On-line mission planning based on Model Predictive Control

Zoran Sjanic

LiTH-ISY-EX-3221-2001

2001-12-05
On-line mission planning based on Model Predictive Control

Master thesis

Division of Automatic Control
Department of Electrical Engineering
Linköping University

Zoran Sjanic

Reg nr: LiTH-ISY-EX-3221-2001

Supervisors: Predrag Pucar Saab Gripen
Johan Löfberg LiTH

Examiner: Fredrik Gustafsson LiTH

Linköping, 5th December 2001
On-line mission planning based on Model Predictive Control

Zoran Sjanic

Modern air battles are very dynamic and fast, and put extreme pressure on pilots. In some unpredictable situations, like new discovered threats or mission plan deviation because of enemy aircraft, the pilots might need to replan their predefined flight route. This is very difficult, if not impossible, to do since numerous factors affect it. A system that can help the pilots to do such a thing is needed.

Previous work in this field has involved methods from artificial intelligence like A*-search. In this master thesis, implementation of a replanning system based on a control theory method, Model Predictive Control (MPC), is examined. Different factors influencing the path, such as terrain and threats, are included in the algorithm.

The results presented in this thesis show that MPC solves the problem. As with every method there are some drawbacks and advantages, but as a summary the method is a very promising one and is worth further development.

Proposals of future work and different improvements of the algorithms used here are presented in this report as well.
Nyckelord
Keyword
MPC, mission planning, threats, terrain database, path refining, prediktionsreglering
Abstract

Modern air battles are very dynamic and fast, and put extreme pressure on pilots. In some unpredictable situations, like new discovered threats or mission plan deviation because of enemy aircraft, the pilots might need to replan their predefined flight route. This is very difficult, if not impossible, to do since numerous factors affect it. A system that can help the pilots to do such a thing is needed.

Previous work in this field has involved methods from artificial intelligence like A*-search. In this master thesis, implementation of a replanning system based on a control theory method, Model Predictive Control (MPC), is examined. Different factors influencing the path, such as terrain and threats, are included in the algorithm.

The results presented in this thesis show that MPC solves the problem. As with every method there are some drawbacks and advantages, but as a summary the method is a very promising one and is worth further development.

Proposals of future work and different improvements of the algorithms used here are presented in this report as well.
Acknowledgements

This master thesis concludes my studies in Computer Science and Engineering at Linköping University. It has been done at Saab Gripen between June and December 2001.

I would like to thank all people that contributed in the making of this report and especially my supervisor at Saab Predrag Pucar and my supervisor at the university Johan Löfberg.

I wish to thank people at Flight Data and Navigation section at Saab Gripen for moral support. Special thanks goes to Daniel Murdin for necessary help with the terrain database and Peter Uhlin for his maps of Sweden.

I would also like to thank people at the HMI area at Saab Gripen and particularly Johan Holmberg and Andreas Ladell for their help with the Gripen simulator.

Linköping, December 2001
# Table of contents

## 1 Introduction

1.1 Background ........................................ 1
1.2 Problem description .............................. 2
1.3 Purpose .......................................... 3
1.4 Method ........................................... 3
1.5 Reader’s guide .................................... 3

## 2 Theoretical background

2.1 System description ................................. 5
2.2 State predictions .................................. 5
2.3 MPC .............................................. 5

## 3 The surrounding world

3.1 Representation of terrain ......................... 9
3.2 Representation of threats ......................... 10
    3.2.1 Straightforward representation of threats .... 11
    3.2.2 Alternative representation of threats ........... 12

## 4 Implementation

4.1 State-space description .......................... 15
4.2 Modification of the MPC algorithm ................ 16
4.3 Implementation of the planning algorithm ........ 18
    4.3.1 MPC algorithm .................................. 18
    4.3.2 Refining of the flight path ..................... 19
    4.3.3 Implementation of the approach bearing ..... 20

## 5 Results

5.1 The terrain ........................................ 23
5.2 Simulation results ................................ 24
    5.2.1 Variation of the prediction horizon .......... 25
    5.2.2 Variation of the step size .................... 27
    5.2.3 Variation of the horizontal angle .......... 30
    5.2.4 Variation of the vertical angle ............ 33
    5.2.5 Variation of the weight matrices ............. 35
    5.2.6 Variation of the amount of threats .......... 42
    5.2.7 Approach bearing ............................... 42
    5.2.8 Radar shadows .................................. 43
6  Conclusions .................................................. 45
7  Future work .............................. ........................................ 47
    7.1  Threat representation .............................................. 47
    7.2  Dynamic threats ...................................................... 47
    7.3  Fuel consumption .................................................... 47
    7.4  Using a linear model .................................................. 47
    7.5  Approach bearing .................................................... 47
References ......................................................... 49
Introduction

This chapter describes the background and the purpose of this thesis and introduces the reader to the on-line mission planning problem and the solution approach.

1.1 Background

Before each mission a detailed planning of a flight path is made. The planned flight path, also called polygon, consists of waypoints and polygon legs. The waypoints are points in three dimensional space, and the polygon legs are straight lines between those points, see Figure 1.1. Waypoints are a kind of landmarks that show the pilot in which direction he must fly, which speed and altitude he must maintain between two waypoints and the time when the pilot must get to a certain waypoint.

There are two waypoints that are different from others. One is the landing base and the other is the target. The landing base is start and end point of the polygon, while the target is situated somewhere in the middle of the polygon. The target point must usually be approached from some special direction called bearing. Sometimes it is necessary that the target also is approached from some distance. Besides all this mission planning also requires that factors like threats, terrain, fuel and weather are paid attention to.

If every mission always goes as planned there should never be any need for replanning. Of course, problems do come and replanning is necessary during the flight. One possible cause for replanning is that the pilot was involved in an air combat and was forced to diverge from the planned flight path. Also new threats that were unknown to the pilot during planning but are discovered during flight can be a reason for replanning, see Figure 1.2. Since the pilot is put under large pressure during the flight it is...
very difficult for him to do the replanning by himself. Some kind of automatic system that helps to replan is needed.

Figure 1.2: Replanning due to a new discovered threat. All distances are in km.

1.2 Problem description

The problem of replanning is simply described as how to find an optimal path between two points in three dimensional space when factors like terrain, threats, path length and fuel consumption are taken into account. This is not a trivial problem and a completely optimal solution with respect to all of these factors may not be found. It is important that the replanning system at least produces flight paths that do not differ too much from flight paths that the pilots would make under the same conditions. Of course all pilots do not make the same paths and what some pilots consider a good path other pilots do not. However, the individualism of the pilots is not considered here since it would be necessary to make different systems for different pilots.

In order to make as good flight paths as possible several things must be considered. The first thing to be taken into consideration is threats. Threats can be divided into two groups, static and dynamic. Static threats are threats that do not move, like radar facilities and missile launchers. Dynamic threats are those threats that move, like mobile missile launchers and enemy aircraft in the air. In this thesis only static threats are considered. Another thing with threats is that the cover area of the threat mainly is determined by the terrain. This means that if a hill stands between a radar station and an airplane then the airplane is safe from that radar. These “radar shadows” should be taken into consideration when planning is done because they can give shorter and more fuel saving paths. Also the uncertainty in threat position should be considered. In this thesis that problem is not considered and it is assumed that exact position of threats is known. On the other hand, uncertainty problems can be solved by adding an extra safety margin to the range of each threat.
Introduction

The second thing to pay attention to is terrain. Just like threats the terrain has to be avoided. The difference between threats and terrain is that the pilot can survive if he enters the threat area while he dies if he flies into the terrain.

The next thing to consider is fuel consumption. Fuel consumption depends on several things like flight altitude, altitude change and flown distance. If the path that is produced by the replanning system is too long then there may not be enough fuel to fulfill the mission and the pilot might decide to abort it. The same thing might happen if there are too many changes of altitude because fuel consumption is larger too. For these reasons the replanning system must take fuel into consideration and warn the pilot if the amount of fuel is insufficient to fulfill the mission.

From the discussion above one can understand that it is very difficult to find an optimal solution when all factors are included.

1.3 Purpose

The purpose of this master thesis is to examine and evaluate if it is possible to make good flight paths using model predictive control when crucial factors are taken into account. These factors are, as mentioned before, terrain, threats, fuel consumption and path length. There is another factor that is very important and it is execution time. It is desirable that the replanning system can produce a satisfactory flight path fast enough. For example, an exhaustive search is likely to come up with the best flight path but the time it would take is probably too long. For that reason near optimal paths must be accepted if they are to be found relatively fast.

Evaluation of the final result will be made in the Gripen simulator and pilots will be consulted about the performance.

1.4 Method

Although path planning is a discipline of artificial intelligence and robotics, a method from control theory is chosen to be used here. The method is called model predictive control (MPC) and is based on a state-space model of a system to be controlled. The predictive power of a model is used to “see in the future” and choose the best control action possible.

A variety of books and papers in the area of MPC and path planning are also studied in order to achieve a good theoretical understanding of the problem. Also interviews with persons who have experience in the area have been made.

The MPC algorithm is implemented in C++. C++ is used for several reasons. One reason is that C++ has faster execution than MATLAB and another is that the simulator software is written in C and C++. In order to graphically present the results MATLAB is used.

1.5 Reader’s guide

Chapter 2 covers the theory of the methods used in solving the path replanning problem. State-space models and model predictive control are explained.
In Chapter 3 representation of threats and terrain is covered.

Chapter 4 describes implementation issues. Necessary changes in the MPC algorithm and implementation of the algorithm are also explained.

In Chapter 5 the result of different simulation scenarios is presented.

Chapter 6 gives the conclusions of the simulations and the results achieved.

In Chapter 7 the reader can find suggestions for future improvements of the replanning system.
2 Theoretical background

In this chapter the theoretical background of methods used in this master thesis is explained.

In Section 2.1 the state-space representation of systems is presented and in Section 2.2 the state predictions based on the state-space description are explained. Section 2.3 covers MPC in more detail.

2.1 System description

Linear time-discrete state-space models have the following form [5]

\[
\begin{align*}
    x(k+1) &= Ax(k) + Bu(k) \quad (2.1) \\
    y(k) &= Cx(k) \quad (2.2)
\end{align*}
\]

while nonlinear state-space models have the more general form

\[
\begin{align*}
    x(k+1) &= f(x(k), u(k)) \quad (2.3) \\
    y(k) &= h(x(k)) \quad (2.4)
\end{align*}
\]

where \( x(k) \in \mathbb{R}^n, u(k) \in \mathbb{R}^m, y(k) \in \mathbb{R}^p \) denote the state, control input and measured output respectively and \( f \) and \( h \) are some functions. The control signal is in most cases constrained and it is written as \( u(k) \in U \), where \( U \) is assumed to be nonempty. As an example, \( u(k) \) could be bounded which is written as \( u_{\min} \leq u(k) \leq u_{\max} \).

2.2 State predictions

The power of a state-space description of a system is that it is easy to predict future outputs. These can be written as

\[
\begin{align*}
    x(k+j|k) &= Ax(k+j|k) + Bu(k+j|k) \quad (2.5) \\
    y(k+j|k) &= Cx(k+j|k) \quad (2.6)
\end{align*}
\]

where \( x(k+j|k) \) indicates the prediction of the state at the instant \( k+j \) calculated at instant \( k \).

Also the nonlinear state-space description can be used to calculate future outputs

\[
\begin{align*}
    x(k+j|k) &= f(x(k+j|k), u(k+j|k)) \quad (2.7) \\
    y(k+j|k) &= h(x(k+j|k)) \quad (2.8)
\end{align*}
\]

2.3 MPC

Model predictive control or MPC is a control strategy that explicitly uses a model of a system in order to predict the system behavior. This is then used to find the best con-
control signal possible by minimizing an objective function. This can be summarized as [2]:

- Explicit use of a model to predict the system output at future time instants (also called horizon).
- Obtaining a control signal by minimizing an objective function.
- Receding strategy, meaning that the horizon is moved forward, and only the first value of the control signal sequence calculated at each step is applied.

The MPC strategy can be compared to driving a car. The driver knows the desired reference trajectory for a finite horizon, and by taking the car characteristics into account, he decides which control actions to take in order to follow the desired trajectory. Only the first control actions (using accelerator, brakes and steering) are taken at each instant and the procedure is then repeated. This can be described in more detail by the following steps [2]:

1. The future outputs for a determined horizon $N$, called the prediction horizon, are predicted at each instant $t$ using the system model. These predicted outputs $y(k+j|k)$ for $j = 1 \ldots N$ depend on $x(k|k)$ and the future control signals $u(k+j|k)$, $j = 0 \ldots N-1$.

2. The set of future control signals is calculated by optimizing a criterion in order to keep the process as close as possible to a reference trajectory $r(k+j|k)$. The criterion usually takes the form of a quadratic function of errors between the predicted output signal and the predicted reference trajectory. The control effort is also included in the objective function in most cases. An explicit solution can be obtained if the criterion is quadratic, the model is linear and there are no constraints, otherwise a general optimization method must be used.

3. The control signal $u(k|k)$ is sent to the system while the control signals $u(k+j|k)$, $j = 1 \ldots N-1$ are rejected and step 1 is repeated with all the states brought up to date. Thus the $u(k+1|k+1)$ is calculated (which will be different from $u(k+1|k)$ since the horizon is moved forward and new information will be available).

The basic structure of MPC can be described as in Figure 2.1.
If the assumption that all states can be measured is made then the cost function to be minimized with respect to \( u \) often has the following form

\[
J(k) = \sum_{j=k}^{k+N-1} x^T(j|k)Qx(j|k) + u^T(j|k)Ru(j|k)
\]  

(2.9)

with positive definite matrices \( Q \) and \( R \). In most cases two horizons are used, output signal horizon and control signal horizon. The control signal horizon is chosen to be shorter and the cost function can then be written as

\[
J(k) = \sum_{j=k}^{k+N-1} x^T(j|k)Qx(j|k) + \sum_{j=k}^{k+M-1} u^T(j|k)Ru(j|k)
\]  

(2.10)

where \( N \) is the output signal horizon and \( M \) is the control signal horizon. It is important that a choice of \( u(k+M|k), ..., u(k+N-1|k) \) is made. The choice often made is that these control signals have the constant value \( u(k+M-1|k) \). But in the case when there is an integration in the system the remaining control signals are often chosen to be zero.

To summarize, the MPC problem can be defined as

\[
\min_u \sum_{j=k}^{k+N-1} x^T(j|k)Qx(j|k) + \sum_{j=k}^{k+M-1} u^T(j|k)Ru(j|k)
\]  

(2.11)

subject to

\[
u(k+j|k) \in U
\]  

(2.12)

\[
x(k+j+1|k) = Ax(k+j|k) + Bu(k+j|k)
\]  

(2.13)

or more generally

\[
x(k+j+1|k) = f(x(k+j|k), u(k+j|k))
\]  

(2.14)

As mentioned before if only Equations 2.11 and 2.13 are used, an explicit solution to the problem can be obtained. In the constrained or nonlinear case no such analytic solution exists and other methods, like quadratic programming or general nonlinear optimization, must be used.
3 The surrounding world

In this chapter the representation of the terrain and threats is covered. In order to successfully find a flight path it is necessary that accurate data about the surrounding world can be obtained. The surrounding world consists of natural objects like hills, rivers, lakes and forests, and man-made objects like houses, power lines and towers. Of course, the threats are also man-made obstacles but they must be treated in a different way compared to the rest of the surrounding world.

Section 3.1 covers the representation of terrain and describes the terrain database used in this thesis. In Section 3.2 the representation of threats used in this thesis is presented and some alternative ways are discussed.

3.1 Representation of terrain

The terrain data are stored in a large database available at Saab. The ground altitudes are sampled with 50 meter intervals and bilinear interpolation is used to get the values between these points. Since the interpolation is an approximation of the altitude there is no need for a higher resolution than 1 meter and that is the resolution used.

If the database samples are denoted \( h_{11}, h_{12}, h_{21}, h_{22} \) as in Figure 3.1, then the following operations are applied to get the ground altitude denoted \( h \) at the point of interest [1]

\[
\begin{align*}
    h_u &= h_{11} + \frac{d_1}{50}(h_{12} - h_{11}) \\
    h_l &= h_{11} + \frac{d_2}{50}(h_{21} - h_{11}) \\
    h_r &= h_{12} + \frac{d_2}{50}(h_{22} - h_{12}) \\
    h_d &= h_{21} + \frac{d_1}{50}(h_{22} - h_{21}) \\
    h &= \frac{1}{2}\left(h_l + \frac{d_1}{50}(h_r - h_l) + h_u + \frac{d_2}{50}(h_d - h_u)\right)
\end{align*}
\]

Figure 3.1: Illustration of ground altitude calculation.
Because only the ground altitude is stored in the database, an extra safety margin of 50 meters is used when altitude is calculated. As an example the Omberg hill near Vadstena is plotted in Figure 3.2 with data from the terrain database.

3.2 Representation of threats

There are a couple of characteristics that describe threats. They of course have a geographical position, but a radar station also has a cover area, and a missile launched from an anti-aircraft site has a range. The covered radar area and missile range depend on the surrounding terrain. If a radar is situated just under a hill, the area behind the hill is not seen by the radar and this is called the radar shadow, see Figure 3.3.

These radar shadows should be used when replanning is done because shorter paths can be obtained.
3.2.1 Straightforward representation of threats

Threats can be represented in several ways. One way could be storing the threats in a database similar to the terrain database, with data for each point in the three-dimensional space inside the threat range. This representation looks like an incomplete three-dimensional array. Storing every point is necessary because some points inside the range could be invisible for the radar because of the radar shadows. That is a very large amount of data if the resolution of the terrain database, which is 50 meters, is used. The good thing about this kind of representation is that it is quite fast to get information about a point in space. As an example, the typical radar has a range of 100 kilometers. For simplicity the covered area is assumed to have the shape of a half sphere with radius equal to the radar’s range. This half sphere must be quantified in order to store the points in the array. To calculate the approximate amount of points in the half sphere two rectangular blocks as in Figure 3.4 could be used to set the upper and lower bound on the number of points. The number of points can then be approximated to lie in between the number of points in each of the rectangular blocks.

As can be seen the large block has the dimension $4000 \times 4000 \times 2000$ points and the small block has the dimension (approximately) $2800 \times 2800 \times 1400$ points. The large block has $3.2 \cdot 10^{10}$ or 32 billion points, while the smaller block has $1.1 \cdot 10^{10}$ or 11 billion points. If the value in between these two values is used as an approximation then $2.2 \cdot 10^{10}$ or 22 billion points is obtained. If only one bit is used for every point then approximately 2.6 GB of storage must be used for each threat. That amount of data is not only impractical to store, but also the calculations to produce it are very demanding. Furthermore, sometimes it must be done during the flight in case that new threats are discovered. Therefore some other way of representing the threats must be found.

Another way of representing the threats is to store only the threat’s geographical position and its range. The amount of data stored here is very small, only four real numbers. It is also very fast and easy to add new threats during the flight. The drawback with this approach is that it is necessary to calculate if the plane lies in the line of sight during the replanning. That makes the execution of the replanning algorithm much slower. The advantages and the drawbacks of both methods are summarized in Table 3.1.
Because of the obvious problems with representing the threats with every point discussed above, the other way, only the geographical position, is chosen here.

3.2.2 Alternative representation of threats

An alternative representation of threats could be something called oct-trees. The oct-trees are well known data structures in the computer games world where they are used to fast determine the line of sight. The basic idea is to partition the space in 8 partitions and then to partition these into 8 partitions and so on. A tree with branching factor 8 is built in this way. Partitioning is stopped when each partition is small enough (usually some predetermined size) and these partitions are leaves in a tree. It is then calculated if each partition which is a leaf is visible from the threat. If all children of one node are visible then the whole partition represented by the node is visible, and there is no need for searching deeper in the tree. In the worst case scenario the search depth is equal to the tree depth which depends on the amount of partitions.

As an illustrative example a quad-tree will be used. The quad-trees are oct-trees counterparts in two dimensions. In Figure 3.5 a simple landscape and quad-tree partitioning is shown. Very thick lines represent first partitioning, thick lines second and thin lines third partitioning.

<table>
<thead>
<tr>
<th></th>
<th>Every point</th>
<th>Only position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage space</td>
<td>Large</td>
<td>Small</td>
</tr>
<tr>
<td>Calculation time</td>
<td>Fast</td>
<td>Slow</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison between threat representations.
If the point of interest lies in the middle of the landscape then the quad-tree representing the line of sight has the form as in Figure 3.6.

![Quad-tree of the landscape](image)

If we want to determine if some object that lies in, for example, partition (1,4) is visible to the object in the middle of the landscape the tree is traversed until the node marked (1,4) is found. As can be seen from the figure node (1,4) is visible. In this case it was necessary to traverse the tree to the maximum depth. But if the object lies in the partition (6,3), then the search stops in the node marked SW. This node is visible, so the depth in this case was only 1.

From the discussion above it can easily be determined what time and space complexity is for oct-trees representation. If we assume that partitioning is ended when the partitions are 50 × 50 × 25 meters then the depth of the tree can be calculated as

\[
d = \left\lceil \frac{\log 200000}{\log 50} \right\rceil = 12
\]

where \(\lceil \cdot \rceil\) is the ceiling operator. In the worst case, when the tree is complete, the amount of nodes is \(\sum_{i=0}^{12} 8^i \approx 7.85 \cdot 10^{10}\). This is very large but in the typical case all nodes are not visible and the amount of nodes is much smaller. The time complexity is equal to the tree depth which is 12 in the worst case. This shows that if the oct-trees are used to represent the threats the line of sight could be found very fast, but the storage needed is quite large in the worst case. For more information about oct- and quad-trees see [7].

---

The surrounding world

13
The surrounding world
4 Implementation

In this chapter the implementation of the planning algorithm will be covered. The whole algorithm can be divided into two parts. First part of the algorithm is the simulation of the MPC controller applied to a point-mass model and the preliminary path is the result from this step. This path however has too many waypoints, one for each step, and this amount must be reduced. This is the task of the second part of the algorithm called refining. The whole algorithm can be illustrated in Figure 4.1.

The state-space description of the system to be controlled is presented in Section 4.1. Necessary changes to the MPC algorithm will be covered in Section 4.2. The implementation of the entire algorithm is covered in Section 4.3.

4.1 State-space description

Some kind of system description is needed in order to implement the MPC algorithm. Therefore a state-space model of the system to be controlled has been made. One choice of model could be the real aircraft model. However this is not chosen here for the reason that refining of the path is applied after MPC. This means that the final path will not look like the path produced by the MPC anyway so the real airplane model is not needed. Instead a simple three dimensional point-mass model with constant speed is used. The point’s velocity vector’s direction can be controlled by two control signals, one for turning in the vertical plane and one for turning in the horizontal plane. This model has the following description

\[
\begin{align*}
\theta(k + 1) &= \theta(k) + \theta_0 u_\theta(k) \\
\varphi(k + 1) &= \varphi(k) + \varphi_0 u_\varphi(k) \\
x(k + 1) &= x(k) + s_0 \cos(\varphi(k) + \varphi_0 u_\varphi(k)) \cos(\theta(k) + \theta_0 u_\theta(k)) \\
y(k + 1) &= y(k) + s_0 \sin(\varphi(k) + \varphi_0 u_\varphi(k)) \cos(\theta(k) + \theta_0 u_\theta(k)) \\
z(k + 1) &= z(k) + s_0 \sin(\theta(k) + \theta_0 u_\theta(k))
\end{align*}
\]
where $\theta$ is the angle in vertical plane and $\Phi$ is the angle in horizontal plane and $x$, $y$, and $z$ are normal cartesian coordinates. The model is time discrete and $s_0$, $\theta_0$, and $\Phi_0$ are step sizes due to a quantification of the three dimensional space, see Figure 4.2.

$\theta$ and $\Phi$ are the control signals for vertical and horizontal plane respectively and where for that different magnitudes of the turn can be accomplished. The reason that $U$ is chosen in this way is discussed in Section 4.2. As can be seen the model is nonlinear and it follows the general form

\begin{align}
    x(k + 1) &= f(x(k), u(k)) \quad \text{(4.6)} \\
    u(k) &\in U \quad \text{(4.7)}
\end{align}

and it is assumed that all states can be measured. Since the analytic MPC solution cannot be obtained in this case, as stated in Section 2.3, some changes and simplifications are necessary to make implementation possible.

### 4.2 Modification of the MPC algorithm

Since the model is nonlinear and the general optimization algorithm would be too slow some kind of simplified brute force search must be used. In order to implement that kind of search the cost function must be calculated for all possible cases of the control signal and it must have a finite amount of values. To make the algorithm execution time relatively short under these circumstances, $U$ is chosen as in Section 4.1.

Furthermore the control signal horizon, $M$, is set to 1. This means that only the first control signal prediction $u(k \mid k)$ is optimized. The rest of the control signals $u(k + j \mid k), j = 1, \ldots, N-1$ are given the value 0 because of an integration in the system 4.1 - 4.5. In this way the cost function, $J(k)$, is evaluated along rays in a cone in three dimensional space, see Figure 4.3.
In order to speed up the algorithm even more, all combinations of the control signals are not applied, but only the combinations where \( u_0 \) and \( u_\theta \) have the same absolute value or some of the control signals are 0. If all the combinations are used then 49 different directions must be examined. If the simplification above is applied, only 25 directions are examined. For example combinations (1,1), (0,-1), (2,0) and (3,-3) are valid, while (1,2), (1,3), (-2,-3) are not.

The cost function has the following form

\[
J(k) = \sum_{j=k}^{k+N-1} g^T(j[k]) Q_1 g(j[k]) + Q_2 h(j[k])
\]  

(4.8)

where

\[
g(j[k]) = \begin{bmatrix} x(j[k]) - x_{end} \\ y(j[k]) - y_{end} \\ z(j[k]) - z_{end} \end{bmatrix}
\]  

(4.9)

\[
h(j[k]) = \begin{bmatrix} \text{terr}(x(j[k]), y(j[k]), z(j[k])) \\ \text{threat}(x(j[k]), y(j[k]), z(j[k])) \\ \text{height}(z(j[k])) \\ \text{fuel}(\theta(j[k])) \end{bmatrix}
\]  

(4.10)
terr(x,y,z), threat(x,y,z) and height(z) are functions that return 1 if the position determined by the states is under the terrain or in the threat cover area or above some height and 0 otherwise. The function fuel(θ(j|k)) reflects the fuel consumption according to a simple model

\[ fuel(\theta(j|k)) = s_0 e^{-\theta(j|k)} \]  

where \( c \) is some constant. \( x_{end}, y_{end} \) and \( z_{end} \) are the final values of the states and these have the same role as the reference signal \( r(k) \). In this case \( r(k) \) has constant value \([x_{end} y_{end} z_{end}]^T\) for all \( k \). \( Q_1 \) and \( Q_2 \) are the weight matrices and they are defined as

\[ Q_1 = \alpha I \]  
\[ Q_2 = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & \beta_4 \end{bmatrix} \]  

It can be noticed in Equation 4.8 that there is no control effort included in the cost function. The reason is that the controlled system is not a real system and the control signals will never be sent to the airplane. The result of the MPC step in the planning algorithm will just be passed to the refining step and the result from this step is a path similar to one made by a pilot. That is why there is no concern for the magnitude of the control signals.

### 4.3 Implementation of the planning algorithm

As mentioned before the planning algorithm consists of two parts, MPC and refining. In this section the implementation of both parts will be explained in more detail.

#### 4.3.1 MPC algorithm

The implementation of the MPC can be described with the following C-like pseudo code:

Input: \((x_{start}, y_{start}, z_{start})\) - start point  
\((x_{end}, y_{end}, z_{end})\) - end point (reference point)  
Output: vector with a preliminary path  
Variables: \(x(k)\) - state vector, \(u(k)\) - control signal

\[ \text{success} = \text{false}; \]  
\[ k = 1; \]  
\[ x(0) = (x_{start}, y_{start}, z_{start}); \]  
while(!success){
  \[ u(.|k) = \arg \min_u \ (4.8), \ subject \ to \ (4.6), (4.7); \]  
  \[ u(k) = u(k|k); \]  
  \[ x(k+1) = f(x(k), u(k)); \ // Equations 4.1 - 4.5 \]  
  put_into_path_vector(x(k));  
  \[ k = k+1; \]  
  if(end_criterion(x(k)))
    \[ \text{success} = \text{true}; \]
There are some things in the algorithm which must be clarified. The end criterion in
the algorithm above must be chosen in such a way that the algorithm terminates the
search after finite number of steps. For example, an end criteria based on an equality
test between the end point and the present state is not appropriate. That is because the
quantification of the space may prevent the states to be exactly equal to the end states,
and the algorithm will never terminate. A better criterion is to test if the distance
between the end point and the present states is sufficiently small.

The second thing that can be done is optimization of the cost function calculation.
Because all parts of the cost function are positive, it can only get a larger value after
each iteration. That is why the smallest value of the cost function is stored and the
calculation can be aborted if the present value gets larger than the stored value.

### 4.3.2 Refining of the flight path

This step in the planning algorithm is inspired by the work in [3]. The main purpose
of the refining is to reduce the number of the waypoints made by the MPC algorithm.
Because every point is stored after each step in the MPC algorithm the number of
points is very large. Assume that the number of waypoints is \( n \) and these are stored in
a vector. The idea with the refining algorithm is very simple, just check for the line of
sight between point 1 and point \( n \). If there is a line of sight then that path is at least as
short as the MPC path if not shorter and we are done. If there is no line of sight then
check for it between point 1 and point \( n-1 \). The line of sight is defined as the straight
line between two points which does not pass through terrain or threats. This is then
repeated until a line of sight is found between point 1 and point \( p \). Then the procedure
is repeated with point \( p \) as the first point and point \( n \) as the last one. For illustration
see Figure 4.4.
The refinement algorithm can be implemented in the following way:

**Input:** \texttt{path\_vec} - vector with a MPC path  
**Output:** vector with a refined path  
**Variables:** \texttt{beg}, \texttt{end} - references to the vector elements

\begin{verbatim}
beg = 0;
end = size(path_vec)-1
put_into_vector(path_vec[beg]);
success = false;
while(!success){
    if(line_of_sight(path_vec[beg],path_vec[end])){
        put_into_vector(path_vec[end]);
        beg = end;
        end = size(path_vec)-1;
        if(beg == end)  \  // Algorithm successfully came to the  
        \  // end
        success = true;
    }
    else {
        end--;  \  // No line of sight found
        if(beg == end){  \  // Moving reference forward
            beg++;  
            put_into_vector(path_vec[beg]);
        }  
    }
}
\end{verbatim}

4.3.3 Implementation of the approach bearing

As stated in Section 1.1 it is sometimes necessary that the target is approached from some particular direction (called bearing) and from some distance. This need might come up if the target can only be reached from some direction, a typical case is when the airplane is heading to a landing runway.

![Figure 4.5: Approach bearing.](image-url)
The solution to this problem might be to include bearing in the end states. The weight factor on this state should then increase with decreasing distance to the end point. This is done to prevent that the algorithm tries to get into bearing to early and risks to enter into the threat or terrain. This solution is not tested and instead a simpler one is implemented. Since the bearing and the distance to the target or the runway always are known in advance, the point which lies in the bearing direction and on the known distance can easily be calculated. Then this new point is chosen to be the end point and the planning algorithm is applied. When it is finished the real target point is added to the final path. The only thing that has to be controlled is that this straight line to the target point is not passing through terrain or threats. As an example see Figure 4.5 where bearing to the target is set to north east or 45 degrees and distance 5 km.

The implementation of the approach bearing can be described with the following pseudo code:

```plaintext
Input: bearing - bearing to the target (in radians),
      d - distance to the target (in km)
Output: (xend_new, yend_new, zend_new) - new endpoint
Variables: phi - angle in the horizontal plane,
           th - angle in the vertical plane

th = 0;
phi = bearing - pi;
xend_new = d*cos(phi)*cos(th);
yend_new = d*sin(phi)*cos(th);
while(!line_of_sight(new_endpoint,target_point)){
    th = th + pi/20;
xend_new = d*cos(phi)*cos(th);
yend_new = d*sin(phi)*cos(th);
    zend_new = d*sin(th);
}
```
Implementation
5 Results

In this chapter the results will be presented. Different scenarios will be studied, using different tuning variables such as the weight factors in the cost function and prediction horizon. Position, range and the amount of threats will also be varied. The terrain that will be used is the real data from the terrain database, see Chapter 3.

5.1 The terrain

The choice of a good terrain for testing the planning system is crucial. If the terrain is too flat then there is no challenge for the system. That means that height variation in the terrain is something to look for. For that reason two parts of Sweden are chosen, one is the landscape around the lake Vättern in the middle of Sweden, see Figure 5.1, and the second is an area near Östersund in the north of Sweden, see Figure 5.2. The distance between points in the figures is 250 meters. This gives that the square area around Vättern is 50 by 50 km large, while the area around Östersund is 100 by 100 km.

Figure 5.1: The area around lake Vättern. The Omberg hill can be seen in the lower right corner.
5.2 Simulation results

In this section the results from different scenarios will be presented. The scenarios will show how different settings on the parameters influence the paths that are found. The parameters that can be varied are shown in Table 5.1.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Symbol</th>
<th>Parameter Influence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction horizon</td>
<td>$N$</td>
<td>Path, Execution time</td>
</tr>
<tr>
<td>Vertical angle</td>
<td>$\theta_0$</td>
<td>Path</td>
</tr>
<tr>
<td>Horizontal angle</td>
<td>$\varphi_0$</td>
<td>Path</td>
</tr>
<tr>
<td>Step size</td>
<td>$s_0$</td>
<td>Path, Execution time</td>
</tr>
<tr>
<td>Weight matrix</td>
<td>$Q_1$</td>
<td>Path</td>
</tr>
<tr>
<td>Weight vector</td>
<td>$Q_2$</td>
<td>Path</td>
</tr>
</tbody>
</table>

Table 5.1: Parameters that can be varied.
For more information about the weight factors $Q_1$ and $Q_2$ see Section 4.2.

The threat cover areas are represented with circles in the figures. The radar shadows discussed in Section 3.2 are also present but since it is quite difficult to visualize them this has not been done. In all MATLAB plots the distance between points is 250 meters.

5.2.1 Variation of the prediction horizon

In this section the prediction horizon, $N$, will be varied and the consequences for the path will be shown. Other parameters are chosen to have constant value. The values of the parameters used in this simulation are (all symbols are explained in Section 4.2):

$\alpha = 1$

\[ \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & \beta_4 \end{bmatrix} = \begin{bmatrix} 5000 \cdot w & 1000 \cdot w & w & 5 \end{bmatrix} \]

$w =$ Euclidian distance between start position and end position

$c = 34$

$\theta_0 = \frac{\pi}{24}$

$\varphi_0 = \frac{\pi}{12}$

$s_0 = 250 \text{ m}$

The parameter $w$ is used to prevent that in long distance paths the first part of the cost function weighted with $Q_1$ has more influence than the second part weighted with $Q_2$. Also the maximum recommended height is set to 600 meters. Two different values of the prediction horizon are considered, 20 and 60. That gives with $s_0 = 250 \text{ m}$ prediction horizons of 5 and 15 km respectively. In Figure 5.3 the path created with $N = 20$ is shown. The start point is in the upper right corner and the end point is in the lower left corner.
Figure 5.3: Path with N = 20.

As can be seen the short prediction horizon "sees" the smallest threat too late and generates an unnecessary turn. In Figure 5.4 N = 60 is used and the path is shorter because the smallest threat could be seen earlier.

Figure 5.4: Path with N = 60.
Results

On the other hand the execution time for the case where $N = 20$ is 2 seconds on SUN Ultra Sparc 10 while it is 4 seconds when $N = 60$. In this case a compromise between path length and execution time must be done.

Another example where the short prediction horizon affects the path is shown in Figure 5.5. The start point is in the upper right corner and the end point is in the lower left. $N$ is too short and the algorithm cannot see the exit from the threat barrier and chooses to go through them.

![Figure 5.5: Path with $N = 40$. $N = 60$ finds the safe path.](image)

### 5.2.2 Variation of the step size

Variation of the step size, $s_0$, influences both path form and execution time just as variation of the prediction horizon does. Three step sizes will be considered, 50, 60 and 250 m. The other parameters are kept the same as in the previous example:

- $\alpha = 1$
- $\beta = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & \beta_4 \end{bmatrix} = \begin{bmatrix} 5000 \cdot w & 1000 \cdot w \cdot w \cdot w \cdot s \end{bmatrix}$
- $w$ = Euclidian distance between start position and end position
- $c = 34$
- $\theta_0 = \frac{\pi}{24}$
- $\varphi_0 = \frac{\pi}{12}$
$N = 60$

It can be seen in Figure 5.6 that $s_0 = 50$ m gives the same path as $s_0 = 250$ m shown in Figure 5.7, but execution time is 86 seconds when $s_0 = 50$ m compared with 7 seconds when $s_0 = 250$ m. Starting point is in the lower left corner and ending is in the upper right.

*Figure 5.6: Path with $s_0 = 50$ m.*
In Figure 5.8 $s_0 = 60$ m is used and the path differs from the one with step sizes 50 and 250 m. Execution time in this case was 27 seconds and the path is longer because prediction horizon can only see 3.6 km ahead and the same problem arises as in the case when $N$ is small.
5.2.3 Variation of the horizontal angle

Variation of the horizontal angle influences the path form, but in some cases also the execution time might differ. The explanation for that is the quantification of the angle. Different angle sizes can cause that a longer path must be taken around threats or terrain and that influences the execution time.

Three different horizontal angle sizes will be used, $\pi/5$, $\pi/12$ and $\pi/40$. Other parameters are chosen as in previous sections. In Figure 5.9 the path with $\phi_0 = \pi/5$ is shown. This can be compared with Figure 5.10 where $\phi_0 = \pi/12$. The start point in both cases has coordinates (60,1) and the end point has the coordinates (80,199).

Figure 5.9: Path with $\phi_0 = \pi/5$. 
In the first case the angle is large enough to find the path between two threats, while in the second the longer path has to be taken. If the comparison is made to Figure 5.11 where $\phi_0 = \pi/40$, then it can be seen that in this case the angle is too small to discover the way to the right of the large threat.
In some other cases the situation is different. In Figure 5.12 the area around Östersund is used, no threats are placed and \( \phi_0 = \pi/5 \) is used. The maximum recommended height is set to 900 meters. The start point is in (1,320) and end is in (240,1). This terrain has much more variation and narrow passages between the hills. If this path is compared to the path in Figure 5.13 where \( \phi_0 = \pi/12 \), then it is obvious that the path in Figure 5.13 is shorter.

Figure 5.12: Path with \( \phi_0 = \pi/5 \).

Figure 5.13: Path with \( \phi_0 = \pi/12 \).
If $\phi_0 = \pi/40$ is used instead then the path as in Figure 5.14 is obtained. This path is very similar to the one in Figure 5.13, but it should be pointed out that the height constraint which is set to 900 m is violated in this case. The waypoint marked with an arrow has the height of 1060 m.

The execution times differ too. If $\phi_0 = \pi/40$ execution time is 14 seconds, while it is 5 seconds if $\phi_0 = \pi/5$ or $\phi_0 = \pi/12$.

5.2.4 Variation of the vertical angle

Just like the horizontal angle, the vertical angle influences the path form, but also the execution time. The reasons for that are the same as for the horizontal angle. As before three different values of the angle will be used, $\pi/10$, $\pi/24$ and $\pi/35$.

In Figure 5.15 $\theta_0 = \pi/10$ is used. Compared to Figure 5.16, where $\theta_0 = \pi/24$, the path is longer. The start point is in the upper right corner and the end point is in the lower left. The reason is that in the first case the vertical angle is too large and because of the fuel consumption factor in the cost function no altitude changes are made. That is why the path must round the hill marked with an arrow. In the second case the angle is small enough to let the path come up to the altitude where the terrain is not a problem, and a shorter path can be obtained.
Results

Figure 5.15: Path with $\theta_0 = \pi/10$.

$\theta_0 = \pi/10$.

Figure 5.16: Path with $\theta_0 = \pi/24$.

$\theta_0 = \pi/24$. 
If $\theta_0 = \pi/35$ is used, the path is almost identical as the path when $\theta_0 = \pi/24$, as can be seen in Figure 5.17. The only difference is that the path with $\theta_0 = \pi/35$ comes up to the height constraint which is 600 m, while the path with $\theta_0 = \pi/24$ does not. That can be explained with the quantification of the angles.

**5.2.5 Variation of the weight matrices**

In this section the weight matrices, $Q_1$ and $Q_2$, which have a quite large influence on path form, will be varied. It is mainly these parameters that should be adjusted in order to get good flight paths. For example if the factor $\alpha$ is increased then the system will try to produce as short path as possible, but the path might pass through the threat areas or even the terrain. Therefore it is crucial to find a good balance in the choice of the weight matrices.

Three different values of $\alpha$ will be used, 0.1, 1 and 30. Other parameters will be set to the values as in the sections above. The path with $\alpha = 1$ is shown in Figure 5.18. If that is compared with the path in Figure 5.19 where $\alpha = 30$, it is clearly seen that a larger $\alpha$ gives more importance to the path length than to the threat avoidance.
Figure 5.18: Path with $\alpha = 1$.

Figure 5.19: Path with $\alpha = 30$. 
In both cases the start point is in (180,170) and the end point is in (1,1).

It should be pointed out that for $\alpha = 0.1$ no path was found. The reason was probably that other factors in the cost function than distance to the end point are more important and the end criteria is never satisfied.

In Figure 5.20 $\alpha = 1$ and in Figure 5.21 $\alpha = 0.1$ are used. No threats are present. The start point is in (1,350) and the end point is in (240,1). The clear difference in path length can be noticed from these two figures.

![Figure 5.20: Path with $\alpha = 1$.](image_url)
Another parameter that can be varied is $Q_2$. This vector consists of four parameters, $\beta_1, \beta_2, \beta_3$ and $\beta_4$, which are the weight factors on the terrain, threats, altitude and fuel consumption. In order to present the influence of these parameters on the path several cases will be considered.

The first factor that will be varied is $\beta_1$ which decides if the terrain will be avoided and how much. Two values of the factor will be chosen, $5000 \cdot w$ and $10 \cdot w$. $w$ is, as before, the Euclidian distance between the start and end point. It is used to prevent that in long distance paths the distance has more influence on the path than the threats or the terrain. Other factors have the same values as before. In Figure 5.22 $\beta_1 = 10 \cdot w$ is used. The start point is in the upper right corner and the end point is in the lower left corner. If that figure is compared to Figure 5.23 where $\beta_1 = 5000 \cdot w$ is used, an obvious difference can be noticed. In the case of low weight on the terrain the algorithm chooses to go through the terrain instead of around the threats. That is why it is very important to choose this weight factor very high in order to always avoid the terrain.
Results

Figure 5.22: Path with $\beta_1 = 10w$.

Figure 5.23: Path with $\beta_1 = 5000w$. 

39
The next parameter to be varied, $\beta_2$, influences if the algorithm should consider the threats as dangerous or not. As with $\beta_1$, two values are chosen $1000 \cdot w$ and $10 \cdot w$ and $w$ is defined as before. Other parameters are fixed as above. In Figure 5.24 $\beta_2 = 10 \cdot w$. It can be seen that the path goes right through the large threat and not around it as in Figure 5.23 where $\beta_2 = 1000 \cdot w$.

![Figure 5.24: Path with $\beta_2 = 10w$.](image)

The constraint on the altitude is controlled by the parameter $\beta_3$. This constraint is not so hard as the constraint on the terrain, and is introduced because some missions have maximum recommended flight altitude. Two values of the parameter will be used $w$ and $0.01 \cdot w$. The path obtained with $\beta_3 = w$ is the same as in Figure 5.23 and the threat is rounded on the maximum recommended altitude of 500 m. If $\beta_3 = 0.01 \cdot w$ is used then the path is very similar, but the threat is rounded on the altitude of 2100 m which is approximately 4 times larger than recommended. In this case the algorithm finds a shorter path since the diameter of the threat is smaller on higher altitudes.

The last parameter, $\beta_4$, influences fuel consumption impact on the path. Two values will be used 5 and 10. In Figure 5.25 the path with $\beta_4 = 10$ is shown and in Figure 5.26 $\beta_4 = 5$ is shown.
Results

Figure 5.25: Path with $\beta_4 = 10$. Max altitude 280 m.

Figure 5.26: Path with $\beta_4 = 5$. Max altitude 500 m.
The path with the larger $\beta_4$ is longer because the algorithm does not change the altitude and it must take the detour around the terrain. The smaller value allows the algorithm to find the path over the terrain and obtain a shorter path.

5.2.6 Variation of the amount of threats

The amount of threats affects mainly the execution time of the algorithm. Since the threats are represented as described in Section 3.2 it must be the calculated if each threat can see the point in every step. In this way the time complexity will grow with the amount of threats. As an illustration the configuration in Figure 5.27 is taken and the path is shown. The same path is obtained if some threats are removed, but calculation time differ. If all three threats are present the calculation time is 4 seconds. If two threats are present, the time is 3 seconds. If only one threat is present the the path is found in 2 seconds, and the same time is obtained if no threats exist.

![Figure 5.27: Configuration and path example for variation of the amount of threats.](image)

This shows that if the amount of threats is too large then the calculation time might become too long, and some better representation of the threats is needed.

5.2.7 Approach bearing

In Figure 5.28 the bearing is 45 degrees and the distance to the target point is 10 km. The start point is in (1,199) and the target point is in (180,40). It can be seen that the Omberg hill is flown over in order to come into right position.
5.2.8 Radar shadows

Although the radar shadows existed in previous examples they were not very useful for the algorithm. In Figure 5.29 it is shown how the radar shadow is used to get a shorter path. The hills on the Vättern coast are used to hide behind since this area is not seen by the radar.
Results

Figure 5.29: Exploiting radar shadows.
6 Conclusions

Previous work in this area used methods which are closer to the area of artificial intelligence like A*-search and greedy search. The problems that arose in those cases is that the search always was limited to two dimensions. The reason is that time complexity increases when one more dimension is introduced. That is why a new method, that supports relatively fast search in all three dimensions, is tested in this master thesis. For further reading about A*-search see [3], [4] and [6].

In this master thesis the following factors are paid attention to when planning is done:

- **Terrain**: The database of the real terrain in Sweden is used to give as realistic paths as possible.
- **Threats**: The threats are represented as static half spheres and radar shadows are taken into account.
- **Maximum recommended altitude**: The maximum recommended altitude is introduced because some missions require it.
- **Approach bearing**: The approach bearing is also required in some missions and is implemented in the planning system.
- **Fuel consumption**: A simple model for the fuel consumption is a part of the cost function and therefore is minimized during calculations in each step.

MPC, with simplifications described in Section 4.2, and the path refining algorithms solved the planning problem stated in Chapter 1. However it is not achieved without any drawbacks. From the results presented in Chapter 5 it can clearly be seen that there are many variables which influence the outcome of the planning system. For example the result is very dependent on the prediction horizon length and long horizons increase the execution time. The execution time could be a problem if the amount of threats is large, since the threat representation problem has not been solved perfectly. That is why a representation that can establish a line of sight very fast is important to find and some ideas can be found in Section 3.2. Another example is the horizontal angle which in some cases should be large in order to find shorter paths, while it should be small in some others, see Section 5.2.3. This shows that it is difficult to find an optimal configuration of all variables suitable for all possible cases. This problem could be solved by adapting the variables until the path is satisfactory according to some criteria such as path length or maximum height. The drawback to this approach is the time consumption which would increase because the algorithm must be executed more times.
Conclusions

MPC is deterministic, meaning that the path form can be predicted if all initial states, such as start and end points, are known. This is very important for the safety and reliability of the system.

As a final word it can be concluded that despite drawbacks the MPC method used here is a very promising method and further development of it is encouraged.
7 Future work

In this chapter some ideas about further development of the mission planning system will be presented.

7.1 Threat representation

As stated before, a good threat representation is important for fast execution of the algorithm. Some ideas that could be implemented are given in Section 3.2 and it would be interesting to see if this can improve the performance of the system.

7.2 Dynamic threats

In this master thesis only static threats are used. Despite that many of the threats are static, such as radar stations and ground-to-air missile sites, there are some moving or dynamic threats. Enemy aircrafts, mobile missile sites or even extreme weather are examples of such threats. The implementation of dynamic threats should not be a problem and could be done in the following way. All dynamic threats are initiated with their start position and moving direction. Even the speed of the threat could be included or could have some default value. Then the threats positions are updated in every step. An alternative is that the planning system gets information about the threats from the airplane’s radar readings in each step.

7.3 Fuel consumption

A very simple model of the fuel consumption is used in this thesis. An improvement of the model to reflect the fuel consumption more realistically is one thinkable alternative.

Another thing that could be done is the control of the consumption in the produced path. If the amount of fuel is not enough, then the weight matrices $Q_1$ and $Q_2$ in the cost function could be changed in order to produce a different path. This path might go through the threats if that means that fuel would be saved. In order to do this a more realistic consumption model is needed.

7.4 Using a linear model

A nonlinear model is used to implement the MPC algorithm in this thesis. The drawback of this approach is that a brute force search has to be applied. An alternative could be that a linearization of the model is done in each step. Since fast algorithms for MPC with linear models exist those could be applied and a better solution could be obtained. If the linearization is done then the quantization on the control signal could be relaxed and would have the form $u_{min} \leq a \leq u_{max}$. Because the algorithms for solving the MPC problem often are written in matrix form, the support for those operations should be implemented in C++. With object oriented methods this should not be a problem.

7.5 Approach bearing

As mentioned in Section 5.2.7 one way of implementing the approach bearing could be that it is included as a state just like the coordinates of the end point. Then the weight on this state could be increased with shrinking distance to the end point. In
this way the demanded bearing in the end point could be obtained without any sharp
turns that are generated now.
References

[1] Stefan Ahlqvist
Development, implementation and validation of software for aircraft navigation simulation
Master thesis report LiTH-ISY-EX-3018
Department of Electrical Engineering
Linköping University, 1999

[2] Eduardo F. Camacho, Carlos Bordons
Model Predictive Control
ISBN 3-540-76241-8
Springer-Verlag Berlin Heidelberg New York, 1999

[3] Laban Källgren
Real-time replanning of mission routes based upon threats
Master thesis report LiTH-ISY-EX-3126
Department of Electrical Engineering
Linköping University, 2001

[4] Anna Lindahl
Omplanering av uppdrag i realtid - beslutsstödsmetoder för JAS 39 Gripen
Master thesis report LiTH-IDA-EX-99/33
Department of Computer and Information Science
Linköping University, 1999

Linear Model Predictive Control, Stability and Robustness
ISBN 91-7219-920-2
Department of Electrical Engineering
Linköping University, 2001

Search problems in mission planning and navigation of autonomous aircraft
Master thesis report
School of Aeronautics
Purdue University Indiana USA, 1988

Last accessed 17th of October 2001
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Uphovsmannens ideella rätt innefattar rätt att bli nämd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Zoran Sjanic