ISSN 0280-5316
ISRN LUTFD2/TFRT--5727--SE

Information-Theoretic Approach for Path Planning of a Moving Platform with Bearings-only Sensor

Rickard Björström

Department of Automatic Control
Lund Institute of Technology
October 2004

Department of Automatic Control **Lund Institute of Technology** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER THESIS *Date of issue* October 2004 *Document Number* ISRNLUTFD2/TFRT--5727--SE
Author(s) Rickard Björström	*Supervisor* Per Hagander LTH, Lund Per Skoglar FOI, Linköping *Sponsoring organization*

Title and subtitle
Information-Theoretic Approach for Path Planning of a Moving Platform with Bearings-only Sensor.
(Informationsteoretisk ruttplanering för en rörlig plattform med passiv sensor)

Abstract
Using flying vehicles for reconnaissance and surveillance has always been interesting, especially in military applications. Unmanned aerial vehicles (UAVs) have been increasingly used in the last decades, but they have often been controlled by an operator on the ground. In an attempt towards higher level of autonomy, the UAV should be able to decide itself where to fly.
This thesis examines a method for autonomous path planning based on the uncertainty of the target locations, a so called information-theoretic approach. A bearings-only sensor is attached to the UAV, such as a video or an infrared sensor, which makes observations of the relative angle to the object, reducing the uncertainty orthogonal to the observed target. The planned path is the solution to an optimization problem, such that the uncertainty is minimized which is equal to maximizing the information in the information-theoretic approach.
When identifying a target, there is a potential benefit to make observations from different views. If the path could be planned, the better are the observations of the target, and the image based identification will be more reliable to target appearance variations and more robust against decoys.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 65	*Recipient's notes*
Security classification		

Acknowledgements

This master's thesis was carried out at the department of Sensor Technology at FOI, the Swedish Defense Research Agency, in Linköping. This thesis finishes my education in Engineering Physics at Lund Institute of Technology, Lund University.

First of all, I would like to thank my supervisors Per Skoglar and Jonas Nygårds for the confidence in me doing this thesis. They have given support throughout the time and provided me with some great ideas. I would also like to thank Morgan Ulvklo for the opportunity of doing my thesis at FOI and the *SIREOS* group for following the development and visiting the test launch of their airborne remote sensing platform, where my work may be implemented later.

Linköping, October 2004

Contents

Nomenclature

Notations

$\mathbf{f}(\cdot)$	Non-linear state transition model (System equations)
$\mathbf{F}$	Linearized state transition matrix
$\mathbf{h}(\cdot)$	Non-linear observation model
$\mathbf{H}$	Linearized observation matrix
$\mathbf{I}(\cdot)$	Observed information
m	Number of parameterized control signal
$P(\cdot)$	Probability distribution
$\mathbf{P}(\cdot)$	Covariance matrix
$\mathbf{Q}$	Process noise covariance
$\mathbf{R}$	Observation noise covariance
t_f	Time horizon in optimization
$\mathbf{u}$	Control input vector
$\mathbf{v}, \nu$	Observation noise
$\mathbf{w}, \omega$	Process noise
$\mathbf{x}$	State vector
$\mathbf{Y}(\cdot)$	Information matrix
$\mathbf{z}$	Observation vector

Abbreviations

EO	Electro Optical
CCD	Charge Coupled Device
FOI	Totalförsvarets Forskningsinstitut (Swedish Defense Research Agency)
FRD	Forward-Right-Down
INS	Inertial Navigation System
GPS	Global Positioning System
IR	Infrared
SIREOS	Signal Processing for Moving EO/IR Sensors
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle

Chapter 1

Introduction

This master's thesis examines how to plan the flight path for an unmanned aerial vehicle (UAV). In this introduction, the task of the UAV is presented as well as how information is used in this thesis.

1.1 The *SIREOS* Project

This thesis is a small part of the *SIREOS* (Sensor Processing for Moving EO/IR Sensors) group at FOI where several algorithms are developed for analysis of electro optical and infrared image sequences from moving platforms like UAVs. The designated task is surveillance and reconnaissance. Many problems must be considered like robust navigation, collision avoidance, image motion computation, route and viewpoint planning, scene topography estimation etc. These tasks are improved by sensor management, which are currently under development. [8]

In order to help the image processing algorithms by collecting useful images, the path of the moving platform must be planned. If a target is about to be identified by the methods of image processing, there is a need to see the objects from different field of views. In other words, the ability to identify a target in an image is depending from where the image was taken. This means, how "good" the images are, is depending on the distance and the angle to the target. If the path of the moving vehicle could be planned, the targets could be identified more reliable. An example of how this would look like is in Figure 1.1, where the UAV collects images from different positions.

1.2 Information

Throughout this thesis, the task of the UAV is to identify different objects on the ground. When an object is said to be identified, there are sufficient number of pictures of the object that it could be identified with the methods from image analysis. To represent the ability to identify targets, the concept of information is introduced in the sense "the more information about an object, the more likely is the identification". By collecting images from the target, the information about the target increases and when a sufficient amount of images are collected, the target is said to be identified. To express this a bit sloppy:

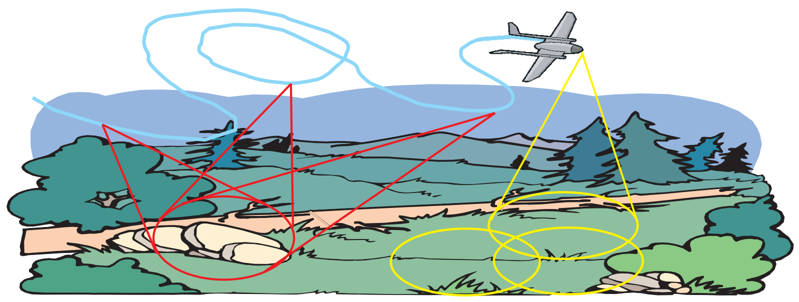

Figure 1.1: UAV surveillance and reconnaissance. [8]

An object is identified when a certain level of information about the object is reached.

There are no real images in this thesis, instead the amount of information is connected to the uncertainty of the target's geometrical location. The term "identify an object" would now be that the uncertainty of the target's location is lower than a pre-defined threshold value. The uncertainty is represented by a covariance matrix, and as will be shown later on, the information matrix is the inverse of the covariance matrix for normal distributions. The path is planned such that the uncertainties of the targets are minimized, which is the same as maximizing the information and the resulting path is the best in terms of information at the optimization horizon.

1.3 Objective

The objective of the thesis is to investigate if the information-theoretic approach is suitable method for path planning of moving platforms.

1.4 Outline

The outline of the thesis is:

- Chapter 2 introduces the theory of information and presents the optimal control problem.
- Chapter 3 is a path planning example for a UAV and shows the solution to the optimal control problem.
- Chapter 4 is the extension to a full area exploration and there are several objects dynamically detected in the area and the planned path must simultaneously consider both the object geolocation and the area coverage.
- Chapter 5 introduces another method called spline optimization. Instead of finding optimal control signals, one could find the optimal trajectory directly by placing points who together creates a spline.

- Chapter 6 complicates the model with a camera, and objects can only be seen if they are visible by the camera and the path consists of both the vehicle trajectory and the camera directions.
- Chapter 7 summarizes the result and some suggestions for future work are given.

1.5 References and Further Readings

The path planning problem as an optimal control problem is formulated and solved in Grocholsky [4]. The definitions of information and the examples are from Grocholsky as well. A more detailed description of information and information filter is found in Manyika and Durrant-Whyte [5]. The solution of the optimal control problem by parametrization of the control signal is described in Bertsekas [1]. The gimballed camera used in the *SIREOS* project is described in Skoglar [8].

Chapter 2

The Use of Information in Optimal Control

In this chapter, the optimal control problem based on the information criteria is formulated. First the concept of information is introduced and how it is used in optimal control, and this results in a utility function from information theory, which should be optimized in the optimal control problem.

2.1 Information measures

Uncertainties of states and observations are represented by probability distributions and the concept of information is introduced as a measure of how much "information" is contained in such distributions. There are two formal definitions of information, the Entropic information and Fisher information.

2.1.1 Entropic Information

Entropic information is defined from entropy. The entropy or Shannon information $H(\mathbf{x})$ associated with a probability distribution $P(\mathbf{x})$, where $\mathbf{x}$ is a random variable, is defined in Grocholsky [4] as

$$H(\mathbf{x}) \equiv -E\{\log P(\mathbf{x})\} = -\int_{-\infty}^{\infty} P(\mathbf{x}) \log P(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{2.1}$$

That is, the entropy is defined as the negative of the expected value of the log-likelihood. Entropic information $i(\mathbf{x})$ is then defined as the negative of the entropy, so information is maximized when entropy is minimized

$$i(\mathbf{x}) = -H(\mathbf{x}). \tag{2.2}$$

When the probability distribution modeling an n-dimensional state $\mathbf{x}$ is Gaussian distributed with mean $\bar{x}$ and covariance $\mathbf{P}$, it could be shown, as in Grocholsky [4], that the entropic information becomes

$$i(\mathbf{x}) = -H(\mathbf{x}) = -\frac{1}{2} \log[(2\pi e)^n \mid \mathbf{P} \mid]. \tag{2.3}$$

2.1.2 Fisher Information

The second measure of information is the Fisher information. It is only defined on continuous distributions unlike entropy. Fisher information gives a measure of the amount of information about $\mathbf{x}$ given observations $\mathbf{Z}^k$ up to time k, in the probability distribution $P(\mathbf{Z}^k, \mathbf{x})$. Like entropy, Fisher information is derived from the log-likelihood. In Manyika and Durrant-Whyte [5] a *score function* s is introduced as

$$s(\mathbf{Z}^k, \mathbf{x}) \equiv \nabla_x \log P(\mathbf{Z}^k, \mathbf{x}) = \frac{\nabla_x P(\mathbf{Z}^k, \mathbf{x})}{P(\mathbf{Z}^k, \mathbf{x})}. \tag{2.4}$$

Then the *Fisher information matrix* is defined as the covariance of the score function as

$$\begin{aligned} \mathcal{J}(k) &\equiv \mathrm{E}\{\nabla_x \ \log \ P(\mathbf{Z}^k, \mathbf{x}) \ (\nabla_x \ \log P(\mathbf{Z}^k, \mathbf{x}))^T\} \\ &= -\mathrm{E}\{\nabla_\mathbf{x} \nabla_\mathbf{x}^T \log P(\mathbf{Z}^k, \mathbf{x})\}. \end{aligned} \tag{2.5}$$

In Grocholsky [4], the definition of Fisher information matrix $\mathcal{J}(\mathbf{x})$ is simplified from (2.5), with $P(\mathbf{Z}^k, \mathbf{x}) = P(\mathbf{x})$ as

$$\mathcal{J}(\mathbf{x}) \equiv -\frac{\mathrm{d}^2}{\mathrm{d}\mathbf{x}^2} \log P(\mathbf{x}). \tag{2.6}$$

Consider again the case when $P(\mathbf{x})$ is a Gaussian distribution, taking logarithm and differentiating twice with respect to $\mathbf{x}$ gives $\mathcal{J}(\mathbf{x}) = \mathbf{P}^{-1}$, the Fisher information becomes simply the inverse of the covariance. The relationship between entropy and Fisher information can be explicit shown for a Gaussian distribution, where n is the dimension of the state, as

$$i(\mathbf{x}) = -\frac{1}{2} \log[(2\pi e)^n \mid \mathbf{P} \mid] = \frac{1}{2} \log[(2\pi e)^{-n} \mid \mathcal{J}(\mathbf{x}) \mid]. \tag{2.7}$$

2.1.3 Information Matrix

Let $\mathbf{x} = \mathbf{x}(t)$ and

$$\mathbf{Y} = \mathbf{Y}(t) = \mathcal{J}(\mathbf{x}(\mathbf{t})) \tag{2.8}$$

be the *information matrix* throughout the work. From (2.7), information $i(\mathbf{x})$ is maximized when the determinant of the information matrix $\mathbf{Y}$ is maximized. The determinant gives a scalar value, which is needed for comparison. The determinant is an example of a utility function, and it is this function which will be optimized in the optimal control problem.

According to Grocholsky [4], the scalar value must combine or weight the elements or eigenvalues of the information matrix or its inverse. Let $\{\lambda_1, \ldots, \lambda_n\}$ be the eigenvalues of the information matrix and the determinant is related as

$$|\mathbf{Y}| \ = \ \prod_{i=1}^{n} \lambda_i. \tag{2.9}$$

The determinant of the information matrix is a candidate for utility function and the method is often referred to, simply as "the entropy". According to Grocholsky [4], entropy is the most appropriate measure of information contained in a probability function, but it is interesting to examine alternative measures and hence different candidates of utility functions in order to see their solutions to the path planning problem. Two alternatives to maximize entropic information are:

$$\begin{array}{lll} 1. & \max & \text{trace}\,(\mathbf{Y}) \;\; \text{and} \\ 2. & \min & \text{trace}\,(\mathbf{Y}^{-1}). \end{array}$$

Or expressed in terms of eigenvalues

$$\begin{aligned} \text{trace}\,(\mathbf{Y}) &= \sum_{i=1}^{n} \lambda_i, && (2.10) \\ \text{trace}\,(\mathbf{Y}^{-1}) &= \sum_{i=1}^{n} \frac{1}{\lambda_i}. && (2.11) \end{aligned}$$

As can be seen from (2.9) to (2.11), there are drawbacks. A poorly scaled information matrix could give high information even when some eigenvalues are small in comparison. To avoid such problems, the utility function could operate on the eigenvalues directly like in [4], with

$$\max(\min \operatorname{eig}(\mathbf{Y})), \tag{2.12}$$

as the most direct one. There are also other alternatives and for details see Grocholsky [4]. These four different utility functions discussed are simulated later in Section 3.2.4.

The eigenvalues of the information matrix are always positive, since the information matrix is the inverse of the covariance matrix and the eigenvalues of a covariance matrix represent covariances, which are quadratic. Therefore are the eigenvalues of the information matrix inverse quadratic and still positive.

2.2 Information Evolution

2.2.1 Observed Information

The information matrix $\mathbf{Y}(t)$ is updated by observations of objects. Let the states to be observed be the vector $\mathbf{x}(t)$ and the observations are in the observation vector $\mathbf{z}(t)$. The observation $\mathbf{z}(t)$ is a function of the observed states $\mathbf{x}(t)$ and some observation noise $\mathbf{v}(t)$ as [4]:

$$\mathbf{z}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{v}(t)). \tag{2.13}$$

The sensor making the observation is a bearings-only sensor, which can only make observations about the angle to the objects. The function $\mathbf{h}$ is a nonlinear function since it is the observed angle, $\tilde{\varphi}(t)$, which is the real angle, $\varphi(t)$, plus the noise:

$$\mathbf{h}(\mathbf{x}(t), \mathbf{v}(t)) = \tilde{\varphi}(\mathbf{x}(t), \mathbf{v}(t)) = \varphi(\mathbf{x}(t)) + \mathbf{v}(t). \tag{2.14}$$

The observation noise $\mathbf{v}(t)$ is modelled as white noise with covariance $\mathbf{R}$. The nonlinear function is linearized about nominal states $\mathbf{x}_n(t)$ and $\mathbf{z}_n(t)$ as in Grocholsky [4]:

$$\delta\mathbf{z}(t) = \mathbf{H}(t)\delta\mathbf{x}(t) + \mathbf{D}(t)\mathbf{v}(t). \tag{2.15}$$

Where

$$\mathbf{H}(t) = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}(t) = \mathbf{x}_n(t) \\ \mathbf{v}(t) = 0}} \quad \text{and} \quad \mathbf{D}(t) = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{v}}\right|_{\substack{\mathbf{x}(t) = \mathbf{x}_n(t) \\ \mathbf{v}(t) = 0}} \tag{2.16}$$

and the new linearized states are

$$\begin{aligned} \delta\mathbf{x}(t) &\equiv \mathbf{x}(t) - \mathbf{x}_n(t) \\ \delta\mathbf{z}(t) &\equiv \mathbf{z}(t) - \mathbf{z}_n(t). \end{aligned}$$

The $\mathbf{H}$-matrix is called the linearized observation matrix and the $\mathbf{D}$-matrix is the linearized observation noise matrix. The reason for the linearization, is that the information filter equations are linear. The linearized filter equations are detailed in Manyika and Durrant-Whyte [5] and simplified in Grocholsky [4], and describes how the *expected observed information* $\mathbf{I}(t)$ is related to the observations, which is

$$\mathbf{I}(t) = \mathbf{H}(t)^T \mathbf{R}^{-1} \mathbf{H}(t). \tag{2.17}$$

2.2.2 Update of Information Matrix

The information matrix is updated not only by the observed information, but there are losses due to process noise and the system dynamics could affect as well. First consider the system equations

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)), \tag{2.18}$$

where $\mathbf{x}(t)$ is the states, $\mathbf{u}(t)$ known control inputs and $\mathbf{w}(t)$ is the process noise, the latter is assumed to be a zero mean uncorrelated Gaussian process with covariance $\mathbf{Q}$. The function $\mathbf{f}$ is a system of non-linear differential equations, that could be linearized about nominal states $\mathbf{x}_n(t)$ and nominal control signals $\mathbf{u}_n(t)$ as

$$\delta\dot{\mathbf{x}}(t) = \mathbf{F}(t)\delta\mathbf{x}(t) + \mathbf{B}(t)\delta\mathbf{u}(t) + \mathbf{G}(t)\mathbf{w}(t). \tag{2.19}$$

The matrix $\mathbf{F}(t)$ is the linearized state transition matrix, $\mathbf{B}(t)$ is the linearized input matrix and $\mathbf{G}(t)$ is the linearized noise matrix. These are given by

$$\mathbf{F}(t) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}(t) = \mathbf{x}_n(t) \\ \mathbf{u}(t) = \mathbf{u}_n(t) \\ \mathbf{w}(t) = 0}}, \quad \mathbf{B}(t) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}(t) = \mathbf{x}_n(t) \\ \mathbf{u}(t) = \mathbf{u}_n(t) \\ \mathbf{w}(t) = 0}} \text{ and } \mathbf{G}(t) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{w}}\right|_{\substack{\mathbf{x}(t) = \mathbf{x}_n(t) \\ \mathbf{u}(t) = \mathbf{u}_n(t) \\ \mathbf{w}(t) = 0}} \tag{2.20}$$

$$\text{where} \quad \delta\mathbf{x}(t) \equiv \mathbf{x}(t) - \mathbf{x}_n(t) \quad \text{and} \quad \delta\mathbf{u}(t) \equiv \mathbf{u}(t) - \mathbf{u}_n(t). \tag{2.21}$$

Given these matrices, the update law of the information matrix $\mathbf{Y}(t)$ is described in Grocholsky [4] as:

$$\dot{\mathbf{Y}}(t) = -\mathbf{F}(t)\mathbf{Y}(t) - \mathbf{F}^T(t)\mathbf{Y}(t) - \mathbf{Y}(t)\mathbf{G}(t)\mathbf{Q}(t)\mathbf{G}^T(t)\mathbf{Y}(t) + \mathbf{I}(t) \tag{2.22}$$

where $\mathbf{I}(t)$ is given in (2.17). This is the continuous version of the prediction and update states of the information filter which is described in [4] and [5]. Since $\mathbf{R}$ and $\mathbf{Q}$ are positive semi-definite, the process noise cannot gain any information and observation cannot lose information. However, the system dynamics in $\mathbf{F}$ could lose or gain information over time.

2.3 Optimal Control

2.3.1 The General Optimal Control Problem

The general optimal control problem could be expressed simply as "choose the control signal such that the system behaves as good as possible" [3]. In mathematical terms, the problem can be formulated as in Glad and Ljung [3]:

Given initial conditions:

$$\mathbf{x}(0) = \mathbf{x}(t_0) \tag{2.23}$$

System equations:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{2.24}$$

Subject to constraints:

$$\psi(\mathbf{x}(t), \mathbf{u}(t)) = 0 \tag{2.25}$$

$$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \leq 0 \tag{2.26}$$

The criteria function to minimize is:

$$\mathbf{J}(\mathbf{x}(t), \mathbf{u}(t)) = \phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \mathbf{L}(\mathbf{x}(t), \mathbf{u}(t))\ dt \tag{2.27}$$

The solution to the optimal control problem would be:

$$\min_{\mathbf{u}(t)} \mathbf{J}(\mathbf{x}(t), \mathbf{u}(t)), \tag{2.28}$$

subject to the constraints (2.25) and (2.26).

2.3.2 Optimal Control In Path Planning

The criteria function $\mathbf{J}$ in (2.27) is the utility function discussed in Section 2.1.3, if the utility function is modified such that it is minimized. A maximum problem, could always be converted into a minimum problem, according to Böiers [2] with:

$$\max g(x) = -\min(-g(x)),$$

and in the implementation, the utility functions that required a maximization, like maximizing the determinant, were converted into minimum problems.

The solution to the optimal control problem is affected by the choice of utility function and the idea is to find a control signal, or sequence of control signals, that maximizes information. However, the optimization is done over a pre-defined optimization time, called the *time horizon* denoted by t_f, and it is only of interest to consider the information at the time horizon t_f. Therefore the criteria function is a function of the time horizon only, that is $\mathbf{J} = \mathbf{J}(t_f)$, and $\mathbf{L}$ in (2.27) is equal to zero.

Most optimal control problems requires a numerical solution and by parameterize the control signal into m steps in every optimization, an approximate solution will be found, for details see Bertsekas [1]. The idea is now to find a sequence of control steps u_i that maximizes information at the time horizon t_f as in Grocholsky [4]:

$$u_i(t) = \mathbf{p}_i \chi_i(t), \quad i = 1, \ldots, m. \tag{2.29}$$

Where χ_i simply holds the control variable over m equal time steps Δt_u as

$$\chi_i(t) = \begin{cases} 1 & \text{if } (j-1)\Delta t_u \leq t \leq j\Delta t_u \\ 0 & \text{otherwise} \end{cases}, \quad \Delta t_u = \frac{t_f - t_0}{m}. \tag{2.30}$$

The optimal control problem in (2.28) is now converted into a nonlinear programming problem [4]:

$$\min_{\mathbf{p}} \mathbf{J}(\mathbf{p}) = \phi(\mathbf{x}(t_f)) + \frac{1}{2}\Delta t_x \sum_{i=1}^{m \cdot n_{steps}} (\mathbf{L}_i(\mathbf{x}_i, \mathbf{u}_k) + \mathbf{L}_{i-1}(\mathbf{x}_{i-1}, \mathbf{u}_k)), \tag{2.31}$$

subject to the constraints in (2.25) and (2.26), where $\mathbf{p} = [u_1, \ldots, u_m]$ is the parameter vector and $\Delta t_x = \frac{\Delta t_u}{n_{steps}}$, $\{n_{steps} \geq 1\}$ is the time between the evaluations of the states, and k is the control index. The original optimal control problem is now in the form of mathematical programming problem [4]. In the path planning problem, $\mathbf{L}$ is zero, but it is given for generality.

There are no equal constraints as in equation (2.25), but there could be lower or equal constraints as in equation (2.26), with the control signals bounded as

$$u_{min} \leq u_i \leq u_{max}, \tag{2.32}$$

since it is reasonable that the movement of a UAV is restricted by its dynamics.

The solution of the optimal control problem is solved by *Matlab*'s optimization toolbox. If there are no bounds on the control signal, the problem is considered to be unbounded problem, solved by the function *fminunc*. Otherwise, the problem are bounded and solved by the function *fmincon*. How these functions are called is shown in Appendix A.

2.3.3 Gradient Determination

The functions in *Matlab*'s toolbox uses the gradient $\nabla_{\mathbf{p}}\mathbf{J}$ and the Hessian $\nabla^2_{\mathbf{p}}\mathbf{J}$ in order to find minimum. The use of the gradient will help the minimizer to reach a minimum in terms of efficiency and reliability. This can be done in two ways, either by letting the optimizer calculate the gradient and Hessian itself as an numerical solution, or by giving the analytical expressions explicitly. The first method requires no knowledge of the partial derivatives with respect to state and control vectors. However, the drawback is the computational load. For the latter method, the details are given here for the gradient, the details for calculating the Hessian are given in Grocholsky [4].

Differentiating (2.31) ($\mathbf{L} = 0$) with respect to $\mathbf{p}$ gives:

$$(\nabla_{\mathbf{p}}\mathbf{J}) = \frac{\partial\phi(\mathbf{x}(t_f))}{\partial\mathbf{x}(t_f)}\frac{\partial\mathbf{x}(t_f)}{\partial\mathbf{p}} \tag{2.33}$$

The first part of (2.33) is simply the derivative of the criteria function with respect to each state at the time horizon t_f. The second part is derived from (2.24) by applying the chain rule, which gives:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial\mathbf{x}}{\partial\mathbf{p}} = \frac{\partial\mathbf{f}}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\mathbf{p}} + \frac{\partial\mathbf{f}}{\partial\mathbf{u}}\frac{\partial\mathbf{u}}{\partial\mathbf{p}}, \text{ with } \left.\frac{\partial\mathbf{x}}{\partial\mathbf{p}}\right|_{t=t_0} = \frac{\partial\mathbf{x}_0}{\partial\mathbf{p}} \tag{2.34}$$

By the use of a Heun scheme, the first order sensitivities could be calculated as:

$$\frac{\partial\mathbf{x}_i}{\partial\mathbf{p}} = \left[\mathrm{I}_n - \frac{1}{2}\Delta t_x\frac{\partial\mathbf{f}_i}{\partial\mathbf{x}_i}\right]^{-1}\left[\left[\mathrm{I}_n + \frac{1}{2}\Delta t_x\frac{\partial\mathbf{f}_{i-1}}{\partial\mathbf{x}_{i-1}}\right]\frac{\partial\mathbf{x}_{i-1}}{\partial\mathbf{p}} + \frac{1}{2}\Delta t_x\left(\frac{\partial\mathbf{f}_i}{\partial\mathbf{u}_k} + \frac{\partial\mathbf{f}_{i-1}}{\partial\mathbf{u}_k}\right)\frac{\partial\mathbf{u}_k}{\partial\mathbf{p}}\right],$$

where I_n is a $(n \times n)$identity matrix, Δt_x is the time between each state evaluation, and $\frac{\partial\mathbf{f}}{\partial\mathbf{x}}$ are the derivatives of the system equations with respect to each state and $\frac{\partial\mathbf{f}}{\partial\mathbf{u}}$ with respect to the control signal.

Chapter 3

A Single Platform Example

To illustrate the use of information in control problems an example is given. A single vehicle is considered, where the task is to localize a feature in the xy-plane with a bearings-only sensor. This is done by seeking control action and trajectory that maximizes the information as described in Chapter 2. The example illustrates information as a performance metric and the effect of varied optimization time horizons. The different utility functions are evaluated and the difference between analytical implementation of the gradient and numerical calculation is examined. First the sensor platform is moving in the plane, and then a height of flight is added. Also the case with n objects is considered, where the path is planned such that all objects are to be localized simultaneously.

3.1 Modelling the Vehicle, Sensor and Environment

3.1.1 Sensor Platform Model

The sensor is attached to a sensor platform which is moving in the xy-plane with constant velocity V. The location $(x, y)^T$ and the direction of the vehicle are described by the state $\mathbf{x}_s(t)$. The direction of the vehicle is the heading and is modelled by the angle ψ between the head of the platform and the x-axis. The rate of change of the platform heading $\dot{\psi}$ is the control variable as in Figure 3.1. This is the same example as in Grocholsky [4]. The denotation "sensor platform" is referring to the vehicle, which is a UGV until the flight height is introduced where it becomes a UAV. The equations describing the sensor platform are summarized:

$$\mathbf{x}_s(t) = \begin{bmatrix} x(t) \\ y(t) \\ \psi(t) \end{bmatrix},\ \mathbf{x}_s(0) = \begin{bmatrix} x(0) \\ y(0) \\ \psi(0) \end{bmatrix},\ \mathbf{u}(t) = \dot{\psi};\ \dot{\mathbf{x}}_s(t) = \begin{bmatrix} V\cos(\psi(t)) \\ V\sin(\psi(t)) \\ \mathbf{u}(t) \end{bmatrix}. \quad (3.1)$$

In Figure 3.1 the coordinate system is not the usual, instead the x-axis is forward, the y-axis is rightwards, and z-axis downwards. This is called a forward-right-down (FRD) coordinate system and is often used for all kinds of flying platforms.

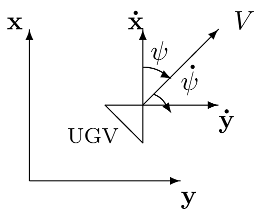

Figure 3.1: 2D sensor platform vehicle model

In real life applications, like the *SIREOS* project, the coordinates of the vehicle is given by GPS/INS and therefore a noise should be added to represent this uncertainty of the vehicle. Adding this to the model yields a quite complex model and is saved for future work.

3.1.2 Feature Model

The feature is represented by a stationary point $\mathbf{x}_f = (x_f, y_f)^T$ in the xy-plane. The uncertainty of the location is captured in the covariance of a two dimensional Gaussian distribution $\mathbf{P}_f(t)$. In the information filter, this is represented by a information matrix $\mathbf{Y}(t)$ as the inverse of the covariance as

$$\mathbf{Y}(t) = \mathbf{P}_f^{-1}(t). \tag{3.2}$$

Since it is a Gaussian distribution the Entropic Information and the Fisher information are the same according to (2.7), and there is no need to distinguish between them.

The feature process model is

$$\dot{\mathbf{x}}_f(t) = \omega(t), \tag{3.3}$$

where $\omega(t)$ is a zero mean Gaussian process with uncorrelated covariance $\mathbf{Q}(t)$ called the process noise. This is a bit contradictorily that a stationary point is modelled by some process noise. By adding a small process noise in the model, the feature is allowed to move a bit and is no longer stationary. However, this is a trick to improve the numerical conditioning since if there are any incorrect information, the impact will be lower and the effect of the new information gained more important.

The uncertainty is an ellipse about the estimated location of the object. When planning the next optimization step, the path is calculated under the assumption that the true location of the object is the estimated location. This is not true in real life, since the object could be in the outer range of its uncertainty ellipse, and the path would not be optimal to the object. Instead of having a Gaussian distribution representing the uncertainty, one could instead use a sum of Gaussian with different weights which gives a more accurate model. This is merely a warning and must be considered in future work.

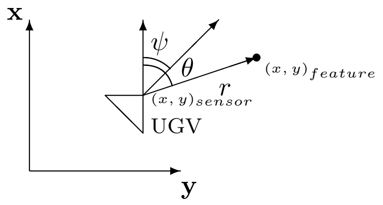

Figure 3.2: 2D sensor model

3.1.3 Sensor Model

The vehicle carries the sensor which makes observations of the feature. The observation is the bearing of the feature, that is the relative angle to the feature, which could be calculated as the angle from the x-axis to the feature θ, minus ψ as shown in Figure 3.2. The observation model is then, from (2.13) and (2.14):

$$\mathbf{z}(t) = \mathbf{h}(\mathbf{x}_f, \mathbf{x}_s) \tag{3.4}$$

$$\mathbf{h}(t) = \theta(t) - \psi(t) + \nu(t) = \arctan(\frac{y_f - y_s}{x_f - x_s}) - \psi(t) + \nu(t) \tag{3.5}$$

where $\nu(t)$ is a zero mean uncorrelated Gaussian process with variance $\mathbf{R} = \sigma^2$. Taking the Jacobian with respect to the feature state gives the linearized relationship between the sensed output and the states according to (2.16):

$$\begin{aligned}\mathbf{H}(t) &= \nabla_{\hat{\mathbf{x}}_\mathbf{f}} \mathbf{h}(\mathbf{x}_f, \mathbf{x}_s) \\ &= [\frac{-(\hat{y}_f - y_s(t))}{(\hat{x}_f - x_s(t))^2 + (\hat{y}_f - y_s(t))^2}, \frac{\hat{x}_f - x_s(t)}{(\hat{x}_f - x_s(t))^2 + (\hat{y}_f - y_s(t))^2}] \\ &= \frac{1}{\hat{r}(t)}\left[-\sin\hat{\theta}(t), \cos\hat{\theta}(t)\right],\end{aligned} \tag{3.6}$$

with $(\hat{x}_f, \hat{y}_f)$ as the estimated feature location, and $(\hat{r}, \hat{\theta})$ estimation of (r, θ) respectively. The resulting observed information is derived according to (2.17) as

$$\mathbf{I}(t) = \mathbf{H}^T(t)\mathbf{R}^{-1}\mathbf{H}(t). \tag{3.7}$$

3.1.4 System Equations

The state of the system consists of the platform model and the information matrix representing the uncertainty of the feature. The update of the vehicle states $\mathbf{x}_s$ is given in (3.1). The update of the information matrix is given by (2.22). A comparison between the feature model in (3.3) with the general expression in

(2.19), yields $\mathbf{F}(t) = 0$ and $\mathbf{G} = I$ (and $\mathbf{B} = 0$). The update of information is now reduced to

$$\dot{\mathbf{Y}}(t) = -\mathbf{Y}(t)\mathbf{Q}\mathbf{Y}(t) + \mathbf{I}(t). \tag{3.8}$$

The rate of change in information is some loss due to process noise and the gain of information by observation. The matrices $\mathbf{Y}(t)$ and $\mathbf{I}(t)$ are symmetric and $\mathbf{Q}$ is a diagonal matrix as

$$\mathbf{Y}(t) = \begin{bmatrix} \mathrm{Y}_x & \mathrm{Y}_{xy} \\ \mathrm{Y}_{xy} & \mathrm{Y}_y \end{bmatrix}, \; \mathbf{I}(t) = \begin{bmatrix} \mathrm{I}_x & \mathrm{I}_{xy} \\ \mathrm{I}_{xy} & \mathrm{I}_y \end{bmatrix} \text{ and } \mathbf{Q} = \begin{bmatrix} \mathrm{Q}_x & 0 \\ 0 & \mathrm{Q}_y \end{bmatrix}. \tag{3.9}$$

The feature information matrix is symmetric and it is therefore sufficient to calculate three of four values, that means the states representing the information would be

$$\mathbf{x}_{info} = \begin{bmatrix} \mathrm{Y}_x \\ \mathrm{Y}_{xy} \\ \mathrm{Y}_y \end{bmatrix}. \tag{3.10}$$

The equations derived for the evolution of the feature information combined with the equations of the vehicle dynamics describe fully the system state and the stacked system equations become

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{\mathbf{x}}_s \\ \dot{\mathbf{x}}_{info} \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\psi}(t) \\ \dot{\mathrm{Y}}_x(t) \\ \dot{\mathrm{Y}}_{xy}(t) \\ \dot{\mathrm{Y}}_y(t) \end{bmatrix} = \begin{bmatrix} V\cos(\psi(t)) \\ V\sin(\psi(t)) \\ \mathbf{u}(t) \\ -\mathrm{Y}_x^2(t)\mathrm{Q}_x - \mathrm{Y}_{xy}^2(t)\mathrm{Q}_y + \mathrm{I}_x(t) \\ -\mathrm{Y}_x(t)\mathrm{Q}_x\mathrm{Y}_{xy}(t) - \mathrm{Y}_{xy}(t)\mathrm{Q}_y\mathrm{Y}_y(t) + \mathrm{I}_{xy}(t) \\ -\mathrm{Y}_{xy}^2(t)\mathrm{Q}_x - \mathrm{Y}_y^2(t)\mathrm{Q}_y + \mathrm{I}_y(t) \end{bmatrix}. \tag{3.11}$$

The task is to reduce the uncertainty of the feature by maximizing information. For a Gaussian distribution, the information is given by (2.7) and the information is maximized when the utility function of $\mathbf{Y} = \mathbf{P}^{-1}$ is optimized. Introduce for example the determinant as utility function $\mathbf{J}(t)$ according to (2.9) as

$$\mathbf{J}(t_f) = \mid \mathbf{Y}(t_f) \mid = \mathrm{Y}_x(t_f)\mathrm{Y}_y(t_f) - \mathrm{Y}_{xy}^2(t_f). \tag{3.12}$$

and maximize it at the time horizon t_f.

3.1.5 Gradient Given Analytically

If the gradient would be given analytically, recall Section 2.3.3. The gradient of the utility function in (3.12) would be

$$\frac{\partial\phi(\mathbf{x}(t_f))}{\partial\mathbf{x}(t_f)} = \begin{bmatrix} 0 & 0 & 0 & \mathrm{Y}_y & -2\mathrm{Y}_{xy} & \mathrm{Y}_x \end{bmatrix}. \tag{3.13}$$

The gradient of the systems equations (3.11) is needed, and is given by

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \left[\begin{array}{cccc} 0 & 0 & -\mathbf{V}\sin(\psi) & \dots \\ 0 & 0 & \mathbf{V}\cos(\psi) & \dots \\ 0 & 0 & 0 & \dots \\ \frac{4(\hat{y}_f - y)^2(\hat{x}_f - x)}{\sigma^2 r^6} & \frac{4(\hat{y}_f - y)^3}{\sigma^2 r^6} - \frac{2(\hat{y}_f - y)}{\sigma^2 r^4} & 0 & \dots \\ -\frac{4(\hat{y}_f - y)(\hat{x}_f - x)^2}{\sigma^2 r^6} + \frac{(\hat{y}_f - y)}{\sigma^2 r^4} & -\frac{4(\hat{y}_f - y)^2(\hat{x}_f - x)}{\sigma^2 r^6} + \frac{(\hat{x}_f - x)}{\sigma^2 r^4} & 0 & \dots \\ \frac{4(\hat{x}_f - x)^3}{\sigma^2 r^6} - \frac{2(\hat{x}_f - x)}{\sigma^2 r^4} & \frac{4(\hat{y}_f - y)(\hat{x}_f - x)^2}{\sigma^2 r^6} & 0 & \dots \end{array} \right.$$

$$\left. \begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -2\mathrm{Y}_x\mathrm{Q}_x & -2\mathrm{Y}_{xy}\mathrm{Q}_y & 0 \\ -\mathrm{Y}_{xy}\mathrm{Q}_x & -\mathrm{Y}_x\mathrm{Q}_x - \mathrm{Y}_y\mathrm{Q}_y & -\mathrm{Y}_{xy}\mathrm{Q}_y \\ 0 & -2\mathrm{Y}_{xy}\mathrm{Q}_x & -2\mathrm{Y}_y\mathrm{Q}_y \end{array} \right] \tag{3.14}$$

where $(x, y)^T$ is the vehicle's position, $(\hat{x}_f, \hat{y}_f)^T$ is the feature's estimated position and r is the distance between the vehicle and the target, and

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \left[\begin{array}{cccccc} 0 & 0 & 1 & 0 & 0 & 0 \end{array} \right]^T. \tag{3.15}$$

These are the equations needed to calculate the gradient $\nabla_{\mathbf{p}}\mathbf{J}$ according to (2.33). One could realize that if the model is further complicated, it would be quite difficult to derive the expressions needed for the analytical gradient. Instead one could let *Matlab* calculate the gradient numerically.

3.2 Simulations

The initial conditions in (3.1) are the starting position of the vehicle and the starting angle as the angle between the heading of the vehicle and the x-axis. Let the vehicle start in origo, and specify a desired starting angle. In the simulation examples the starting angle was set to $\pi/2$ *rad*, so that

$$\mathbf{x}_s(0) = \left[\begin{array}{ccc} 0 & 0 & \pi/2 \end{array} \right]^T.$$

There must also be some starting information, since in order to plan how to localize a target, some information is needed. In other words, there is a need to know that the target exists, otherwise it is hard to plan the path. The details used in the simulations are

$(x_f, y_f) = (10, 10)\ m$	Feature location (stationary)
$V = 1\ m/s$	Constant velocity
$\sigma = 2.5° \Rightarrow \mathbf{R} = \sigma^2$	Observation noise
$\mathbf{Y}(0) = \mathrm{I}_2 \cdot 10^{-3}$	Low starting information
$\mathbf{Q} = \mathrm{I}_2 \cdot 10^{-6}$	Process noise
I_2	A 2×2 identity matrix
$m = 2$	Number of parameterized control signals in each optimization step
$t_f = 1\ s$	Optimization time horizon
$\lvert u_i \rvert \leq 1\ rad/s$	Bounded control signal
$\mathbf{J}(t_f) = \lvert \mathbf{Y}(t_f) \rvert$	Utility function is determinant of information matrix

These values are chosen for simplicity and are not values for real applications, it is the principle that is interesting. The resulting path is presented in Figure 3.3 and the observed information and the parameterized control signal are shown in Figure 3.4. The control signal is bounded and then the optimization is solved by the function *fmincon*, see Appendix A for how it was used. The solution is calculated as following. In each step, a control signal, parameterized into m equal long parts, will be calculated such that the information at the time horizon t_f is maximized, given the system equations and subject to constraints. The first optimum will be a trajectory from origo of length $V \cdot t_f = 1\ m$, and this point is reached by the optimal control signal consisting of $m = 2$ steps. The first part of the control signal u_1 is valid between the time 0 and $t_f/2$ and the next part u_2 is valid between $t_f/2$ and t_f. After the first optimum is reached, and the states $\mathbf{x}$ have been updated, a new optimization is done. The uncertainty of the feature location is plotted as an ellipse about the feature, and the procedure is terminated when the uncertainty is so low that the object could be said is localized. The total number of optimizations are the number of 'x':s in the figure, a total of 25, that is the number of optimizations required for localizing the target in this case. Since then, the information is higher than a pre-defined value, and the target is said to be localized.

As can be seen in the path, the sensor is trying to see the object orthogonally with respect to the last observed direction. For a bearings only sensor, an object could be localized with just two observations if the observations are orthogonal to each other. The optimization time is much smaller than the time it would take for the sensor to go to a position where the second observation is orthogonal to the first observation, and the step is taken in the orthogonal direction.

The path is the planned path for the sensor platform. However, since there are no uncertainties of the platform's position, it will also be the performed path by the platform. The control signal varies all the time and it is undesirable since the controller would be worn out. But it is understandable that the solution gives a varying control signal, since it plans very short ahead. To avoid this problem, one could formulate penalty functions on varying control signals, but that is not the main task of this thesis and is therefore left out.

The calculation times for this problem are up to ten seconds in each optimization, and thus the total calculation time would be about 2 minutes[1]. The calculation time is highly dependent on the number of control parameterizations m. A test with $m = 3$ gave a total time of 9 minutes. The calculation time

[1]Celeron 2.0 GHz 760 MB RAM

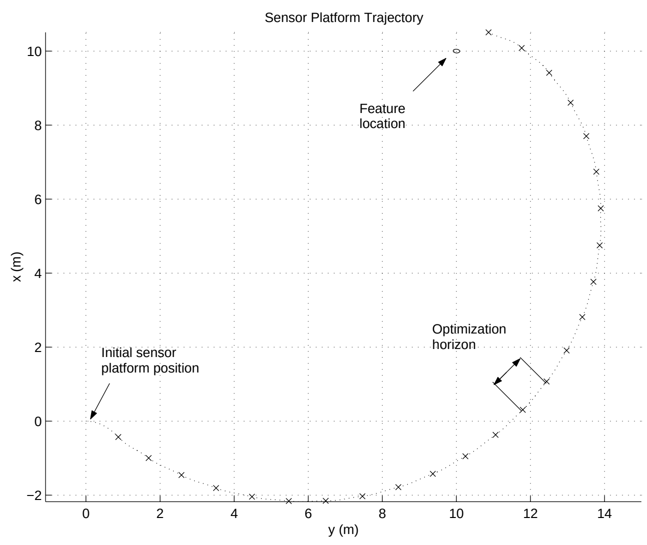

Figure 3.3: Trajectory of the sensor platform. Each 'x' marks a new optimization. The feature location's uncertainty is an ellipse about its location.

is also dependent on the tolerance chosen for the optimization. The tolerance is set in the *Matlab* function *optimset*, which sets the parameters used in the optimization such as number of maximum iterations etc, see Appendix A.

The use of information as a performance index is intuitive, since the more information you have of a feature, the more certain are you of its location. It is also a convenient criteria for the optimization process, since when the information is maximized, the uncertainty of the feature location is minimized.

3.2.1 Plotting the Criteria Function

To illustrate the complexity of the optimization problem, the criteria function could be plotted over the first optimization step, that is over the first parameterized control signal $[u_1\ u_2]$. Recall the criteria function used here as the determinant of the information matrix from (3.12)

$$\mathbf{J}(t_f) = \Upsilon_x(t_f)\Upsilon_y(t_f) - \Upsilon_{xy}^2(t_f).$$

This function is plotted for two cases of bound on the control signal, first bounded as $-10\ rad/s \leq u_i \leq 10\ rad/s,\ i = 1, 2$, plotted in Figure 3.5. The optimizer is trying to find the maximum, and the problem is that there are several local maxima, and there is a risk that the optimizer will find a local maxima instead of the global. The reason why there are so many maxima is that when having a high bound on the control signal, the vehicle could either turn around with a high control signal and end up about the same place as a low control signal. For this example, tighten the bound on the control signal to $-1\ rad/s \leq u_i \leq 1\ rad/s,\ i = 1, 2$, for the same situation yields the situation in Figure 3.6. The problem of many maxima is not solved by this, instead one could accept any maxima inside the bound since all the solutions inside the

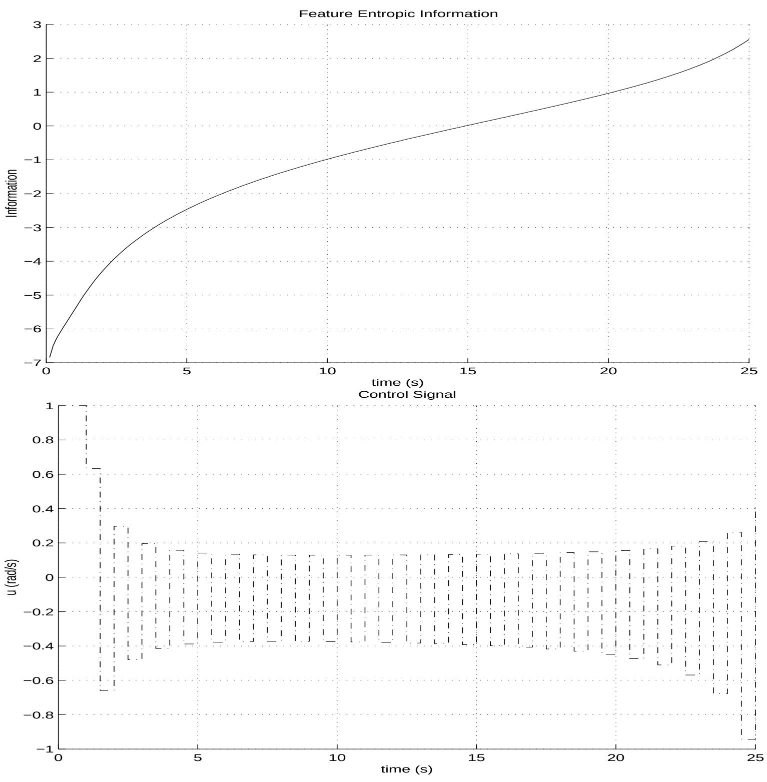

Figure 3.4: Feature entropic information and the parameterized control signal.

bounds are said to be good enough, but there are less maxima than for the high control signal. The reason why the criteria function was plotted, is to illustrate that the optimization is not perfect, but since it is an application of engineering, one could satisfy with a good solution and not the best in the theoretical sense. The *Matlab* functions *fminunc* and *fmincon* uses line search when it cannot solve the problem otherwise. The line search algorithm has problem with local maximum, and instead one could implement other optimization methods when it is known that there are several maxima. This would however take more time than scheduled for the thesis.

3.2.2 The Effect of Optimization Time Horizon

The sensor platform's trajectory is affected by the time horizon t_f in the optimization. Figure 3.7 shows a comparison between three different time horizons. The number of parameterizations of the control signal m differs, since in a longer optimization, each parameterization of the control signal is valid longer. The three time horizons are short ($1\ s$) for 16 optimization steps, intermediate ($4\ s$) for 4 optimization steps and long ($8\ s$) for 2 optimization steps. The reason is that the total time would be 16 s for all cases. The cases can be summarized in Table 3.1.

As can been seen, the platform with long optimization time travels more

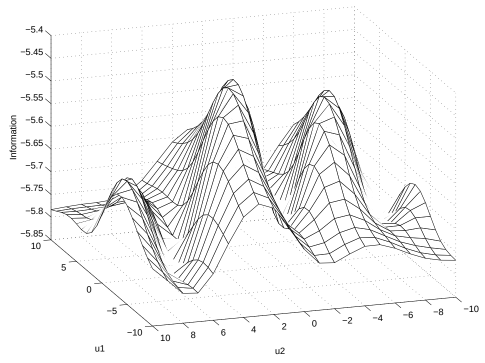

Figure 3.5: The criteria function over a parameterized control signal $[u_1\ u_2], -10 \leq u_i \leq 10$

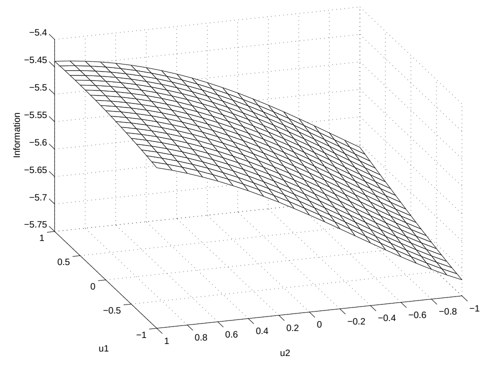

Figure 3.6: The criteria function over a parameterized control signal $[u_1\ u_2], -1 \leq u_i \leq 1$

direct to the feature. It will not gain as much information in the beginning as the case with short optimization time, but since it plan further in the future, the information at the time horizon will increase as seen in Figure 3.8.

According to Figure 3.8, a long time horizon is preferable compared to a short time horizon. But there are problems with a long horizon. The calculation time was different for the three cases. For the first case was about 1.8 minutes, for the second case about 2.3 minutes and in the third case about 6.3 minutes. This is just a simple case with one object, later on the model will be extended to n objects and there is a need to simulate with different time horizons later on, to evaluate the choice of time horizon.

3.2.3 The Effect of Prior Information

In the simulations, the starting information was set to 10^{-3} in both the x- and y-direction. The starting information is a measure of how much is known about the object when planning the first step in terms of uncertainty. When

Case	1	2	3
Time horizon $t_f(s)$	1	4	8
Number of optimizations	16	4	2
Total time (s)	16	16	16
Control parameters m	2	4	8

Table 3.1: Details of the three cases used to investigate the effect of optimization time horizon.

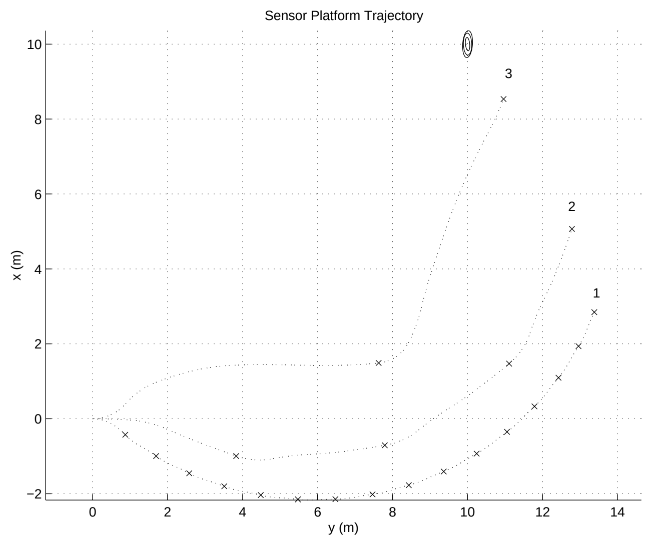

Figure 3.7: The trajectories for platforms with different optimization times for the three cases in Table 3.1.

knowing more about the object, *i.e.* the starting information was set to 1 in both directions, the resulting path is seen in the left part of Figure 3.9. It will travel in the direction of the object at first, since it will not gain that much information by taking a step orthogonally. Instead it needs to move closer, which it does until one point where it turns and continues like in the first simulation in Figure 3.3.

Another simulation is tried where the starting information is set to 1 in the x-direction and 10^{-3} in the y-direction, shown in the right part of Figure 3.9. This is the case when knowing one coordinate of the object's position and it is needed to reduce the uncertainty in the other direction. The vehicle is trying to reduce the uncertainty in the y-direction by moving in the direction which is orthogonal to the initial uncertainty, that is in the y-axis. The vehicle continues until the uncertainty in the y-direction is about the same as the x-axis and it turns to reduce the uncertainty in both directions.

3.2.4 Evaluation of the Utility Function

In the simulations below, three different utility functions are evaluated. The details of the different simulations are given in Table 3.2. The first one is

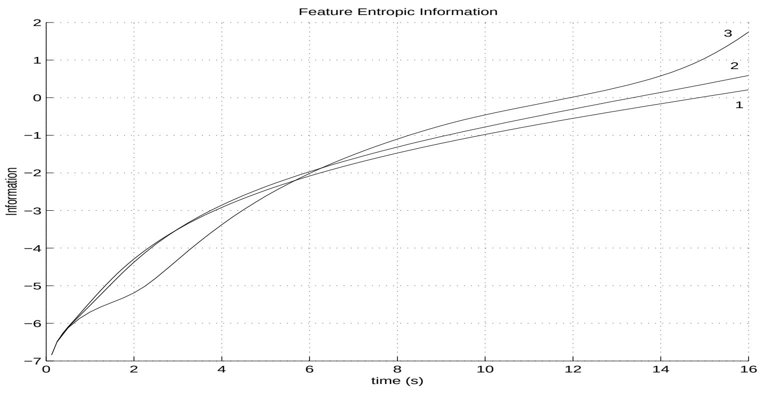

Figure 3.8: The information for platforms with different optimization times for the three cases in Table 3.1.

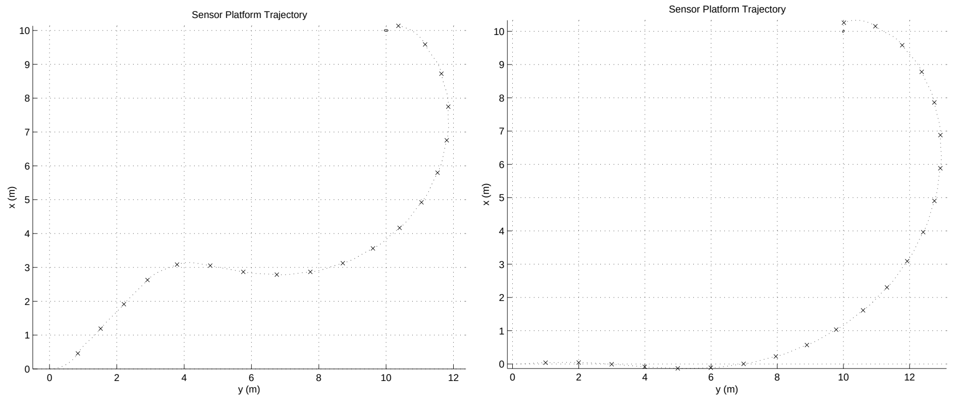

Figure 3.9: Left: The effect of high starting information. Right: High starting information in x-direction and low staring information in y-direction.

maximizing the determinant of the information matrix, and tried for two cases where the gradient of the utility function is given analytically, case 1, and by letting *Matlab* approximate the gradient numerically, case 2. The optimization is made by *Matlab*'s function *fmincon*, where it is possible for the user to choose whether the gradient is given explicitly or not, and for a simple case like this with only one object, the gradient could be calculated as described in Section 2.3.3. Case 3 is the path from the trace of the information matrix's inverse and case 4 is maximizing the minimum of the eigenvalues. The resulting paths are shown in Figure 3.10.

All paths have the same structure, shaped as a spiral. The time horizon was set to 1 s for simplicity and except for the utility function and gradient, all other simulation parameters are the same. The information is plotted in Figure 3.11.

It seems that all paths are reasonable, and the maximizing of the determinant is considered to give the best result, since the gain of information is faster than for the other utility functions as seen in Figure 3.11. Based on the same result, the analytical implementation of the gradient is better since information

Case	1	2	3	4
Utility function	$\max\|\mathbf{Y}\|$	$\max\|\mathbf{Y}\|$	$\min(\mathrm{tr}\mathbf{Y}^{-1})$	$\max(\min\mathrm{eig}(\mathbf{Y}))$
Gradient	Analytical	Numerical	Numerical	Numerical

Table 3.2: Details of the four cases used to investigate the effect of different utility functions.

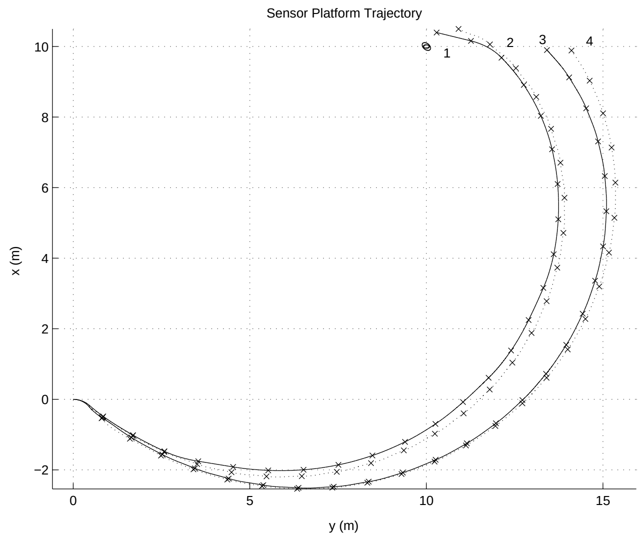

Figure 3.10: Sensor platform trajectory calculated from different utility functions: 1) $\max|\mathbf{Y}|$ with gradient analytically, 2) $\max|\mathbf{Y}|$ with gradient numerically, 3) min(trace $\mathbf{Y}^{-1}$), 4) max(min eig($\mathbf{Y}$))

is higher, and the calculation time is faster. But there are not so much difference in information between the analytical and numerical gradient, and when the model gets more complicated, it is difficult to derive the analytical expression explicitly. Especially when objects are removed or added to the model, and one could satisfy with the numerical approximation of the gradient.

3.3 n Objects

Following the methods of this chapter, it seems that the vehicle could successfully localize an object. To make things more interesting, the vehicle must be able to handle the case of n objects on the ground.

3.3.1 Modelling 2 Objects

First the model should be extended to the case with two objects. A symmetric information matrix $\mathbf{Y}$ is now of (4×4) as:

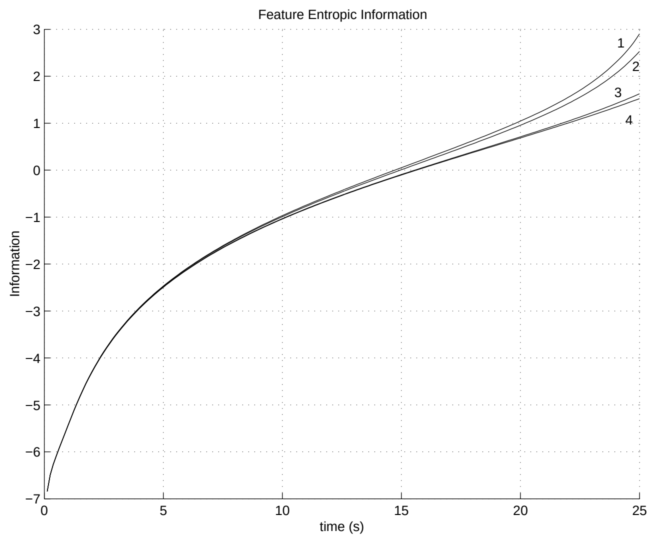

Figure 3.11: Information from the different utility functions: 1. max $|\mathbf{Y}|$ with gradient analytically, 2. max $|\mathbf{Y}|$ with gradient numerically, 3. min(trace $\mathbf{Y}^{-1}$), 4. max(min eig($\mathbf{Y}$))

$$\mathbf{Y}(t) = \begin{bmatrix} \mathrm{Y}_{x_1} & \mathrm{Y}_{x_1y_1} & \mathrm{Y}_{x_1x_2} & \mathrm{Y}_{x_1y_2} \\ \mathrm{Y}_{x_1y_1} & \mathrm{Y}_{y_1} & \mathrm{Y}_{x_2y_1} & \mathrm{Y}_{y_1y_2} \\ \mathrm{Y}_{x_1x_2} & \mathrm{Y}_{x_2y_1} & \mathrm{Y}_{x_2} & \mathrm{Y}_{x_2y_2} \\ \mathrm{Y}_{x_1y_2} & \mathrm{Y}_{y_1y_2} & \mathrm{Y}_{x_2y_2} & \mathrm{Y}_{y_2} \end{bmatrix}. \tag{3.16}$$

With just one more object, there were seven new states introduced. There is now three states representing the vehicle, same as in (3.1) and ten states from the information matrix above. The question arises whether some states, especially the cross states $\mathrm{Y}_{i_1j_2}$, could be zero. Recall the information rate of change from (3.8) as

$$\dot{\mathbf{Y}}(t) = -\mathbf{Y}(t)\mathbf{Q}\mathbf{Y}(t) + \mathbf{I}(t), \tag{3.17}$$

and the same observation model as for one object from (3.4) now extended as

$$\begin{aligned} \mathbf{z}(t) &= [\mathbf{h}_1(\mathbf{x}_{f_1}, \mathbf{x}_s), \mathbf{h}_2(\mathbf{x}_{f_2}, \mathbf{x}_s)]^T \\ \mathbf{h}_1(t) &= \theta_1(t) - \psi(t) + \nu_1(t) \\ \mathbf{h}_2(t) &= \theta_2(t) - \psi(t) + \nu_2(t). \end{aligned} \tag{3.18}$$

The linearized observation matrix $\mathbf{H}(t)$ is

$$\begin{aligned} \mathbf{H}(t) &= \nabla_{\hat{\mathbf{x}}_\mathbf{f}} \mathbf{h}(\mathbf{x}_f, \mathbf{x}_s) \\ &= \begin{bmatrix} -\frac{\sin\theta_1(t)}{r_1(t)} & \frac{\cos\theta_1(t)}{r_1(t)} & 0 & 0 \\ 0 & 0 & -\frac{\sin\theta_2(t)}{r_2(t)} & \frac{\cos\theta_2(t)}{r_2(t)} \end{bmatrix}. \end{aligned} \tag{3.19}$$

The $\mathbf{R}$ matrix is the covariance matrix of the two noise processes $\nu_1(t)$ and $\nu_2(t)$ as

$$\mathbf{R} = \begin{bmatrix} \operatorname{var}(\nu_1) & \operatorname{cov}(\nu_1, \nu_2) \\ \operatorname{cov}(\nu_1, \nu_2) & \operatorname{var}(\nu_2) \end{bmatrix}. \tag{3.20}$$

Combining (3.19) and (3.20) gives the observed information according to (3.7) as

$$\begin{aligned} \mathbf{I}(t) &= \mathbf{H}^T(t)\mathbf{R}^{-1}\mathbf{H}(t) \\ &= \frac{1}{(-\operatorname{var}(\nu_1)\operatorname{var}(\nu_2) + \operatorname{cov}(\nu_1, \nu_2))} \left[\begin{array}{cc} -\operatorname{var}(\nu_2)\frac{\sin^2\theta_1}{r_1^2} & \dots \\ \operatorname{var}(\nu_2)\frac{\sin\theta_1\cos\theta_1}{r_1^2} & \dots \\ \operatorname{cov}(\nu_1, \nu_2)\frac{\sin\theta_1\sin\theta_2}{r_1 r_2} & \dots \\ -\operatorname{cov}(\nu_1, \nu_2)\frac{\sin\theta_1\cos\theta_2}{r_1 r_2} & \dots \end{array} \right. \end{aligned}$$

$$\left. \begin{array}{ccc} \operatorname{var}(\nu_2)\frac{\sin\theta_1\cos\theta_1}{r_1^2} & \operatorname{cov}(\nu_1, \nu_2)\frac{\sin\theta_1\sin\theta_2}{r_1 r_2} & -\operatorname{cov}(\nu_1, \nu_2)\frac{\sin\theta_1\cos\theta_2}{r_1 r_2} \\ -\operatorname{var}(\nu_2)\frac{\cos^2\theta_1}{r_1^2} & -\operatorname{cov}(\nu_1, \nu_2)\frac{\cos\theta_1\sin\theta_2}{r_1 r_2} & \operatorname{cov}(\nu_1, \nu_2)\frac{\cos\theta_1\cos\theta_2}{r_1 r_2} \\ -\operatorname{cov}(\nu_1, \nu_2)\frac{\cos\theta_1\sin\theta_2}{r_1 r_2} & -\operatorname{var}(\nu_1)\frac{\sin^2\theta_2}{r_2^2} & \operatorname{var}(\nu_1)\frac{\sin\theta_2\cos\theta_2}{r_2^2} \\ \operatorname{cov}(\nu_1, \nu_2)\frac{\cos\theta_1\cos\theta_2}{r_1 r_2} & \operatorname{var}(\nu_1)\frac{\sin\theta_2\cos\theta_2}{r_2^2} & -\operatorname{var}(\nu_1)\frac{\cos^2\theta_2}{r_2^2} \end{array} \right],$$

where

$$\begin{aligned} r_1 &= r_1(t) \\ r_2 &= r_2(t) \\ \theta_1 &= \theta_1(t) \\ \theta_2 &= \theta_2(t). \end{aligned}$$

It can clearly be seen that if the two noise processes $\nu_1(t)$ and $\nu_2(t)$ are independent, there are no covariances and the observed information matrix $\mathbf{I}(t)$ would become a block matrix as

$$\mathbf{I}(t) = \begin{bmatrix} \mathrm{I}_1(t) & 0 \\ 0 & \mathrm{I}_2(t) \end{bmatrix}, \tag{3.21}$$

where $\mathrm{I_i}(t), i = 1, 2$ are (2×2) matrices representing the observed information from respective object. From this, a model with n objects could be easily be derived by just extending the observed information with block matrices. This is not sufficient for the cross states $\mathrm{Y}_{i_1 j_2}$ in (3.16) to be zero. The term in the update law corresponding to the loss due to the process noise

$$-\mathbf{Y}(t)\mathbf{Q}\mathbf{Y}(t)$$

must also be considered. The process noise is modelled with a block matrix, each block containing process noise for the two objects, extended from (3.9) to

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & 0 \\ 0 & \mathbf{Q}_2 \end{bmatrix} = \begin{bmatrix} \mathrm{Q}_{x_1} & 0 & 0 & 0 \\ 0 & \mathrm{Q}_{y_1} & 0 & 0 \\ 0 & 0 & \mathrm{Q}_{x_2} & 0 \\ 0 & 0 & 0 & \mathrm{Q}_{y_2} \end{bmatrix}. \tag{3.22}$$

If the cross states $\mathrm{Y}_{i_1 j_2}$ are set to zero, then the term $\mathbf{Y}(t)\mathbf{Q}\mathbf{Y}(t)$ would be a block matrix as

$$\mathbf{Y}(t)\mathbf{Q}\mathbf{Y}(t) = \begin{bmatrix} \mathbf{Y}_1(t)\mathbf{Q}_1\mathbf{Y}_1(t) & 0 \\ 0 & \mathbf{Y}_2(t)\mathbf{Q}_2\mathbf{Y}_2(t) \end{bmatrix}. \tag{3.23}$$

Since both terms, $\mathbf{I}(t)$ and $\mathbf{Y}(t)\mathbf{Q}\mathbf{Y}(t)$, in the update of the information matrix in (3.17) are block matrices with non-diagonal blocks as zeros, there will not be any information update from the cross states $\mathrm{Y}_{i_1 j_2}$. With no update of those states, they will not affect the optimization and therefore those states could be set to zero. Then the information matrix in (3.16) would be a diagonal matrix, with (2×2) block matrices on the diagonal representing the information of each object as

$$\mathbf{Y}(t) = \begin{bmatrix} \mathrm{Y}_1(t) & 0 \\ 0 & \mathrm{Y}_2(t) \end{bmatrix}, \tag{3.24}$$

where

$$\mathbf{Y}_i(t) = \begin{bmatrix} \mathrm{Y}_{x_i} & \mathrm{Y}_{x_i y_i} \\ \mathrm{Y}_{x_i y_i} & \mathrm{Y}_{y_i} \end{bmatrix}.$$

The matrix in (3.24) could easily be extended to n objects, with adding block matrices for each object.

3.3.2 Path for 2 Objects

In these simulations, the noise processes $\nu_1(t)$ and $\nu_2(t)$ are independent of each other, and the resulting path is examined. An information matrix is constructed as in (3.24), and from this the determinant is taken as utility function, and information could be maximized.

The observed information is inverse proportional to the squared distance and angle dependent, and there is a risk that the optimizer will only "zoom" on the nearest object, since it will gain much information by getting closer all the time. In order to avoid such problem, the object is removed from the model when localized, since there is no point of getting more information about something that already is localized. The states representing the information from the objects are removed from the system equations and the new information matrix is simply the information matrix from the non-localized object.

In the following simulation two features was placed in $(10, 10)$ and $(10, 0)$, and the vehicle starts in origo with heading in the y-axis direction. The optimization time horizon t_f is set to 1 s in the first simulation and to 8 s in the second simulation, and the utility function is the determinant of the information matrix. The noise is the same for the two objects.

The resulting paths are shown in Figure 3.12, where the short time horizon to the left and the long time horizon to the right. In both cases, the resulting path is where information is gained the most from both objects. For the short time horizon, the first step is as much orthogonal to both objects as possible, and at a certain point, it will gain more information to "zoom" the object closest and will continue until the first object is localized. Next the full attention will focus on the remaining object and it will be the case of localizing one object.

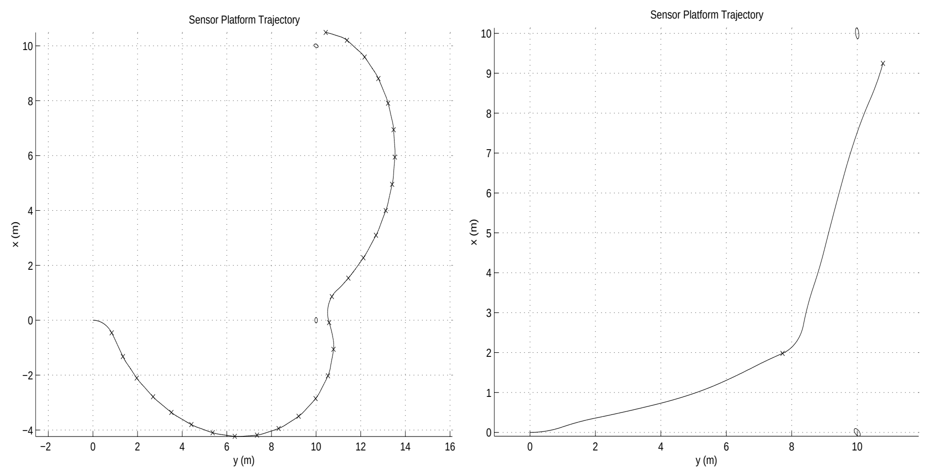

Figure 3.12: Left: Simulated sensor platform trajectory with two objects, time horizon 1 s. Right: Simulated sensor platform trajectory with two objects, time horizon 8 s.

For the long time horizon, it could place its first optimum close and orthogonal to both objects.

From the two paths, it is very tempting to make the conclusion that the long time horizon is better than the short time horizon. But if the threshold for localization is set higher, the long time horizon must plan a long time ahead close to an object. In Figure 3.13 a path is shown, where the problem with a long time horizon is seen. The first two points are the same as the right part of Figure 3.12, but then the path must be planned a long time ahead close to the target, until it is localized and then turns back to the other object, which is not very effective. The short optimization horizon is not much more effective, but it takes less time to calculate. This example was given to illustrate that a long optimization horizon is not always preferable as one could assume be just looking at Figure 3.12.

3.3.3 Extension to n Objects

The model can now be extended to handle n objects, since it is about adding and removing states to the model and hence to the information matrix. When modelling n objects, each object i has its own (2×2) information matrix $\mathrm{Y_i}$. Then a global information matrix could be constructed by putting these on the diagonal as

$$\mathbf{Y} = \begin{bmatrix} \mathrm{Y}_1 & 0 & \dots & 0 \\ 0 & \mathrm{Y}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathrm{Y_n} \end{bmatrix}. \tag{3.25}$$

From this matrix, all discussed utility functions could be formulated such as maximizing the determinant and minimizing the trace of its inverse.

A simulation is made with five objects randomly placed in the xy-plane. The

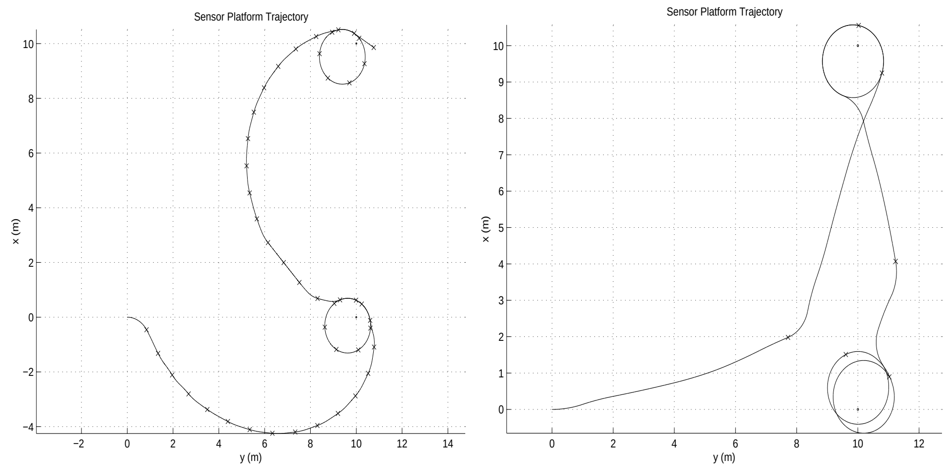

Figure 3.13: With long time horizon (right), the UAV must turn back to the initially nearest object compared to the short time horizon (left).

same parameters is used as for the simulation with two objects and the result is shown in Figure 3.14.

There are most certainly many maxima of the utility function, but as said before, any local maxima would be good enough and new optimizations are done until localization.

3.4 3D-modeling

The previous simulations has been in two dimensions, but a UAV has a certain height of flight and it is natural to extend the model to three dimensions. The height could be taken into account as an optimization parameter in a more complex model treating the dynamics of an aircraft. A simplification is keeping the height constant in the simulations justified by keeping down the complexity of the problem and that an UAV often fly at a constant height in reconnaissance missions.

There is also a singularity in the model, since the observed information is inverse proportional to the squared distance to the feature. This means that the vehicle could not fly straight over an object. The problem was ignored by removing object when localized or by bounding the control signals, so the vehicle never had to fly straight over an object. Later on in this section, a method is derived where the information is observed in a different coordinate system and then transformed into Cartesian coordinates, and the singularity is avoided.

3.4.1 A 3D Object

As described previously, the uncertainty of an object's location is represented by its information matrix $\mathbf{Y}(t)$. The concept is taken into three dimensions with an extended information matrix:

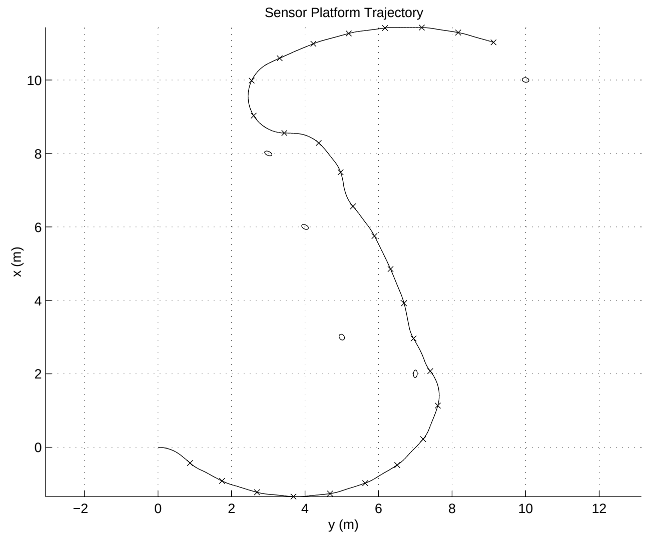

Figure 3.14: Sensor platform trajectory with five objects.

$$\mathbf{Y}(t) = \begin{bmatrix} \mathrm{Y}_x & \mathrm{Y}_{xy} & \mathrm{Y}_{xz} \\ \mathrm{Y}_{xy} & \mathrm{Y}_y & \mathrm{Y}_{yz} \\ \mathrm{Y}_{xz} & \mathrm{Y}_{yz} & \mathrm{Y}_z \end{bmatrix}. \tag{3.26}$$

By modelling an object in three dimensions, the uncertainty in the z-direction will also be considered. Even though the objects are on the ground, there should be an uncertainty in the z-direction, because the vehicle does not fly at an exact high. But the vehicle's position is assumed to be known exactly, and therefore the uncertainty is on the object instead.

The information matrix is symmetric as before and it is sufficient with six states to represent an object. The update of the information matrix is still:

$$\dot{\mathbf{Y}}(t) = -\mathbf{Y}(t)\mathbf{Q}\mathbf{Y}(t) + \mathbf{I}(t), \tag{3.27}$$

but both the process noise $\mathbf{Q}$ and the observed information $\mathbf{I}$ are now (3×3)-matrices.

3.4.2 Information in 3D

Let the UAV fly in the xy-plane and the distance to the ground, z_s, is kept constant. An object on the ground could be described by the angle α same as θ in the two dimensional case and the angle β, which is the angle from the xy-plane to the object, shown in Figure 3.15.

Introduce a sensor direction $\hat{n}$ as the vector from the vehicle to the feature. At this moment, the sensor is considered to "point" at every direction and a camera will be introduced later on in Chapter 6. For now, the sensor has unlimited field of view. A coordinate system is set in the end of the vector at the object, as in Figure 3.16. The information is observed orthogonal to the direction of the camera, that is in the $\hat{\theta}$ and the $\hat{\phi}$-direction. In other words, the

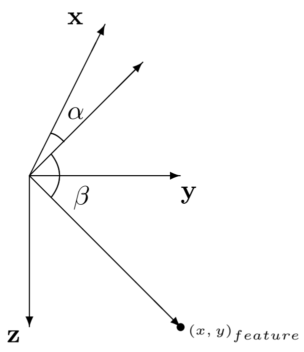

Figure 3.15: 3D modelling of a feature described by the angles α and β.

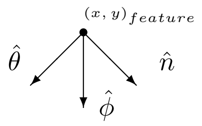

Figure 3.16: Local coordinates for observing information.

variance of the noise process in the $\hat{n}$-direction is infinite. The variance in the $\hat{\theta}$- and the $\hat{\phi}$-directions are σ_θ^2 and σ_ϕ^2 respectively. The inverse of the covariance matrix $\mathbf{R}$ could be set as

$$\mathbf{R}^{-1} \approx \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{\sigma_\theta^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_\phi^2} \end{bmatrix}, \tag{3.28}$$

if the noise processes are uncorrelated. The information could now be transformed into the global Cartesian coordinates, by using the angles α and β. The angle α could be expressed as the rotation about the z-axis in positive direction and the angle β would be the rotation about the y-axis in negative direction. The rotation matrices are described in [7] as

$$\mathbf{Rot}_z(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and} \tag{3.29}$$

$$\mathbf{Rot}_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix}. \tag{3.30}$$

The resulting rotation matrix which transforms the global Cartesian coordinates into the local coordinates would be [7]

$$\mathbf{Rot}_{tot} = \mathbf{Rot}_y(\beta)\mathbf{Rot}_z(\alpha), \tag{3.31}$$

which is equal to align the x-axis of the global Cartesian coordinate system with the direction $\hat{n}$ of the sensor by first rotate an angle α about the z-axis and then an angle β about the y-axis. The information is gained into cartesian coordinates by first transform with $\mathbf{Rot}_{tot}$ into the sensor coordinates, and there make an observation $\mathbf{I}_{local}$ and then transform back with $\mathbf{Rot}_{tot}^{-1}$ as

$$\mathbf{I}_{global} = \mathbf{Rot}_{tot}^{-1}\ \mathbf{I}_{local}\ \mathbf{Rot}_{tot}. \tag{3.32}$$

It can be shown as in [7] that the rotation matrix has following useful properties

$$\mathbf{Rot}_i^{-1}(\gamma) = \mathbf{Rot}_i^{T}(\gamma) = \mathbf{Rot}_i(-\gamma),$$

and (3.32) would be

$$\mathbf{I}_{global} = \mathbf{Rot}_z(-\alpha)\ \mathbf{Rot}_y(-\beta)\ \mathbf{I}_{local}\ \mathbf{Rot}_y(\beta)\ \mathbf{Rot}_z(\alpha). \tag{3.33}$$

This yields in a (3×3) information matrix, used in the update law in (3.27), without a singularity. The $\mathbf{H}$ in this case is simply the identity matrix in the local coordinates, since the feature location and the observed location is the same point.

A simulation with the same properties as for the two dimensional model in Figure 3.3 is shown in Figure 3.17 for two different heights. In the left picture the height of flight z_s is 1 m and the β-angle is close to zero, and the rotation matrix in (3.30) is almost the identity matrix, and there is hardly any rotation about the y-axis, just like in the two dimensional case. That is why the simulations are similar. The flight height affects more when it is higher, and in the right picture of Figure 3.17 it was set to 10 m. The rotation matrix about the y-axis differs now from the identity matrix, and there will be essential amount of information in all states in the information matrix (3.26) and this will affect the eigenvalues of the information matrix and hence the utility function. The scalar measure of the utility function will be lower with increasing flight height and could be interpreted as the higher the UAV flies, the less information is seen about the object since it will be farther away. The resulting path with higher flight height is longer, since there will be less information gained and the vehicle must therefore seek more orthogonality than before in order to gain more information.

3.5 Conclusion and Remarks

The use of information as a performance metric is reasonable since the more information about an object is better when trying to localize it, and the path of the moving platform is planned such that information is maximized.

The path is planned by optimization in each step. That is the reason the control signal varies a lot, and there are no penalty in choosing the control signal allowing the most greedy solution in terms of information. The optimization uses the *Matlab* optimization toolbox, which uses line search in the examples and there is a risk to get stuck in local optimum when the optimization is done over a non-convex or non-linear area. Introduce bounds on the control signal in the optimizer, reduces the number of local optimums but the problem does not disappear, instead one could accept any local minima as a good solution and make another optimization step, until the information is high enough.

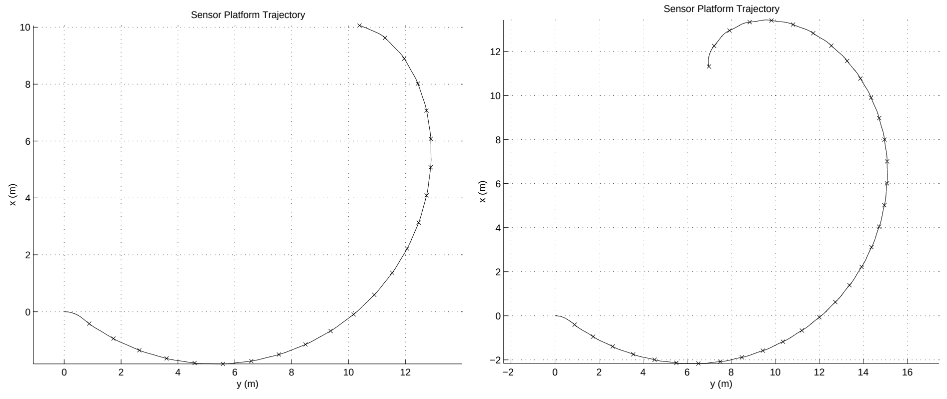

Figure 3.17: The trajectory for an UAV with different flight heights. Left: $z_s = 1\ m$. Right: $z_s = 10\ m$. Feature located in $(10, 10)$.

There is the difficulty of choosing the optimization parameters. The longer time horizon, the faster information gain, but it takes longer time to calculate and the there is the problem of planning the path close to an object with long time horizon. Therefore the short time horizon is chosen in the next chapter, where an area is about to be explored. The utility function which gives the best result with respect to information is the determinant, with the trace of the inverse as second best. By calculating the gradient analytically, the calculation time decreases, but the resulting path is about the same as for the numerical approximation of the gradient. However, the analytical gradient is hard to implement in an area search in the next chapter, and therefore the numerical approximation is chosen. The vehicle was also able to localize n objects, and the resulting path considered that all objects were to be localized simultaneously.

Finally the model was extended to three dimensions where the information is observed in a local coordinate system about the object and then transformed into global coordinates and thus avoiding the singularity in the linearization.

Chapter 4

Area Exploration

A common mission surveillance and reconnaissance is to search an area. The area is represented by a number of grid points, and each point has a certain information, so an information matrix could be constructed. The flight path is calculated from utility functions derived from the information matrix.

4.1 Area Search

The area search is usual done after a pre-defined path. The UAV is sweeping the area back and forth until the whole area has been covered. The usual area search principle is shown in Figure 4.1. This method is useful when scanning an area. However, this is not what this thesis is trying to achieve. Instead the vehicle is about to search an area where there are a number of unknown objects. By calculating the next step from an information matrix consisting of information about both the area and the objects, the UAV is able to "react" at objects on the ground. This can't be done if the path is pre-defined. The vehicle will now take the best possible step at the time horizon in terms of information. The reason is that the UAV should decide in real-time where it should go, and not just be able to follow some pre-defined path.

Consider for example a game of battleships. In those games, you are sweeping your opponents area, and if you hit something, you know that there is a ship and you continue to search that part of the area until the ship has been sunken. Instead of ships and sinking, change the terms to objects and localizing and that will be the case of the UAV.

4.1.1 Modelling the Area

An area is represented by a grid and each grid point is considered almost like an object. This means, if the area is about to be "explored", all grid points have to be "checked". The main difference between grid points and objects is that a grid point does not have to be localized in the same manner, the important thing is whether there is an object or not, in or close to the grid point. The information from an object was modeled in (3.26) by a (3×3) information matrix, that is by six states. One could imagine that the complexity of a discretisized area with N grid points will grow large to $6N$ states. Instead let each grid point be

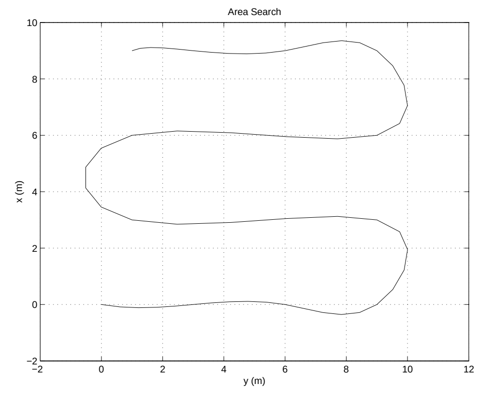

Figure 4.1: The conventional area search. Sweeping the area according to a pre-defined path.

represented by just one state, total N states, justified by that the grid points do not have to be localized in the same manner as the object, and hence keeping down the complexity and decreasing the calculation time. This one state has no physical interpretation. If there is something at the grid point, it will be taken into the model and the grid point are "checked", and when all grid points are checked, the whole area has been explored. A grid point is said to be checked, when the information about the grid point is higher than a pre-defined threshold value.

The state vector $\mathbf{x}$ consists of the three states representing the vehicle, that is $\mathbf{x}_s$ and the N states representing the grid points $\mathbf{x}_{gp}$ as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_s \\ \mathbf{x}_{gp} \end{bmatrix}. \tag{4.1}$$

The state vector consists now of $(3+N)$ states. The information matrix consists of the states representing the information and is constructed in the same manner as for n objects as

$$\mathbf{Y}_{area}(t) = \begin{bmatrix} \mathrm{Y}_{\mathrm{gp}_1}(t) & 0 & \dots & 0 \\ 0 & \mathrm{Y}_{\mathrm{gp}_2}(t) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \mathrm{Y}_{\mathrm{gp}_\mathrm{N}}(t) \end{bmatrix}, \tag{4.2}$$

where the index $\mathrm{gp_i}$ denotes grid point i. The noise in each observation is considered uncorrelated and the information matrix becomes an diagonal matrix. The dimension of the information matrix is $(N \times N)$.

4.1.2 Limited Sensor Range

The sensor is attached to the vehicle and has its limitations. It is reasonable that the sensor has a limited range, since it has a certain resolution, and it is needed to find a mathematical representation of this limitation. Grocholsky [4] argues that real world sensors typically exhibit an exponential variation in measurement uncertainty up to the maximum range. Introduce a distance r_{max}

which is the maximum range of the sensor and introduce an exponential penalty function, such that the covariance of the noise becomes

$$\mathbf{R} = \sigma_0^2 \exp\left(4.6(\frac{r}{r_{max}})^2\right), \tag{4.3}$$

and the observed information is proportional to the inverse of the covariance as

$$\mathbf{R}^{-1} = \frac{1}{\sigma_0^2}\exp\left(-4.6(\frac{r}{r_{max}})^2\right), \tag{4.4}$$

where σ_0 is the observed standard deviation at zero range, r is the distance between the sensor and the grid point and r_{max} is the maximum range of the sensor. The penalty function is the exponential function of r

$$\exp\left(-4.6(\frac{r}{r_{max}})^2\right),$$

which does not penalize at zero distance, and at distance r_{max} there will hardly be any information observed, and is plotted in Figure 4.2. Other functions could be used and for examples see Grocholsky [4].

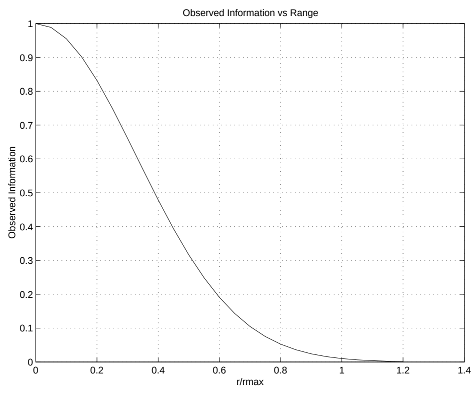

Figure 4.2: Exponential modeling of range dependent measurement errors in a mathematical model of a realistic bearings-only sensor.

4.1.3 Observed Information

The observation model for an area exploration is according to Grocholsky [4]:

$$\mathbf{z}(t) = \mathbf{T}(x, y) + \mathbf{v}(t), \tag{4.5}$$

where $\mathbf{T}(x, y)$ is a function describing the terrain characteristic and $\mathbf{v}(t)$ zero-mean uncorrelated Gaussian sequence with variance $\mathbf{R}$ as a function of the distance r as in (4.3).

This is a model and in this model, the terrain characteristics are said to be the states representing the grid points. For simplicity, these states could be observed directly by the camera. This means that there are no transformation between the measurements and the states. The reason is that the linear observation matrix $\mathbf{H}$ becomes simply the identity matrix according to (2.15) as

$$\begin{aligned}\mathbf{T}(x,y) &= \mathbf{x}_{gp} \\ \mathbf{z}(t) &= \mathbf{x}_{gp} + \mathbf{v}(t). \end{aligned} \tag{4.6}$$

The information observed from the grid points follows the usual update law as

$$\begin{aligned}\mathbf{I}(t) &= \mathbf{H}^T(t)\mathbf{R}^{-1}\mathbf{H}(t) = \mathbf{R}^{-1} \\ &= \frac{1}{\sigma_0^2}\begin{bmatrix} \exp\Big(-4.6(\frac{r_1}{r_{max}})^2\Big) & 0 & \dots & 0 \\ 0 & \ddots & & 0 \\ \vdots & & & 0 \\ 0 & \dots & 0 & \exp\Big(-4.6(\frac{r_N}{r_{max}})^2\Big) \end{bmatrix}.\end{aligned}$$

Each grid point is stationary, but as for stationary objects, a small process noise is included to improve numerical conditioning, as described in Section 3.1.2. The model for a grid point i is

$$\dot{\mathbf{x}}_{gp_i} = \omega(t), \tag{4.7}$$

where $\omega(t)$ is a zero mean Gaussian process with uncorrelated covariance $\mathbf{Q}(t)$. The observed information matrix $\mathbf{I}(t)$ is also an $(N \times N)$ matrix and the update law of the information matrix is given by

$$\dot{\mathbf{Y}}(t) = -\mathbf{Y}(t)\mathbf{Q}\mathbf{Y}(t) + \mathbf{I}(t).$$

4.2 Simulations

In the simulations a quadratic area (10×10) was created, and discretisized with a distance of one between each grid point. This means that there are $11^2 = 121$ points representing the area.

The complexity of the problem is now increased. There are now 121 states for the area plus the original three representing the vehicle, to a total of 124 states, to be compared with for example nine states when localizing an 3D object. There is again need to evaluate the performance of the different utility functions.

4.2.1 Determinant as Utility Function

The major problem with using the determinant as utility function is the numerical difficulties. Since the area consists of 121 grid points with low starting information, *i.e.* 10^{-3} for the states representing grid points, and the first determinant calculated would be in the range of 10^{-363}. This is considered to by zero by *Matlab*, and when the determinant is zero the utility function is also zero for all control signals and no minimum will be reached by the optimizer. To avoid this problem one could simply put 1 as starting information for each grid point, but there would still be numerical problems after a while. For example if the states has high values, the determinant would be close to infinity, therefore a normalization must be done. Recall the definition of entropic information from (2.7):

$$i(\mathbf{x}) = \frac{1}{2}\log[(2\pi e)^{-n}|\mathbf{Y}|].$$

Maximizing $\log|\mathbf{Y}|$ would also maximize the entropic information. However, with an even finer partition of the area, many grid points must be considered such that the $\log|\mathbf{Y}|$ could be infinite. Therefore, the second best utility function from Section 3.2.4 is tested.

4.2.2 Trace of the Inverse as Utility Function

When calculating trace of the inverse information matrix, a summation is done instead of a multiplication and the numerical difficulties could be avoided. The question arises whether a matrix is invertable when the numerical determinant is zero? Since the matrix in (4.2) is a diagonal matrix, it is simple to invert. The *Matlab* function *sparse*, see Appendix A, helps matrices containing lot of zeros, and therefore it is possible to invert such information matrix. With this utility function, there are no numerical problems, and is therefore used in the following simulations. Remember that the states representing objects or grid points are states in the information matrix, and it is just calculating the trace of the inverse instead of the determinant.

4.2.3 Simulating the Area Search

Figure 4.3 shows two area searches with different bounds on the control signal. The vehicle starts in origo with heading upwards and the time horizon is set to 1 s, otherwise the calculation time would be too long. For the same reason the control signal is parameterized into two steps $m = 2$. The grayscaled background is the information level of the surface, the brighter the more information. The area search is terminated when the information for all grid points is higher than a predefined threshold value. The path is about the same for the two simulations, but with the tighter bound on the control signal, the vehicle will not turn as sharp. The flight path will begin on the diagonal since there is the most information to gain, but when the sensor's maximum range reaches the outer rim of the area, it turns and continues the search in the direction of maximum information. Maybe the flight paths do not seem ”optimal” to the eye, since the vehicle crosses its own path four times, but they are (locally) optimal in the sense of maximizing information at the time horizon.

4.3 The Combined Model

It is now possible to derive a model which treats the problem of searching an area with n objects. The UAV should be able to decide where to go, by solving the optimal control problem while flying. The combined problem is of interest since it is a common problem in reality. The flight path must be calculated with respect to the grid points and the objects.

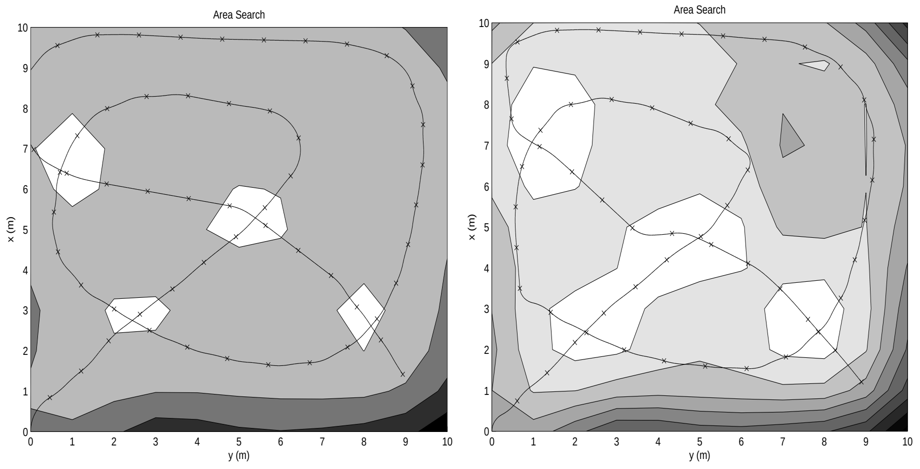

Figure 4.3: Trajectory of sensor platform for an area search with different bounds on the control signals. Left: $-1 \leq u \leq 1$. Right: $-\pi \leq u \leq \pi$

4.3.1 Global Information Matrix

The locations of the objects are unknown to the sensor initially and first when an object is in the sensor's range, the information matrix from the grid points are extended with the information matrix from the object. So the global information matrix would be:

$$\mathbf{Y}_{tot} = \text{diag}\begin{pmatrix} \text{Y}_{\text{gp}_1} & \ldots & \text{Y}_{\text{gp}_\text{N}} & \text{Y}_{\text{obj}_1} & \ldots & \text{Y}_{\text{obj}_\text{n}} \end{pmatrix}. \tag{4.8}$$

All Y_{gp} are scalar values, and the Y_{obj} are (3×3) matrices. The total information matrix is a diagonal block matrix containing lot of zeros, and by using the properties of such matrices it could be invertable and the trace could again be calculated as the utility function.

As mentioned earlier in (4.4), the observations of grid points are range dependent. This is also true for objects, and the same exponential behavior is applied to objects. The information observed from objects are done in local coordinates, with no information in the direction of the camera. The new range dependent inverse covariance matrix is the same as in (3.28) with the penalty function added as

$$\mathbf{R}^{-1}(t) \approx \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{\sigma_\theta^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_\phi^2} \end{bmatrix} \exp(-4.6(\frac{r}{r_{max}})^2). \tag{4.9}$$

4.3.2 Simulations

Since the grid points and the objects are modeled differently, there is a problem to get balance in the simulations. The first simulation is an area search where five objects has been placed out randomly. The path reminds of the paths in Figure 4.3, which could be explained by instead of concentrating on an object, there are more information to be gained by searching the unexplored area. It seems that the model for the grid points gives higher information than the model for the objects, and therefore it is more information to gain by searching the area.

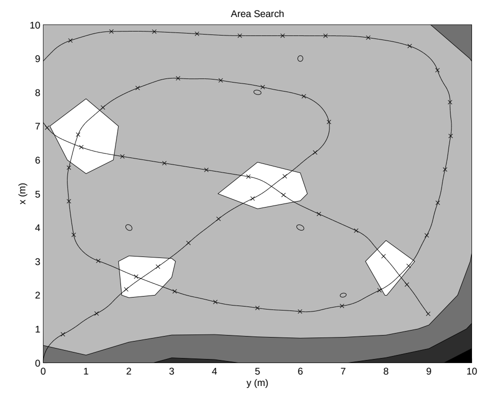

Figure 4.4: Trajectory of sensor platform for an area search with five objects. $-1 \leq u \leq 1$

However in reality, and especially in military applications, one could set different priorities for objects and grid points, since an unidentified object could be hostile. The priorities could be chosen such that the information values from the grid points could be compared to the information values from the objects.

As comparison a simulation with a weighing matrix $\mathbf{W}$ is including in the utility function as in [6] as

$$\mathbf{J} = \text{trace}\ (\mathbf{W}\ \mathbf{P}), \tag{4.10}$$

where $\mathbf{P}$ is the covariance matrix $\mathbf{Y}^{-1}$. The states representing the objects are weighted higher than the states representing the grid points. This could reflect the need to identify targets faster in surveillance and reconnaissance, since a target could be hostile. The resulting path is shown in Figure 4.5 and it is more desirable for the vehicle to localize a target within the sensor's range.

The question of how to choose weights arises. The user specifying the mission of the UAV could set weight depending of how much more interesting information from targets are compared to information about the area. If the mission is road surveillance, it seems as a good idea to weight targets more, otherwise the UAV must turn back to the end of the road to accomplish the localization of the targets.

4.4 Summary

In this chapter the model was extended such that the vehicle searched an area. Area searches are usually done after a pre-defined path, where the vehicle sweeps the area. This information-theoretic approach lets the vehicle optimize a trajectory at a time horizon, such that it can take a good step depending where it is and what it has already seen. In this way, the vehicle can act autonomously.

The combined model of an area with objects was also tried by solving the optimal control problem. The models for grid points and objects differs, and by

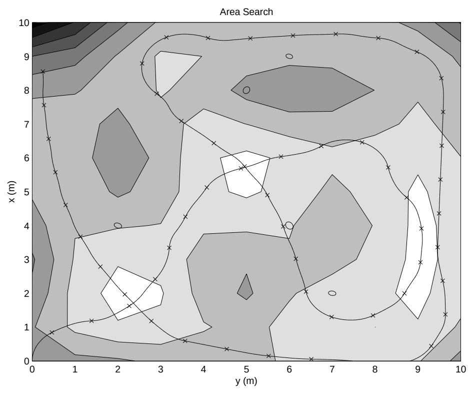

Figure 4.5: Trajectory of sensor platform for an area search consisting of five objects with weighted utility function. $-1 \leq u \leq 1$

introducing weight matrices, these weights or priorities could be set by the user according to the mission. A simulation for an entire area exploration takes some time, approximately a lunch break and it is therefore undesirable for real-time applications, but the principle works. Instead of using *Matlab's* optimization toolbox, one could write your own optimizer for this problem and reduce the calculation time.

Chapter 5

Spline Optimization

The simulation model has so far one big simplification compared to the reality. In the *SIREOS* project the sensor is a gimballed camera, more in Chapter 6, and objects and grid points could only be observed if within the field of view of the camera. The camera has two degrees of freedom, and is therefore controlled by two control signals. The optimal control problem would now be over three parameterized control signals, which implies a heavy computational load. An alternative method is to find the optimal trajectory directly by placing points who together creates a spline. In this chapter, the method with splines is described and the resulting trajectory is compared with the previous paths. Later on in Chapter 6, the spline method is extended with a trajectory representing the camera motion and the resulting optimization would be two trajectories, describing fully how both the vehicle and camera should move respectively.

5.1 Properties of Splines

5.1.1 Advantages

The main advantage with splines will be shown first after the camera is introduced, which is that the splines represent the trajectories directly. The trajectories do not longer have to be calculated by sending in control signals. The camera is controlled by two control signals and combined with the control signal for the UAV, the optimization is done over three parameterized control signals if the problem is solved like an optimal control problem. This implies a heavy computational load. Instead one could set up one spline representing the vehicle's trajectory and one spline representing the camera motion, which would hopefully decrease the computational time, since the resulting trajectories are given directly. In this chapter, there is just one spline for the vehicle's trajectory and the camera is introduced in Chapter 6.

Another advantage is that it is now possible to force the vehicle to pass certain points, by making them spline points. In military applications, one could imagine that there are some locations more strategic than other, like an hill, and it is necessary to know if there are any hostile targets at that hill. By including the location of the hill in the spline, the UAV is guaranteed to look at that point.

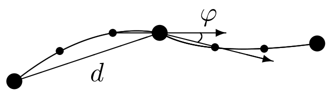

Figure 5.1: Geometrical bounds on spline. The bigger points are the spline points and the smaller points interpolated points in between. The distance d between the spline points is bounded, as well as the angle φ in the intersection of two splines.

5.1.2 Disadvantages

The main difficulty is how to place the spline points. The points must be placed such that it is possible for the UAV to fly the resulting trajectory. The spline points must be placed in such way that the control signals corresponding to the path are bounded. This was handled by introducing geometrical bound on the spline points, as shown in Figure 5.1. The spline points are the bigger points, and the smaller points are the interpolated points in between. The first bound is that the spline points must be within a certain distance d from each other, corresponding to a constant velocity. The second bound is that the angle φ in the intersection of two splines, calculated about a spline point and the nearest interpolated points, is limited which corresponds to a bounded control signal. These bounds are nonlinear but the *Matlab* function *fmincon* can handle such bounds.

Another difficulty is that the first spline point will affect the trajectory the most, and if it is somehow misplaced by for example too low tolerance in the optimization, the optimal final point will never be reached.

5.2 The Single Platform Example

This illustrates the first example when localizing one object on the ground. It is the same example as in Chapter 3, but with splines instead of control signals. The information is in three dimensions, with flight height 1 m, and it would be the same situation as the left picture of Figure 3.17.

5.2.1 Optimizing over Splines

The feature and sensor is modelled as in Chapter 3. The states representing the vehicle $[x, y, \psi]^T$ are removed from the model and the spline is used. The position could be taken directly from the spline, since it is the trajectory and the angle ψ could be calculated by using the spline points.

Instead of having a parameterized control signal as variable in the optimization process, a number of points is used which should be placed according to a given start point and the direction of the previous spline. The output of the optimizer is the spline points, who together build up the resulting trajectory. The problem with several minimums of the utility function is still there, and the resulting trajectory would not be optimal, but hopefully good enough. Then the states are calculated by solving the update law of the information matrix, where observations are made at the spline points and interpolated points in between. The nonlinear bounds on the spline are introduced in order to get a reasonable flight path.

5.2.2 Simulations

In the first simulations, the parameters describing the spline are

$d = 1\ m$	Distance between the spline points constant.
$\varphi = 30°$	Angle bound between the splines.
$M = 3$	Number of spline points in each optimization step.

And the other simulations parameters are

$(x_f, y_f) = (10, 10)\ m$	Feature location
$\mathbf{Y}(0) = \mathrm{I}_3 \cdot 10^{-3}$	Low starting information
$\mathbf{Q} = \mathrm{I}_3 \cdot 10^{-6}$	Process noise
I_3	A (3×3) identity matrix
$\sigma = 2.5° \Rightarrow \mathbf{R} = \sigma^2$	Observation noise

The distance between the spline points is kept constant to resemble constant velocity, and the angle bound is there to avoid large control signals. $M = 3$ means that three spline points are placed and they are building up the trajectory for the vehicle. The number of spline points and how long they may be placed from each other are setting the length of the trajectory which resembles the optimization time horizon. The information at the final spline point, which is depending on the trajectory, is building up the information matrix from where the utility function is calculated.

The resulting trajectory is still shaped as a spiral since the information gain is most favorable orthogonal to the last observed direction and inverse proportional to the squared distance. The spline optimization gives about the same result as for the control signal optimization, which is correct since the same information model are used.

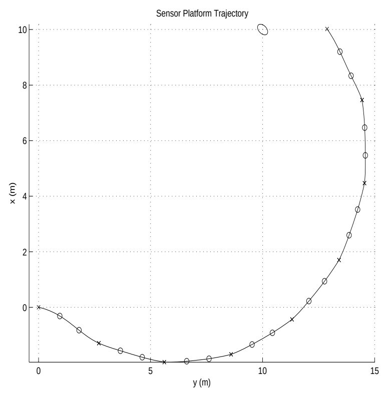

Figure 5.2: The optimal trajectory with constant distance 1 m between the spline points. Each spline point is marked with an 'x' or a 'o', the 'x' points indicates a new optimization.

The distance between the spline points could vary and is shown in the left part of Figure 5.3. The distance is set to 2 m and 3 m respectively. The situation resembles much of the effect of different optimization time, and there is a difficulty planning the number of optimization steps.

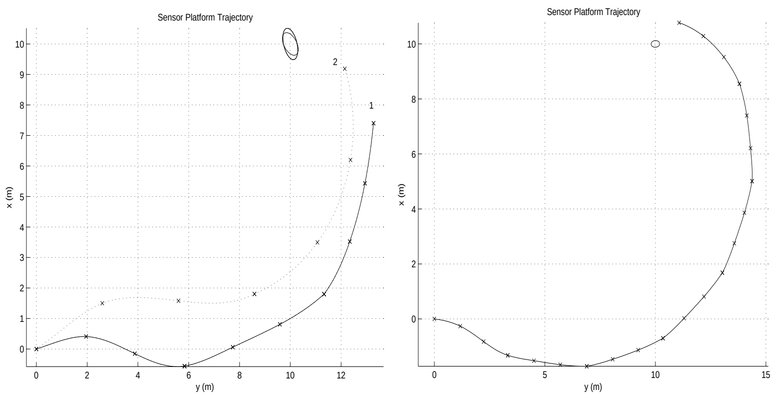

Figure 5.3: Left: The resulting trajectory for distance between spline points: 1) 2 m and 2) 3 m. Right: The resulting trajectory for a path with distance varying 0.8 and 1.2 m between two spline points. All spline points are marked with an 'x'.

The simulations are so far very similar to the simulations in Chapter 3. Arguing for the use of splines, it is very easy to place the spline points with different distance in between. The next simulation is the same as for in Figure 5.2 but the distance is allowed to vary within twenty percent, that is the spline point distance is between 0.8 m and 1.2 m. This would be harder to implement in the original model. The bound on the angle is tightened to 20° in order to avoid some local minima, the same problem as in Section 3.2.1. The resulting path is shown in the right part of Figure 5.3 which requires less optimization steps than for the simulation in Figure 5.2 since the vehicle will take longer step far away from the feature and could use shorter step if necessary.

A problem with splines is to overplan, which means that a spline is too long and spline points must be placed far from the feature. By having the length between the spline points vary with the distance to the target as seen in Figure 5.4 but not the angular bound, there is no risk to overplan and instead the vehicle will localize the object faster. This opens up for two interpretations. The first is that the vehicle will no longer travel with constant velocity, but instead travel fast when it is far away and slow when it is close to the object. The number of observations is the same between any two spline points and more observations are made closer to the object, which is reflecting a real life situation. The other interpretation is that the velocity is still constant and instead the number of observations varies with the distance to the target. The closer the target, the more observations and it is also reasonable since nearer the target, the better are the images.

A simulation like in Figure 5.4 takes about one minute, which is faster than the optimal control problem. This is promising since the main reason for introducing splines was the computational time.

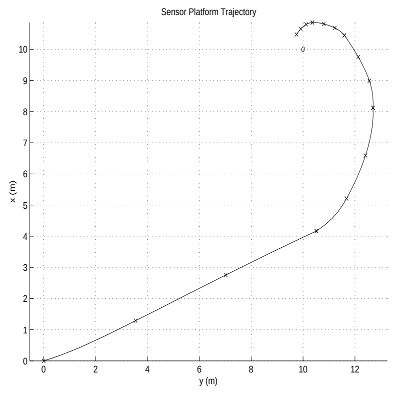

Figure 5.4: The optimal trajectory with varying distance in each subspline.

5.3 Discussion

The method of using splines is appropriate since the resulting trajectory is about the same as for optimizing over control signals. Spline points are very flexible since it is very easy placing the points at different distance between each other.

Splines are also preferable if the user knows that the UAV must pass a certain point. For example, if it is known a priori that there are a target at a certain location, the path could be planned such that that point is passed. Then there is no risk that the UAV will miss those points. This was not simulated, but by adding a point in the spline that is fixed, then the vehicle will pass this point.

The user chooses the distance between the spline points and the angle bound. There is still the problem with many optimum in the utility function, and the number of local optimum increases with more spline points or less tighten bounds. Most choices of parameters will lead to local optimums. At this moment, the solution is affected by the starting values and the parameters, and the user has to choose carefully until the optimization method is changed.

Chapter 6

The Gimballed Camera

In this chapter the states representing the camera are introduced. Objects are observed if seen by the camera, and the resulting path will not only be the trajectory for the vehicle, but also a trajectory of what the camera sees. Splines are used to create the trajectory for the camera.

6.1 Experimental Sensor System

The sensor system is a gimballed camera with an IR camera and a CCD video sensor as in Figure 6.1. The camera is able to rotate 360° about its own axis, called the pan angle, and the tilt angle is approximately $+10/-90°$. The angles are shown in Figure 6.2.

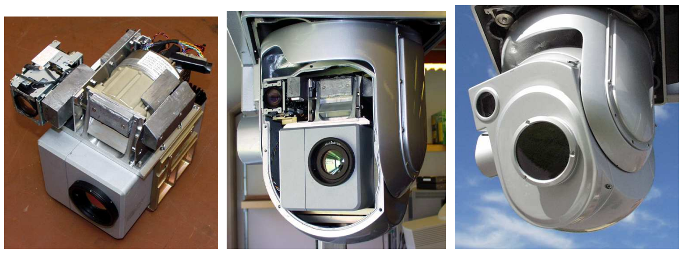

Figure 6.1: The gimbal system. Left: Inner gimbal consists of an IR camera and a color CCD. Middle: The gimbal with demounted front. Right: Gimbal with mounted front. [8]

6.1.1 Modelling the Camera

The states $\mathbf{x}_c = (x_c, y_c)^T$ representing the camera are the states which the camera points at on the ground. The spline representing the camera would be the points on the ground where the camera looks at. Originally the camera had a certain visibility set to a distance from $(x_c, y_c)^T$, and all points outside this circle were not visible. This may be true in real life but there was a problem in the optimization, since if the visibility is zero, the gain of information is zero and no minimum will be found since the optimizer will not have a gradient and

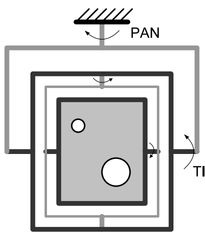

Figure 6.2: Schematic view of the gimballed camera. The four joints are marked, outer joints are called pan respectively tilt angle. [8]

will not know in which direction the visibility increases. Therefore a reasonable visibility function is a function that is one inside a certain distance r_{high}, and outside decrease exponentially and different candidates are seen in Figure 6.3 and the function in equation form in (6.1).

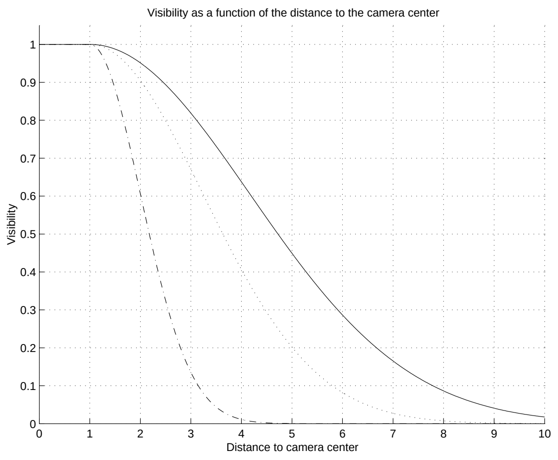

Figure 6.3: The visibility as a function of where the camera points at. Visibility inside the radius $r_{high} = 1$ is one, and outside decreasing as $e^{-const(r-r_{high})^2}$. The solid line has $const = 0.05$, the dashed $const = 0.1$ and the dash dotted $const = 0.5$.

$$\begin{aligned} r &= \sqrt{(x-x_c)^2+(y-y_c)^2} \\ \text{visibility} &= \begin{cases} 1 & \text{if } r < r_{high} \\ e^{-const(r-r_{high})^2} & \text{o.w.} \end{cases} \end{aligned} \tag{6.1}$$

6.1.2 Limitation of the Camera

The camera is controlled by two control signals, controlling the pan respectively the tilt angle. The optimization is by using splines and to represent the limitation, bounds have to be constructed. Each spline point is a point where the camera looks at, and the resulting spline would be a spline of camera point vectors.

The range of the camera is taken into account in the model as a bound. The spline points of the camera may not be placed any further away from the vehicle than the maximum range r_{max}, since the camera is attached to the vehicle in $(x_s, y_s)^T$ and the camera points at $(x_c, y_c)^T$. This means that if the vehicle is far away from the object, it will hardly get any information about it and must therefore move towards it until it could begin the information gain.

The next bound is that the spline points of the camera must be at a certain distance from each other. The differences from the vehicle trajectory are that the camera is must faster and may move in any direction, and therefore the points creating the camera spline do not have such tight bound and the distance between two camera spline points could be more varying.

6.2 Localizing Objects on the Ground

Planning the path for localizing one object with a camera is not very interesting, since the camera will always point in the direction of the object. With two objects the camera must choose to look at either object. Once again, the utility function chosen will affect the solution.

6.2.1 Using all Spline Points

The splines for the trajectory as well for the camera are optimized by two points each, and the information observed is taken in the direction of the camera. Following the same method as previously, the utility function consist of the information at the final spline point. The spline points placed before the final spline point will have more effect if considered in the utility function. Since two points are building up the spline, the first point at t_{f-1} is also considered in the utility function (6.2). The reason is to get rid of some local optimums in this very complex problem. Another idea when using splines is to add an extra point with less constraints, and the information is maximized by a sum of the real spline points and this extra point. The latter could be interpreted as an estimation of the future information and the resulting path would be in the direction somewhat between what would be good now and not so bad in the future. The utility function for two real spline points placed in t_{f-1} and t_f would be

$$\mathbf{J} = a\ \mathbf{J}(t_{f-1}) + b\ \mathbf{J}(t_f) + c\ \hat{\mathbf{J}}(t_{f+1}), \tag{6.2}$$

where a, b and c are weights and $\hat{\mathbf{J}}(t_{f+1})$ is the estimation of the future information gain. In the simulations, the weights were set to $a = 1, b = 2$ and $c = 1$, so the greater influence is from the real final spline point. The extra point is not used in the update of the states and is only there for the optimization.

In the simulations the task is to localize two objects in the plane, same as in Section 3.3.1, with the difference that the information model is in three dimensions. The objects are located at $(10, 0)$ and $(10, 10)$. The vehicle starts in origo with heading rightwards and the flight height is constant, $z_s = 1\ m$, throughout the simulations. Into each optimization step, the initial trajectory guess is from the current vehicle location to a point between the objects, and the initial guess for the camera trajectory is the last point where the camera looked at, initially straight down. The optimization is continued until both objects are localized. If an object is localized, it is removed from the information matrix and in the next optimization step the information matrix is constructed just from the non-localized object.

6.2.2 The Utility Function

The utility function affects the solution, maybe more in this case than ever. The two main candidates are the determinant and the trace of the inverse from (2.9) and (2.11), given again in terms of eigenvalues

$$\begin{aligned} \max\ |\mathbf{Y}| &= \max \prod_{i=1}^{n} \lambda_i \\ \min\ \text{trace}\,(\mathbf{Y}^{-1}) &= \min \sum_{i=1}^{n} \frac{1}{\lambda_i}. \end{aligned}$$

The determinant maximizes the product of the eigenvalues, and could therefore compensate a low eigenvalue with a high. In other words, the determinant can be large by just focusing on one object. For the trace of the inverse to be minimized it is more important to affect all eigenvalues and must consider both objects. These effects are seen in the following simulations.

6.2.3 Simulations

The simulations are done with first the determinant as utility function in Figure 6.4, and then with the trace of the inverse as utility function in Figure 6.5. Both objects are in the range of the camera, otherwise the camera will just look at objects inside its range. The trajectory is plotted in solid line, where each spline point is marked. The camera spline is plotted in dotted line. The camera spline is the points on the ground where the camera looks at and the trajectory is at the flight height, and these splines are plotted in the same picture. The 'x':s in the plot are all points building up the spline, including the interpolated points between the spline points. The constant of the penalty function in (6.1) was set to 0.05. The distance between the spline points was set to 1 m and the distance between the camera spline points was allowed to vary between 0 and 10 m.

For both cases, this optimization problem is very complex. There are several local maximum for the information, and the method of line search is far from the best, but the resulting path from the different utility functions could be seen.

The starting values of the information are small, set to 10^{-3}. These values are the initial eigenvalues of the information matrix. The information matrix consists on two (3×3) block matrices representing each object, and is therefore a (6×6) matrix. If only one object is observed, the three eigenvalues corresponding to that object become high, and the other three eigenvalues will have their starting values. If both objects would be observed partially, there will roughly be six low eigenvalues. For simplicity, lets say:

$$\lambda_{start} = 10^{-3},\ \lambda_{low} = 10^{-2} \text{ and } \lambda_{high} = 1,$$

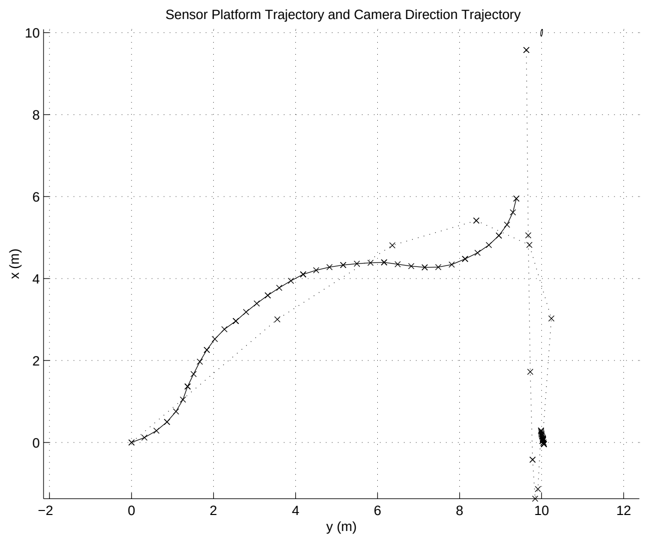

Figure 6.4: Sensor Platform Trajectory (solid) and Camera Direction Trajectory (dashed), utility function is determinant.

where the λ_{high} is the eigenvalue of an observed object when the camera focuses on that object. The λ_{low} is the eigenvalue when the camera looks somewhere between both objects, since the visibility function in Figure 6.3 decreases with the distance between where the camera points and the object. The λ_{start} is the starting value, and is still the eigenvalue for a object if not observed.

The determinant is a product of the six eigenvalues, and the idea is to maximize the determinant. Multiplying six λ_{low} is lesser than multiplying three λ_{high} and three λ_{start}:

$$(\lambda_{low})^6 = (10^{-2})^6 = 10^{-12} < (\lambda_{high})^3(\lambda_{start})^3 = (10^{-3})^3 1^3 = 10^{-9}$$

That is the reason that the determinant tries to see one object at a time, as seen in Figure 6.4.

For the trace of the inverse, the utility function will be a sum of the inverse eigenvalues, and the idea is to minimize the trace of the inverse. If it would focus on one object, there will be three λ_{high} and three λ_{start}, and the sum of the inverse eigenvalues would be higher than the sum of six λ_{low}:

$$\frac{6}{\lambda_{low}} = \frac{6}{10^{-2}} = 600 < (\frac{3}{\lambda_{start}} + \frac{3}{\lambda_{high}}) = (\frac{3}{10^{-3}} + \frac{3}{1}) = 3003.$$

That is the reason why the first observation is somewhere between the objects, since it will observe something from both objects. This effect is seen in the Figure 6.5. This example illustrates why the paths in the two simulations differ. But with other starting values, other resulting paths would be obtained.

After the first object has been localized, it is removed from the model and the starting values for the camera spline points is where it looked last time, that is in the direction of the object which has just been localized. In the simulation with the trace of the inverse as utility function, there is a problem after the first

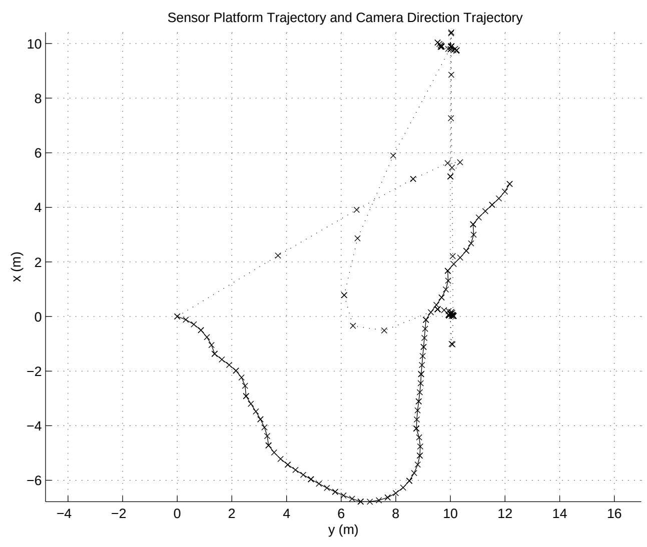

Figure 6.5: Sensor Platform Trajectory (solid) and Camera Direction Trajectory (dashed), utility function is trace of the inverse.

object in $(10, 0)$ has been localized. The distance to the other object in $(10, 10)$ is quite long, and the information observed from it is so small, that it is hard to find an optimum. When no optimum is found, the vehicle travels in the direction of the initial guesses, that is in the direction of the non-localized object, and the vehicle is just getting closer. The optimization is not perfect, this is undesirable but with smart initial guesses, as in the direction of the non-localized object, the vehicle is getting closer and the gained information becomes high again.

6.3 Summary

Including the camera into the model increases the complexity of the optimization. Not only has the trajectory of the vehicle to be optimized, but also the trajectory consisting of how the camera should point. These trajectories are bounded to each other, since the spline points of the camera vector may only be placed up to a certain distance from the spline points for the vehicle trajectory. Information gain through observation is depending on where the camera looks at, and farther away from the object gives less information. For information to be maximized, the object must be within the camera's field of view, orthogonal and close to the object. With two objects, there are several possibilities for the optimizer, that is several local maximum for information and the path in the simulations may not be the best one.

It is more interesting to see the effects of the different choices of utility functions. The determinant manages to find a path somewhere between the objects and sweep the camera at the objects. This is helped by the initial guess for the trajectory, which is in between the objects. When localizing two objects, it seems reasonable to fly somewhere in between, and this solution is found by the optimizer even though there are several optimum in the utility function.

The weights in (6.2) of the utility function has also effect on the solution and they were chosen such that the main part of the solution is at the time horizon t_f. The time horizon corresponds to the last of the spline points that will build up the trajectory. An extra point is added to include an estimate of the information to be gained in the future, so the vehicle is allowed to take a step in some direction that may not be the best at the time horizon, but better in the future. Different weights will give different trajectories and they could be chosen by the user on what is most important.

The trace of the inverse as utility function gives a quite different path compared to the determinant. Both simulations have the same threshold for localizing objects and the path when using the trace of the inverse is longer, which means that the number of optimizations is higher and the path would be less good. This implies that the determinant is better as utility function, but in Chapter 4, the determinant was not chosen due to numerical difficulties when having a large information matrix. The conclusions to be drawn are that it is hard to choose one utility function as the best one. Maybe one could use both utility functions, since it is just about calculations, and find something in between.

There are many suggestions for the future. The camera model is very simple and it should be improved. Another optimization method is preferable where the global or a sufficient good local optimum will be found. But it is interesting that the vehicle and the camera motion trajectory could be planned with respect to each other.

Chapter 7

Summary and Conclusion

In this chapter the results from the report are summarized, conclusions are drawn and some suggestions of future work are given.

7.1 Summary

The flight path was calculated from an information-theoretic approach. The problem was first formulated and solved like an optimal control problem, and then solved by using splines. The applications are in UAV surveillance and reconnaissance.

7.1.1 Information-Theoretic Approach

For both methods, the optimization is done by setting up an information matrix constructed from those states representing the objects or grid points to be localized. The use of information is very elegant, since the UAV is able to calculate the maximum information at the optimization time horizon. The evolution of the information matrix is depending on the linearization of the system equations and observations, and with incorrect modelling, there is a risk that the resulting path would not be that good.

There are two alternatives for path planning of an autonomous vehicle. Either by a pre-defined path or by on-line planning, where the latter means that the path is planned interactively by the vehicle. The information-theoretic approach is one method of on-line path planning.

Making no claims that on-line path planning is better than pre-defined paths, like the conventional area search principle in Chapter 4, there are cases where the on-line path planning is preferable. For example, the vehicle could adapt when seeing objects, and change its path autonomously. It is not necessary to know everything in before, which is required when using pre-defined paths.

According to Grocholsky [4], the information-theoretic approach has several advantages. First of all, it is an intuitive correct measure when having a bearings-only sensor. Information is also connected to probability distributions, which are used to describe the uncertainties of the objects. When trying to localize an object, information is appropriate since the higher information about an object, the better is the localization. The information-theoretic approach is also suitable if several sensors on different platforms are used to search an area. With communication between the sensors, information measures such as mutual information could be used, and the solution would be how all vehicles and sensors should move respectively.

7.1.2 Optimal Control

By formulating the path planning problem as an optimal control problem, one must first calculate the optimal control signals as those control signals that maximizes information. By sending in these control signals in the UAV, the optimal flight path will be obtained. The path is optimal for each optimization step, that is at the time horizon, but the optimum could be a local optimum since it is not a convex problem. Different parameters were discussed, such as different utility functions, time horizon and prior information. The method was tried on both a simple case by identifying just one object, then extended to n objects, as well as for an entire area exploration, and worked satisfactory in all cases as shown in Chapter 3 and Chapter 4.

7.1.3 Spline Optimization

Spline optimization is simpler in the sense of finding points, since the resulting trajectory is found directly. This method is extended in Chapter 6 with the camera model. Two splines are now created, one for the vehicle and one for the camera and the solution is how the vehicle and camera should move respectively. The natural extension is using splines for an area exploration and compare those results with the optimal control solutions, which is saved for future work.

7.2 Discussion

During the thesis several methods were tried. There is a problem of how to choose the optimization parameters. The two most important choices are which utility function to use and how to formulate the problem.

7.2.1 Utility Function

As seen in the simulations, different utility functions result in different paths. It is hard to choose one utility function as the best in all situations. All utility functions operates on the eigenvalues of the information matrix, and information is high when the eigenvalues are large. In the example with only one object in Figure 3.10, the resulting paths are similar. But when the matrix increases with more objects, there are some differences as in the camera example in Figures 6.4 and 6.5. Grocholsky [4] has only compared the paths when identifying one object, not with several objects. There is nothing that says that either the determinant or the trace of the inverse is the best in all cases. Instead one could set up a determinant for each object and sum up these, to avoid the numerical problems when taking the determinant of a large information matrix containing lots of small values.

7.2.2 Optimal Control vs. Splines

Even though the optimal control solution worked well, it was not sufficient since the camera has not been introduced. The ”real” optimal control problem is more complex and should be formulated over three control signals, two for the camera and one for the vehicle. This seemed like a heavy computational load and therefore the spline optimization was introduced. However, it would have been interesting to examine the solution with optimal control and compare the result with the spline optimization.

What could be compared are the solutions to similar problems, with as many optimization parameters the same as possible. The calculation time for the simplest case when just identifying an object is about half the time when using splines. But the calculation time is depending on many optimization parameters. For example, by reducing the number of parameterized control signals in the optimal control problem, the calculation time will also be decreased. But the method of splines seems promising though, however the splines was not that much faster. There is still the problem of many local maximum, which should be solved by using another optimization method, yet not decided how.

7.2.3 Comments on References

The main problem with several local optimums is never discussed in Grocholsky [4], and one could think that the optimization functions will find the global optimum. But as it is now, the optimization parameters affects the solution. For example, if the starting values are set such that only local optimum could be reached by the line search method, the global optimum will never be found. The wanted method is that the optimizing procedure will find the global minimum for any kind of starting values.

7.3 Conclusion

Information-theoretic approach is a method suitable for the cases of path planning where the vehicle is about to make its own decisions. But the problem is quite complex, since it is about finding optimum where there are several local optimums, and there is no rule that guarantees the best solution. The method of splines is promising since the calculation time is lower than for the optimal control problem, and spline could solve a complex problem like the optimization over both the vehicle and camera trajectory.

7.4 Future Work

There are a lot to do in the future. First of all, another optimization method should be used such that it is possible to find the global, or a sufficient good local minimum. Other things that could be done later on are for example planning the path with obstacles like trees or buildings which make some targets only visible if observed in a certain direction. It is also possible to optimize with other criterions like tactical constraints such that stealth. A first scenario of the area exploration is road surveillance, where the area is the road and the task is to search through the road for targets.

In order to get the simulations more realistic the uncertainty of the vehicle must be considered, since the UAV position is estimated by GPS/INS. This will lead to a quite complicated model and is saved for future work.

The camera model is very simple and should be improved. At this moment the camera has a certain visibility of where it looks, but the closer the camera, the better are the images and this has not been considered.

Eventually the UAV has to be tested and should be able to calculate its next step in real-time. A way to decrease the computational time is instead of using *Matlab's* optimization toolbox write your own optimizer for this special case.

Appendix A

Matlab functions

fminunc

```
u = fminunc([function, u0, options, x]);
```

Where

`u`	The solution to the minimization, the optimal control signal u.
`function`	The function to be minimized by u.
`u0`	Initial guesses for u.
`options`	Optimization parameters defined in *optimset*.
`x`	Extra parameters into the optimization, here the states $\mathbf{x}$.

fmincon

```
u = fmincon([function, u0, A, B, Aeq, Beq, u_low, u_high, nonlincon,
options, x]);
```

Differs from *fminunc* with

`A,B`	Minimizes the function with respect to inequalities $\mathrm{A}u \leq \mathrm{B}$.
`Aeq,Beq`	Minimizes the function with respect to equalities $\mathrm{A}_{eq}u = \mathrm{B}_{eq}$.
`u_low`	Lower bound of control signal u.
`u_high`	Higher bound of control signal u.
`nonlincon`	Minimizes the function with respect to nonlinear constraints.

optimset

`options = optimset('fminunc');` or `options = optimset('fmincon');`

Used to set optimization parameters like

`options = optimset(options,'TolX',tol);`	Tolerance on constraints set to value *tol*
`options = optimset(options,'MaxIter',iter);`	Maximum number of iterations *iter*

ode45

```
[T,X] = ode45(@xdot, tspan, X0, options, u);
```

Where the `ode45` solving differential equations with

`@xdot`	The system equations to be solved $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$
`[T,X]`	The solutions $\mathbf{x}$ at time t in the vectors `[T,X]`
`tspan`	The time interval where the differential equations are to be solved
`X0`	Starting values of $\mathbf{x}$
`options`	Solution parameters specified by *odeset*
`u`	Control signal as extra parameter in the system equations

sparse

```
S = sparse(S);
```

Converts a sparse or full matrix into sparse form by squeezing out any zero elements.

Bibliography

[1] Bertsekas, Dimitri P., *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont 2000.

[2] Böiers, Lars-Christer, *Lectures on Optimization*, KFS AB, Lund 2001.

[3] Glad, Torkel and Ljung, Lennart, *Reglerteori. Flervariabla och olinjära metoder*, Studentlitteratur, Lund 1997. In Swedish.

[4] Grocholsky, Ben, *Information-Theoretic Control of Multiple Sensor Platforms*, Ph.D. Thesis, University of Sydney 2002.

[5] Manyika, J. and Durrant-Whyte, H., *Data Fusion and Sensor Managment: A Decentralized Information-Theoretic Approach*, Ellis Horwood, London 1994.

[6] Mihaylova, L., De Schutter, J., and Bruyninckx, H., A Multisine Approach for Trajectory Optimization Based on Information Gain, Katholieke Universiteit Leuven, Heverlee 2003.

[7] Sciavicco, L. and Siciliano, B., *Modeling and Control of Robot Manipulators*, Springer-Verlag, London 2003.

[8] Skoglar, Per, Modeling and Control of EO/IR-gimbal for UAV surveillance applications, Scientific Report, Swedish Defence Research Agency, Linköping 2003.