and to send MAVLink commands programmatically. This algorithm was tested on 3D simulated environments and on real outdoor environments. “Sense and Avoid” algorithm relies only in LiDAR external

sensor readings to calculate objects in collision course and then, making decisions. These decisions are based on how an UAV is being controlled. For each different way of controlling an UAV, the algorithm behaves according to the situation. This algorithm needs to be always running in order to work. So, in this case and since it’s installed on Raspberry Pi’s OS, the “Sense and Avoid” algorithm always runs at Raspberry Pi’s boot, by running a startup script. In this chapter, every “object” word denomination can not only be referred as an object itself. An “object” can be a person, animal, wall, tree or other “instrument” that has an opaque surface. The flow chart depicted in figure 1 describes how “Sense and Avoid” algorithm works. This chart was developed by Bizagi Process Modeler, a free-to-use software for Windows environments developed by Bizagi company, that allow users to create process models for business purposes. In this case, this software was used for modeling a cycling process, which accurately describes the internal functionality of “Sense and Avoid” algorithm.

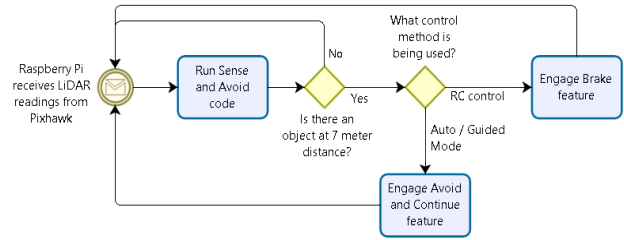


Figure 1. “Sense and Avoid” internal functions.

“Sense and Avoid” algorithm starts by receiving LiDAR readings from Pixhawk’s flight controller through MAVProxy. Then proceeds by running a cycling (threaded) code that runs every 0.25s (this value was chosen as an example initial value and worked on experimental purposes). On every cycle, “Sense and Avoid” algorithm checks LiDAR distance read (that automatically outputs object distance in meters). If LiDAR doesn’t detect an object that has a distance inferior or equal to 7 m, it means no action is required and UAV doesn’t change its behavior. This ends cycle process and starts a new one. But, if LiDAR detects an object that has a distance inferior or equal to 7 m, the algorithm rapidly detects which controlling mode active at the moment: RC control method or autonomous methods such as “Auto” or “Guided” flight modes. If an UAV is being RC controlled, then the algorithm activates “Brake” feature. Or, if the UAV is flying on “Auto” or “Guided” flight modes, the algorithm will activate “Avoid and Continue” feature. At the end of “Brake” or “Avoid and Continue” features, the process cycle ends. The previous controlling method is resumed and new cycle begins, as it can be seen on the flow chart. The 7 m LiDAR distance is the maximum value that the UAV can be distanced of an object on collision course before activating one of the algorithm’s features. Inferior distance values were considered as well in order to predict instant object appearing which could cause UAV collision with that object. The 7 value is not a threshold value, but was given as an example. It was initially chosen when performing experimental tests on a 3D simulator and there were no complications on achieving tests goals. The “Sense and Avoid” algorithm has two modes: “Brake” and “Avoid and Continue”. Each mode is enabled accordingly to UAV’s current controlling method. The “Brake” mode is for RC controlling only and “Avoid and Continue” mode is for autonomous tasks, such as “Guided” flight mode and “Auto” flight mode.

3.1.1 Brake mode: The “Brake” mode is only engaged if LiDAR “sees” an object at a distance, *e.g.* 7 m or inferior, and if the UAV is currently being controlled by an RC controller. “Brake” mode is an essential feature for users that want an UAV with an object collision avoidance system. This feature prevents RC controlled UAV from object collision, activating the “Brake” flight mode available on Pixhawk’s Copter firmware. This mode will only work if UAV has GPS position enabled, because Brake flight mode depends on GPS. This mode switches UAV’s current flight mode to “Brake” flight mode. This flight mode will stop the UAV mid-air and holding the same altitude the UAV had upon “Brake” flight mode switching. This will cause UAV to drift until hold position. Higher speeds mean higher brake times. After activation, any other form of controlling the UAV is disabled, and this means RC attitude control. Then, UAV proceeds to fly backwards until LiDAR measures object distance greater than 7 m. When that reading occurs, previous flight mode is activated and RC control is resumed. GPS is required to save current position. This position refers to the UAV’s actual position when Brake flight mode was activated. In case of drifting, this mode will readjust UAV’s position to the position when Brake flight mode was activated. The flow chart represented by figure 2 demonstrates how “Brake” mode changes UAV behaviors.

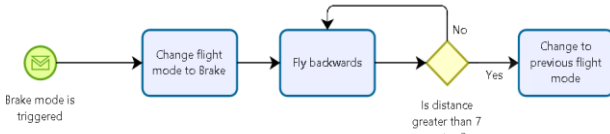


Figure 2. “Brake” mode flow chart.

3.1.2 Avoid and Continue mode: This mode is useful for fully autonomous tasks, such as missions. Implementing this feature not only guarantee object collision avoidance, but also resumes previous established goals. This mode was developed for search and rescue missions that require object collision maneuvers. The flow chart represented by figure 3 describes how “Avoid and Continue” mode behaves.

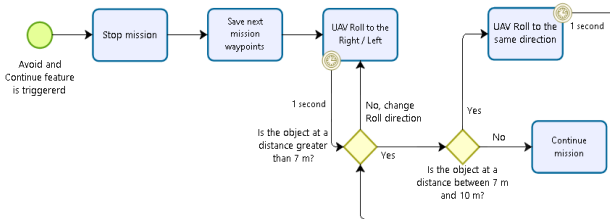


Figure 3. “Avoid and Continue” mode flow chart.

When “Avoid and Continue” mode is triggered, it comes from an autonomous flight mode: “Guided” or “Auto” flight mode. Those modes use pre-defined GPS positions. These positions are also called waypoints, and it sets a fly course for the UAV. This fly course is called mission. Missions are performed autonomously, flying orderly from the first waypoint created until the last one. A mission can have multiple waypoints. But, if in between these waypoints there’s an obstacle in collision course, the algorithm will be triggered. Firstly, the algorithm will stop the mission, in order to prevent collision. Then, will orderly save the list of non-traveled waypoints, *i.e.*, the next mission waypoints. This step is very important, because the UAV needs to keep record of its mission main goals. After saving waypoints, the UAV proceeds to Roll to the right or left. This UAV direction change (left or right) is chosen randomly, because the LiDAR sensor has a

narrow beam (SF11/C beam divergence is 0.2°) and it can only sense the object at one small point in UAV’s front. For that reason, it is impossible to know the width of the object in order to choose direction that the UAV should take for a fast avoidance (shortest path to an open area). But, a low divergence beam can be useful, because it makes object distance measurement more accurate. If the object on sight is at a distance greater than 10 m, then the algorithm will resume mission by loading previous saved waypoints. If the distance is between 7 m and 10 m, the UAV will repeat the same Roll attitude in the same direction. This prevents the cases when the Roll attitude duration of 1 s is not enough to continue mission safely. These 3 m distance value is also called “threshold”. But, if the distance is not greater than 7 m, there is the need of changing actual Roll direction. For example, if the UAV’s LiDAR is reading an object at 7 m, then it proceeds to Roll right, and that distance increases even more, the UAV could be facing the object with a certain angle that will rapidly increase the distance between UAV and that object. To avoid that, the UAV will invert the previous Roll rotation until distance is higher than 7 m so it can resume mission as described.

4. SIMULATION AND EXPERIMENTAL RESULTS

4.1 Environments

“Sense and Avoid” algorithm experimental tests were made on two different environments. Firstly, and for security reasons, there was a need to find a solution that was secure, easy to deploy and close to real UAV behaviors. A 3D simulated environment was the perfect solution for test “Sense and Avoid” algorithm. After a series of trial-and-error tests, resulting on many successful results on “Sense and Algorithm” main features made on a 3D environment, some tests were made on the custom UAV built on an outdoor environment. The following subsections describe the two test environments and what tests were made on each one.

4.1.1 3D simulation: In order to test the “Sense and Avoid” algorithm on a 3D simulated environment, it was necessary to find a solution that represented real outdoor environments and UAV “close-to-real” behaviors. Gazebo is an Apache 2.0 licensed open-source 3D environment simulator that is capable of simulating multi-robot behaviors on indoor or outdoor environments. It also features sensor readings (for object awareness) and “physically plausible interactions” between rigid objects (“Gazebo,” n.d.; Gomes Carreira, 2013). This simulator operates on Robot Operating System (ROS), which is BSD licensed open source library toolbox to develop robot-based applications. The ROS toolbox allows installing third-party plugins in order to add extra features useful to simulate all UAV’s flying requirements, such as control and communication methods (Gomes Carreira, 2013; Open Source Robotics Foundation, 2014). The Gazebo simulator makes it possible to test “Sense and Avoid” algorithm because it provides realistic scenarios, with 3D model inclusions and real physics simulation. The ROS/Gazebo implementation will work as an external simulator for Pixhawk’s ArduCopter firmware. Its main features are:

- Simulate 3D UAV’s physical stability and user command responses;
- Read sensor data from the virtual-created UAV;
- Communicate between user inputs and UAV model;
- GPS information and SoNAR / LiDAR distance readings.

Gazebo features the 3D environment, including the UAV model and an example indoor scenario. ROS contains a list of plugins that can be used in order to develop the UAV 3D solution. The ArduPilot SITL Gazebo plugin is responsible for interfacing the

virtual-created UAV (ArduPilot SITL simulation) with Gazebo simulation. The MAVROS plugin package, or Micro Air Vehicle ROS, provides communication features between ArduPilot firmwares (ArduCopter in this case). It uses the MAVLink communication protocol to exchange data. Additionally, MAVROS provides a bridge for MAVProxy GCS, which will be useful to “Sense and Avoid” algorithm, since it relies mainly on MAVProxy GCS to establish a communication and control gateway. The Rotors Simulation plugin provides Gazebo simulator the UAV kinematics and sensor readings (IMU and LiDAR / SoNAR) (Furrer, Burri, Achtelik, & Siegwart, 2016; Robotics, n.d.). With these plugins, it is possible to test “Sense and Avoid” algorithm by experiencing real UAV responses to algorithm decisions. The LiDAR sensor simulation on Gazebo can read real distances from rigid 3D objects on the presented scenario, which gives an approximate real-feel experience from outdoor UAV flights and LiDAR readings. Since this test main goal is to use “Sense and Avoid” algorithm on a Search and Rescue approach, there was the need to create a scenario that includes the UAV facing an object in collision course. In early test stages, a 3D cube was used as an object-to-avoid scenario model. All Gazebo and ROS installation procedures and some 3D models were made by Erle-Robotics company, which presents a well-made guide that was essential in order to start our 3D UAV model in an outdoor environment.

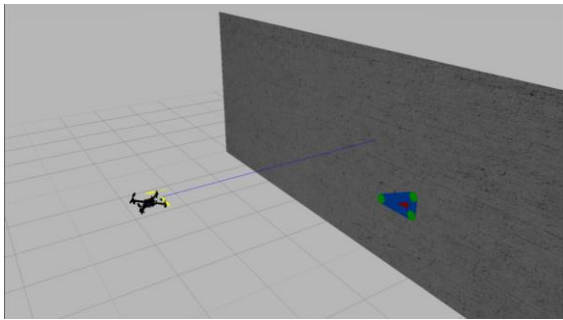


Figure 4. Quadcopter UAV model in a simple environment.

In early algorithm development stages, a simpler environment was made to test algorithm’s efficiency. The figure 4 presents the UAV 3D model, with a LiDAR sensor installed and is facing a concrete wall model. All models are available to use upon Gazebo installation. The UAV model named “Erle-Copter” was created by Erle-Robotics company. In order to add the laser “effect” on the UAV 3D model, some additional configurations were made. This simple environment served as the main field test of the “Brake” mode of “Sense and Avoid” algorithm. By receiving LiDAR readings of wall distance, it was possible to perform RC-based inputs, and therefore, acting like a real user sending RC controller commands. The RC-based inputs were sent via MAVProxy console. The simulation on this test scenario is a user-deliberated UAV control in order to collide with the concrete wall. The “Brake” mode will prevent it, and when compared with real cases, it relates to non-intentional collisions, classified as human errors. The figure 5 shows a more realistic environment. This environment was the base environment to test “Avoid and Continue” mode.



Figure 5. 3D Outdoor environment simulation.

This environment was created to test “Avoid and Continue” mode of the “Sense and Avoid” algorithm. In order to simulate a Search and Rescue scenario, this simulation represents an UAV flight using Pixhawk’s Auto flight mode (waypoint navigation), flying from point A to point B in order to perform a certain SAR mission. Since it doesn’t have a clear path, the UAV must avoid the object in collision course and proceed to reach B point successfully. The point A is at UAV’s current position and point B is behind the brown house model.

4.1.2 Real outdoor: In order to test “Sense and Avoid” algorithm, using the custom built UAV, some experimental tests were made on an outdoor environment. Firstly, some Guided and Auto missions were made to ensure UAV’s stability and to avoid errant behaviors when testing the algorithm. This tests were performed in an open-air area and obstacle-free environment. After ensuring UAV’s stability on autonomous missions, a obstacle was needed to test “Brake” mode. Unfortunately, only “Brake” mode was tested on a real environment, because tests location was on a urban environment, and UAV failures could result on damaging the UAV or the object itself in case of crashing.

4.2 Results and performance evaluation

The subsections below show the best-case scenarios when testing one of each “Sense and Avoid” modes in both 3D and real outdoor environments.

4.2.1 Brake mode: In “Brake” mode This model will be used as object in collision course with the UAV. The parameters are totally configurable, such as LiDAR minimum and maximum distance measure, field of view, update rate and actual sensor distance measure. Replicating as much as possible a real case scenario of LiDAR readings, the parameters were adjusted according to the SF11/C LiDAR altimeter specifications, used in the custom-built UAV. The start point distance from the wall is approximately 12.77 m. The value was chosen as an example. The UAV is on LOITER flight mode and is hovering at 2 m altitude from ground plane. In order to reproduce an UAV movement towards the wall, a RC input simulation was needed. This means simulating a Pitch attitude to the front. For this test, only RC channel 3 was used to change UAV’s Pitch attitude. By running the command “RC 3 1480”, will perform a smooth UAV Pitch to the front. It’s smooth because RC channel’s PWM neutral value is 1500 μ s, and the difference was only 20 μ s. Maximum and minimum PWM values are 1000 μ s and 2000 μ s, respectively. These values can vary according to RC transmitter used (Ardupilot Dev Team, 2014). That’s why an RC calibration is always needed to establish limits on RC maximum and minimum values. The “Brake” mode is configured to change to “Brake” flight mode when UAV is being controlled by RC inputs and LiDAR reads a distance inferior than 7 m. Since it was a very slow and smooth Pitch movement, and wind is not a problem on a 3D environment, the UAV stayed at distance of approximately 6.95 m from the concrete wall. “Brake” mode is prepared to “null” the drift effect movement by flying on a symmetric way. In this case, and to raise the distance again to a value higher than 7 m, a symmetric Pitch attitude is done. Since UAV is on “Brake” flight mode, simulated RC inputs are disabled. To overcome this, the algorithm used a MAVLink command called “RC_CHANNEL_OVERRIDE” to programmatically gain control of the UAV (ETHZ, 2014). In order to obtain the Pitch value to nullify the drift behavior, the equation (3) was used:

$$S = RC_n + [RC_n - RC_i] \quad (3)$$

where S means the symmetric PWM value, RC_n corresponds to the nominal value of the RC channel (which is a constant value of 1500), and RC_i represents the RC PWM value when “Brake” mode was activated. In this case, $S = 1520$.

The test scenario was made by a human UAV pilot on the environment presented on figure 6 and the LIDAR distance readings can be observed on figure 7.



Figure 6. UAV facing the “Impact zone”.

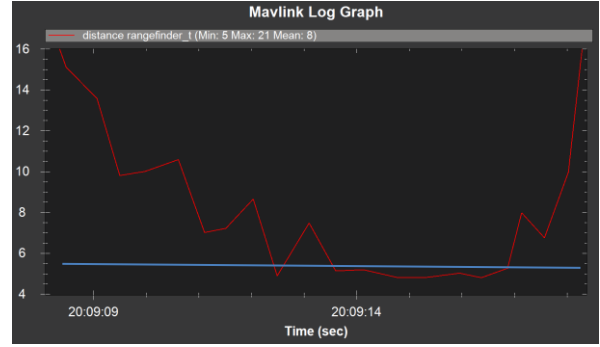


Figure 7. LiDAR distance readings. Retrieved from Mission Planner software logs.

The test was made at a distance of approximately 16 m from the “Impact zone”, on a day with a wind speed of 19 km/h.

4.2.2 Avoid and Continue Mode: The “Avoid and Continue” mode was only tested in a 3D simulated environment. Tests on real outdoor environment were not performed for security reasons only. Since it was an urban environment, a slight non-expected behavior could cause damage on the objects around test area, including human beings, or on the UAV itself. This test was one of the successful tests made with “Avoid and Continue” mode. If the Roll attitude lowers even more the distance from the object, then the “Avoid and Continue” mode will change the Roll direction, by performing the symmetric Roll attitude of the previous one. For example, if the right Roll attitude was chosen by the algorithm and the distance from object lowered, if the RC Roll attitude had a PWM value of 1400, the symmetric Roll attitude will be 1600 μ s. This follows the principle explained in the “Brake” mode, but in that case, it was for reducing UAV drifting effects. After the Roll attitude, the UAV is not facing the brown house anymore and it can proceed with the mission safely. The “Avoid and Continue” mode, which previously saved the waypoint behind the house model, will change to Guided flight mode again and proceeds to fly to the mission waypoint. The “Avoid and Continue” was developed for city environment purposes, where buildings are tall enough to discard the “flying above the object” solution. This was the reason why the Roll attitude was considered in “Avoid and Continue” mode. This mode performance was like expected but it requires some improvements. If the UAV encounters a concave wall, it will be trapped forever and occasionally it will cause the UAV to crash. The Roll attitude behaviors can cause the collision with side objects, since LiDAR sensor is pointed to the front and has no visibility to the other sides. Also on Roll attitude behaviors, the path chosen can be the farthest one to an object-free area being caused by the limitations of the LiDAR sensor used. These problems can be solved adding a system that analyzes the environment in every 3D direction.

5. CONCLUSIONS

The main goal of this work was to develop a system that was capable of taking control of an UAV autonomously if an object in collision course was detected. Therefore, “Sense and Avoid” algorithm, considering “Brake” and “Avoid and Continue” modes, was developed to avoid UAV collisions and to ensure autonomous object collision avoidance maneuvers. The algorithm was tested in two different test environments: a 3D simulation with realistic physics and collision simulations, and a real outdoor environment. A custom-built UAV was made in

order to test algorithm on a real aircraft. All the work and research done, and after test results and performance analysis, some conclusions were made:

- The LiDAR sensor proved to be a reliable distance measurement sensor. The accuracy of SF11/C LiDAR was outstanding and it was the best solution to detect with precision objects in collision course;
- The “Brake” mode of “Sense and Avoid” algorithm worked flawlessly after a long number of trial-and-error tests. The successful results on a 3D simulated environment proved that same behavior should be expected on a real outdoor experiment. The UAV maneuvers were as expected on real outdoor environment.
- The “Avoid and Continue” mode proved to be useful feature for every UAV on autonomous missions. Unfortunately, its results showed some flaws caused by SF11/C LiDAR’s capabilities and limitations. The main reason of this is because the algorithm was developed and adjusted according to the LiDAR capabilities. “Avoid and Continue” mode potential should be expanded by improving current system (for example, a 360° LiDAR environment sweep) or implementing another distance sensing system.

ACKNOWLEDGEMENTS

The work developed and presented in this paper was produced in Instituto de Telecomunicações Laboratories - ISCTE-IUL. The equipment used was supported by IT and ISCTE-IUL.

REFERENCES

- Ardupilot Dev Team. (n.d.). About ArduPilot. Retrieved from <http://ardupilot.org/about>
- Ardupilot Dev Team. (2014). Radio Control Calibration in Mission Planner. Retrieved from <http://ardupilot.org/copter/docs/common-radio-control-calibration.html>
- Bin, H., & Justice, A. (2009). The Design of an Unmanned Aerial Vehicle Based on the ArduPilot. *Indian Journal of Science and Technology*, 2(4), 12–15. Retrieved from <http://www.indjst.org/index.php/indjst/article/view/29423>
- Brock, J. D., Bruce, R. F., & Cameron, M. E. (2013). Changing the world with a Raspberry Pi. *Journal of Computing Sciences in Colleges*, 29(2), 151–153. Retrieved from <http://dl.acm.org/citation.cfm?id=2535418.2535441>
- Cocaud, C. (2007). *Autonomous tasks allocation and path generation of UAV's*. ProQuest Dissertations and Theses. University of Ottawa, Ottawa, Ontario, Canada. Retrieved from <https://www.ruor.uottawa.ca/bitstream/10393/27508/1/MR34061.PDF>
- Duh, G. (n.d.). Light Detection and Ranging (LiDAR) Types of aerial sensors. Retrieved October 21, 2016, from <http://code.google.com/creative/radiohead/>
- ETHZ. (2014). MAVLINK Common Message Set. Retrieved from <https://pixhawk.ethz.ch/mavlink/>
- Furrer, F., Burri, M., Achtelik, M., & Siegwart, R. (2016). RotorS---A Modular Gazebo MAV Simulator Framework. In A. Koubaa (Ed.), *Robot Operating System (ROS): The Complete Reference (Volume 1)* (pp. 595–625). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-26054-9_23
- Gazebo. (n.d.). Retrieved from <http://gazebosim.org/>
- Ghelle. (n.d.). MAVLink Java generator and library. Retrieved from <https://github.com/ghelle/MAVLinkJava>
- Gomes Carreira, T. (2013). Quadcopter Automatic Landing on a Docking Station.
- LIDAR - Technology and System Hardware Overview. (n.d.). Retrieved January 13, 2016, from http://lidarlite.com/docs/v2/technology_and_system_overview/
- Murilhas, L. (2015). New system to drive future ground vehicles supported by heterogeneous wireless networks.
- Open Source Robotics Foundation. (2014). Documentation - ROS Wiki. Retrieved from <http://wiki.ros.org/>
- Robotics, E. (n.d.). Simulation | Erle Robotics Docs. Retrieved from <http://docs.erlerobotics.com/simulation>
- Saraiva, T. (2015). *Reliable Air-to-Ground Communication for Low Altitude Unnamed Aerial Vehicles*. ISCTE-IUL.
- Scherer, M. (2015). *Development and Implementation of an Unmanned Aerial Vehicle with Stereoscopic Cameras Controlled via a Virtual Reality Head-Mounted Display*. Frankfurt University of Applied Sciences. Retrieved from http://baunvorlesungen.appspot.com/Abschlussarbeiten/Marcus_Scherer_Masterthesis_2015.pdf
- Schmid, K. A., Hadley, B. C., & Wijekoon, N. (2011). Vertical Accuracy and Use of Topographic LIDAR Data in Coastal Marshes. *Journal of Coastal Research*, 27(6), 116–132. <https://doi.org/10.2112/JCOASTRES-D-10-00188.1>