TU/e
Eindhoven, April 22, 2016

USE assignment: Delivery Drone

OLAUK0
Quartile 3, 2015-2016

Case coordinator:	dr. ir. M.J.G. Molengraft	
Scientific panel:	dr.ir. R.H. Cuijpers	
	mr.dr.ir.ir. L.M.M. Royakkers	
	dr. V.A.J. Borghuis	
	dr.ir. E.I. Barakova	
Group 2:	S. van der Loo	0813076
	E. van de Put	0897289
	J. Setz	0843356
	M. Tibboel	0909136
	M.G. Vlaswinkel	0899061
	M. Zararsiz	0865084

Contents

List of symbols

Symbol	**Quantity**	**Unit**	**Unit Abbreviation**
α	Angle	Degrees	$[°]$
A	Area	Cubic Meters	$[m^2]$
h	Height	Meters	$[m]$
l	Length	Meters	$[m]$
m	Mass	Gram	$[g]$
p	Abstract position	-	$[-]$
v	Speed	Meters per second	$[m/s]$
t	Time	Minutes	$[min]$
$\vec{x}$	Position vector	Meters	$[m]$

1 Introduction

Drones can take goods that are ordered from the storage to somebodies house. The delivery is quick and orders can be handled a lot faster. This way, delivery companies can handle more orders at a higher rate. But people also get the items they have ordered really quick, which helps the satisfaction.

Autonomous flying is no longer considered the main problem. Some companies already want to carry out their autonomous drone in the near future [19]. Those companies however, are using less urban areas for testing and we are curious about how their drones are going to hold in busy cities. Drones need to be more reliable, they still have a tendency to crash and run into objects. Some experiments, for example from MIT, are getting better at avoiding object autonomously [5], but this problem is still big because of unexpected events that asks the drone to react very quick. Energy consumptions could also be a problem [23]. For long flights the drones need large batteries, but bigger batteries also means less space for cargo.

Another problem with delivery drones these days is the "problem of the last meters". These last problems aren't so much about the technology, since most of it already exists, but how to implement all these technologies to make it actually work. These problems are for example: how do we deliver packages in apartment buildings and how do people (and animals) react to these kind of deliveries [14]. The article from the Washington Post gives a great start to start asking questions which aren't technical, but more to the side of users. What do we want as society?

For technical problems, like the battery-life problem as given above for example, often many different technical solutions are quickly presented, like changing the battery of the drone in midair until better battery can be made [13]. Solving user problems however, often requires a different kind of thinking. The concept Amazon Prime Air is the best example of todays drone delivery progress [2], which is (unfortunately) still a concept and not working yet. Also Google [4] and Walmart [15] are joining the competition to get the first working delivery drones ready. These three competitors all want to be the first company that can use the drones, which means that a lot of research (and money) is involved. The problem those companies are working on is the reliability of the drones.

Our idea is to look at drones and find out what the best human interaction is when they want to land. Also the drone should be able to find the right landing spot by itself. In further investigation the human interaction can be related to this landing procedure so the drone is able to land on the right spot in a comfortable way for the customer.

The next section (section 2) is about the focus of this paper. The requirements, the state of the art and the detailed focus are explained. After that, in section 3, the User, Society and Enterprise (USE) aspects are written down. In section 4 the way a drone finds it's landing spot is explained. Section 5 tells what the right approach is for the user. At last, the conclusion (section 6) and the discussion (section 7) are written down.

2 Focus

The focus of this paper is explained in this chapter. First, the requirements of the drone are given, so that it is clear what the drone should be able to do. After that the state of the art will be explained. Previous research, done in the second quartile of the year 2014/2015 by group 1 [3] and other companies, answered some of the questions that are related to this subject. They investigated the way of navigating and verifying. Their finding's, and some background information, will be discussed briefly in the sections below. After that the detailed focus of this paper is explained. The research that will be done in this paper is most likely going to be combined with this previous research.

2.1 Requirements

The requirements are given and explained to tell what the drone should be able to do. Small pieces of technology can be investigated, before getting to the final product.

- The drone can make a flight plan
  - The drone knows from the map where it is able to fly (course flight)
  - The drone can set a landing area for itself from a given destination
  - The drone knows what to do if the destination can't be reached
- Flying
  - The drone follows a trajectory path which is comfortable for it's surroundings
  - The drone can stabilize itself during flight
- Landing and take off
  - The drone can decided what a good landing spot is within a landing area
    - The drone can make a detailed and up to date map of the landing area
    - There is enough free space for the drone to land in the landing area
    - The drone lands on a comfortable distance of the user
    - The drone knows what to do if the local destination can't be reached
  - The drone is able to land autonomously (fine navigation)
    - The drone is able to evade inanimate objects
    - The costumer or drone won't be endangered when something doesn't go as planned, for example when the landing spot is uneven
- Localization and navigation
  - Where is the drone flying and to what destination (course navigation)
    - The drone can access a map and knows where it is on this map
- Communication and verification
  - The drone can hand over the package
  - The drone can verify that the package is at the right person
  - The drone knows what to do with emergencies
    - There is a function for the user to call the home depot
  - The drone knows how to contact the user in a good way
- Safety of the drone
  - The drone is legislated
  - The drone can and knows how to react to different weather conditions
  - The drone has procedures for safety
    - Against stealing
    - For avoiding (flying) objects
    - For something unexpected happening
  - The drone has contact with the home depot at any time
    - The connection is protected
    - The home depot can track and adjust the drone's path

Putting these requirements in a flowchart, gives a good overview and what parts are connected to other parts.

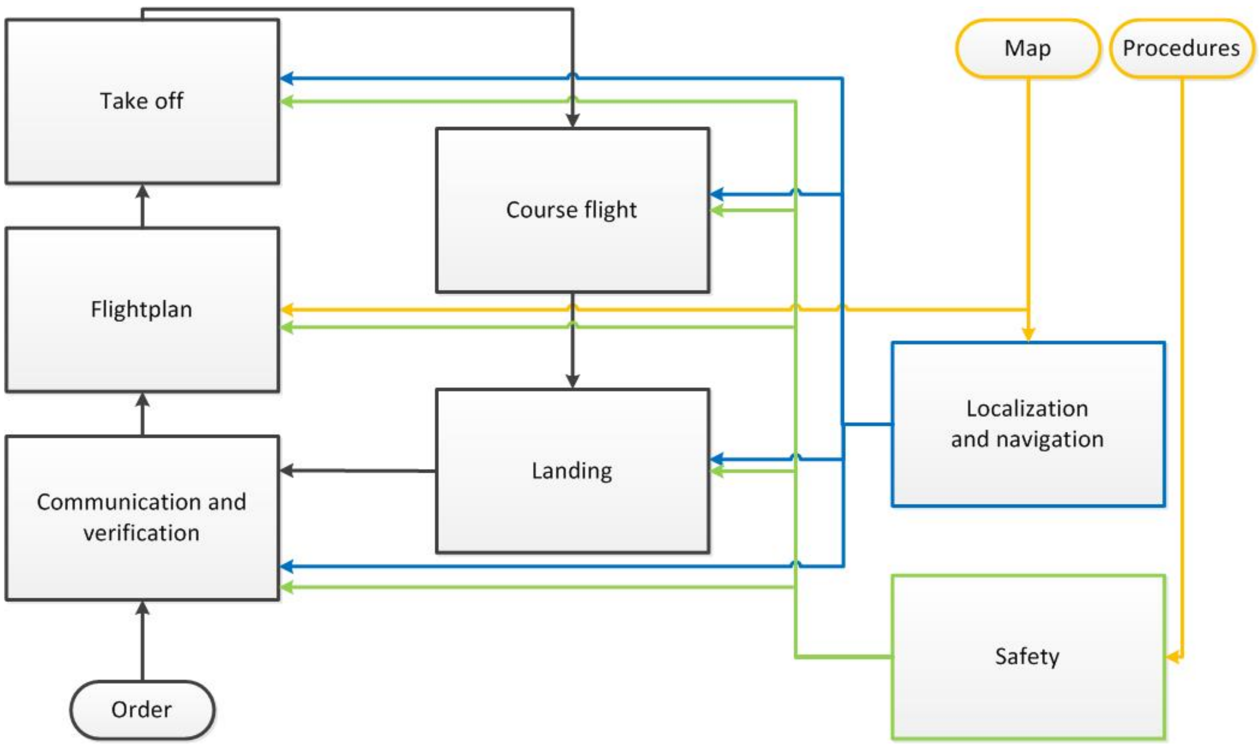


Figure 2.1: Flowchart

2.2 State of the art

The state of the art is the research that is already done. A lot of research is going on in the drone industry. A couple of big companies are trying to be the first to make drone delivery a real thing.

Being at the brink of being introduced, drone delivery still encounters many problems. Most of these problems are not in autonomous flying however, but mainly in the implementation of these technologies. The first companies are already using less urban areas as their test-grounds. The concept is working, but remains unreliable. The drones still have the tendency to crash into objects and the endless list of unexpected events that can possibly happen ask for a high reaction speed. Also the limited battery lifetime appears to pose problems. Below Amazon's Prime Air is described, as the leader in the current developments, followed by more general current developments and points that still are being researched on.

2.2.1 Amazon Prime Air

The leader in the current developments is the Amazon Prime Air concept from Amazon. [2] Unfortunately the drones are not ready to be embedded in society, but successful flights already have been made. One of the drones they are using is small plane that has the possibility of vertical takeoff. Flying the drone as a plane increases the speed and range of the drone, resulting in a delivery time of 30 minutes or less and a range of 24 kilometers. As the drone approaches its destination, a message is send to notify the customer that the package is arriving. The drone recognizes its destination by a big 'A' that is placed on the ground. Before lowering itself, the landing area is scanned first for obstacles. After the landing, a small valve is opened releasing the package. A second drone that is being developed is a quad-copter. This drone has a range of about 14 miles and can carry packages of about 2.5 kg. This may not seem much, be around 80% of the packages delivered by Amazon falls within this weight margin, making the likelihood of this technology being introduced in the recent future rather high. Hopes are, that it is possible to make this around 2020. [1,6]

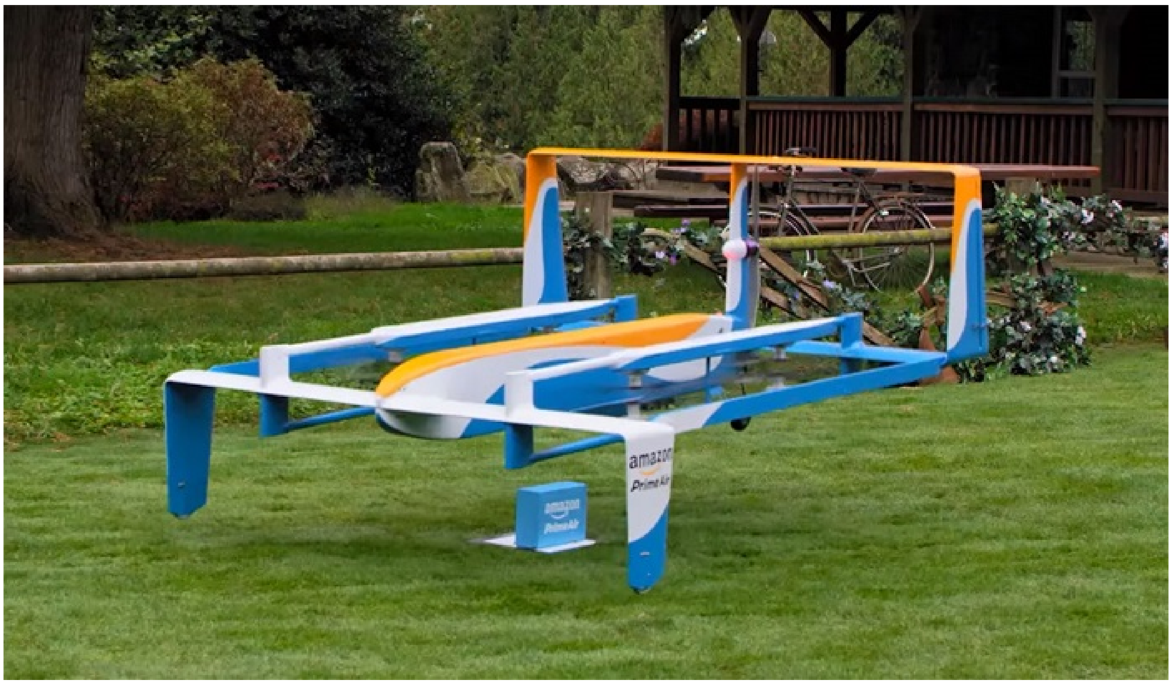


Figure 2.2: Amazon prime

Also Google [4] and Walmart [15] are joining the competition to get the first working delivery drones ready. These three competitors all want to be the first company that can use the drones, which means that a lot of research (and money) is involved. The problem those companies are working on is the reliability of the drones.

2.2.2 Related USE project

Project Group 1, Project Robots Everywhere year 2014/2015 [3], worked on a similar project regarding delivery drones. They have developed autonomous localization and flying. They did this with the Parrot AR Drone 2.0 using a Flight Recorder, an Arduino and a WiFi shield. With this the drone could fly to a target GPS coordinate. Then using and NFC shield, the NFC tag in the phone of the customer to which the package is brought would be verified. When verification is successful, a red LED simulates the flap opening, releasing the package.

2.2.3 Can and cannot

Autonomous flying is not the main problem of drone delivery anymore, some companies already want to carry out their autonomous drone in the near future. [19] Those companies however, are using less urban areas for testing, but will not hold in busier areas. Drones need to be more reliable, they still have a tendency to crash and run into objects. Because of this, the laws of drones are not yet adjusted to legislate drone-delivery. See Appendix G for the current laws. Some experiments, for example from MIT, are getting better at avoiding object autonomously, but this problem is still on hand because of unexpected events that asks the drone to react very quick. How for example to cope with moving objects like people, animals (an enthusiastic dog for example) or flying objects (a football for example). Simply stuffing the drones with numberless sensors would drive up the price drastically, hopes are to be able to develop smart software that only uses few and or cheaper sensors to make the drones more attractive on the market.

A second problem is the limited battery-life of the drones. Increasing the size of the battery would reduce the loading capacity, but a small battery again will drastically decrease the range of the drone. This would require many distribution centers, which on their turn need to be supplied as well. An alternative however is given by K. Fujii, proposing to have the battery replaced in midair [14].

Another problem with delivery drones these days is the problem of the last meters. These last problems are not so much about the technology, since most of it already exists, but how to implement all these technologies to make it actually work. These problems are for example: how do we deliver packages in (high) apartment buildings and how do people (and animals) react to these kind of deliveries [13]. The article from the Washington Post gives a great start to start asking questions which are not technical, but more to the side of users. What do we want as society? Areas where relatively less research is done, compared to the more technical side. This however is exactly what this project about, making a next step in the development of user-friendly or rather user-centered drones.

2.3 Detailed focus

As seen at the requirements above, not all the subjects can be tackled at the same time and some of them are already done by others. This project is focused on the last few meters and even more on only two parts. The subsections below explain the focus on the landing spot and the user approach in particular.

2.3.1 Finding landing spot

One of the problems found during the delivery process done by drone is finding where the drone can land.

Landing the drone is thought to be easy, simply ascent till you hit the ground and that's it. Reality however proves to be much complexer [10] [8] [11]. Questions arise like; how does the drone avoid objects on the ground? And how does he find the right spot to land? One way would be to have the customer find an open space and put a printed "A" on the ground like Amazon's Prime Air [2]. A disadvantage is that the responsibility then is given to the customer. Should the drone instead find his own way to find the right spot?

Giving customers a lot of responsibility for the landing of the drone, proves not to be a good idea. The company that uses the drone is responsible for the package until the customer verified that the package is in their hands. This way it gives a great risk for the company for using a printed label like Amazon wants to do. Bringing this back to this project, an alternative would be to measure the height of the drone relative to the surface below, converting this into a height map. If there is enough flat space on the map for the drone to fit, it can land there. For this a safety margin can be put for an extra 10 centimeters on the sides of the drone.

2.3.2 Approaching the user

So far little research has been done on the field of approaching humans, when it comes to robots. Even less so for drones in specific. The behavior-based navigation architecture is one way of how robots can decide which way to approach people. Previously done research by E. Torta [25], regarding approaching people with robots, gives a good insight and starting point. Based on the results of these experiments a model of a person's personal space concerning the Nao robot was made. After that a smart algorithm was made to find the optimal spot for communicating, while keeping in mind obstacles that could block certain positions and or routes.

A comparable model could be defined for the use of delivery drones, as they (depending on the way of delivering) could be required to approach people as well. The interaction between the drone and user will not be of the same level as the Nao robot interacts with the user, though that does not make the way of approaching less important. Some differences between the Nao and delivery drones could be that the delivery drone will (unlike the Nao) not attempt any humanlike interaction, but rather just the needed verification and the handing over of the package. The distance between the drone and user as well might depend on different factors than for the Nao robot. Also the appearance of both robots might play a factor. Where the Nao robot, as a humanoid, tries to imitate humans to a certain extent, the delivery drone will remain a ‘mechanical machine’, requiring a different kind of trust of the user in the robot. On one hand drones give a extra dimension to this research, since also height should be implemented. On the other hand the robot used in the experiments described by Torta first approaches people and then seeks, for the user will be attending other busyness at that moment. The delivery drone we are talking about however will have the attention of the user from the start, meaning the aspect of orientation of the user can be left out.

Whereas the Nao robot is the initiator of the communication, the delivery drone can be assumed to have the attention of the user from the start. Because of this the factor of direction of approach can be left out, for the user will turn towards the drone. The two factors that will be tested for comfort during this project are the distance between the user and the drone and the flying path to take for approaching a user. A more extensive description of the experiments is given in section 5, Approach users.

3 USE aspects

In designing a technique it is not merely about finding a technical working system that solves the problem as explained in the previous chapter. User preference is playing a big role in the success or failure of such a technique. By analyzing the User, Society and Enterprise factors for a given product, the needs and requirements for the product can be determined and specified better. For this, an scenario is made, to explain how the future is seen for this new way of delivery. (see Appendix A)

3.1 Users

Users can be categorised into three different categories. Primary, secondary and tertiary users.

3.1.1 Primary user

Primary users are the users that the technique is aimed at, the main people to interact with. In the case of the delivery drone these are the people that will have their package handed over by the drone:

- Consumers, people who order online

3.1.2 Secondary users

Secondary users are the people who also will be making use of the technique, but have less direct contact with the drone than the primary users. Some secondary users of the delivery drone are listed below.

- Companies and shopkeepers
- Drone developers
- Drone manufacturers

3.1.3 Tertiary users

The last group of users are the tertiary users. This group often contains the users that are only incidentally confronted with the technique, people working in the same environment or people who perform maintenance on the drones for example:

- Mechanics
- Safety instances (in case of accidents)
- People walking/using the streets
- Other airspace users
- Government instances (new laws)

3.2 User needs

The main focus of the project will be interaction with the drones of the primary users. Therefore an in depth analysis of these users and their needs is needed. Other users however may play a part in this project as well, so for some their needs will be illustrated briefly as well.

General primary user needs for drone delivery are fast, trustworthy and safe delivery of their packages. Note that no discrete values can be given to what is fast, trustworthy or safe delivery. In the scope of the project these needs however are slightly different. Safety still remains a priority, the user should in no way be exposed to risky and or dangerous situation regarding the drone. Since the flight of the drone to the address is assumed to be no problem, fast delivery falls a little more to the background. Trustworthy delivery remains an user need, but its meaning changes with the context. Here trustworthy delivery is not so much about not damaging or losing the package on the way, but rather about a being able to find a location to land no matter the environment. Last an extra user desire comes within focus, the comfortability of the delivery. This involves the drone's behavior towards the user which should not only be safe, but also feel safe and comfortable.

Companies and shopkeepers will be the ones providing the service of drone delivery. Therefor they will take a large portion of the responsibility for the drones. Their needs will lay in reliability of the drones. Another important need for companies is for the drones to be cheap, or at least affordable. A right balance between price and quality must be found. Also for they are to provide the service to the consumers, consumers needs automatically become needs for companies and shopkeepers as well.

Of course with the increasing use of drones, developers and drone-producingcompanies will be able to make money with it. Also new developments will be stimulated. For companies producing drones, the ease of producing will be an important need as well as the expense of separate parts.

Generally, taking full responsibility as the drone producer can be seen as a generous gesture towards customers and will also push the development of autonomous vehicles onto the main audience. Other autonomous machine producers have already done so: "Volvo, Google and Mercedes have now all said that they will accept full liability if their self-driving vehicles cause a collisio" [9].

The primary need for mechanics is that the drones are easy to repair or preform maintenance on, as well as safety doing.

A need or wish for safety instances it to have the drones to be able to fly without accidents, for their priority is to provide a safe living environment. And in case of an accident, which will unfortunately be inevitable, the damage must be minimal.

For people using the streets the main need concerning drone delivery is to be able to walk the streets safely without the fear or risk of an accident.

4 Landing phase

The goal of this part of the project is to create an implementation which allows the drone to find a good landing location near the user. This will ensure that the drone does not land on inconvenient locations such as edges and unstable terrain. In order to achieve this, the drone creates a map that contains information about the terrain. Using this map the drone tries to find a safe area to land on.

4.1 Mapping

A drone is not able to have a whole map of the area that is fully accurate when it arrives at it's destination. To find an area that is safe for landing the drone needs to know what the terrain looks like. The drone control software, generates a 2D map with measurement points that it receives from the drone. Each point contains information about that specific area. In the current state of the project only the height of the terrain is taken into account. With the altitude points, a so-called height map is created.

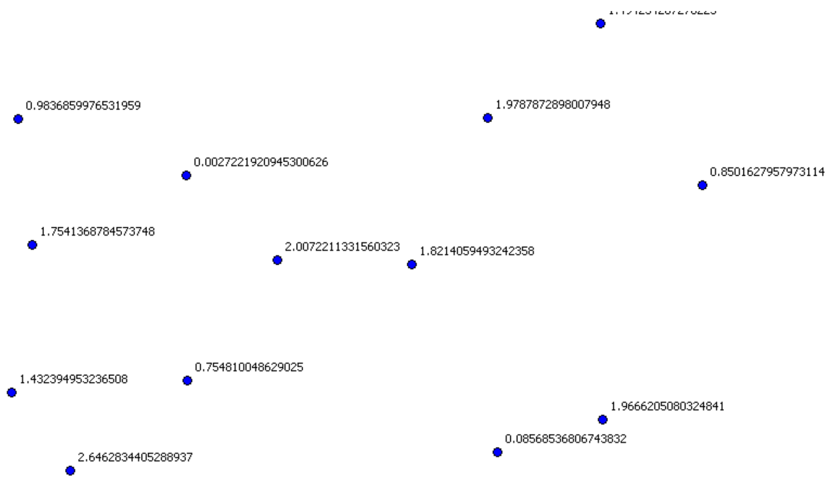


Figure 4.1: Mapping points

To find a suitable landing area, the measured points are converted into areas in a Voronoi diagram. If the area's of the Voronoi diagram are smaller than the area of the cone created by the ultrasone sensor, it can be assumed that the Voronoi diagram gives a good representation of the real world.

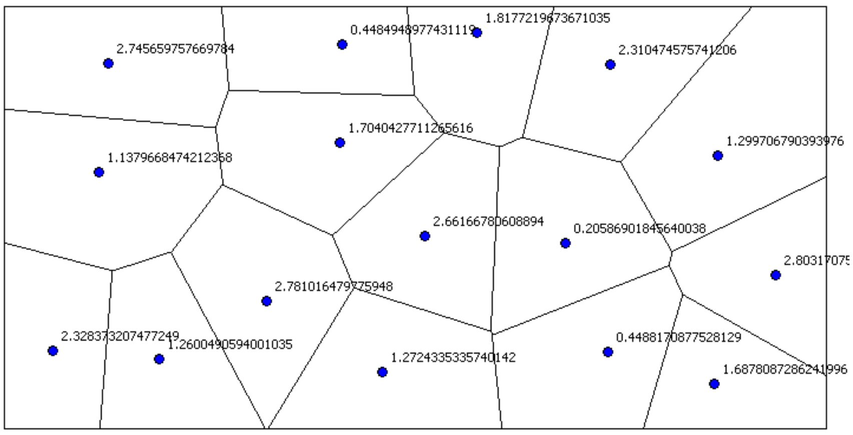


Figure 4.2: Create Voronoi diagram

Once the algorithm has a set of probing points with the desired Voronoi diagram, it can search for landing candidates. This works by comparing each individual area with its neighbors. If the altitude of the neighbors is similar to the tested area, it will become a possible landing candidate. Figure 4.3 shows an example of a landing candidate in green.

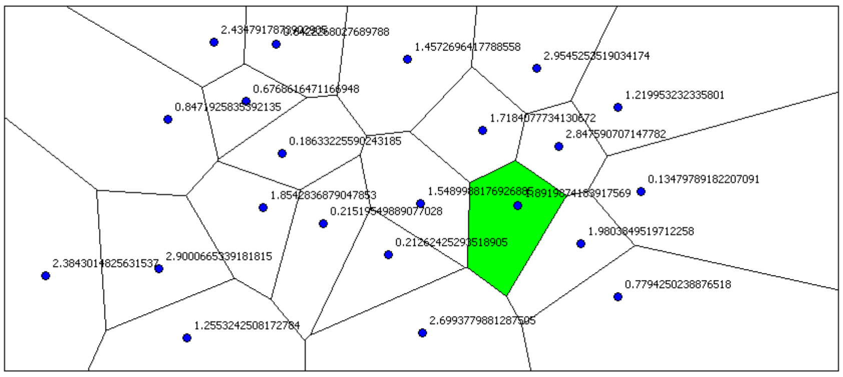


Figure 4.3: Candidate selection

To get the best possible landing area, the candidates are being sorted based on difference between neighbors, in case two areas have the same differences, one with the highest number of neighbors will gain priority due a higher amount of probing accuracy.

It is also possible to make a rough scan of the environment. Selected a possible candidate area and scan this individual area again but with more precision.

4.2 Optimal exploration strategy

Mapping the possible landing area can take quite a long time if no flat and safe area can be found. To do this as fast as possible an optimal exploration strategy has to be used. To cover a random determined area the best strategy to use is to zigzag over the area [24]. There are two possible directions the drone can sweep the area, along the short or long side as is depicted in figure 4.4.

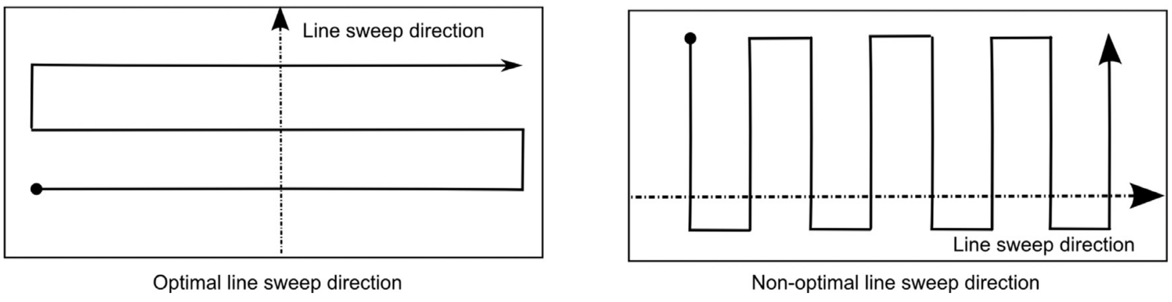

Line sweep direction
Optimal line sweep direction
Line sweep direction
Non-optimal line sweep direction

Figure 4.4: Two possible sweep directions

Since making turns takes some time the amount of turns needs to be minimized. This can be done by choosing to fly parallel to the long sided edge of the area.

After the selection process, the algorithm could be executed again on a selected area to improve the results. This step could be repeated until the area is at least as big as the drone itself and can assure that the final area is flat. In the current implementation this is not done because the goal is to be able to avoid large objects.

4.3 Practical implementation

It is decided to make an example on how it is possible to do mapping with low specifications of a drone. The drone should be able to determine it's own position, make a height map to find a possible landing spot and finally going to that landing spot. In this paper the Parrot AR.Drone 2.0 is used. This is a remote controlled flying quadcopter. It was designed to be controlled by mobile or tablet with operating systems such as iOS or Android [17]. The Parrot AR.Drone 2.0 has the following specifications:

Dimensions (A):	0.451x0.451 m^2	(0.517x0.517m^2 with Indoor Hull)
Weight (m):	380 g	(420 g with Indoor Hull)
Battery life (t):	12 min	(in theory)
Charging time (t):	60 to 90 min	
Interfaces:	USB and Wi-Fi	

The drone also comes with a frontal HD camera (720p, 30FPS) and a QVGA bottom camera (480p, 60FPS), both with the possibility of direct streaming. The height of the drone is measured with onboard ultrasound sensors.

To get this example working, a program has to be written to extract the measurement points of the drone's location to do the mapping. Also, the control of the drone has to be made. The following parts are used (these are explained in Appendix B):

1. Official AR.Drone SDK by Parrot
2. JAVA AR.Drone SDK
3. OpenCV (Computer Vision Library)
4. PvAPI (Library for GigE cameras)
5. GigE GE1900C Camera

4.3.1 Determining position

The drone will be located inside an area where the drone can be followed and can be located. The position of the drone will need coordinates in the x, y and z direction as can be seen in figure 4.5 to locate itself.

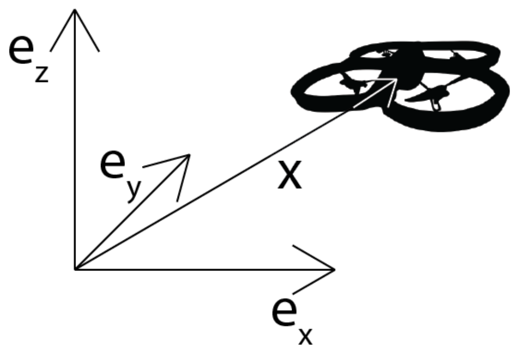


Figure 4.5: Schematic representation of the position of the drone

The height of the Parrot 2.0 drone is found by using the onboard ultrasone sensor. This sensor will send the current height of the drone to the connected device. The received height will be the height between the drone and the nearest surface. This could be the ground, but also an object.

Commonly known and used GPS-technologies could be used in order to determine the position in 3D-space, however this method is not always applicable in certain areas. In this case of indoor testing, other technologies need to be used to reach the same localization GPS can offer outdoors. To solve this issue, one could use a camera to track the drone realtime in 2D-space. This works by locating the drone inside the video-feed and then translate this relative position inside the video to real life coordinates. One requirement is that the camera should always stay at the same position in order to keep the reference point equal. Once these initial steps are taken, OpenCV is used to analyze the video input and locate the drone. The drone control program uses openCV to search for red and blue shapes inside the videostream. This is convenient in our specific situation since the test field is mostly green with white lines whereas the drone happens to have a red/blue cover. Once the areas are located, the algorithm returns the center-points of these shapes to give a total of two 2-dimensional points (red and blue coordinates). The reasoning behind two points instead of one is the fact that it becomes possible to determine the orientation of the drone since the angle α can be calculated by comparing the line that goes through point blue and red with the vertical y-axis line. (also see Appendix C

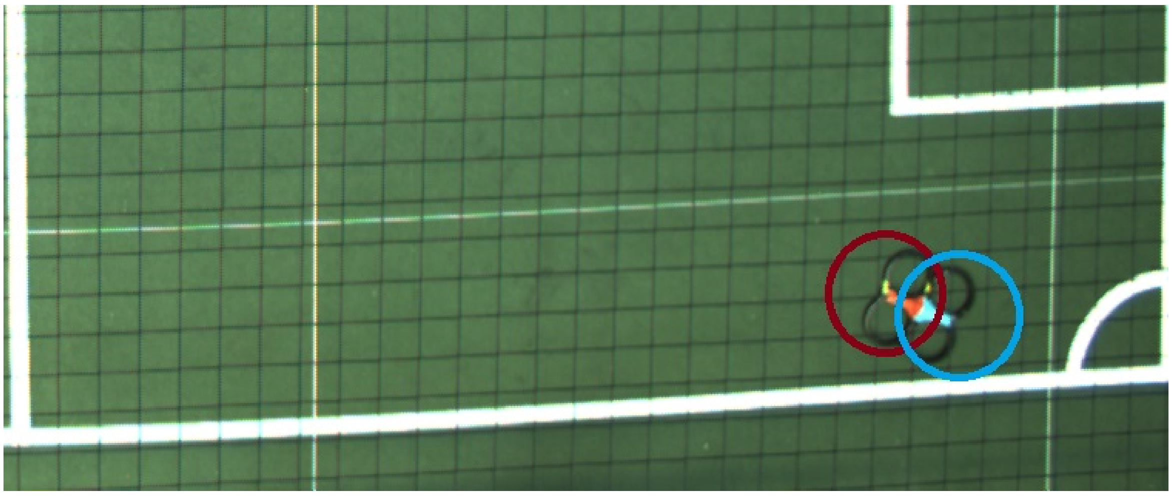


Figure 4.6: Section of the field where the drone is found and its location is determined

Once the points are being sent as a livestream data, the points are being converted to real-life points using the following formula:

$$\vec{x} = \frac{p_x l_{x,\text{field}}}{l_{x,\text{video}}}\vec{e}_x + \frac{p_y l_{y,\text{field}}}{l_{y,\text{video}}}\vec{e}_y + l_{z,\text{sensor}}\vec{e}_z \tag{4.1}$$

Where $\vec{x}$ represent the position of the drone in meters with the top left corner (0,0,0) as the origin. p represent the position in the videoframe where the color centerpoint is located. l_{video} equals the dimensions of the videoframe in pixels, whereas l_{field} represents the dimensions of the field in meters. $l_{z,\text{sensor}}$ is found while reading out the height sensor of the drone.

4.3.2 Drone control

In order to fly a route and do the mapping, the drone should be able to fly autonomously. To do this a command based system with feedback is used. The command given is a message that specifies the position and rotation the drone should move to. This includes the altitude at which the drone should fly, the amount of horizontal rotation and the horizontal distance the drone should fly. Commands the drone should execute are added to a FIFO queue. (See also Appendix D) The control algorithm fetches an item from the queue and tries to execute it. At the start of the execution of a command, the current position, altitude and rotation of the drone are stored. After that the control program compares the current position, rotation and altitude with the starting data and moves the drone until the desired position is reached. In the current implementation only proportional control is implemented.

4.4 Final program

The final program can track the drone's location and orientation through an external camera and the sensors the drone has. The drone follows a preprogrammed path in order to scan the environment. At certain points on this path the location and the measured terrain height are put into a map. Using a voronoi diagram the points on the map are converted to areas. The program looks at areas where all neighbours have almost equal height and marks that areas as possible landing areas. The drone is send to the closest landing area to land there.

5 Approach users

Research has to be done to get the right procedure for the way of landing with a drone. Finding the right way of approaching the customer with a drone is one of the subjects.

Multiple factors can play a role for users to feel safe and comfortable with the drone approaching them. In order to be able to construct constrains and preferences for the approaching, four variables have been devised:

- Variable 1: Flying speed
- Variable 2: Approaching height
- Variable 3: Landing distance
- Variable 4: Flying path

The first two variables are technical constraints and thus attached to the approach. Variables 3 and 4 are coming from 2 experiments where values are computed for the optimal landing distance (relative to the user) and for a preferred flying path when approaching the user. These variables can be defined for only one drone, because users experience different drones with different feelings. For example, the size of the drone is very important for how close people want the drone to land. The drone that is used for the approaching users is the Parrot AR.Drone 2.0, which is the same drone as the drone that is used in Section 4. For this drone the four variables can be determined. There is assumed that the angle of arrival in the horizontal plane does not matter. This is because the user is waiting for the drone to come and automatically turns his/her face to the drone when it arrives.

5.1 Flying speed

The flying speed of the drone is important for approaching people. If the drone flies to fast, people can get afraid but if it flies to slow it would take to long. Humans average walking speed is researched to be 1.4m/s second [12], and it is assumed that it is the right speed to test with. For safety and the accuracy reasons of the experiments however, the speed of the drone has been set slightly lower; approx $v=$ 1m/s.

5.2 Approaching height

For the approaching height, a height of 1.0m is chosen. This is because of the following: Lower heights would result in issues with obstacle avoidance, whereas higher heights might pose danger for the user. Eye-height of possible users might vary from 1.50m to 2.20m [7], making this domain unsuitable for flight. Given the accuracy of the drone for keeping the height another 0.5m is implemented as safety feature.

5.3 Landing distance: Experiment 1

The variable landing distance is about the distance that users are still comfortable with the drone around. The optimal distance that users like and the nearest distance that people are comfortable with drones around are determined with an experiment. The subject (an user) stands on a given spot (l=0). The distances 0.5, 1, 1.5...7 meters are marked with masking tape (distance to test subject) on the ground. The drone will start at a distance of 7 meters ($= l_{start}$) as seen in figure 5.1.

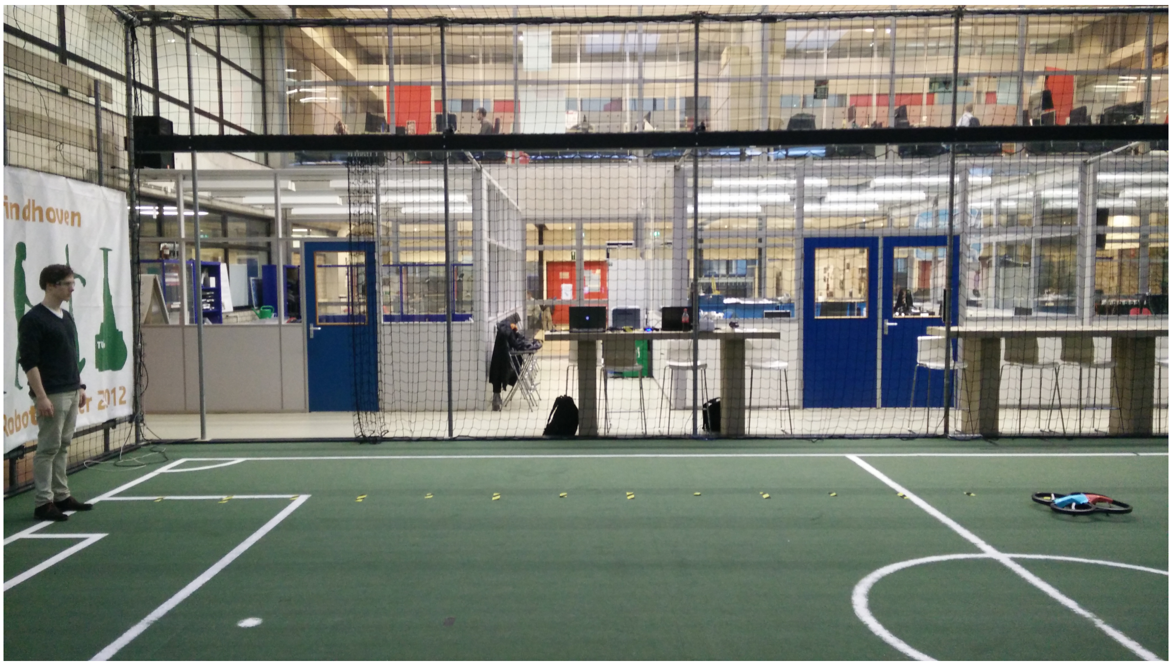


Figure 5.1: Setup

The drone will approach the person at a steady speed of approximately v= 1.0m/s. It does so at a height of h= 1.0m. Whenever the test subject feels like the current distance between him and the drone is the most comfortable distance to land, the test subject will give off a sign and the drone will be given the order to land (l_{end}). The subject will redo the test to determine the nearest distance where he or she feels comfortable. Those distances are measured and rounded per 0.25m. The results of this experiment are found in Appendix E. Because of limited battery life, later the two different runs were taken together in one single run, where both the most comfortable and the closest distance where tested for.

The mean value of the optimal distance that comes out of this experiment is 2.36m with an standard deviation of 0.59m. The nearest distance has a mean of 1.04m with an standard deviation of 0.49m. The drone should be programmed to keep these distances as boundary conditions for the landing procedure. Note again that these values are specific for the Parrot AR.Drone 2.0 and might vary for different drones.

5.4 Flying path: Experiment 2

It's not only interesting to look at the best landing distance, but also at the way the drone approaches the user. For this, a distinction is made between three different situations. For a description of these situations see the list below. In all these situations the test person is positioned at l= 0m. The drone starts at a distance l_{start} that is from experiment 1. It starts however at a height h_{start} of approximately 4 meters as this will be the case in reality as well, for the drone flies at higher altitudes during its main trip.

Situation A
The drone flies horizontally to a certain distance l_{end} then the drone lands vertically.

Situation B
The drone flies diagonally, at an angle α, to a certain point at distance l_{end} and height h_{end}. Then the drone lands vertically.

Situation C
The drones lowers itself vertically to a certain height h_{end}. It then flies horizontally to a certain distance lend before it lands vertically on the ground.

For the distance l_{start} a distance of 7.0m is chosen. The ending distance l_{end} is chosen according to the results of experiment 1 at roughly 2.5m.

To visualise those situations, picture 5.2 is shown below.

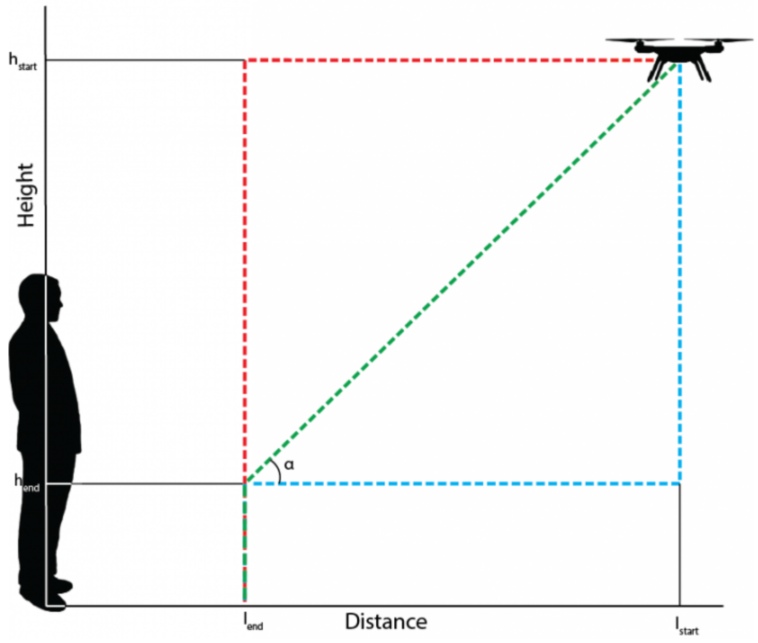


Figure 5.2: Theoretical setup

After each test variation the test person is asked to rate the experience with the values very bad/bad/neutral/good/very good. The participants need to fill in the values without seeing the others results. This for preventing the participants to get influenced by other participants. The participants were also asked to enlighten their findings and feelings to explain their choices. Some of these explanations were also reflecting on the first experiment. After a couple of experiments it was noted that the time that is consumed by this experiment is a lot. More participants can stand next to each other at the same time to speed up the process. It was also noted that the results were turning to one side. By looking at facial expressions, it helped to verify these results. The results of this experiment are found in Appendix F.

Those results show that there is a lot of variation of choice. Most of the participants felt good with the drone at a low height. Participant 6 told, that flying with only 1 of the 2 axis (vertical or horizontal) at the same time felt safer. This was confirmed while looking at the expressions of the participants.

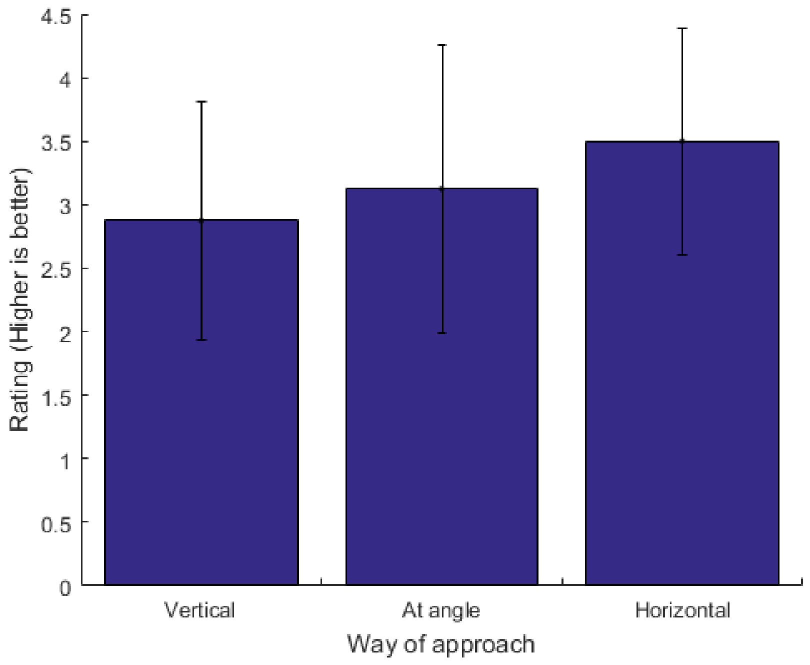


Figure 5.3: The found result of the way of approach

When the data is analysed, as can be seen in figure 5.3, it also shows that flying at low heights gives an overal better experience for the user. When a T-test has been conducted and the p-values of the possible ratings are compared, it also shows that the chance of rating flying at low heights with a “good” is around 22.3% compared to the 11.1% chance of getting this rating while approaching the user diagonally.

6 Conclusion

The goal of this paper was finding a landing procedure and user friendly approach of the last few meters. The practical outcome is the autonomous landing script for a drone. The approach itself is not universal, because of a lot of different drones. However, the way of determining this approach is.

First, the research field is investigated to find a not invented part of drone-delivery. After that, the focus of the research is determined with a detailed focus for this paper. The USE aspects are investigated and then the research started. The landing procedure is made, to find a landing spot autonomously. Meanwhile with experiments, the user friendly approach is determined for a specific drone and this verified with a T-test.

The final outcome of this paper is an universal way to find the user approach. Also, the used drone can scan an area and can find its own landing spot to land.

It is possible to autonomously find a right landing spot. The drone is controlled by a PC and a camera mounted above the testing area will be used to determine to location and orientation of the drone. With the known location a hight map can be created, the ultrasone sensor mounted underneath the drone is used for this purpose. The location of the drone and the measured height will then be stored in a map. With this map and the help of the Voronoi algoritme, it is possible to determine area's where the drone is able to land. If a safe landing area is found the drone is send to this location and will be given the signal to land.

The best way of approaching a user is to fly at low heights and land at a distance of 2,5 meters in front the user. This distance can be seen as safe since the user has the whole drone in its field of vision and has enough time to react when the drone does something the user is not expecting. The user also has the feeling that its head is the part of the body the drone can do most damage, so flying under shoulder level gives the user a more safe feeling.

Since both parts of this paper aren't connected to each other the final product isn't applicable. Further research has to be done to develop a user friendly drone delivery system to implement in the real world.

7 Discussion

The implementation of the autonomous landing could be improved by implementing the approach technique. At this moment this is not possible, because there is a need for scanning the area completely. The used drone and its components were not optimal. These components (or the whole drone) could be improved so that the mapping can be improved. For example, sensors like Lidar or radar can be used. Maybe there is even a way to use the pre made maps. Eventually the scanning of the area is optimized, so that approach can be used immediately and that there is no special scanning manoeuvre needed anymore. The landing procedure can also be improved for different kind of area's. For example, the drone can not see the difference between land and water. For that, the procedure needs to be improved.

Another point of improvement is to have the drone fly completely autonomous. At this moment the drone is controlled by a PC, but in the future however this might not be the case. So the whole program written to controle the drone could be implemented inside the drone. To make it really useful the drone should be moved to an outside area. GPS can than be used to determine the location of the drone. In this situation the drone also has real life influences, wind for example.

The approaching technique of the drone could be improved by investigating a bigger and a more diverse group of participants. According to the obtained results, the conclusion of the experiment will only be valid for Dutch males, aged between 20 and 25 years and who have a technical background. When using more participants with a more diverse background, the results are more reliable for use in common area's. Also, when using more participants, the opinion of what participants think that is the best way of approaching can be investigated. This can result in a more general outcome of the expectations from the users. When there is a systematic outcome, this could be used for future drones. This analysis was not possible with these amounts of participants. The same improvement counts for investigating more sorts of drones. By verifying the best approach for a lot of different drones, a more universal approach could emerge by combining all the data.

Since the experiments are conducted with a human controller their is quite a big difference in the quality of flying. To make the results more reliable it might be wise do let the drone fly autonomously to the user. This will result in a way more constant approaching path through all the test subjects.

Also more different ways of approaching could be investigated. Changing the approaching angle or change the distance of l_{end} are some variables that can be changed. It might be possible that the user rates the ways of approaching different when the drone lands further away.

A Scenario

The scenario starts with the customer buying a package in 2025 from the best-selling company in the Eindhoven. The customer lives in the so called "Parklaan". Within 20 minutes after payment, the drone with the package is in a radius of 50 meters of the destination.

When the drone is nearby it switches to a more accurate way of orientation with more sensors, instead of the global GPS. The drone does this to find the exact location of the front of the customer, but also to have the most accurate map as possible. The customer lives in a flat, so because of laws he still has to come out of the apartment to the front door of the building. That drones are flying around with cargo is very special, in the beginning of the drone-era they weren't even allowed to do anything.

When the drone has detected the presence of the costumer it starts 'scanning' the environment. During this proces the drone flies from the left to the right a couple of times and detects 10 objects that it can't land on. Two areas are found that are big enough for the drone to land. One area is safer because it is further away from a tree and the other is more close to the costumer. This time, the drone chooses for the area that is more close to the customer. The tree is not in the danger zone of the drone, and it is not an object that will move quickly. The drone flies to the landing spot and lands there, it gives the customer a signal that the drone is ready to transfer. The drone can hand over the package and will wait for the customer to give feedback. With ratings given by this, and other costumers, the best landing approach is updated so that none of the costumers feel threatened by the drone. After this, the drone can take off and head back to it's station to wait for it's next task.

B Implementation parts

B.1 Official AR.Drone SDK

The Official AR.Drone SDK by Parrot is written in the program language C, it is rather bulky and not all components are well-documented. Parts of this library have been reverse-engineered to detect the specific structure of certain navigation packets and the semantics of certain enumeration types, for example the different detection tag types that are reported and how to extract them. (The SDK can be found in source [18])

B.2 JAVA AR.Drone SDK

The JAVA AR.Drone SDK is a library that allows you to control the AR.Drone through JAVA, it implements the protocol to send commands to the drone, as described in the offical SDK manual. The library is multi-threaded, it utilizes three threads, one thread is used by the command queue and is responsible for sending pending commands to the drone. The second thread is concerned with receiving and parsing navigation data, delivering it to its listeners. The third thread processes incoming frames from the video stream if enabled.

This library is extended in this research since its basic functionality is not adequate and its design and documentation lack quality. The navigation data options that are parsed are limited to DEMO and Vision Detection, moreover there is also no support for setting the navigation data options. On top of this SDK there is built a specific controller, which tries to achieve line movement, from point A to B, by executing a series of move commands and in combination with feedback, tries to correct its flight direction.

B.3 OpenCV

OpenCV is a computer vision library, which makes certain computer vision tasks, like object recognition or object tracking easy. It has been used for drone tracking, where each received frame from the camera stream is processed and where is tried to determine the position of the drone in pixel coordinates in the current frame.

B.4 PvAPI

PvAPI is the legacy SDK from Allied Vision to operate GigE complient cameras. GigE is a standard protocol for controlling a certain class of IP cameras. The PvAPI SDK contains support for JAVA, by including a JNI API that wraps around the native library. JNI is the JAVA Native Interface, which allows JAVA programs to invoke programs written in different languages, like a C/C++ library. The SDK also includes some examples in JAVA for performing certain basic camera tasks like a streaming program, which is extended for the needs of this project.

B.5 GigE GE1900C Camera

The GE1900C is a camera manufactured by the company Allied Vision it adheres to the GigE Vision protocol. It is mounted on top of the center of the soccer field, such that its visible scope is the soccer field itself. The resolution of the camera is 1920 x 1080 pixels.

C Position Tracking

To determine the position of the drone on the soccer field, the mounted GigE GE1900C Camera is used. This camera is aimed orthogonally to the soccer field and centered above the soccer field, however the camera is not perfectly installed, the field is askew or the camera uses a convex lens, since the right half bottom edge bends into an other direction, as can be seen in the following figure.

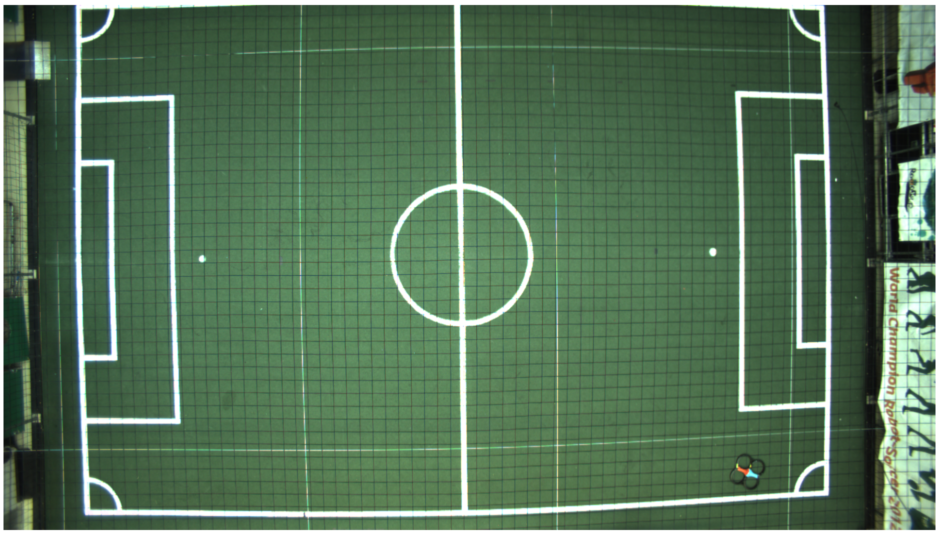


Figure C.1: Default frame from the camera.

To control this camera, the legacy SDK "PvAPI" from Allied Vision is adopted, since it comes with JNI support and JAVA examples of how to receive a video stream. By default the output frames of the camera are set to "Mono8", which is a grayscale pixel format, for tracking purposes, color is required, to achieve this its output type must be manually changed to "24-bits BGR". The file "examples/Java/JStream.java", inside the SDK installation folder, has been altered, it includes all the code for showing the live video stream of the camera and already does the conversion of a camera frame to a BufferedImage object. This projects intercepts when conversion from frame to BufferedImage is done and the signal is feeded into the OpenCV program. OpenCV analyzes the frame and publishes the location of the drone as a (x,y) pair in pixel coordinates. (with pixel coordinates is meant a 2D cartesian coordinate system with the y-axis is flipped) The center of this coordinate system overlays with the upper left corner of the frame. The resolution of the camera is 1920 x 1080 pixels, so the x-value of the position is in the domain [0,1920] and the y-value in the domain[0,1080]. The drone has a cover on top which consists of two colored areas, where one area is colored red and the other cyan (light blue). The drone is detected by filtering each buffered image for the colors red and cyan. The OpenCV algorithm transforms the BufferedImage into a matrix representation of type "24-bits GBR"

A problem encountered was that the red/orange robot and the orange-like text on the banners at the right side of the soccer field, were picked up by the detection algorithm as the red cover of the drone, efforts in trying to resolve this by adapting the color detection range have failed, therefore we have resoluted to cropping the image, removing its surroundings such that only the soccer field remains in the frame. This decision has as a consequence new domain ranges for the x-value and y-value. See the next figure.

The cropped matrix is eventually used for the color detection filter.

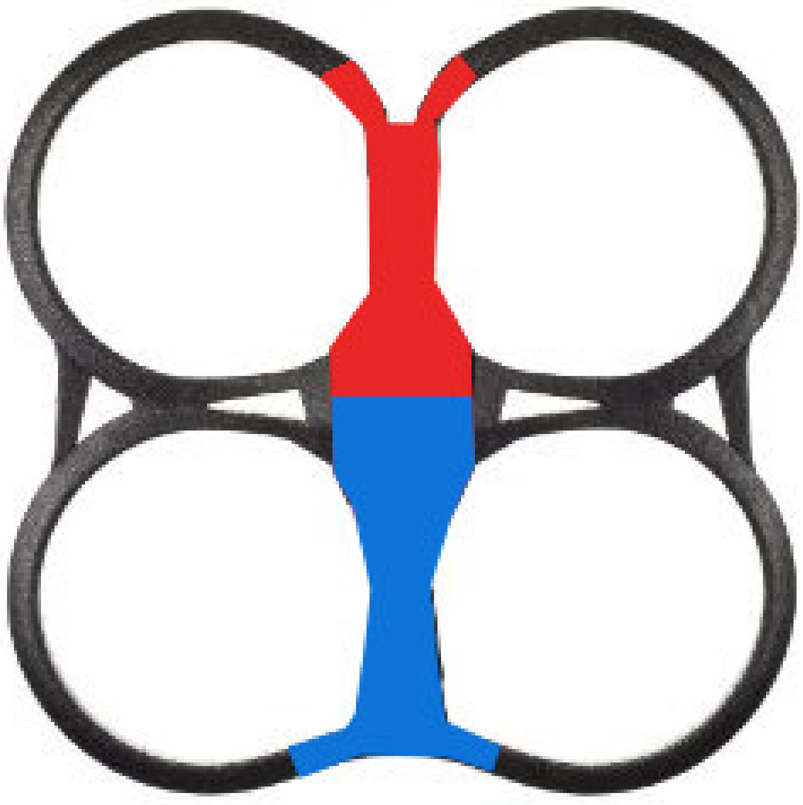
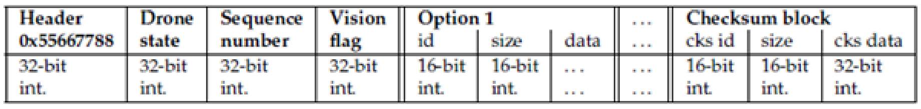


Figure C.2: Hull of the drone.

C.1 Efficiency

The achieved throughput of the camera was initially a rate of 5 FPS for a frame payload of . To improve that one can employ jumbo packets. Jumbo packets must be activated inside the network interface settings, this allows packet delivery of 9 Kb in size. Also some attributes of the camera itself must be changed, like the "PacketSize" attribute where you can specify the size of the packets the camera uses to deliver it frames in. Further the bandwith constraint on the camera output can be disabled, such that it does not limit its sending rate to the available bandwith, although this introduces a less reliable video stream.

D Navigation Data



Header 0x55667788	Drone state	Sequence number	Vision flag	Option 1			…	Checksum block		
				id	size	data	…	cks id	size	cks data
32-bit int.	32-bit int.	32-bit int.	32-bit int.	16-bit int.	16-bit int.	… …	… …	16-bit int.	16-bit int.	32-bit int.

Figure D.1: Navigation Data Options

The drone sends navigation data packets to the client approximately every 5 ms, which translates to a rate of 200 Hz.

Each packet is in Little-Endian byte order, this means that the Least Significant Bit (LSB) is stored at the lowest memory address, or when you write it out on paper, it is the most left bit. The 4-byte header of each packet is the unique identifier 0x55667788 in hexadecimal. The following 4 bytes form a 32-bits field which represents the drone's state. Bytes 8 through 11 form the sequence number, this is useful for synchronization, since the packets are sent over UDP which is an unreliable data transfer protocol, which means that packets can get lost and can be delivered out of order, so it is possible to receive old navigation data packets, which can be discarded. Bytes 12 through 15 form the fly state bit field, the semantics of this field are unknown, nothing about this field is explained in the SDK documentation and nothing about it has been discovered from the official SDK library. After these first 16 bytes, the option segments follow. Each option has the common structure where the first 4 bytes represent the option ID and the length of the option segment, the length includes the 4 bytes header. This means that the length of the actual option data is $size - 4$ bytes long. After the 2-byte size field the payload will follow, whose structure depends on the specific option. The SDK documentation does not tell you anything about their structure you have to derive their structure yourself by inspecting the C-Structures in the file "navdata_common.h" in the official SDK library.

Each navigation data option has a fixed length, even when fields are unused. Probably done to make the parsing consistent and easier compared to the case where option data size would shrink with the amount of useful data contained. Be aware of this when you are parsing an option segment like Vision Detect, although one tag might have been detected, the packet contains still information for 4 tags, the fields of the other 3 tags are simply padded with dummy data.

To figure out which navigation data options are enabled one has to inspect the configuration parameter: "general:navdata_options", the value stored at this parameter is the integer value of a 32-bits field. Even if all the options are deactivated the drone will send at least the 16-byte header with the identifier, drone state, sequence number and fly state, and the 8-byte checksum option, so you should always receive a packet of at least 24 bytes in size.

The following navigation data options are available listed with their corresponding ID, name and description. In total there are currently 28 available options, excluding the checksum option, which is not an option you can activate yourself. For now, only a description with the most important options is provided. The documentation does not tell anything about their structure, to figure out their structure one must reverse-engineer some of the official C-code library, which can be difficult, the library file that describes these structures is "navdata_common.h". However many of the fields in these structures are undocumented, sometimes one can figure out their semantics just by the variable name, but not always and for some properties it is useful to know the value range, which is also unspecified and must be figured out by yourself. Another quirk is that some of these fields are not actually in use, but were reserved for future use.

Note: you won't find the correct structure of "ZIMMU_3000", which represents the GPS data, in the "navdata_common.h" file, the actual structure is published in source [16].

To check whether a certain option is enabled one must compute the bit mask for a specific option and compute the bitwise logical AND of the options value with the mask, if the computed value equals

Table D.1: Navigation data options

ID	Name	Description
0	DEMO	Minimal set of navigation data, this option is probably the most useful one, since it contains the values of all interesting parameters like: altitude, pitch, roll, yaw, battery level, fly state, control state.
1	TIME	Internal time of the drone.
2	Raw Measures	Raw sensor measurements
3	Phys Measures	Contains something about temperature measurements.
4	Gyro Offset	
5	Euler Angles	
6	References	
7	Trims	Trim values.
8	RC_References	
9	PWM	Data about the motors, probably about the modulus of the Pulse Width Modulation signal.
10	Altitude	Multiple sensor data values about the drone's altitude.
11	Vision Raw	
12	Vision Of	
13	Vision	
14	Vision Perf	
15	Trackers Send	
16	Vision Detect	This option publishes the information about the detected tags if tag detection is enabled.
17	Watchdog	Watchdog status.
18	ADC Data Frame	
19	Video Stream	
20	Games	
21	Pressure Raw	
22	Magneto	Contains fields that represent the sensor data from the magnetometer, which is basically the compass of the drone.
23	Wind Speed	
24	Kalman Pressure	
25	HD Video Stream	
26	WIFI	
27	ZIMMU_3000	This option represents the GPS data.
65537	Checksum	The checksum option is always the last option of the packet, this option is 8 bytes long, where the first 4 bytes are the usual header and the next 4 bytes represent the 32-bits checksum.

the mask then the specific option is enabled. The mask is computed with the following expression: bitmask(id) = $1 << id$, which boils down to the id-th bit is set to 1, all other bits are 0. Where "<<" denotes the left-shift operator and id is the id of the desired option.

Example:
Suppose the options bit field is "1 0000 0000 0000 0001". To check if vision detect is enabled. Vision detect has option ID: 16 and therefore mask: "1 0000 0000 0000 0000". Computing this gives:

$$\begin{array}{rl} 1\ 0000\ 0000\ 0000\ 0001 & \\ 1\ 0000\ 0000\ 0000\ 0000 & \& \\ \hline 1\ 0000\ 0000\ 0000\ 0000 & \end{array}$$

Noted is that the result equals the mask, thus the vision detect option is enabled.

The maximum packet size is 4096 bytes, this value comes from the official SDK from the file "navdata_common.h".

D.1 Checksum

One can compute the checksum of a packet by summing up all the bytes in the packet, excluding the last 8 bytes which form the CHECKSUM option, treating each byte as an 8-bits unsigned integer. The computed checksum must equal the value in the checksum option, if not one should ignore the packet since it was corrupted.

E Results experiment 1

The results of experiment 1 are seen below.

Table E.1: Results experiment 1

Experiment	**Optimal distance (m)**	**Nearest distance (m)**
1	2.25	1.0
2	2.75	0.75
3	2.5	1.0
4	2.25	0.75
5	2.0	0.75
6	1.75	0.5
7	3.5	2.0
8	3.5	1.75
9	1.75	0.5
10	2.0	1.5
11	2.5	1.5
12	2.25	1.25
13	1.5	0.5
14	2.5	0.75

F Results experiment 2

Results approach A

Table F.1: Results approach A

Experiment	Very bad	Bad	Neutral	Good	Very good
1A		X			
2A	X				
3A		X			
4A				X	
5A			X		
6A				X	
7A			X		
8A				X	

Results approach B

Table F.2: Results approach B

Experiment	Very bad	Bad	Neutral	Good	Very good
1B				X	
2B				X	
3B				X	
4B			X		
5B				X	
6B	X				
7B	X				
8B				X	

Results approach C

Table F.3: Results approach C

Experiment	Very bad	Bad	Neutral	Good	Very good
1C				X	
2C					X
3C				X	
4C		X			
5C			X		
6C				X	
7C				X	
8C			X		

Explanations participants

Participant 3's opinion was that he likes to look to top of drone. If he could see into the rotors, it felt a lot safer than if he looked from the bottom and saw the rotors coming down.

The 5th participant felt the best when he saw the drone going down while coming to him. He felt that the drone was not going to run into him and that it knew what it was doing.

Participant 6 told that 2 degrees of freedom scared him. If a drone does only action at the time, like moving horizontal or moving vertical, it felt a lot safer.

Participant 7 said that as long as he could see the drone, it felt right.

As was already put in the report, participant 8 told that flying on eye-height was not nice to encounter. The drone didn't fly on eye-height, but probably it felt this way because of the flying was by hand and not autonomously. (So there were small height changes)

G Legislation

Rules from the Netherlands, category light (4-150 kg). [20] [21] [22]

Private

- Stay under 120 meters
- Keep direct sight on your drone during the whole flight
- You can only fly during daylight, with an environment that allows for clear sight
- Avoid people and animals
- It is not allowed to fly above cities, towns (150 m), railways, roads and docks(50 m)
- Stay away from airfields
- No flying within no fly zones

Commercial

As a company it is obligatory to have a ROC (RPAS operator certificate) for flying drones. (Since 2015 1st of July) This means the following:

- The drone pilot has a certificate for flying drones
- The drone has a BvL (bewijs van luchtvaardigheid, 'prove of airworthiness')
- You are in the possession of an approved operations manual

Also:

- Keep direct sight on your drone during the whole flight
- You can only fly during daylight, with an environment that allows for clear sight
- Other airspace users will have precedence at all times

Concerning privacy

Besides the normal privacy rules:

- Not allowed to (systematically) gather data about people, including public
- Not allowed to use advanced camera like infrared, night-vision, heat-camera, camera with built-in analytic techniques or cameras which do nothing but monitor
- Not allowed to gather information that is publicly published Filming through windows or looking in buildings is direct infringement of privacy.

References

[1] Amazon. Amazon prime air, 2015.

[2] Amazon. Amazon prime air, 2016.

[3] J. Boonen, L. de Jong, R. Kerstens, J. Kruijtzer, and J. Linssen. Flexible drone delivery, 2014.

[4] T. Bradshaw. Google tests drone deliveries in australia, August 2014.

[5] A. Conner-Simons. Self-flying drone dips, darts and dives through trees at 30mph, October 2015.

[6] The Daily Conversation. Amazon prime air: The drone takeover, 2013.

[7] Ergotron Ergonomics Data. Ergonomics data and mounting heights.

[8] TU Delft. New theory allows drones to see distances with one eye.

[9] S. Elmer. Volvo, google and mercedes to accept responsibility in self-driving car collisions, October 2015.

[10] S. Fecht. Spacex launch successful, but drone ship landing fails.

[11] IMR FEE. Parrot ar-drone autonomous takeoff and landing.

[12] British Heart Foundation. Walks and treks faqs.

[13] T.C. Frankel. Biggest obstacle for delivery drones isn't the technology: It's you and me. the washington post, 2016.

[14] K. Fujii, K. Higuchi, and J. Rekimoto. Endless flyer: A continuous flying drone with automatic battery replacement, 2013.

[15] N. Kulkarni. Walmart is looking to get into the drone delivery game, October 2015.

[16] Lesire. Get gps info, 2014.

[17] Parrot. A.r. 2.0 drone parrot.

[18] Parrot. Parrot for developers.

[19] Qualcomm. Qualcomm video pronkt drone vliegen in autonome modus, January 2016.

[20] Rijksoverheid. Drones en privacy. 2015.

[21] Rijksoverheid. Handleiding voor een gebruik van drones dat voldoet aan de waarborgen voor bescherming van de privacy. 2015.

[22] Rijksoverheid. Mag ik een drone inzetten voor zakelijk gebruik? 2016.

[23] K Stolaroff. The need for a life cycle assessment of drone-based commercial package delivery. 2014.

[24] D.A. Verdegay J.L. Torres J.C. Torres, M. David A. Pelta. Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction. 2016.

[25] E. Torta. Approaching independent living with robots.