Contents

List of Figures

Chapter 1

Introduction

Unmanned Aerial Vehcles (UAV) are today widely used in ground observation, and by equipping them with different sensors they can be used in different situations. While the use of UAV eases many cases of ground obsercation, there are some difficulties related to the attitude of the aircraft. When the sensor is attached directly to the aircraft the sensor will be coupled with the UAVs states, so that any change in the UAV states will cause a change in what is actually observed by the sensor. An illustration of this is shown in figure 1.1.

A common solution to decouple the sensor from the UAV states is to attach the sensor to a gimbal which will countaract most of the movements of the UAV. While this is a good solution for decoupling, it raises some new issues regarding its weight and size. As one of the benefits of UAVs is their small size the gimbal can quickly be too big and heavy for the UAV, and it may give less effective aerodynamics. This may again lead to increased fuel consumption for the UAV.

This paper will investigate methods to reduce image errors caused by the UAVs attitude, while also avoiding the extra costs associated with a gimbal. This will be accomplished by optimizing a pre-defined curved path that is to be observed with a model of the UAV. The control method will be developed with a hyperspectral pushbroom camera that is fixed to the UAV in mind.

1.1 Related Work

The most common method to decouple the UAV attitude states from the sensor today is to equip the aircraft with a gimbal, which results in easy UAV operation without losing track of the features that is to be observed. However, the gimbal angles have limited range and if the UAV angles are to big the features may be lost from the sensor field of

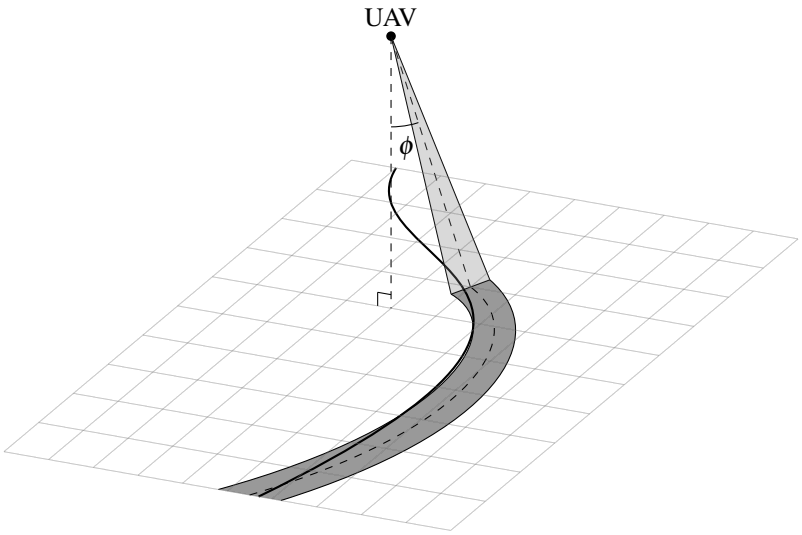

Figure 1.1: An illustration showing a UAV that uses a camera fixed to its body to observe a ground path, and how the position of the camera footprint relates to the UAV attitude.

view (FOV). Optimization may be used to create trajectories that ensure that the camera stays focused on the object [1]. Another solution that uses optimization without the use of gimbal is to put constraints on the the UAVs roll angle and altitude [2].

A simpler solution to avoid lateral movements of the FOV is to change the UAV course by using the rudder instead of the ailerons. The rudder deflection creates a yawing moment [3] which causes the aircraft to change course. This type of controller is referred to both as a Rudder Augmented Trajectory Correction (RATC) controller [3] and a skid-to-turn (STT) controller [4]. Results show that the performance of these controllers are comparable to conventional controllers using roll to change course, and that errors in the images is greatly reduced [3] [4] [5].

While the controllers offer a solution to the control problem that reduces the errors in the images, they do not ensure that the ground path that is to be observed will always stay inside the sensors FOV. Optimization is a commonly used method for ensuring that the application succeeds at some ”outside” goal, and can be used to e.g. minimize the risk of being identified by radars [6] or plan a path that ensures that a slung load attached to a helicopter do not collide with obstacles [7]. The use of optimization to minimize the error of the sensor footprint for a fixed camera has been done by Jackson [8], by using on-board motion planning.

Jackson presents a path planner that aims to minimize the error between the target on the ground, and the footprint of a camera fixed to a UAV by using a Nonlinear Model Predictive Controller (NMPC). The NMPC is compared to a PID and a sliding-mode controller that seek to follow the same path. Simulations of the three controllers show

that while the PID controller had much bigger crosstrack errors than the other two, the NMPC and the sliding-mode controller had similar performance. Simulations proved that the NMPC controller was able to find a near optimal solution with the performance characteristics of a real-time application.

One important point made by Jackson is that perfect tracking of a ground path with a fixed camera is not possible, as the controller that attempts to solve this would be unstable. The reason for this is the camera positions dependence on the roll angle. When the path turns right and the aircraft rolls right to follow this path, the camera position will move left, away from the path. This only applies to controllers that attempt a perfect tracking of the path, so that a near-perfect tracking is still achievable.

1.2 Hyperspectral Imaging

The control method developed in this paper will be developed with the use of a fixed hyperspectral, pushbroom sensor in mind. A hyperspectral sensor/camera makes it possible to accurately detect types of material from the UAV by sensing the wavelength of the received light.

1.2.1 Description

Hyperspectral imaging uses basics from spectroscopy to create images, which means that the basis for the images is the emitted or reflected light from materials [9]. The amount of light that is reflected by a material at different wavelengths is determined by several factors, and this makes it possible to distinguish different materials from each other. The reflected light is passed through a grate or a prism that splits the light into different wavelength bands, so that it can be measured by a spectrometer.

When using a hyperspectral camera for ground observation from a UAV, it is very likely that one pixel of the camera covers more than one type of material on the ground. This means that the observed wavelengths will be influenced by more than one type of material. This is called a composite or mixed spectrum [9], and the spectra of the different materials are combined additively. The combined spectra can be split into the different spectra that it is build up of by noise removal and other statistical methods which will not be covered here.

1.2.2 UAV Ground Observation

Hyperspectral imaging is already being used for ground observation from UAVs. Its ability to distinguish materials based on spectral properties means that it can be used to retrieve information that normal cameras are not able to. For example in agriculture it can be used to map damage to trees caused by bark beetles [10], or it can be used to

measure environmental properties, for example chlorophyl fluorescense, on leaf-level in a citrus orchard [11].

Systems for ground observation with hyperspectral cameras can be very complex, which often leads to heavy systems. In [12], a lightweight hyperspectral mapping system was created for the use with octocopters. The purpose of the system is to map agricultural areas using a spectrometer and a photogrammetric camera, and the final ”ready-to-fly” weight of the system is 2.0 kg. The resolution of the final images made it possible to gather information on a single-plant basis, and the georeferencing accuracy was off by only a few pixels.

The tests were performed at a low altitude, maximum 120 m. While this was mainly because of local regulations, it also gave a benefit as there was less atmosphere disturbance in the measurements. The UAVs orientation data combined with surface models was used when recovering the positional data in the images. However, they found that externally produced surface models was not accurate enough as they do not take vegetation into consideration. For this reason they supplemented the existing surface models with information gathered during flight.

1.3 Thesis Outline

Chapter 2

Kinematics

What is captured by the camera, the camera footprint, when the camera is fixed to the aircraft body is dependent of the position and the attitude angles of the aircraft. In this section a model for calculating the camera footprint on the ground assuming flat earth will be presented, as well as the necessary UAV states for this thesis.

2.1 UAV Model

In this thesis two different UAV models will be used, both of them are 6 DOF models presented by Beard & McLain [13]. For simulations of the UAV the nonlinear model will be used, while a linearized version of this model will be used as the prediction model in the MPC that will be presented in chapter 3. In this section the UAV states described by this model will be presented, as well as the linearization method. The linearization method will be the one described by Beard & McLain [13].

2.1.1 UAV States

The position of the UAV will be given using the North East Down (NED) coordinate frame denoted $\{n\}$:

$$\mathbf{p}_{b/n}^{n} = \begin{bmatrix} N \\ E \\ D \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix}. \tag{2.1}$$

Following the notation used in [13], the velocities of the UAV will be given in the body frame denoted $\{b\}$:

$$\mathbf{V}_g^b = \begin{bmatrix} u \\ v \\ w \end{bmatrix}. \tag{2.2}$$

The attitude $\mathbf{\Theta}_{nb}$ of the UAV will be given as Euler-angles, with the corresponding angular velocities $\dot{\mathbf{\Theta}}_{nb}$:

$$\mathbf{\Theta}_{nb} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad \dot{\mathbf{\Theta}}_{nb} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2.3}$$

2.1.2 Linearizing the Model

The UAV model is given on state space form with 12 states $\mathbf{x}$ and four control inputs $\mathbf{u}$. The nonlinear model is linearized about the *trim state* $\mathbf{x}^*$ of the UAV, the state where the UAV maintains a straight level flight without any changes in the control inputs. The trimmed states satisifies the following equation [13]:

$$\dot{\mathbf{x}} = f(\mathbf{x}^*, \mathbf{u}^*) = 0. \tag{2.4}$$

When linearizing the model, the difference between the nonlinear states $\mathbf{x}$ and the linearized states $\bar{\mathbf{x}}$ is calculated. The relationship between these two are defined as $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$. By calculating the derivative of this relationship, the linearized dynamics of the model is found.

In addition to being linear, the linearized model is decoupled into lateral and longitudinal models. The lateral and longitudinal states are given as:

$$\begin{aligned} \dot{\mathbf{x}}_{lat} &= \begin{bmatrix} v & p & r & \phi & \psi \end{bmatrix}^\mathsf{T}, \mathbf{u}_{lat} = \begin{bmatrix} \delta_a & \delta_r \end{bmatrix}^\mathsf{T} \\ \dot{\mathbf{x}}_{lon} &= \begin{bmatrix} u & w & q & \theta & h \end{bmatrix}^\mathsf{T}, \mathbf{u}_{lon} = \begin{bmatrix} \delta_e & \delta_t \end{bmatrix}^\mathsf{T}. \end{aligned} \tag{2.5}$$

2.2 Camera Footprint

The camera footprint is coupled with all of the three angles given in $\mathbf{\Theta}$. The position of the camera footprint will be calculated using forward kinematics, and the situation is shown in figure 2.1.

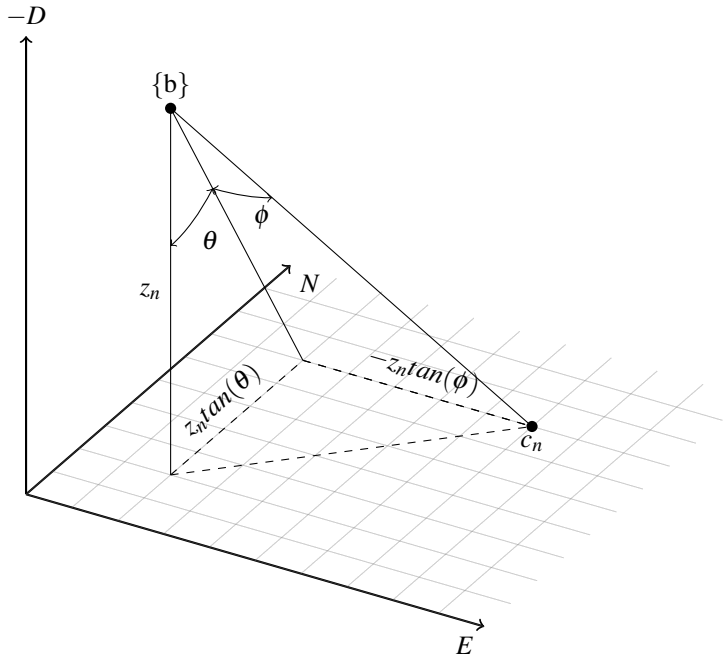

Figure 2.1: Illustration of how the aircraft attitude influence the camera position.

2.2.1 Centre Position

The attitude of the UAV is given in the body frame $\{b\}$ and the height z_n is given in the NED frame $\{n\}$, and the model assumes flat earth. The position of the footprint centre point $\mathbf{c}_b^b$ in the body frame $\{b\}$ is expressed as the geometric (???) distance from the UAV position to the footprint centre point:

$$\mathbf{c}_b^b = \begin{bmatrix} c_{x/b}^b \\ c_{y/b}^b \end{bmatrix} = \begin{bmatrix} z_n tan(\theta) \\ -z_n tan(\phi) \end{bmatrix}. \tag{2.6}$$

The coordinates of the camera position in $\{n\}$ can be found by rotating the point $\mathbf{c}_b^b$ with respect to the aircraft heading ψ, and by translating the rotated point to the aircrafts position in the $\{n\}$ frame. The rotation matrix for rotating with respect to the heading is given as

$$\mathbf{R}_{z,\psi} = \begin{bmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{bmatrix}. \tag{2.7}$$

The final expression for the camera footprint centre position $\mathbf{c}^n$ in the $\{n\}$ frame then becomes:

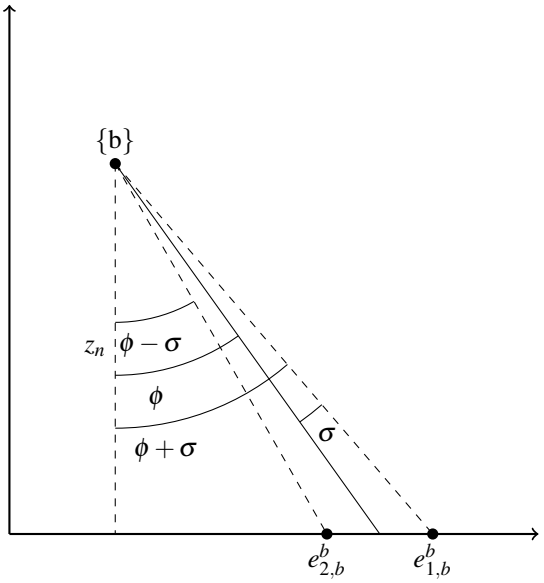

Figure 2.2: Illustration of how the field of view for a pushbroom sensor is calculated.

$$\begin{aligned} \mathbf{c}^n &= \mathbf{p} + \mathbf{R}_{z,\psi}\mathbf{c}^b_b \\ &= \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \mathbf{R}_{z,\psi} \begin{bmatrix} x^b_{x/b} \\ c^b_{y/b} \end{bmatrix} \end{aligned} \tag{2.8}$$

2.2.2 Edge Points

A hyperspectral pushbroom sensor captures images in a line, and the centre point of the camera footprint does not express the entire area that is captured by the sensor. The edge points of the camera footprint are calculated with respect to the sensor's field of view, as shown in figure 2.2. These points **e** can be found by altering 2.6:

$$\mathbf{e}^b_{1,b} = \begin{bmatrix} z_n tan(\theta) \\ -z_n tan(\phi + \sigma) \end{bmatrix}, \quad \mathbf{e}^b_{2,b} = \begin{bmatrix} z_n tan(\theta) \\ -z_n tan(\phi - \sigma) \end{bmatrix}. \tag{2.9}$$

The steps for writing the edge points **e** in the $\{n\}$ is similar as in equation 2.8:

$$\mathbf{e}^n = \mathbf{p} + \mathbf{R}_{z,\psi}\mathbf{e}^b_b. \tag{2.10}$$

Chapter 3

Model Predictive Control

Model Predictive Control (MPC) is a term used to describe control methods that uses knowledge about the process to calculate the future control inputs to the system in order to follow a reference trajectory [14]. In this chapter the equations for an *offline intervalwise MPC* that seeks to minimize the distance between the camera centre point and the ground path that is to be observed will be given. A linear state space-model for the UAV will be used to predict the future states and control inputs.

3.1 MPC Method

The MPC strategy can be broken down into three tasks [14]:

1. Predict the future outputs of the process for the given prediction horizon using past inputs to the process and the past measured states of the process, and by using the future control signals.
2. Optimize an objective function in order to determine the future control signals that follows a given reference trajectory as closely as possible.
3. Apply the optimal control signals to the process, and measure the resulting output so that it may be used to calculate the next prediction horizon in the first task.

In short MPC problems are made up of three elements [14]: Prediction model, objective function and the control law. The prediction model represents the model of the process that is to be controlled, and will in this case consist of the differential equations for the states of the UAV. The objective function is the function that is to be minimized by the optimization algorithm, in this case this will be the distance from the camera centre point to the desired ground path together with some of the UAV states that will give a stable flight. The objective function represents the reference trajectory that the UAV is

to follow. The control law introduces constraints on the problem, reducing the number of feasible solutions. These constraints can be put on either the states or the control inputs for the UAV.

A common mathematical formulation of the three elements that make up the optimization problem is shown in 3.1 [15]. $f(x)$ represents the objective function that is subject to equality and inequality constraints respectively. The equality constraints are used to represent the UAV model, while the inequality constraints represent the constraints used for the control law. A MPC differ from other optimization problems mostly in the objective function, which will be described in detail chapter 3.3.

$$\begin{aligned} \min_{x \in R^n} \quad & f(x) \\ \text{s.t} \quad & c_i(x) = 0, i \in \varepsilon, \\ & c_i(x) \geq 0, i \in I. \end{aligned} \tag{3.1}$$

3.2 Offline Intervalwise MPC

The control problem in this thesis will be solved by using an offline intervalwise MPC to generate an optimal path that will reduce the image error when using a fixed camera to survey a ground track. The generated path is intended to be tracked by the autopilot on the actual UAV that will perform the survey, with th eintention of optimally surveying the ground path.

3.2.1 Offline MPC

An *offline MPC* means that the initial state of the MPC is not a measurement of the UAV states, but rather the result of a simulation of the UAV. This means that the result from the prediction model used in the MPC will act as the physical system, and the outputs of the model will be fed back as inputs to the MPC for every iteration. The equations of the offline MPC are the same as the ones for the online version.

Rawlings & Mayne [16] refers to this kind of problem as a *deterministic problem* since there is no uncertainty in the system. A feedback loop in this kind of system is also not needed in principle, since it does not present any new information. They also state that an MPC action for a deterministic system is the same as the action from a *receding horizon control law* (RHC), which is another kind of predictive control.

3.2.2 Intervalwise MPC

Altough the feedback is not needed to give new information, it eases the computational load of the control problem as optimizing the path over a long time horizon leads to a very complicated problem. For this reason a *intervalwise MPC* will be used. The term

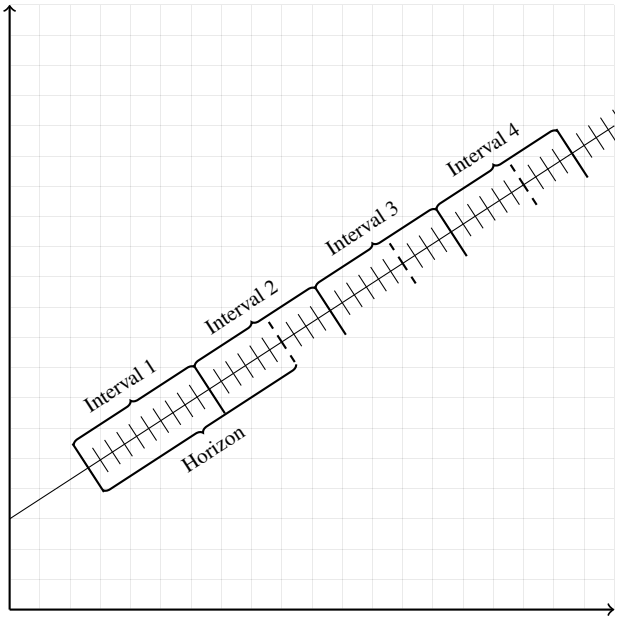

Figure 3.1: The figure shows how the path is divided into sections and horizons when using the intervalwise MPC.

intervalwise has been introduced by Kwon & Han [17] to describe a type of receding horizon controller that implements the same strategy.

Commonly a MPC is used to optimize the model over a given *horizon*, where the initial states are given. After the optimization has finished the first timestep of the optimization is returned and applied to the system, before a measurement of the system is performed. The new measurements are given as initial states for the next horizon, and so on.

The principle is the same for an intervalwise MPC. However, instead of only returning the first timestep an *interval* of timesteps are returned, and the last timestep of the interval is used as intial states for the next optimization horizon. This way the number of MPC iterations is reduced, and the increased complexity by having long optimization horizons is avoided. Figure 3.1 shows how timesteps, intervals and horizons relate to each other.

3.3 Objective Function

The main objective of the MPC developed in this thesis is to minimize the cross track error between the centre point of the camera footpring and the ground path that is to be observed. This, together with other objectives, will be defined in the objective function

of the optimization problem. In this section a way of formulating the objective function, least-squares, will be described, and how it can expressed to function as a MPC.

3.3.1 Least-Squares Problem

In many applications the objective function is formulated as a least-square (LSQ) problem. LSQ is a form of regression where the distance between a measurement and a known model is computed. In this case the known model is the reference signals, and the distance between the current states and the reference signal is calculated by the LSQ. The general mathematical formulation for LSQ is [15]:

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x) = \frac{1}{2}\sum_{j=1}^{m} |\phi(x,t_j) - y_j|. \tag{3.2}$$

In equation 3.2 r_j is called the residual function, which represents the distance between the measurement y_j taken at time t_j, and the model ϕ. In the optimization problem the residual function is what the algorithm seeks to minimize by chossing the parameters x that gives the lowest possible value of the residual function r_j.

In order to have a reference model that the measurements can be compared to the desired values will be associated with timepoints. This means that the optimization algorithm will at given timepoints compare the current values of x to the value of the reference model at the same time. A visual representation of this is shown in figure 3.2.

3.3.2 MPC Objective Function

The objective function is where the goal of the optimization is expressed, together with the optimization horizon of the problem. Typical goals of the optimization is to follow a predefined trajectory or reference signal while reducing the control inputs used. This can be expressed as follows [14]:

$$J(N_1,N_2,N_3) = \sum_{j=N_1}^{N_2} \delta(j)[\hat{y}(t+j|t) - w(t+j)]^2 + \sum_{j=1}^{N_u} \lambda(j)[\Delta u(t+j-1)]^2. \tag{3.3}$$

The first term of equation 3.3 represents the costs from the states of the model, and the second term represents the cost of the control effort. In the first term $\hat{y}$ is the value of the prediction model, which is compared to the desired trajectory w. In the second term the changes in control Δu is expressed. The change in control is used instead of the value of the control signal itself, since the steady state of the control signal may not be zero. δ and λ are weighting variables which offers a way of tuning the MPC. The three different N coefficients defines the horizon over which the states and the control

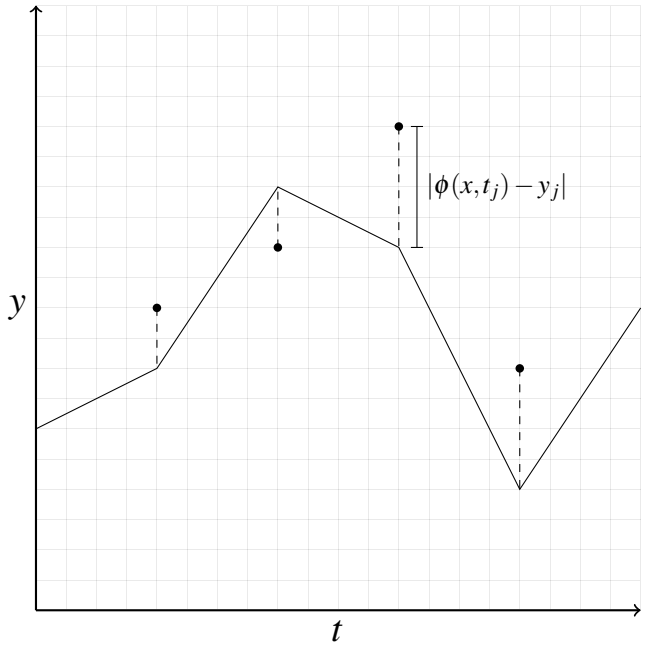

Figure 3.2: The distance between the measurements, represented by dots, and the model, represented by a line.

effort should be optimized. The optimization horizon for states and control effort can be different, but they will stay the same for this problem.

3.4 Problem Definition

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \mathbf{J}_{i+k} = \tfrac{1}{2}\sum_{j=i}^{j+L}[(\mathbf{y}(\mathbf{x}_j)-\mathbf{y}_{d,j})^\intercal \mathbf{Q}(\mathbf{y}(\mathbf{x}_j)-\mathbf{y}_{d,j}) + (\Delta\mathbf{u}_j)^\intercal \mathbf{R}(\Delta\mathbf{u}_j)] \\ \text{s.t} \quad & \mathbf{x}^{low} \leq \quad \mathbf{x}_j \quad \leq \mathbf{x}^{high} \\ & \mathbf{u}^{low} \leq \quad \mathbf{u}_j \quad \leq \mathbf{u}^{high} \\ & \Delta\mathbf{u}^{low} \leq \quad \Delta\mathbf{u}_j \quad \leq \Delta\mathbf{u}^{high} \\ & \dot{\mathbf{x}}_{j+1} \quad = \quad f(\mathbf{x}_j, \mathbf{u}_j) \end{aligned} \tag{3.4}$$

The equations for the full optimization problem is shown in equation 3.4. The objective function uses the same setup as shown in equation 3.3, but on matrix form. Each of the three components of the problem definition will be described in detail in the following sections.

The objective function $\mathbf{J}$ will be minimized over the entire optimization horizon, which consists of L timesteps. The current timestep is denoted i. Since this is an intervalwise MPC, as described in section 3.2, all states within in the interval, denoted k, will be stored. The number of intervals for the entire optimization problem is denoted n, which means that for the entire problem there will be a total of nk timesteps to store.

3.4.1 Prediction Model

The linear decoupled 6 DOF UAV model presented in chapter 2 will be used as the prediction model for the MPC. The model is associated with the following states and control inputs:

$$\mathbf{x} = \begin{bmatrix} p_N & p_E & h & u & v & w & \phi & \theta & \psi & p & q & r \end{bmatrix}^\intercal \tag{3.5a}$$

$$\mathbf{u} = \begin{bmatrix} \delta_e & \delta_a & \delta_r & \delta_t \end{bmatrix}^\intercal. \tag{3.5b}$$

The prediction model relates to the equality constraints of equation 3.1 in the form of differential equations. As explained in the previous chapter the control rates $\Delta\mathbf{u}$ shown in equation 3.10, which means that by the optimization solver $\Delta\mathbf{u}$ will be handled as the control input. The control surfaces $\mathbf{u}$ are calculated from the rates $\Delta\mathbf{u}$ through integration:

$$\dot{\mathbf{u}} = \Delta\mathbf{u}. \tag{3.6}$$

As shown in equation 3.5a, the attitude angles of the UAV will be given by the Euler angles ϕ, θ and ψ. Even though quaternions offer more efficient computations and no gimbal lock [13], this optimization will be run offline before the flight takes place so

that computation capacity is not a big issue. In addition the UAV will not by performing any high-angle maneuvers so that a gimabl lock should never occur.

3.4.2 Objective Function

$$\mathbf{J} = \frac{1}{2}\sum_{i=0}^{L}[(\mathbf{y}(\mathbf{x}_i) - \mathbf{y}_{d,i})^\intercal \mathbf{Q}(\mathbf{y}(\mathbf{x}_i) - \mathbf{y}_{d,i})] + \frac{1}{2}\sum_{i=0}^{L}[(\Delta\mathbf{u}_i)^\intercal \mathbf{R}(\Delta\mathbf{u}_i)] \tag{3.7}$$

The first term of the objective function calculates the distance between the UAV states and the reference trajectory. The vector $\mathbf{y}_d$ is the *measurement vector* which is the references for the states:

$$\mathbf{y}_d = \begin{bmatrix} c_{x_d} & c_{y_d} & h_d & u_d \end{bmatrix}^\intercal. \tag{3.8}$$

The function $\mathbf{y}(\mathbf{x})$ holds the current values for the optimization problem. While the height h and velocity u can be used as-is, the camera centre point $\mathbf{x}^n$ needs to be calculated using equation 2.8:

$$\mathbf{y}(\mathbf{x}) = \begin{bmatrix} p_N + h\cos(\psi)\tan(\theta) - h\sin(\psi)\tan(\phi) \\ p_E + h\sin(\psi)\tan(\theta) + h\cos(\psi)\tan(\phi) \\ h \\ u \end{bmatrix}. \tag{3.9}$$

In order to reduce the control effort for the optimization problem the rate of change of the control inputs $\Delta\mathbf{u}$ will be minimized. Since all the control rates is to be compared to zero no function is needed. The vector u contains of the four control rates:

$$\Delta\mathbf{u} = \begin{bmatrix} \Delta\delta_e & \Delta\delta_a & \Delta\delta_r & \Delta\delta_t \end{bmatrix}^\intercal. \tag{3.10}$$

The matrices $\mathbf{Q}$ and $\mathbf{R}$ are the weighting matrices. They are diagonal matrices where each row represent one state or control rate. The higher the value in the row, the more value is given to the difference between the corresponding state or control rate and reference.

3.4.3 Control Law

$$\mathbf{x}^{\text{low}} \leq \mathbf{x} \leq \mathbf{x}^{\text{high}} \tag{3.11a}$$

$$\mathbf{u}^{\text{low}} \leq \mathbf{u} \leq \mathbf{u}^{\text{high}} \tag{3.11b}$$

$$\Delta\mathbf{u}^{\text{low}} \leq \Delta\mathbf{u} \leq \Delta\mathbf{u}^{\text{high}} \tag{3.11c}$$

The control law for the optimization problem consists of inequality constraints put on the UAV states and on the control signals. When constraints are put on the optimization

problem the number of feasible solutions is reduced, so if too many constraints are put on the problem it will make the problem more difficult than necessary.

In an attempt to reduce the complexity of the optimization problem, the constraints put on UAV states shown in equation 3.11a will not be included to begin with. This is because it is assumed that the ”easiest” way to fly the aircraft is the ”correct” way. However, if this proves wrong during testing, constraints on some or all of the states may be included.

The control states, shown in equation 3.11b, are restricted by the physical maximum value of deflection. Since these constraints directly relates to physical values they are needed in order to get a meaningful optimization of the aircraft.

The constraints put on the control rates of the control surfaces and throttle shown in equation 3.11c also relate to a physical value, and are therefore needed to get a meaningful simulation as well. It is worth noting that $\Delta\mathbf{u}$ is also present in the objective function shown in equation 3.7, with a reference signal of zero. This means that the problem already seeks to minimize the values of $\Delta\mathbf{u}$ so that the constraints may be unnecessary.

Chapter 4

MPC Implementation

The offline intervalwise MPC presented in chapter 3 will be implemented using C++ and the ACADO Toolkit [18]. The implementation conists of two main parts: the MPC algorithm that prepares the optimization problem, and the optimization solver that will use the ACADO Toolkit to solve the optimization problem.

This chapter will describe the ACADO Toolkit, how to use it and how it works; as well as the MPC algorithm.

4.1 ACADO toolkit

The ACADO Toolkit [18] is an open-source toolkit that supports several different methods for solving optimization problems. The toolkit provides methods to solve four different classes of optimization problem: Optimal control problems, multi-objective optimization and optimal control problems, parameter and state estimation problems, and model predictive control. Even though the toolkit will be used to create an MPC in this case, the optimal control problems (OCP) class will be used instead. The reason for this is that between each iteration of the MPC algorithm a new trajectory must be generated, and the MPC problem class does not have the functionality needed to do this. How the trajectory is calculated and how it is passed to the toolkit will be explained later in this chapter.

4.1.1 Discretization (Workin' Title)

The model and optimization problem will be written on continuous time form, which means that it has to be discretized in order to be solved. ACADO solves this by using a method called multiple-shooting discretization.

4.1.2 Runge-Kutta Method

The Runge-Kutta method is a form of *numerical integrator* that can be used to solve differential equations, and is used by the ACADO toolkit to integrate the prediction model. The method is based on the Euler method, which is a very simple method for numerical integration.

The ACADO toolkit provides algorithms for *explicit* Runge-Kutta methods [18], where explicit means that the method calculates the state of the system at a later time based on the current state (THIS NEEDS SOURCE). The Runge-Kutta method calculates the later state of the system by calculating several approximations of the derivative of the system. The current state of the system together with a linear combination of the approximated derivatives gives the next state of the system [19]. For a system on the form $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t)$, the Runge-Kutta method can be mathematically expressed as:

$$\mathbf{k}_i = \mathbf{f}(\mathbf{y}_n + h \sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j, t_n + c_i h), i = 1, ..., \sigma \tag{4.1a}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{j=1}^{\sigma} b_j \mathbf{k}_j \tag{4.1b}$$

where t_n is the current time, $\mathbf{y}_n$ is the output of the system at the current time, h is the step size and σ is the number of approximations of the derivative that is calculated. a, b and c are parameters for the specific Runge-Kutta method used, and c must satisfy $0 < c < 1$.

4.1.3 Solver

In order to solve the optimization problem, ACADO uses some kind of solver.

KKT-Tolerance

In order to rate the ”goodness” of a solution, ACADO uses some kind of KKT-Tolerance.

4.1.4 Nonlinear Model (Working Title)

Initially, effort was made to implement the nonlinear model presented by Beard & McLain [13] as the prediction model in the optimization problem. This would have given more precise results as the nonlinear model is a closer representation of the real UAV, and since the nonlinear model also includes wind the path could be optimized with the knowledge about the wind conditions as well. The level of calculation needed for the nonlinear model is significantly higher; however, since this implementation

is intended to run offline before the flight occurs, computation time is not a critical concern.

Achieving stable flight within in the optimization problem with the nonlinear model on the other hand, turned out to be a difficult task that was far from trivial. This is somewhat due to the nonlinearity, but also the high coupling between states in the model. The coupling causes changes in one state to affect many other states which results in a much more complex problem. Several different algorithm and solver settings was tried, as well as different objective functions and weighting of these functions. After many attempts the decision to use the linear model instead was made, largely due to this being a project with limited time available.

4.2 MPC

The task of the MPC part of the implementation is to supply the ACADO implementation with the information needed to perform the optimization, and also control the optimization algorithm so that the correct horizon is calculated, as well as storing the results in the correct order. The pseudocode for the MPC implementation is shown in algorithm 1 and 2.

Algorithm 1 Offline Intervalwise MPC Algorithm

procedure MPC
- *path* ← path from file
- *timestep* ← duration of timestep [s]
- *horizonlen* ← number of *timestep* in horizon
- *intervallen* ← number of *timestep* in interval
- *intervals* ← number of intervals needed to cover *path*
- x_0 ← initial values of states
- u_0 ← initial values of control states
- Δu_0 ← inital values of control rates
- *results[]* ← empty list to store result from optimization
- **for** each *interval* **do**
  - $\mathbf{c}^n$ ← calculate camera centre position using equation 2.8
  - *trajectory* ← GENERATEHORIZON(*path, timestep, horizonlen,* $\mathbf{c}^n$)
  - Solve optimization with initial states x_0, u_0, Δu_0 for current *horizon*
  - x_0 ← last x value in the *interval*
  - u_0 ← last u value in the *interval*
  - Δu_0 ← last Δu value in the *interval*
  - *result[]* ← the first *intervallen* number of *timesteps* from *horizon*

Algorithm 2 Generate horizon

procedure GENERATEHORIZON(*path, timestep, horizonlen,* $\mathbf{c}^n$)
  distance ← distance travelled during one timestep
  pos ← find the point in *path* that is closes to current camera position $\mathbf{c}^n$
  trajectory[] ← empty list to store the generated trajectory
  for each *timestep* in *horizonlen* **do**
    Find point pos_{temp} on *path* with the given *distance* away from current *pos*
    trajectory[] ← pos_{temp}
    pos ← pos_{temp}
  return *trajectory*

4.2.1 Generating the Trajectory

The ground path that is to observed is assumed to be *time independent*, meaning that it does not matter when a section of the path is captured by the sensor. However, the function that minimizes a least-squares objective function that is provided with the ACADO toolkit requires that the path is given as values with associated time points.

In order to meet this requirement a time dependent path will be generated at the beginning of every iteration of the MPC. This will be done by making the assumption that the UAV will maintain its reference speed throghout the horizon. With this assumption it is simple to calculate the distance the UAV will travel between every timestep, and based on this distance the desired position for the UAV at the next timestep may be calculated. Since the horizon is in the order of a few seconds and the predicted path is updated every iteration, this assumption will not lead to big errors. The principle behind the calculation is shown in figure 4.1.

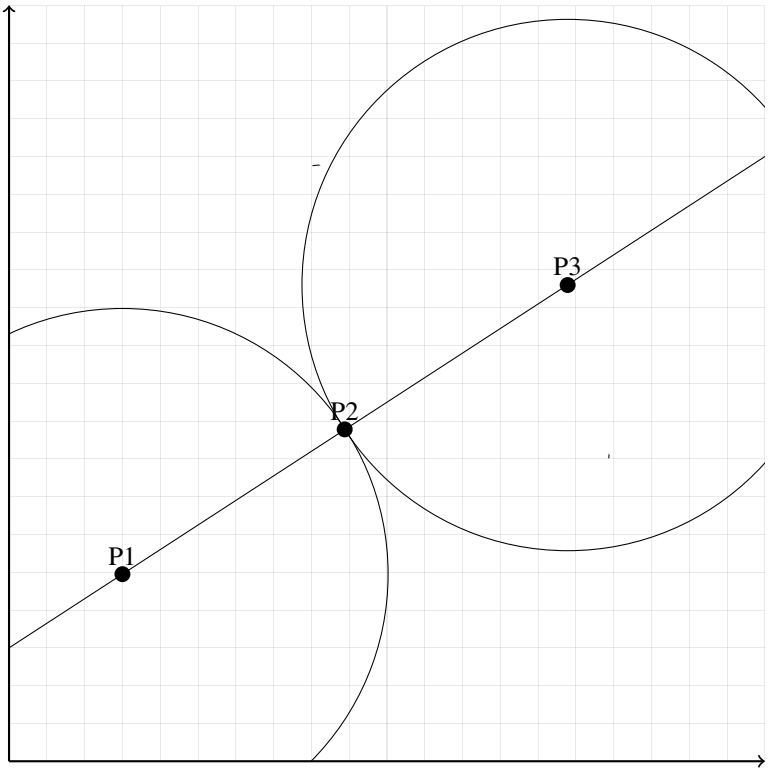

Figure 4.1: CAPTION

Chapter 5

Simulation Environment

5.1 Finding Trim Conditions

The trim conditions, as described in section 2.1, play an important role when using a linear model as this model is linearized about these points. Since the trim conditions also represent a straight level flight they are good initial states and controls for the simulation and are feasible points optimization. In order to find the trim conditions a built-in Matlab function together wtih a Simulink model will be used and the procedure that is used, which will be summarized in this section, is described by Beard & McLain [13].

Matlab features a built-in function `trim` that calculates the trim conditions of a given Simulink model. The Simulink model must be set up with the four control signals as inputs, and the output of the model is the airspeed V_a, the angle of attack α and the sideslip β. The out states are chosen as they easily expresses a trimmed stable flight. The airspeed is set to the desired cruise speed, while the angle of attack and sideslip is set to zero for a straight level flight. The function uses initial guesses of the states and inputs, also the derivatives, and what the desired output is. The Simulink model used for this thesis was developed by Gryte for his master thesis [20].

Chapter 6

Optimized Paths

6.1 Turns

6.1.1 Different Degrees

1. Linear paths
   (a) 45° turn
   (b) 90° turn
2. Curved paths
   (a) 45° turn
   (b) 90° turn

6.1.2 Different Radii

1. Curved paths
   (a) 45° turn
   (b) 90° turn
2. Radii
   (a) 100 m
   (b) 150 m
   (c) 200 m

(d) 250 m

6.2 Horizon Length

1. Linear paths
2. Curved paths
   (a) 150 m
   (b) 200 m
3. Degrees:
   (a) 45° turn
   (b) 90° turn
4. Horizon length
   (a) 50 steps
   (b) 75 steps
   (c) 100 steps
   (d) 120 steps

6.3 Path

1. Curved
2. Linear
3. Two opposite turns
4. Three turns?

Chapter 7

Simulating the Optimized Path

7.1 Curved paths

7.2 Linear paths

Chapter 8

Conclusion

Appendices

Appendix A

ACADO Code

Appendix B

MPC Code

Bibliography

[1] E. Skjong, S. A. Nundal, F. S. Leira, and T. A. Johansen. 2015 international conference on unmanned aircraft systems (ICUAS). In *Autonomous Search and Tracking of Objects Using Model Predictive Control of Unmanned Aerial Vehicle and Gimbal: Hardware-in-the-loop Simulation of Payload and Avionics*, Denver, Colorado, USA, June 2015. IEEE.

[2] J. Egbert and R. W. Beard. Proceedings of the 2007 American control conference. In *Low Altitude Road Following Constraints Using Strap-Down EO Cameras on Miniature Air Vehicles*, New York City, USA, July 2007. IEEE.

[3] Thomas M. Fisher. Rudder augmented trajectory correction for unmanned aerial vehicles to decrease lateral image errors of fixed camera payloads. *All Graduate Theses and Dissertations*, 2016.

[4] S. Mills, J. J. Ford, and L. Mejias. Vision based control for fixed wing UAVs inspecting locally linear infrastructure using skid-to-turn maneuvers. *Journal of Intelligent and Robotic Systems*, 61(1):29–42, 2011.

[5] M. Ahsan, H. Rafique, and Z. Abbas. Multitopic conference (INMIC). In *Heading Control of a Fixed Wing UAV Using Alternate Control Surfaces*. IEEE, December 2012.

[6] T. Inanc, K. Misovec, and R. M. Murray. 43rd IEEE conference on decision and control. In *Nonlinear Trajectory Generation for Unmanned Air Vehicles with Multiple Radars*, Atlantis, Paradise Island, Bhamas, December 2004. IEEE.

[7] Anders la Cour-Harbo and Morten Bisgaard. *State-Control Trajectory Generation for Helicopter Slung Load System using Optimal Control*. American Institute of Aeronautics and Astronautics Meeting Papers on Disc. 2009.

[8] Stephen P. Jackson. Controlling small fixed wing UAVs to optimize image quality from on-board cameras. *ProQuest Dissertations and Theses*, 2011.

[9] Randall B. Smith. *Introduction to Hyperspectral Imaging*. MicroImages, Inc., 2012.

[10] R. Nsi, E. Honkavaara, P. Lyytikinen-Saarenmaa, M. Blomqvist, P. Litkey, T. Hakala, N. Viljanen, T. Kantola, T. Tanhuanp, and M. Holopainen. Using UAV-based photogrammetry and hyperspectral imaging for mapping bark beetle damage at tree-level. *Remote Sensing*, 7(15467-15493), 2015.

[11] P. J. Zarco-Tejada, V. Gonzlez-Dugo, and J. A. J. Berni. Fluorescence, temperature and narrow-band indices acquired from a UAV platform for water stress detection using a micro-hyperspectral imager and a thermal camera. *Remote Sensing of Environment*, 117(322-337), 2012.

[12] J. Suomalainen, N. Anders, S. Iqbal, G. Roerink, J. Franke, P. Wenting, D. Hnniger, H. Bartholomeus, R. Becker, and L. Kooistra. A lightweight hyperspectral mapping system and photogrammetric processing chain for unmanned aerial vehicles. *Remote Sensing*, 6(11013-11030), 2014.

[13] Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, Princeton, NJ, USA, 2012.

[14] E. F. Camacho and Bordons C. *Model Predictive Control*. Springer London, 1999.

[15] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[16] J. B. Rawlings and D. Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, Madison, Wisconsin, 2015.

[17] W. H Kwon and S. H. Han. *Receding Horizon Control: Model Predictive Control for State Models*. Advanced Textbooks in Control and Signal Processing. Springer London, 2005.

[18] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[19] O. Egeland and J. T. Gravdahl. *Modeling and Simulation for Automatic Control*. Marine Cybernetics, Trondheim, Norway, 2002.

[20] K. Gryte and T. I. Fossen. *High Angle of Attack Landing of an Unmanned Aerial Vehicle*. Norges Teknisk-Naturvitenskapelige Universitet, Trondheim, Norway, 2015.