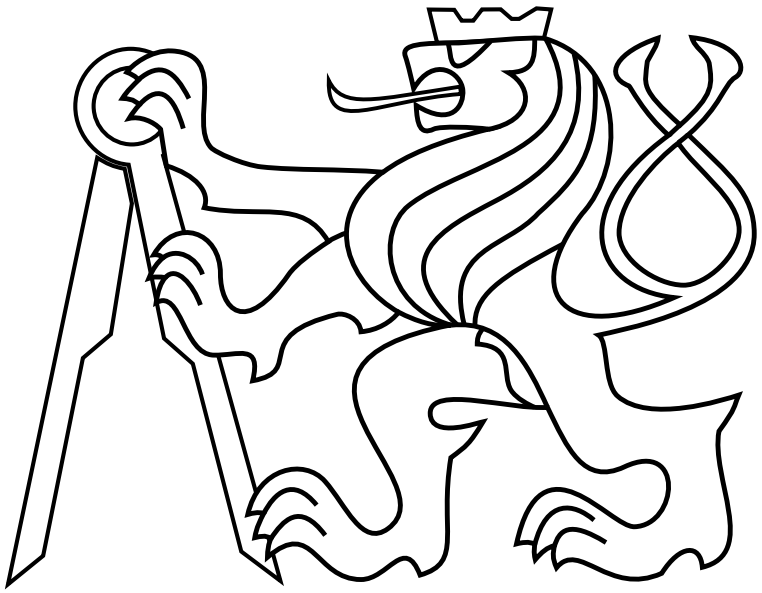
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR'S THESIS



Štěpán Klouček

Android application for control of an unmanned helicopter carrying a bird repeller

May 2018

Department of Control Engineering

Thesis supervisor: **Dr. Martin Saska**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Klouček** Jméno: **Štěpán** Osobní číslo: **457180**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Systémy a řízení**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Mobilní aplikace pro řízení bezpilotní helikoptéry nesoucí plašič ptactva

Název bakalářské práce anglicky:

Android application for control of an unmanned helicopter carrying a bird repeller

Pokyny pro vypracování:

The goal of the thesis is to design, implement in Android and ROS (Robot Operating System), and experimentally verify in a Gazebo simulator and real experiments an application to support control of an unmanned aerial vehicle (UAV) equipped by an acoustic bird repeller. The following tasks will be solved:
1. To design an Android application for control of UAVs equipped by an onboard computer with ROS in the task of protection of vineyards against birds. The user will be able to set GPS way points defining a UAV path and a profile of sound properties along this path.
2. To select a proper sound generator for initial, integrate it into the UAV system of the MRS group at CTU [1,2] and implement an interface for its smart using based on current altitude and distance to the flocks.
3. To implement a suitable/proper path planning algorithm (based on a variant of TSP or OP problem) to find a feasible short path through selected places of the bird repeller operation.
4. To verify system functionalities in Gazebo and with a real platform in outdoor conditions (only technical properties of the system will be evaluated, while the influence of the flying repeller on birds will be a subject of consequence research in cooperation with Czech university of life sciences Prague).

Seznam doporučené literatury:

[1] T. Baca, P. Stepan and M. Saska. Autonomous Landing On A Moving Car With Unmanned Aerial Vehicle. In The European Conference on Mobile Robotics (ECMR), 2017.
[2] G. Loianno, V. Spurny, J. Thomas, T. Baca, D. Thakur, D. Hert, R. Penicka, T. Krajnik, A. Zhou, A. Cho, M. Saska, and V. Kumar. Localization, Grasping, and Transportation of Magnetic Objects by a team of MAVs in Challenging Desert like Environments. IEEE ICRA and RAL, 2018.
[3] Shripad Gade, Aditya A. Paranjape, and Soon-Jo Chung. "Herding a Flock of Birds Approaching an Airport Using an Unmanned Aerial Vehicle", AIAA Guidance, Navigation, and Control Conference, AIAA SciTech Forum, 2015.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Martin Saska, Dr. rer. nat., Multirobotické systémy FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **14.02.2018** Termín odevzdání bakalářské práce: **25.05.2018**

Platnost zadání bakalářské práce: **30.09.2019**

Ing. Martin Saska, Dr. rer. nat.	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university thesis.

Prague, date ....

signature

Acknowledgements

I would like to thank everyone in Multi-robots research group for their advice. Especially Dr. Saska for his leading, D. Heřt for his help with sound system and T. Báča for his advice. Also, I would like to thank my coworker P. Ješke for many late night discussions about how the application should look like.

Abstract

This work aims to create a complete solution for bird repulsing on vineyards by creating a mobile application capable of controlling UAV with sound system mounted onboard. The user is able to create and edit waypoints in an interactive map implemented in the created application. These waypoints can be sent to UAV to perform area patrolling, to fly exact trajectory defined by these waypoints or to fly computed trajectory by path planning algorithm. To compute trajectories feasible for the UAV an algorithm based on a variant of orienteering problem is presented here. Multiple options of the sound system are discussed in this thesis. Experiments are verifying the functionality of the whole solution in an outdoor environment.

Keywords: Android application, UAV, bird repulsing, vineyard protection

Abstrakt

Cílem této práce je vytvořit kompletní řešení pro plašení ptactva na vinicích, pomocí mobilní aplikace schopné ovládat bezpilotní letoun a zvukové zařízení na něm umístěné. Uživatel aplikace je v interaktivní mapě schopný vytvořit a upravovat letové body. Tyto body lze odeslat UAV, které je schopné střežit prostor, letět trajektorii podle zadaných bodů nebo aby letělo trajektroii vypočítanou navrhnutým algoritmem. Pro vytváření proveditelných trajektorií, byl vytvořen algoritmus na základě "orienteering problem". V práci je také řešeno několik možností jak vytvořit zvukový systém. Experimenty testují funčnost navrhnutého řešení v reálných podmínkách.

Klíčová slova: Android aplikace, UAV, plašení ptáků, ochrana vinic

Contents

1 Introduction

An unmanned aerial vehicle (UAV) is a remotely controlled or autonomous flying vehicle. Vertical Take-Off and Landing (VTOL) multirotor vehicles are popular, due to their capability to fly and land almost everywhere. Most common vehicles from this category are helicopters and multirotors vehicles with fixed propellers. In particular, the popularity of multirotors vehicles is on the rise because of low their price, easier usage and maintenance compared to the helicopters.

Quadrocopter is typically powered by four electrical motors. Hexacopters with six motors are used for heavier payloads. Also the advantage of hexacopters is that in the case of motor malfunction, the opposing motor to the malfunctioned one can be turned off to stabilize the vehicle. This approach is only possible when the weight of the vehicle is not exceeding capabilities of four remaining motors. Purpose of such an aircraft can vary greatly with sensors installed and equipment mounted. Cameras are a common utility for hobby drones . The thermal camera is usable in a search for lost people [7] or detecting a faulty solar panels [22]. Sensors can be used for performing aerial surveillance of volcanic area [8] or environmental scanning [24] [10] [33]. Many large-scale business projects emerged in the last years that use UAV for goods delivery [26] or for warehouse management [2].

This thesis aims to propose and implement a solution for the bird repulsing using UAV controlled by a mobile application. This thesis specification match real industrial project requirements. To create such a solution three parts have to be created: the mobile application, sound system and path planning.

Path planning is described in chapter 4. There is a short description of orienteering problem and its usage for bird repulsing. Furthermore, implementation of the used algorithm is described in that chapter. The sound system is discussed in chapter 6. Several solutions are described there and two of them were tested on real UAV.

Many similarities can be found between bird repulsing and ground mapping in precise agriculture, which are also solved by our team in this industrial project and which are discussed in the work of P. Ješke [16]. These similarities are in the communication from UAV to the mobile phone and the mobile application itself. For creating a better final solution, and as was requested in the project these two parts were created to work with both applications. The mobile application was mainly developed for this thesis and is described in chapter 8. In Ješke's thesis application is used with new added features such as an obstacle for trajectory which was not necessary to implement in this thesis.

Communication is briefly described in chapter 7 as the major part of it was created to support many different area coverages needed for ground mapping. For this thesis, communication is needed to receive status messages about UAV and to send flight commands. A status message consists for example of UAV position and battery status.

2 State of the art

Gade, Paranjape, Aditya and Chung [14] were solving a similar problem. Their work is focused on the bird herding near airports, using the n-wavefront algorithm. So they are diverting flocks away from the airport protected zone, instead of repulsing. The UAV for bird repulsion also exists, it is called "prohawk" and is advertised at bird-x.com. Prohawk is a MAV with mounted sonic bird repeller. Which is entirely automated (launch, patrol, land). The MAV used in this project is more modular as the mounted repeller can be easily replaced. Furthermore, the application created for this project allows for a higher level of automation, due to the possibility of saving trajectories and the usage of path planning algorithm.

2.1 Mobile application

An essential part of this project is the mobile application. There is no universal communication standard for UAVs. Every manufacturer has its unique mobile application. The most advanced application comes with the DJI drones as their market share is as high as 85%. A free version of the application allows a user to manually control the UAV and to capture photos or videos with it. The paid version of the application allows the user to create autonomous flight plans.

Also, some alternative application exists such as Litchi for DJI drones or PIX4Dcapture. These applications can be used only with limited types of UAV, due to the communication restrictions. These applications offer specialization of UAV eq. creating 3D models and ground mapping. The proposed application is designed to be able to control UAVs with Pixhawk and ROS, designed at CTU Prague [5].

2.2 Orienteering problem

Handling the UAV is a difficult task. Manual control can vary with different types of UAV manufacturer. Also the pilot needs to control multiple aspects to fly safely. Air traffic and remaining battery time are one of them. For the bird repulsion mission, the pilot would have to fly through numerous way-points, controlling the status of the UAV during the flight. To perform such a mission pilot would have to make a flight plan to ensure safety. Even with the flight plan, there is a chance of a human mistake. To prevent these mistakes path planning algorithm is used for the bird repulsion. Using such an algorithm can also save time as it is faster than creating a flight plan and create more efficient trajectories. To handle these tasks path planning algorithm based on orienteering problem is proposed.

The name of the "orienteering" algorithm originates from an outdoor game orienteer-

ing [9] [36] [29] played in rural terrains such as a forests. Competitors are usually equipped with a magnetic compass and a map. The map contains many points which are associated with a score. Contestants are starting from one of the points, and their goal is to visit as many of these points as possible within a limited time and reach the endpoint. Each point needs to be visited only once. Contestants of the game which arrive at endpoint after time limitation are disqualified. The winner is declared by the highest collected score. Often it is not possible to visit all of the points. The competitors have to carefully select which points they are going to visit, in order to maximize their total score and not violating the time restriction.

The Orienteering problem (OP) can be viewed as modified Traveling salesmen problem (TSP) described in [23] [30]. The TSP is declared as a problem in which a salesman wants to visit multiple cities in a way that the path generated is the shortest possible. If we add time constraint so that salesmen can not visit all the cities and all cities have non zero rewards modeled as expected sales value, we get OP or from older literature "selective traveling salesman problem", [20] or [21]. For both of the problems the starting point and the endpoint can be same, but generally, they are not.

As the OP is NP-hard (non-deterministic polynomial-time) [19] an optimal algorithm is time-consuming, so the researchers are focusing on a heuristic approaches. These approaches can be indeterministic using Monte Carlo technique [36] or more often deterministic [32] [9] [37]. To create algorithms for real-life vehicles, Dubins [11] approach can be used to create curvated trajectories. These trajectories are used for UAVs [29] or for underwater vehicles [31]. Also, evolution algorithms are used for the OP [17].

The OP has many real-life practical uses. One of the earliest models is presented by Golden, Assad, and Dahl [15]. This model simulates fuel delivery company, which distributes fuel among customers. The reward is modeled as the urgency of fuel delivery, that is increasing with decreasing fuel supply of a customer. Each day path which minimizes travel distance and maximizes customer satisfaction is chosen. Another use is building telecommunication network [35] or planning a tourist trips [34].

In this thesis, unique usage of the OP in the mobile application is being solved. Also the possibility of reward collection correlation is discussed here, as the action on TSP city could also affect neighbor cities. This possibility is further explained by using the close enough orienteering problem.

2.3 Close enough orienteering problem

Close enough orienteering problem (CEOP) or very similar variant correlated OP [38], extends rewards collecting from point to area. These approaches allow data from sensors to be collected more effectively. Furthermore the CEOP with the Dubins vehicle also exists [28] [13].

3 Specification

The goal of this thesis is to propose and implement a solution for a bird repulsion using UAV controlled by a mobile application. To create such a solution mobile application, sound system and path planning algorithm has to be created.

In the mobile application, the user is able to:

- connect from mobile phone to the UAV
- take of with the UAV remotely
- manually control the UAV
- create a new trajectory
- choose between three autonomous flight modes

Creating trajectory will be done in a terrain map. Every point of a trajectory is editable. The user can edit the height of a flight, forbid performing of a repulsion maneuver and set a reward for path planning algorithm. Points can be dragged to change their position. Unwanted points can be deleted.

To fulfill these goals UAV has to be equipped with the Wi-Fi and the GPS. The civilian accuracy of GPS is sufficient. Continuous connection to the Wi-Fi is needed. A UAV has to know its GPS position and from the ground distance. Also, it has to be equipped with an onboard computer able to run the ROS and the Xbee board to control the sound system. The whole solution should work with basic UAV [5] used by Multi-robot System group displayed in figure 1.

Figure 1: UAV of Multi-robot System group with neccesery components [4]

To start the application, the user needs Android device with API (Application Programming Interface) level at least 15 (Ice cream sandwich). This device must be able to connect to the same network as the UAV. Also, the internet connection or pre-downloaded Google Maps are needed to display the map correctly.

The sound system has to be mounted on the UAV. Control of the sound system is managed from the application by selecting different modes of flight. Furthermore 12V source is needed to power the sirens.

Planning algorithm will be based on the OP. As an input, it takes set of way-points defined by a user in the Android application. Every way-point has the position and the reward. Planning algorithm searches for feasible trajectory connecting way-points with the highest collected reward within defined time budget. Planning algorithm should be fast enough, that user do not recognize the background computation.

4 Path planning

Trajectory created by the user consists of multiple points. These points represent the position of bird flocks which are needed to be scared off. Location of bird flock is obtained from the user by the mobile application. Each point is associated with a non-zero reward based on user estimation, which can be influenced by the amount of grapes or price of wine in particular part of the vineyard. UAV is capable of performing repulsing maneuver which scares the flock on such point. The maneuver takes twenty seconds to be performed. The UAV is limited by its battery capacity, so the trajectory created by the user might not be feasible.

For feasible set of points path planning can shorten the length of the trajectory, resulting in saving flight time. For unfeasible trajectories, the new feasible trajectory will be computed. Such a planning algorithm can take away a lot of responsibility from the user and allows a less experienced user to use the application. Path planning algorithm described in this chapter is based on OP.

4.1 Repulsing birds and OP

A set of N vertices is given. Each vertice x represents a point where the bird repulsing maneuver should be executed. Starting and ending point coincide. Each of these points contains non-negative reward S_x, which represents the user-defined priority of maneuver based on the importance of that point. Time t to travel between each pair of vertices is known. Because the repulsing maneuver takes twenty seconds to perform, twenty seconds is added to each vertice. As the battery of the UAV is limited, not all vertices can be visited. The battery flight time is T_{max}, also referenced as time budget. The goal is to determine the path which maximizes the reward.

For computing the trajectory, a non-deterministic approach is used. At first, a circle with radius $T_{max}/2$ is made around the starting location. Every point outside this circle is deleted from the computation as it is not reachable within given time budget. If at least one point beside the starting point remains, the algorithm continues. To determine a new trajectory, the Monte Carlo technique is used. Such a method is used to iterate multiple times over stochastic procedure to determine an empirical mean or in this case to select the best outcome. Selecting such a result is shown in algorithm 1. The number of iterations is limited by time, rather than a fixed number of repetitions to ensure smooth user experience on any device.

Algorithm 1 Selecting best trajectory using the Monte Carlo technique

```
1: function getBestTrajectory
2:     timeStart
3:     while 1 > timeCurrent - timeStart do
4:         newTrajectory ← createTrajectory()                ▷ Algorithm 2
5:         if newTrajectory.reward > bestTrajectory.reward then
6:             bestTrajectory ← newTrajectory.
7:         else if newTrajectory.reward = bestTrajectory.reward then
8:             if newTrajectory.length < bestTrajectory.length then
9:                 bestTrajectory ← newTrajectory
10:            end if
11:        end if
12:    end while
13:    return bestTrajectory
14: end function
```

Creating a feasible trajectory is described in algorithm 2. Only points not violating the time budget are added to the trajectory. Function *timeBetween()*, returns the flight time between two points with additional twenty seconds added because of the length of the repulsing maneuver.

Algorithm 2 Creating feasible trajectory

```
1: function createTrajectory
2:     while unusedPoints.empty do
3:         newPoint ← nextPoint()                            ▷ Algorithm 3
4:         T_selected ← timeBetween(currentPoint, newPoint)
5:         if T_max > T_selected + T_current + T_toReturn then
6:             T_current ← T_selected + T_current
7:             currentPoint = newPoint
8:             trajectory.add(currentPoint)
9:              unusedPoints.remove(currentPoint)
10:        else
11:             unusedPoints.remove(currentPoint)
12:        end if
13:    end while
14:    return trajectory
15: end function
```

Algorithm 3 shows randomness in choosing the next point for the trajectory. From all unused points, four points are chosen based on the reward and the time of flight to the next point. From these points, one is selected based on the chance proportional to that point reward. Property of point, the difficulty is evaluated by a simple heuristic, seen in algorithm 3 on line five.

Algorithm 3 Stochaistic point selection

```
1:  function nextPoint
2:      unsudedPoints.sortByTime()
3:      unsusedPoints ← unsudedPoints.getFourBest
4:      for i = 0; i < unusedPoints.size; i++ do
5:          point[i].difficulty = timeBetween(point[i], pointCurrent)/point[i].reward
6:          totalTime = totalTime + point_i.time
7:      end for
8:      chance = getRandomNumber(from : 0, to : 1)
9:      for i = 0; i < unusedPoints.size; i++ do
10:         if point[i].difficulty/totalTime + timeFromOther > chance then
11:             return point[i]
12:         else
13:             timeFromOther = timeFromOther + point[i].time/totalTime
14:         end if
15:     end for
16:      unusedPoints.remove(currentPoint)
17:     return bestPoint
18: end function
```

All algorithms in this chapter are written in a pseudo-code only. Implementation in Java was relaying at arrays of edges and vertices, where vertices represent locations of bird repulsion and edges lines between them. To improve this algorithm, heuristic evaluating difficulty of the point can be changed. Such a change could be introducing of a center of the gravity for points. The disadvantage of this algorithm is a problem to escape local maximum. The concrete example would be four points with the same reward on one side and a greater number of points with a little lower reward on the opposite side. The algorithm would always choose four points with the higher reward at the beginning a thus ending in local maximum as it would not have enough time to visit points on opposite side. This scenario is possible but unlike in a real-life mission, but it could be prevented by a previously mentioned center of the gravity or by dynamic computation of points to choose from instead of fixed four points.

5 Path planning with close enough neighbors

So far the proposed approach was using the simplifying assumption that each maneuver is only associated with a given location. This assumption is correct for distant locations but for closer ranges, which can be easily inserted by the user, there is a correlation between points as performing maneuver on one point can also repulse flock on nearby points.

The usefulness of the Close enough OP for bird repulsion will be discussed here with an assumption it is going to be used in a path planning using growing self-organizing array proposed by J. Faigl [12]. Furthermore, a chance for bird repulsion is explained and computed here as it is needed for the GSOA (growing self-organizing array).

5.1 Close enough neighbors for bird repulsing

For the path planning with neighbors, many practical scenarios are proposed, which are all bind to retrieving the data from sensors. It is understandable that such neighborhood can be modeled as a communication range, as it is not required to be precisely on sensor's location, but it is sufficient to be in communication range to collect data from a sensor. In these paragraphs, the CEOP will be used for more specific non-sensor usage. So far to collect the reward, stopping the UAV on point and performing repulsing maneuver with t_{rm} length was required. Common vineyard practice is to scare the birds by using short and loud sounds, for example gun-shots. For the GSOA $\mathrm{t}_{rm} = 0$ is considered, as only a short and loud sound will be played. Precise visiting of each point is not necessary because the flock itself covers some area and sound system can provide sound loud enough to scare the birds from a distance. This means each reward can be collected from repulsing range δ which will be dependent on the strength of the sound system. Furthermore, there is a chance that performing this maneuver will affect nearby flocks outside repulsing range δ. This chance is presented for the distance from repulsing range δ to correlation distance ζ. Afer collecting sensor data the rewards of nearby sensors were decreased, as the information gain from nearby sensors is smaller after such a collection. For bird repulsion the reward of the correlated points would be decreased as well but also it would be added to the collected reward of the visited point, as there is a probability that repulsing was successful on nearby points. The proposed correlation is probability-based and it is possible to collect a portion of the reward multiple times during the flight through the correlation radius but is not considered here as it would be too different from CEOP. Also making such a noise too often would not be pleasant for people, and the birds are known to adapt quickly, so overusing this, it could lead to birds ignoring the device or shortening the effective range of UAV.

For the UAV in a range not exceeding δ whole reward of the point in radius will be collected. This applies to all points in the range. For range exceeding correlation distance

ζ, no reward will be collected. A portion of the reward from the points between these two ranges will be added to the collected reward, based on probability λ (probability of successful bird repulsing). The decreased reward of a point will be $r_i\lambda$. Also, the total collected reward is increased by that amount.

5.2 Probability of repulsing computation

To implement the GSOA algorithm, a probability to repulse the birds needs to be computed. At first it is neccesery to calculate the sounds intensity.

$$I = \frac{P}{S} = \frac{P}{4\pi R^2}. \tag{1}$$

and because senses including hearing are logaritmical (Fechner-Weber law) it is more logical to use decibels.

$$L = 10log\frac{I}{I_{ref}}. \tag{2}$$

Whole equation of a logaritmical intesity is created by merging the equation (2) into the equation (1) as follows

$$L = 10log\frac{P}{4\pi R^2 I_{ref}}. \tag{3}$$

The repulsing distance δ and correlation distance ζ needs to be computed, in order to know where to use final probability equation. Also two power values k_1 and k_2, which are explained later, has to be experimentally found or estimated, in order to compute concrete values. k_1 is power value where a chance of repulsing is $\lambda = 1$, in other words, it is minimal power value for successful repulsing. For values lower then k_1, there is a chance for a successful repulsing λ. Lowest power value for repulsing is k_2, even at this level chance of repulsing can be proposed as λ_{min}. Values k_1 and k_2 are dependent on base level of environment noise. From parameter k_1 the value of δ and from k_2 the value ζ can be computed, by substituting R as ζ or δ and expressing this value from equation (3) as

$$\delta = \sqrt{\frac{P}{\frac{4\pi I_{ref}}{10^{\frac{k_1}{10}}}}}, \tag{4}$$

$$\zeta = \sqrt{\frac{P}{\frac{4\pi I_{ref}}{k_2}}}. \tag{5}$$

With these values, it is possible to find function f which would map sound pressure level in decibels Ł to chance λ

$$f(L) = \lambda \tag{6}$$

If we compute equation (6) with value k_1, the result is 100% as it is definition of that value. For most precise assumptions the value k_2 result is zero. But it is better to propose new variable λ_{min}. This variable has to be set manually, and can make computation faster as with higher λ_{min} the correlation distance ζ is smaller, thus fewer points are affected by repulsion. With $\lambda_{min} = 1$ this problem is reduced to the CEOP.

$$f(k_1) = 1 \tag{7}$$

$$f(k_2) = \lambda_{min} \tag{8}$$

With knowledge that function f has to satisfy equation (7) and (8) the linear function f can be found as

$$\lambda = \frac{(L - k_2)(1 - \lambda_{min})}{k_1 - k_2} + \lambda_{min}. \tag{9}$$

This function (9) after merging into equation (3), gives us an equation to compute the chance of repulsiong based on distance R and power of speaker P.

$$\lambda = \frac{(10log\frac{P}{4\pi R^2 I_{ref}} - k_2)(1 - \lambda_{min})}{k_1 - k_2} + \lambda_{min} \tag{10}$$

Function (10) is used for computing chance of repulsing between range δ and ζ. Determining power values k_1 and k_2 will be solved by the cooperating university. Also, it is possible that the function (9) will be changed to match reality better, but is sufficiently satisfying for the initial implementation of the GSOA.

6 Sound system

The carried repulsor or sound system is an essential part of the bird repulsion process. The UAV itself is as loud as 90dB from 1m. Most of these noises are produced by whirling blades. The UAV-mounted sound system is proposed to increase the effectiveness of repulsion. Three approaches will be discussed:

- Ultrasound repulsor
- Playing sounds of predators
- Loud siren

Ultrasound is commonly used for commercial repulsors. Repulsor effectivity is enhanced by changing the frequency, within the ultrasound spectrum. Also, sudden change of power can further enhance effectivity of repulsor. Both these changes make affected area unpleasant for birds, resulting in birds leaving the area after a short time period. The main advantage is that ultrasound with a frequency lower than 20kHz is that it is on the borderline of human (audible) perception. Anyway, it is possible that the birds will ignore ultrasound for long-enough time to destroy valuable crops.

Older but still used method is using loud noises. A farmer can scare away the flock by shooting in the air. This approach requires someone patrolling area, moreover, shooting can be dangerous for birds or humans alike. A more modern system consisting of several generators capable of generating loud noises can be bought. The disadvantage is high construction/deconstruction time and loudness.

Playing pre-recorded predator voices is a more viable solution. These sounds are naturally threatening birds and do not have to be as loud as the previous method. Which makes this method more pleasant for humans around the area.

6.1 Speaker

Firstly playing pre-recorded sounds was chosen for this thesis. To overcome the loudness of properels at least 5W, reproductor was needed. Proposed reproductor is **Niceboy SOUNDair** seen in figure 2 with 10W of power, 480g weight, 3.5mm jack, and its own battery.

For handling sounds, SFML C++ library was used. An advantage of using a speaker is an easy connection to onboard computer and omnidirectionality. When tested with real UAV device, the speaker was as loud as propellers. Also, its high weight can compromise UAV's ability to maneuver. Because of that, another solution is proposed.

Figure 2: Niceboy SOUNDair speaker used for bird repulsing [6]

6.2 Siren

This solution consists of two ESP PS82 sirens 3. With 156g both, they are a third of the weight of the speaker. The sound pressure level is 122 dB from 1m. The signal is sent via Xbee board by Ros service. Powered by linear stabilizer on 12V and switched by a transistor. Main advantages of using sirens are low price, high durability, low power requirements and small weight. The disadvantage in this approach is controlling loudness, as siren can function from 3V to 14V with only 30% power drop between these values.

Figure 3: ESP PS82 sirens, final solution for bird repulsing [6]

7 Communication

Communication with a UAV is essential for this thesis. Current status of communication technologies and their usability for communication between the UAV and mobile device is described here. Subsection [2,3] of this chapter requires at least basic knowledge of ROS system. If the reader is not familiar with ROS system, it is suggested to read [25] or [27] as a basic introduction.

7.1 Communication technologies

Radio control is common method for manual control of a UAV. Range varies greatly with a used module. For cheaper modules range can be up to 5km. More expensive modules can provide control up to tens of kilometers. Low latency and great range make this technology best for safe manual control. It is possible to broadcast video feed from the UAV. As the mobile phone does not have a possibility to broadcast at these frequencies, it has to be connected to other controller or device with such a capability, which makes this technology impractical for mobile application.

Bluetooth effective range depends on its performance class. Most effective class 1 offers range about 100 meters. Mobile phones have Bluetooth as a basic utility, most of them are equipped with class 2 with an effective range of 10 meters. Also, support libraries exist for ROS platform making this viable solution for close range communication.

Wi-Fi technology offers up to 100 meters of reach with standard 2.4GHz frequency. This range can be greatly reduced in urban areas, due to overusing this communication band. It is possible to use 5GHz version, which is not as common as 2.4GHz type, to remove communication overusing problem (interference problem). Using this version leads to quicker transfer rates but also lowers reach of the network. The reach of a network for both versions can be enhanced by using antennas. The main advantage of using Wi-Fi for communication is that every Android with access to Google Store has to use this communication technique.

7.2 Communication via Wi-Fi network

The selected option for communication is Wi-Fi network as it is easily extendable and widely supported. When UAV with ROS and mobile phone are connected to same Wi-Fi network, three possibilities of communication were found: secure shell, packet communication and RosJava.

7.2.1 Secure Shell

The secure shell, shorter SSH is a secure communication protocol. After such a connection is established, between mobile phone and UAV, the instructions can be sent directly to a command line. This allows quick calling of ROS services and ROS messages without creating publishers or subscribers. For sending more complex messages as whole trajectories, extra communication layer would be required. Obtaining data about the UAV status is possible but not easy to achieve.

7.2.2 Packet comunication

With knowlodge of ROS system it is possible to send publisher like messeges without the need of ROS running on mobile phone.

```
1st Byte - Sync Flag (Value: 0xff)
2nd Byte - Sync Flag / Protocol version
3rd Byte - Message Length (N) - Low Byte
4th Byte - Message Length (N) - High Byte
5th Byte - Checksum over message length
6th Byte - Topic ID - Low Byte
7th Byte - Topic ID - High Byte
N Bytes  - Serialized Message Data
Byte N+8 - Checksum over Topic ID and Message Data
```

Figure 4: Packet format of rosserial protocol

To achieve such a communication rosserial protocol must be implemented with defined packet format seen in figure 4. This type of communication would be preferred as it would be independent from the ROS master running on UAV. However, it would require a considerable amount of time to create.

7.2.3 RosJava

This option implements the already mentioned rosserial protocol. RosJava is used to build Android applications as it provides the client library for ROS communication and architecture style of ROS. Which allows similar logic on the side of the Android application and the ROS running on the UAV.

As it is the recommended solution for the Android application, as it allows to quickly create an application without higher knowledge of the ROS operating system. Also, Github

repository named android core [18] exist. Android core contains several small projects as a reference how to use many tools stored here. This application is built on top of a tutorial presenting basic publisher and subscriber.

7.3 Comunication protocol

At first, every type of message has its own publisher and subscriber. Every publisher and subscriber need its thread. Application with more than 6 threads started to crash unexpectedly at the side of ROS core. To minimize the number of threads communication protocol was created. There is one subscriber waiting for a status message from UAV and one publisher sending commands to UAV, at the side of the mobile application. The UAV have one publisher sending status messages and one subscriber taking commands. Only parts important for this thesis are explained in next subsection. Further information can be found in the P. Ješke thesis [16].

7.3.1 UAV status message

The UAV status message from UAV to a mobile phone, containing all important data about UAV.

First element	**Messege type**
0	Position x in UTM coordinates
1	Position y in UTM coordinates
2	UAV altitude
3	UAV latitude
4	UAV longitude
5	Battery voltage
6	Aproximate battery voltage
7	Confirmation of recieved message

7.3.2 Command messages

Multiple commands messages can send. Each message is an array of stdFloat64. The first element of the message is determining a type of message. These messages are from mobile phone to the UAV.

First element	Messege type
1	Take off
2	Manual control message
7	Area message
8	Standart trajectory message
10	Siren test

Take off message This message contains no other usable data. The only purpose is to start to take off sequence. If the UAV is already in the air take of message has no purpose.

Manual control message Controlling UAV manualy is desriped in application chapter 8.2.1.

Element in array	Name
1	Left joystick angle
2	Left joystick distance from start
3	Right joystick
4	Right joystick distance from start

The angle of the joystick is in radians, distance from the start is number between zero for, a starting position, and one for the most distant locations.

Standart trajectory message

Element in array	Name
1	Size of trajectory
2	Number of parameters
3	UTMx
4	UTMy
5	Height
6	Will perform maneouvering
7	Reward
...	...

Size of the trajectory defines the number of the points. The number of parameters helps with backward compatibility as if any parameter is added in the Android application, it is still possible to use the older software on a UAV.

The total length of an array is *3 + trajectory size * number of parameters*, as the parameter 3-7 for this case are repeating base on size of trajectory. "Will perform maneuvering" parameter defines if the UAV performs repulsing maneuver here, if false then point serves only as a waypoint.

Area message Area message shares same data format with standard trajectory message. Parameter maneuvering point is unused for area message as at the start is the sound system turned on and is shut down at end of this trajectory.

Siren test This message is only for turning sound system on and off. Whenever this message arrives, sound system status is changed.

8 Android application

Nowadays it is possible to see UAVs almost everywhere. They can be used for various tasks as filmmaking, searching for lost people or delivering a cargo. These tasks can be quite hard even for a skilled pilot. An advantage for the pilot is that he can do almost anything with proper equipment mounted to UAV. A pilot is not sufficient when you need really precise operations, also not every time it is possible to have such person in your company or hire one. Using autonomous UAV can erase these difficulties. If coded well, application for UAV give logical boundaries to the user (e.g. a user cannot threaten air traffic or can manage returning of the UAV with low battery) also such application can be used with basic knowledge about UAVs and does not require a trained operator.

This application aims to be user-friendly and easy to use with minimal previous experience with UAVs. The user can select previously define trajectory or create his own. On this trajectory, he can choose from 3 types of flight.

- Patroling inside an area
- Flying with given trajectory
- Flying computed trajectory

Patroling inside is done by back and forth motions which is implemented as a trapezoidal algorithm in P. Ješke's work [16]. After user-defined time repulsing maneuver is made. Flying with given trajectory is flying point after point and repulsing on them. Orienteering problem described in the chapter 4 takes user-defined points and tries to find a trajectory with limited time budget that has the highest rewards.

8.1 Used technologies

8.1.1 Android

The mobile part of this thesis is created for Android devices. With over 73% of Android users on the market, this app cover majority of the market. One of the basic philosophies of Android development is supporting different screen sizes with responsive design. To ensure that, relative layouts and dpi (density-independent pixel) was used instead of regular pixels. These techniques are sufficient for eliminating problems with responsivity and work well on small devices to big screens of tablets. On the other hand, larger screens often seem to be ineffective in using the given screen space. To make use of this space, fragments were used through creating the application. For smaller devices, there is one fragment per screen. However, for bigger screens these fragments can be combined together as shown in

figure 5. To make the application even more accessible, Czech and English localization using XML and Android resource system were added. More languages can be easily added. The basic functionalities of the application can be found at `https://youtu.be/GXst6LBxzro`.

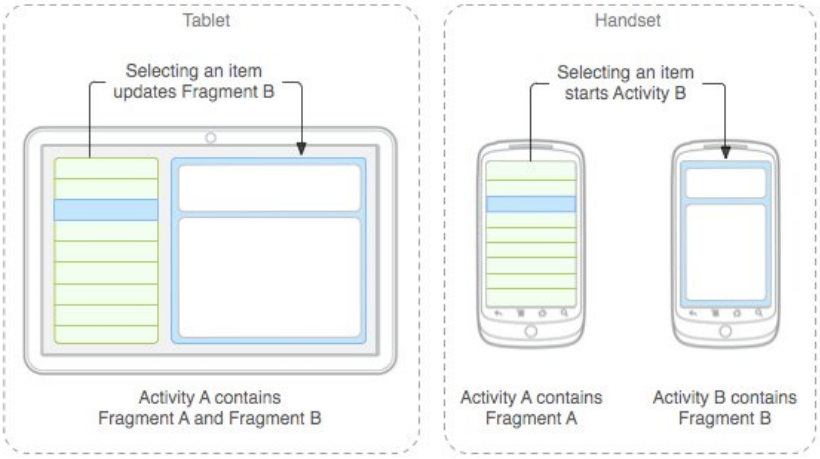


Figure 5: Example of fragments functionality. Two fragments are merged into one screen, but separeted for handset. [3]

8.1.2 SQLite

For purposes of this application, only basic functions of the database are needed. SQLite can satisfy these demands and also is a recommended option for Android developers. Zero initial configuration and no actions needed after crash make SQLite engine great option for storing valuable data on a mobile phone. Model of my database is shown in figure 6.

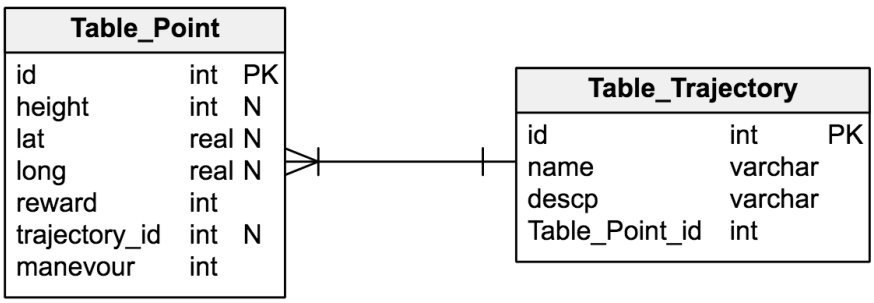


Figure 6: Database model to store important values about trajectories.

This allows a user to have multiple trajectories, which are saved after the application is closed. So it is possible for the operator to rerun his favorite trajectories or update ones that were not successful.

8.1.3 Google Maps

Most important part of the application is where a user is defining his desired trajectory. Google Maps has been chosen for this application, because of support, rich API and a significant level of editability. Typical input is a marker. Unfortunately, Google Maps marker cannot exist without map displayed, which is a considerable problem for persistent data application. It was necessary to encapsulate this marker with a custom class, so a user will not lose his work when switching screens. Google Places were used for quicker navigating on the map as the user is not limited only to scrolling, but can also type in the location where he wants to create a trajectory.

8.1.4 Ros

ROS is an open-source, meta-operating system for robots. It provides many services including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management[1]. One part of the ROS is running on the UAV itself. The second part is running on mobile. Ros is primely implemented in Python and $C++$, but after discussion with the colleague, experimental Java libraries were. Because of this it is possible to write the android application in the native language and use similar methods which were used in part of the application running on the UAV in $C++$.

8.2 User interface

One of the goals for the application was to create an easy to use and intuitive user interface. Google Material Design guidelines were followed to create such an interface. Material Design is a comprehensive guide for visual, motion, and interaction design across platforms and devices. Almost every UI element is described here. The description does not contain how to use this element programmatically but how to position it in view and how it should behave based on user interaction to create pleasant user experience.

8.2.1 Navigation

Navigation between the screens can be done in multiple ways. One way to create navigation is the main menu, which will be displayed as an initial screen. Then the user is able to pick the wanted action. After performing the selected action or pressing back button, the user returns to the main menu from where it is possible to continue. Another way for navigation is, creating a global menu which is same for all non-detail screens. For a small number of items, which is adviced to be up to five by Google Material Design, it is possible to create a bottom menu. Bottom menu is easily visible and can combine text and images for menu items. On the other hand, the bottom menu takes quite a considerable portion of the screen. With basic navigation bar and landscape orientation, it could take up to a quarter of the screen. Implementing this kind of navigation could be disorientating for the user. Also, the application now has five menu items, and that could lead to problems in future if another item would be added. From these possibilities side navigation bar, as seen in figure 7, was implemented. The bar is scrollable so almost unlimited amount of items can be added later. The user can bring side menu visible by taping on it in the navigation bar or by swiping right on the edge of the screen. To hide the menu swipe left, choose an item or click outside the menu area.

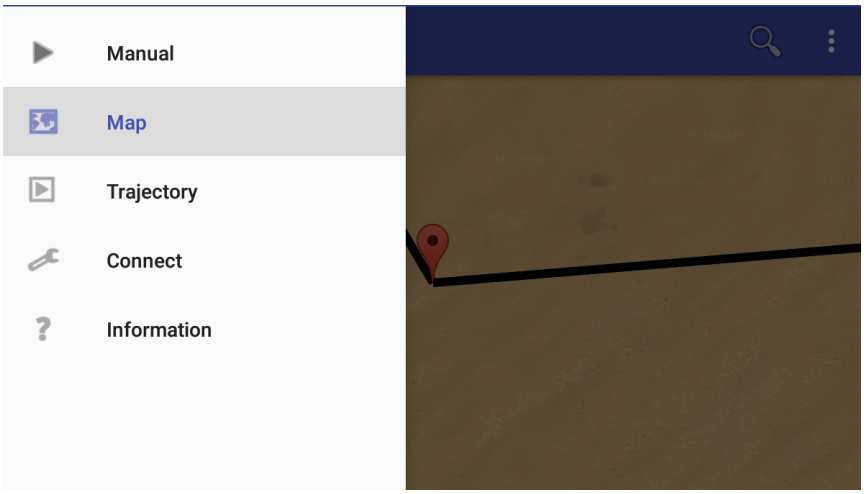


Figure 7: Application navigation done by side bar with five menu items.

As can been seen in figure 7, menu consists of five items.

- Manual control
- Google Maps
- Trajectory manager
- Connection info

- Information

8.2.2 Manual control

Manual control (figure 8) allows the user to take off with the UAV and in air control. UAV is handled by two joysticks unlike sport UAVs and helicopters, UAV is not controlled directly by yaw, pitch, roll, and thrust. As this type of control can be hard to master. Instead, left joystick serves for movement in the horizontal plane. Namely forward, backward, left and right. Right joystick pulled on sides serves as a control of an yaw. Altitude can be controlled by pulling right joystick up or down. For safety reasons descending is much slower than ascending.

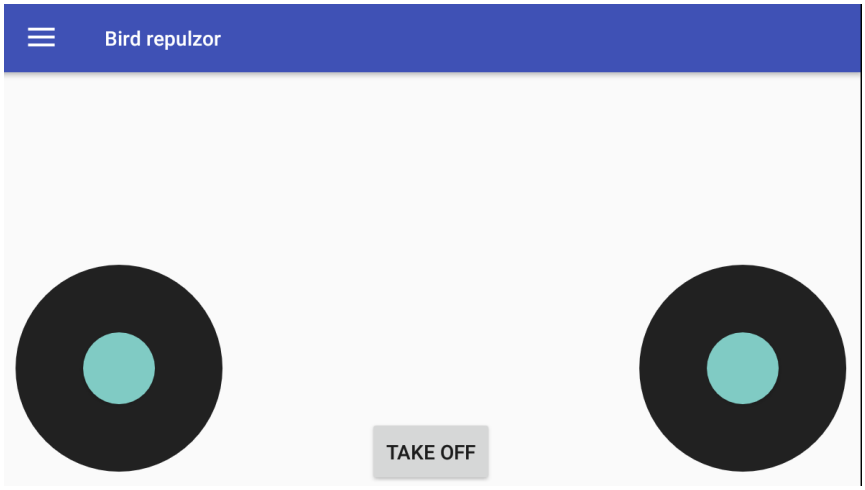


Figure 8: Manual control tab consisting of take off button and two joysticks.

8.2.3 Google Map

The map is where most of the user interaction is done. After loading the map, a position with trajectory will be displayed. If no trajectory exists, the map will be positioned on the user's location that is determined from the available information from his or hers mobile phone. As seen in chapter 9 navigation bar has a unique menu for this screen consisting of six items. Sixth item of menu the clue icon is hidden behind a drop-down menu.

Navigation on the map is well known for most of the users. Several Google Map gestures are available. Dragging for moving on the map, pinching for zooming the map and rotation for rotating the map. New points of trajectory can be added by clicking on the map. Long click on point makes that point draggable allowing repositioning on the map. Taping on point will bring up a detail screen, where it is possible to change whether

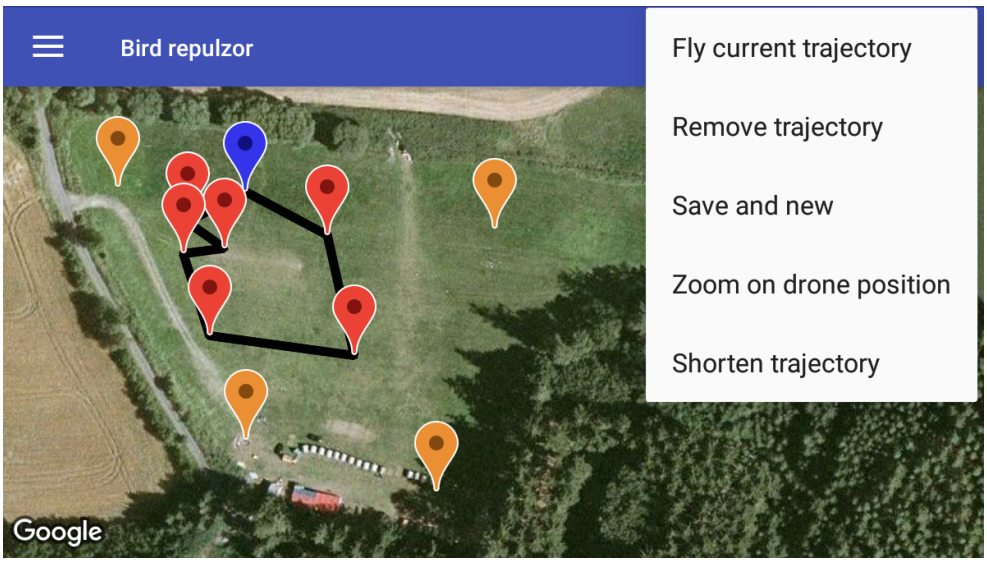


Figure 9: Google Map fragment with opened options menu on the right side. Trajectory defined by multiple red markers is visible in terrain map.

repulsing maneuver will be executed on the point, height, and reward. Also, the point can be removed here. In drop-down menu items remove trajectory, save trajectory, new trajectory and zoom on UAV position are self-explanatory. Shorten trajectory will change a sequence of points to create the shortest trajectory. After clicking on $flycurrenttrajectory$, the user can pick one of three flying modes. Fly trajectory for point by point flying. Fly the computed trajectory for a budget based flying and patrol inside for covering the whole area. Clue icon serves for searching and navigating to a different location. The command option shorten trajectory runs OP algorithm with an unlimited time budget.

8.2.4 Trajectory manager

Next screen, shown in figure 10, serves for managing trajectories. Every trajectory is displayed here on its own card. In the card, the user can select, edit or delete the respective trajectory. After selecting trajectory, a user is taken to map with that trajectory. Editing trajectory takes the user to detail screen where he can rename it, enter an additional description or set height for all points globally, so he does not have to change the height for each point manually. Lastly, the user is able to delete trajectory both on the card and when in detail. The new trajectory can be added by tapping ”plus” in the navigation bar.

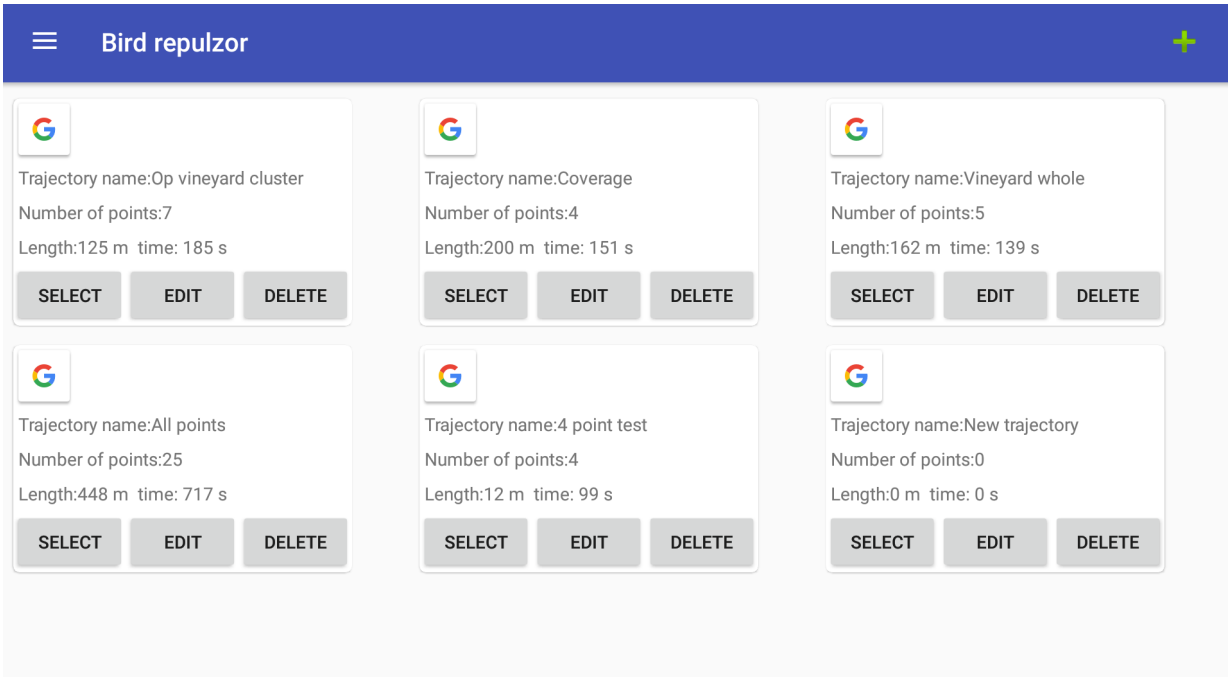


Figure 10: View for managing trajectories. Select button takes the user to Map fragment, edit button show editing popup and delete removes trajectory. Green "plus" adds new trajectory.

9 Experiments

To verify the functionality of complete proposed solution, three experiments were performed. These experiments test communication between drone and mobile phone, sound system performance and usability of computed path. The first experiment was performed to verify these functionalities, other two experiments were created to match real-life scenarios. All the flights were controlled by mobile application from take-off. Trajectories and their parameters were also created in the application. More tests were performed but are not described here, as they were more of prerequisites to these tests. Namely, connection, correct sending and receiving messages and manual control. Through the chapter, there are orange and blue points in figures. Orange points are safe to fly area boundaries, and blue marker is the starting point.

9.1 Basic functionality test

The first experiment tests basic functions of the proposed system. These functions are fly to the position, change flight height and perform maneuver on designated point with the sound system on. Flight plan as was seen in the application is in figure 11. Concrete values which were sent to drone are in table 1. As it can be seen in figures 12,13 and 14, UAV can perform all actions. Repulsing maneuver can be recognized by the change of height by ten meters. During the repulsing, sounds system was turned on.

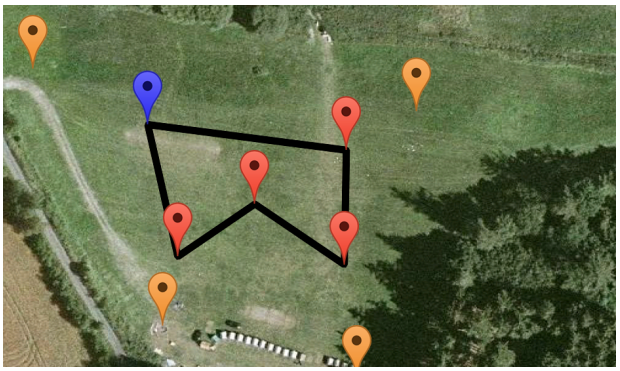


Figure 11: Croped view from Android application of first experiment

Index	UTM x	UTM y	Height	Maneuver
1	-24.2	36.8	4	yes
2	37.7	29	8	yes
3	36.8	-8.2	6	yes
4	9.8	11.2	10	no
5	-5.3	5.8	4	yes

Table 1: First experiment important values

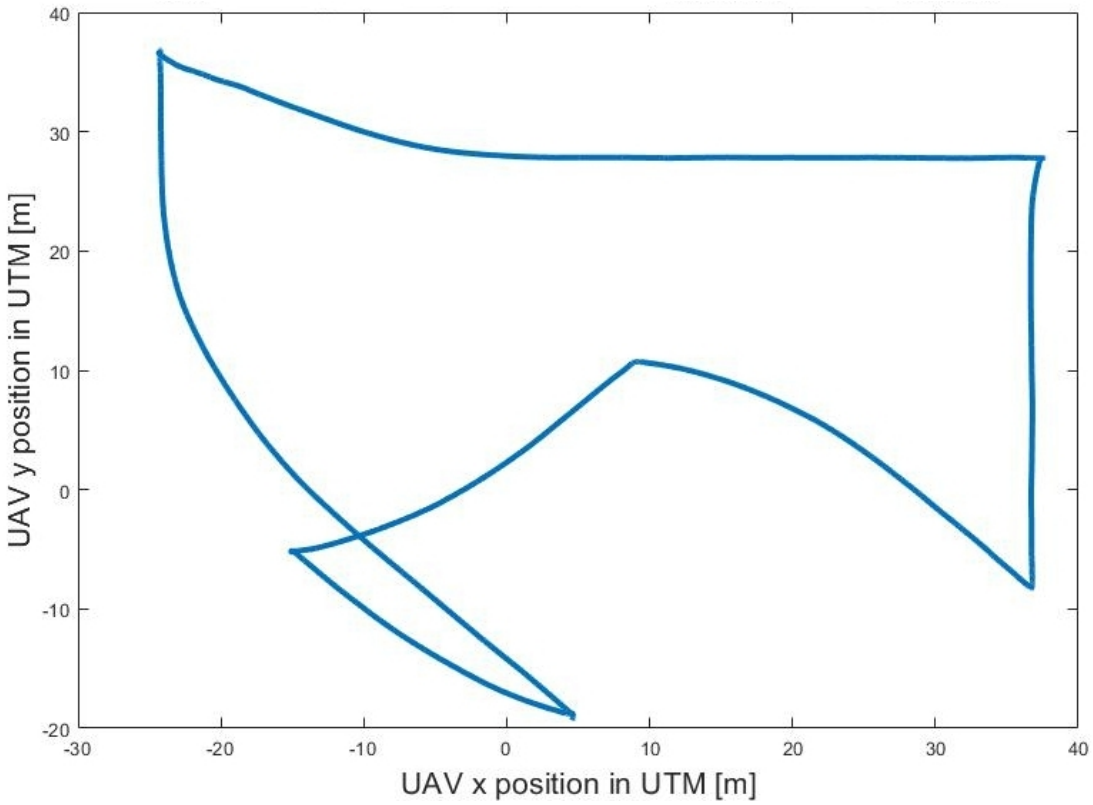


Figure 12: Flight path of basic functionality test from the bird perspective ploted in Matblab

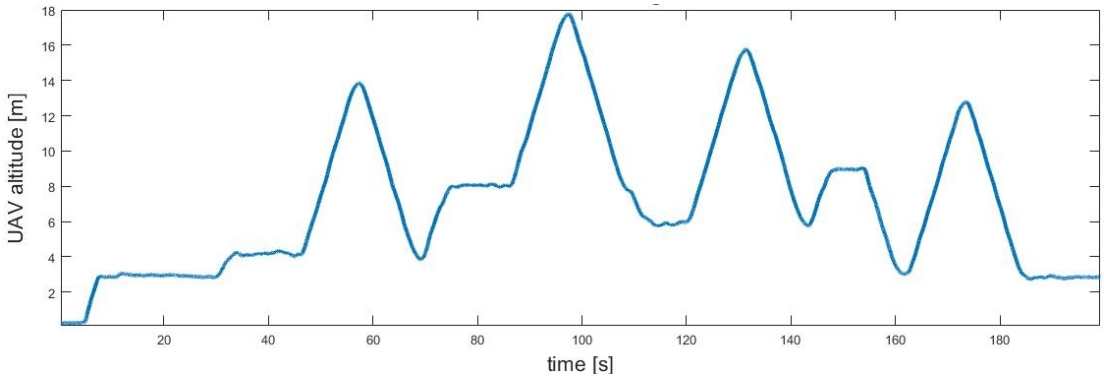


Figure 13: Change of altitude during flight of basic functionality test

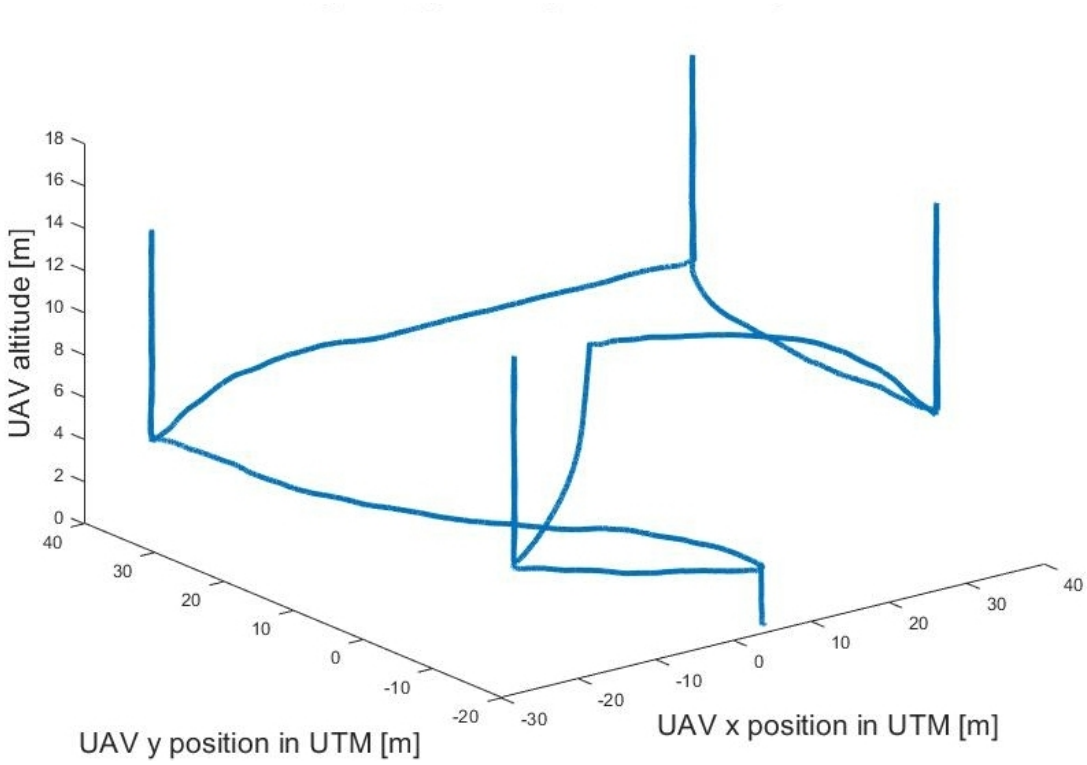


Figure 14: Trajectory of basic functionality test plotted in space

9.2 Testing coverage algorithm

This experiment is testing area coverage by the trapezoidal algorithm. Four points in figure 15 are bordering points of theoretical vineyard rather than the exact position of birds flock. In figure 16 bird perspective of this algorithm can be seen, as the height is remaining same through the flight. Flight plan values are supplied in table 2. For completeness figure of altitude change is in figure 17 and trajectory plotted in space is in the figure. 18. In figure 15 can be seen that the trajectory did cover only the middle of the selected area. This is caused by setting higher speed than the UAV is capable of, resulting in a wrong sampling of trajectory. Because of that, the final flown trajectory is taking "shortcuts" to satisfy wanted trajectory at least partially.

Figure 15: Croped view from Android application of area coverage experiment. Area covered is bordered by black lines.

Index	UTM x	UTM y	Height
1	-24.2	36.8	8
2	37.7	23	8
3	36.8	-8.2	8
4	-14.2	-14.2	8

Table 2: Area coverage flight plan values

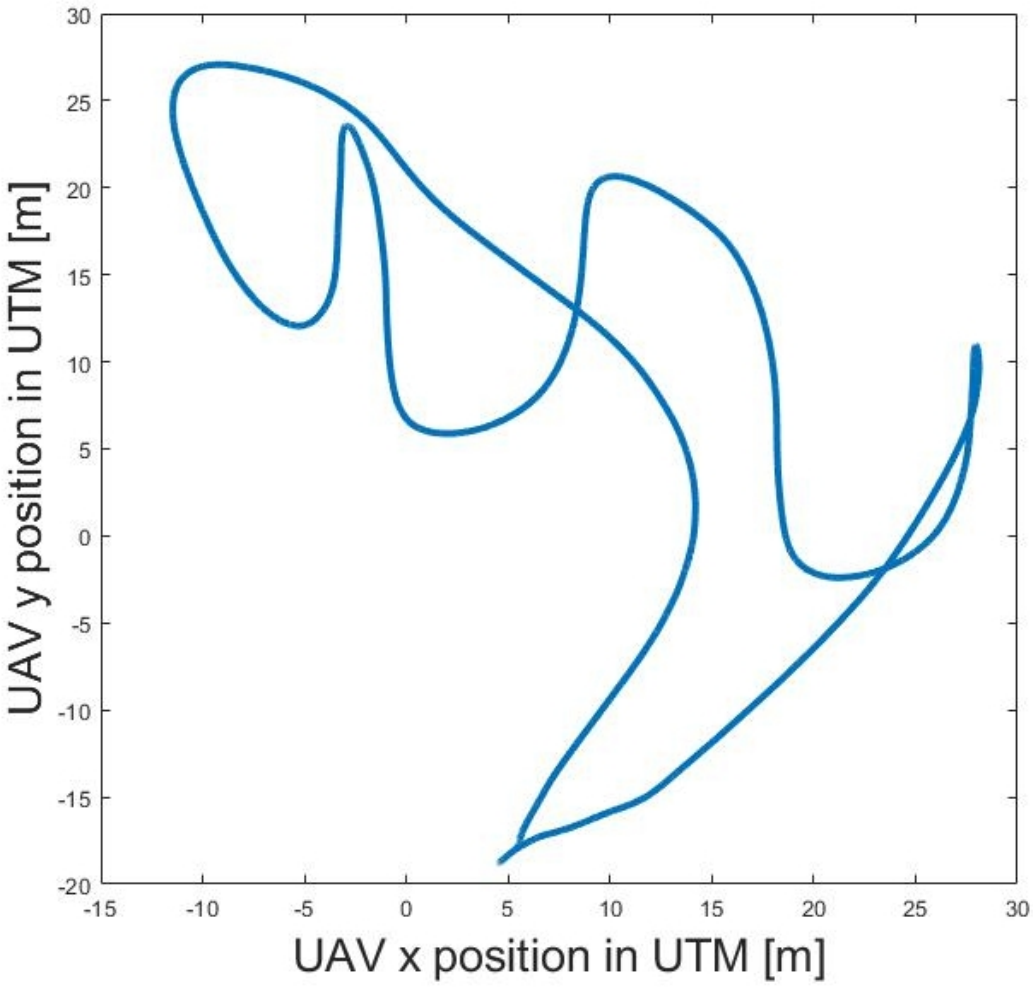


Figure 16: Trajectory from the bird perspective during the trapezoidal algorithm

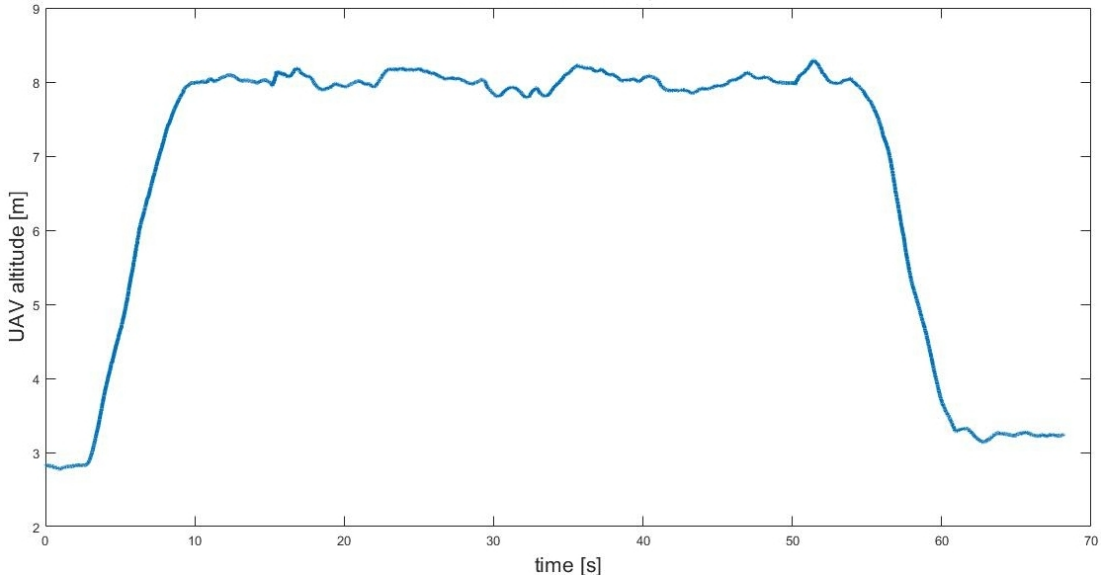


Figure 17: Change of altitude during flight of the trapezoidal algorithm

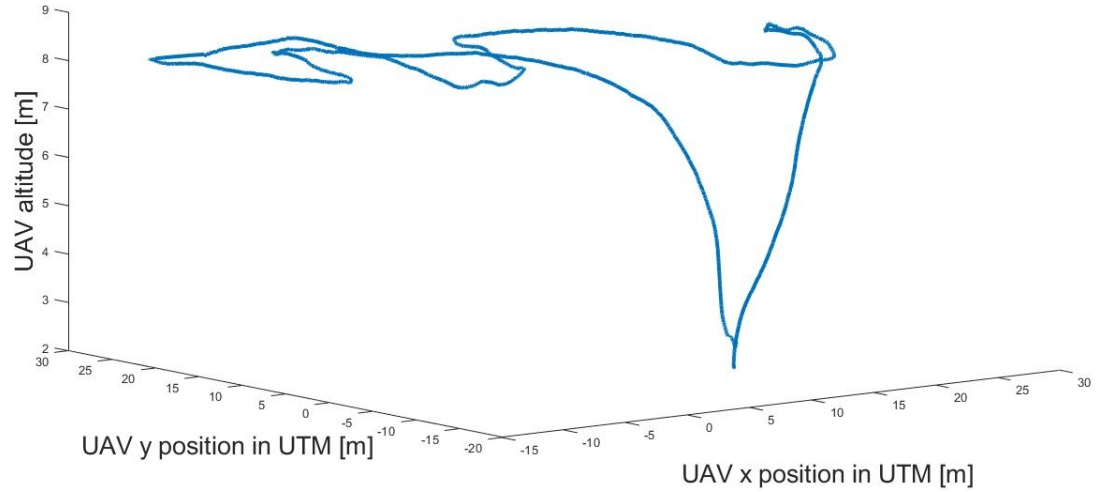


Figure 18: Trajectory of the trapezoidal algorithm ploted in space

9.3 Testing orienteering algorithm

In this scenario, four vineyards with different types of grapes and thus rewards are created. Each vineyard is represented by four points. The UAV starting position is in the middle. Situation before computation is in figure 19. Points chosen by the algorithm are in figure 20. The algorithm correctly chose higher reward points and used almost whole time budget (202/220). So no more points could not be added as repulsing takes twenty seconds. The algorithm did collect maximal reward possible (16 reward points were collected within the time budget of 220 seconds) but did not find the best path as there are two possibilities of a shorter path with the same reward. Whole algorithm input is shown in table 3.

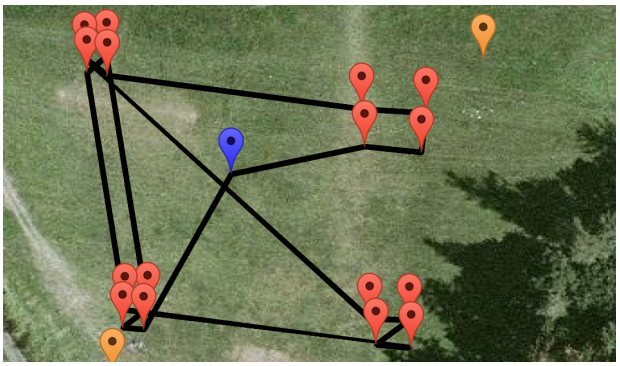


Figure 19: Flight plan as seen in the Android application before computation

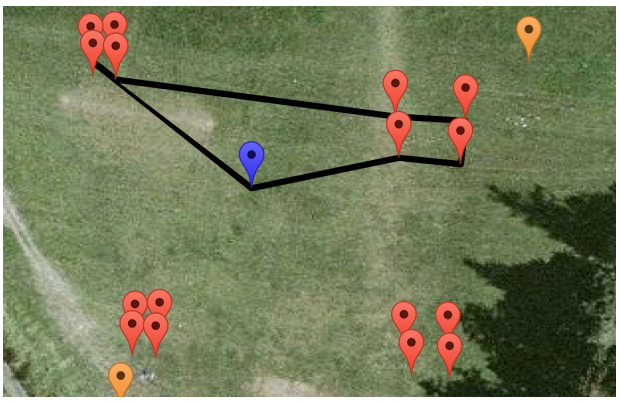


Figure 20: Flight plan after computation in algorithm presented in this thesis. Points joined by lines were chosen by algorithm.

Index	UTM x	UTM y	Height	Reward
1*	-25	41	4	2
2	-22	41	4	2
3	-25	38	4	2
4*	-21	37	4	2
5	30.8	-16.2	4	1
6	36.8	-6.2	4	1
7	30.8	-10.2	4	1
8	36.8	-10.2	4	1
9*	33.8	29.8	4	3
10*	47.8	28.8	4	3
11*	46.8	19.8	4	3
12*	46.7	19.8	4	3
13	-15.2	-3.2	4	1
14	-14.2	-7.2	4	1
15	-14.2	-10.2	4	1
16	-14.2	-14.2	4	1

Table 3: Flight plan values of orienteering algorithm. Values marked with * were chosen by algorithm.

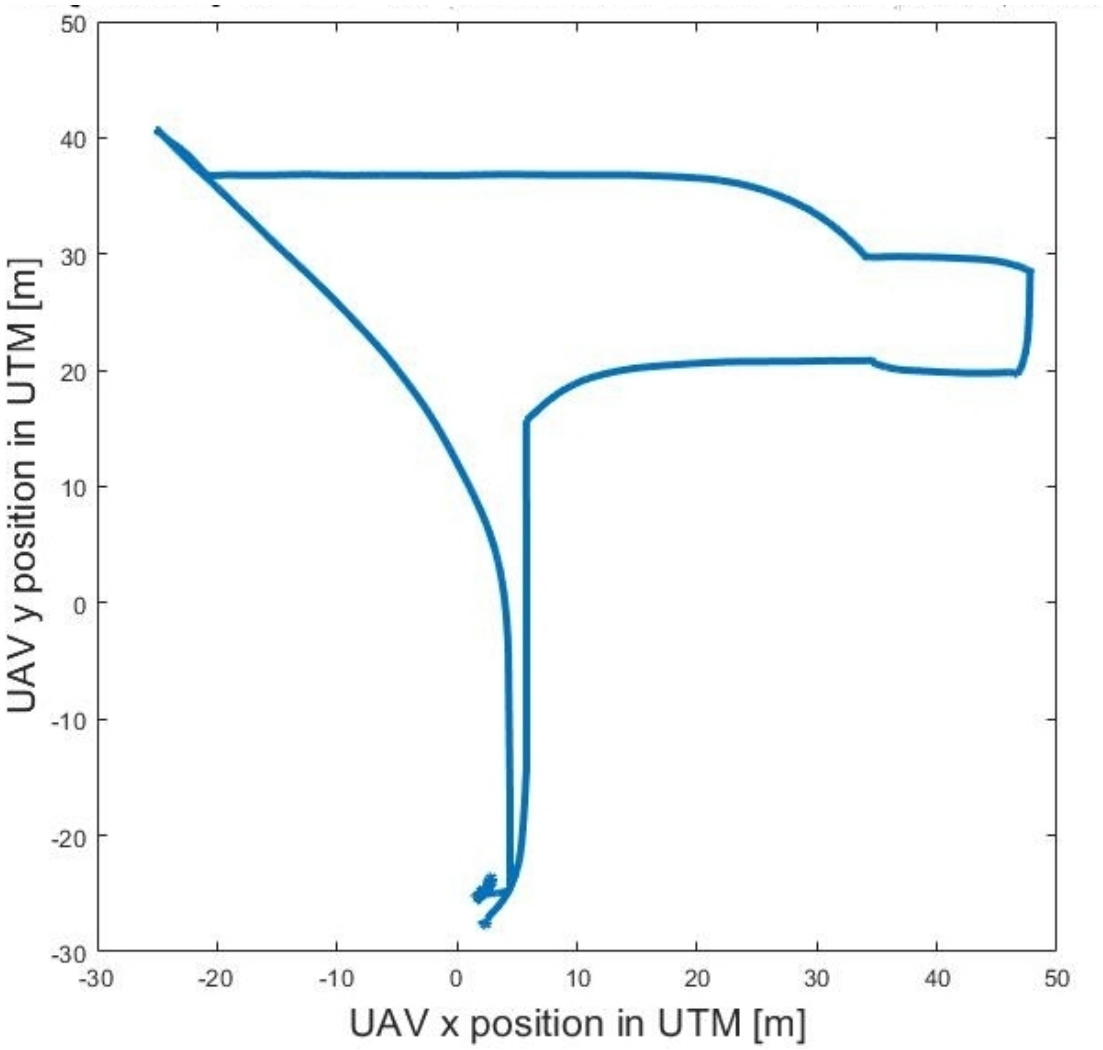


Figure 21: Trajectory from the bird perspective during the test flight of OP algorithm

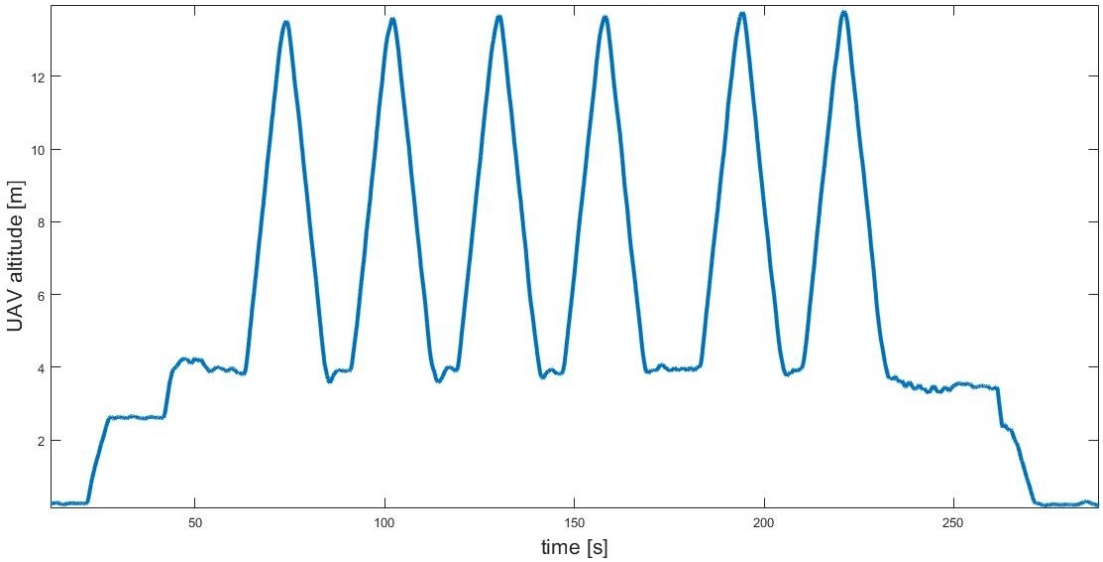


Figure 22: Change of altitude during flight of the OP algorithm experiment

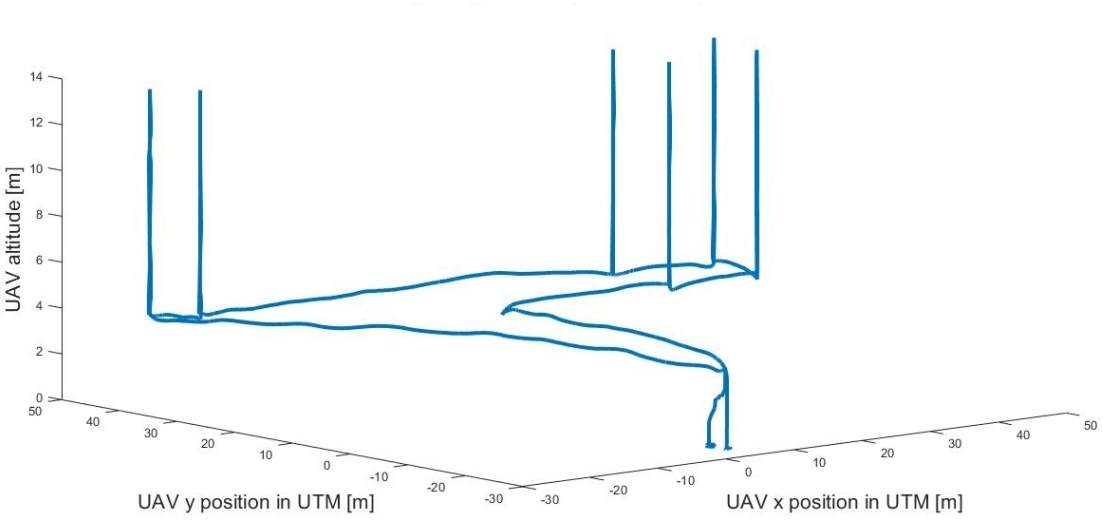


Figure 23: Trajectory of the computed path by path planning algorithm ploted in space

9.4 Testing sound system capabilities

To select the best solution for the sound system, multiple tests were performed by sound level meter *testo*815. This device allows two mods of filtering. Frequency filter A, detach low frequency human inaudible noises from the measurement. Frequency filter C even the low-frequency noises. Both types of measurement were carried as the birds have wider range audible frequency than the humans. The siren was powered with 12V during the test and the speaker was playing an 800Hz Sawtooth tone. The speaker was louder than the propellers while playing continues tone. Playing predator noises on the speaker was not recognizable on measured power value. The siren has highest power values a thus is selected option for bird repulsing.

Distance [m]	ESP siren		Speaker		UAV Proppelers	
Filter modes	(A)	(C)	(A)	(C)	(A)	(C)
1	105	115	93	97	82	87
2	101	107	88	89	77	84
5	88	98	83	85	73	78
10	82	86	78	81	68	73
15	78	82	74	76	65	69
20	75	76	67	71	60	66
Enviroment	51	71	53	68	53	68

Table 4: Power values [dB] messuered by *testo*815 to verify usability of sound system. Two power values are shown here, each for different type of filtering (A and C)

10 Conclusion

The solution for bird repulsion at vineyards by the UAV is presented in this thesis. At first, an Android application was created. The created application allows the user to quickly create trajectories for the UAV, without the knowledge of the ROS system running on it. It is possible to control the sound system directly from the application, or by selecting one of the flight modes. Trajectories created in the application are saved on the mobile phone and are editable with a possibility of adding additional parameters based on the future requirements of users.

The sound system was at first implemented with the speaker. Testing the speaker revealed that the emitted sound is inaudible in comparison with the noise emitted by the UAV's propellers. After these power tests, the sound system was reimplemented with piezo sirens. These sirens are easy to use in the application or by calling ROS services.

The path planning algorithm is based on the Orienteering problem and solved with a stochastic approach using the Monte Carlo technique. This algorithm was used to find a feasible path for the UAV. Furthermore, usage of the sound system with regard to the distance and the power was computed. Functionalities of proposed application were at first tested in Gazebo. Real platform experiments were described in this thesis. These tests verified the possibility of using the path planning algorithm proposed in this thesis for bird repulsion and also verified the functionality of the whole solution.

For the future, much more work could be done on the application and the sound system. Implemented path planning algorithm could be substituted by a more advanced a GSOA. The logic for using GSOA is already described in this thesis, however, it has not been used for computing the trajectories. Also, it is possible that the sound system and repulsing maneuver will require modification after a research carried by Czech University of life sciences.

References

[1] Ros/introduction, 2014. Available at http://wiki.ros.org/ROS/Introduction.

[2] Walmart testing warehouse drones to catalog and manage inventory, 2016. Available at *https* : *//www.supplychain*247*.com/article /walmart_testing_warehouse_drones_to_manage_inventory*.

[3] An example of how two ui modules defined by fragments can be combined into one activity for a tablet design, but separated for a handset design., 2018. Available at https://developer.android.com/guide/components/fragments.

[4] Mav used by multi-robot systems group, 2018. Available at *http* : *//mrs. felk.cvut.cz/images/images/main_page_icons/single_MAV_hexa_compressed.jpg*.

[5] Micro Aerial Vehicle - MRS CVUT. 2018. Available at http://mrs.felk.cvut.cz/research/micro-aerialvehicles.

[6] Soundair speaker, 2018.
Available at *https* : *//iczc.cz/cuogtvukisgubb*01*o*0*cag*07*l*52_4*/obrazek*.

[7] Drones for search & rescue missions. *Altigator.com*, c2018. Available at https://altigator.com/drones-for-search-rescue-missions.

[8] G. Astuti, G. Giudice, D. Longo, C. D. Melita, G. Muscato, and A. Orlando. An overview of the "volcan project": An uas for exploration of volcanic environments. *Journal of Intelligent and Robotic Systems*, 54(1):471–494, Mar 2009.

[9] I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475 – 489, 1996.

[10] J. Chudoba, M. Saska, T. Baca, and L. Preucil. Localization and stabilization of micro aerial vehicles based on visual features tracking. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014.

[11] L. E. Dubins. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79(3):497+, July 1957.

[12] J. Faigl. Data collection path planning with spatially correlated measurements using growing self-organizing array, 2018.

[13] Jan Faigl and Robert Pěnička. On close enough orienteering problem with dubins vehicle. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5646–5652, 2017.

[14] Shripad Gade, Aditya A. Paranjape, and Soon Jo Chung. Herding a flock of birds approaching an airport using an unmanned aerial vehicle. In *AIAA Guidance, Navigation, and Control Conference 2015, MGNC 2015 - Held at the AIAA SciTech Forum 2015*. American Institute of Aeronautics and Astronautics Inc., 2015.

[15] Bruce Golden, Arjang Assad, and Roy Dahl. Analysis of a large scale vehicle routing problem with an inventory component. *Large scale systems*, 7(2-3):181–190, 1984.

[16] P. Ješke. Autonomous helicopter control by a mobile phone with android for precision agriculture, 2018.

[17] Gorka Kobeaga, María Merino, and Jose A. Lozano. An efficient evolutionary algorithm for the orienteering problem. *Computers & Operations Research*, 90:42 – 59, 2018.

[18] D. Kohler and K. Conley. rosjava–an implementation of ros in pure java with android. 2011. Available at https://github.com/rosjava/rosjava_core.

[19] Golden Bruce L., Levy Larry, and Vohra Rakesh. The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318, 1987.

[20] G. Laporte, F. Semet, V V Dadeshidze, and L J Olsson. A tiling and routing heuristic for the screening of cytological samples. *Journal of the Operational Research Society*, 49(12):1233–1238, Dec 1998.

[21] Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2):193 – 207, 1990.

[22] S. Lee, K. E. An, B. D. Jeon, K. Y. Cho, S. J. Lee, and D. Seo. Detecting faulty solar panels based on thermal image processing. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–2, Jan 2018.

[23] Mathieu Lihoreau, Tamara Gómez-Moracho, and Cristian Pasquaretta. *Traveling Salesman*, pages 1–4. Springer International Publishing, Cham, 2017.

[24] A. S. Lomax, W. Corso, and J. F. Etro. Employing unmanned aerial vehicles (uavs) as an element of the integrated ocean observing system. *Proceedings of OCEANS 2005 MTS/IEEE*, pages 184–190 Vol. 1, Sept 2005.

[25] Aaron Martinez and Enrique Fernández. *Learning ROS for Robotics Programming*. Packt Publishing Ltd., 2013, birmingham edition, 2013.

[26] Andrew Meola. Shop online and get your items delivery by a drone delivery service: The future amazon and domino's have envisioned for us, 2017. Available at http://www.businessinsider.com/delivery-drones-market-service-2017-7.

[27] Jason M. O'Kane. *A Gentle Introduction to ROS*. O'Kane, Columbia, 2014 edition, 2014.

[28] Robert Pěnička, Jan Faigl, Petr Váňa, and Martin Saska. Dubins orienteering problem. *IEEE Robotics and Automation Letters*, 2(2):1210–1217, April 2017.

[29] Robert Pěnička, Jan Faigl, Petr Váňa, and Martin Saska. Dubins orienteering problem with neighborhoods. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1555–1562, June 2017.

[30] Robert Pěnička, Martin Saska, Christophe Reymann, and Simon Lacroix. Reactive dubins traveling salesman problem for replanning of information gathering by uavs. In *European Conference of Mobile Robotics (ECMR)*, pages 259–264, 2017.

[31] Robert Pěnička. Motion planning for seabed monitoring by autonomous underwater vehicles, 2016.

[32] R. Ramesh and Kathleen M. Brown. An efficient four-phase heuristic for the generalized orienteering problem. *Computers & Operations Research*, 18(2):151 – 165, 1991.

[33] M. Saska, V. Kratky, V. Spurny, and T. Baca. Documentation of dark areas of large historical buildings by a formation of unmanned aerial vehicles using model predictive control. In *IEEE ETFA*, 2017.

[34] Wouter Souffriau, Pieter Vansteenwegen, Joris Vertommen, Greet Vanden Berghe, and Dirk Van Oudheusden. A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence*, 22(10):964–985, 2008.

[35] T. Thomadsen and T. Stidsen. The quadratic selective travelling salesman problem. 2003.

[36] T. Tsiligirides. Heuristic methods applied to orienteering. *The Journal of the Operational Research Society*, 35(9):797–809, 1984.

[37] C. Verbeeck, K. Sörensen, E.-H. Aghezzaf, and P. Vansteenwegen. A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research*, 236(2):419 – 432, 2014.

[38] J. Yu, M. Schwager, and D. Rus. Correlated orienteering problem and its application to persistent monitoring tasks. *IEEE Transactions on Robotics*, 32(5):1106–1118, Oct 2016.

Appendix A CD Content

In Table 5 are listed names of all root directories on CD.

Directory name	**Description**
kloucste.pdf	the thesis in pdf format
thesis_sources	latex source codes
android_sources	android application source files
android application	genereted apk and how to install readme
communication_sources	source files for comunication running on drone

Table 5: CD Content

Appendix B List of abbreviations

In Table 6 are listed abbreviations used in this thesis.

Abbreviation	Meaning
API	Application programming interface
GPS	Global Positioning System
UAV	Unmanned Aerial Vehicle
MAV	Micro-Air Vehicle
VTOL	Vertical Take-Off and Landing
OP	Orienteering problem
CEOP	Close enough orienteering problem
TSP	Travelling salesman problem
GSOA	Growing Self-Organizing Array
UI	User interface

Table 6: List of abbreviations