Introduction to Unmanned Aerial Systems Technology – 5 ECTS

Using the IRIS$^+$ quadcopter for lifeguarding

Mathias Nielsen
Steffen Ivan Nielsen
Ignacio Torroba
Mikael Westermann
Keerthikan Ratnarajah
Ander Cruz

Faculty of Engineering - MMMI

Project period: 9th of February 2015 to 29th of May 2015.

Exam dates: 17th to 18th of June 2015.

Supervisor: Assistant Professor Kjeld Jensen.

May 29, 2015

Abstract

This paper describes a solution to the problem of coast and sea patrolling using a quadrotor UAS flying autonomously with video feed and processing capabilities. The aim is to enable the user to specify areas of interest for the drone to patrol, on a graphical user interface (GUI). Already implemented solutions have proven successful and the demand for drone applications is increasing.
This work focuses on the creation of a guide to setting up a 3DR Iris+ multirotor for autonomous flight along custom paths. Focus is also laid on the path generation for search and rescue missions at sea using search patterns and vision-based software utilizing a camera system (GoPro and Tarot Gimbal). The drone should be able to autonomously follow search pattern paths generated by user-specifiable parameters, and be able to detect humans in water. The implementation in this project is not online, the image postprocessing takes place on a computer. It is assumed that the software running offline on the computer can be extrapolated to an online implementation on the drone. The results of this project show the steps necessary to make the drone fly autonomously and to connect to it through a ROS package named Mavros. It is shown that automated and parametrized search pattern based path generation is possible through a developed C++ library for waypoint manipulation and Matlab scripts for search pattern generation. Also shown are the results of an implementation of a human detection algorithm which is able to detect humans in water and to distinguish them from other objects or living beings. It is concluded that this work facilitates the UAS-based coast and patrol system at seas.
Further work will focus on enabling online image processing, a fast data connection to a ground station and a customized application with a GUI utilizing Mavros.

Contents

Introduction

This report was written as part of the course SDU-UAS-Intro 2015 and presents the work of group 1. The aim of the developed system is to enable an existing drone model to easily be configured and programmed for performing offshore vision aided patrolling. The used equipment was a 3DR Iris+ drone with a Tarot T-2D Brushless gimbal mounted Go Pro HERO 3 camera. This enabled the drone to get visual data from user-specified areas. During its watch, if any human being is detected on this area, the drone should store the acquired images and send them to the ground control station on the fly, to be analysed by the user and activate the rescue protocols.
The source code and materials used are freely available at
`https://github.com/madsherlock/SDU-UAS-15-Group1-Report/blob/master/Final.zip?raw=true`
and at `https://goo.gl/jVWK9b`.

Motivation

Patrolling and ensuring safety at remote and humanly inaccessible areas such as the sea has been a difficult task for a long time, requiring expensive equipments and many man hours, see [18]. With the recent development of readily accessible drones, which are easy to use and fairly simple to operate, a revolution within the area of life guarding seems within reach.
Utilizing the drone's capability of operating and monitoring humanly inaccessible areas to control coast areas could not only result in a more safe coast environment, but also a decreased cost in regards to life saving. The use of drones in regards to emergencies and rescue missions has already showed promising results [21], [6], [14], [22], [22], but mostly using humanly controlled flights.
The vision is to explore the possibilities of making an easy to use patrolling system and making a drone fly autonomously within a specified area. Using computer vision to detect changes in the current scene or movement it should report back with images from the scene. The images can then be processed by a professional. As this project is considered a research project the material presented will be limited to the areas of autonomous flight in good flying conditions and computer vision.
Using this approach, a minimum amount of man hours is needed, while the possibilities of autonomous drone flight will be utilized.
To ensure efficiency, the group was divided into two subgroups: One for the Autonomous flight and one for the Human detection through computer vision. The group work distribution and contributions to the report by each group member can be seen in A.

Abbreviations

UAS Unmanned Aerial System

ROS Robot Operating System

UART Universal asynchronous receiver/transmitter

RTL Return to launch

RC Remote Control

FCU Flight Controller Unit

FOV Field of view

ROI Region of interest

fps Frames per second

GUI Graphical User Interface

1. Autonomous flight

This section describes the drone selection, gives an overview of the system setup and how autonomous flight was achieved on the Iris+. The means of communication between ground control station and drone are discussed in relation to the problem of developing an easy to use graphical user interface. Finally, this section describes how the autonomous flight can be applied to rescue missions at sea using search patterns generated from intuitive user-specifiable parameters, three example scripts developed in Matlab and a C++ library developed for the generation and manipulation of flight paths.

1.1 Drone selection

To fulfill the vision of using an easy to use and readily available drone platform, the drone selection was quickly narrowed down to what was at hand. As the primary focus of the project was to explore autonomous flight, other concerns such as resistance to rough weather could be neglected.
During the course, two drones matching the giving requirements were introduced: The DJI Phantom 2 [7] and the 3DR Iris+ [20].
The obvious choice between the 2 drones was the Iris+, as it used the open source Pixhawk FCU [17]. Having an open source FCU allowed for easy interfacing and more flexibility with many tasks concerning the drone. The Iris+ drone can be seen on figure 1.1.

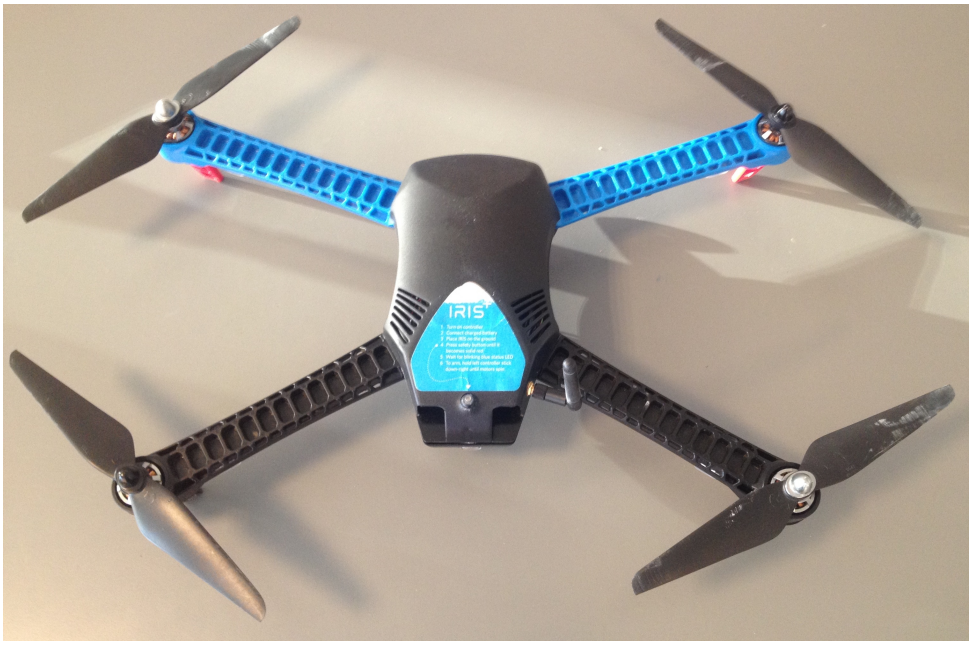


Figure 1.1: The 3DR Iris+ drone.

The main features and components of the Iris+ drone used for this project are:

- 16-22 minutes flight time.
- Payload capacity of 400g.
- Manual remote control.
- Telemetry link.
- Autonomous waypoint navigation.
- Feature rich ground control software.
- Open source FCU.

1.2 Setup

Radio Controller

The first part of the setup was to bind the radio controller to the radio receiver. The receiver used was the $2.4Ghz$ FrSky D4R-II Reciever [11] and can be seen on figure 1.2.
To bind the controller, the top cover of the drone has to be taken off to access the radio receiver. The next steps are as follows:

1. Turn on the radio transmitter while holding down the button on the back of the radio transmitter.
2. Once the remote is beeping let go of the button on the back.
3. Power up the drone while holding down the F/S button on the radio receiver.
4. Release the F/S button once the radio receiver LED is flashing red and green.
5. Power off drone.
6. Power off radio transmitter.
7. If the binding is done correctly the radio receiver LED should be solid green when connected to the radio transmitter and the LED should blink red when data is transferred.

For further assistance the first part of the following video is suggested: `https://www.youtube.com/watch?v=5ygCbdR4FCE`.

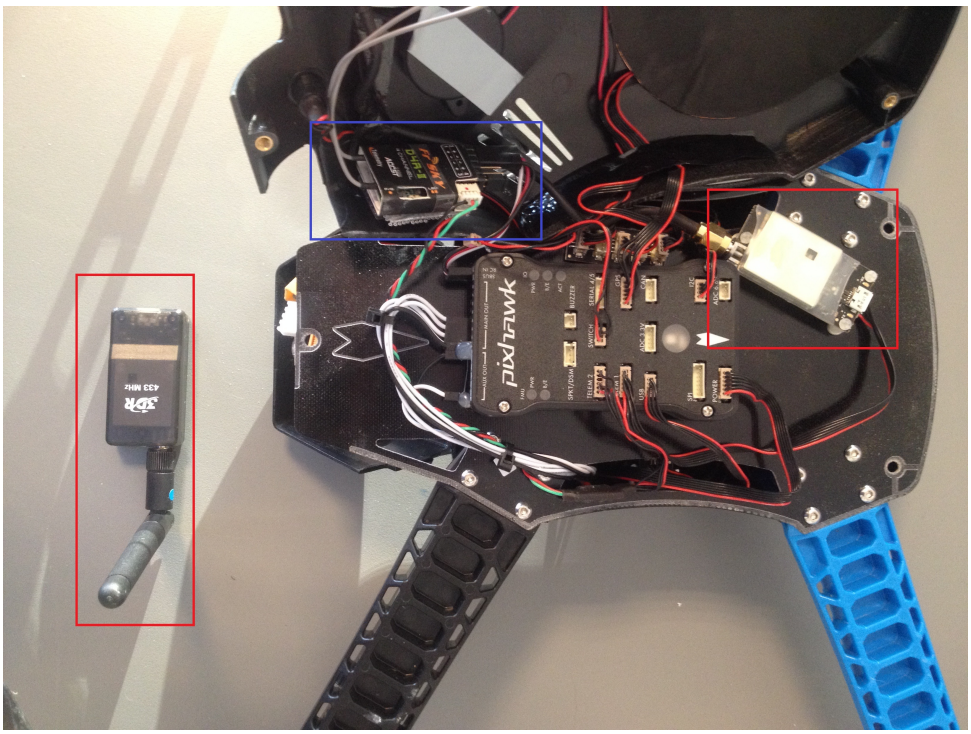


Figure 1.2: The insides of the Iris+ drone. The red rectangles marks the telemetry and the blue rectangle marks the radio receiver.

Telemetry

The telemetry used is the 3DR Radio v2 [4] operating at $433Mhz$. The purpose of the telemetry is to transmit and receive data from and to the ground station via the MAVLink protocol [19].
To connect the ground telemetry to the telemetry on the drone the 3DRRadioConfig software was used on a windows machine. The two telemetries were connected to the PC by USB one at a time and set to the desired configuration. The configuration used can be seen on figure 1.3, where the most significant parameters are the baud rate and the net id. When setting the configurations on the drone telemetry, it is important that it is not connected to the pixhawk FCU, as this would otherwise cause connection issues.
When connected, the green LED should be solid and the red LED should blink whenever data is transferred. The drone telemetry can now safely be connected to the Pixhawk FCU again.

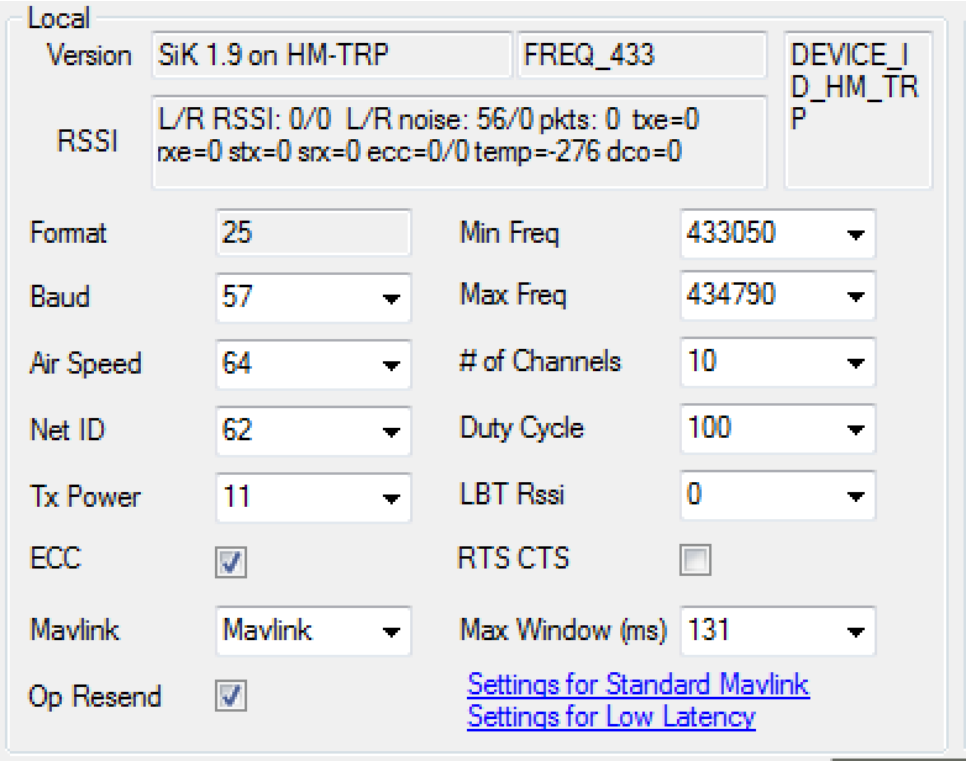


Figure 1.3: Telemetry configuration used.

Gimbal

Since pictures had to be taken during flight and as these pictures had to be good enough to use computer vision on, it was decided to use the Tarot T-2D Brushless Gimbal Kit [5]. The gimbal stabilizes the camera, but adds extra weight to the drone and uses extra power from the battery to power the two motors. Both of these factors leads to a significant decrease in flight time.
The integration and usage of the gimbal will be further explained in the vision chapter.

APM Planner 2.0

The firmware for the Pixhawk was a choice between the original px4 and Arducopter [9]. As 3DR recommend flashing the Arducopter firmware, and as it contains several features not available in the Px4 firmware, Arducopter was chosen.
To communicate with the Arducopter firmware for calibration purposes and as a general interface, APM Planner 2.0 was used [8]. When this software was used on OSX Yosemite it crashed several times during calibration and firmware flashing. The best solution found for this problem on this platform was a simple reboot. On Ubuntu 15.04, the software ran without problems. While running the flight data mode where it communicated with the drone by telemetry it was very stable and responsive.
Note that there is a small bug in APM Planner 2.0 with regards to displaying waypoints as text: In the waypoint editor, the latitude and longitude of the waypoints have switched places. When adding waypoints by clicking on the map, this is no problem, but when trying to make sense of the waypoints, it can be very confusing. However, the software exports the waypoints correctly, and uploads them correctly to the drone. This can be verified in the "Onboard waypoints" view in APM Planner 2.0, where the latitudes and longitudes of the uploaded waypoints are displayed correctly. Refer to the MAVLink protocol in [19] and have your current latitude and longitude coordinates ready when in

doubt. The APM Planner 2.0 software was primarily used for calibration and testing of waypoint planning, as waypoint navigation is a built-in feature of the arducopter firmware.
All of the setup for the Pixhawk FCU was done under the initial setup menu in APM planner 2.0.

Calibration

After flashing the drone with new firmware, and before every flight with the drone at a new location, the drone was recalibrated using APM Planner 2.0. A proper calibration was a key element in securing accurate and safe flight with the drone. It is of great importance that the rotors are removed doing calibrations as unpredictable behaviour sometimes occur.
The 4 mandatory calibrations that had to be done were:

- Frame type selection
- Compass calibration
- Accelerometer calibration
- Radio calibration

To calibrate for the physical appearance of the drone the proper frame type had to be selected. This is done under the "initial setup" menu where "Frame type" should be selected. The proper frame type is selected and then downloaded. This sets all the parameters for the specific drone type. For this project the Iris+ with Tarot gimbal was used.

The compass calibration involves rotating the drone around all axes to fill in the calibration matrix. The matrix and all the calculations are handled by APM Planner 2.0. First, the pixhawk is selected in the menu and the live calibration is chosen. This gives the user a minute to rotate the drone around all axes. It is advised to power the drone with battery during this process, as any wires might get tangled and make the calibration more complicated. Sometimes several program crashes were encountered during the calibration process on OSX Yosemite.

The calibration of the accelerometer was pretty straight forward. To calibrate the the accelerometer, the software asks you to put the drone in 6 distinct positions and then press enter.

The last mandatory calibration needed is the radio calibration. When the radio is turned on and properly connected to the drone the radio calibration can be initiated. All of the control sticks and switches are moved to their outer positions. The process can be followed on the GUI to see if everything is working properly. After moving everything to its outer position, the throttle stick is moved down while the other stick is in its center position.

All of these calibrations are of great importance in order to fly the Iris+ drone properly. Several problems were encountered due to bad calibration and not checking everything without rotors. If anything seems off during the initial flight or testing without rotors it is suggested to either do a complete recalibration or re-flashing with the newest firmware. A guide for the mandatory calibration can be found at `http://copter.ardupilot.com/wiki/initial-setup/configuring-hardware/`. However, it should be noted that some elements of this guide are a bit outdated.

Flight modes

As the Arducopter firmware has several different flight modes and the preset flight modes do not match the desired flight modes for this project, they had to be changed. The desired flight modes are the manual modes stabilize and altitude, and the autonomous mode auto mode. It is important to set these flight modes, so that one can quickly overrule auto mode during flight, if something is not going according to plan.
As different radio controllers are available, one should always check that the flight modes are mapped to the used controller as desired, without rotors.
Two additional fly modes were used to easily land the drone and to return the drone to the position it was armed at. These modes were land and return to land (RTL). The land flight mode performs a landing at the current spot, disabling the throttle switch but enabling control of the position stick.

Fail safe

To control the behavior in case of a malfunction on the radio controller side or the drone running low on battery, the fail safe options have to be set. The software will give a warning to dismount the rotors during this setup. This is important as the motors are controllable in this mode.
To do this, the the throttle fail safe value is set approximately 30 units below the lowest value of the throttle stick. This ensures that if radio connection is lost, it will do the desired action. For our case we chose return to launch.
The other fail safe option is that when the drone's battery level goes below a certain voltage it should perform the desired action. Return to launch was also chosen as this action.
The fail safe menu also provides sliders representing the values coming from the radio controller and the motor output. This feature was used to check if the flight modes gave the expected output to the motors.

1.3 Flight

To do a proper flight with the drone all of the prearm actions have to be done. This involves keeping the drone in a steady position and having a GPS fix. The drone is armed by first holding down the front red button on the drone. It is now ready to be armed, but not yet armed. The arming procedure varies with the controller, but the arming sequence on the used controller was to keep the throttle in the bottom right corner. To disarm it, the throttle had to be in the bottom left corner. The prearm error messages can be looked up on the following page: `http://copter.ardupilot.com/wiki/flying-arducopter/prearm_safety_check/`. It should be noted that arming the drone indoors can be a tedious task due to the need of a GPS fix.

1.3.1 Preflight & safety

During the extended testing phase several safety procedures were established. This led to a preflight test check list.
The preflight checks in lab consisted of:

1. Check firmware is the most recent stable version.

2. Do calibrations.
3. Set desired flight modes and fail safe settings.
4. Check output in different flight modes with motors on (no rotors).
5. Ensure proper connection to APM so data can be monitored in APM Planner 2.0.
6. Mount rotors and perform test flight in drone cage.

The preflight checks at the airport consisted of:

1. Do calibrations.
2. Ensure proper connection to APM so data can be tracked in APM Planner 2.0.
3. Contact tower to get clearance to fly and get wind speed.
4. Alert bystanders of flight.
5. Find spot for controlled crash i case of emergency.

1.3.2 Autonomous flight

To test out the autonomous flight feature on the Arducopter frimware, APM planner 2.0 was used to plot a simple route at the airport. A route in APM Planner 2.0 was generated using waypoints. These waypoints were placed at a relative altitude to the drones home position which is the arming position, also known as "launch". The drone would fly a straight path to the waypoints and wait there for a designated time before continuing. It is possible to control orientation of the drones as well as proximity to the waypoint before it is accepted.
The APM Planner 2.0 software offers a collection of other actions such as loiter, set region of interest etc. These were not used of this project. Instead, generation of waypoint list files was investigated, as described in section 1.5.

1.3.3 Tests

The path shown on figure 1.4 was the one flown to test the autonomous flight. All of the waypoints were 8 meters above ground. The accepted distance to the waypoint was set to a 2 meter radius and the orientation of the drone was always in the direction of flight. The ground footage recorded during the flight can be viewed at `http://youtu.be/qYRAMeGNFuA`.

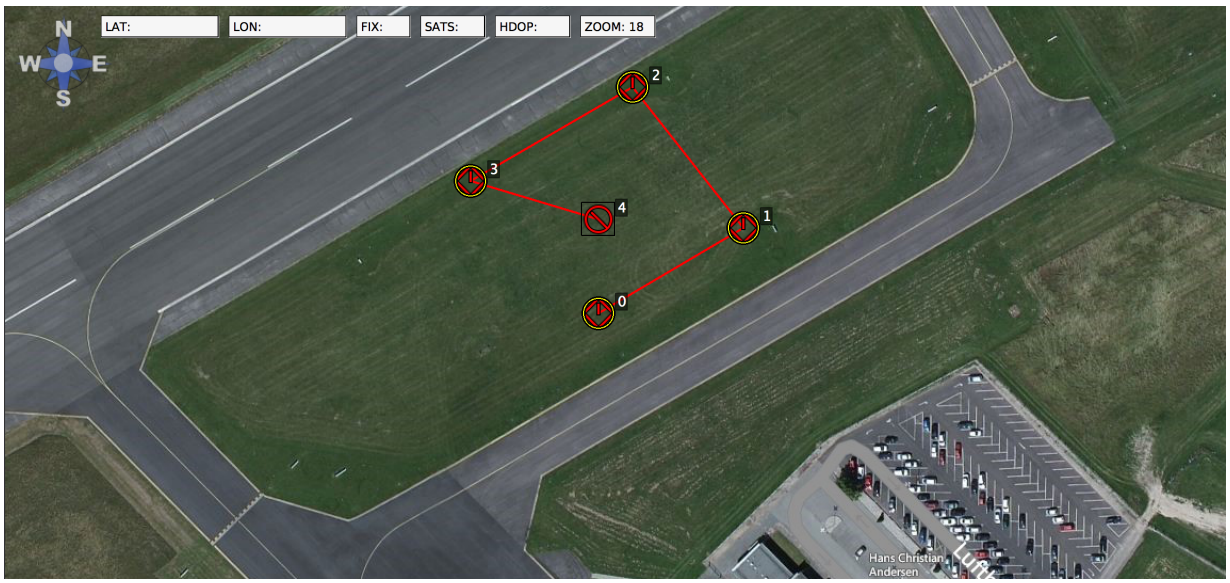


Figure 1.4: Test path at HCA Airport. See `http://youtu.be/qYRAMeGNFuA` for ground footage.

On the day of the test, moderate winds were present, which resulted in a slower flight against the wind but the overall performance of the drone was good as it followed the designated route, indicating its usefulness in patrolling applications.
Several initial tests had been done without success due to either safety issues concerning being able to switch to manual mode and general flight planning issues.
One test with an altitude of 30 meters above ground was tested, but had to be switched to manual mode and landed as there was too much wind.

1.4 Mavros

Since the aim of the project was to explore autonomous flight with an easy to use interface, a way to communicate directly with the drone without APM Planner 2.0 was needed. To do this the Mavros [10] ROS package was used. By running a main Mavros node it was possible to connect to the drone's telemetry and control it.
As this node was in a beta state when used, there was not a lot of documentation, but by some trial and error it was possible to connect to the drone, arm it and upload a waypoint list to the Pixhawk FCU. Data from the drone was also published on several topics which could be used for further control. ROS was used on ubuntu 14.04.
The approach was as follows:

1. Modify the apm2.launch file's url to reflect the telemetry port and baud rate, e.g. arg name="fcu_url" default="/dev/ttyUSB0:57600".

2. roslaunch the modified node and verify it is connected.

3. If it failed to get permission try "sudo chmod 666 /dev/ttyUSB0", setting permissions on the serial port.

4. To enable topics, set streamrate from pixhawk by calling "rosservice call /mavros/set_stream_rate 010 1".

5. To arm, use "rosservice call /mavros/cmd/arming "value: true" ".

6. To load waypoint list use "rosrun mavros mavwp load /path/waypoints.txt" where waypoint.txt contains correctly MAVLink formatted waypoints.

The tests mentioned in the previous section were with waypoints uploaded via mavros.
Getting the basics of uploading way points and arming the drone to work with implemented rosrun functions was simple once the syntax was mastered. Implementing this into a joint program with a path generation algorithm would be trivial. This could result in an easy to use GUI application which takes a simple coordinate and an altitude as input, and then sending the drone to search the nearby area.

1.5 Mavros waypoint file generation

Mavros uses a specific MAVLink waypoint file format for uploading waypoints. This format follows the MAVLink protocol, which can also be used to send other commands than waypoints. The format is the same as the one used by the APM Planner 2.0 application, which is able to visualize the waypoints in a satellite image. Thus, an easy way to learn the basics of the waypoint file format is to study waypoint files exported from APM Planner 2.0. An application named waypointgen1 was developed for conversion between coordinate paths generated in Google Earth's .kml file format and MAVLink protocol formatted waypoint files. This application is based on a C++ library also developed, contained in a file named waypoint.h, for simple manipulation of waypoints. The library provides functionality necessary to convert .kml files and .csv files to waypoint files. The second application developed in C++ is named csv2wp, and converts .csv files containing latitudes, longitudes (in degrees) and altitudes to MAVLink waypoint text files. Both C++ applications and the library were written using Qt Creator and are cross-platform compatible.
Google Earth's .kml file format is simple to read if the file contains only one path or polygon. It follows XML layout and under the <coordinates> header, all the coordinates are listed. The coordinates are listed in the following format: longitude latitude altitude. This makes the coordinate extraction process simple: Navigate the .kml file to the <coordinates> header and tokenize the coordinate string until reaching the end of the section (indicated by </coordinates>).

1.6 Generating search patterns

Several search pattern generation scripts were developed in Matlab for use with the C++ waypoint library. The Matlab scripts work with vectors in a cartesian coordinate system, assuming that both horizontal axes use the same scale. This means a UTM coordinate representation is necessary. For this, the scripts utilize the utm2lonlat script, Copyright (c) 2013, Erwin N.
The Matlab scripts and C++ library described here are not one-click routines that make the drone fly. Instead, the process is split into smaller parts, separating the path planning somewhat from the vision of an easy to use GUI. However, the process outlined here can be integrated into a GUI at a later point.

1.6.1 Zig-Zag

The Matlab script named zigzag generates a "Zig Zag" flight pattern. This simple pattern is a simple example of an overseas search pattern, as it widens as it progresses forwards. This could be applicable to rescue missions where the location where the person went missing is known. From this location, the person would have followed the stream. The stream direction is not an exact straight line, and the geographical line of possible locations the person could be in expands as the stream progresses. Hence the widening search pattern.
The script takes four arguments: Two coordinates, a density parameter and a search angle. The first coordinate is the flight starting coordinate. The second coordinate is the flight ending coordinate. The search angle describes how the search zone from starting coordinate to ending coordinate expands in a conic manner. For example, a search angle of 0 would mean the flight path is straight from starting coordinate to ending coordinate. The density parameter indicates the spacing between the back and forth zig zag flights. A small density parameter will place the flights close to each other so that the flight path will be very long and go back and forth a lot of times before reaching the ending coordinate. At least two factors could influence the choice of density parameter: The stream current and the camera field of view. An example of how the search path setup looks like in Google Earth is shown in figure 1.5. The procedure to generate a zigzag search path starting with Google Earth is as

Figure 1.5: Search path setup example in Google Earth.

follows:

- Generate a two-point path in Google Earth (or anywhere else) and export it as a .kml file.
- Convert the .kml file to MAVLink waypoint text format using the waypointgen1 application.
- Run the zigzag Matlab script, giving the name of the newly generated text file as input, along with the search angle, density parameter and the output .csv filename.
- Convert the generated .csv file to MAVLink waypoint text format using the csv2wp application.

After the UTM conversion, two bounding vectors are created at ± half the input angle with respect to the vector from starting coordinate to ending coordinate. The zig-zag pattern is then created within the bounding vectors, from the starting coordinate to the ending coordinate. It was chosen to end the zig-zag pattern before getting further away than the ending coordinate, so that the ending coordinate defines the maximum distance.
Examples of outputs of the zigzag script are shown in figure 1.6. The input path is the one shown in figure 1.5. The parameters for the zigzag script output of figure 1.6a are a 10 degree angle and a 10 m spacing betweem back-and-forth flights, while the parameters for figure 1.6b are a 30 degree angle and a 20 m density.

(a) 10 degree angle and 10 m density.

(b) 30 degree angle and 20 m density.

Figure 1.6: Zig-Zag pattern examples. The pattern is generated from the path of figure 1.5 using the zigzag script.

1.6.2 Spiral

Another search pattern generation Matlab script, named spiral, takes two input coordinates and a density parameter like the zigzag script, and generates an Archimedian spiral of waypoints from the starting point, extending outwards, with the input spacing, until the radius is equal to the distance to the ending point. The user also has the option to reverse the order of generated waypoints, so that the flight path is towards the center of the spiral. This search pattern, or more advanced versions of it, could be applied to searches where the last known location of the missing subject is known, but the current location is not restricted to any pattern (except, perhaps, a circle).
Figure 1.7 shows an example of a spiral generated from the example path of figure 1.5. The spacing between turns is 50 m.

1.6.3 Sector

The final search pattern generation script developed in Matlab is named sector. It is an implementation of the Sector Search Pattern, see [13]. The script takes two points as input. The first point is the starting point (the center of the search). The second point indicates the radius of the sector search and the drift direction (if any). As stated in [13], this search pattern is applicable when the approximate location of the target is known.
Figure 1.8 shows an example of the sector search pattern generated from the example two points of figure 1.5.

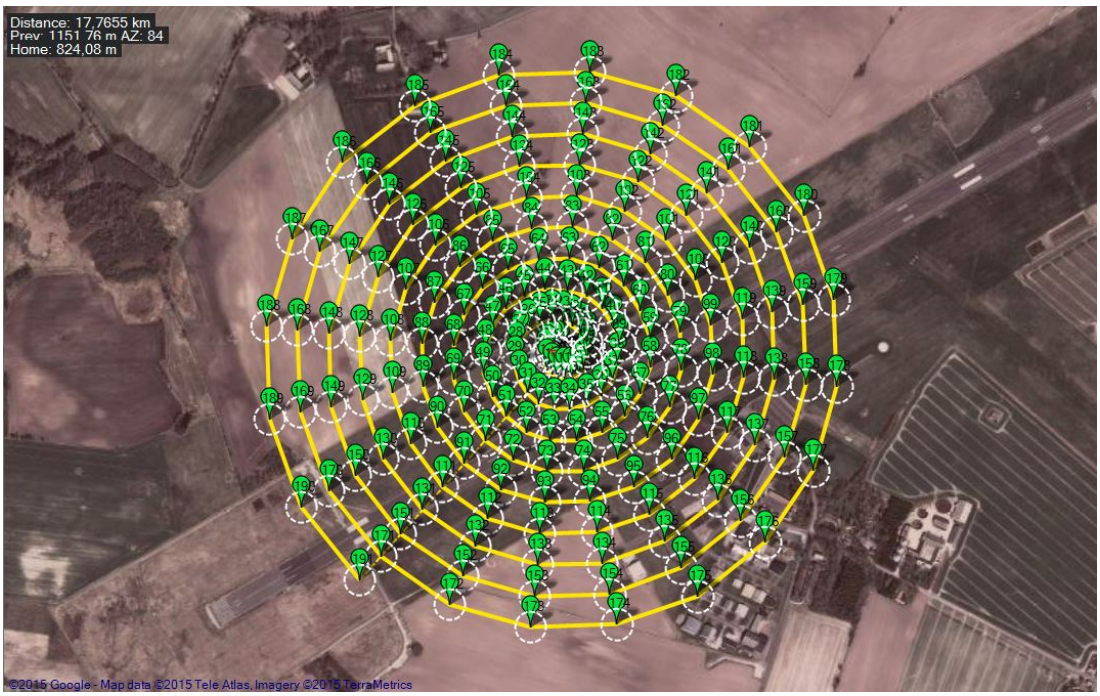


Figure 1.7: Spiral pattern example generated from the path of figure 1.5 with a 50 m spacing between turns.

1.7 Summary

It has been shown that it is possible to achieve satisfactory, stable, autonomous flight using paths based on overseas search patterns generated in an automizable process which can be integrated into a GUI. Two-sided communication between ground control station and drone has been achieved through a GUI in APM Planner 2.0 and through a command line interface using the Mavros ROS package. This facilitates the future development of the desired easy to use patrolling system.

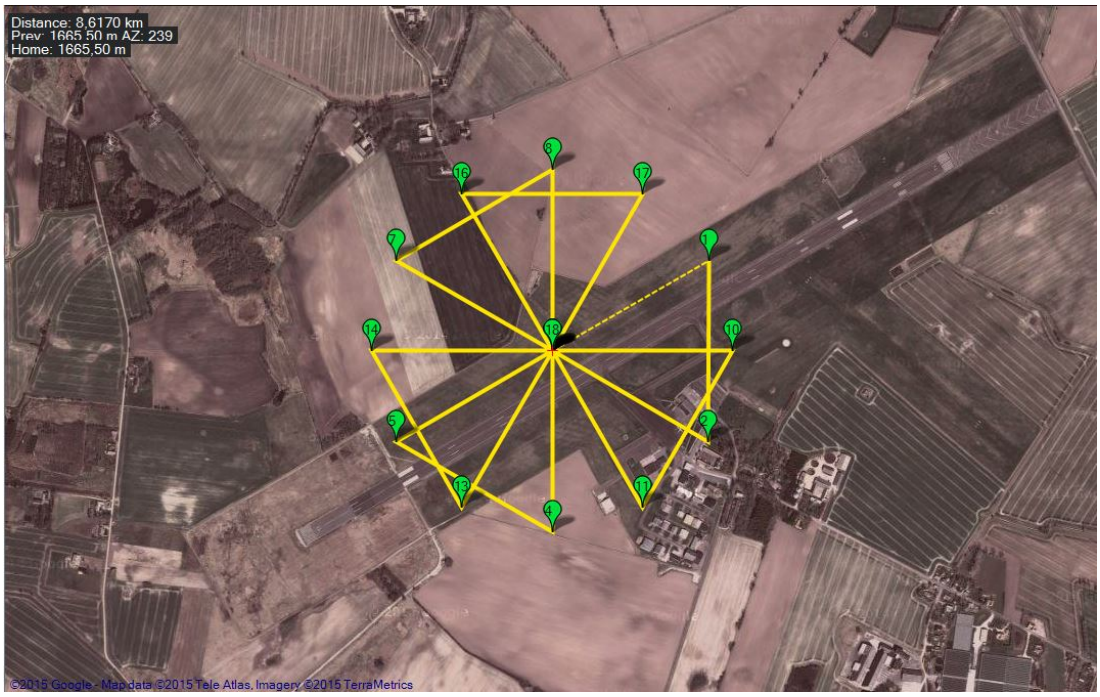


Figure 1.8: Sector search pattern example generated from the path of figure 1.5. Waypoints 0, 3, 6, 9, 12, 15 and 18 are the center (starting) coordinate.

2. Human detection

In this section, the solution adopted for the vision-based human detection part of the project, the arguments that led to this approach, the assumptions made, the tests performed and the obtained results are presented.

2.1 Problem definition

Once the purpose of the vision system has been well defined, the operating conditions and the environment must be carefully studied in order to find the most suitable solution. An embedded vision system portable by a drone, robust and fast enough and with low energy consumption is required for the task. The searching for people in the sea entails specific problems that must be treated such as:

1. False positives due to any kind of ocean animals or rocks.
2. Changes in illumination conditions due to outdoors work.
3. Forecast behavior: wind, rain.
4. Moving background and platform.
5. Highly reflective background.

Therefore, the selection of the hardware and vision algorithm has to be taken consequently with these points.

2.2 Equipment

According to the ideas above and the available resources, a logically consistent choice seems to go through the image processing options. This is due to the fact that any other kind of visual system (thermal camera, structured light, time of flight measurements, etc) does not fulfill the requirements or turns out to be too expensive. Hence, a GoPro HERO 3 camera has been chosen as a robust, light and powerful enough solution [12]. In order to deal with the instability and disruptions caused by the platform, the camera has been mounted on a Tarot T-2D Brushless Gimbal Kit, whose technical details can be found in [5], which is used as a stabilizer.

2.2.1 GoPro HERO 3

The technical parameters of the chosen camera model have influenced the way the vision algorithm has been approached. The characteristic small focal length of this camera model, together with the flight altitude of the drone, leads to a wide field of view, and therefore a big scanned area of ocean per frame. The high resolution of the frames allows a precise image processing with a low loss of information in the acquisition even from high altitude. Its elevated frame rate for video is, however, constrained by the processing time of the vision algorithm, which has been developed striving for low

memory and time consumption since it processes the images online and it can be the bottleneck of the system.

2.2.2 GoPro gimbal

The installed gimbal Tarot T-2D Brushless Gimbal Kit offers roll and tilt stabilization and has been configured to point the camera downwards due to its wide FOV. By doing this it is ensured that once the drone is offshore the images will only contain a background of the ocean, facilitating the image processing and speeding up the preprocessing of the pictures. Furthermore, as said before, the gimbal provides high stability for precise image acquisition and filters out disruptions on the flight, which allows an easier and more reliable human detection system.

2.3 Vision task

Given the hardware platform, the image processing algorithm must deal with the remaining problems in order to reach the goal of the system. The initial proposals for people detection, based on movement analysis and human morphological features extraction where discarded after being subjected to more detailed studies. The moving platform and the waves made unfeasible a reliable solution based on movement tracking assuming that the people to be found were moving. Furthermore, the possibility of the targets being drowning, partially or almost completely covered by water prevented any kind of solution from succeeding employing morphological analysis. In the view of the arguments above, the final solution was decided to perform human skin detection based on color under the assumption that at least a part of the targets would be floating.

2.3.1 Human skin identification

In order to guarantee a robust detection, the RGB-H-CbCr human skin color model presented in [15] and [16] has been implemented in the algorithm and applied for image thresholding.
As a result of the training presented in the paper, specific constraints in the subspaces have been determined, so that a general skin color model invariant to illumination and ethnicity was yielded. The model is given by the set of bounding rules in the three subspaces in equations 2.1, 2.2, 2.3 and 2.4, explained below.
As a result of the training, specific constraints in the subspaces have been determined, so that a general skin color model invariant to illumination and ethnicity was yielded. The model is given by the set of bounding rules in the three subspaces in equations 2.1, 2.2, 2.3 and 2.4, explained below.
The RGB space constraints are obtained from [15] and represent the skin color at uniform daylight illumination 2.1, and under flashlight or daylight lateral illumination 2.2. These two excluding conditions are merged into rule A by means of an OR logical condition.

$$\begin{cases} (R > 95) AND (G > 40) AND (B > 20), & \text{AND} \\ (max(R,G,B) - min(R,G,B) > 15), & \text{AND} \\ (abs(R-G) > 15) AND (R > G) AND (R > B) \end{cases} \tag{2.1}$$

$$\begin{cases} (R > 220) AND (G > 210) AND (B > 170), & \text{AND} \\ (|R-G| <= 15) AND (R > B) AND (G > B) \end{cases} \tag{2.2}$$

According to [16], the discrimination in Cb-Cr domain is defined by the constraints showed in 2.3. These boundaries define a 2D region which contains all the skin color pixels. Therefore, these conditions must be all fulfilled at the same time, and are therefore combined by and AND to become the second rule B.

$$\begin{cases} Cr \leq 1.5862Cb + 20 \\ Cr \leq 0.3448Cb + 76.2069 \\ Cr \leq -4.5652Cb + 234.5652 \\ Cr \leq -1.15Cb + 301.75 \\ Cr \leq -2.2857Cb + 432.85 \end{cases} \tag{2.3}$$

Finally, as explained on this report below, in the HSV space, the hue values shows the strongest separation between skin and non-skin regions. Since as a previous step in the algorithm the H subspace is thresholded dynamically, this third set of rules for the color model could be neglected. However, for the sake of completeness they are exposed here. The constraints in equation 2.4 are combined by an OR, becoming rule C.

$$\begin{cases} H < 25 \\ H > 230 \end{cases} \tag{2.4}$$

As a result, the three rules must be accomplished by any pixel in the image to be considered human skin, and so they conform the filter. This filter is applied after the input has been binary segmented and ROI's of the white blobs has been selected, as it will be detailed in the next section.

Each ROI will be subjected to the filter, and if it deems the ROI to contain a human it will be noted, otherwise ignored. The filter is applied on every pixel of the image, the response of the pixel is saved in a map. If the filter response deems the pixel to be something from a human it will output a white pixel if not a it will output a black pixel. If the percentage of white pixels in the ROI is above a threshold value, will it deemed as a human. An example of successful filtering can be seen in images 2.1 and 2.2.

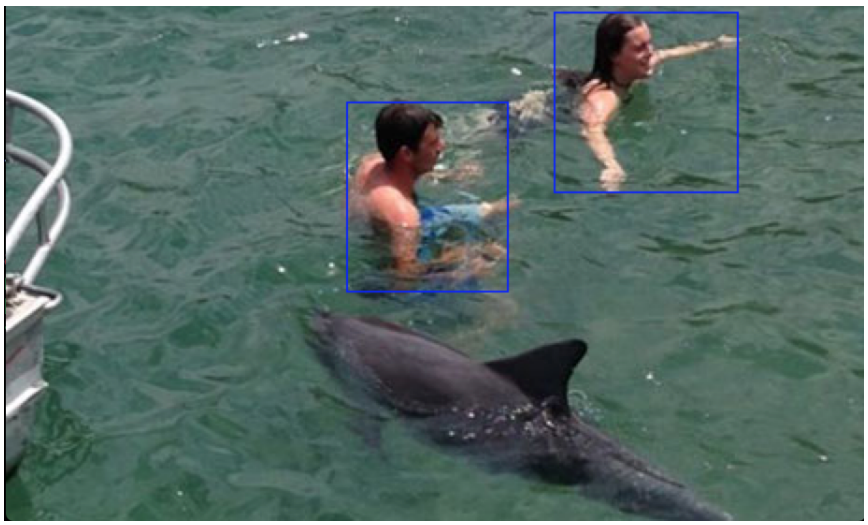


Figure 2.1: Skin detector capable of distinguishing between animals and humans.

Figure 2.2: Response map of the detected ROI, see figure 2.1.

Ordinary detectors such as HaarCascades, are trained to detect people within a certain pose, as this is a factor which cannot be kept consistent for the application the choice of applying this detector seemed reasonable.

Applying these types of filter on a image can be time consuming, esspeacialy if the camera has a wide angled lens. This issue becomes redundant wihtin this program, as the preproccesing part creates ROI at which the filter can be applied on. This way is computational time needed shrinken very much, and make decision of angle of view redundant.

2.3.2 Preprocessing

The pixel-wise searching method is a very accurate but heavy process that entails a high time consumption. In order to speed up the overall performance, some regions of interest in the image with higher possibilities of containing people must be found as a previous step. To do so, an analysis and comparison of example images of the ocean, with and without people and under different illumination conditions was carried out to determine a procedure to find these regions.
A first approach to this consisted in filtering in the frequency domain in order to remove constant low frequencies, which would be, in this case, related to the background. Thus, only the high frequency objects would be left, containing all the isolated candidates for the detector. Notwithstanding, it turned out that a simpler and faster threshold in the hue color space could be set up to distinguish between floating objects and the background water in all the images. These floating objects always involved a disruption or change in the average hue level of the whole picture, that otherwise remained constant (except for strong glare cases, which will be treated below). However, a threshold value in hue could not be fixed, due to its dependency with the general illumination level. Thus, the solution adopted was to set up a dynamic threshold applying Otsu's method, as described in [3]. Proceeding this way, a few small regions containing all the candidates could be input to the skin scan process, getting rid of the majority of the image with no relevant information. This part of the process contains the strongest assumption made for the algorithm. For the binarization of the image a uniform background is assumed, meaning that this step will tend to fail for images with strongly non-homogeneous surroundings. It can handle ocean and sky or distant coast lines though.
Furthermore, an estimation was made here concerning the size of the input ROIs. According to flight altitude and the focal length of the camera, an approximate calculation of the size of the pixel are of people in the image can be made. With it all the objects out of this boundaries can be filtered out, easing the posterior process.
As the last point on the preprocessing part, a simple and easy to implement method for dealing with

the glare of the sun light on the water is suggested here, although not implemented in the project, since we came out with this problem at the very end. The thresholding in hue was proved to fail in the cases of light reflection on small spots for certain intensities, since this glare effect caused also a strong enough change in the hue image histogram. To eliminate these false positives, the easiest solution would be to include a polarizer filter in the setup, which could also facilitate the color detection.

2.3.3 Postprocessing

The outputs of the whole process are the images considered to contain people on the water in the patrolled area. These pictures are stored and sent to the ground control to be analyzed by the staff in charge, introducing a human in the control loop. If the pictures are confirmed to contain people, this means that they are swimming in a forbidden zone or drowning, and the specific intervention activities are triggered, while the drone keeps its patrol.

2.4 Performance tests

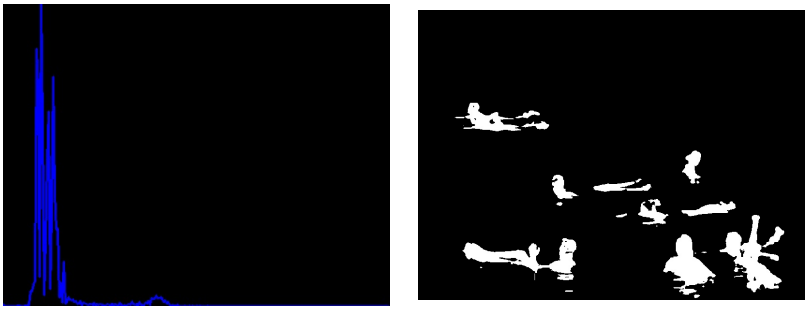
The design and implementation of the vision algorithm, developed in C++ and OpenCV, has been an iterative process consisting of several steps of coding and testing of the different parts of the program. For testing purposes, a database with several relevant images and videos that fulfilled the specifications for the application has been created. It must be said here that, due to the special characteristics of the problem (listed in 2.1) and the constraints applied during the design of the vision algorithm to optimize it, the videos recorded manually trying to emulate similar conditions of the problem failed to help test the program. In order to simulate the ocean, a consistent background had to be present in the videos, but could not be defined for the trials.
The following example of successful test for an image in the PC deploys in Figures 2.3, 2.4, 2.5 and 2.6 the main outputs of the intermediate processes explained above, together with the initial image and the final result.

(a) Image 1: original image (b) Image 1: processed image

Figure 2.3: Original and processed images.

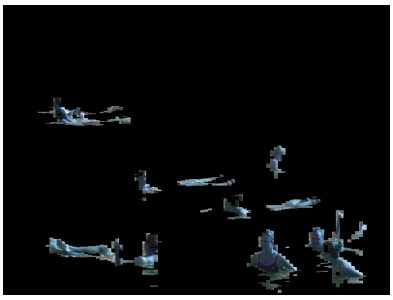
(a) Image 1: hue (b) Image 1: saturation (c) Image 1: value

Figure 2.4: HSV color space decomposition

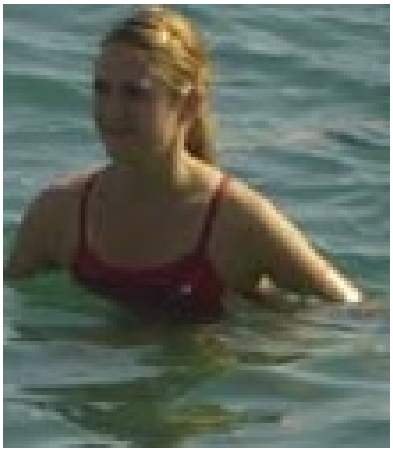


(a) Image 1: hue histogram (b) Image 1: binarized

Figure 2.5: Dynamic thresholding in hue



(a) Image 1: ROIs before particles filtering



(b) Image 1: one of the ROIs amplified after filtering

Figure 2.6: Example of region of interest extracted.

2.5 Results

The developed vision system has proved to be robust and reliable for the stated problem. Its use mounted on a drone, although not tested, can be assumed to be successful since the sum of the developed and tested parts should result in a working system. The OpenCV application resulting of the implementation of the algorithm designed demonstrates to be able to filter out animals, rocks and boats and detect humans off shore under the stated assumptions. In addition it can deal automatically with illumination variations, disruption on the platform and glare effects. However, there are still some

particular and very seldom cases in which the vision process fails. These mistakes are not critical though since the process includes, as said in the statement of the project, a human in the control loop, who will be, in the last case, the one in charge of filtering out false detections before activating rescue procedures.

2.6 Summary

An algorithm capable of detection human far away in the ocean has been developed. The algorithm uses a skin color model to detect humans, making it robust and able to distinguish between a human other items. The skin color model, from the report [16] is based on different form of illuminations, making it invariant to illumination differences. The algorithm has been tested with different images found on the internet, at which it performs well. It is capable of distinguishing between humans and animals as seen in figure 2.1. The chosen hardware works well on a moving background, and is thereby capable of retrieving useful footage for analysis.

2.6.1 Further work

In order to constrain the scope of the project, it was assumed from the beginning that a successful offline performance of the application in a laptop processing a video could be extrapolated to good results for online performance. According to this, the tests have been carried out making use of acquired or downloaded videos. As future work, it is left the implementation of the necessary hardware on the drone in order to stream the video and send it to the ground control to be treated. It is suggested the use of an added board such as a variant of the Raspberry Pi or similar, together with a USB Wifi adapter. This setup would provide an easy to install and fast connection for video streaming that would solve the communication. Furthermore, the ultimate test for the system to be subjected to would be the treatment of a video of the defined scenario acquired by itself.

3. Conclusion

In section 1, autonomous flight on the Iris+ drone was explored using the APM Planner 2.0 software. It was discovered that communicating with the drone through this software and through the Mavros ROS package, as well as uploading a path to the drone, was achievable. A system setup guide explaining the steps necessary to make the Iris+ drone fly autonomously has been provided in section 1.2. A path generation framework was developed in the form of a file format and waypoint manipulation C++ library, and in the form of three example Matlab scripts for generating search pattern paths. It was shown possible to design and upload custom designed search patterns to the drone via Mavros. This gives the possibility of easily generating a flight path and launching a drone, without any advanced knowledge of drone flight, if the processes described are integrated into a GUI application.

During the tests it was found that security and pre-flight checks are crucial in autonomous drone flight as a lot of unpredictable elements are involved. The tests also showed the excellent capabilities a relatively cheap drone has in following a path with no obstacles, even in windy conditions.

In section 2, an algorithm capable of detecting humans drowning offshore has been developed and presented and can be founded in [2] . In order to test its functionality, a database with some of the pictures and one video used for its verification can be found in [1]. The detection algorithm, based on the human skin color model presented in [16], has proved to be robust, illumination and motion invariant and able to distinguish between people and other objects on the ocean. The images at which the algorithm has been tested on, shows the algorithm works best on images with consistent background, as intended.

Bibliography

[1] Location of test images and video used for the vision algorithm. https://github.com/madsherlock/SDU-UAS-15-Group1-Report/tree/master/Data/Test_images.

[2] Location of the vision algorithm. https://github.com/madsherlock/SDU-UAS-15-Group1-Report/tree/master/Data/Code/Vision.

[3] A threshold selection method from gray-level histograms. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(1):62–66, Jan 1979.

[4] 3DR. 3dr radio v2. http://copter.ardupilot.com/wiki/common-optional-hardware/common-telemetry-landingpage/common-3dr-radio-version-2/ - 18/05-2015.

[5] 3DR. Tarot t-2d brushless gimbal kit. https://store.3drobotics.com/products/tarot-t-2d-brushless-gimbal-kit - 20/05-2015.

[6] DanDrone. Drone til beredskab. http://dandrone.dk/shop/drone-til-beredskab-223p.html - 18/05-2015.

[7] dji. dji phantom. http://www.dji.com/product/phantom-2 - 18/05-2015.

[8] Dronecode. Apm mission planner 2 home. http://planner2.ardupilot.com/ - 20/05-2015.

[9] Dronecode. What is a multicopter and how does it work. http://copter.ardupilot.com/wiki/introduction/what-is-a-multicopter-and-how-does-it-work/ - 20/05-2015.

[10] Vladimir Ermakov. mavrose. http://wiki.ros.org/mavros - 20/05-2015.

[11] FrSky. D4r-2. http://www.frsky-rc.com/product/pro.php?pro_id=24 - 18/05-2015.

[12] GoPro. Hero 3 silver edition user manual. http://www4.uwm.edu/psoa_er/manuals/cameras/GoProHero3-Silver.pdf - 18/05-2015.

[13] Virtual United States Coast Guard. SAR Fundamentals Lesson 5 – Search Patterns, 2008. http://www.vuscg.org/training/pilot/SAR_Fund/SAR_School_L5.htm.

[14] Åke Refsdal Moe. Droner: reddende engler i arktis? http://teknologiradet.no/sikkerhet-og-personvern/droner/droner-reddende-engler-i-arktis/ - 18/05-2015.

[15] J. Kovac, P. Peer, and F. Solina. Human skin color clustering for face detection. In *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, volume 2, pages 144–148 vol.2, Sept 2003.

[16] J. See N. A. Abdul Rahim, C. W. Kit. Rgb-h-cbcr skin colour model for human face detection. In *MMU International Symposium on Information and Communications Technologies (M2USIC)*, Petaling Jaya, Malaysia, 2006.

[17] Pixhawk. px4 autopilot. https://pixhawk.org/choice - 18/05-2015.

[18] Politikken. 18-årig fundet druknet to uger efter kæntring. 2013. http://politiken.dk/indland/ECE1963367/18-aarig-fundet-druknet-to-uger-efter-kaentring/ - 18/05-2015.

[19] qgroundcontrol. Mavlink micro air vehicle communication protocol. http://qgroundcontrol.org/mavlink/start - 18/05-2015.

[20] 3DR Robotics. Iris+. https://store.3drobotics.com/products/iris - 18/05-2015.

[21] ScienceDaily. Shape-shifting robot plane offers safer alternative for maritime rescue. 2010. http://www.sciencedaily.com/releases/2010/08/100826103835.htm - 18/05-2015.

[22] S. Waharte and N. Trigoni. Supporting search and rescue operations with uavs. In *Emerging Security Technologies (EST), 2010 International Conference on*, pages 142–147, Sept 2010.

Appendices

A. Work contribution

Mathias:
Report - Document setup, Frontpage, Introduction, Section 1.1 to 1.4, Conclusion, referencing.
Practical - Drone setup, tests, mavros, debugging.
Ignacio:
Report - Introduction, Section 2.
Practical - Vision system setup, image processing algorithm design and testing.
Keerthikan:
Report - Human detection section 2
Image processing algorithm, implementing and designing it.
Steffen:
Report - Abstract, Introduction, Project Proposal
Practical - Pathplanning, Robo-Cage Test Pilot, Driver.
Mikael:
Practical - Path planning, C++ and Matlab software development, drone setup, field testing (at laptop).
Report:

- Corrections and additions to document setup, frontpage, abstract, introduction, conclusion and sections 1.1 through 1.4.
- Sections 1.5 and 1.6.
- Corrections to section 2.

Ander:
Practical - First phase of image processing algorithm.

B. Project Proposal

The Lifeguarding Drone - Group 1 project proposal

1 DESCRIPTION

Patrolling and ensuring safety at remote and humanly inaccessible areas such as the sea has for a long time been a difficult task requiring expensive equipment and many man hours. With the recent development of readily accessible drones, which are easy to use and fairly simple to operate, a revolution within the area of life guarding seems within reach.
Utilizing the drones capability of operating and monitoring humanly inaccessible areas to control coast areas could not only result in a more safe coast environment, but also a decreased cost in regards to life saving. The use of drones in regards to emergencies and rescues has already showed promising results, but mostly using humanly controlled flights.
This group's vision is to explore the possibilities of an easy to use and implement drone that fly autonomously within the specified area. Using vision to detect people in the current scene or movement it should report back with images from the scene. The images can then be processed by a professional. Using this approach a minimum amount of man hours is needed while the possibilities of autonomous drone flight will be utilized.
Furthermore, the possibilities of dropping rescuing equipment will be investigated if it is found feasible.
The idea is to deal with the problem as a sum of modules that can be developed individually by different group members and tested together, so that the maximum group efficiency can be reached while being flexible within each module.

1.1 EXAMPLE PROBLEM: PATROLLING THE FORBIDDEN ZONE

In crowded beaches, the lifeguard is busy watching the people in the water. The people far out in the sea are at greater risk than the people close to the coast due to two reasons: The water is more treacherous, and they are less visible from the shoreline. The risk needs to be limited, and this is done by saying that you are not allowed to swim out further than a given distance d defined by the coast guard regulations / beach rules. The sea should be patrolled in this "forbidden zone" (some bounded area of sea further out than the distance d) in order to ensure that no one is swimming there. Currently, this patrolling is typically done by a manned boat. We want to replace this patrolling of the sea further out than d with an UAS. The reason: It's cheaper to have a drone patrol than a manned boat. Bear in mind that rescue teams are still needed in emergencies. This drone is meant as a replacement for the constant patrolling.

Patrolling:

1. The operator gives the drone a polygon defined by GPS coordinates of the area of sea to patrol.
2. The operator selects a patrolling strategy (circle/spiral/etc) and coverage percentage.
3. The drone will fly at an approximately constant altitude (high altitude) until it detects a possible person in the water. Here, it employs its scene investigation strategy.

Scene investigation:

1. Find out if the scene contains a person: Move down to a lower altitude and determine if it's a person.
2. If a person was found, the drone alerts a ground system and performs its response task:
    - Take a number (possibly 6) of pictures from different angles of the scene and send them to the operator. The drone flies in a circle around the scene (person) and takes the pictures while keeping the person within the field of view of the gimbal-mounted camera.
    - Possible additional tasks, such as asking operator if a "swim ring" or something similar should be deployed.
3. The drone returns to the high altitude and continues its patrolling.

Regarding battery life of drone: When battery level is low, the drone should return to a ground station. Battery recharging and/or replacement is out of scope of our project. Weather conditions (in the air) are not something we will focus on. This makes the drone choice more trivial. Focus points: Intelligent path planning algorithm, useful computer vision algorithm, and innovative scene investigation strategy. Assumptions: The drone can communicate with the operator system and transmit (6 images) and receive (commands) the necessary data.

2 WORK PACKAGES

2.1 SYSTEM DETERMINATION

Selection of drone platform, software platforms, hardware etc. Division of group into the next work packages. Everyone works on this work package. Starting point: The drone flies autonomously and sends pictures of interesting features and their GPS coordinates back to the operator. It continues on its path, and only returns to the features if the operator changes the flying path. Based on what the firefighters are using right now, we go with the copter approach (as opposed to the winged drone).

2.2 SEPARATED WORK

2.2.1 Path planning: Steffen, Mikael and Mathias

- Interface with the chosen drone.
- Choose/implement waypoint navigation (point to point flying) interface (eg. Google Earth)
- Outline GUI: Display drone intput and output in a simple way.
- Choose point to point flying strategy (eg. straight line)

- Choose scene investigation strategy.
- Define basic protocol for scene investigation - when does it start and when does it end? This is for the interaction with the feature detecction code.
- Implement point to point flying strategy on chosen drone.
- Implement scene investigation strategy on chosen drone.
- Make working GUI with point to point path planning and picture feed.
- Implement "Manual mode" with chosen drone and GUI.
- Implement altitude control for manual mode and for autonomous flight. Take vision algorithm requirements into account.
- Implement the switch from point to point flying to scene investigation based on detected features.
- Test usability of GUI for path planning and scene investigation feedback.

2.2.2 Feature extraction: Keerthikan, Ignacio and Ander

- Defining problem and how to solve it: How to solve it, Resources, Embedded devices going to use, Camera choice, Range definition, Flight speed, FPS, Robustness check => lens check.
- Solving the problem on PC => code the solution: Coding C++, OpenCV, Optimization
- Implement to the devices. – embedded device: implement the code
- Mounting on drone: Adjusting code according to placement of camera, Testing the setup

2.2.3 Interface specification: Everyone.

Very important, to make the merging of code easy and goals clear.

2.3 SYSTEM FINALIZATION (27TH OF APRIL TO 22ND OF MAY)

Merging of code and final tests. Everyone works on this work package.

2.4 WRITING THE REPORT (22ND OF MAY TO 29TH OF MAY)

Everyone focus on writing the report and documenting our work.

3 MILESTONES

17th of March: Initial choices

- Documentation of drone choice and interfacing basics.
- Documentation of GUI choices so far – including pictures/drawings.
- Documentation of the flying strategies, including the scene investigation protocol.

14th of April: Current source code

- Flying strategy implementation source code.
- Working simple GUI source code.

27th of April: Working, ready source code

- Documentation of "Manual mode".
- Documentation of altitude control.
- Documentation of the GUI and the usability test results.

4 TIME PLAN

17th of February – 3rd of March: Selection of drone platform, software platforms, hardware etc.
17th of February – 10th of March: Outline GUI displaying some drone input and output in a simple way.
24th of February – 17th of March: Choose flying strategies and define some sort of protocol for the flight state transitions.
3rd of March – 17th of March: Choose approach to the machine vision problem and develop a beta version of the machine vision algorithm running on a PC.
17th of March – 7th of April: Implement first version of machine vision on the drone.
17th of March – 14th of April: Implement code to make the drone fly with our own flying strategies for waypoint navigation and scene investigation. Simultaneously, make a simple, working GUI.
7th of April – 21st of April: Optimization of implementation of machine vision code on the drone. Test machine vision code.
14th of April – 21st of April: Implement altitude control.
14th of April – 27th of April: Implement remaining functions, including "Manual mode". Test usability for path planning and scene investigation feedback.
27th of April – 22nd of April: Overall system testing and merging of code.

5 HUMAN RESOURCES AND MATERIALS

The entire group studies Robot Systems. Keerthikan and Mikael have less visual processing experience. However, as Keerthikan is writing his bachelor's thesis on face detection, he will quickly become accustomed to computer vision throughout the semester. Therefore, he, Ignacio and Ander will be working on the computer vision parts of the project, combining their experiences. Steffen, Mikael and Mathias, having experience with the Qt framework and GUI construction, will focus on the path planning parts of the project. This common ground makes this a robotics project, and everyone will be able to help each other.
IRIS+ drone: General purpose, has GPS, is known to us, and it is here on TEK already.
Optimal, but out of scope drone: Wind-resistant, waterproof drone.
Camera: GoPro Hero4+ or GoPro Hero3 with the Tarot T-2D gimbal for GoPro. This gimball works with a cable connection to the GoPro.
IRIS+ includes a set of tall legs to use with the Tarot Gimbal. Users with an original-edition IRIS will need to purchase a set of tall legs to use the gimbal (available soon).