NTNU – Trondheim
Norwegian University of
Science and Technology

Optimal Path Planning for Unmanned Aerial Systems

Erik Johannes Forsmo

Master of Science in Engineering Cybernetics
Submission date: June 2012
Supervisor: Thor Inge Fossen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

This thesis is a contribution to the Unmanned Aerial Vehicle (UAV) project at the Department of Engineering Cybernetics, which is a project where contributions from master students and Phd students will result in an autonomous aerial vehicle. The unmanned vehicle laboratory has its own UAV, the Odin Recce D6 delta-wing aircraft which is to be considered in the overall project.

When the UAV is in the air on a mission, one important thing is to ensure that the UAV detects obstacles, such as mountains, buildings and other aircrafts. No-fly areas should be avoided by the path planner. This thesis considers a guidance system that will set up a path from the initial position to the final destination, and make sure that the generated trajectory is safe.

One problem with the design of the optimal path has been that the designed path gives *corner cutting* when obstacles from the environment is included in the path-planner. To avoid this problem, which happens because discrete time is considered, two different solutions to avoid this problem have been discussed closer.

Implementation of constraints and different cost functions for path planning with collision avoidance using the Mixed Integer Linear Programming (MILP) is one of the purposes of this thesis. The MILP algorithm is developed for the case of planar motion where the UAV has to fly around the obstacle, and can't fly over or under it.

The design of the path path planner using MILP is done in two different ways. One where obstacles are known at the beginning of the optimization, and one where obstacles are added as information to the path planner when they are in the range of the UAVs radar. It is shown that the implementation with obstacle radar detection is more realistic, and that it also improves the computation time. As the author knows this method has not been published in articles up to this date.

Two different approaches for search of a defined area with an arbitrary number of UAVs with camera systems have been developed and implemented through this thesis. As far as the author of this thesis knows these approaches for search have not been published up to this date. Efficient search and low computational complexity has been important design factors during the development of these approaches.

The final systems are simulated in MATLAB for some test-scenarios. Also, reflection and discussion on further improvement on the path planning system are included in the report. This includes further improvement of the guidance system using receding horizon strategies.

A literature study on path planning with receding horizon has been done.

Abstract

Denne masteroppgaven er et bidrag til det ubemannede fly prosjektet, som er et forskningsprosjekt på Institutt for Tekniske kybernetikk på NTNU. Bidrag fra mastergradstudenter og doktorgradsstudenter skal resultere i et autonomt ubemannet fartøy. Prosjektets laboratorium har sitt eget fly av typen Odin Recce D6 deltavinge fly. Dette flyet vil bli betraktet i det overordnete prosjektet.

Når flyet er i ute oppdrag er viktig å sørge for at flyet har nødvendige systemer slik at det kan unngå hindringer som fjell, bygninger og andre hindringer. Forbudte flysoner må flyet holde seg unna. Denne masteroppgaven omhandler et baneplanleggingssystem som skal finne en bane fra en startposisjon og til en sluttposisjon og sørge for at denne banen er trygg.

En av utfordringene med systemet har vært at den optimale banen flyr over hjørnene til hindringene som er i omgivelsene nå hindringene har blitt modellert som rektangler. For å hindre dette problemet, som oppstår fordi systemet er på diskret form, har to ulike løsninger for å unngå dette problemet blitt inngående diskutert.

Implementasjon av begrensninger og ulike kostnadsfunksjoner for baneplanlegging med kollisjonshindring ved bruk av MILP (Mixed Integer Linear Programming) er et av formålene med denne masteroppgaven. MILP algoritmen er utviklet for det tilfellet der kun horisontal flybevegelse kommer i betraktning. Dette medfører at flyet må fly rundt hindringer i banen og kan ikke fly over eller under disse.

Designet av baneplanleggeren er gjort på to ulike måter. I det ene designet er alle hindringer i omgivelsene kjent for baneplanleggeren når baneplanleggingen starter. I det andre designet blir informasjon om nye hindringer i omgivelsene gitt til baneplanleggeren når disse er innenfor rekkevidden til flyets radar. I denne oppgaven blir det vist at designet der hindringer blir tatt med i baneplanleggeren når de er i en viss avstand fra flyet, er en mer realistisk antagelse, og at det i tillegg forbedrer beregningstiden for å finne en optimal løsning.

To ulike måter for å søke et definert område med et vilkårlig antall fly med kamera, har blitt utviklet og implementert i denne oppgaven. Effektivt søk og liten beregningstid har vært viktige faktorer å få nærmere belyst under arbeidet med disse metodene.

De utviklede metodene har blitt simulert i MATLAB for ulike scenarioer. Refleksjon og diskusjon som omhandler videre utvikling av baneplanleggeren er en del av denne oppgaven. Dette inkluderer videre forbedring av baneplanleggeren ved bruk av *receding horizon* strategier. Et litteratursøk på *receding horizon* har blitt har blitt gjort.

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Engineering Cybernetics

MSC THESIS DESCRIPTION SHEET

Name: Erik Johannes Forsmo
Department: Engineering Cybernetics
Thesis title (Norwegian): Optimal baneplanlegging for ubemannede farkoster
Thesis title (English): Optimal Path Planning for Unmanned Aerial Systems

Thesis Description: The purpose of the thesis is to develop a guidance system for Unmanned Aerial Systems (UAS). *Mixed Integer Linear Programming* (MILP) should be used for optimal path planning. Methods for collision avoidance and search missions should also be developed using MILP. The search path should be parameterized using an Archimedes spiral.

The following items should be considered:

1) A literature study on guidance systems and collision avoidance for UAS.
2) Design a guidance system for optimal path planning and obstacle avoidance using MILP.
3) Extend the path planner to handle multiple aircraft.
4) Design a guidance system for automatic path planning and show how the guidance system can be used in search missions.
5) Conclude your results in a report and include case studies verifying your theoretical derivations.

Start date: 2012-01-09
Due date: 2012-06-04

Thesis performed at: Department of Engineering Cybernetics, NTNU
Supervisor: Professor Thor I. Fossen, Dept. of Eng. Cybernetics, NTNU
Co-supervisor: Dr. Esten Grøtli, Dept. of Eng. Cybernetics, NTNU

Preface

This Master thesis is written as a part of my Master of Science program at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology.

I chose to join the *Unmanned Areal Vehicle* (UAV) project because of curiosity about the project, and I wanted to contribute to an ongoing research project - making a difference on the final realization.

No contributions to this project had earlier considered the guidance system, which in my opinion is one of the key factors in obtaining a robust aircraft control system. That is the main reason for choosing an issue with focus on optimal path planning with collision avoidance in my Master thesis. Different tasks where the objective is take in different search missions is an important field and I think UAVs can be very beneficial and can be an important resource, for example in rescue missions. This was the main motivation for me further develop the guidance system to conduct efficient search and rescue missions, by considering a specific search and rescue scenario.

This project has challenged me in the field of guidance and optimization, a subject that I have not emphasized in my previous studies. I have gained considerable knowledge in this specific field during the work on my Master thesis.

I would like to thank my supervisor Professor Thor I. Fossen for his commitment to the UAV project and for support and discussions during the last semester. I would also like to thank Dr. Esten Grøtli for his recommendation to the YALMIP modeling language and Gurobi solver. I would also like to thank Grøtli for his interest in my work, and for the discussion we have had during the semester.

I want to thank my fellow students at the *Unmanned Vehicle Laboratory* for a good work environment and for the support we give each other during the semester.

Trondheim, 2. Juni 2012

Erik Johannes Forsmo

Contents

List of Figures

Part I

Path Planning by applying Mixed Integer Linear Programming

Chapter 1

Introduction

The use of Unmanned Aerial Vehicles (UAVs) is a field with a lot of potentials. In recent years UAVs have been used for both military operations as well as civilian operations. In the USA, UAVs with a camera system has been established for surveillance purposes of the continent. The challenge has been to detect and register guideposts along the roads, and collect the registered information in a database. This process would have been very time consuming and expensive if one were to use manual labor to do this registration. Other situations where the application of UAVs can be of substantial interest, is in cases where documentation or confirmation of conatamination and pollution on earth is of importance for further evaluation, both nationally and globally. As an example, the threat of ice melting in both Artic and Antartic areas is of importance to observe. Use of UAV technology would probably offer the best resource for an indisputable scientific verdict of the situation in both polar regions.

Applications of UAVs described in literature include localization of radars, wildfire management, polar climatology, agricultural monitoring, border surveillance, reconnaissance, geophysical survey, environmental and meteorological monitoring, aerial photography, and search-and-rescue tasks as described in [Grøtli and Johansen, 2011b]. In general, autonomous vehicles are chosen for tasks that are either dirty, dull or dangerous, or to missions where there is a cost reduction [Grøtli and Johansen, 2011b].

Many more areas for the use of UAV will probably arise when commercial actors learn about the possibilities and benefits with UAVs, and the use of UAV in different operations will probably be an important industry in the future.

To get permission to operate with UAVs, the UAV has to operate in an *Equivalent Level of Safety (ELOS)*, in the same way as for manned vehicles. "*Commercial use of UAVs is currently limited by their inability to detect, sense and avoid airborne hazards*" [Hutchings et al., 2007].

Figure 1.1 shows a picture of the Odin Recce D6 delta-wing aircraft, which is considered in the overall UAV project.

1.1 The contributions to the project up to this date

The Unmanned Vehicle Laboratory is a laboratory at the Department of Engineering Cybernetics, where master students working on this topic are gathered. The overall goal is to use the research from master students and Phd student to

Figure 1.1: Illustration of the Odin Recce D6 delta-wing aircraft (from www.odin.aero)

get the Odin aircraft to be an autonomous UAV. The objective of the research is to create conditions for commercial investment.

The UAV project is still young, and the first master students started with projects and master thesis's in this field in the fall semester of 2009 and spring semester of 2010, respectively. According to Professor Thor I. Fossen many more people will work on this project (master students and Phd student) for the next years.

In the spring of 2011 [Dønnestad, 2011] developed a mathematical model of the equations of motion for the UAV. From these equations he made a simulator in MATLAB, which illustrates the behaviour of the UAV when it gets different inputs from the actuators (flaps, propeller and engine), and when it is affected by disturbances. Dønnestad [2011] also describes other topics which his fellow students worked on in the project thesis (TTK4550) fall 2010:

1. Inertial Navigation Systems (INS), Observer design and Extended Kalman Filter (EKF).

2. Framework for Operating System and Peripheral Interfacing related to UAS.

3. Flight simulator framework, and aircraft modeling.

4. Modeling of Global Navigation Satellite System (GNSS) for Hardware-in-the loop (HIL) testing.

5. Hardware and software integration of the UAV.

Some students also worked on the UAV project as a summer internship in 2011, and improved some of the work done before and corrected errors in the models.

In the fall of 2011, Master students at the Department of Cybernetics wrote project thesis's about different topics related to the UAV project. The topics which were considered at the Unmanned Lab where the following items:

1. Implement different control structures and see which one performs best as an autopilot for take off.

2. Improvement and correction of errors in the assumptions done in Dønnestad [2011].

3. Literature study on the L_1 adaptive control method.

4. Theory on guidance and control which guarantees stability.

5. Hardware "in the loop" test-platform for hardware in the loop testing.

6. Guidance system and collision avoidance.

The last item was the topic the author of this thesis worked on in the fall 2011.

1.2 Motivation

Related to the UAV project, there are a lot of topics that needs to be explored and systems that need to be designed to make it possible for the UAV to operate in an *equivalent level of safety* to that of manned aerial vehicles. At the Unmanned Vehicle Laboratory the following topics related to the UAV, among others, has been worked on has been worked on by graduate students during the spring of 2012:

1. Develop a guidance system for take-off, and implement sliding mode control for pitch and altitude.

2. Develop a system identification experiment and do implementation such that the UAV model parameters can be found when the Odin aircraft has been on a flight and system data has been collected.

3. Implementation of L_1 adaptive control law for UAV, and simulate the results in a flight simulator.

4. Guidance system for optimal path and development of path for efficient search missions.

The Odin aircraft has not been on a test flight to collect data for system identification. The model parameters have therefore been unknown for the students at the Unmanned Lab, and the derivations have therefore been based on guessed values.

1.3 Contribution

The objective of this thesis is to do a literature study on optimal guidance with collision avoidance using Mixed Integer Linear Program (MILP), and implement this in order to perform simulations for different test-cases. The main motivation research the possibility of applying MILP in the guidance system on the Odin aircraft. Simplified aircraft dynamics, where the UAV is assumed to be a point mass which moves in a geographic reference frame will be shown to be a good approximation. **Main contributions to the UAV project in this thesis**:

- Implementations of obstacles are done with the assumption that every detected obstacle is a point in the geographic frame, and a safety margin is included around each obstacle to ensure that unmodeled aircraft dynamics, will not lead to collision when the real UAV uses the guidance system.

- Different ways to design the cost function in the optimization problem, to get an optimal trajectory with respect to minimum arrival time to destination, and minimum fuel consumption is discussed and implemented for different cases. The optimization of the trajectory is done for the case where initial position, final position and different waypoints, which the UAV should visit on the way from initial to final position, is given *a priori*. The UAV is required to move from an initial state to a final state, through different waypoints, without colliding with static obstacles.

- Static obstacles are included in two different ways, one where all the obstacles are known *a priori* by the guidance system, and another way where the obstacles are assumed to be observed and taken into account by the guidance system when the UAV is some certain distance from the obstacle. This is done to simulate that obstacles are detected from an UAV with a radar system which has limited range. It is also pointed out how this technique can be applied to let the guidance system take into account obstacles which moves slowly in the area.

Two approaches for efficient search of an defined area has been designed and implemented for a relevant rescue mission:

1. One of the method uses MILP optimization, and an algorithm has in this case been developed to generate waypoints such that the defined search area is covered by the UAVs camera system. The design and implementation has been done for a general case, such that an unlimited number of UAVs with arbitrary base stations can take part in the same search mission and allocate different parts of the search grid. The time the MILP optimizer needs to compute the path for the search area is important if it is urgent to start the search mission immediately. Different factor that have been found dominant for the computation time has therefore been investigated, and is closer discussed in this thesis.

2. A path planning system which searches an arbitrary area without using MILP optimization is developed, and this algorithm uses classical trigonometric relations. The equation for an *Archimedes spiral*, simplified UAV dynamics and the camera range is the parameters that are applied to generate a spiral-path for the UAV in search and rescue area. A discussion about the area

of application for the MILP approach and the spiral approach for search missions is included in this thesis.

Some discussion on how the guidance system, which is based on MILP, can be further improved with the use of *receding horizon* is pointed out at the end of this thesis.

1.4 Thesis outline

In chapter 2 of this thesis, the MATLAB interface YALMIP and the solver, Gurobi, is presented. YALMIP is used to formulate the optimization problem, while Gurobi is used to solve it. Chapter 3 describes the implementation of the aircraft dynamics and constraints which will be used in the optimization problem. This include constraints in the aircraft dynamics and implementation of constraints to make sure that waypoints are visited and that obstacles are avoided. Different approaches for finding a cost function has been implemented. Some simulations are included in this chapter to illustrate some important issues. In Chapter 4, simulations of the implemented system will be presented and the results will be further discussed. Chapter 5 describes the usefulness of UAVs in search and rescue missions, and presents a case-study which will be considered in Chapter 6 and Chapter 7. Chapter 6 and Chapter 7 describes two approaches for the guidance system for search and rescue missions. Finally 8 includes some remarks and further improvements on path planning for the UAV are pointed out in this chapter.

1.5 Main assumptions

- In this thesis it is assumed that the UAV only can navigate in the x- and y coordinates within a geographic frame, and therefore, only planar motion is considered. This means that the UAV has to fly around obstacles and can not fly over or under them. According to [Bicchi and Pallottimo] *"planar motion is common assumption because air space is structured in layers"*. If a simple radar is going to be used on the UAV for obstacle detection, this will result in a two dimensional picture, and based on this, the UAV will do planar motion. Also, with respect to computation time to solve the path problem, a two dimensional case is preferred instead of a three dimensional problem. Computation time has to be considered since the optimization problem has to be solved in real time, locally on the UAV or by a ground station which communicates with the UAV. If the problem can be solved locally on the UAV this will give benefits since this eliminates the need for ground stations in the operation area. This will be beneficial for a rescue mission out in the ocean for example.

- As described in [Fossen, 2011b] The yaw rate of the UAV can be controlled by either using the rudder, or by applying the ailerons on the UAV. If the ailerons are used to change the roll angle (ϕ) this will result in a *banked to turn*. It is in this thesis assumed that the rudders are not applied to control the yaw rate, but rather that the yaw rate of the UAV is limited by the maximum roll angle and maximum speed as will be derived in Chapter 3.

- The optimal path is found by minimizing the cost function in the optimization problem with respect to the restrictions. The cost function in this thesis has been designed to minimize the arrival time at the destination, to minimize the energy consumption of the UAV during flight, or a combination of energy and time consumption. A measurement of energy consumption has been the forces that is applied on the UAVs point mass. If time consumption was the only optimization objective in the cost function, it is obvious that this will be achieved by flying at the maximum speed, if there are no obstacles between start and the goal. However, due to drag (air friction) this will not be economical with respect to fuel consumption. Drag forces are not considered in the simulations, since these are unknown for the UAV considered.

- Static obstacles are considered, but dynamic obstacles can be included in the model by assuming them to be stationary obstacles at every time step if they are moving slowly.

- In this thesis wind forces, and other unknown forces will not be taken into account by the path planner. This is a reasonable assumption since the path-planner only uses a very simplified model of the aircraft, with a lot of uncertainties. The regulator has to be robust enough to be able to follow the path regardless of these uncertainties.

- The path planning has been done over a limited range, because *a fixed* horizon approach has been used.

Chapter 2

Guidance system applied on UAVs

2.1 A Literature Study

This section is based on [Goerzen et al., 2009] which covers a survey of motion planning algorithms for UAVs

Different methods for solving a guidance problem where the objective is to move an object through an obstacle field to a goal state is discussed in the literature. *The movers problem* solves this problem by modeling the vehicle as a rigid body, and may result in a state space model with up to 12 variables: Three position variables, three velocity variables, three orientation variables and three orientation rate angles. The configuration space has then a larger dimension than the geographic frame in which the vehicle moves. When a geographic frame is considered the position and the translation of the vehicle will be described with three position variables and the derivative of these two describes the motion.

As argued in [Goerzen et al., 2009], UAVs do not have to fit into tight spaces while flying, and therefore describing the UAV as point mass compared to a rigid body, will have little effect on the trajectory generated by the guidance algorithm. The *movers problem* therefore has more complexity than what is actually needed for UAV tasks.

UAV-motion planning is especially strikingly due to several complexities not considered by earlier planning strategies, such as vehicle constraint, and atmospheric turbulence which makes it impossible to follow a pre-computed path precisely. Uncertainty in the vehicle state and limited knowledge of the environment due to inadequate sensor capabilities are other disturbing factors. These differences have motivated an increase in use of feedback and other control engineering techniques for motion planning.

An UAV is typically modeled as having velocity and acceleration constraints, and higher order differential constraints given by the equation of motion. Also accounting for aerodynamic effects may be an important issue. The objective is to guide the vehicle towards a target through a field which may contain obstacles. In problems with differential constraints, states have to satisfy the equations of motion of the vehicle. By approximation the vehicle to by a point mass the equation of motions for UAV can be found by applying Newtons Second Law. The states

are constrained by hard limits on velocity and acceleration, and sometimes also on higher-order derivatives of position.

Different methods for trajectory planning described in the literature, considers planning strategies where differential constraints such as the equations of motions is not considered. If the object is to plot the optimal path of a car across a continent, there is no need to apply dynamic-constraints in the guidance system, since only the route is considered, not position as a function of time.

Most trajectory planning problems which is relevant for UAV applications have to consider the vehicle dynamics. The behaviour of aerial vehicles is often not sufficiently well approximated by their kinematics, which is the case for ground vehicles. Taking into account the equation of motion of an UAV, is directly relevant for guaranteeing the soundness of the planner. The equation of motion are also relevant in details of the vehicle manoeuvring affecting energy or duration of the trajectory. This class of planning problems can be expected to be more difficult to solve due to the dependency between time and the equations of motion given by the differential constraints.

Mathematical programming methods treat the trajectory planning problem as a numerical optimization problem. Popular methods described in the literature include Mixed Integer Linear Programming (MILP) and non-linear programming approaches. These methods are known as trajectory optimization methods, since they find a trajectory to a goal point that is optimal with respect to a defined cost criteria. For this type of problem, one strategy is to enforce the equation of motions as constraints.

2.2 Mixed Integer Linear Programming using YALMIP and Gurobi

In this thesis the Mixed Integer Linear Programming (MILP) algorithm will be used in finding the path for the UAV, that minimizes the time or fuel consumption, or a combination of these, from the initial position to the destination. MILP has been applied since this method makes it possible to solve non-convex problems. This makes the guidance system more robust, and gives the reader more choices when it comes to further improvements of the path planner.

Mixed integer linear programming (MILP) methods have attracted attention because of their modeling capability, and because powerful solvers are available. "*Powerful software packages such as CPLEX or Gurobi can solve MILPs efficiently for problems in which the number of binary variables is of a reasonable size. One major disadvantage of MILP is that the method is NP-hard, and therefore computational requirements can grow significantly as the number of binary variables needed to model the problem increases*" [Matthew and Raffaello, 2005]. In this thesis the optimization problem and the constraints will be written in the *YALMIP* language and the solver will be the *Gurobi* optimizer.

This section gives some background information of the *YALMIP* and *Gurobi* packages. There are also other packages available which offers the same functionality as *YALMIP* and *Gurobi*. The *AMPL* and the *CPLEX* optimizer are mentioned

a lot in the literature, but they are not considered in this thesis. The reason for choosing the *YALMIP* language and the *Gurobi* solver is because the author of thesis was first introduced to these packages.

2.2.1 YALMIP

In this thesis the optimization problem is described in the *YALMIP* language.

> "*YALMIP is a modeling language for advanced modeling and solution of convex and non-convex optimization problems. It is implemented as a free of charge toolbox for MATLAB*" Löfberg [2004].

The YALMIP variables can be written in a m-code file in MATLAB if the path to the YALMIP package has been set in MATLAB. YALMIP offers different functionalities which can be used for different purposes. In this section the functions which is applied in this thesis is presented. A list of all functionalities YALMIP can offer can be found in Löfberg [2004].

binvar() is used to define decision variables constrained to be binary (0 or 1) *YALMIP* notation:

```
x = binvar(n)
x = binvar(n,m,)
x = binvar(n,m,'type'
x = binvar(n,m,'type','field')
binvar x
```

In this thesis binary variables will be declared for many purposes, and some of them are:

- To tell if a waypoint is visited or not
- To tell if one of the four restrictions which describes obstacles are enforced or not
- To tell if one of the restriction which gives the minimum and maximum speed is enforced or not.

The sdpvar() function is used to define YALMIPs symbolic decision variables:

```
x = sdpvar(n)
x = sdpvar(n,m)
x = sdpvar(n,m,'type')
x = sdpvar(n,m,'type','field')
x = sdpvar(dim1,dim2,dim3,...,dimn,'type','field')
sdpvar x
```

Decisions variables in this thesis is applied on x- and y- position and velocity of the UAV.
Some important remarks:

- Decision and binary variables in YALMIP can be casted to MATLAB variables using the *double();* operator. This is useful since the MATLAB variables can be plotted, which makes it possible to visualize the results, which is done is this thesis.

- If the size of a matrix defined by *binvar()* or *sdpvar()* is quadratic, YALMIP will think that the matrix is diagonal. To avoid problems with this it is therefore important to include *'full'* as type element in the *sdpvar()* and *binvar() functions.*

Examples:
To give an example of the implementation of a binary variable, the binary variable which tells if a waypoint is visited or not will be used as an example. The binary variable is declared in the following way:

$$\mathrm{wp_i} = \mathrm{binvar(W, N, \ 'full')}; \tag{2.1}$$

where W is the number of waypoints which are included in the optimization problem, N is number of discrete time samples the optimization should calculate position and velocity. The 'full' notation is included to tell that the binary variable $\mathrm{wp_i}$ is not a diagonal binary matrix, but a binary matrix with full rank. If W = N the binary variable could have been declared as $\mathrm{wp_i} = \mathrm{binvar(N, N)}$. In this case the binary variable should be declared as $\mathrm{wp_i} = \mathrm{binvar(N, N, 'full')}$ such that *YALMIP* don't think that $\mathrm{wp_i}$ is a quadratic matrix with binary variables only at the diagonal.

If waypoint c is visited at time step k, then $\mathrm{wp_i(c, k)} = 1$ for this value. In some of the implementation in this thesis more than one UAV will be used for the same mission. To distinguish between the UAVs which visits the waypoints the binary variable is extended with one more dimension, such that the UAVs are included:

$$\mathrm{wp_i} = \mathrm{binvar(W, N, n_p, 'full')}; \tag{2.2}$$

where n_p is the number of UAVs which are considered in the problem.

To give an example of an implemented decision variable the following decision variable which declares the state variables for the UAV:

$$\mathrm{s_i} = \mathrm{sdpvar(4, N, 'full')}; \tag{2.3}$$

where the first argument in *sdpvar()* is used to indicate that there are four state variables: x- and y-position and velocity in x- and y- direction. The constant N indicates how many samples that need to be generated for all variables. The decision variable (2.3) can be restricted such that feasible values for the solution for x- and y- positions is limited to be inside some boundaries. This can be implemented in the following way:

$$F = F + [x_{low} \leq s_i(1,:) \leq x_{heigh}] \tag{2.4}$$
$$F = F + [y_{low} \leq s_i(2,:) \leq y_{heigh}] \tag{2.5}$$
$$\forall x_{low}, x_{heigh}, y_{low}, y_{heigh} \in \mathbb{R} \tag{2.6}$$

where F is the matrix which includes all restrictions which is a part of the optimization problem. The decision variables for the x- value has been restricted to be lower and upper bounded by x_{low} and x_{heigh} respectively. The y- position has been restricted to be lower and upper bounded by y_{low} and y_{heigh} respectively. It is good programming convention to insert upper bounds on the decisions variables to make the decision space smaller, which again can have a positive effect on the computational complexity.

2.2.2 Gurobi

The Gurobi solver is used to solve the mixed-integer linear program (MILP), which in this thesis is written in the YALMIP language.

> *"The Gurobi Optimizer is a state-of-the-art solver of linear programming (LP), quadratic programming (QP) and mixed-integer programming (MILP and MIQP)."* as described in [Gurobi, 2011].

The Gurobi optimizer is deterministic, which means that two separate runs on the same model results in identical solution paths.

The Gurobi solver is in this thesis downloaded as a free academic license. The terms of agreement for the academic license is as follows [Gurobi, 2011]:

- They can only be used by faculty, students, or staff of a degree-granting academic institution.
- It can only be used for research or educational purposes.
- They must be validated from a recognized academic domain.

The Gurobi optimizer was called in the in MATLAB with the following line:

```
diag = solvesdp(F,J,sdpsetting('solver',gurobi));
```

Where F is the array with the constraints, and J is the cost function.
Table A.1 in Appendix A shows the supported platforms for Gurobi Optimizer 4.5.

2.2.3 Gurobi mex and c++ compiler

A C/C++ compiler for the for the actual MATLAB copy needs to be downloaded an installed on the computer which is applied for the simulations. For the simulations done in this thesis, Visual Studio 2012 was used. Also *Gurobi Mex*, which is a MATLAB interface for Gurobi needs to be downloaded on the computer. More details about the installation can be found in mex [2012].

Chapter 3

Implementation of Dynamics and Constraints

In this chapter the model dynamics and the constraints that will be considered for the UAV by the guidance system, will be derived. It will further be explained how these dynamics can be described and implemented in the Mixed Integer Linear Program (MILP). An approximate model of the aircraft dynamics in a geographic reference frame using only linear constraints is developed, enabling the MILP approach to be applied as the guidance system with collision avoidance. The equation of motion in the geographic reference frame will be derived from the equations of motion in the Body frame and then transformed to discrete form. Assumptions on maximum UAV speed, maximum roll angle, and lift coefficient will be done to be able to model constraints in yaw rate, under the assumption that lift force equals gravity force. The equation for drag force will be shown, but the drag will not be included in the model due to little knowledge about the effect it will have on the UAV. It will be discussed how physical constraints in the actuator dynamics can be modeled, and how minimum speed can be implemented as an constraint in the problem. A lower restriction on the speed is an important restriction to keep the UAV up in the air. It can, however, be challenging to implement this restriction since it will depend on the current situation of the aircraft. During take off and landing the UAV should be allowed to stand still (zero speed). During the flight, the lower restriction on the speed has to be physically justified such that the aircraft does not stall. During the flight the lower restriction on the speed can obvious not be zero. Implementation of waypoints, obstacles and strategies for design of the cost function will be presented for a fixed horizon path planner.

3.1 Equation of motion in geographic coordinate frame

For a guidance problem it is interesting to describe the equations of motions in the NED frame or another geographic coordinate system as the *tangent frame (t-frame)* for instance. The t-frame is fixed at the Earth's surface with the x-axis pointing to the north and the y-axis pointing to the east, and the z-axis pointing downward perpendicularly to a local reference ellipsoid. The reference ellipsoid is more accurate than the global ellipsoid in the area of interest. This makes is possible to relate GPS and INS measurements to the actual terrain and local maps as described in [Vik, 1999]. Therefore, it is probably beneficial to use a *t-frame*

for the case with guidance and collision avoidance of the UAV in a local area, and include GPS and INS measurements. In the following derivations, the NED frame is considered, but the geographic frame can easily be changed to another frame by the reader.

3.1.1 Equation of motion in continuous form

The equations of translational motions about the *Center of Gravity* (CG) expressed in the body $\{b\}$ frame of the UAV, can in the three dimensional space be written as (based on equation (3.16) in [Fossen, 2011a]):

$$m\left(\dot{\boldsymbol{v}}_{g/n}^{b}+\boldsymbol{\omega}_{b/n}^{b}\times\boldsymbol{v}_{g/n}^{b}\right)+\boldsymbol{D}\boldsymbol{v}^{b}=m\boldsymbol{R}^{T}\boldsymbol{g}^{n}+\boldsymbol{\tau}^{b}+\boldsymbol{\tau}_{wind}^{b}+\boldsymbol{\tau}_{other}^{b} \tag{3.1}$$

where m is the mass of the UAV, $\boldsymbol{D}\in\mathbb{R}^{3x3}$ is a positive definite damping matrix, and is further assumed to be diagonal. $\boldsymbol{g}^{n}\in\mathbb{R}^{3}$ is the gravity vector in the NED frame, $\boldsymbol{v}^{b}\in\mathbb{R}^{3}$ is the velocity in body frame, $\boldsymbol{\tau}^{b}\in\mathbb{R}^{3}$ is the control inputs on the aircraft which affects the translational motions. $\boldsymbol{\tau}_{wind}\in\mathbb{R}^{3}$ is forces on the aircraft body affected by the wind, and $\boldsymbol{\tau}_{other}\in\mathbb{R}^{3}$ is all other forces that affects the UAV. $\boldsymbol{R}=\boldsymbol{R}_{b}^{n}\in\mathbb{R}^{3x3}$ is the rotation matrix from the body frame to the NED frame.

Using the skew-symmetric matrix, (3.1) this can be written as:

$$m\left(\dot{\boldsymbol{v}}_{g/n}^{b}+\boldsymbol{S}(\boldsymbol{\omega}_{g/n}^{b})\boldsymbol{v}_{g/n}^{b}\right)+\boldsymbol{D}\boldsymbol{v}^{b}=m\boldsymbol{R}^{T}\boldsymbol{g}^{n}+\boldsymbol{\tau} \tag{3.2}$$

where the property $\boldsymbol{S}(\boldsymbol{\omega}_{g/n}^{b})v_{g/n}^{b}=\boldsymbol{\omega}_{b/n}^{b}\times\boldsymbol{v}_{g/n}^{b}$ has been used. In the following, the derivation will be based on (3.1),

The derivative of the velocity component in the NED frame can be written as:

$$\dot{\boldsymbol{v}}^{n}=\boldsymbol{R}\left(\dot{\boldsymbol{v}}^{b}+\boldsymbol{\omega}^{b}\times\boldsymbol{v}^{b}\right) \tag{3.3}$$

Inserting (3.3) in equation (3.1) gives:

$$m\boldsymbol{R}^{T}\dot{\boldsymbol{v}}^{n}+\boldsymbol{D}\boldsymbol{R}^{T}\boldsymbol{v}^{n}=m\boldsymbol{R}^{T}\boldsymbol{g}^{n}+\boldsymbol{R}^{T}\boldsymbol{\tau}^{n}+\boldsymbol{R}^{T}\boldsymbol{\tau}_{wind}^{n}+\boldsymbol{R}^{T}\boldsymbol{\tau}_{other}^{n} \tag{3.4}$$

By multiplying (3.4) with the rotation matrix $\boldsymbol{R}$ from the left, this gives:

$$\boldsymbol{R}m\boldsymbol{R}^{T}\dot{\boldsymbol{v}}^{n}+\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^{T}\boldsymbol{v}^{n}=\boldsymbol{R}m\boldsymbol{R}^{T}\boldsymbol{g}^{n}+\boldsymbol{R}\boldsymbol{R}^{T}\boldsymbol{\tau}^{n}+\boldsymbol{\tau}_{wind}^{n}+\boldsymbol{\tau}_{other}^{n} \tag{3.5}$$

Since m is a scalar, the equation of motions in the NED frame becomes:

$$m\dot{\boldsymbol{v}}^{n}+\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^{T}\boldsymbol{v}^{n}=m\boldsymbol{g}^{n}+\boldsymbol{\tau}^{n}+\boldsymbol{\tau}_{wind}^{n}+\boldsymbol{\tau}_{other}^{n} \tag{3.6}$$

where the property of a rotation matrix, $\boldsymbol{R}\boldsymbol{R}^{T}=1$ has been used.

By dividing by m on both sides, (3.6) can be written as:

$$\dot{\boldsymbol{v}}^{n}=-\frac{1}{m}\boldsymbol{\Omega}\boldsymbol{v}^{n}+\boldsymbol{g}^{n}+\frac{1}{m}\boldsymbol{\tau}^{n}+\frac{1}{m}\boldsymbol{\tau}_{wind}^{n}+\frac{1}{m}\boldsymbol{\tau}_{other}^{n} \tag{3.7}$$

where

$$\boldsymbol{g}^{n}=[0\ 0\ -g]^{T} \tag{3.8}$$

is the gravity vector in the NED frame, and

$$\boldsymbol{\Omega} := \boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T \tag{3.9}$$

is a positive definite damping matrix.
As described an argued in Section (1.5), wind forces and other unknown forces are not included in the path planner, and therefore $\boldsymbol{\tau}^{\boldsymbol{n}}_{\boldsymbol{wind}} = 0$ and $\boldsymbol{\tau}^{\boldsymbol{n}}_{\boldsymbol{other}} = 0$.

Since the $\boldsymbol{g}^n$ term in (3.7) only effects the system in the z-direction in the geographic frame, it can be removed from the equation. Equation (3.7) can be written in the following way, by considering x- and y- motion in the geographic frame only. In the following, all symbols are given in the NED frame.

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} = \frac{1}{m}\left(-\boldsymbol{\Omega}^T \begin{bmatrix} v_x \\ v_y \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix} \right) \tag{3.10}$$

Using the first and the second derivative of position, (3.10) can be written in the following way for planar motion:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \frac{1}{m}\left(-\boldsymbol{\Omega}^T \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \right) + \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix} \tag{3.11}$$

$$\dot{\boldsymbol{v}} = -\frac{1}{m}\boldsymbol{\Omega}\boldsymbol{v} + \frac{1}{m}\boldsymbol{\tau} \tag{3.12}$$

$\boldsymbol{D}$ is a 2x2 diagonal damping matrix which will depend on the aerodynamic coefficients of the UAV. It can be assumed that the damping matrix is diagonal, meaning that motion in x-direction only will give drag force in the x-direction and motion in y-direction only will give drag in y-direction.

$$\boldsymbol{D} = \begin{bmatrix} X_u & 0 \\ 0 & Y_v \end{bmatrix} \tag{3.13}$$

To model drag when assuming a point mass that moves in a geographic frame, a suggestion is to set $X_y = X_x$ where X_x is the drag force that affects the aircraft in the body frame.

In this thesis, optimal path planning and collision avoidance is the main focus, and for simplicity the equations of motion is modelled without the damping term, that is $\boldsymbol{D} = 0$. This is of course not physically justifiable since, in this case, the vehicle never will stop by itself, if the initial velocity is different from zero, and it has to use an external force to stop. The following Lyapunov function is used to analyze the stability properties:

$$V(t) = \frac{1}{2}\boldsymbol{v}^T\boldsymbol{v} > 0, \qquad \forall \boldsymbol{v} \neq 0 \tag{3.14}$$

where $\boldsymbol{v} \in \mathbb{R}^2$. V(t) is positive definite, and it is zero only when $v = 0$ The derivation of (3.14) is found to be:

$$\dot{V}(t) = \boldsymbol{v}^T\dot{\boldsymbol{v}} = \boldsymbol{v}^T(-\frac{1}{m}\boldsymbol{\Omega}\boldsymbol{v} + \frac{1}{m}\boldsymbol{\tau}) = -\boldsymbol{v}^T\frac{1}{m}\boldsymbol{\Omega}\boldsymbol{v} + \frac{1}{m}\boldsymbol{v}^T\boldsymbol{\tau} \tag{3.15}$$

As can be seen from (3.15) the derivative of the unforced system ($\tau = 0$) is:

$$\dot{V}(t) = -\boldsymbol{v}^T\frac{1}{m}\boldsymbol{\Omega}\boldsymbol{v} \Rightarrow \dot{V}(t) < 0 \;\; \forall \boldsymbol{v} \neq 0 \tag{3.16}$$

The system is in this case globally exponentially stable (GES) [Khalil, 2002].

If damping is removed from the equation ($\boldsymbol{\Omega} = 0$), the Lyapunov derivative (3.15) will be:

$$\dot{V}(t) = \frac{1}{m}\boldsymbol{v}^T\boldsymbol{\tau} \tag{3.17}$$

In this case stability will depend on the chosen control force. This has to be designed such that stability is achieved in a Lyapunov sense. It can be designed such that the GES property is achieved for example by requiring that $\boldsymbol{\tau} = -\boldsymbol{v}$

When assuming no damping, (3.11) can be written as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & \frac{1}{m} \end{bmatrix} \begin{bmatrix} f_x \\ f_y \end{bmatrix} \tag{3.18}$$

where the state vector from from (3.11) has been extended to include x- and y position as well.

Equation (3.18) can be written as:

$$\dot{s} = As + Bu \tag{3.19}$$

where:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.20}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & \frac{1}{m} \end{bmatrix} \tag{3.21}$$

where $s := [x\ y\ \dot{x}\ \dot{y}]^T$.

Remark: This equation could also have been found by applying Newtons Second Law on a point mass. The derivation of the equation of motion from the body frame to the geographic frame proves that the UAV can be viewed as a point mass in the geographic frame. The input vector in the geographic frame will be bounded by the inputs applied on the UAV and the kinetics of the UAV.

3.1.2 Comments

In this thesis it is assumed that there is no damping that affects the UAV. This is of course not a realistic case, but the main goal for this thesis is to develop an optimal path planner. The system equation and constraints can be further developed when the real parameters and constraint to the UAV are known due to parameter estimation.

However, the damping will be described here such that parameters that need to be found, to include damping in this guidance system, during parameter estimation of the UAV, is known for the reader. Stevens and Lewis [2003] describes

damping that affects an aircraft, and a simple form of this included here. All variables are given in *wind-axes*.

The drag force (D) that affects an aircraft can be described as:

$$D = \overline{q} S C_D \tag{3.22}$$

where $\overline{q}$ is the dynamic pressure, S is the total wing area and C_D is the drag coefficient, which will be a function of several variables: Angle of attack (α), Mach number (M) and the elevation to mention some. The dynamic pressure is written as:

$$\overline{q} = \frac{1}{2}\rho V_T^2 \tag{3.23}$$

where ρ is the density of air and V_T is the speed in wind-axes [Fossen, 2011b] .

Remark: One important remark which can be viewed from (3.23) is that the drag increases quadratic with the speed. When implementing drag in the UAV's equation of motion the equation will be non-linear. The Mixed Integer Linear Programming (MILP) method which is applied in this thesis is, as the name tells, an optimization technique which describes optimization problems with only linear restrictions. When implementing the drag to the equations of motion the non-linear terms has to be "removed" from the equations. The author of this thesis suggests to linearize the the drag about some "working-speed" and apply the linearized drag in the equations. Another proposal is to use feedback linearization so that the input move the non-linearities that comes in the equations of motion because of the drag.

3.1.3 Equations of motion in discrete form

The equations of motion in continuous form that was given in (3.19) need to be transformed to discrete form to be implemented on a computer. The discrete dynamics of the UAV can be written as:

$$\forall i \in [0, \ldots, T]$$

$$\boldsymbol{s}_{i+1} = \boldsymbol{A}_d \mathbf{s}_i + \boldsymbol{B}_d \boldsymbol{u}_i \tag{3.24}$$

$$\mathbf{s}_0 = \boldsymbol{s}_I$$

where $\boldsymbol{A}_d$ is the discrete system matrix and $\boldsymbol{B}_d$ is the discrete input matrix, $\mathbf{s}_i$ is the discrete state vector for each time-step i, and $\mathbf{s}_I$ is the initial state of the UAV.

The continuous time equation in (3.18) can be written in the discrete form [Chen, 1999]:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ v_{x,k+1} \\ v_{y,k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_d & 0 \\ 0 & 1 & 0 & T_d \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ v_x \\ v_y \end{bmatrix} + \begin{bmatrix} \frac{(T_d)^2}{2m} & 0 \\ 0 & \frac{(T_d)^2}{2m} \\ \frac{T_d}{2m} & 0 \\ 0 & \frac{T_d}{2m} \end{bmatrix} \begin{bmatrix} f_{x,k} \\ f_{y,k} \end{bmatrix} \tag{3.25}$$

$$\begin{bmatrix} x(0) \\ y(0) \\ v_x(0) \\ v_y(0) \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ v_{x,0} \\ v_{y,0} \end{bmatrix}$$

In the implementation in MATLAB, the discretization is done with the MATLAB command $[Ad, Bd] = c2d(A, B, T_d)$, as described in [Chen, 1999]. A and B are the continuous A and B matrices, and T_d is the discretization time. In this thesis the discretization time T_d has been guessed to be 3 seconds.

3.2 Modelling constraints in yaw rate

In the geographic frame, the forces that effects the vehicle can be viewed as forces that affects a point mass in the x- and the y directions, according to Newtons second law. However, the UAV can not move as a point mass, because of its kinetics. The UAV has to use either the *ailerons* to change the roll angle which will give a *banked turn* because of the centripetal force, which will point towards the given turning circle, or use the rudder to set up yaw moment about body the z-axis, which will change the direction of the force vector from the engine and result in a changed velocity vector of the UAV in the geographic frame.

In the following, only yaw rate caused by using the *aileron* will be considered, because it can be derived quite easy by physical consideration of the UAV.

3.2.1 Yaw rate caused by the use of Aileron

When considering the use of *aileron* to give a *banked turn*, the change in roll angle results in a decomposed lift vector, one which will point in the opposite direction of the gravity vector, and the centripetal force which will point towards the center of the turning circle (in the x-y plane in the geographic frame), which the UAV turns around. The maximum value of the centripetal force for the UAV, can be found if the maximum speed, the aerodynamic lift force and the maximum roll angle is known. The maximum absolute value of the centripetal force can be used to give an indication of the maximum turning rate of the UAV. The greater the centripetal force is, the greater will the turning rate be.

Information of the maximum turning rate can be included as a constraint to give restrictions on the forces that the path planner is allowed to apply on the modelled point mass, and can justify the modelling of the UAV as a point mass in the MILP optimization problem.

Figure 3.1 shows a plot of the path when there are no restrictions on the forces that can be applied to the UAV.

As can be seen from the figure, the corners are sharp and it is not guaranteed that the path will be feasible for the UAV. Since the modelling of the UAV is done by considering a point mass (which does not rotate in the geographic frame) the kinetics of the UAV is not included in the model. Allocations of inputs to the UAV in body frame relative the geographic reference frame is lost when transforming from body frame to a geographic frame. Therefore, the restrictions on the forces applied on the point mass, which the path planner will consider, has to be upper bounded by the maximum centripetal force, to ensure a feasible turning rate. Before one inserts limitations on the applied forces in the guidance

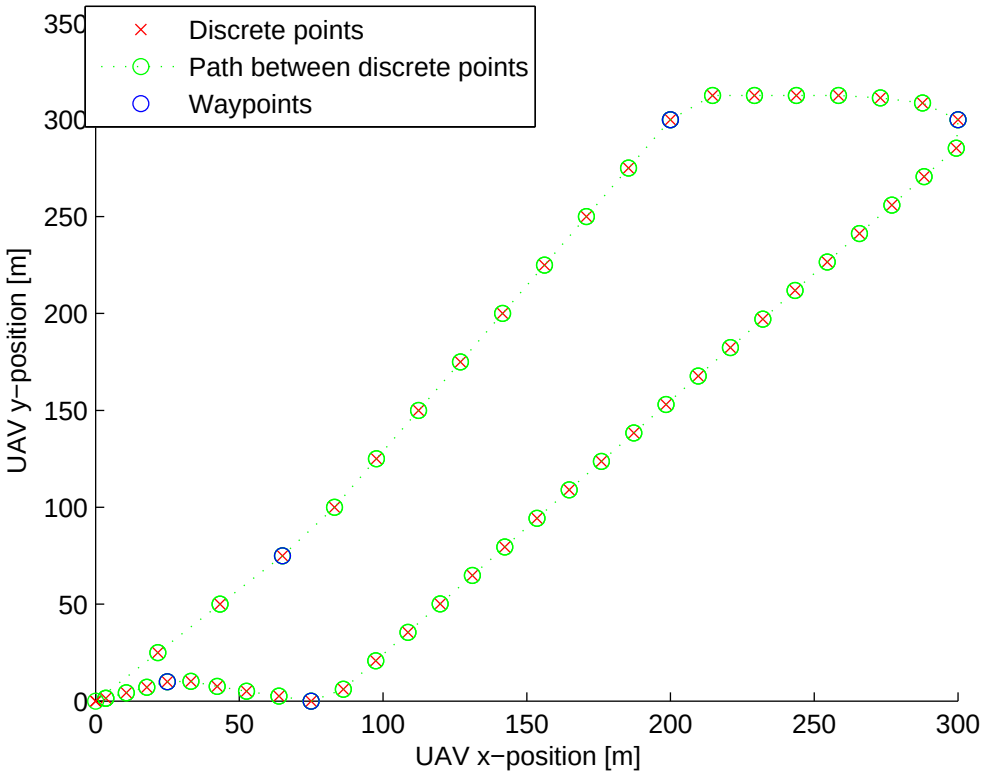

Figure 3.1: Infeasible trajectory for the UAV. There are no constraints on the forces in the x- and y direction. The corners are not feasible because limitations in turning rate is not considered.

system, the forces that affects the UAV has to be analyzed in more details. There are two situations that has to be considered before one conclude on the maximum force which should be implemented in the model:

1. The maximum centripetal force is bigger than the maximum force from the propeller, which moves the UAV forward. The maximum centripetal force can be found at the maximum speed and at the maximum roll angle (ϕ_{max}).

2. The maximum force from the propeller is found to be bigger than the maximum centripetal force. The maximum force from the propeller is given by the engine and propeller parameters and the angular speed of the propeller.

Before one conclude on which value that should be considered as the maximum value for the forces in the model, one should find out which of the two situations above that is correct.

1. If the absolute value of the centripetal force that effects the UAV is smaller than the maximum force from the engine, this value should be included in the model. This will however also represent an unwanted upper restriction on the forces that can be applied by the propeller. This will be problematic if the UAV is flying along a straight line and if it is preferable to use as much power from the propeller as possible. For this cases a bound on the force to the maximum centripetal force, will be the bound on the force

from the propeller as well, since the bound on the force represent a bound to all forces which affects the UAV by actuators. The guidance system do not distinguish between if the UAV is flying along a straight line (power from the engine only), or if it is flying along a curvature (power from both engine and the centripetal force). This can give problems if the maximum force from the engine is much bigger than the maximum centripetal force that the real UAV is affected by. This is because the the path planner will put restrictions on the engine which is not desirable. An example of this can be if the optimization problem is to minimize the time consumption from start to goal along a straight line where there is no need to turn the UAV.

2. If the maximum force from the engine is found to be smaller than the maximum centripetal force, the maximum engine force has to be the force constraint applied by the path planner, to ensure a feasible path when only motion from start to goal along a straight line is considered. In this case the turning rate will get an unwanted upper restriction, which in more restrictive that the physical constraint on the UAV. This is not beneficial, since the restriction on the turning rate becomes more restrictive then the physical restrictions in given by the UAV dynamics. But, also here, since the model does not distinguish between if the UAV is flying in a curvature or along a straight line this restriction needs to be included.

It is important that the lowest force value between all the forces that effects the UAV in the planar motion area, are implemented in the model. It is more important that the guidance system will generate a feasible path, than it is to minimize time consumption by utilizing the actuators full potential. Running actuators at full speed over long time may also cause excessive wear and tear on the actuator components.

In the following sections in this thesis, it will be assumed that the absolute value of the centripetal force is smaller then the force from the engine, and therefore has to be the limited force in the path planner.

When the roll angle is different from zero ($\phi \neq 0$), the lift vector can be decomposed in two vectors, one which will be parallel to the z-axis in the geographic frame, opposite of the gravity vector (which will keep the UAV up in the air), and one vector perpendicular to the geographic z-axis which will point in the x-y plane in the geographic frame, which will give centripetal acceleration. In the following, the expression for this force will be derived.

If the UAV is flying with constant velocity, with constant altitude and the roll angle is zero ($\phi = 0$), the forces that affects the UAV, can be simplified to only consider the gravity force, the lift force and the drag force. The lift force will be further discussed in the following section.

According to [Stevens and Lewis, 2003] the lift force on an aircraft can be written as:

$$L = \frac{1}{2}\rho V_T^2 S C_L(\alpha) \tag{3.26}$$

where ρ is the density of air, V_T is the speed of the UAV measured in wind-axes and C_L is the lift coefficient, which will among other factors, be dependent of the *angle of attack* .

In the following calculations the forces from the wind will be neglected ($\tau_{wind} = 0$), as described earlier in this chapter, and the velocity vector of the UAV is assumed to be along the body x-axis. We can then write $V = V_T$, where V is the speed of the UAV in body frame. Therefore (3.26) can be written as:

$$L = \frac{1}{2}\rho V^2 S C_L(\alpha) \tag{3.27}$$

In (3.27) the coefficients apart from $C_L(\alpha)$ can be found by calculating the area of the wing on the UAV and by assuming a speed and that roll angle equals zero, such that the lift force equals the gravity force, $L = mg$. $C_L(\alpha)$ can then be found from

$$C_L(\alpha) = 2 \cdot \frac{L}{\rho V^2 S} = 2 \cdot \frac{9.81 \cdot m}{1.225 \cdot V^2 \cdot S} = \frac{16 \cdot m}{V^2 \cdot S} \tag{3.28}$$

The largest possible value for V should be used in the calculations to guarantee that the yaw rate is always feasible. A large speed will give a larger turning circle, which implies that the yaw rate decreases with an increased speed. In the following, the values for the total wing-surface S for the Odin delta-wing, assumed by [Dønnestad, 2011] will be used in the calculations to find a value of the maximum centripetal force, $f_{c,max}$.

In [Dønnestad, 2011] the following values was assumed:

$$V_{max} = 25\left[\tfrac{m}{s}\right],\ S = 2 \cdot 0.3815\,[m^2]$$

In the following derivation, it is assumed that the UAV is flying with a speed, $V = 20\left[\tfrac{m}{s}\right]$ before the *banked turn*. It is further assumed that the UAV will increase the speed during the bank turn to keep the altitude constant, and that the maximum roll angle is, $\phi_{max} = 10°$.

The value of $C_L(\alpha)$ before the *banked turn* can then be calculated from (3.28).

$$C_L(\alpha) = \frac{16 \cdot 2.8}{20^2 \cdot 2 \cdot 0.3815} \approx 0.15 \tag{3.29}$$

If it can be assumed constant altitude during the *banked-to-turn*, the following derivation to find f_{max} can be done. The assumption can be realistic if the speed of the UAV increases when the roll angle is changed.

Lift before *banked turn* starts, $\phi = 0$:

$$L = \frac{1}{2}\rho V^2 \cdot S \cdot C_L\,(0) \tag{3.30}$$

Lift when the roll angle is different from zero:

$$L_\phi = \frac{1}{2}\rho V_\phi^2 \cdot C_L(\phi) \cdot \cos\,(\phi) \tag{3.31}$$

If it can be assumed that $C_L(\phi) \approx C_L$, then for $L_\phi = L$ to be valid when $\phi = 10°$, (3.30) and (3.31) gives

$$\frac{1}{2}\rho V^2 \cdot S \cdot C_L = \frac{1}{2}\rho V_\phi^2 S \cdot C_L \cdot \cos(\phi) \tag{3.32}$$

which again gives that

$$V_\phi = \frac{V}{\sqrt{\cos(\phi)}} \tag{3.33}$$

$$V_{10}\frac{20}{\sqrt{\cos(10)}} = 20.15\left[\frac{m}{s}\right] \tag{3.34}$$

The maximum centripetal force ($f_{c,max}$) can then be found by:

$$f_{c,max} = L\cos(\phi_{max}) = \frac{1}{2}\rho V_\phi^2 C_L(\phi_{max})\cos(\phi_{max}) \tag{3.35}$$

By inserting the equation for V_ϕ in (3.33) into (3.35) gives the following equation for the centripetal force.

$$f_{c,max} = \frac{1}{2}\rho(\frac{V}{\sqrt{\cos(\phi_{max})}})^2 C_L(\phi_{max})\sin(\phi_{max}) = V^2 C_L \tan(\phi_{max}) \tag{3.36}$$

Inserting for ϕ_{max} gives

$$f_{c,max} = 20^2 \cdot 0.15\tan(10°) = 10.5N \tag{3.37}$$

This calculated force is unreasonable high, and the guessed values of lift, speed, surface area and aerodynamic values are in this thesis concluded to be unreasonable assumptions.

Since the UAV parameters are not estimated, no reasonable values for the forces that affects the UAV can be calculated exactly. Further in this thesis it will be assumed that the maximum force that effects the UAV is 1.5N. Figure 3.2 shows simulations of the trajectory when the restriction of the absolute value of applied forces is restricted to be less than or equal to 1.5N.

3.2.2 Comments

Since this thesis considers a point mass which travels in a geographic frame, the equations of motion in the body frame are not a part of the optimization problem. Therefore, it is not known if the forces applied comes from the engine or as a resulting centripetal force, because of roll angle. Therefore the forces from the engine also has to be limited to 1.5N in the optimization problem, as described above.

According to [Stevens and Lewis, 2003], the forces from a propeller-engine in the longitude direction can be written as:

$$f_t = \frac{1}{2}C_T\rho|n|nD^4 \tag{3.38}$$

where C_T is an engine-constant ρ, is the density of air, n is the rotation velocity of the propeller in revolution per minute $[\frac{rev}{min}]$ and D is the diameter of the propeller.

According to [Dønnestad, 2011] the parameters in his work has been chosen to be:

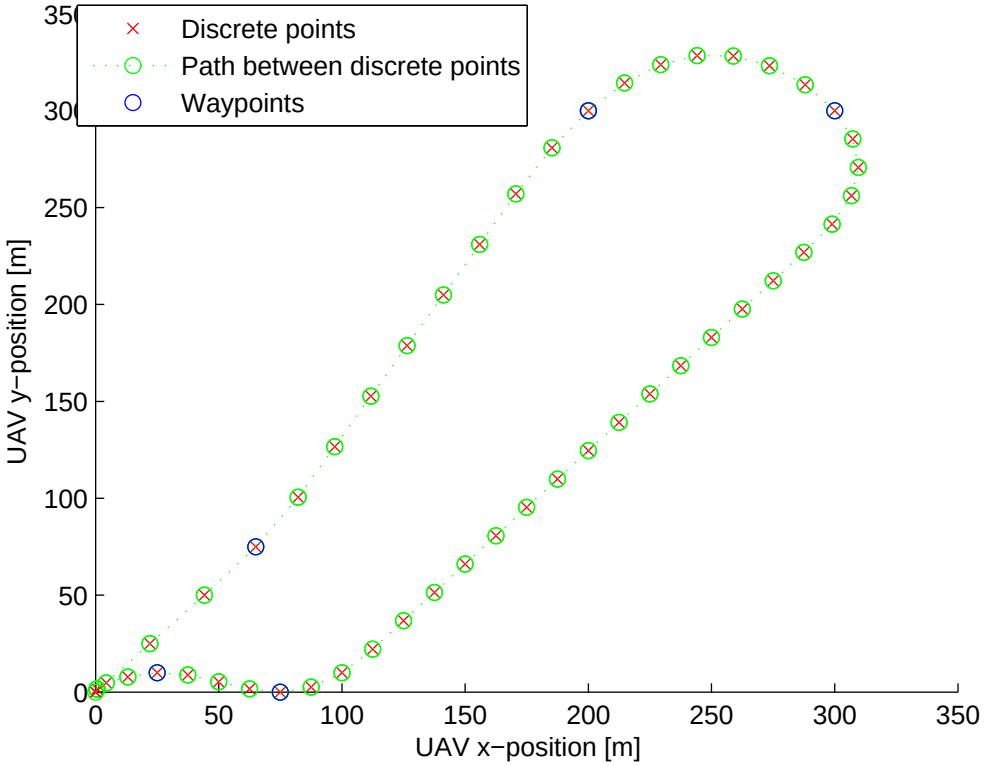

Figure 3.2: Feasible trajectory for the UAV. Constraints on the force is implemented

$$C_T = 0.033\left[\frac{m^2}{s^2}\right],\ \rho = 1.2928\left[\frac{kg}{m^3}\right],\ D = 0.3[m].$$

The values in [Dønnestad, 2011] for the angular velocity for the propeller have been around $180\left[\frac{rev}{min}\right]$.

If one insert the engine parameters and the propeller speed from [Dønnestad, 2011] in (3.38) the following value for the force which affects the UAV from the propeller is calculated:

$$f_t = \frac{1}{2} \cdot 0.033 \cdot 1.2928 \cdot |180|180 \cdot 0.3^4 = 11.19N \tag{3.39}$$

When considering the Recce delta-wing aircraft this can obvious not be correct, since this will result in an acceleration of 0.4g when the mass is 2.8 kg and the damping is neglected! Better measurements of the propeller and system identification has to be performed to identify the dominating forces that affects the UAV. The calculation of maximum centripetal acceleration which was derived in this section needs to be redone, when correct UAV parameters has been derived. The parameters used further in this thesis is just bases on guesses.

It is however assumed that for some cases, trajectory will not give any unwanted limitations to the engine, since when considering minimum fuel consumption it will be suitable to use as small engine force as possible. This will not be the case, however, when the objective of the optimization problem is to minimize the arrival time to destination, since then it is natural to assume that the forces will be in the upper bound of the force restriction.

3.3 Implementation of velocity- and force-constraints

Both the velocity and the forces acting on the UAV will have magnitude constraints. As described in last section, the force-constraint was assumed to be 1.5N to get a feasible path with respect to both the curvature path and the straight line path. Since the orientation of the UAV is not considered in the geographic frame, the forces that affects the UAV in $\{b\}$ frame has to be transformed to the $\{n\}$ frame. This means that the absolute value of the forces (f_x and f_y) in $\{b\}$ should be less than 1.5 N when the transformation to the $\{n\}$ frame has been performed. In a two dimensional coordinate system, where the axes are f_x and f_y, this constraint will enclose a circle on the X-Y plane with radius 1.5. This restriction can be mathematically described as:

$$f_x^n = f\cos(\omega) \qquad \forall\omega \in [0, 2\pi] \tag{3.40}$$

$$f_y^n = f\sin(\omega) \qquad \forall\omega \in [0, 2\pi] \tag{3.41}$$

$$f_x^2 + f_y^2 \leq f_{max}^2 \tag{3.42}$$

Inserting for f_x and f_y from (3.46) and (3.47) gives

$$f^2\cos^2(\omega) + f^2\sin^2(\omega) \leq f_{max}^2 \tag{3.43}$$

$$f^2(\cos^2(\omega) + \sin^2(\omega)) \leq f_{max}^2 \tag{3.44}$$

$$|f| \leq f_{max} \tag{3.45}$$

where the identity $\cos^2(\omega) + \sin^2(\omega) = 1$ has been applied.
The restriction for the maximum speed can be described in the same way

$$v_x^n = V\cos(\omega) \qquad \forall\omega \in [0, 2\pi] \tag{3.46}$$

$$v_y^n = V\sin(\omega) \qquad \forall\omega \in [0, 2\pi] \tag{3.47}$$

$$v_x^2 + v_y^2 \leq V_{max}^2 \tag{3.48}$$

$$V^2\cos^2(\omega) + V^2\sin^2(\omega) \leq V_{max}^2 \tag{3.49}$$

$$V^2(\cos^2(\omega) + \sin^2(\omega)) \leq V_{max}^2 \tag{3.50}$$

$$|V| \leq V_{max} \tag{3.51}$$

The representations in (3.45) and (3.51) are non-linear and can therefore not be implemented in MILP directly. One simple way to approximate this to linear constraints is to use magnitude limit on each element, which will restrict the forces within a square inside the circle [Richards and How, 2002].

A simple linear magnitude limit on the the forces and the velocity in both x- and y- direction restricts the values within the square inside the circle.

This restrictions can be written as:

$$|f_{xi}| \leq f_{max} \quad and \quad |f_{yi}| \leq f_{max} \tag{3.52}$$

$$|v_{xi}| \leq V_{max} \quad and \quad |v_{yi}| \leq V_{max} \tag{3.53}$$

where f_{xi} and f_{yi} is the force and, along the x- and y axes respectively for each time-step, in the geographic frame. v_{xi} and v_{yi} is the velocity in x- and y- direction respectively.

As shown in Figure 3.3, it can be see that this is a bad approximation. As described in [Richards and How, 2002] a better way to do this is to use M number of constraints to get a better approximation. For both velocity and force constraints, these are given by:

$$\forall i \in [0, \ldots, N-1] \, \forall m \in [1, \ldots, M]$$

$$f_{xi} \sin\left(\frac{2\pi m}{M}\right) + f_{yi} \cos\left(\frac{2\pi m}{M}\right) \leq f_{max} \tag{3.54}$$

$$\forall i \in [1, \ldots, N] \, \forall m \in [1, \ldots, M]$$

$$V_{min} \leq v_{xi} \sin\left(\frac{2\pi m}{M}\right) + v_{yi} \cos\left(\frac{2\pi m}{M}\right) \leq V_{max} \tag{3.55}$$

In this thesis, the maximum velocity (v_{max}) is set to be 20 $\left[\frac{m}{s}\right]$ and the minimum velocity (v_{min}) is further guessed to be 5 $\left[\frac{m}{s}\right]$. However, in some of the implementations in this thesis the minimum velocity v_{min} has been set to zero, because it is assumed in some missions that the UAV starts and stops in it's base station where the speed is zero. In implementation on an UAV the minimum speed during flight should be set to a minimum value $V_{min} > 0$ such that it do not loose elevation or stall. During take-off and landing, it should be allowed to have zero speed. The best way to solve this problem is probably to have a guidance system which only considers take-off and landing, and the path planner developed in this thesis only consider the path between take-off and landing.

The approximation to the actual constraint consists of a set of linear approximations. The number M defines the number of linear constraints, and in this thesis, M = 10.

Figure 3.3 shows the the actual non-linear constraint, and two linear approximations using 4 and 10 linear constraints. The approximation using 4 linear constraints is the *square approximation*. The linear approximation using 4 constraints is the constraint given by equation (3.52), and the M constraint approach is given by (3.54). As can be seen from Figure 3.3, the square approximation is a bad approximation compared to the approximations using 4 and 10 linear constraints. The more linear restrictions that is used the better will the approximation to the actual non-linear restriction be. As $M \to \infty$ the approximation in (3.54) will approximate the real non-linear constraint. As M gets bigger, there will be many constraints but the approximation will be more accurate. According to [Richards and How, 2002] this will have little effect on the computational complexity since they include only linear variables. However the author of this thesis does not agree with this statement, since more linear variables will result in more binary variables for each time-step, and the number of binary variables will be important for the

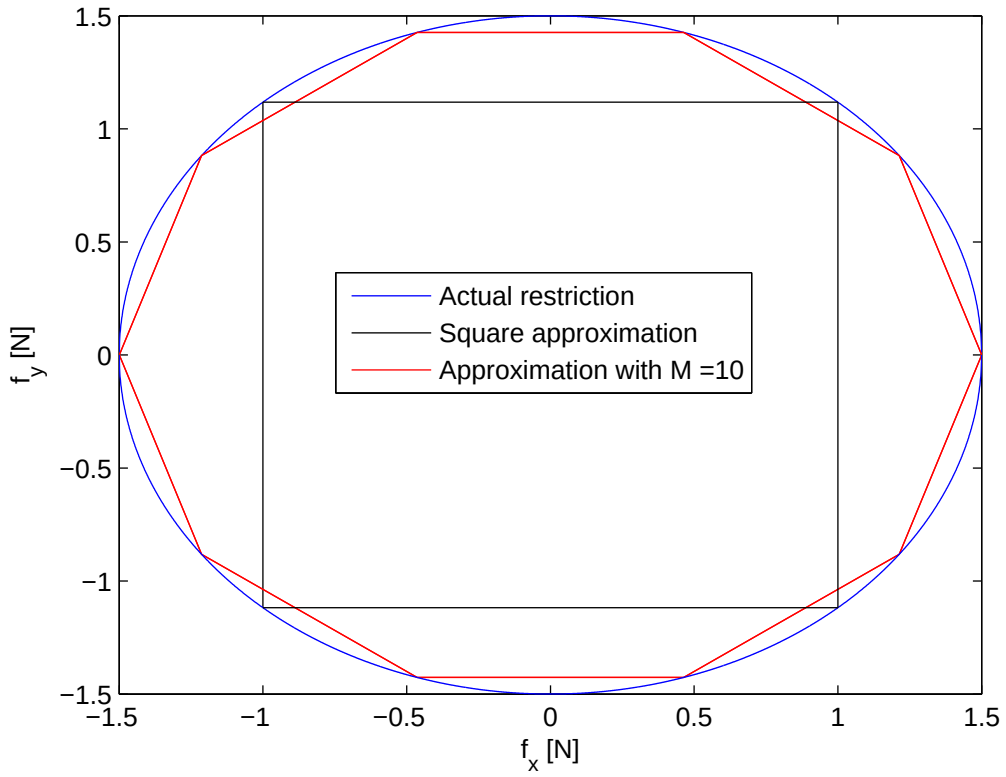

Figure 3.3: Illustration: The actual restriction is plotted against the two approaches for linear approximation to the actual non-linear restriction

computation time, which also is stated in [Matthew and Raffaello, 2005]. The author of this thesis suggest to investigate the use of quadratic programming in this case, since it will more easy to apply when restrictions becomes circles.

In section 7.4 the effect the approximation of the speed and forces are discussed for a case where the force and speed limitations are active.

3.4 Implementation of Multiple Waypoints

It is assumed that it is desirable to have the possibility for the UAV to visit waypoints on its way from start to goal. In this section it is shown how this is included in the optimization problem. According to [Richards and How, 2002], this can be implemented as:

$$\forall i \in [1, \ldots, N] \, \forall c \in [1, \ldots, W]$$

$$\begin{aligned} x_i - x_{Wc} &\leq M_{big}(1 - b_{ic}) \\ x_i - x_{Wc} &\geq -M_{big}(1 - b_{ic}) \\ y_i - y_{Wc} &\leq M_{big}(1 - b_{ic}) \\ y_i - y_{Wc} &\geq -M_{big}(1 - b_{ic}) \end{aligned} \tag{3.56}$$

$$\forall c \in [1, \ldots, W] \sum_{k=1}^{T} b_{kc} = 1$$

where N is the number of time steps, W is the number of waypoints, x_i, y_i is the position of the UAV at time step i and x_{Wc}, y_{Wc} is the x- and y- position of waypoint number c. b_{ic} is an binary variable which indicates whether a waypoint c is visited at time step i or not. M_{big} is a large positive number which is much bigger than any difference between all waypoint positions and UAV positions such that (3.56) is feasible. In the simulations in Chapter 4 M_{big} is set to be, $M_{big} = 1000$. Later in this thesis, M_{big} will be changed due to a bigger operation area.

The last equation in (3.56) is an *hard constraint* which ensures that the UAV visit each waypoint once during the time horizon. In the equation it is not defined in which order the waypoint should be visited.

Remark: The restriction in (3.56) forces the time step on the generated path to be placed exactly at the coordinates of the waypoint. In some situations this restriction is probably too restrictive, and the constraint described in [Richards and How, 2002] is therefore, in this thesis, suggested to be modified to view a waypoint as visited, if the time step on the generated path is in a range Δ from the waypoint. Δ is an arbitrarily positive number which determines the minimum range from the waypoint, the time step has to be to be registered as visited. When the Δ value is implemented in the MILP optimization, this may give a smoother flight path.

$$\forall i \in [1, \ldots, N] \, \forall c \in [1, \ldots, W]$$

$$\begin{aligned} x_i - x_{Wc} - \Delta &\leq M_{big}(1 - b_{ic}) \\ x_i - x_{Wc} - \Delta &\geq -M_{big}(1 - b_{ic}) \\ y_i - y_{Wc} - \Delta &\leq M_{big}(1 - b_{ic}) \\ y_i - y_{Wc} - \Delta &\geq -M_{big}(1 - b_{ic}) \end{aligned} \tag{3.57}$$

$$\forall c \in [1, \ldots, W] \sum_{k=1}^{T} b_{kc} = 1 \qquad \Delta \geq 0$$

In the simulations in Chapter 4, the Δ value is set to be 5 meters. This means that a waypoint will be viewed as visited if the time sample generated in the optimization problem is 5 meters from the waypoint. However, the UAV might actually also get closer to the waypoint than 5 meters between two time-samples, but this will not be registered by the guidance system, since it only performs calculations at each time-step. If the waypoints are implemented with $\Delta = 0$, this can in many cases probably be too restrictive, since one often assume that a waypoint is visited if the visitor is in some region around the waypoint, due to for example the range of the camera, if this is applied on the mission. If there is no slack included in the restrictions the optimizer will force one of the samples to be placed exactly at the waypoint, which in many cases will be a unnecessary hard requirement. If the Δ variable is included with $\Delta \geq 0$ this will give the optimization problem more possible places to "insert" the time-sample inside the Δ radius around each waypoint. It it therefore reasonable to assume that this will result in a longer computation time. The path planner can however be able to a find a new path, with respect on the new restriction, which makes the cost function smaller. The restrictions in (3.57) only enforce each waypoint to be visited one time and no more than one time. The cost function will find the most effective order to do the visitation, with respect to the cost function.

We propose to select the variable Δ with care, such that it no becomes a restriction on the path. If Δ is to big compared to the velocity the UAV is travelling with it will not be possible for the path to cross the Δ circle since restriction (**??**) does not enable more than one sample to be inside the Δ square. We suggest therefore that Δ should be upper bounded by the following restriction:

$$\Delta \leq \frac{V \cdot T_d}{2} \tag{3.58}$$

where V is the speed of the UAV and T_d is the discretization time.

3.5 Implementation of obstacles

3.5.1 Stationary obstacles

When the UAV is in the air it has to consider no-fly areas, mountains and other obstacles. The obstacles can be non-convex, but can be implemented as a convex obstacle by putting a rectangular square around the obstacle. The square has to be big enough such that the actual obstacle is contained inside the square and this square will be viewed as the obstacle constraint in the MILP optimization problem. The position of the obstacle can be denoted by the coordinates of its lower left and upper right corner points: (x_{min}, y_{min}) and (x_{max}, y_{max}). At every time step i the position (x_i, y_i) of the vehicle must lie in the area outside of the obstacle.

This can be formulated as [Schouwenaars et al., 2001] :

$$\forall i \in [1, \ldots N,], \forall c \in [1, \ldots, O]$$

$$\begin{aligned} & x_i \leq x_{min} \\ or \quad & x_i \geq x_{max} \\ or \quad & y_i \leq y_{min} \\ or \quad & y_i \geq y_{max} \end{aligned} \tag{3.59}$$

(3.60)

The *or* constraints in (3.60) can be transformed to *and*-constraints when introducing binary slack variables [Schouwenaars et al., 2001]. Let the t_{ik} be a binary variable (0 or 1) and let M_{big} be a large arbitrary positive number. The constraints in (3.60) may then be replaced by the following mixed-integer linear constraints:

$$\forall i \in [1, \ldots, N], \forall c \in [1, \ldots, O]$$

$$\begin{aligned} & x_i \leq x_{min,c} + M_{big} t_{ic1} \\ and \quad & -x_i \leq -x_{max,c} + M_{big} t_{ic2} \\ and \quad & y_i \leq y_{min,c} + M_{big} t_{ic3} \\ and \quad & -y_i \leq -y_{max,c} + M_{big} t_{ic4} \end{aligned} \tag{3.61}$$

$$\sum_{k=1}^{4} t_{ik} \leq 3$$

where N is the number of time steps, O is the number of obstacles, and $t_{icu} \forall u \in [1, \ldots, 4]$ is a binary variable.

If the k^{th} original or-constraint is not satisfied on the i^{th} time step, the corresponding binary variable t_{ik} equals 1. The last *and*-constraint ensures that at least one of the original *or*-constraints is satisfied, which ensures that the path always is outside the defined obstacle area. The constraints (3.61) can be formulated for every obstacle in the manoeuvrable space. The constraints are not enforced at the initial state (i=0) because the position is at the initial state at this time, and this has to be assumed to be a feasible state. An arbitrarily shaped planar obstacle can be described in this way. Also non-convex constraint will be handled by the MILP optimizer. An example is shown in Chapter 4. It is straight forward to extend this technique to also include 3D-obstacles [Schouwenaars et al., 2001], but this is not considered in this thesis.

In Chapter 4 it is shown how the implemented MILP can be modified to also let this obstacle avoidance take into account obstacles which changes slowly in the area.

3.6 Techniques to avoid corner cutting in the MILP optimization

Obstacles which are in the operational area of the UAV are included in the MILP optimizer, but as has been discussed earlier the path between time samples can result in corner cutting. This means that the path between two feasible discrete time-steps intersects the obstacle, and this problem has to be taken into account such that the calculated path from the MILP never lead to a collision between the UAV and any obstacle. In the simulations in Chapter 4, the obstacles are implemented as a single point in the geographic frame, and are further enclosed by a square with arbitrarily sides which are considered as the obstacle-boundaries by the MILP optimizer. However, in a real environment, obstacles will not be single points in the operation area, but rather objects which covers a part of this area. The problem with corner cutting will increase with the speed of the UAV and the discretization time which has been applied in the discretization of the UAV dynamics. This coherence enables that two time samples are at feasible positions, and also that the path is allowed to pass the obstacle. In the following, two approaches to avoid corner cutting is discussed:

1. In [Reinl and Stryk, 2007] an approach to avoid corner cutting is described by inserting more constraint which introduces more restrictions in the feasible areas for the generated x- and y- time-samples. The method suggest to include the original obstacle in the MILP optimizer and to include more restrictions on x and y such that the straight line between (x_k, y_k) and (x_{k+1}, y_{k+1}) do not intersect the obstacle. To implement this approach, more binary variables must be added to determine which obstacle that is considered, and which of the four corners in the obstacle that is considered by the MILP. This approach to avoid corner cutting introduces additional binary variables for each time step and for each obstacle. As described in [Matthew and Raffaello, 2005] more binary variables will make the problem more complex, which will have a negative effect on the computation time in the MILP optimization. Since it is preferred to keep the optimization problem as simple as possible due to lack of processing power on-board the UAV, it is preferable to find other methods to avoid collision with obstacles caused by corner cutting.

2. In some of the literature that describes MILP optimization in areas where obstacles has to be avoided, suggests to "increase" the obstacle before it is implemented in the MILP optimizer. The increase in the size of the obstacle represents a "safety margin" compared to the real obstacle, and has to be big enough such that corner cutting in this increased obstacle don't result in a collision with the real obstacle in the environment. In [Matthew and Raffaello, 2005] a safety margin for obstacles which is implemented as circles in the environment was discussed. In the following a method for defining a safety margin for obstacles which are formed as a square or rectangle are discussed and developed.

Generally, the real obstacle with the safety margin around should be big enough such that it is not feasible for the path planner to make a path directly through the obstacle. This means that (3.62) and (3.63) should to be fulfilled.

$$x_{max} - x_{min} > V_{max} \cdot T_d \tag{3.62}$$

$$y_{max} - y_{min} > V_{max} \cdot T_d \tag{3.63}$$

Further the size of the safety margin must be determined. For simplicity the following derivation will assume that all obstacles in the environment are formed as squares with equal size. The distance from the real obstacle's boundaries and to the obstacles which will be implemented in the MILP optimizer, is defined to be the safety-margin, δ. Since the obstacle is a square this safety margin will be the same for all four restrictions which represents the boundaries around the obstacle. The length of the δ value is in the following calculated to ensure that the shortest path between two points that will be in a feasible area, do not intersects the obstacle. This is illustrated in Figure 3.4.

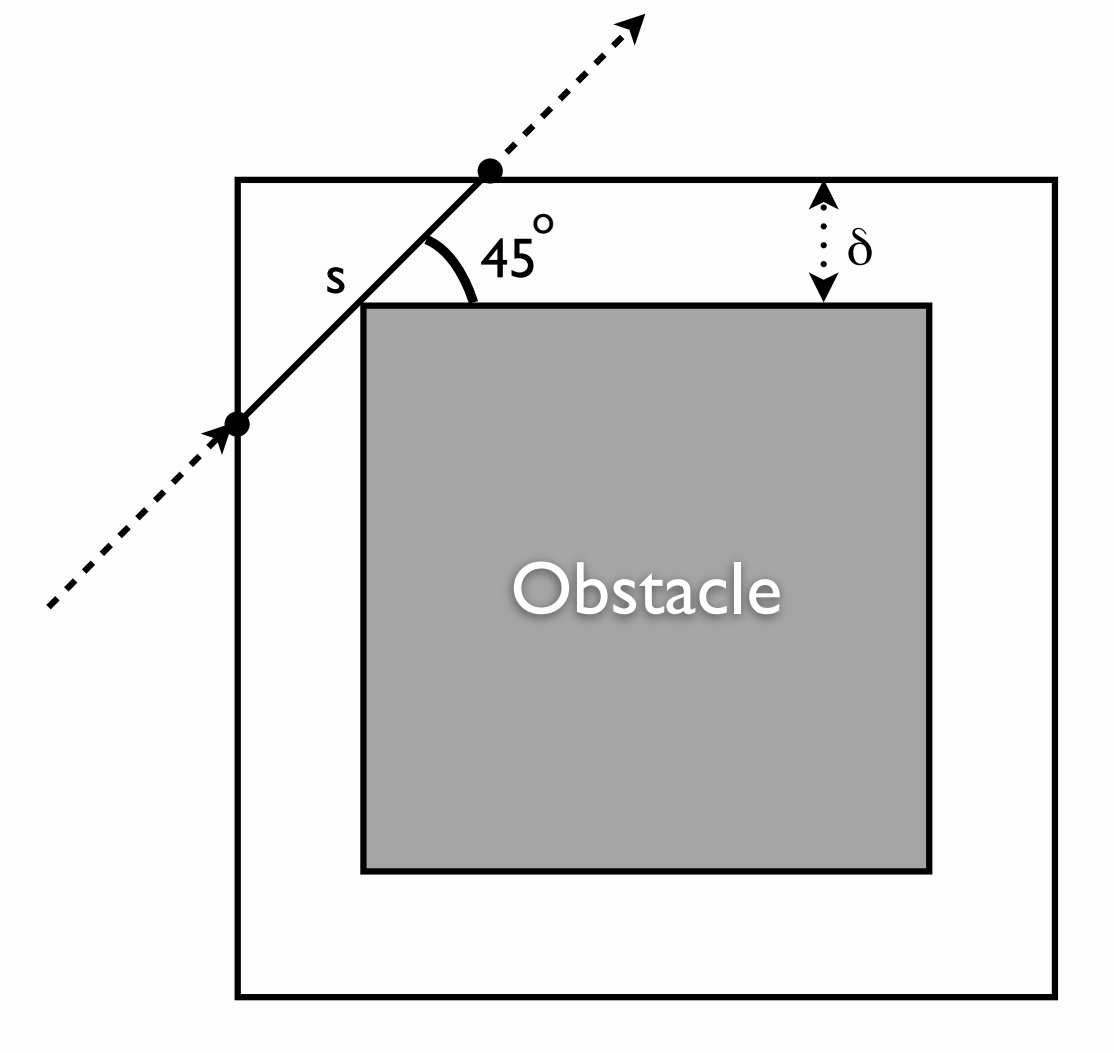

Figure 3.4: The figure shows how one can define the safety margin, δ such that the path between two waypoints will not intersect the obstacle. The outermost boundaries are the "obstacle" which is implemented in the MILP

The distance the UAV can travel between each time-step can be described by the following equation

$$s > V_{max} \cdot T_d \tag{3.64}$$

where s is the travelled distance, V_{max} is the maximum speed of the UAV, and T_d is the discretization time.

As can be viewed from Figure 3.4 the distance s, can be described as a function of the safety distance, and by applying calculus, this this relation can be found by the following equation

$$s = \frac{2\delta}{\sin(\Upsilon)} \tag{3.65}$$

where δ is the safety margin and Υ is the angle between the boundary and the path between the waypoints. In Figure 3.4 the angle, Υ, is set to be 45°, since this gives the shortest path between the two waypoints and therefore is the most extreme situation which the safety margin has to be designed for.

Inserting (3.65) in (3.64) and solving for δ gives

$$\delta > \frac{V_{max} \cdot T_d}{2} \cdot \sin(45°) \tag{3.66}$$

where the value for Υ from Figure 3.4 has been used. Equation (3.66) is the lower bound on the safety-margin for a given value of V_{max} and T_d.

3.7 Restrictions on the positions state variables

When the optimization problem is solved on a computer, all parameters which describes the dynamics of the UAV are included as restrictions in the optimization problem. Also, waypoints and obstacles are included as restrictions to ensure that these areas are visited and not visited, respectively. Depending on the area which are going to be searched, a restriction which defines upper bounds on the x- and y- decision variables should be included to narrow the solution area. A realization of this restriction was shown as an example in Chapter 2.2.

3.8 Fixed Horizon Cost Functions

A fixed horizon controller can be implemented where the objective is to minimize the arrival time or minimize energy consumption or a combination of both energy and time consumption. For the case of the UAV, the situation of minimizing time or energy consumption will depend on the situation. In some cases a combination can be preferred, because a combination of both minimizing time and minimizing fuel can be the most reasonable. If the cost function in the optimization problem only considers fuel consumption, the path planner will use as small amount of energy as possible, and this will result in a situation where the whole horizon will be used to reach the goal if the initial velocity is zero. Because the input to the system will be calculated to be as small as possible the goal is reached at the last time step. If the initial velocity is different from zero and the velocity vector is pointing towards the goal, the goal can be reached before the last time step if the drag force is not included in the optimization, and if there are no obstacles in the way.

If only minimum arrival time is considered in the cost function, the fuel consumption will not be considered at all, and the use of forces will be as large as possible to reach the goal in the shortest time. This is in many cases not preferred, and it turns out that a combination of both minimum time and minimum energy consumption can be the best way to design the cost function. Penalty can be added to the part of the cost function which is assumed to be most important.

In the following, the objective of the cost function is designed to minimize energy consumption and to arrive at the destination as fast as possible. In chapter 4 the objective of the cost function is to minimize the energy consumption and in the rescue scenario in Chapter 6 the objective is to minimize the arrival time at the destination.

3.8.1 Minimum Time Controller

According to [Grøtli and Johansen, 2011a] a cost function to minimize the time it takes to arrive the base station is given by

$$\begin{aligned} &\forall i \in \{1, ..., N\} \\ \theta^{finish} &\leq M^{finish}\left(1 - b_{iw}^{wp}\right) + i b_{iw}^{wp} \\ \theta_p^{finish} &\geq i\left(1 - b_{iw}^{wp}\right) \end{aligned} \tag{3.67}$$

Where M^{finish} is constant which is sufficiently large, for instance as $M^{finish} := N$ as described in [Grøtli and Johansen, 2011a]

The objective is to minimize

$$J^{finish} = \theta^{finish} \tag{3.68}$$

such that the aircraft arrives at the destination as fast as possible.

This cost function will be further discussed and implemented in Chapter 6.

3.8.2 Minimum energy consumption

The cost function can also be designed to minimize energy consumption. The simulations in this thesis is done with respect to energy use.

The cost function can be written as:

$$J = \sum_{i=1}^{N} |f_{i,x}| + |f_{i,y}| \tag{3.69}$$

where $f_{i,x}$ and $f_{i,y}$ are the forces in x- and y direction applied at time step i.

It is not preferable to have an absolute value in the cost function, since this introduces non-linearities in the cost function, and the MILP method is only able to solve linear programs. Equation (3.69) is therefore transformed to linear form by using slack variables and additional constraints [Bertsimas and Tsitsiklis, 1997].

$$\begin{aligned} J = \sum_{i=1}^{N} w_{i,1} + w_{i,2} & \\ s.t & \\ f_{i,x} \leq w_1 & \\ -f_{i,x} \leq w_1 & \\ f_{i,y} \leq w_2 & \\ -f_{i,y} \leq w_2 & \end{aligned} \tag{3.70}$$

Where w_1, w_2 are slack variables.

3.9 Discussion

The UAV will fly to reach the desired waypoints and has to avoid obstacles and restricted regions. Obstacles the UAV has to avoid can be buildings, mountains and other air planes. Buildings and mountains will be stationary obstacles, but other aerial vehicles has to be modelled as dynamic obstacles, since they will change position every time step. If the other vehicles are following a predefined path which is known, this can be included in the MILP optimization problem. Otherwise, if the path of the other aerial vehicles is not known they can be assumed to be stationary at every time step if the obstacle position is updated in the optimization at every time step. This method will probably work bad if the obstacles are moving fast and the computer which solves the path planning is slow.

In this report, dynamic obstacles will not be considered and only stationary obstacles will be implemented in the test cases. Every obstacle is represented as a point with a safety margin to each side. The safety margin is implemented as a square, where each side is 100 meters, and the center is located at the obstacle (x,y) point. The safety margin should also take into account that the unmanned aerial vehicle is not a point mass, but has a wing span which need to be considered.

In the first test case in this report, all the obstacles are known *a priori*, and the optimal trajectory is based on this information. In the second case, the implementation of the problem is modified, such that only the waypoints is known *a priori*, and the optimal trajectory is based on this. Obstacles are in this case detected when the UAV is a certain distance from the obstacle, to make the test case more realistic because of the range of the radar, (R_{range}). When an obstacle is detected

it will be included as a constraint in the optimization problem to ensure collision avoidance. This is implemented in a similar way as a *receding horizon* approach. The difference is that the horizon in this case is fixed, but the optimization problem is re-optimized the first time a new obstacle is detected by the radar. This is done because it is interesting to see the response of the optimization algorithm to find out which range the radar should have, and how the Gurobi optimizer handles new obstacles. The range of the radar has to be adapted to be long enough in relation to the aircraft dynamics to avoid a situation where the optimization problem becomes infeasible. All detected obstacles is saved in an database such that the guidance system can take the obstacles into account, if they have been detected earlier, even if they are outside the range of the radar. This database should then only consider stationary obstacles, since obstacles as other aircraft will change as a function of time and this will give problems with the registration. If it is preferable to detect moving obstacles as well, the implementation of the obstacle avoidance system will handle the moving obstacle as a stationary obstacle at every time step. In this case the guidance system should "forget" the position of obstacles which has been detected earlier. This implementation can be improved if it is assumed that the moving obstacle is following a predefined path which can be modelled and implemented in the optimization problem. This will make the optimization more effective, since the generated optimal trajectory will be depending on the direction of the moving obstacle. This is not considered in this thesis.

The radar will make a picture of the environment in some limited region around the UAV. The specification of the radar used in this case is not known. The range of the radar will be an important specification for the anti-collision system. A greater travelling speed will require a radar with a longer range. Also the manoeuvrability of the UAV has to be considered when defining the requirements for the radar range, to make sure that the guidance system has enough time to compute a new trajectory, when the inertia of the UAV is taken into account.

In this thesis simulation with radar obstacle detection is simulated with a radar range, $R_{range} = 130$m. The simulation was also tested with a radar range of $R_{ange} = 80m$, but this gave infeasible solutions with the implemented equations of motion, and restrictions.

Chapter 4

Simulation and Results

This chapter shows the optimal path, calculated using the MILP from Chapter 3, for the UAV when linear vehicle dynamics and physical restrictions are applied as approximations to the aircraft dynamics. Results from simulations of path planning with obstacle avoidance will be presented and discussed. A general result from the simulations is that the trajectory cut's the corners of the obstacles if the operating area gets large and the relationship between time horizon and discretization time is small. This is because collision avoidance is only enforced at each discrete time step as was discussed in Chapter 3. Therefore, the obstacle regions in the optimization must be larger than the real obstacles to make a safety margin. A suggestion about how this safety margin can be implemented was given in section 3.5.

In the simulations in this chapter, the path was generated with a horizon of 150 seconds, and the the discretization time was 3 seconds. This results in 50 time samples.

The simulations was done on a computer with the following specifications:

- Operating System: Windows 7 Enterprise 64-bit
- Processor: Intel(R) Core(TM)i5-2500 CPU @ 3.30GHz (2 CPUs),
- Memory : 8,00 GB RAM

The Optimization has been done with the Gurobi solver specified in A.1.

4.1 Test cases where obstacles are known *a priori*

In this case the position of the obstacles are known before the vehicle is flying from the initial position to the final position, and the optimization is done only once to find the optimal path with respect to minimum energy consumption. In the simulation shown in Figure 4.1 one can see that the path cuts the corners of the obstacles. The path generated in Figure 4.1 took 110 seconds to be generated. In this simulation the initial position vector was $s_0 = [x_0, y_0, v_{x,0}, v_{y,0}]^T = [0, 0, 4, 4]^T$, the minimum speed restriction was 5 $\left[\frac{m}{s}\right]$ and the restriction on the absolute value of the forces was 1.5N. The specification for this path-planning was:

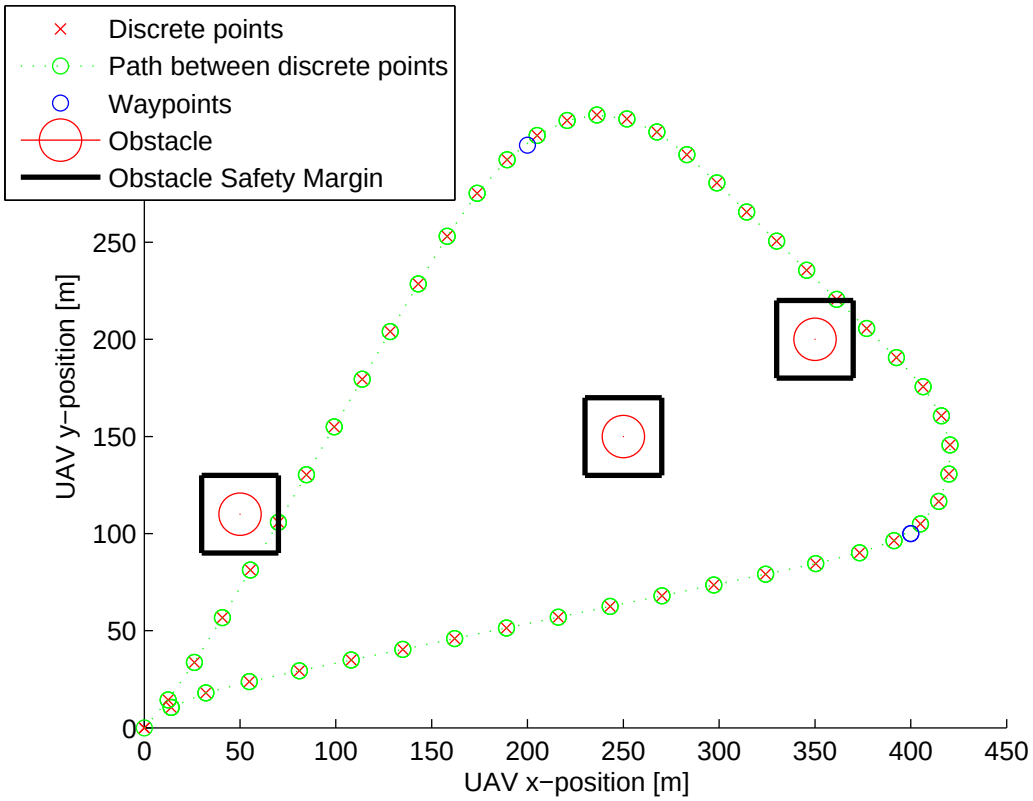

Figure 4.1: The figure shows generated path with horizon 150 second and Discretization time 3 seconds, resulting in 50 time steps. The simulation took 110 seconds.

Visit two way points at the locations (400,100) and (200,300) respectively, and finish at time step N, with the following specification: $s_N = [x_N, y_N, v_{x,N}, v_{y,N}]^T = [0, 0, -4, -4]$. The obstacles are placed at the following coordinates: $(50, 110), (250, 150)$ and $(350, 200)$.

Figure 4.2 shows the absolute value of the desired forces applied on the UAV, and the desired UAV speed. The figure shows that both the force and velocity constraints are held, except for certain cases where the desired limit is exceeded. This is due to the linear approximation of the non-linear constraints which was done in (3.54) and (3.55), where the value M describes how well the approximation is. In this thesis, $M = 10$. If M increases, the linear constraints will be a better approximation to the non-linear constraints, and the problem with values that exceeds the desired constraints will be smaller.

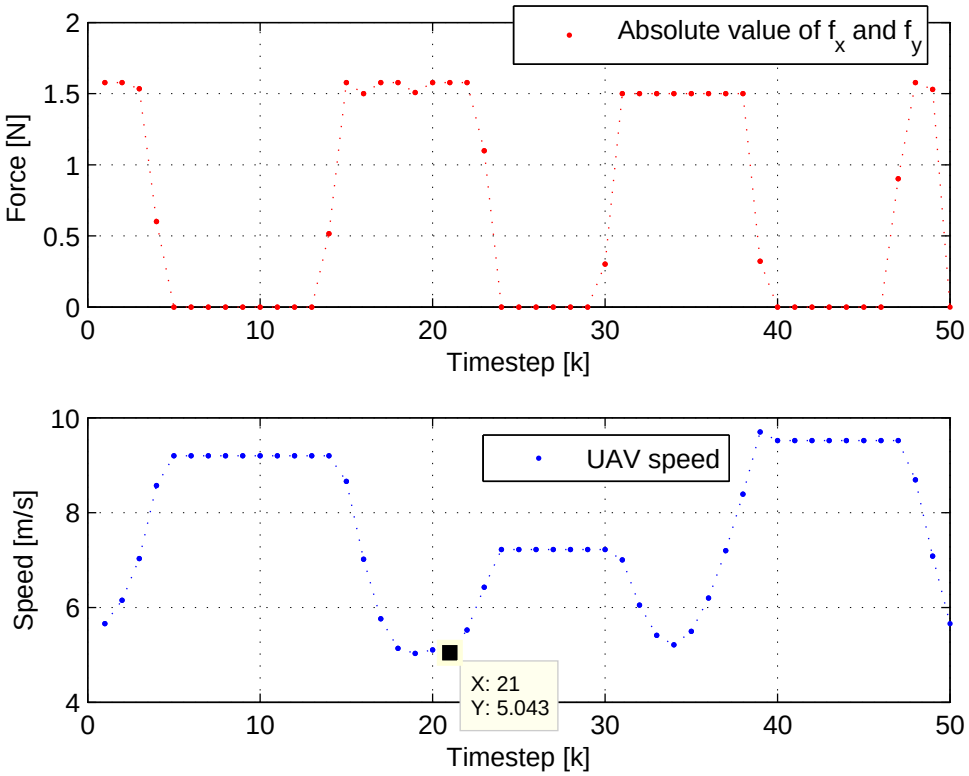

Figure 4.2: The figure shows applied forces and the speed of the UAV. As one can see the force exceeds the force constraint and this is due to the linear approximation of the non-linear constraint.

Figure 4.3 shows almost the same scenario as above. The only difference is that one waypoint and one obstacle are moved to $(425, 175)$ and $(425, 140)$ respectively. When the obstacle and the waypoint are moved closer to each other, the optimization took more time. In this case, the computation took 830 seconds. Figure 4.4 shows the applied force and the speed in this case.

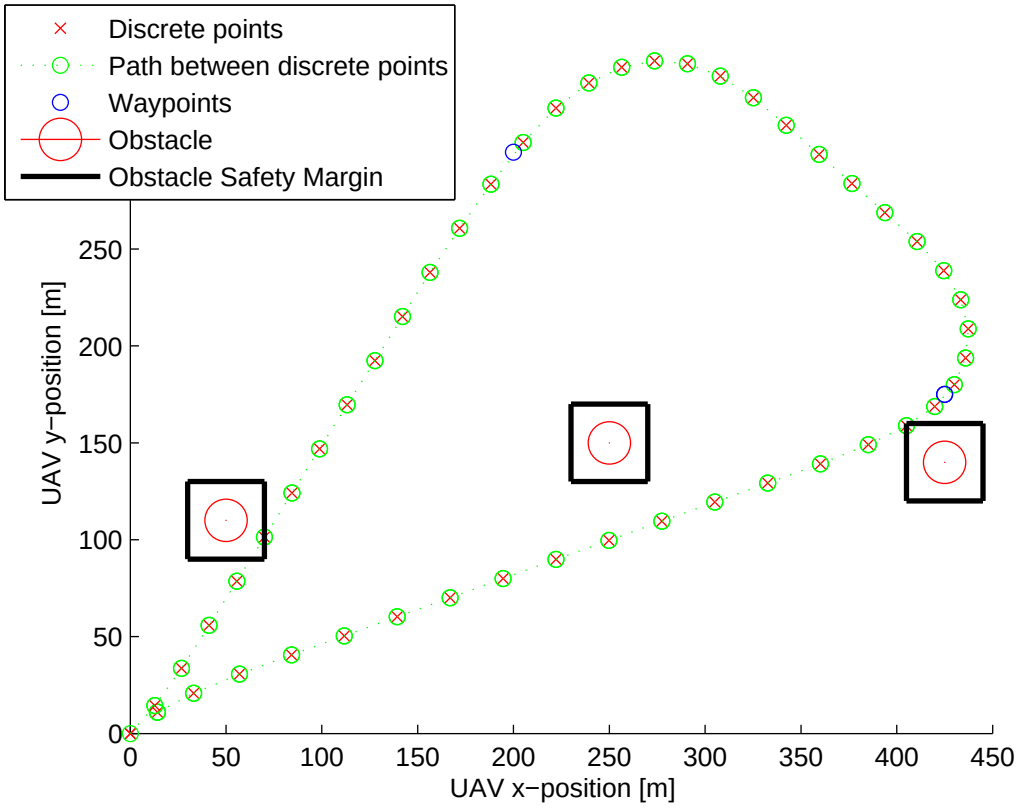

Figure 4.3: The figure shows generated path with horizon 150 second and discretization time 3 seconds, resulting in 50 time steps. The simulation took 830 seconds.

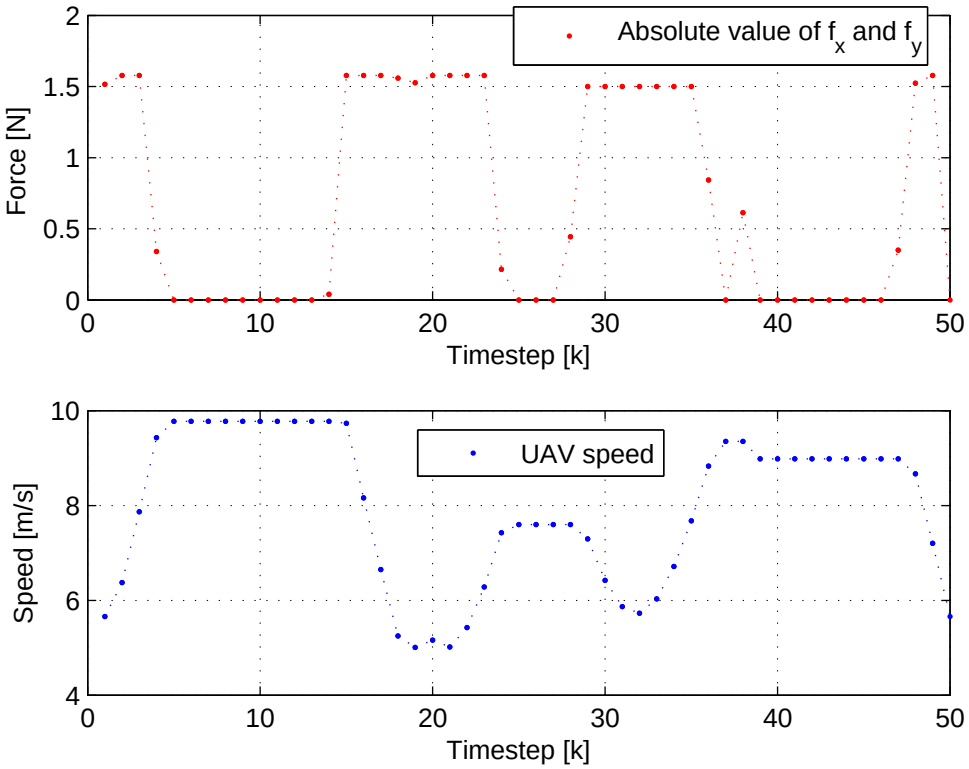

Figure 4.4: The figure shows applied forces and the speed of the UAV. As one can see the force never exceeds the force constraint, and the speed never exceeds the upper an lower speed constraint.

Figure 4.5 shows another scenario. In this case the UAV is supposed to go from $s_0 = [0, 0, 0, 0]$ visit two way points located at $(200, 300)$ and $(400, 100)$, and end at the destination $(x_N, y_N) = (425, 300)$.

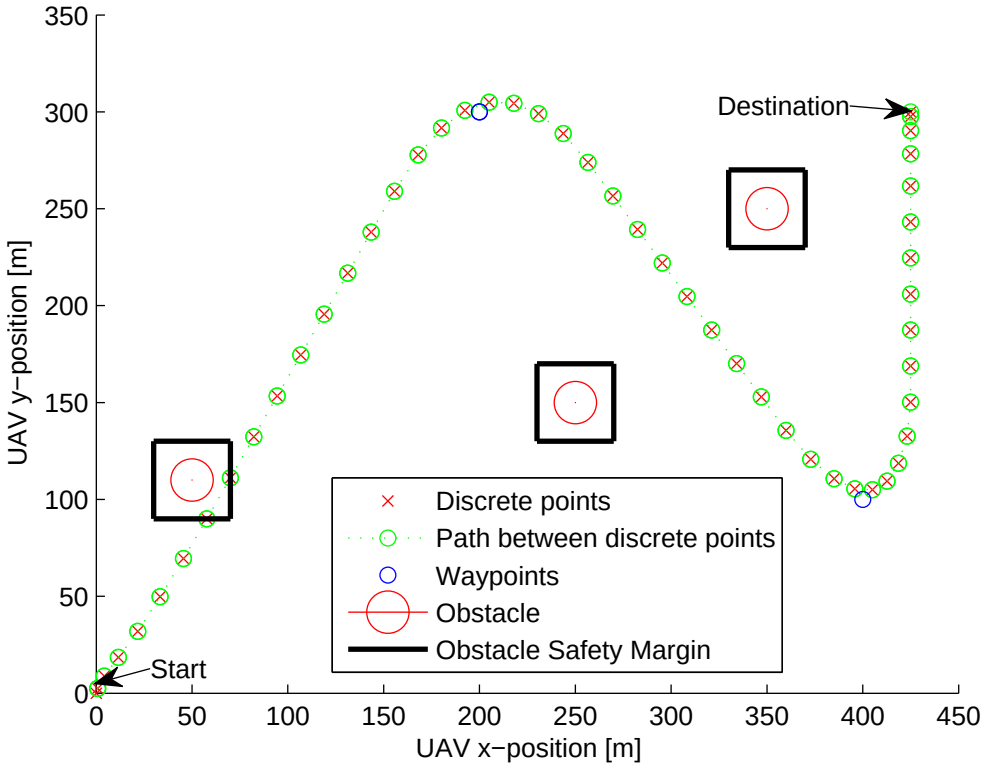

Figure 4.5: Desired optimal fuel trajectory. The horizon is 150 seconds, discretization time is 3 seconds. Simulation took 72 seconds to reach the goal (425,300)

The obstacles has been placed at $(50, 110)$,$(250, 150)$ and $(350, 250)$. Note that in this case the UAV has no initial velocity, and there are no specified values for the velocity at the goal. The minimum value for the speed (V_{min}) is in this case set to zero. This can not be justified physically, because the UAV path needs a lower restriction on the speed such that the UAV has enough lift along the path. However, in the case of testing responses, simulation time and debugging the design, it is sensible to do tests for different cases. Figure 4.6 shows the applied forces and the speed along the path for this case.

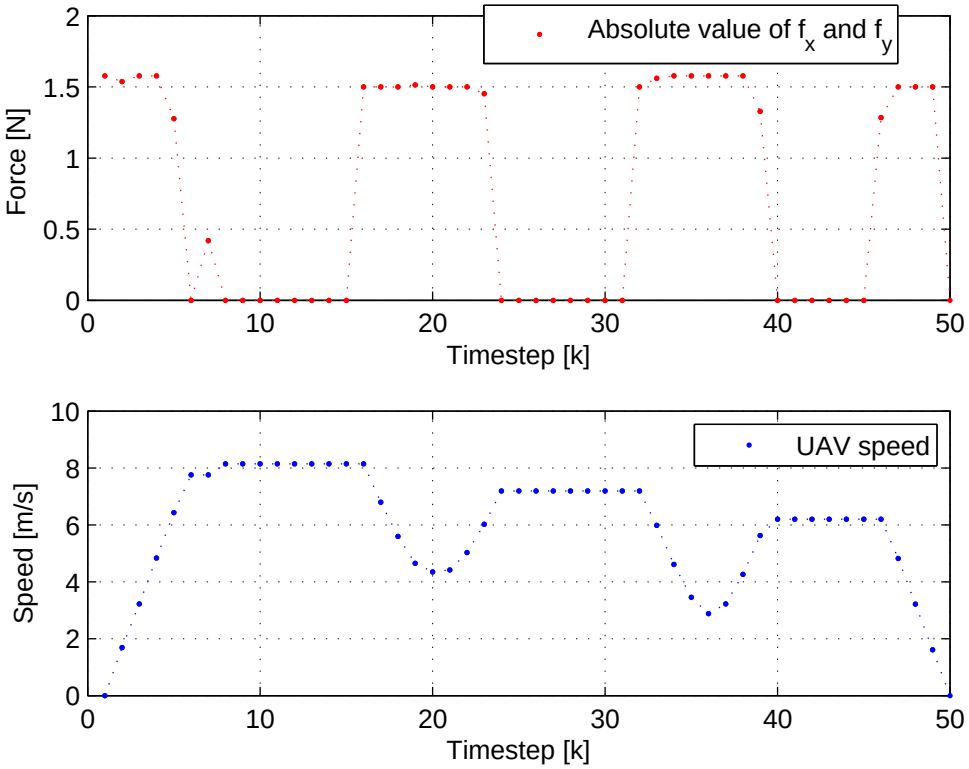

Figure 4.6: Desired optimal speed and trajectory. The horizon is 150 seconds, discretization time is 3 seconds. Simulation took 630 seconds to reach goal (425,300)

Figure 4.7 shows a complex path-planning where the UAV is trapped inside two obstacles and a no-fly area. This illustrates that the MILP technique is a good tool to solve complex guidance problems.

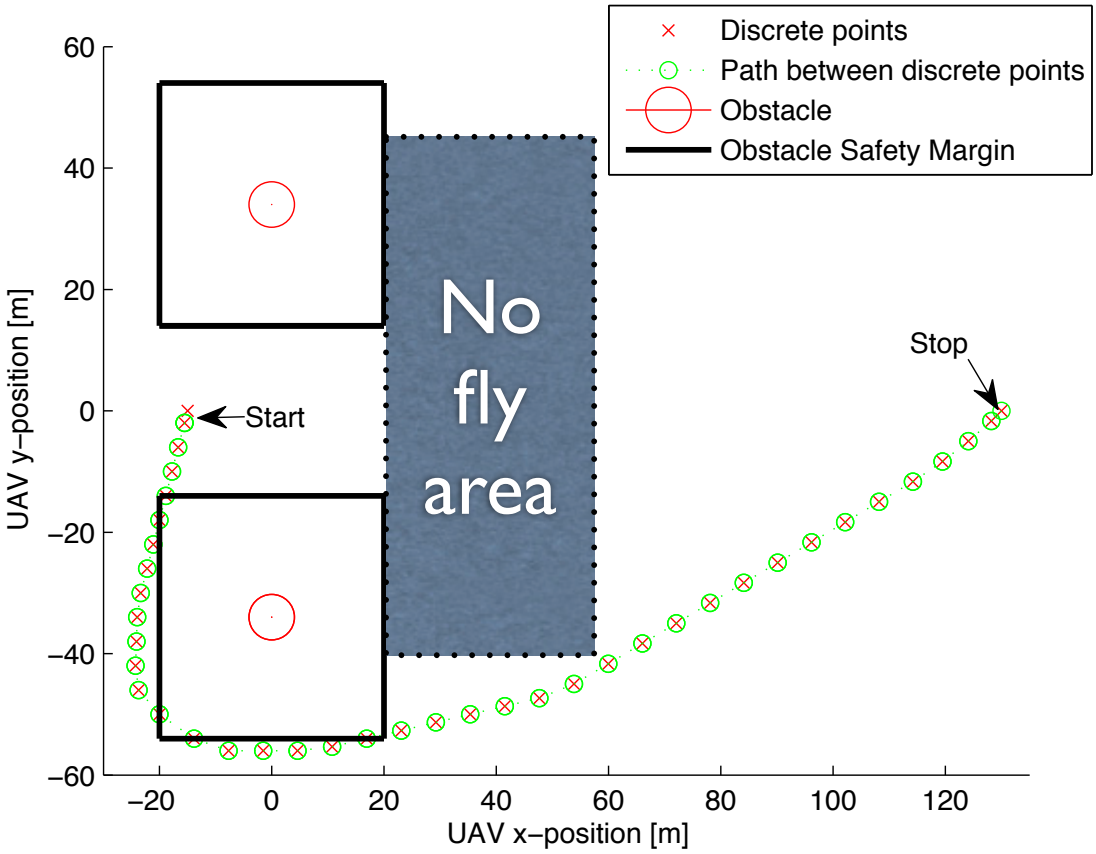

Figure 4.7: The optimal path when the UAV is surrounded by obstacles and no-fly areas. The computation time took 885 seconds

4.2 Path-planner with radar detection for obstacle avoidance

In this situation, only the waypoints the UAV is supposed to visit is assumed to be known before the optimization starts. This implementation is done such that the optimization problem is re-optimized when the the position of the UAV is in a specific distance from an obstacle R_{range}, and the obstacle is then included in the optimization problem. The motivation for doing this, is to make the optimization more realistic with respect to a real flight, and how the optimization problem will deal with obstacles that are added to the problem during the flight.

When system identification has been applied to the Odin delta-wing aircraft, the path planning model derived in this thesis could be further improved. The path planning with radar detection can be applied to simulate the system, and to detect responses with use of radar with different ranges. If the range is too short the optimization problem will become infeasible, because there is not enough time to avoid obstacles. The minimum radar range can be found for which the UAV can detect and avoid obstacles. In the test simulation, obstacles is added to the optimization if they are detected by the radar, and the optimization will be done again with the obstacles included. The implementation can easily be done in two different ways for different purposes as described under. In the implementation the design is done for the first scenario, and a simplified flow chart for this developed design is illustrated in Figure 4.8.

In the simulation the range of the radar is set to be, $R_{range} = 130m$. Two different approaches to implementation of the radar detection is presented below:

1. If an obstacle is detected by the radar, it will be added to the *ObstacleDetectedArray* and it will be there for all time, as long as the program is running. This will be a sensible way to do the implementation if only stationary obstacles are added, since the obstacles that are detected already have been taken care of by the optimization problem and the optimization does not need to be alerted again before a new obstacle is detected. However, this implementation will give problems if dynamic obstacles are considered. Each time step when the obstacle moves, a new obstacle will be added to the optimization problem, but previous positions of the obstacle will not be deleted and will therefore still be viewed as obstacles by the guidance system, even though the dynamic obstacle has moved away from this position.

2. The design as previously described, can easily be changed to view every detected obstacle as a "new" obstacle for each time step. This can be done by defining the array *ObstacleDetectedArray* in the Matlab code inside the *while()*- loop. The motivation for doing this is due to the reason that obstacles are allowed to change position. If an obstacle is detected along the optimal path, it will be included in the optimization problem and the optimization will be re-optimized. At the same time, previously discovered obstacles will be forgotten. The UAV will move to the next state generated by the optimization, and if an obstacle is still detected by the radar, it will do the optimization again(even if a previously discovered obstacle is still in the same position). This implementation will take care of slowly varying obstacles in the environment. However, the time horizon should be short if this method should be applied to ensure a reasonable computation time. This

method will possibly be efficient if a receding horizon strategy is applied in the implementation.

4.3 Test case with the obstacle radar detection against design with *a priori* obstacle information

In this section the designed optimal path planner with radar obstacle avoidance will be tested against the optimal path planner with a priori knowledge about all the obstacles in the area. The same test scenario is applied on both designs. The initial state vector is given as:

$$s_0 = [x_0, y_0, v_{x,0}, v_{y,0}]^T = [0, 0, 0, 0]^T \tag{4.1}$$

The issue is to move from s_0 to the waypoint at position $(x_w, y_w) = (0, 300)$. In the area there are obstacles at the following positions: $(0, 200)$, $(150, 150)$ and $(-30, 243)$. In the test case with *a priori* obstacle information the path planner will identify all obstacles when designing the optimal path. The test case with obstacle radar detection will recognize the obstacles as they are detected by the radar. In the simulations obstacles that are added to the optimization problem is shown with a safety margin around the obstacle. Obstacles that are not added to the optimization problem will only show the position of the obstacle with a red circle to illustrate that they are not a part of the optimization problem.

Figure 4.9 shows the optimal path when all obstacles are known a priori. As one can see, all the obstacles are included with a safety margin around the actual obstacle. Figure 4.10 shows the optimal path when obstacles are added to the path planner as they are detected by the radar.

When radar with a limited range is applied to sense the obstacles, it can be observed that only two obstacles are detected. The obstacle at position $(150, 150)$ is not detected by the radar, because it is too far away from the UAV. The path becomes different in the case of *obstacle avoidance with radar detection* compared to the path when obstacles where known *a priori*. The reason for this is that the obstacle at the position $(0, 200)$, is detected when the UAV is on its way to the goal, and the path planner did not have previous knowledge about this obstacle when the first path optimization was done. When the obstacle is detected the MILP re-optimizes with the detected obstacle included in the problem. As one can see in Figure (4.10) the new path turns to the left. When the UAV detects yet another obstacle in $(-30, 243)$, the optimizer has to re-optimize again with the new obstacle included in the MILP. The path now turns to the right around the obstacle and to the goal.

This can be better viewed in Figure 4.11 where the where the positions where the UAV detects the two obstacles are included in the plot.

At this position the radar will detect the obstacle. As one can see, the path direction is not changed immediately. This is for the reason that the modeled UAV dynamics can't change immediately, and need some time change direction due to conservation of inertia of the modeled UAV.

4.3.1 Discussion

In the test case both time horizons were 150 seconds and the discretization time was 3 second. The computation time of the test case where all obstacles were known *a priori*, took 680 seconds. The test case were obstacles where detected by radar took only 3 seconds! The design which considers obstacle avoidance with radar detection is shown to be significantly better approach to solve the problem, compared to the case where all obstacles are known *a priori*. The approach with radar detection makes the optimization problem smaller, since only obstacles in a certain area from the UAV are considered, which makes the MILP program easier to solve. The obstacle avoidance with radar detection is not is not an optimal solution with respect to minimum energy consumption due to the lack of a priori information, but it can be said to be sub-optimal.

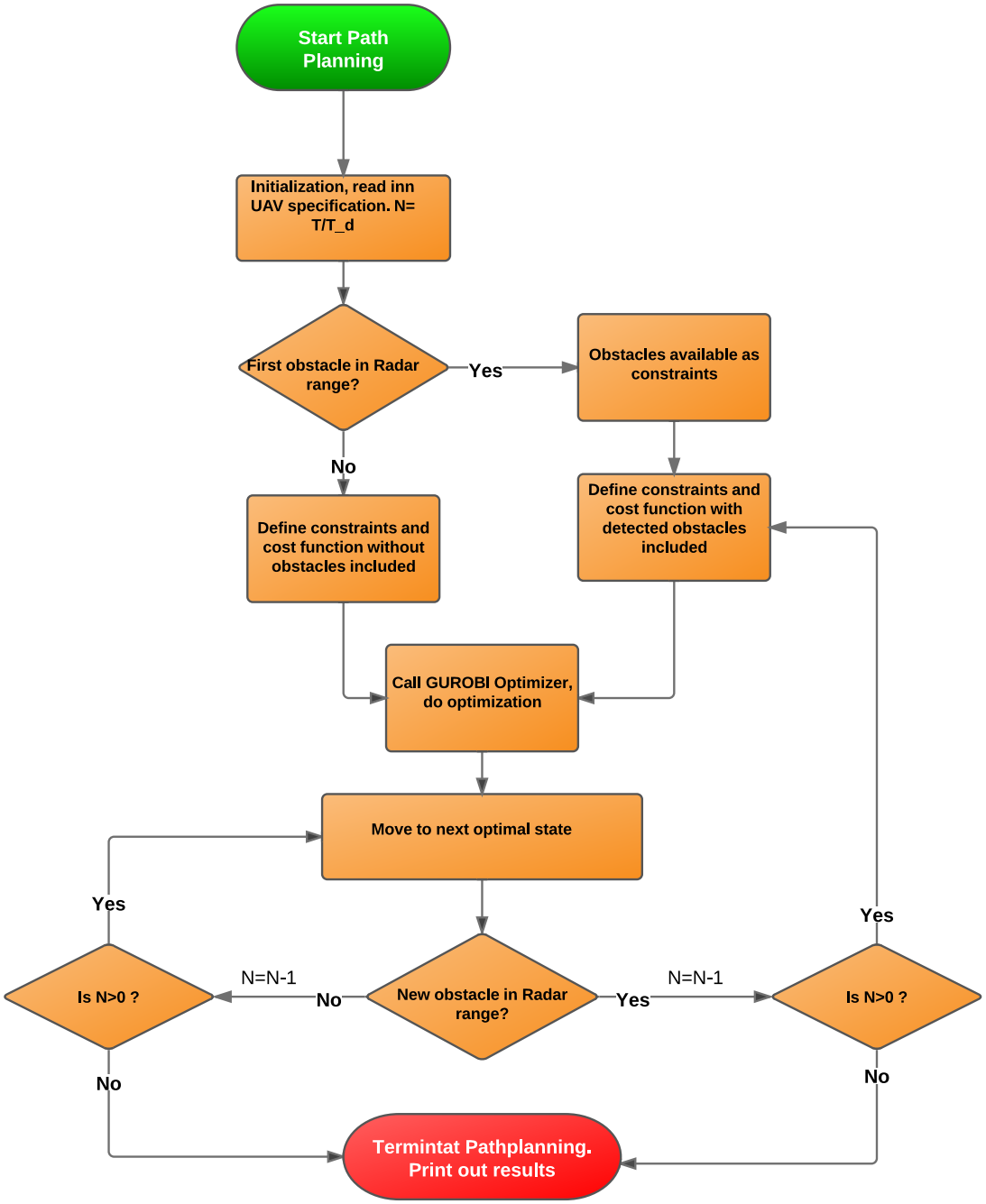

Figure 4.8: Flow Chart of the designed path planner which includes obstacle avoidance with radar detection.

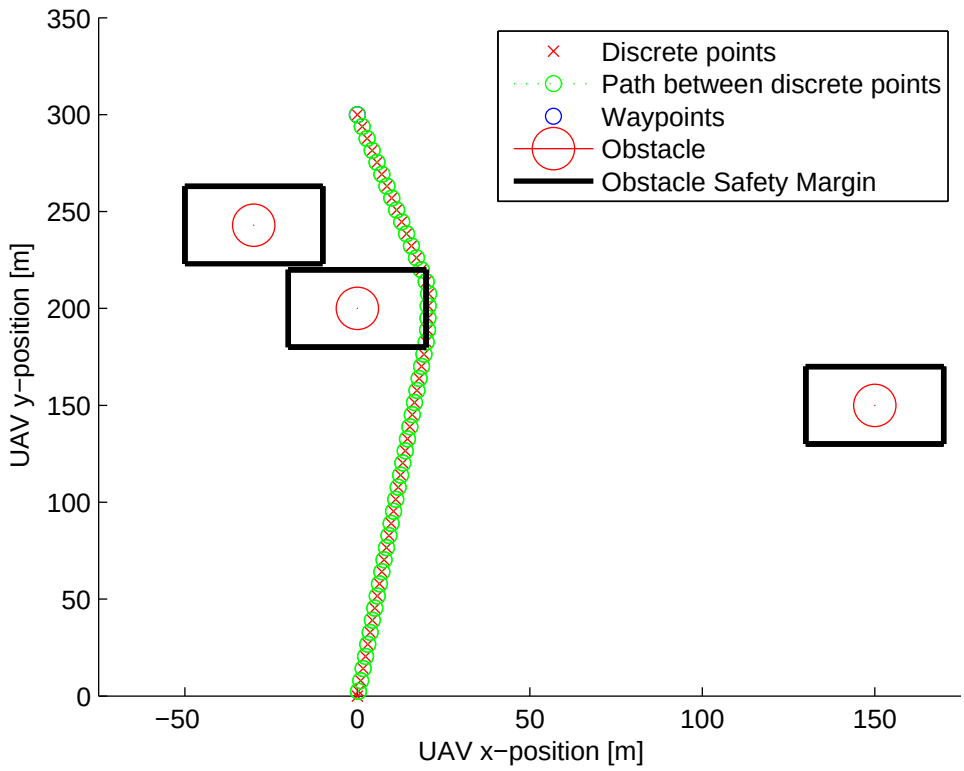

Figure 4.9: Path planning, obstacle avoidance *a priori* obstacle information. Computation time: 680 seconds.

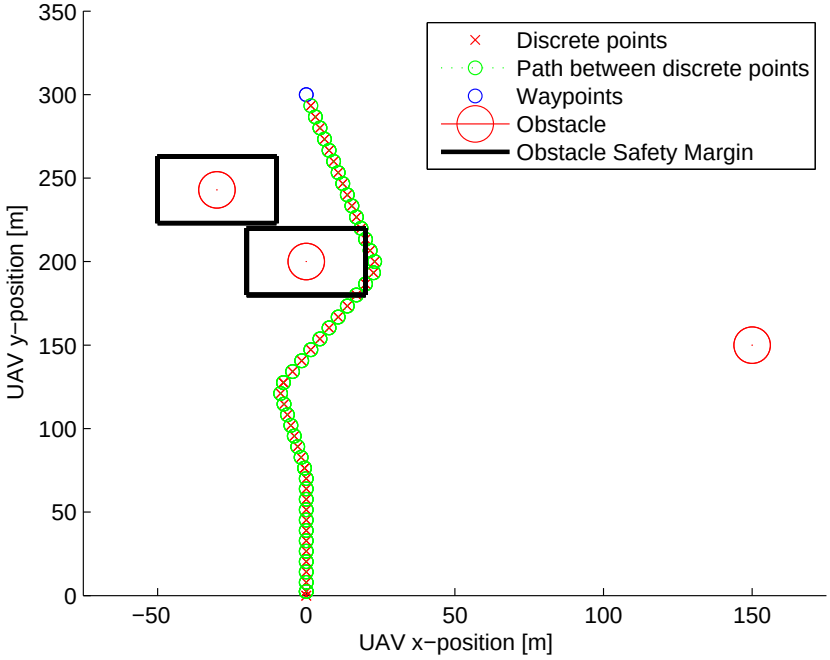

Figure 4.10: Path planning, obstacle avoidance with radar detection. The figure shows that two obstacles are considered by the MILP during the path. The path turns because obstacles are detected during the path, and the MILP is than re-optimized. As one can see the obstacle at (150,150) is not included in the path planner during the mission since it is outside the radar range. Three computations was done and the total computation time was 3 seconds.

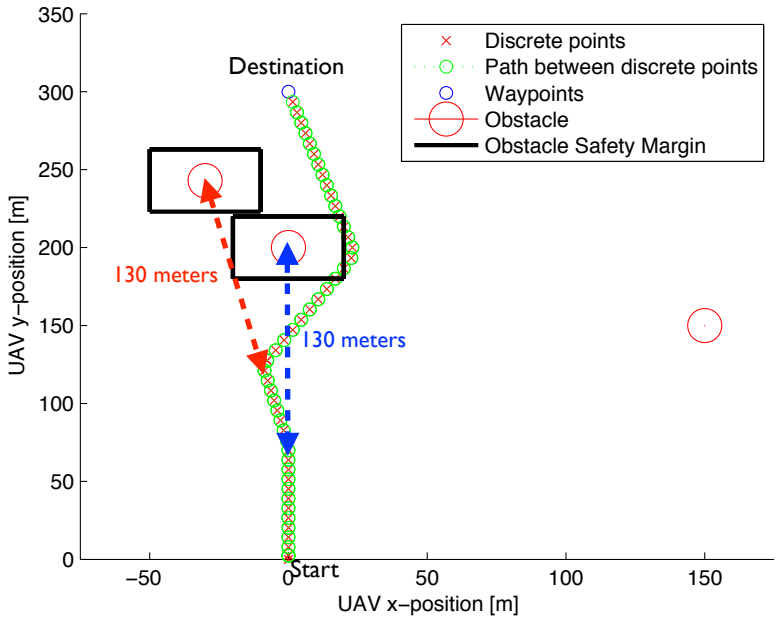

Figure 4.11: Path planning, obstacle avoidance with radar detection. The places where the respective obstacles are detected by the radar is given in the figure. The blue line illustrates when the obstacle at (0,200) is detected, and the red line illustrated when the obstacle at (-30,243) is detected by the radar.

Part II

Search Missions

Chapter 5

Introduction

Its evident that the use of UAVs can be of relevance in a wide range of domains, varying from general search missions to specific military operations. The UAV technology can thus be applied in investigating the location of the enemy or perform other surveillance tasks of strategic importance to the army in general. Another specifically important area for Norway, is of course the monitoring of oil installations. Surveillance of the Northern Regions will obviously be a challenge for the country in the years to come. In many areas that today are closed for the general public, and have a guard force to protect the area, could also need UAVs to search the area regularly in order to recognize unauthorized persons, and inform the guard force.

There are furthermore civilian search missions where the use of UAVs can be more efficient, less expensive and it could also reduce the risk for the human operator. Today many search missions take place in complicated areas such as sea- and mountain regions where search crews frequently are facing challenging and dangerous situation. Therefore, to reduce the risk and the cost of a rescue mission, it could be beneficial to apply an UAV for some applications as described in Chapter 1.

In this chapter, the author propose methods which can be applied for efficient search missions, when considering one specific search and rescue mission. The overall goal is to search a specific area as fast as possible. The UAV is assumed to be equipped with a suitable camera system which can detect and analyze the area under and around the UAV in a given sector.

Below is two suggested methods to be applied for the search and rescue mission:

1. One search method that will be used, is to apply MILP optimization to design the search path for the whole search mission. This method will be designed such that an arbitrary number of UAVs can take part in the mission. When more than one UAV is considered in the search mission, mainly two different strategies for coordinating the search mission can be used. They can briefly be described as [Inalhan et al., 2002]:

    - *Centralized Optimization Problem:* One computer which has access to all system information. Briefly this means that one cost function is applied, to coordinate the path for the overall system by collecting optimization variables from each vehicle.
    - *Decentralized Optimization Problem:* One optimization problem in each vehicle (subsystem) are calculated, and there are restrictions and communication to the neighbour subsystems.

When using MILP optimization for more than one UAV, the *centralized Optimization Problem* will be applied in this thesis.

The author purpose an algorithm to automatic generate waypoints which ensures that a defined area is searched by the UAV.

2. Another method to be considered for search and rescue mission is to apply an Archimedes spiral to cover the defined area. The use of a spiral for search and rescue purposes is described briefly in [Al-Helal and Sprinkle, 2010]. The author purpose methods for automatic design of an Archimedes spiral for different search missions.

In this part of the thesis the following scenario will be considered:

Case: The main rescue center in Bodø (Northern Norway) receives a mayday signal from a fisherman who has fallen overboard from his boat somewhere in the ocean outside Bodø. The conversation between the fisherman and the rescue center is very short, and the operator at the rescue center does not receive any information of the fisherman's position before the connection to the fisherman is broken. The weather in the area is calm, but the ocean is very cold at this time of the year. It is therefore urgent to find the fisherman before he dies as a result of the cold water. The operator fortunately knows that the rescue center has UAVs available through the company "*Rescue Vehicles AS*" which also has made software for efficient search of a defined area. The UAVs are sent on the search-mission in a limited area in the coast outside Bodø with a suitable IR-camera for locating the fisherman. When the position is located a rescue boat will go directly to this position and save the frozen fisherman. It is assumed that the fisherman is located inside the defined search area, and that there is no places inside the search area where it is more likely to find the fisherman than in other sections.

Chapter 6

Path-planning for search and rescue mission using MILP

This Chapter will describe a method to do the search in the defined area with the use of MILP optimization. The design and implementation of this search algorithm has been done for a general case, such that an unlimited number of UAVs with arbitrary base stations can take part in the same search mission and allocate different parts of the search grid. However, in the simulations discussed in this thesis a maximum number of two UAVs will be considered for simplicity. In the implementation, the cost function will be defined to minimize the time it takes to search the defined area, and return back to the base station(s).

6.1 Implementation of multiple vehicles

The search of the area where the fisherman is expected to be found must be done in a systematic way, such that the whole area will be covered by the camera on the UAV. If there are more than one UAV available, it will be natural to expect that the time spent for searching the area can be done faster than if only one UAV has been used. However, coordination issues arise when using more than one UAV for the search mission. The UAVs must be coordinated such that:

1. The UAVs does not collide with each other.
2. The UAVs does not search the same areas. A situation with too much overlap will be cause the search mission to be sub-optimal when it comes to search time.
3. When the UAVs have finished the search mission, they should return back to their respective base stations.

Based on these requirements, the optimal path planner should generate a path for all available UAVs, such that they are being used in the most efficient way to search the area where the fisherman is assumed to be located.

6.2 Implementation of search area

The operator of the UAVs has to know the area where the search mission should be performed. In the design developed in this thesis. this is done by assuming four

boundaries in the $\{n\}$ frame, which will limit the search area. The limits are given by $x^n_{min}, x^n_{max}, y^n_{min}$ and y^n_{max} , which the operator has to input in the path planner. The developed path planning system will automatically generate waypoints inside the search area based on the UAVs equations of motions and the camera system on-board the UAV. The waypoints should be close enough such that all the area between the waypoints are covered by the camera system, regardless of the order the waypoints are visited in. The distance between the waypoints will depend on the specifications on the camera system implemented on the UAV, and the number of waypoints will depend of the size of the search area. In this implementation it is assumed that the camera can search in the range r_{cam} meters around the the UAV in all directions.

6.3 Implementation of an anti-collisions system for missions with more than one UAV.

In this implementation it is not assumed that there are any stationary obstacles such as mountains, buildings and similar objects in the area where the search mission takes place. For some search mission in the ocean this can be a reasonable assumption. It is however possible to include obstacles in the MILP optimizer in the same way as was done in Chapter 3, it will only lead to more restrictions in the optimization problem. The main motivation with this chapter is to find methods which can be used to search a defined area, and for simplicity, obstacles will not be included in the path planner.

Since this implementation is designed to coordinate more than one UAV, the guidance system has to guarantee that the generated path will not lead to a collision between the UAVs.

To avoid collision between UAVs for situations where the number of UAVs on mission is more than one, the following restriction from [Richards and How, 2002] is implemented:

$$\begin{aligned} &\forall i \in [1,\ldots,N]\, \forall p,q | q > p \\ &x_{ip} - x_{iq} \geq d - Rc_{ipq1} \\ \text{and } &x_{iq} - x_{ip} \geq d - Rc_{ipq2} \\ \text{and } &y_{ip} - y_{iq} \geq d - Rc_{ipq3} \\ \text{and } &y_{iq} - y_{ip} \geq d - Rc_{ipq4} \end{aligned} \tag{6.1}$$

$$\text{and } \sum_{k=1}^{4} c_{ipqk} \leq 3$$

Remark: The UAVs should never get close to each other during the search mission, since they are covering parts of the same area if they are coming to close. they are covering parts of the same area. This will not be efficient. Therefore, the restriction in (6.2) should indirectly be a part of the system which generates the path. The restriction (6.2) should be included to make sure that the MILP does not generate a path which leads to a collision between the UAVs. If the UAVs are using the same base station, a restriction which provides that the UAV leaves the UAV base, and arrives at the UAV base at different times could also be implemented in order to avoid that the UAVs collide during take-off and landing.

6.4 Assumption

Since the weather is calm, it is further assumed that the currents in the ocean can be neglected, and therefore the fisherman is being assumed to be located at the same coordinates all the time during the search mission. This can be described by the following equations:

$$x^n_{\text{fisherman}}(t) = x^n_{\text{fisherman}}(0) \tag{6.2}$$
$$y^n_{\text{fisherman}}(t) = y^n_{\text{fisherman}}(0) \tag{6.3}$$

where $x^n_{\text{fisherman}}(0)$ and $y^n_{\text{fisherman}}(0)$ are the initial x- and y- coordinates in $\{n\}$, which is the position when the fisherman fell over board).

Remark: If there actually are small currents in the ocean and if the search operation can be done very fast compared to the currents, then the currents can be neglected by the path planner. If the currents are a dominating factor they should be taken into account by the path planner, such that the search area changes as a function of the currents during the search. One approach would be to include an extra safety margin for the search area. If the direction of the wind and currents are known, the search area should be increased in that direction such that it is considered that the search-object can move during the UAV search mission. A safety margin should also be included, which takes into account that there are disturbances such as wind and currents, unknown and unmodeled UAV dynamics and error in the given camera specifications. It should also be taken into account that the control system on-board the UAVs should be able to track the path generated by the path-planner, and that this represents an uncertainty, and therefore should be taken into account by the path planner. It is important to know that the camera range depends on the elevation of the UAV, since the camera will cover a sector to each side of the UAV, and this area will increase with an increase in the UAV elevation. If the camera is installed in the hull on the underside of the aircraft such that it rotates with respect to the $\{n\}$ frame when the UAV accelerates or is doing a turning, this will have an effect on which area that is covered by the camera. The impact of these events must be considered when the guidance system is designed. In this part of the chapter it is further assumed that the UAV has a defined elevation which it should use during the search mission. It is further assumed that the camera is mounted on a flexible rig with a stabiliser such that the camera is not rotated with respect to the $\{n\}$ frame when the UAV moves.

6.5 Development of a search algorithm to find the fisherman

In the following, an algorithm for search of a defined area will be presented. It is assumed that the operator at the rescue center has some idea of where the missing fisherman is located. The operator will insert the boundaries of the area where it is assumed that the fisherman will be found into the path planner. This area is described as a closed area with four restrictions, and the whole area inside the restrictions must be searched. When the search area is defined, the path planner will generate waypoints inside this area which makes sure that, when all the waypoints are visited, the whole area inside the square is covered by the

camera system which is available on the UAV. The distance between each waypoint therefore has to be given by the range of the camera on the UAV, and the dynamics of the UAV. It is important that the distance between the waypoints are as far as possible, to avoid that the same area is searched more than one time. It is also beneficial to use as few waypoints as possible to reduce the number of binary variables which has a negative effect on the computation time as described in Chapter 2.2. The cost function in the optimization algorithm will minimize the time used by the available UAV to arrive the search area, visit the waypoints and return back to base station when the search is completed. It is reasonable to apply a cost function which minimizes the search time, such that the fisherman is found as fast as possible.

6.6 Generation of waypoints

In this thesis the following algorithm for generation of waypoints has been developed, to ensure that the hole search area is covered by the UAVs camera system. The algorithm will ensure that the there are enough waypoints such that the defined search area is covered when the UAV has visited all the waypoints, independent of the order the waypoints are visited in. In the waypoint generation algorithm the only information that will be considered is the $x_{min}, x_{max}, y_{min}$ and the y_{max} value of the search area which is given by operator. As described earlier in this chapter, it may be suitable to include a restriction which forces the waypoints to be visited in a given order. Therefore the waypoint generation algorithm is developed in such a way that a restriction which forces the waypoints to be visited in specific order can be implemented in the problem.

The waypoints are generated and added in the table wp_{set} in the following way:

1. The following calculation is done to find the coordinates of the first waypoint:

$$(wp_{set,x,1} - x_{min})^2 + (wp_{set,y,1} - y_{min})^2 \leq r_{cam}^2 \cdot \gamma \tag{6.4}$$

   This can be written as

$$(\Delta x)^2 + (\Delta y)^2 \leq r_{cam}^2 \cdot \gamma \tag{6.5}$$

   where $\Delta x := wp_{set,x,1} - x_{min}$ and $\Delta y := wp_{set,y,1} - y_{min}$ is the distance to the x- and y- coordinates to the first waypoint from the boundaries x_{min} and y_{min} respectively.

   The distance to the first waypoint from x_{min} and y_{min} has been set to be equal, this means that (6.5) can be rewritten, and the coordinate of the first waypoint can be found:

$$\Delta x = \frac{r_{cam}}{\sqrt{2}} \Rightarrow wp_{set,x,1} = x_{min} + r_{cam}\sqrt{\frac{\gamma}{2}} \tag{6.6}$$

   and

$$\Delta y = \frac{r_{cam}}{\sqrt{2}} \Rightarrow wp_{set,y,1} = y_{min} + r_{cam}\sqrt{\frac{\gamma}{2}} \tag{6.7}$$

2. As long as not all the area from y_{min} to y_{max} is covered at the actual x-position by the camera, more waypoints will be added until y_{max} and the upper left corner is covered. This is done in the following way:

$$wp_{set,y,i+1} = wp_{set,y,i} + 2 \cdot r_{cam}\gamma \tag{6.8}$$

where $wp_{set,y,i+1}$ is the y-coordinate for the next waypoint and $wp_{set,y,i}$ is the y-coordinate of the last previously generated waypoint.

3. When the area at the actual x-coordinate is covered by the camera, the next waypoint is added at the same y-value, but on the next x-value given by the algorithm:

$$wp_{set,x,i+1} = wp_{set,x,i} + 2 \cdot r_{cam}\gamma \tag{6.9}$$

and the waypoints will be added from y_{max} to y_{min} in the following way:

$$wp_{set,y,i+1} = wp_{set,y,i} - 2 \cdot r_{cam}\gamma \tag{6.10}$$

To decide if the waypoints are added from low y-values to high y-values or from high y-values to low y-values, the *modulo* operator has been applied:

$$x_{counter} \cdot \mathrm{mod}(2) = x_{counter} - \left\lfloor \frac{x_{counter}}{2} \right\rfloor \cdot 2 \tag{6.11}$$

$x_{counter} \cdot \mathrm{mod}(2)$ will return 0 if $x_{counter}$ is an odd number, and will return 1 if $x_{counter}$ is an even number. This property is used in the waypoint generation algorithm to decide if the waypoints should be added with increasing y-values or with decreasing y-values, since for odd numbers the waypoints are added from lower y- boundary (y_{min}) to the upper y-boundary y_{max} and for even numbers the waypoints are added from the upper y-boundary (y_{max}) to lower y-boundary (y_{min}). If the waypoints are visited in the same way as they are ordered in the wp_{set} table, this will result in a *zig-zag* path. The algorithm which is developed for waypoint generation is given as a flowchart in Figure (6.1). The implementation in MATLAB is given in Appendix (C)

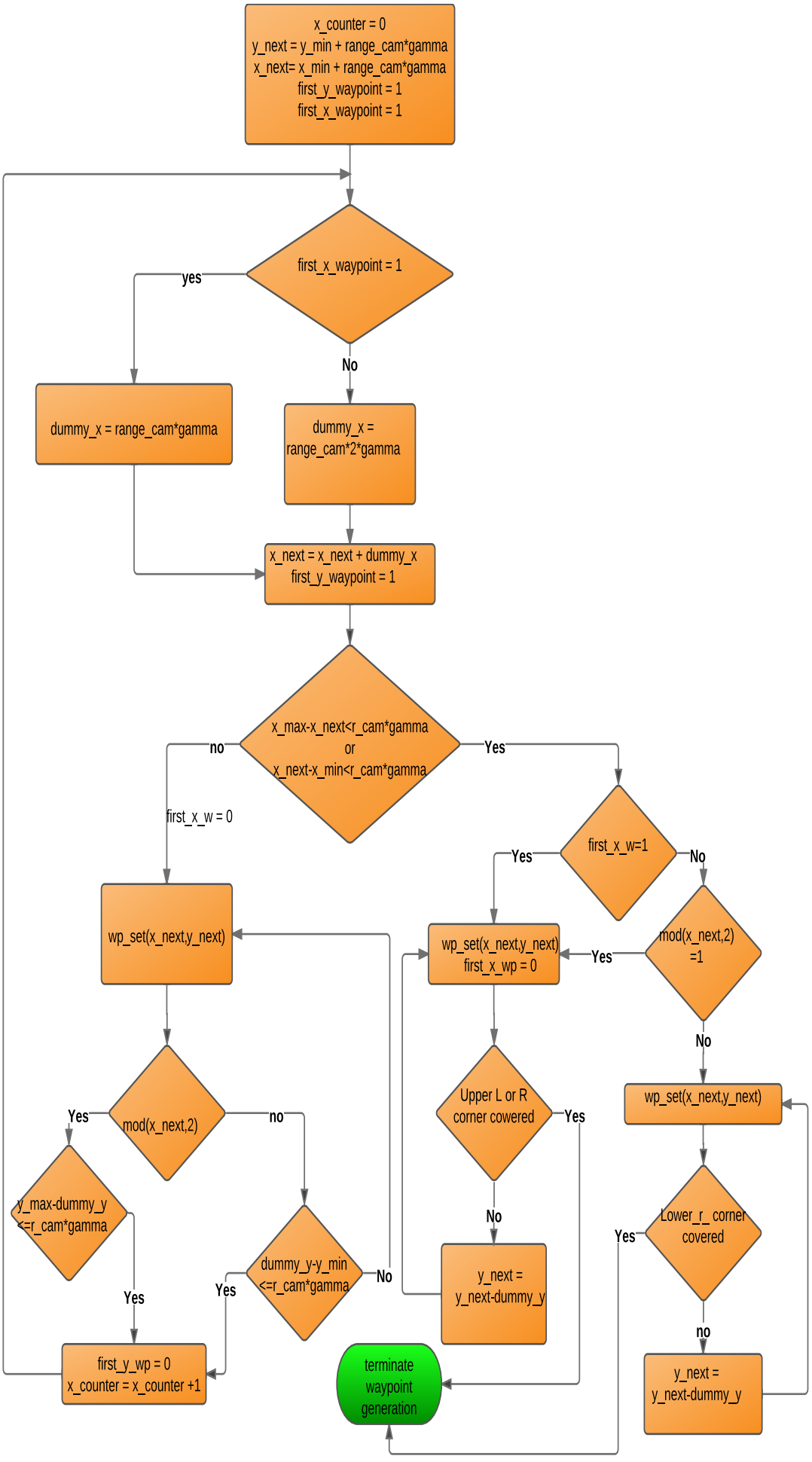

Figure 6.1: Flowchart which shows the developed algorithm for waypoint generation

6.7 The optimization problem in MILP

Since the search area has been defined and waypoints have been generated to ensure complete cover of the search area, one need to define the MILP such that the UAV behaves as desired. In this section the desired behaviour will be described and implemented.

In this implementation it is important to find optimal solution which minimizes the time the UAV uses from it's base, through all the waypoints and back to the UAV base, to ensure that the fisherman is found as fast as possible. Therefore, in this implementation one needs to define the cost function, such that it minimizes the search and rescue time. Since this implementation should be able to handle multiple aircrafts, equation (3.67) from [Grøtli and Johansen, 2011a] is applied in extended form to take into account multiple aircrafts:

$$\forall p \in \{1, ..., n_p\}, \forall i \in \{1, ..., N\} \tag{6.12}$$

$$\theta_p^{finish} \leq M^{finish}\left(1 - b_{piw}^{wp}\right) + ib_{piw}^{wp} \tag{6.13}$$

$$\theta_p^{finish} \geq i\left(1 - b_{piw}^{wp}\right) \tag{6.14}$$

Where M^{finish} is a constant which is sufficiently large, for example $M^{finish} := N$ as suggested in [Grøtli and Johansen, 2011a].

$$\eta^{finish} \geq \theta_p^{finish} \tag{6.15}$$

The objective is to minimize

$$J^{finish} = \eta^{finish} \tag{6.16}$$

such that all available aircrafts returns to the base as fast as possible.

In the implementation, the waypoints which are going to be visited are inserted in the wp_{set} table as described in the algorithm above. At the end of the table the UAV base for each of the UAVs participating on the mission are included. If it is only one UAV on the mission only one UAV base will be included in the table. If there are two UAVs, the last two rows in wp_{set} will contain the x- and y- coordinates for both UAVs respectively. The restriction (3.57), which makes sure that all waypoints in the search are visited, has to be modified since the last element(s) in W_c includes the base station(s). It is not desirable to include a restriction that allows the UAV base to be visited for only one time sample. It is therefore reasonable to modify this restriction such that the UAV is allowed to stay at the base station for more than one time sample. Equation (6.17) is a restriction which enforces all waypoints and base stations to be visited, and (6.18) is a restriction which allows each waypoint to only be visited once by one of the UAVs. This means that only one time sample is allowed to be inside the range Δ around each waypoint, and that is desirable.

$$\begin{aligned}
&\forall i \in [1, \ldots, N]\, \forall c \in [1, \ldots, W]\, \forall p \in [1, \ldots, n_p] \\
&\quad x_{ip} - x_{Wc} - \Delta \leq M_{big}(1 - b_{icp}) \\
&\quad x_{ip} - x_{Wc} - \Delta \geq -M_{big}(1 - b_{icp}) \\
&\quad y_{ip} - y_{Wc} - \Delta \leq M_{big}(1 - b_{icp}) \\
&\quad y_{ip} - y_{Wc} - \Delta \geq -M_{big}(1 - b_{icp})
\end{aligned} \tag{6.17}$$

$$\forall c \in [1, \ldots, W - n_p]$$
$$\sum_{p=1}^{n_p} \sum_{i=1}^{T} b_{icp} = 1 \tag{6.18}$$

The following restriction (6.19) is included to make sure that the UAV's base station is visited, by the respective UAVs, and that there are no upper bound for how long it can stay there.

$$\forall p \in [1, \ldots, n_p] \quad \forall c \in [W - n_p + 1, \ldots, W]$$
$$\sum_{i=1}^{T} b_{icp} \geq 1 \tag{6.19}$$

The following restriction is included to restrict the UAV to stay at the base station for all future time steps, when it arrives the base.

$$\forall p \in [1, \ldots, n_p], \forall i \in [1, \ldots, N-1] \forall w \in [W - n_p + 1, \ldots, W]$$
$$b_{(i+1)cp} \geq b_{icp} \tag{6.20}$$

The reader might think that the restrictions given in (6.19) and (6.20) prohibit the UAV for leaving the base station. since these restrictions are fulfilled when the program starts, one could think that the binary variable for the base station than equals one, and that restriction (6.20) prohibits the UAV to leave the base. However, the other restrictions (6.17) which tell us that the waypoints in the search area needs to be visited results in a feasible program only if the UAV leave it's base station and visit these waypoints. Therefore the only feasible solution in the overall MILP optimization is that (6.19) and (6.20) are fulfilled after all the waypoints has been visited.

The constraints described in (3.54) and (3.55) has to be extended to include more aircrafts. It is further assumed that the search and rescue mission will consist of three phases which in this chapter will be implemented in the path planner. It is assumed that the first 3 time steps is a part of the take-off phase. In this part the UAV should be allowed to have a lower limit on the speed, $V_{min} = 0 \left[\frac{m}{s}\right]$. The steps from 4 and to N-3 is the phase when the UAV is airborne and in this part $V_{min} = 5 \left[\frac{m}{s}\right]$. The last tree time steps will be the landing phase and at this phase $V_{min} = 0 \left[\frac{m}{s}\right]$. The implementation is done as follows:

Constraints phase 1:

$$\forall i \in [1, \ldots, 3] \forall m \in [1, \ldots, M]$$

$$V_{min,1} \leq v_{xi} \sin\left(\frac{2\pi m}{M}\right) + v_{yi} \cos\left(\frac{2\pi m}{M}\right) \leq V_{max} \tag{6.21}$$

Constraints phase 2:

$$\forall i \in [4, \ldots, N-3] \, \forall m \in [1, \ldots, M]$$

$$V_{min,2} \leq v_{xi} \sin\left(\frac{2\pi m}{M}\right) + v_{yi} \cos\left(\frac{2\pi m}{M}\right) \leq V_{max} \tag{6.22}$$

Constraints phase 3:

$$\forall i \in [N-2, \ldots, N] \, \forall m \in [1, \ldots, M]$$

$$V_{min,3} \leq v_{xi} \sin\left(\frac{2\pi m}{M}\right) + v_{yi} \cos\left(\frac{2\pi m}{M}\right) \leq V_{max} \tag{6.23}$$

Where $V_{min,1} = V_{min,3} = 0 \left[\frac{m}{s}\right]$ and $V_{min,2} = 5 \left[\frac{m}{s}\right]$.

6.8 Discussion

The time it takes from the operator inserts the defined search area in the program, and until the program has found the optimal path, and are ready to start on the mission, is important. It is therefore reasonable to find ways to minimize the computation time. The computation time to find the optimal route is time consuming if the number of waypoints in the problem is very large, as described in Chapter 2.2. If the search area becomes large, and the camera range (r_{cam}) is small, the time to calculate the optimal path can be very time consuming. In these cases it it can be sensible to discuss if it is worth the spent time it takes to find the optimal route, since in the fixed horizon approach MILP applied in this thesis has to be solved before the UAV can start on the mission. The author purpose that the question about optimality also has to take into account the time it takes to calculate the optimal path, since the overall objective is to search and rescue the fisherman as fast as possible. If it takes longer time to calculate the optimal path, and fly that path, then it takes to do visit the waypoints in some arbitrary order, it would be not be optimal to do the optimization since the search could than have been done in a more efficient way with respect to time. If the optimization problem can be simplified in some ways which reduces the computation time, this could be the preferable solution. One suggestion given in this section was to implement one more restriction in the optimization problem, which determines the order of the waypoint visitation. It was argued earlier in this chapter that this could result in a reduced computation time for the optimization problem.

To make the number of possible paths in the problem smaller, a restriction which forces the order the waypoints should be visited could be included in the model. The following restriction (6.24) tells that the waypoints should be visited in the same order as they where included in the path planner [Grøtli and Johansen, 2011a].

$$\theta_{c+1} \geq \theta_c \tag{6.24}$$

where θ_c is the time it takes to reach waypoint c.

When the restriction (6.24) is included in the optimization problem, it is sensible to assume that this will result in a a cost function which has the same or higher

value as before the restriction was included, since the objective is to minimize the cost function. Since the cost function in this case study represents the time it takes to fulfil the the mission, one should expect that the solution will be a path which it can take longer time to travel. If the decreased computation time achieved by implementing (6.24) is lower than the the extra time it takes to travel the new path, the author propose this restriction to be included, since it will reduce the overall time to complete the search.

If the UAV is stationed on a base station and the the mission is to search a fixed area regular, it would be preferable to solve the optimization problem without restriction (6.24), since this will ensure as low cost function as possible. Since the same are is going to be search many times, it would be sensible to do find the path which minimize the time consumption, even if it takes some time to compute this path. Since the UAV for these missions are flying the same route many times, will it be preferable to find an optimal path once, even if it takes some time do solve the optimization problem. For the purposes in this case-study, the search area will change between the various missions, and therefore the optimization problem has to be solved just before take off.

The available computing power on-board the UAV is assumed to be very limited. Large optimization problems which needs to be computed before the UAV leaves the base, will take a long time, compared to the time it takes to solve the same problem on a computer with higher computing power. In the approach which is considered in this thesis, the whole path from initial position and to the final position is found before the UAV leaves the base station. It may therefore be appropriate to solve the optimization problem on a computer with high computing power at the base station and transfer the solution to the UAV when it is found. If the approach used to generate the path is further developed to also recompute the path during the flight, this could be implemented such that the computer on-board the UAV can re-optimize the original path during the flight.

6.9 How to design the horizon, T in the MILP

The only operation the UAV operator should have to do, is do define the lower and the upper values of the x- and y- boundaries in the search area as described earlier in this chapter. Since the implementation in this thesis considers a fixed horizon, it should be expected that the program is able to design the necessary horizon for each respective search mission. The horizon has to be long enough such that the UAV is able to get out to the search mission, do the search and get back to the UAV base. Otherwise the MILP will become infeasible. As will become clear in the next section, the horizon length has substantial influence on the computation time, and it is therefore beneficial to initialize the optimization problem with as small a time horizon as possible to get the computation time as small as possible. If a restriction is included to determinate the order of the waypoint visitation, as given in (6.24) the author propose the following approach for finding the time horizon, T:

- Find the total length of the path from the UAV base, through all the waypoints in the given order, and back to the UAV base. This length will be denoted s_{path}.

- Since the objective of this mission is to minimize the time spent on the

mission, it is natural to assume that the speed of the UAV will be close to the maximum speed during the search mission.

- The lower bound on the time horizon can then be found by considering the following calculation:

$$T \leq \frac{s_{path}}{V_{max}}\Gamma_1 \qquad \Gamma_1 \geq 1 \tag{6.25}$$

where T is the time horizon which one wants to find an upper bound on, s_{path} is the distance given by straight lines from the UAV base, through the waypoints in the determined order, and back to UAV base. V_{max} is the maximum speed the UAV can have. Γ_1 is a constant which gives a safety margin on the upper bound for the time horizon.

- The lower bound can be found with the following calculation:

$$T \geq \frac{s_{path}}{V_{min}}\Gamma_2 \qquad \Gamma_2 \geq 1 \tag{6.26}$$

where Γ_2 is a constant which gives a safety margin on the lower bound for the time horizon.

It is important that the time horizon does not become too short, since the optimization problem might become infeasible. Therefore a safety margin should be included to take into account situations where the UAV do not fly at maximum speed all the time (needs to have acceleration for take-off and when it arrives the UAV base). A Γ constant which gives a safety margin on 10% (this gives $\Gamma = 1.10$) can be included. The γ variable should be found during experiments to find out

Remark: For the case where the optimization problem finds the optimal path with no restrictions on the order of waypoint visitation the restriction in (6.25) can still be used as an upper bound, since an optimization problem with one less restriction included will have higher degree of freedom, and will therefore have an optimum which is equal or better then the optimum which was found with the restriction included.

6.10 Simulation

In this section simulations from search and rescue missions will be presented and discussed. The following items have been considered during the simulation:

- Simulations of a search mission when one UAV is applied in a search mission
- Simulations of a search mission where two UAVs are applied in a search mission, to show that the path planner handles more than one UAV. One of the UAVs is located in $(x, y) = (0, 0)$ and the other UAV is located in $(x, y) = (400, 0)$
- Identify the time it takes to solve the optimization problem when the restriction (6.24) is implemented and when it is not implemented.

- Identify the time it takes from the Gurobi solver starts computation on the optimization problem, and until the UAV has fulfilled the search along the generated path. These results will be applied in the discussion on the overall time it takes to complete a search mission, for the case where the restriction on waypoint visitation is included in the problem, and for the case when it's not included in the problem. The time it uses to finish the search has been found by investigating the time the UAV arrives at the base station. This is found by checking if the binary value describing if the base station is visited or not, is set to 1. The sample number (k), the UAV arrives the base station on, is multiplied by the discretization time T_d to calculate the flight time in seconds.

- Identify that the camera system on the UAV covers the defined search area to verify that the waypoint generation algorithm described in Section 6.6 works efficiently. The camera range, r_{cam}, is changed to verify that the waypoints is placed in a different way to cover the search area.

The implementation has been done such that a waypoint is registered as visited when a time sample is placed exactly at the waypoint. This means that $\Delta = 0$ in equation (6.17). The discretization time (T_d) is 3 seconds in all simulations in this section.

6.10.1 Case 1 - one UAV and no restriction on the order of waypoint visitation

The first search mission considers a search and rescue mission where one UAV participate in the search, and it's base is located in the following coordinates: (x,y) = (0,0). The operator assumes that the missed fisherman is inside the rectangle given by the boundaries $x_{min} = 0, x_{max} = 850, y_{min} = 50$ and $y_{max} = 550$.
The camera range r_{range} has been set to 200 meters, and the value β in the waypoint generation algorithm has been set to 0.8.

The optimal path when there are no restrictions on what order the waypoints are visited in is given in Figure (6.2).

The time it takes to solve the optimization problem and fly the calculated path took in this case 209 seconds.

In the simulation the time horizon was set to be 150 seconds. The MILP problem took 74 seconds to be solved. The UAV is back in it's base after 135 seconds, which means that the total search time took 209 seconds. Figure (6.3) shows the same path with the camera range included as blue circle around each time sample. As can be seen from the plot, the whole search area is covered by the UAV camera system.

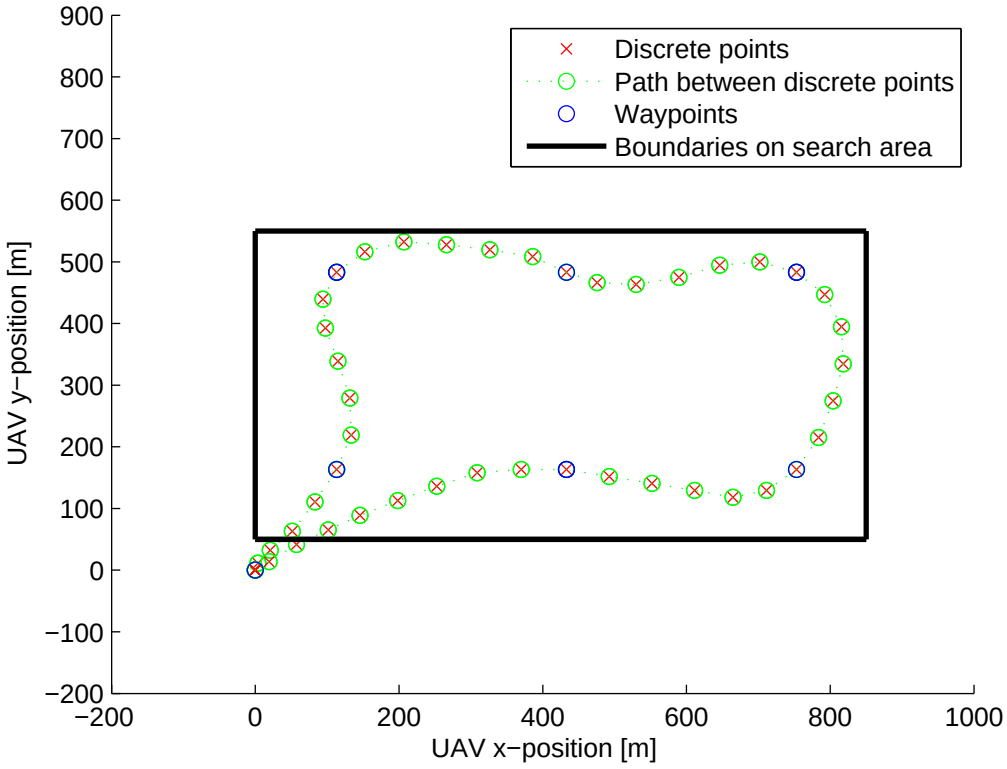

Figure 6.2: Simulation for a search with one UAV with base in (0,0). The search area is defined by $x_{min} = 0, x_{max} = 850, y_{min} = 50$ and $y_{max} = 550$. The computation time took 74 sec. Time horizon, T = 150 seconds, discretization time $T_d = 3$. Camera range = 200 meters, $\gamma = 0.8$ Reaches base on sample 45

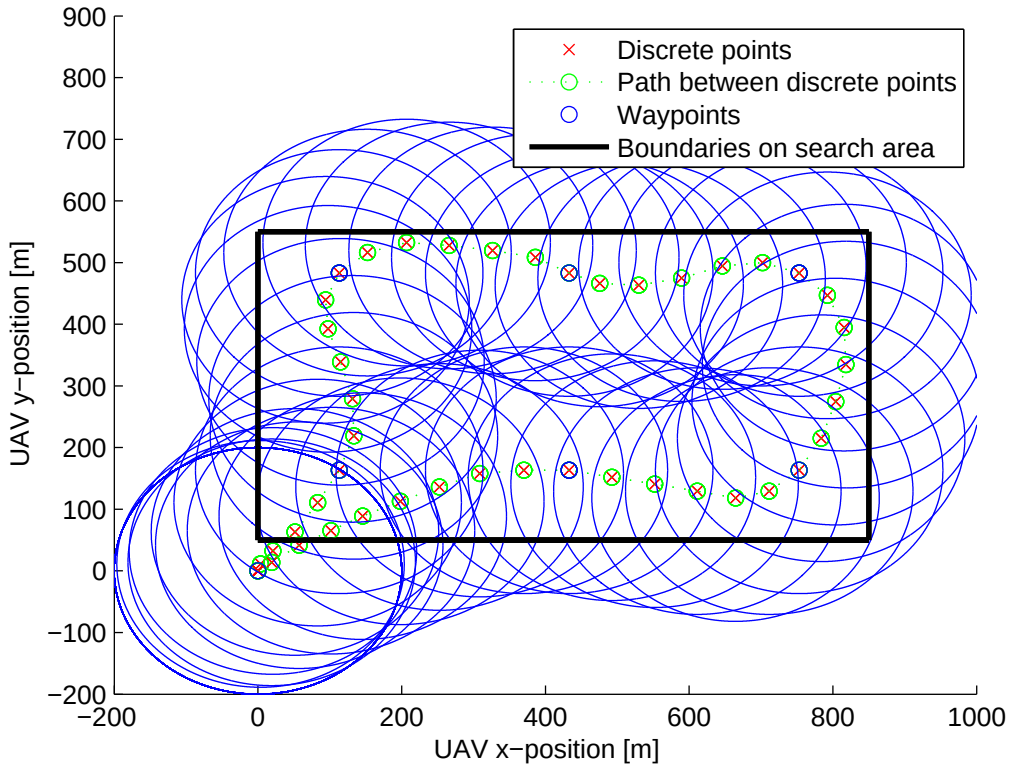

Figure 6.3: Simulation for a search with one UAV with base in (0,0). The search area is defined by $x_{min} = 0, x_{max} = 850, y_{min} = 50$ and $y_{max} = 550$. The computation time took 74 sec. Time horizon, T = 150 seconds, discretization time $T_d = 3$. Camera range = 200 meters, $\gamma = 0.8$ Reaches base on sample 45 The blue circles shows the camera range at every time sample.

Remark: If the horizon was increased to 180 seconds, the MILP optimizer used 327 seconds to solve the same problem. This illustrates that the time horizon is an important factor for the computation time

6.10.2 Case 2 - one UAV and a restriction on the order of waypoint visitation

This Case considers the same scenario as Case 1, but in this case the restriction on waypoint visitation (6.24) is included in the optimization problem. The time horizon (T) was in this case increased to 180 seconds, to be able to solve the optimization problem. The computation time took in this case 24 seconds and the UAV used 156 seconds to fly the search mission. This means that the overall time, when both computation time and flight time is included, took 180. seconds. The path is shown in Figure (6.4). In this case one can see that it takes longer time to do the flight search compared to the case where no restrictions in the waypoint order is added. This make sense because when one add one more constraint to the optimization problem the time it takes to do the flight should be longer or equal to the time when the constraint was not included. Figure (6.5) shows the generated path with the camera range included around each time sample.

The time it takes to solve the optimization problem and fly the calculated path took in this case 180 seconds.

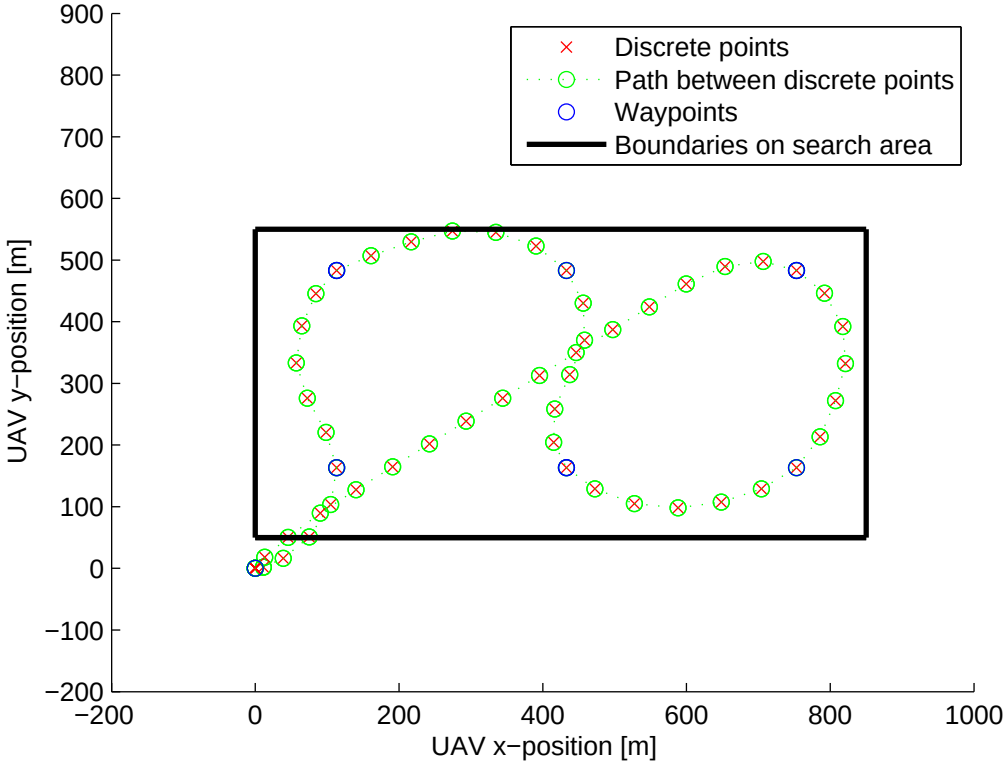

Figure 6.4: Search with one UAV with base in (0,0). $x_{min} = 0, x_{max} = 850, y_{min} = 50$ and $y_{max} = 550$ Computation took 24 sec. T = 180, $T_d = 3$. $r_{cam} = 200$, $\gamma = 0.8$ Reaches UAV base at time step 52

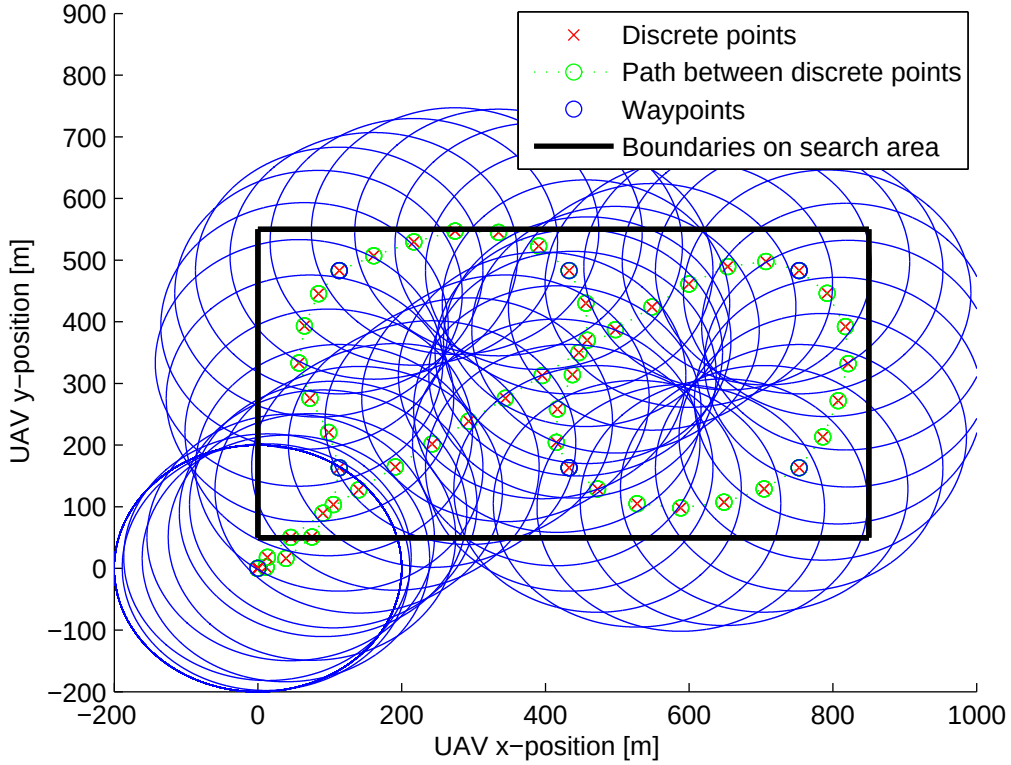

Figure 6.5: Search with one UAV with base in (0,0). $x_{min} = 0$, $x_{max} = 850$, $y_{min} = 50$, $y_{max} = 550$ Computation took 24 sec. T = 180, $T_d = 3$. $r_{cam} = 200$, $\gamma = 0.8$ Reaches UAV base at time step 52

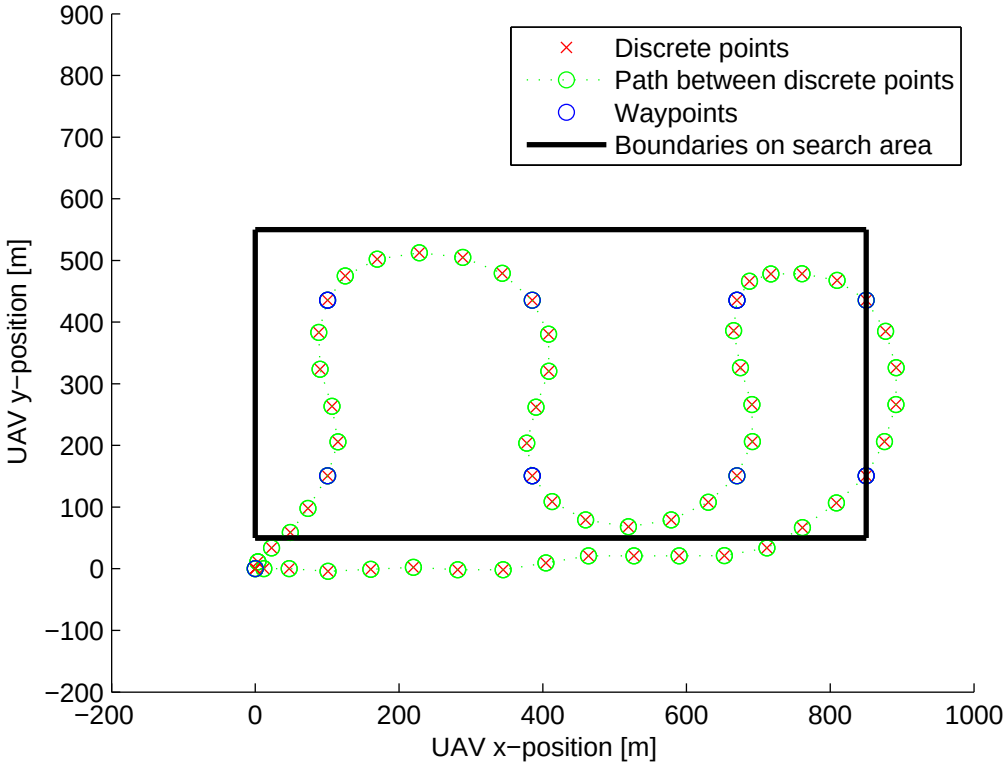

Figure 6.6: Search with one UAV with base in (0,0). $x_{min} = 0$, $x_{max} = 850, y_{min} = 50, y_{max} = 550$ Simulation took 29. T = 180, $T_d = 3$. $r_{cam} = 178$, $\gamma = 0.8$ Reaches base station at time step 58

6.10.3 Case 3- the camera range, r_{cam}, is changed

If the camera on the UAV has a smaller range, this will result in closer waypoints which is given from the from the waypoint generation algorithm. In Figure (6.6) the search area was defined to be the same as in the simulations above, but the camera range has been set to be 178 meters, which is 22 meters smaller than the simulations in Case 1 and Case 2. As can be seen from Figure (6.6), this new camera range results in more waypoints which has to be visited to ensure that the area is covered by the camera. In this case the horizon was changed to be 180 seconds, since a time horizon on 150 second would result in an infeasible optimization problem. The computation time took 185 seconds.

The time it takes to solve the optimization problem and fly the calculated path took in this case 354 seconds.

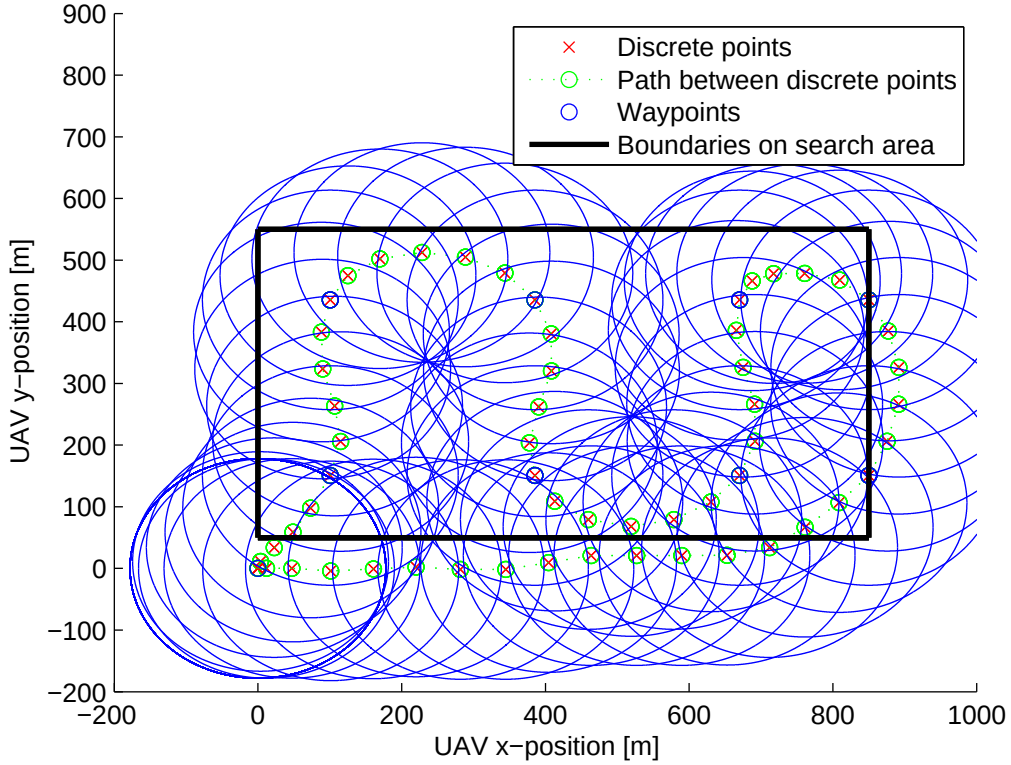

Figure 6.7: Search with one UAV with base in (0,0). $x_{min} = 0$, $x_{max} = 850$, $y_{min} = 50$, $y_{max} = 550$ Simulation took 29. T = 180, $T_d = 3$. $r_{cam} = 178$, $\gamma = 0.8$ Reaches base station at time step 58

6.10.4 Case 4 - Two UAVs, no restriction on the waypoint visitation

In some search missions there might be more than one UAV available. In the implementation of the MILP optimizer it is taken into account that it can be arbitrarily UAV taking part in the search mission. A simulation with two UAVs are shown in Figure (6.8). In Figure (6.9) the camera range has been plotted with a circle around each UAV sample to show that the whole search area has been covered by the camera during the search. There is no upper restriction on how many UAVs that can be implemented in the model, but the base station for each UAV must be included.

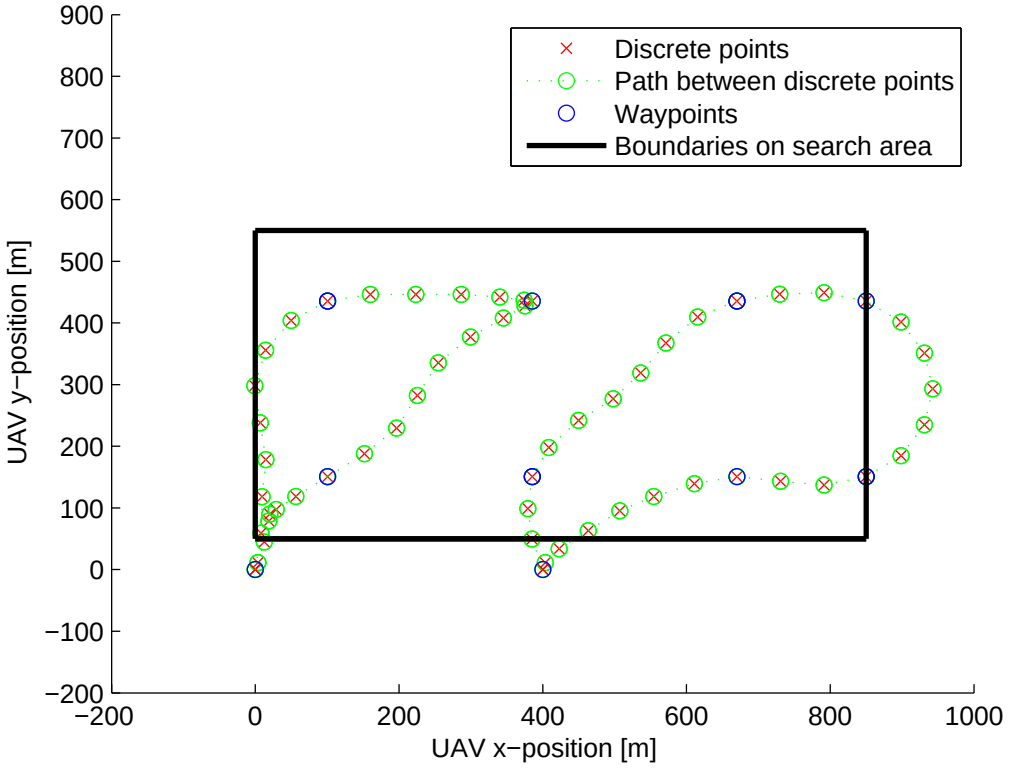

Figure 6.8: Search with UAV number one with base in (0,0) and UAV number two in (400,0). The search area was bounded by: $x_{min} = 0, x_{max} = 850, y_{min} = 50$ and $y_{max} = 550$ Optimization took 7 seconds. T = 90, $T_d = 3$. $r_{cam} = 178$, $\gamma = 0.8$

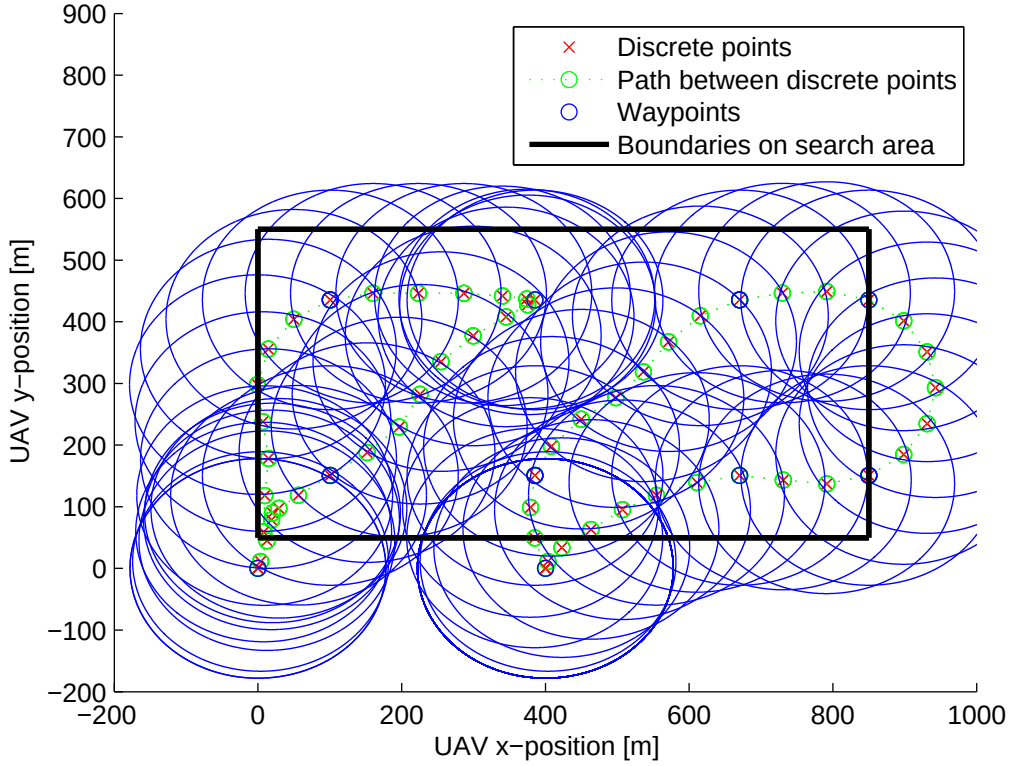

Figure 6.9: Search with UAV number one with base in (0,0) and UAV number two in (400,0). The search area was bounded by: $x_{min} = 0, x_{max} = 850, y_{min} = 50$ and $y_{max} = 550$ Simulation took 7 seconds. T = 90, $T_d = 3$. $r_{cam} = 178$, $\gamma = 0.8$

6.10.5 Case 5 - Two UAVs and a restriction on the order of waypoint visitation

In Figure (6.10) there is included a restriction which tells that the UAV with base in $(0, 0)$ should visit the first half-part of waypoints in chronological order, and that the UAV with base in $(400, 0)$ will should the last half-part of waypoints. When the horizon was 90 seconds the optimization took 6 seconds. The main reason for this short optimization time with two UAVs is because it is than possible to use a shorter horizon since the mission is divided between two UAVs.

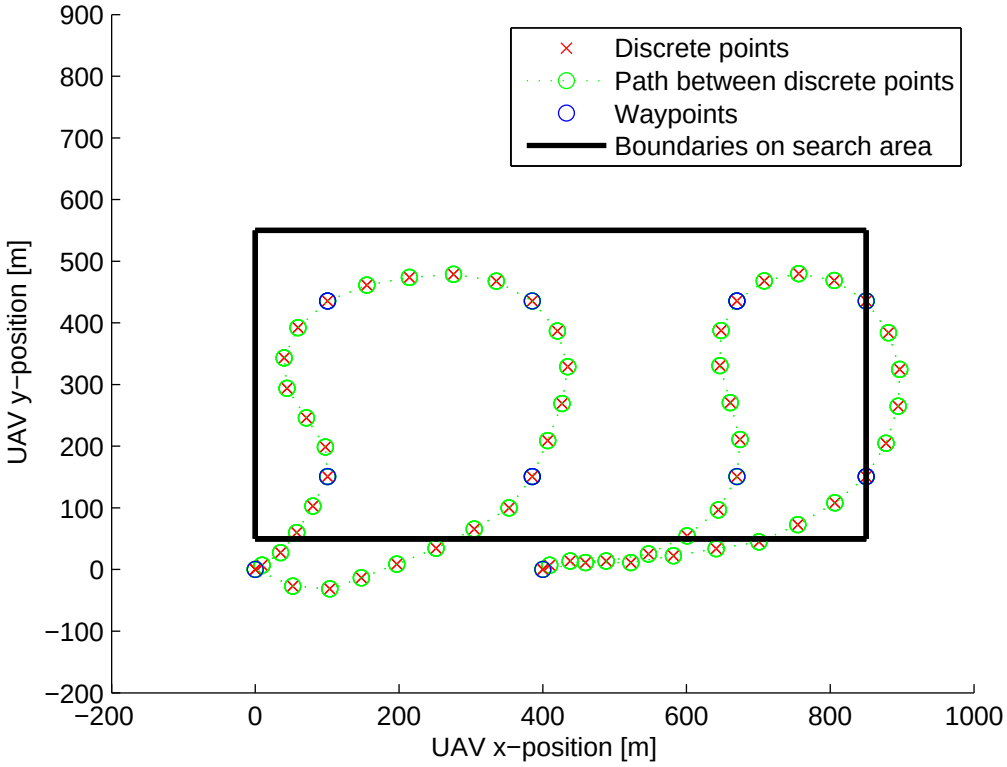

Figure 6.10: Search with one UAV with base in (0,0) and one in (400,0). $x_{min} = 0$, $x_{max} = 850, y_{min} = 50, y_{max} = 550$ Simulation took 80 sec. T = 90, $T_d = 3$. r_{cam} = 178, $\gamma = 0.8$.

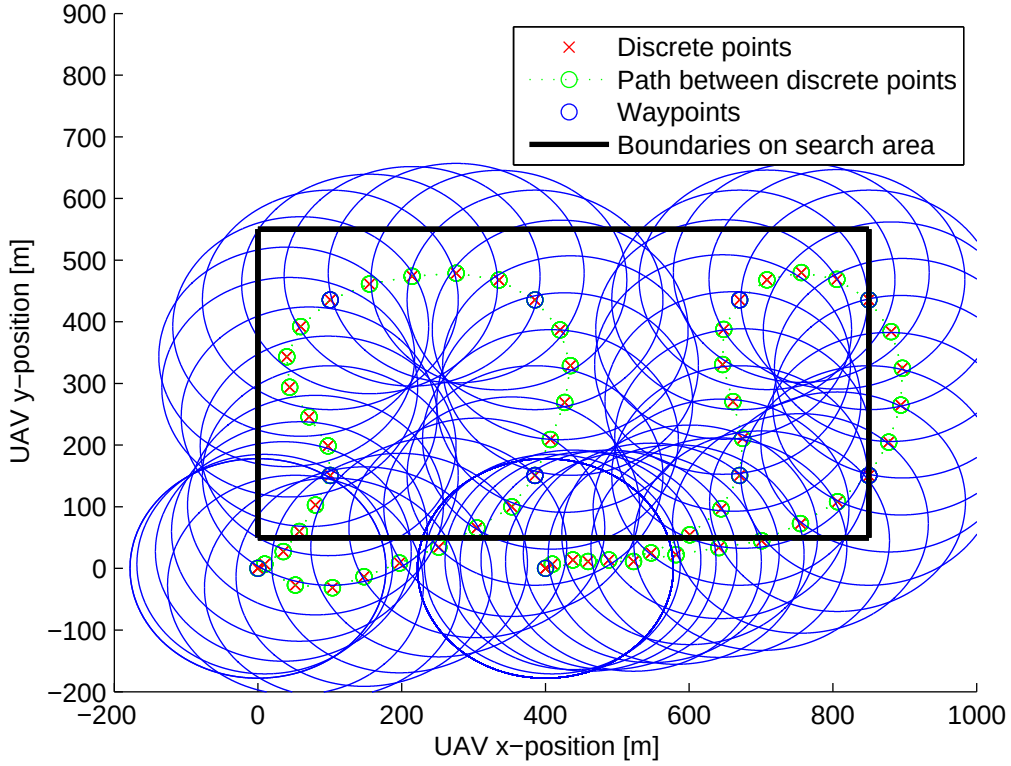

Figure 6.11: Search with one UAV with base in (0,0) and one in (400,0). $x_{min} = 0$, $x_{max} = 850, y_{min} = 50, y_{max} = 550$ Simulation took 80 sec. T = 90, $T_d = 3$. r_{cam} = 178, $\gamma = 0.8$.

Chapter 7

Search-mission by applying an Archimedes Spiral

When designing a path for a search and rescue mission it can be suitable to try some other methods than the one with MILP optimization. As was shown, the optimization was time consuming when the MILP approach was applied with many waypoints in the optimization problem. As was discussed in Chapter 6, the overall time spent to find the fisherman, from the mayday is received, and to the search is finished is important, since the overall object is to find the fisherman as fast as possible. When the path was generated by the MILP optimizer, it took some time to compute the optimal path, and the computation time becomes more dominant as the search area becomes big, and the number of waypoints becomes large. Since the time it takes from an actual search is determined and to the search is completed is the essential factor, the author is motivated to find other less time-consuming approaches to determine the path for a search and rescue mission. In [Al-Helal and Sprinkle, 2010] it is argued that a spiral could be a good approach to be a path in search missions. In this chapter the Archimedes Spiral will be further investigated, and it will be developed such that it automatically is adapted to different search and rescue missions.

7.1 The Archimedes Spiral

The Archimedes spiral can in polar coordinated be described as

$$r = a + b \cdot \theta \quad \forall a, b, \theta \geq 0 \tag{7.1}$$

where r is the distance from the center of the spiral after θ successive turnings. In this case the spiral starts at the distance a from the spiral center and ends after θ rotations. the impact the a and the b constants has on the spiral can be seen in Figure 7.1. The following calculation has been done to find this relationship:

$$r(\theta + t\pi) - r(\theta) = (a + b(\theta + 2\pi)) - (a - b\theta) = 2\pi b \tag{7.2}$$

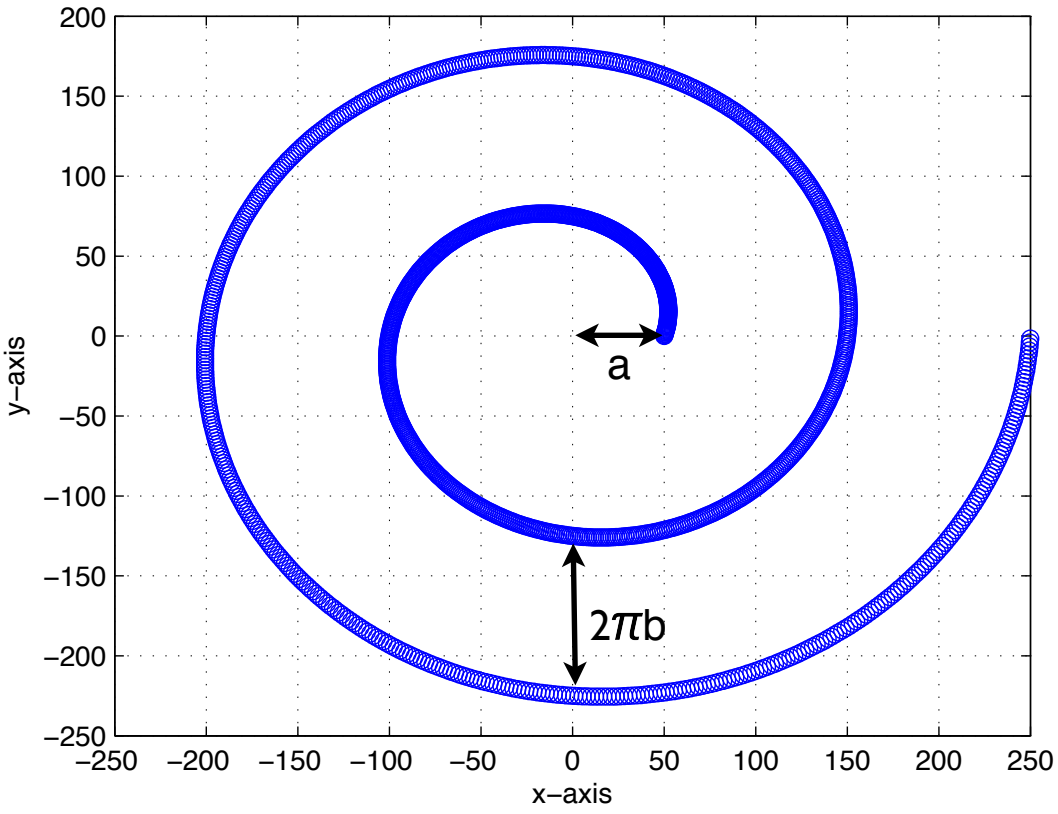

Figure 7.1: The Archimedes Spiral and the properties of the a and b constants. In this figure $a = b = 50$.

7.2 Adaptation of the Archimedes spiral for the search mission

If one wants to define the spiral such that it instead start on the outermost point and ends at the innermost point it can be written as

$$r = \rho - (a + b \cdot \theta) \qquad \forall a, b, \theta, r \geq 0 \tag{7.3}$$

where the last restriction on r in (7.3) gives the following upper bound on θ

$$r = \rho - (a + b \cdot \theta) \geq 0 \Longrightarrow \theta \leq \frac{\rho - a}{b} \tag{7.4}$$

where ρ is the distance to the outermost part of the spiral from the search center,a is a constant which determine at which radius from the origin the spiral should end, and b determine the distance between the successive turnings. These constants has to be decided such that the spiral, which is the path the UAV is suppose to follow, is feasible.

The parameters also has to be decided in proportion to the camera-system on the UAV, which searches the surface of the earth. This has to be done such that the hole area between the successive turnings are covered by the camera system. Since the dynamics of the UAV are fixed, the camera has to be designed such that it covers the area between the successive turnings, since the UAV can not be designed to satisfy the camera specifications.

When (7.3) is implemented this will give a spiral which starts at the outermost point of the search area, and the radius to the center of the spiral decreases with increased θ.

7.3 Assumption

- It is assumed that the probability to find the fisherman is equal in the whole search area. In this thesis the implementation of the search mission has been chosen to start at the outermost point, and work towards the innermost point. However, if the reader wants to start the search in the center instead, this will only result in a small modification of the implementation.

- In some search missions, it is perhaps more suitable to start in the center of an area and do the search outwards until one locates the object one is searching for. In other search mission the search area can be defined to be inside some arbitrary boundaries in the $\{n\}$ frame. And this is assumed for this case study.

- Since the issue of the search and rescue mission is to locate the fisherman as fast as possible, the object of the path planner is to make a path which can search the defined area as efficient and as fast as possible. It obvious that it is preferable to let the UAV fly as fast as possible. If it starts at the outermost point of the search area it obvious that it is less probable that the UAV needs to slow down the speed, than if it starts at the innermost part of the search area since the curvature κ is smaller there. κ is given by (7.5) [Edwards and Penny, 2002]. If the guidance system was implemented such that the UAV should start the search at the innermost point, one unwanted effect of this could be that the UAV has to slow down to be able to follow the path, since the curvature of the spiral is biggest at this point. When the radius from the center increases the curvature will become smaller and the UAV can again increase the speed. Taking into account that the fuel consumption also should be considered, it will be most economic to start at the outermost point of the search area. The curvature is given by:

$$\kappa = \frac{1}{r} \tag{7.5}$$

  where r is the distance from the center of the spiral to the point of the spiral given by (7.3).

Another approach the reader could think about, is to apply optimization techniques which finds the optimal value for κ such that the curvature is developed such that it is optimal with respect to the UAV dynamics. If the curvature on the innermost point is such that the UAV can fly at maximum speed, then it will be beneficial to start the search at the innermost point.

7.4 Implementation of the spiral-search approach

In the spiral search approach, the Archimedes Spiral is applied $r = \rho - (a + b \cdot \theta)$, where a and b are decided constants, and θ is the angle to the point on the spiral at the distance r from the center.

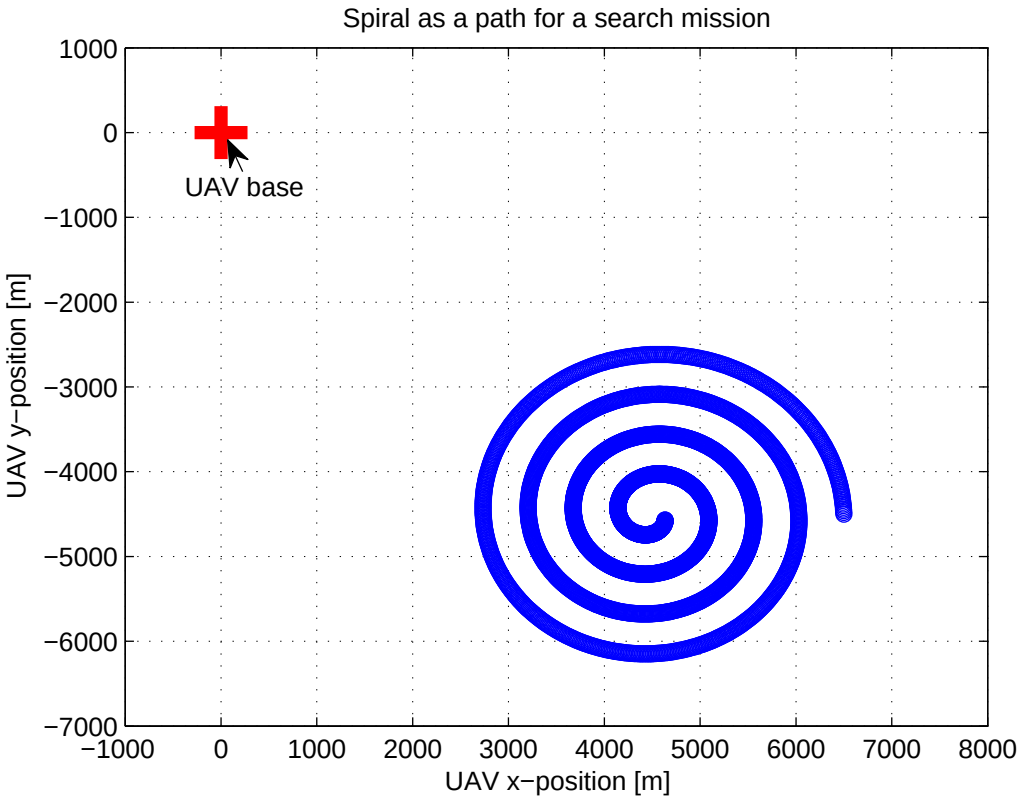

Figure 7.2: Illustration of a bad oriented spiral path with respect to the UAV base
.

The constants a and b has to be decided such that the spiral path is fulfilled with respect to both the UAV dynamics and the range of the camera which is used in the rescue search. The UAV operator needs to insert a center for the search operation in coordinate system $\{n\}$, and a radius around this center, where it is likely to find what one is looking for within. The base station is the UAVs initial position and it is known by the guidance system. The center for the search will be the center of the spiral path.

The author propose that the spiral is designed such that the orientation is adapted to the UAV base station. As one can see from Figure 7.2, the orientation is not adapted to the UAV base in this case. The spiral should be designed such that the UAV can fly directly into the spiral when it arrives the search area from the UAV base.

In the following the author propose an algorithm for automatic design of the Archimedes spiral:

The center for the search area has a local reference coordinate system $\{s\}$ which is rotated with an angle ψ with respect to the earth fixed coordinate system $\{n\}$. The spiral will in the following be generated in this frame. The issue of the algorithm is to rotate $\{s\}$ such that the outermost point on the spiral starts on a point which is related to the UAV base station. In Figure 7.3, a spiral is oriented such that the UAV is able to fly straight into it when it arrives the search area. This angle ψ is decided such that the vector product of the vector from the UAV base to the spiral center in $\{n\}$ and the vector from the spiral center to the

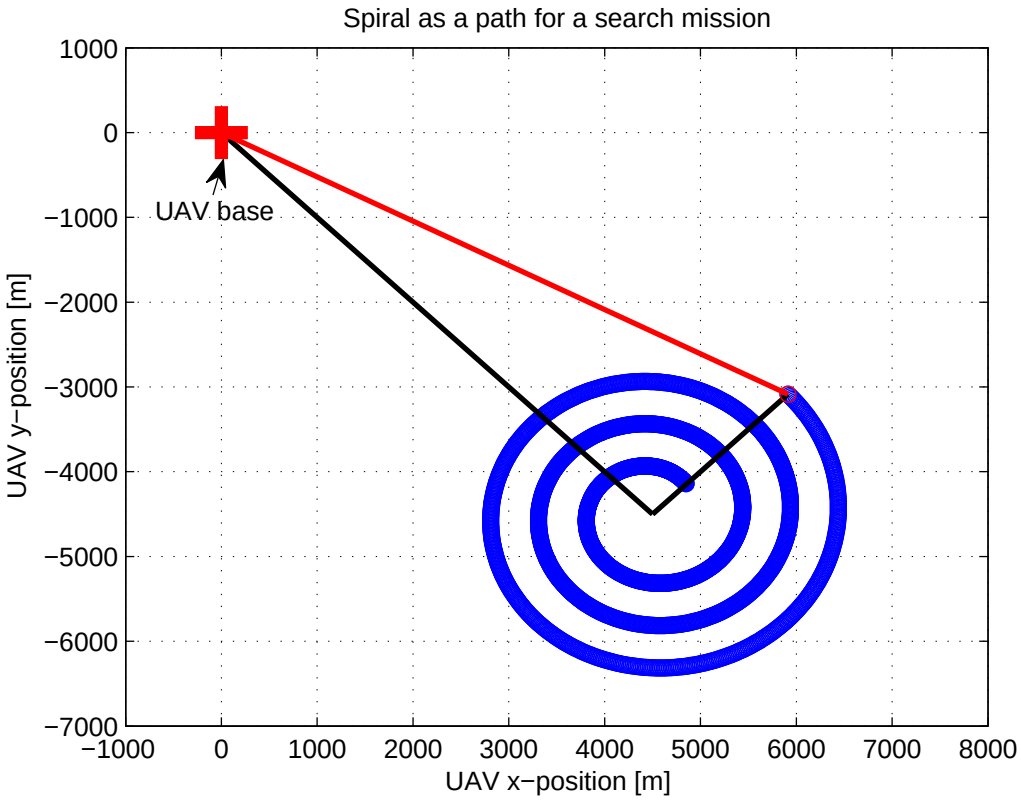

Figure 7.3: Illustration of a good shaped search spiral. a = 250, b = 250

outermost point of the spiral in $\{s\}$ is zero.

The following calculation is done:

$$\mathbf{C}^n := \mathbf{S}_c^n - \mathbf{UAV}_{base}^n \tag{7.6}$$

where $\mathbf{S}_c^n = [x_c^n \; y_c^n]^T$ is the center for the search given by the x- and y coordinates in $\{n\}$, and $\mathbf{UAV}_{base}^n = [x_b^n \; y_b^n]^T$ is the x- and y coordinates to the UAV base station in $\{n\}$. $\mathbf{C}$ is than the vector from the base station to the spiral center.

By defining

$$\mathbf{R}^s =: \begin{bmatrix} x_r^s \\ y_r^s \end{bmatrix} \tag{7.7}$$

where $\mathbf{R}^s$ is the vector from the center of the spiral to the outermost part of the spiral given in $\{s\}$.

By requiring that

$$(\mathbf{C}^n)^T \mathbf{R}^s = 0 \tag{7.8}$$

and that

$$||\mathbf{R}^s||^2 = \rho^2 \tag{7.9}$$

it is possible to find the start point of the spiral. ρ is the radius of the searching area.

By applying the two equations (7.8) and (7.9) it is possible to find the unknown vector $\mathbf{R}^s$.

$$(\mathbf{C}^n)^T \mathbf{R}^s = [x_c^n \; y_c^n] \begin{bmatrix} x_r^s \\ y_r^s \end{bmatrix} = 0 \tag{7.10}$$

This gives the following equation

$$x_c^n \cdot x_r^s + y_c^n \cdot y_r^s = 0 \Rightarrow x_r^s = -\frac{y_c^n}{x_c^n} y_r^s \tag{7.11}$$

$$||\mathbf{R}^s||^2 = (x_r^s)^2 + (y_r^s)^2 = \rho^2 \tag{7.12}$$

By inserting (7.11) into (7.12)

$$\left(-\frac{y_c^n}{x_c^n} y_r^s\right)^2 + (y_r^s)^2 = \rho^2 \tag{7.13}$$

$$\left(\left(-\frac{y_c^n}{x_c^n}\right)^2 + 1\right)(y_r^s)^2 = \rho^2 \tag{7.14}$$

$$(y_r^s)^2 = \frac{\rho^2}{\left(\frac{y_c^n}{x_c^n}\right)^2 + 1} \tag{7.15}$$

We get the coordinates for y_r^s as a function of the spiral center and the radius ρ.

$$y_r^s = \pm\sqrt{\frac{\rho^2}{\left(\frac{y_c^n}{x_c^n}\right)^2 + 1}} \tag{7.16}$$

By inserting (7.16) in (7.12), an expression for the x_r^s components can be found:

$$x_r^s = \pm\sqrt{\rho^2 - \frac{\rho^2}{\left(\frac{y_c^n}{x_c^n}\right)^2 + 1}} = \pm\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1 + \left(\frac{y_c^n}{x_c^n}\right)^2}} \tag{7.17}$$

As one can see from (7.16) and (7.17) there are two different solutions for both equations. But only two of the x- and y- solution are feasible, since the tangent to the spiral at the outermost point should intersect the UAV base. Which of the solutions that should be applied to generate the search spiral is given by the position of the search center relative to the UAV base.

In the implementation the following combination of x- and y- can be used. The different quadrants are shown in Figure 7.4

quadrant	x-component	y-component
1	$-\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$	$\sqrt{\frac{\rho^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$
1	$\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$	$-\sqrt{\frac{\rho^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$
2	$\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$	$\sqrt{\frac{\rho^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$
2	$-\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$	$-\sqrt{\frac{\rho^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$
3	$-\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$	$\sqrt{\frac{\rho^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$
3	$\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$	$-\sqrt{\frac{\rho^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$
4	$-\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$	$-\sqrt{\frac{\rho^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$
4	$\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$	$\sqrt{\frac{\rho^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$

This means that there are two possible solution for the position the outermost part of the spiral to fulfil the equations given in (7.8) and (7.9).

In the implementation, the following solutions have been applied:

quadrant	x-component	y-component
1	$\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$	$-\sqrt{\frac{\rho^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$
2	$-\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$	$-\sqrt{\frac{\rho^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$
3	$-\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$	$\sqrt{\frac{\rho^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$
4	$\sqrt{\frac{\left(\frac{y_c^n}{x_c^n}\right)^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$	$\sqrt{\frac{\rho^2}{1+\left(\frac{y_c^n}{x_c^n}\right)^2}}$

In this case the quadrant is defined when the UAV base is in the origin in the earth fixed coordinate system $\{n\}$. The orientation of the spiral is shown in Figure 7.4 for situations where the search area is in different quadrants realative to the UAV base.

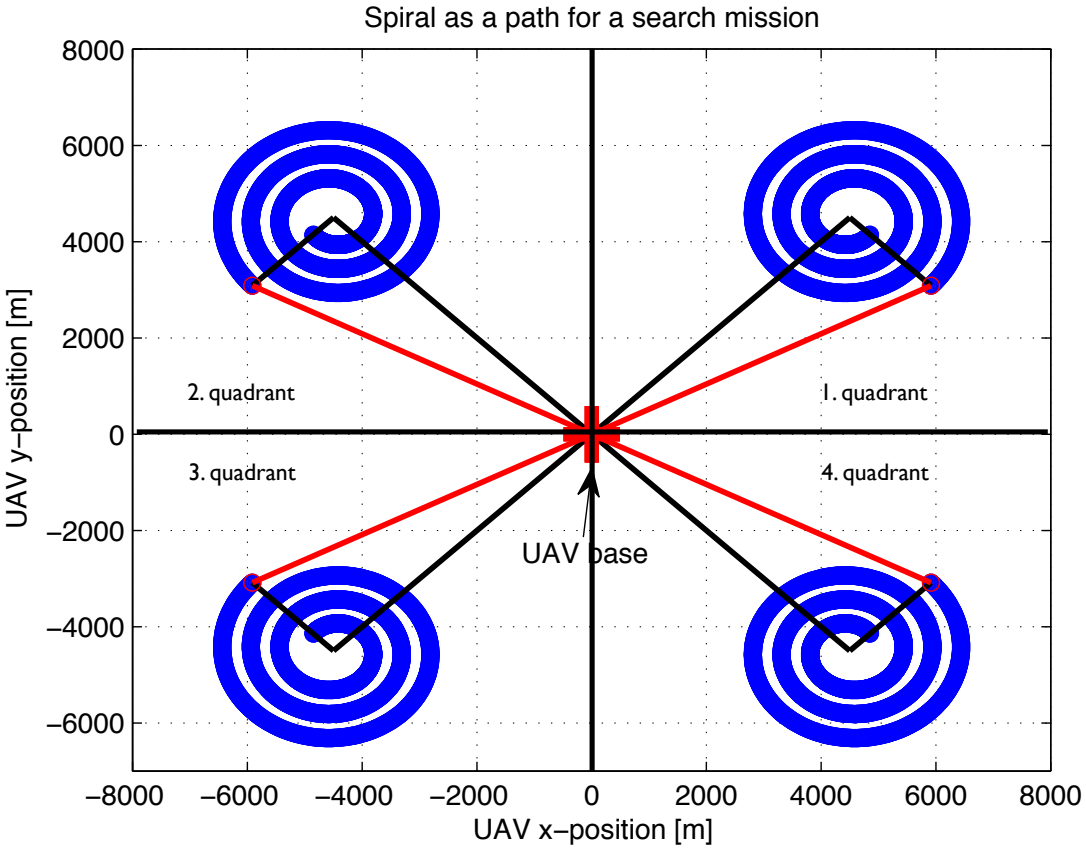

Figure 7.4: Illustration of spirals for all quadrants. a = 250, b = 250.

7.5 The time it takes to search along the spiral

To calculate the time it takes to search the area, when the UAV is flying along the spiral, the average speed of the UAV during the search has to be known as well as the distance which will be travelled. In this section the distance the UAV has to travel during a given search mission will be derived. The motivation is to apply the distance and the average speed of the UAV to calculate the time it uses to complete the search in the area.
As illustrated in Figure 7.5 the relationship between ds, r and θ can be described by the following equation:

$$ds = \sqrt{(rd\theta)^2 + (dr)^2} \tag{7.18}$$

The time it takes to fly the spiral can be found by calculation the following equation

$$t = \frac{s\left(\theta_{max}\right)}{v_{avr}} \tag{7.19}$$

where $s\left(\theta_{max}\right)$ is the distance which is travelled after θ_{max} successive turns and v_{avr} is the average speed the UAV will have along the spiral path.
$s(\theta_{max})$ can be found by integrating (7.18) from 0 to θ_{max}.

The value (θ_{max}) can be found since it is known that $r\left(\theta_{max}\right) = a$. The equation then becomes

$$a = r\left(\theta_{max}\right) = \rho - (a + b + \theta_{max}) \Rightarrow \theta_{max} = \frac{2 \cdot a - \rho}{b} \tag{7.20}$$

When $d\theta$ becomes infinite small the summation one finds when integrating along ds from $\theta = 0$ to $\theta = \theta_{max}$ will be the length of the spiral.

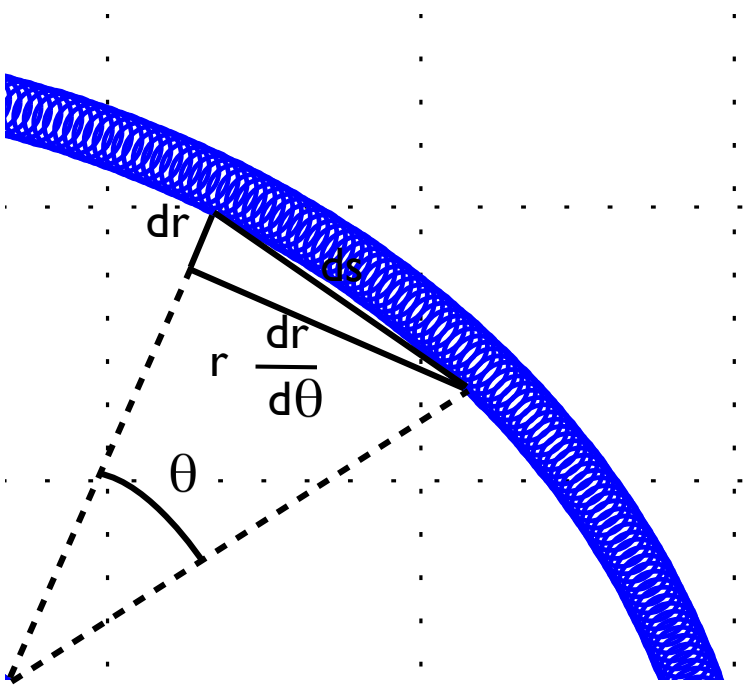

Figure 7.5: Illustration of the method to find the length of the spiral.

This can be written as

$$ds = \sqrt{r^2 + \left(\frac{dr}{d\theta}\right)^2} d\theta \tag{7.21}$$

Inserting the equation for the radius r from (7.3) gives

$$ds = \sqrt{(\rho - (a + b\theta))^2 + b^2} d\theta \tag{7.22}$$

By taking the b- constant outside the square root, the equation can be written as

$$ds = b\sqrt{\left(\frac{(\rho - (a + b\theta))}{b}\right)^2 + 1} d\theta \tag{7.23}$$

By defining

$$x := \frac{(\rho - (a + b\theta))}{b} \tag{7.24}$$

a variable change in (7.23) can be done. By differentiating x with respect to θ the following is obtained:

$$dx = 1 \cdot d\theta \tag{7.25}$$

Inserting this into (7.23) gives

$$ds = b\sqrt{1 + x^2} dx \tag{7.26}$$

To find the length of the spiral, one integrate 7.26:

$$s = \int ds \tag{7.27}$$

Inserting for ds gives:

$$s = b \int \sqrt{1 + x^2} dx \tag{7.28}$$

The integral in (7.28) is solved by using calculus and the details in this work is shown in Appendix B. The solution of the integral is:

$$b \int \sqrt{1 + x^2} = \frac{b}{2} \left\{ x\sqrt{1 + x^2} + \ln \left| x + \sqrt{1 + x^2} \right| \right\} + C \tag{7.29}$$

where C is an arbitrary integration constant depending on the initial values.
Since x was defined to be

$$x = \frac{(\rho - (a + b\theta))}{b} \tag{7.30}$$

it can be inserted in (7.29) to get the equation as a function of θ.

$$s(\theta) = b\int \sqrt{1+\left(\frac{(\rho-(a+b\theta))}{b}\right)^2}\,d\theta \tag{7.31}$$

$$= \frac{b}{2}\left\{\frac{(\rho-(a+b\theta))}{b}\sqrt{1+\left(\frac{(\rho-(a+b\theta))}{b}\right)^2}\right.$$

$$\left.+\ln\left|\frac{(\rho-(a+b\theta))}{b}+\sqrt{1+\left(\frac{(\rho-(a+b\theta))}{b}\right)^2}\right|\right\}+C \tag{7.32}$$

Since the term inside the absolute value brackets always will be positive (since a,b, and θ always is non-negative), the absolute value brackets can be removed, and the equation can be now written as

$$s(\theta) = \frac{b}{2}\left\{\frac{(\rho-(a+b\theta))}{b}\sqrt{1+\left(\frac{(\rho-(a+b\theta))}{b}\right)^2}\right.$$

$$\left.+\ln\left(\frac{(\rho-(a+b\theta))}{b}+\sqrt{1+\left(\frac{(\rho-(a+b\theta))}{b}\right)^2}\right)\right\}+C \tag{7.33}$$

Since $s(0) = 0$, implies that $C = 0$. The expression can now be written as

$$s(\theta) = \frac{b}{2}\left\{A(\theta)\sqrt{1+A(\theta)^2}+\ln\left(A(\theta)+\sqrt{1+A(\theta)^2}\right)\right\} \tag{7.34}$$

where

$$A(\theta) := \frac{(\rho-(a+b\theta))}{b} \tag{7.35}$$

The time it takes to search the area can now be expressed by using (7.19).

The time it takes to do the search, when the UAV has arrived the search area is given by

$$t_{spiral} = \frac{b}{2}\cdot\frac{\left\{A(\theta_{max})\sqrt{1+A(\theta_{max})^2}+\ln\left(A(\theta_{max})+\sqrt{1+A(\theta_{max})^2}\right)\right\}}{v_{avr}} \tag{7.36}$$

where the value for $s(\theta_{max})$ from (7.33) has been inserted into (7.19).

7.5.1 Strategy to decide the camera specifications

The a- and b- constants in the search spiral can be decided if the range of the camera is known.

The a- constant is the distance from the center of the search area to the innermost point of the spiral, and the b constant describes the distance between the successive turnings as was illustrated in Figure 7.1 . To avoid that the same area is covered more than one time by the camera, the spiral path will be generated a function of the camera range.

Since the object can be assumed to be stationary, the path should be generated such that the distance between each successive turning should be twice the range of the camera, such that the same area are not covered twice by the camera.

The distance to spiral as a function of θ is given by:

$$r = \rho - (a + b\theta) \tag{7.37}$$

The distance between each turning is given by

$$\Delta r_i = (\rho - (a + b\theta_i)) - (\rho - (a + b\theta_{i+1})) = b(\theta_i - \theta_{i+1}) \tag{7.38}$$

where i is the number of turnings

Since $\Delta r_i = 2 \cdot r_{cam}$, and since $b(\theta_i - \theta_{i+1}) = 2\pi$, (7.38) can be written as

$$2 \cdot r_{cam} = b \cdot 2\pi \Rightarrow b = \frac{r_{cam}}{\pi} \tag{7.39}$$

$$a = r_{cam} \tag{7.40}$$

Where r_{cam} is the range of the camera (this is a requirement). The range of the camera to each side of the UAV will be given by the sector the camera covers, and the elevation of the UAV.

$$r_{cam} = h \tan \phi \tag{7.41}$$

Then the a- and b- constants can be written as:

$$a = h \tan \phi \tag{7.42}$$

$$b = \frac{h \tan \phi}{\pi} \tag{7.43}$$

where ϕ is the sector the camera on the UAV can detect, relative to the focus line, and h is the elevation of the UAV. It is assumed that the camera always will be oriented such that the focus line is aligned to the z- axis in $\{n\}$, and such that the search range ϕ is symmetric around the z-axis. This is the same assumption as was presented in the algorithm developed in Chapter 6.

In this search and rescue mission the elevation of the UAV and the angle of the camera (ϕ) will be the decision variables.

The the expressions for A in (7.35) can be written as a function of θ and the UAVs elevation (h), by inserting for a and b in (7.42) and (7.43):

$$A(\theta, h) = \frac{\left(\rho - \left(h \tan \phi + \frac{h \tan \phi}{2}\theta\right)\right)}{\frac{h \tan \phi}{2}} \tag{7.44}$$

Since ϕ was assumed to be constant, the only variables that can affect the time it takes to fly around the spiral-path t_{spiral} is the elevation (h) and the speed of the UAV. This can be shown by inserting (7.44) into (7.36). One can then see that t_{spiral} increases with an increase in v_{avr} and with an increase in the elevation h.

7.5.2 Searching for moving elements

If the object one is searching for is moving in the $\{n\}$ frame, this information needs to be taken into account by the spiral path planner. In the implementation so far, there have been no overlap in the area which have been searched by the UAV. If the object which one is searching is moving with a given speed in some arbitrary direction, the result can be that the object will never be found.

If it can be assumed that the object is moving with a speed which is bounded below by $||v(t)|| \leq v_{max}$

where $v(t)$ is the velocity vector of the moving object, and v_{max} is the maximum speed the object can move with in an arbitrary planar motion in $\{n\}$.

In the worst case the velocity vector of the object will intersect the spiral center. The maximum distance the object can move, in the period of one successive turn, is then given by

$$s_{object} = v_{max} \cdot t_{turn,max} \tag{7.45}$$

where $t_{turn,max}$ the maximum time it takes to do one rotation, and that will be the first turn (from $\theta = 0$ to $\theta = 2\pi$), since the length of the spiral is greatest at the first turn which is furthest away from the center of the spiral.

The a- and b- constants in (7.42) and (7.43) can be modified to take into account that objects are moving in the area. This has been is done with the following definition:

$$\hat{a} := a - \beta = h \tan \phi - \beta \tag{7.46}$$

$$\hat{b} := b - \beta = \frac{h \tan \phi}{\pi} - \beta \tag{7.47}$$

where β is an design variable that depends on the maximum speed of the object one is searching for and the range of the search area. The parameter tells how much overlap there should in the area which the camera covers in each rotation.

The distance between the successive turnings can be written as:

$$\Delta r := \rho - \hat{a} - \hat{b}\theta_{turn} \tag{7.48}$$

where $\hat{a} = a - \beta = r_{cam} - \beta$ and $\hat{b} = \frac{r_{cam}}{2} - \beta$.

Then the equation (7.36) can be written as

$$t_{spiral} = \frac{v_{max}}{v_{avr}} \frac{1}{2} \{G\sqrt{1+G^2} + \ln\left(G + \sqrt{1+G^2}\right)\} \tag{7.49}$$

where

$$G = \frac{\rho - \hat{a} - \hat{b}\theta}{\hat{b}} \tag{7.50}$$

7.5.3 Methods which can be used to let the UAV arrive the start point on the spiral

The UAV has to arrive at the starting point the spiral path, and follow the spiral until the end point of the spiral is reached. When the search area has been searched, the UAV will will return back to it's base. The search mission can be divided in three parts:

1. The path planner which takes the UAV from the base to the starting point in the search area.

2. The path planner which makes the path the UAV should follow in the search area.

3. The path planner which takes the UAV from the search area and back to it's base station.

The overall guidance system will change between the different path planners. In this chapter it is assumed that the developed path using an Achimedes Spiral will be applied as the spiral in the search area. The path planner which takes the UAV from the base to the starting point in the search area and the path planner which takes the UAV from the search area and back to it's base station, has to be desinged.

In the following two different methods will be discussed.

7.5.3.1 Implementation with MILP

The path from the base station to the search area can be generated by solving a MILP problem, as done earlier in this thesis. The objective of the optimization problem will be to minimize the time to get to the search area. The path from the search area and back to the UAV base do need not need to be optimized with respect to time consumption. It is not urgent to get back to the base, and a more natural approach can be to fly with a cruise speed to minimize the fuel consumption. The speed can either be defined to be a constant speed, or be found by solving the MILP optimization problem which also takes into account the fuel consumption.

This approach, with solving the MILP optimization problem will be a good approach if there are obstacles in the environment between the base and the search area. If there are no obstacles in the area, the shortest path between the base and the start point on the spiral will obvious be the straight line between these two points. To arrive the spiral as fast as possible the UAV should fly as fast as possible. It should not be necessary to solve an optimization problem to find this path. Therefore a guidance system which is implemented with a Line of Sight approach is suggested.

Remark: When finding the path from the UAV base station to the start the search area, one could solve the MILP optimization. If the objective is to minimize the time consumption, the optimizer will find out that it is optimal to let the UAV fly as fast as possible. In Figure (7.6) the optimal path with respect to the constraints and the cost function is plotted. In Figure (7.7) the camera range in the search area is plotted. As one can see, the optimal path is not a straight line between the UAV base and the starting point in the search area, as would be the most intuitive solution. The reason for this is because the non-linear constraints on the UAV velocity and the constraints on the input (given in equation 3.52 and 3.53) are approximated by the linear constraints given in (3.54) and (3.55). Since the objective is to arrive the search area as fast as possible, the optimizer will let the commanded force and speed be as close as possible to the restrictions. This can result in a situation where the optimal path in the beginning is not moving directly towards the arrival point on the spiral, such that the the speed and the force will be

in a corner between two linear constraints. At this point the values of the speed and the commanded input can be at it's greatest with respect to the constraints. If this corner is not pointing in the same direction as the straight line between the base and arrival point, then the path will be generated by following one corner solution in linear constraint approximation, and change to other corners during the path, such that the UAV arrives the destination. The main reason for this situations is that the non-linear constraints for speed and inputs on the UAV, are approximated with linear constraints. Another problem is that one or two of the constraintsare active most of the time. In the approximation the non-linear restrictions (3.45) and (3.51) was approximated with 10 linear constraints. The approximation can be done better by increasing the number linear constraints, which will give a more correct model and give a more sensible path. However, if the number of linear constraints are increased, this will introduce more binary variables to the problem which again will give a more complex problem, resulting in longer computation time. In Figure (7.8) a path which is generated when the non-linear constraints are approximated with 20 linear constraints are shown. As one can see, the path is much straighter in this case. If the cost function in the optimization problem is designed in a way, such that the restrictions on force and speed is never activated, the path will be a straight line. This can be shown by using a cost function which for examples minimizes the fuel consumption, since in this case the input and the speed will never push the restrictions. This is shown in Figure (7.9) .

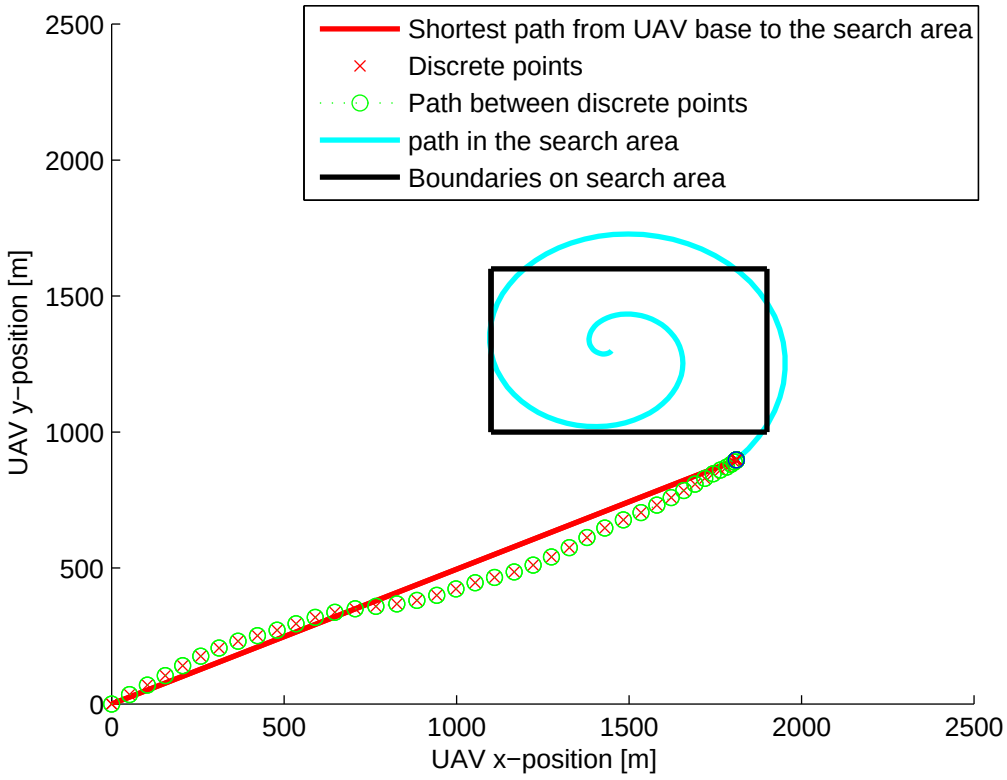

Figure 7.6: MILP is used to find the optimal path from the base and to the start point in the search area. In the linear speed and force approximations $M = 10$.

The UAV also needs to return from the search area and get back to it's base station. In the design of the spiral the direction of the innermost point of the

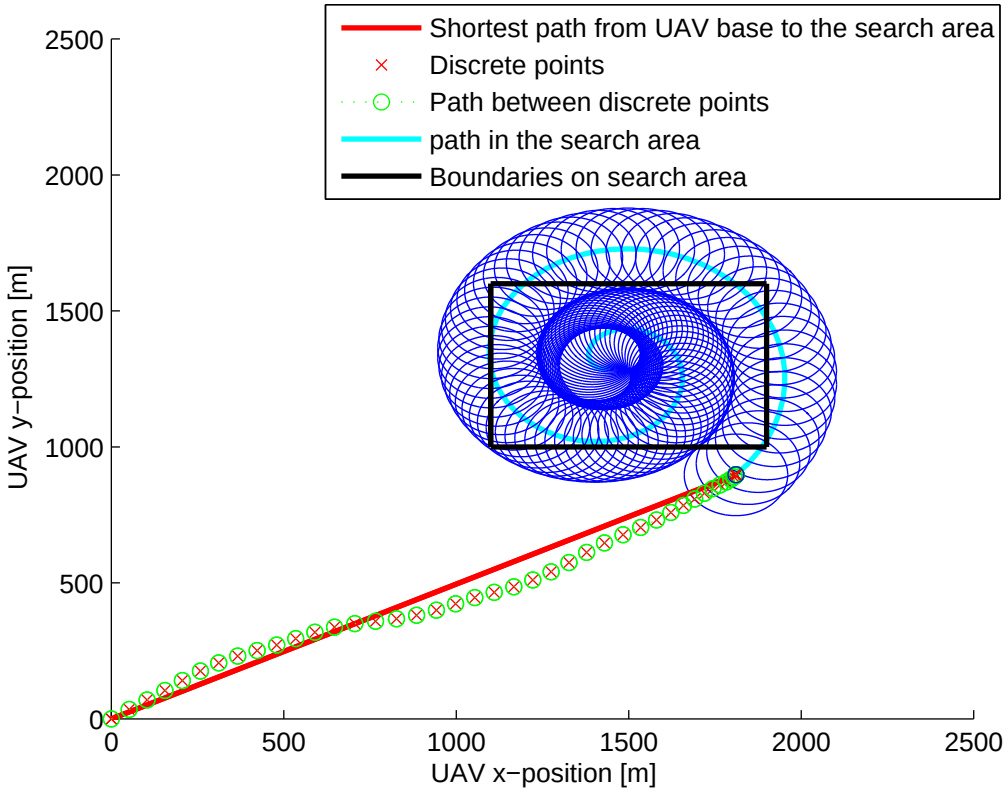

Figure 7.7: MILP is used to find the optimal path from the base and to the start point in the search area. In the linear speed and force approximations $M = 10$. The blue circles shows the camera range. As one can see the defined search area is covered by the camera system.

spiral have not been given any restriction about which direction is should point. It would be reasonable to know the direction of the innermost point of the spiral, since another guidance system will take the UAV from this position and back to the UAV base. The following derivation gives the position of the UAV when it is at the innermost point of the spiral, where $\theta = \theta_{max}$:

$$x = (\rho - (a + b\theta_{max})) \cdot \cos(\theta_{max}) \tag{7.51}$$

$$y = (\rho - (a + b\theta_{max})) \cdot \sin(\theta_{max}) \tag{7.52}$$

$$\frac{dx}{d\theta_{max}} = -(\rho - (a + b\theta_{max})) \cdot \sin(\theta_{max}) - b\cos(\theta_{max}) \tag{7.53}$$

$$\frac{dy}{d\theta_{max}} = (\rho - (a + b\theta_{max})) \cdot \cos(\theta_{max}) - b\sin(\theta_{max}) \tag{7.54}$$

To find the velocity vector the UAV will have at this point, the following calculation has to be done:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \frac{1}{\sqrt{\left(\frac{dx}{d\theta_{max}}\right)^2 + \left(\frac{dy}{d\theta_{max}}\right)^2}} \begin{bmatrix} \frac{dx}{d\theta_{max}} \\ \frac{dy}{d\theta_{max}} \end{bmatrix} \cdot V(\theta_{max}) \tag{7.55}$$

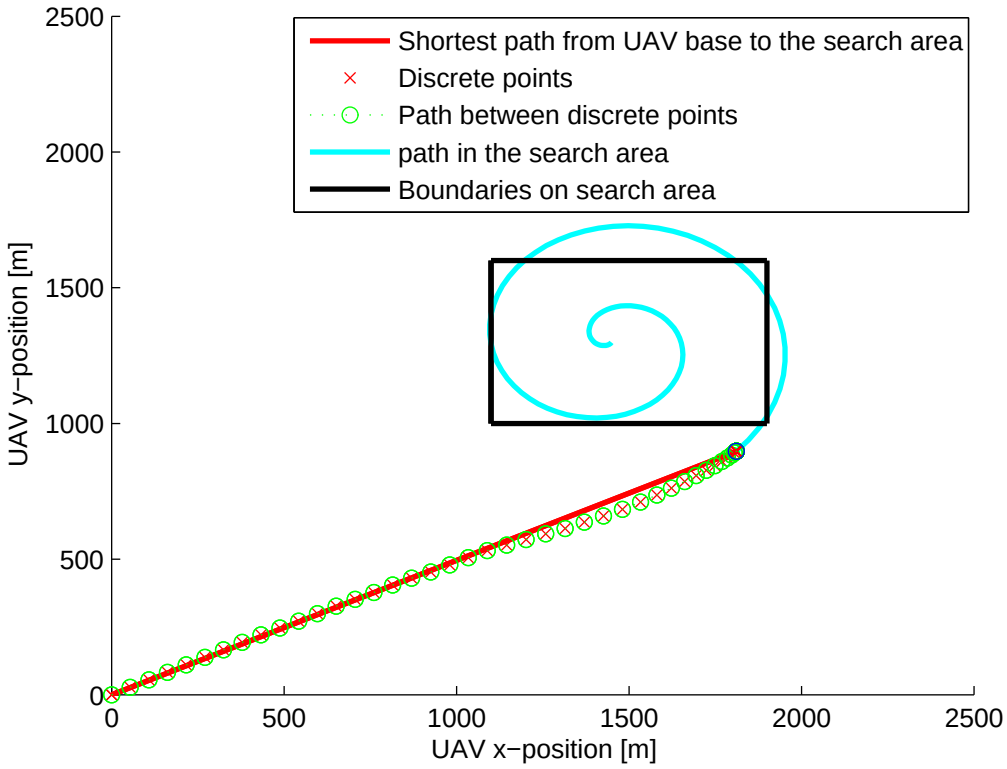

Figure 7.8: In this simulation the non-linear restriction which describes the force and speed constraints are each approximated by $M = 20$.

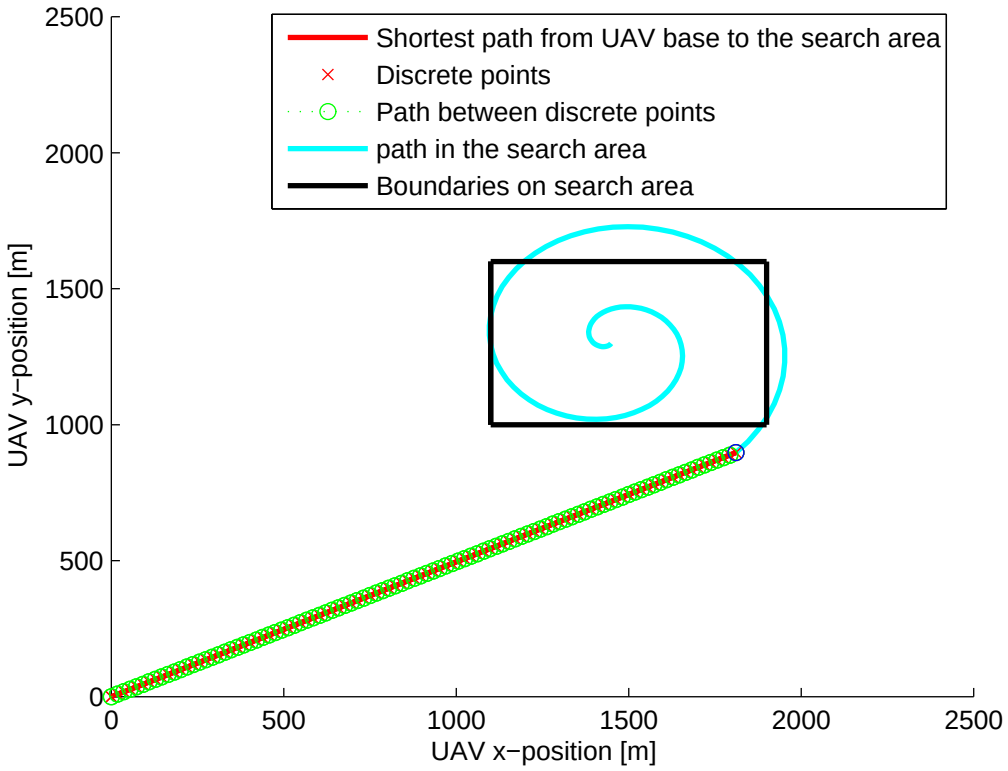

Figure 7.9: MILP is used to find the optimal path from the base and to the start point in the search area. In this case minimum fuel is the objective of the cost function, and therefore the non-linear approximations are never enforces which results in a straight line path.

v_x and v_y should be implemented as the initial velocity by the guidance system that takes the UAV back to base from the search area.

7.5.3.2 Implementation with velocity-reference model and line-of-sight methods

If there are no obstacles in the area from the UAV base and the starting point in the search area, other methods should be applied to take the UAV from the base and to the start of the search area.

When the UAV stays in it's base station and are sent on the search mission, the operator will give the UAV order to fly at the maximum speed.

As described in Fossen [2011a] the guidance system should low-pass filter the commanded speed such that the reference speed, which goes to the regulator, is feasible compared to the UAV dynamics. The low-pass filter should at least be of order two to obtain smooth reference signals for the desired velocity v_d and the acceleration $\dot{v}_d$. Based on Fossen [2011a], a desired velocity for the UAV can be given as

$$\ddot{v}_d + 2\Delta\Omega\dot{v}_d + \Omega^2 v_d = \Omega^2 r^b \tag{7.56}$$

where v_d is the desired velocity, $\ddot{v}_d$ is interpreted as the desired "jerk", to get the low-pass filter as a second order filter. Δ is the relative damping ratio and Ω is the natural frequencies of the UAV.

In this thesis, the implementation has not been done with the damping included, because of little information about the damping on the actual UAV which is considered in the UAV-project.

Therefore, in the following, the damping term Δ will be set to zero.

The equation can now be written as

$$\ddot{v}_d + \Omega^2 v_d = \Omega^2 r^b \tag{7.57}$$

Taking the Laplace transform of this equation gives

$$s^2 v_d + \Omega^2 v_d = \Omega^2 r^b \tag{7.58}$$

This can be written as

$$\left(s^2 + \Omega\right) v_d = \Omega^2 r^b \tag{7.59}$$

Which gives

$$v_d = \frac{\Omega^2}{s^2 + \Omega^2} r^b \tag{7.60}$$

As one can see, when time goes to infinity, $v_d = r^b$. The response of v_d depends on Ω, which again depends on the UAV dynamics.

When there are no obstacles in the area between the UAV base and the starting point of the spiral, the shortest path between the UAV base and the spiral will be the straight line between the between the two points. As described in Fossen [2011a], LOS Steering Laws can be applied. Two methods are described:

1. *Enclosure-based steering*
2. *Lookahead-based steering*

In this thesis the lookahead-based steering scheme will be applied because it is less computationally intensive than the enclosure-based approach.

In Lookahead-Based Steering, the course angle assignment is separated into two parts:

$$\chi_d(e) = \chi_p + \chi_r(e) \tag{7.61}$$

where

$$\chi_p = \alpha \tag{7.62}$$

is the path-tangential angle, while

$$\chi_r(e) := \arctan\left(\frac{-e}{\Delta}\right) \tag{7.63}$$

is a velocity-path relative angle, which ensures that the velocity is directed toward a point on the path that is located a lookahead distance $\Delta(t) > 0$ ahead of the direct projection of $\mathbf{p}^n(t)$ on the path.

e(t) is the cross-track error (normal to path) which is described in Fossen [2011a], and is calculated by solving:

$$e(t) = -\left[x(t) - x_1\right]\sin(\alpha_1) + \left[y(t) - y_1\right]\cos(\alpha) \tag{7.64}$$

α_k is the rotation the path-fixed reference frame has been rotated relative to the x-axis, such that it is aligned with the path.

There are only necessary with two waypoints to make a path from the UAV base and to the starting point in the search area. Therefore, in (7.64), x_k and y_k will be the x and y values for the UAV base station. Since the origin is placed here, these values will be zero.

This means that (7.64) can be written as

$$e(t) = -x(t)\sin(\alpha) + y(t)\cos(\alpha) \tag{7.65}$$

The angle α_k can be found when the destination point in the search area is known. If this is denoted x_2 and y_2, this can be calculated as

$$\tan(\alpha_k) = \frac{y_2 - y_1}{x_2 - x_1} \Rightarrow \alpha = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \tag{7.66}$$

The value $\chi_r(e)$ will be the reference signal which goes into the regulator.

7.5.4 Discussion

The speed the UAV has along the spiral path should be to close as maximum to possible, but the curvature of the path will force an upper limit on the speed. According to newtons second law for a object which is rotating around a given center, the centripetal force, which is pointing towards the center, given by

$$F_c = mV^2\kappa = m\frac{v^2}{r} \tag{7.67}$$

where F_c is the centripetal force, V is the tangential speed, and r is the distance from the center to the point in the spiral where the UAV currently is located. The equation in (7.67) has to fulfill (3.36), since this equation contains the limitations

in the maximum centripetal-force which the UAV can have due to modeled UAV dynamics and aerodynamics.

To ensure that the UAV can follow the spiral path, the speed can be described as a maximum speed as a function of the radius. The follow restriction needs to be included:

$$F_c \leq f_c \tag{7.68}$$

$$m\frac{V^2(r)}{r} \leq V_{max}^2 C_L \tan(\phi_{max}) \Rightarrow V(r) \leq V_{max}\sqrt{\frac{m}{rC_L \tan(\phi_{max})}} \tag{7.69}$$

As long as the restriction (7.69) is fulfilled, it will be possible for the UAV to follow the path.

Chapter 8

Further Improvement of the Optimal Path Planning Design

- The path planning with collision avoidance, is in this thesis, optimized from the initial state to the the final state. This approach with a fixed time horizon, gives a large optimization problem, since every time step from the start to the goal has to be included and calculated in the optimization problem. The complexity of the optimization problem increases with the distance from the initial state to the final state, number of way points which should be visited and the number of obstacles and no-fly areas. The simulations in the previous chapters where done in a fixed horizon strategy, and provides good results in a limited area. It could be satisfactory for a car operating in a relatively small area. For an UAV however, it has to be assumed that the area of operation must be reasonably large when compared to the simulations done in this thesis. This implies that the time horizon has to be large in order to find a feasible solution to the optimization problem. At the same time, the optimization has to make a sufficient amount of time steps such that problems with corner cutting of obstacles will not cause too much problems. The solution to this problem is discussed and pointed out in the in the following subsection. The author suggests that a receding horizon path planner is developed for the UAV project.

- The path planning design developed in this thesis needs to be updated with the best estimates of the UAV coefficients, to make sure that the path planner generates a feasible and optimal path for the considered UAV. It also needs information about which forces that is the restraining one, between the engine and maximum centripetal acceleration due to roll angle. This should be considered by the path planner to make a feasible path, as is discussed closer in Chapter 3. Also the rudder could further be included in the model.

- The author suggests that the actuator dynamics is implemented in the MILP. As one can see from the figures in this thesis that considers the forces, one will notice that they can do relatively large changes in magnitude between time steps. This is because no restriction related to this has been implemented in this thesis. It is very important to include a restriction on the actuator to model it's dynamics, because the generated path will not be feasible when considering a real UAV.

- Damping due to air friction should be further included in the path planner. In Chapter 3 two methods for incorporating the damping term was discussed.
- Improve the path planner to also consider a three dimensional guidance system.
- Other modelling languages as AMPL and the CPLEX solver could be applied to describe the optimization problem and solve it respectively, to find out if they work better than the combination og the YALMIP language and the Gurobi solver.
- The methods described in this thesis for search missions could be further improved. The path planner which is generated by applying an Archimedes spiral could be improved additionally.

8.1 Receding Horizon Control

This section is based on information from [Bellingham et al., 2002]

To reduce the computation time required by MILP for large trajectories, a receding horizon framework could be used. Receding horizon control (also called Model Predictive Control) designed an input trajectory that optimizes the plant's output over a period of time, called the planning horizon. The input trajectory is implemented over a shorter execution horizon, and the optimization is performed again starting from the state that is reached.

The receding horizon strategy incorporates feedback to account for disturbances and plant modeling errors. To further develop the fixed horizon MILP to a receding horizon one needs to have a *cost to goal* path, which indicates in what direction the target is. The MILP will solve small optimization problems along this *cost to goal path*. As described in Chapter 2.2 , the MILP method is NP-hard which indicate that the computation required to solve the problem grows non-linearly with the number of binary variables. To keep the number of binary variables low, it is preferred to have as short horizon as possible. Therefore, to reduce the computation time required by MILP, the problem can be formulated as many small problems along the *cost to goal* path, which is the receding horizon framework.

The major difference between the fixed arrival time and the receding horizon is the approach in the optimization criteria. In fixed arrival time the final states and the arrival time is specified in the constraints. When using receding horizon the solution is locally optimal on every segment. The total receding horizon path will probably require more fuel than a vehicle would consume if the path was calculated over the entire time range [Schouwenaars et al., 2001].

Chapter 9

Summary and Conclusion

In this thesis the equations of motion for an UAV described in a geographic frame was derived from the equations of motion in the body frame and discretized. Restrictions on the forces has been included, due to limitation in maximum speed and maximum roll angle of the UAV. The rudder was not considered in formulating constraints in yaw rate, and it was assumed that yaw rate is a result of roll angle.

The equation for restrictions in forces was derived, and maximum forces that affects the UAV was guesses concerning maximum speed, wing area and aerodynamic coefficients. The calculated value was found to be unreasonably high, and another guessed values where was used instead. When the parameters of the Odin Recce D6 delta-wing aircraft has been estimated, it can readily be changed in this model to improve the path planner developed in this work. Drag forces where not implemented in the simulations due to unknown aerodynamic coefficients, but this can be implemented without problems in the model once a more sensible behaviour of the UAV path planner is assured. Restrictions on initial state, final state, waypoints to visit, UAV dynamics, inputs to the UAV, obstacles, and no-fly area included in the Mixed Integer Linear Program. It was demonstrated that problems with corner cutting could be avoided in a systematic way, by introducing a safety margin around the obstacles.

Two different ways of path planning where designed. The first one was considering that the obstacles where known *a priori*, and the optimization was done once. The second one was designed presuming that only waypoints and final position was known by the optimization problem when the optimization takes place. Obstacles where implemented in the optimization problem when UAV was in a certain distance from the obstacle, to model that obstacles where detected by a radar with a limited range. Both designs where done using the YALMIP programming language, and it was concluded that the design with radar detection was more realistic, and a simple change in the code, will make the design suitable for slowly varying dynamic obstacles as well. The path planner with radar detection was proven to have faster computation time than the design where the obstacles where known *a priori*. The improved computation time is an important result, since there will be limited computing power available for the Odin Racce D6 delta-wing aircraft.

The path planner using MILP was further improved to be capable to do path planning for an arbitrary number of UAVs and it was shown how UAVs could take part in different search missions. An algorithm for automatic generation of waypoints to ensure efficient search of a defined area was developed. It was argued that the time it takes to compute the optimal solution also has to be taken

into consideration, since the UAV can not start on the search mission before the computation is computed.

Another path planner for efficient search was developed by applying an Archimedes spiral. The spiral was generated to change its position, radius, distance between the successive turnings depending on the range of the UAV camera. The spiral was also designed to account for moving obstacles.

Bibliography

Hussain. Al-Helal and Jonathan Sprinkle. Uav search: Maximizing target acquisition. 2010.

John. Bellingham, Arthur. Richards, and Jonathan P. How. Receding horizon control of autonomous aerial vehicles. 2002.

Dimitris. Bertsimas and John N. Tsitsiklis. Introduction to linear optimization. *Athena Scientific*, 1997.

A. Bicchi and L. Pallottimo. On optimal cooperative conflict resolution for air traffic management systems.

Chi-Tsong Chen. *Linear System Theori and Design.* Oøxford Uøniversity, 1999.

Kristoffer. Dønnestad. Simulation, control and visualization of uas. 2011.

Henry. Edwards and David. Penny. *Calculus 6e.* Prentice Hall, 2002.

Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons, 2011a.

Thor I. Fossen. Mathematical models for control of aircraft and satellites. 2011b.

C. Goerzen, Z. Kong, and B. Mettler. A survey of motion planning algorithms from the perspective of autonomous uav guidance. 2009.

Esten Ingar. Grøtli and Tor Arne. Johansen. Task assignment for cooperating uavs under radio propagation path loss constraints. 2011a.

I. El Grøtli and A.T Johansen. Path Planning for UAVs Under Communication Constraints Using SPLAT! and MILP. 2011b.

Inc. Gurobi. Product overview, October 2011. URL `www.gurobi.no`.

T. Hutchings, S. Jeffryes, and S.J. Farmer. Architecting uav sense & avoid systems. *Thales UK*, 2007.

Gökhan. Inalhan, Dusan. Stipanovic, and Claire Tomlin. Decentralized optimization, with application to multiole aircraft coordination. 2002.

Hassan K Khalil. *Nonlinear Systems.* Prentice Hall, 2002.

J. Löfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. URL `http://users.isy.liu.se/johanl/yalmip`.

G. Earl Matthew and D'Andrea. Raffaello. Iterative MILP Methods forVehicle-Control Problems. *IEEE*, 2005.

Gurobi mex. A MATLAB interface for gurobi, February 2012. URL `http://www.convexoptimization.com/wikimization/index.php/Gurobi_,Mex:_,A_MATLAB_,interface_,for_,Gurobi`.

Christian. Reinl and Oskar Von. Stryk. Optimal control of multi-vehicle-systems under communication contraints using mixed-integer linear programming. 2007.

Arthur. Richards and Jonatan. p. How. Aircraft Trajectory Planning With Collition Avoidance Using Mixed Integer Linear Programming. *American Control Conference*, 3(E5):1936–1941, 2002.

Tom. Schouwenaars, Bart De Moor, Eric. Feron, and Jonathan. How. Mixed integer programming for multiple-vehicle path planning. *ECC2001 Conference*, 2001.

Brian L. Stevens and Frank L. Lewis. *Aircraft control and simulation, second edition*. John Wiley & Sons, 2003.

Bjørnar Vik. *Internal Satellite and Inertial Navigation Systems*. Department of Engineering Cybernetics, NTNU, 1999.

Appendix A

Supported platforms for Gurobi Optimizer 4.5. From [Gurobi, 2011]

Platform	Operating System	Compiler
Windows 32-bit (win32)	Windows XP®, Windows Vista®, Windows 7,Windows Server 2008 R2®	Visual Studio 2008®,Visual Studio 2010®
Windows 64-bit (win64)	Windows XP®, Windows Vista®, Windows 7,Windows Server 2008 R2®	Visual Studio 2008®,Visual Studio 2010®
Linux 32-bit (linux32)	Red Hat® Enterprise Linux® 5.3, 5.4, 5.5, 5.6	gcc 4.1
	SUSE® Enterprise Linux 11	gcc 4.3, 4.4
	Ubuntu® 8.04, 10.04, 10.10	gcc 4.2, 4.3, 4.4
Linux 64-bit (linux62)	Red Hat® Enterprise Linux® 5.3, 5.4, 5.5, 5.6	gcc 4.1
	SUSE® Enterprise Linux 11	gcc 4.3, 4.4
	Ubuntu® 8.04, 10.04, 10.10	gcc 4.2, 4.3, 4.4
Mac OS 64-bit (mac64)	Mac OS X® 10.6 (Snow Leopard®)	Xcode® 3.2, 4.0
AIX® 64-bit (power64)	AIX® 5.3, 6.1, 7.1	XL C 9

Table A.1: Supported plattforms for Gurobi Optimizer 4.5

Appendix B

Derivation of integral

$$s = b \int \sqrt{1 + x^2} dx \tag{B.1}$$

The time it takes to search the defined area is given by the distance which is travelled and the average speed of the UAV.

$$\sqrt{1 + x^2} = \sec(\alpha) \tag{B.2}$$

$$\tag{B.3}$$

$$x = \tan(\alpha) \tag{B.4}$$

$$\tag{B.5}$$

$$dx = \sec^2(\alpha)\, d\alpha \tag{B.6}$$

This gives

$$s = b \int \sec^3(\alpha)\, d\alpha \tag{B.7}$$

Solving this with integration by parts gives

$$b \int \underbrace{\sec(\alpha)}_{u} \underbrace{\sec^2(\alpha)}_{dv} d\alpha = b \left\{ \underbrace{\sec(\alpha)}_{u} \underbrace{\tan(\alpha)}_{v} - \int \underbrace{\sec(\alpha)\tan(\alpha)}_{du} \underbrace{\tan(\alpha)}_{v} d\alpha \right\} \tag{B.8}$$

$$b \int \sec^3(\alpha)\, d\alpha = b \left\{ \tan(\alpha)\sec(\alpha) - \int \left[\sec(\alpha) \left(\sec^2(\alpha) - 1 \right) \right] d\alpha \right\} \tag{B.9}$$

$$= b \left\{ \tan(\alpha)\sec(\alpha) + \int \sec(\alpha)\, d\alpha - \int \sec^3(\alpha)\, d\alpha \right\} \tag{B.10}$$

$$2b \int \sec^3(\alpha)\, d\alpha = b \left\{ \sec(\alpha)\tan(\alpha) + \int \sec(\alpha) \left(\frac{\tan(\alpha) + \sec(\alpha)}{\tan(\alpha) + \sec(\alpha)} \right) d\alpha \right\} \tag{B.11}$$

By defining

$$z = \tan(\alpha) + \sec(\alpha) \tag{B.12}$$

We find that

$$dz = \left(\sec^2(\alpha) + \tan(\alpha)\sec(\alpha)\right) d\alpha \tag{B.13}$$

$$2b \int \sec^3(\alpha)\, d\alpha = \sec(\alpha)\tan(\alpha) + \int \frac{dz}{z} \tag{B.14}$$

$$b \int \sec^3(\alpha)\, d\alpha = \frac{b}{2}\left\{\sec(\alpha)\tan(\alpha) + \ln|z|\right\} + C \tag{B.15}$$

Inserting for $\tan(\alpha)$, $\sec(\alpha)$ and z gives

$$b \int \sqrt{1+x^2} = \frac{b}{2}\left\{x\sqrt{1+x^2} + \ln\left|x + \sqrt{1+x^2}\right|\right\} + C \tag{B.16}$$

Appendix C

Matlab - MILP fixed horizon, *a priori* obstacle information

C.1: UAV specification and initial values

```
%% CLEAR ALL %%

yalmip('clear');
clc;
clear all;
close all;

%% PARAMETERS %%
T            = 300;          %Horizon
Td           = 3;            %Discretixation time-step
N            = T/Td;         %Number of steps
M            = 10;            %Number of constraints
M_big        = 1000;         % Big number applied in the constraints

%% UAV assumed spesification
v_max        = 20;           %Max speed
v_min        = 5;           %Min speed
f_max        = 1.5;       %Max control force
m            = 2.8;         % UAV mass


%% Initial values
x_uas_0      = 0;            %Initial position UAV x
y_uas_0      = 0;            %Initial position UAV y
v_x_uas_0    = 0;            %Initial velocity UAV x
v_y_uas_0    = 0;            %Initial velocity UAV y
```

C.2: Adding waypoints and obstacles

```
%% Adding waypoints (x-and y position)
wp_set = [];
wp_set(:,1) = [100;10];
wp_set(:,2) = [50,0];
%wp_set(:,3) = [400,100];
%wp_set(:,2) = [200,300];
%wp_set(:,2) = [100,200];
```

```
8
9  %% Obstacles
10 % Adding small obstacles
11 O_set = [];
12 %O_set(:,1) = [0;-40];
13 %O_set(:,2) = [0;40];
14 %O_set(:,3) = [40;40];
15 safety_margin = 20;
16
17 % Adding bigg obstacles
18 O_setBig = [];
19 %O_setBig(:,1) = [40;0];
20 safety_margin_big = 60;
21
22 % approximation of obstacle to be a square
23 numberofSidesObstacle = 4;
```

C.3: YALMIP decision and binary variables

```
1  %% Defining YALMIP decision variables and YALMIP binary variables %%
2
3  % decision variables
4  f_i            = sdpvar(2,N);  %Control force vector
5  s_i            = sdpvar(4,N);  %State vector
6  speed          = sdpvar(1,N);  % Used to spesify
7
8  w_slack = sdpvar(2,N); %Variable which is used to avoid absoulute value
9                         %in the cost function.
10
11 %binary variables
12 b_uas_i        = binvar(length(wp_set(1,:)),N);  %UAS reaches a waypoint 1/0
13 b_m            = binvar(M,N)
14
15 % Binary variables used in obstacle avoidance, small obstacles
16 if(length(O_set)>0)
17 o_uas_i        = binvar(length(O_set(1,:)),numberofSidesObstacle,N);
18 end
19 % Binary variables used in obstacle avoidance, big obstacles
20 if(length(O_setBig)>0)
21 o_uas_Big       = binvar(length(O_setBig(1,:)),numberofSidesObstacle,N);
22 end
```

C.4: Discretization of UAV dynamics

```
1  %% UAV model in geographic frame, from continuous to dicrete %%
2
3  A = [0 0 1 0;
4       0 0 0 1;
5       0 0 0 0;
6       0 0 0 0];
7
8   B = [0 0;
9        0 0;
10       1/m 0;
```

```
        0 1/m];

Ad              = [];
Bd              = [];

%Discretisation
[Ad,Bd] = c2d(A,B,Td);
```

C.5:

```
%% Adding constraints

F = [];
% Adding initial constraints (and constraints for state N)
F = F + [s_i(1,1) == x_uas_0,...
         s_i(2,1) == y_uas_0,...
         s_i(3,1) == v_x_uas_0,...
         s_i(4,1) == v_y_uas_0];%,...
         %s_i(1,N) == x_uas_0,...
         %s_i(2,N) == y_uas_0,...
         %s_i(3,N) == -v_x_uas_0,...
         %s_i(4,N) == -v_y_uas_0];%,...
         %f_i(1,1) == 0,...
         %f_i(2,1) == 0];

 % Adding model dynamics
for k = 1:N
    if k < N
        F = F + [s_i(:,k+1) == (Ad*s_i(:,k) + Bd*f_i(:,k))];
    end
end
```

C.6: Adding acceleration and velocity constraints

```
%% Acceleration and velocity constraints

    for k = 1:N
        for m = 1:M
        F = F + [((f_i(1,k)*sin((2*pi*m)/M)) + ...
                  (f_i(2,k)*cos((2*pi*m)/M))) ≤f_max];

        F = F + [((s_i(3,k)*sin((2*pi*m)/M)) + ...
                  (s_i(4,k)*cos((2*pi*m)/M))) ≤ speed(1,k)];

        F = F + [((s_i(3,k)*sin((2*pi*m)/M)) + ...
                  (s_i(4,k)*cos((2*pi*m)/M))) ≥ speed(1,k) - ...
                  80*(1-b_m(m,k) )];
        end
    end

   for k = 1:N

       F = F + [sum(b_m(:,k)) == 1];
       F = F + [v_min ≤ speed(1,k) ≤ v_max];

   end
```

C.7: Adding waypoint constraints

```
%%  Waypoint constraints

Δ = 5; % A slack-variable wich specify that a waypoint
            %is wisited if the UAV is 5m from it

 % Adding waypoints to visit
for c = 1:length(wp_set(1,:))
    for k = 1:N

            F = F + [(s_i(1,k) - wp_set(1,c) -Δ) ≤...
                      (M_big*(1 - b_uas_i(c,k)))];

            F = F + [(s_i(1,k) - wp_set(1,c) -Δ) ≥...
                    - (M_big*(1 - b_uas_i(c,k)))];

            F = F + [(s_i(2,k) - wp_set(2,c) -Δ) ≤...
                      (M_big*(1 - b_uas_i(c,k)))];

            F = F + [(s_i(2,k) - wp_set(2,c) -Δ) ≥...
                    - (M_big*(1 - b_uas_i(c,k)))];


    end
        % Enforces that each waypoint shall be visited
        F = F + [(sum(b_uas_i(c,:))) == 1];

end
```

C.8: Adding small and big obstacles

```
%% Obstacles

ObstacleArray = [];
ObstacleBigArray   = [];

% Making safetymargins around obstacles, if there are any
if(length(O_set)>0)
for c=1:length(O_set(1,:))
    kollonnevektor = [O_set(1,c) - safety_margin; O_set(1,c) + ...
    safety_margin; O_set(2,c) - safety_margin; O_set(2,c) + safety_margin];

    ObstacleArray = [ObstacleArray kollonnevektor];

end

% Adding obstacles with safety margin
% to the optimization problem
for c = 1:length(O_set(1,:))
   for k = 1:N

        F = F + [s_i(1,k) ≤...
             ObstacleArray(1,c) + M_big* o_uas_i(c,1,k)];

        F = F + [-s_i(1,k) ≤...
```

```
            -ObstacleArray(2,c) + M_big*o_uas_i(c,2,k) ];

        F = F + [s_i(2,k) <=...
             ObstacleArray(3,c) + M_big* o_uas_i(c,3,k)];

        F = F + [-s_i(2,k) <=...
            -ObstacleArray(4,c) + M_big*o_uas_i(c,4,k)];

    end

     F = F + [o_uas_i(c,1,:) + o_uas_i(c,2,:) + o_uas_i(c,3,:) + ...
         o_uas_i(c,4,:) <= 3];
end
end
% Adding big obstacles
if(length(O_setBig)>0)
for c=1:length(O_setBig(1,:))
    kollonnevektor = [O_setBig(1,c) - sikkerhet; O_setBig(1,c) + ...
        sikkerhet; O_setBig(2,c) - safety_margin_big; O_setBig(2,c) +...
        safety_margin_big];

    ObstacleBigArray = [ObstacleBigArray kollonnevektor];

end

for c = 1:length(O_setBig(1,:))
   for k = 1:N

        F = F + [s_i(1,k) <=...
             ObstacleBigArray(1,c) + M_big* o_uas_Big(c,1,k)];

        F = F + [-s_i(1,k) <=...
            -ObstacleBigArray(2,c) + M_big*o_uas_Big(c,2,k) ];

        F = F + [s_i(2,k) <=...
             ObstacleBigArray(3,c) + M_big* o_uas_Big(c,3,k)];

        F = F + [-s_i(2,k) <=...
            -ObstacleBigArray(4,c) + M_big*o_uas_Big(c,4,k)];

    end

     F = F + [o_uas_Big(c,1,:) + o_uas_Big(c,2,:)+ ...
         o_uas_Big(c,3,:) + o_uas_Big(c,4,:) <= 3];
end
end
```

C.9: Adding cost function and optimization with GUROBI

```
% %% Cost function minimizing forces with a penalty
J = 0;
    for k = 1:N

        J = J +  w_slack(1,k) + w_slack(2,k);

        F = F + [f_i(1,k)  <= w_slack(1,k)];
        F = F + [-f_i(1,k) <= w_slack(1,k)];
        F = F + [f_i(2,k)  <= w_slack(2,k)];
```

```
        F = F + [-f_i(2,k) ≤ w_slack(2,k)];
    end



%% Cost function minimum time to end position

% theta = sdpvar(1);
% eta = sdpvar(1);
% J = 0;
% for k = 1:N
%
%   J = J + eta(1);
%
% end
%
% for k=1:N
%     F = F + [theta(1) ≤ M_big*(1-b_uas_i(1,k))];
%     F = F + [theta(1) ≥ k*(1-b_uas_i(1,k))];
%     F = F + [eta(1) ≥ theta(1)];
% end

%% optimization
diag = solvesdp(F,J,sdpsettings('solver','gurobi'));

%casting from sdpvar to matlab-language
res.s_i = double(s_i);
res.f_i = double(f_i);

 finiteHorizonPlot;
 %finiteHorizonPlotWithLegend;
```

C.10: Adding cost function and optimization with GUROBI

```

wp_set = [];

x_min = 0;
x_max = 850;
y_min = 150;
y_max = 550;

r_cam = 177;
dummy_x = x_min;
dummy_y = y_min;
waypoint_number = 1;
next_x_waypoint = 1;
next_y_waypoint = 1;

first_x_waypoint = 1;
first_y_waypoint = 1;
gamma = 0.8;
y_start = dummy_y + (1/sqrt(2))*r_cam*gamma;

x_counter = 0;

while(next_x_waypoint == 1)

```

```
25      if(first_x_waypoint == 1)
26      dummy_x = dummy_x + (1/sqrt(2))*r_cam*gamma;
27      elseif(x_max - dummy_x < r_cam*2*gamma)
28          dummy_x = x_max;
29      else
30          dummy_x = dummy_x + r_cam*2*gamma;
31      end
32              disp('done x')
33      while(next_y_waypoint == 1)
34         if(mod(x_counter,2) == 0 )
35
36                      if(first_y_waypoint==1)
37                         dummy_y = y_start;
38                         first_y_waypoint = 0;
39                           wp_set(:,waypoint_number) = [dummy_x; dummy_y];
40                         waypoint_number = waypoint_number +1;
41                         disp('1 x')
42                      elseif((y_max - y_start) > r_cam*2*gamma)
43                         dummy_y = dummy_y + r_cam*2*gamma;
44                          wp_set(:,waypoint_number) = [dummy_x; dummy_y];
45                         waypoint_number = waypoint_number +1;
46                          disp('2 x')
47                      elseif(sqrt( (dummy_x - x_min)^2 + (y_max - y_start )^2 )...
48                              > r_cam*gamma && dummy_x - x_min ≤ r_cam*gamma)
49                         dummy_y = y_max;
50                         wp_set(:,waypoint_number) = [dummy_x; dummy_y];
51                         waypoint_number = waypoint_number +1;
52                         next_y_waypoint = 0;
53                         disp('3 x')
54                      elseif( sqrt( (x_max - dummy_x)^2 + (y_max -
    y_start )^2 ...
55                          ) > r_cam*gamma && x_max - dummy_x ≤ r_cam*gamma)
56                         dummy_y = y_max;
57                         wp_set(:,waypoint_number) = [dummy_x; dummy_y];
58                         waypoint_number = waypoint_number +1;
59                         next_y_waypoint = 0;
60                         disp('4 x')
61                      else
62                        next_y_waypoint = 0;
63                        disp('5x')
64                      end
65
66
67          else
68                      if(first_y_waypoint==1)
69                         dummy_y = y_start;
70                         first_y_waypoint = 0;
71                         wp_set(:,waypoint_number) = [dummy_x; dummy_y];
72                         waypoint_number = waypoint_number +1;
73                         disp('1 x ned')
74                      elseif( (dummy_y - y_min) ≥ r_cam*gamma  )
75                         dummy_y = dummy_y - r_cam*2*gamma;
76                         wp_set(:,waypoint_number) = [dummy_x; dummy_y];
77                         waypoint_number = waypoint_number +1;
78                              disp('2 x ned')
79                      elseif( sqrt( (x_max - dummy_x)^2 + (dummy_y -
    y_min )^2 )...
80                              > r_cam*gamma && x_max - dummy_x ≤ r_cam*gamma)
81                         dummy_y = y_min;
82                         next_y_waypoint = 0;
```

```
83                      wp_set(:,waypoint_number) = [dummy_x; dummy_y];
84                      waypoint_number = waypoint_number +1;
85                      disp('3 x ned')
86                  else
87                     next_y_waypoint = 0;
88                     disp('4 x ned')
89                   end
90
91
92          end
93       y_start = dummy_y;
94       end
95
96       x_counter = x_counter +1;
97       first_y_waypoint = 1;
98       next_y_waypoint  = 1;
99       first_x_waypoint = 0;
100    if( (x_max- dummy_x) < r_cam )
101              next_x_waypoint = 0;
102    end
103
104 end
```

Appendix D

Digital Appendix

1 CD is attached with two folders: One which contains the Fixed horizon path planner design with obstacles known *a priori* and the *obstacle radar detection* design. Also a file which plots the results is attached. The other file contains the the implementation which was done for the two different search missions.