Frederik Stendahl Leira

Object Detection and Tracking With UAVs

A Framework for UAV Object Detection and Tracking Using a Thermal Imaging Camera

Thesis for the degree of Philosophiae Doctor

Trondheim, January 2017

NTNU Norwegian University of Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

NTNU

Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

ISBN 978-82-326-2010-4 (printed version)
ISBN 978-82-326-2011-1 (electronic version)
ISSN 1503-8181 ITK Report: 2016-14-W

Doctoral theses at NTNU, 2017:1

Printed by Skipnes Kommunikasjon AS

Summary

In recent years there has been a drastic increase in the commercial availability of small UAVs for civil and commercial applications. These applications includes, but are not limited to, inspections of structures, search and rescue, monitoring and surveillance of oil spills and autonomous border control. All of these applications emphasize the UAV's role as a remote sensing platform with the main focus on visual sensors and data collection, and also includes an object detection and tracking component. Object detection and tracking in UAVs using visual sensors can generally be divided into two subproblems, where the first problem is that of controlling the path and orientation of the UAV in such a way that the UAV is able to capture useful data with its on-board sensors. The second problem is that of making the UAV able to automatically detect and track objects of interest using the UAV's on-board sensors and autopilot. The former subproblem is considered a problem of path planning, while the second subproblem is considered a problem of machine vision.

This monograph is motivated by the lack of a commercial agile and adaptable object detection and tracking framework for UAVs equipped with a visual sensor. Existing solutions are often application specific and implemented ad hoc, resulting in rigid object detection and tracking systems which are difficult to adapt and reconfigure for use in other similar applications. This thesis presents a novel framework for object detection and tracking in UAVs equipped with a visual sensor where the focus is to make it as configurable, modularized and as hardware independent as possible.

Chapter 2 presents a modularized object detection and tracking framework consisting of a path planner, machine vision and object handler modules. Each module solves a specific sub-problem of the overall object detection and tracking problem, and each module has a wide range of applicable solutions which could be implemented and mixed with the remainder of the system. Chapter 2 further describes a software toolchain which provides a bridge and abstraction layer for the object detection and tracking framework's architecture and the UAV's on-board hardware

components. This makes the software toolchain compatible with the idea of easily being able to configure the individual modules and hardware of the overall UAV object detection and tracking system. This is exemplified by the implementation of the hardware and software modules of the object detection and tracking framework in two different UAV platforms.

In Chapter 3, a path planner module based on model predictive control (MPC) is developed. This controller uses a kinematic model of the UAV with certain constraints on the UAV's motion to generate a feasible object tracking path for the UAV. That is, the MPC assumes that the visual sensor is mounted in a gimbal with two degrees of freedom and supplies the on-board autopilot with waypoints and gimbal control input which enables the UAV to track an object or group of objects. The gimbal is controlled in a way to ensure that the gimbal is pointing towards the tracked object's estimated position. The MPC is demonstrated through field experiments in Chapter 3 and 5 to be able to successfully make both of the UAV platforms developed in this thesis able to track objects.

Chapter 4 develops a solution for the machine vision module as an object detection, recognition and tracking module compatible with the developed object detection and tracking framework. The module enables the UAV to perform on-board image processing of thermal images in order to automatically segment the images into either background or objects of interest. Using the on-board telemetry data indicating the UAVs position and attitude coupled with the segmented images, the module is able to use a Kalman filter to track and estimate the position and velocity of the detected objects of interest.

Chapter 5 demonstrates that the developed object detection and tracking framework is agile and adaptable by replacing the machine vision module of the system with a module more suitable for use in a sea ice management application. Furthermore, the nominal tests of the alternative modules developed in this chapter show promising results for the use of UAVs in the area of sea ice management.

A full scale test of the object detection and tracking framework where all of the modules of the framework are activated and interacting simultaneously is still a test that has to be conducted, although the nominal testing in Chapter 5 demonstrates the viability of the framework and its implemented modules.

Preface

This monograph is submitted in partial fulfilment of the requirements for the degree of philosophiae doctor (PhD) at the Norwegian University of Science and Technology (NTNU).

The work presented herein has been completed at the Centre for Autonomous Marine Operations and Systems (NTNU AMOS), Department of Engineering Cybernetics, in the period June 2013 to October 2016. My supervisors have been Professors Thor I. Fossen and Tor Arne Johansen.

Acknowledgements

Funding was mainly provided by the Research Council of Norway through the Centres of Excellence funding scheme, Project number 223254. Partial funding was provided by "Autonomous Unmanned Aerial System as a Mobile Wireless Sensor Network for Environmental and Ice Monitoring in Artic Marine Operations" - innovation project funded by the Research Council of Norway (235348) in collaboration with Radionor Communications, Kongsberg Seatex and Maritime Robotics.

First of all I would like to thank Professors Thor I. Fossen and Tor Arne Johansen, my supervisors, who have guided me through the PhD. It is safe to say that without your enthusiasm, suggestions, wisdom and input, this monograph would have never been. I am forever grateful that you entrusted me to take on this PhD project.

I would like to express my deepest gratitude to Lars Semb and Pål Kvaløy for their tireless efforts to make the field testing of my work a reality. Further I would like thank Professor João Sousa, head of the Laboratório de Sistemas e Tecnologia Subaquática robotic lab at the University of Porto, and all the people working there for letting me participate, and field test my work, in their yearly Rapid Environment Picture exercise in 2013 and 2015.

I would also like to thank all of my colleagues at the Department of Engineer-

ing Cybernetics, and also a thanks to the people that participated in our cageball sessions. They have sometimes been an outlet for frustration, but most of all just fun!

Second to last I would like to thank my friends and family for their support. Finally, I would like to give a special thanks to my parents for their unconditional support and for always believing in me.

Frederik Stendahl Leira
Trondheim, November 2016

Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

AHRS	Attitude Heading Reference System
AIS	Automatic Identification System
AUV	Autonomous Underwater Vehicle
BLOS	Beyond Line of Sight
CAMshift	Continuously Adaptive Meanshift
CO	Coordinate Origin
DUNE	DUNE Uniform Navigation Environment
ECEF	Earth-Centered Earth-Fixed
EKF	Extended Kalman Filter
ENU	East-North-Up
EO	Electro Optical
ESC	Electrical Speed Controller
GNN	Global Nearest Neighbor
GPS	Global Positioning System
GUI	Graphical User Interface
HIL	Hardware-in-the-Loop
IMC	Inter-Module Communication
LIDAR	Light Detection and Ranging
LiPo	Lithium Polymer

LSTS	Laboratório de Sistemas e Tecnologia Subaquática
LWIR	Long-Wave Infrared Camera
LWIR	Long-wavelength Infrared
MILP	Mixed Integer Linear Programming
MPC	Model Predictive Control
MV	Machine Vision
NE	North-East
NED	North-East-Down
NLP	Nonlinear Programming
NN	Nearest Neighbor
POMDP	Partially Observable Markov Decision Processes
PTU	Pan-Tilt-Unit
PWM	Pulse-Width Modulation
QP	Quadratic Programming
RADAR	Radio Detection and Ranging
RANSAC	Random Sample Consensus
RHIB	Rigid-Hulled Inflatable Boat
RTSP	Real-Time Streaming Protocol
SAR	Synthetic Aperture Radar
SBC	Single Board Computer
TLD	Tracking-Learning-Detection
UAV	Unmanned Aerial Vehicle
WGS84	World Geodetic System 84

Symbols and notation

a	The semi-major Earth axis [m]

$a(h)$	Square meters per pixel of an image captured at altitude h [m^2/pixel]
α	The gimbal's tilt angle [rad]
$\boldsymbol{A}$	A camera's intrinsic parameter matrix
$\hat{\boldsymbol{A}}$	An estimate of a camera's intrinsic parameter matrix
b	The semi-minor Earth axis [m]
β	The gimbal's pan angle [rad]
$\mathbf{B}$	A binary image
$\mathbf{B}_{filled}$	Binary image with filled contours
χ	The UAV's course angle relative in frame {e} [rad]
$\mathbf{C}$	Observation model for a linear motion model
$\mathbf{C}_{cam,ned}$	The position of the camera given in the NED frame [m]
$\mathbf{C}_{ned,m}$	Position of the origin of the NED frame given in map frame coordinates [m]
δ	The desired UAV distance to the object(s) being tracked [m]
Δt	The time passed from time step k to time step $k+1$ [s]
$D_{i,j}$	Distance metric for distance between measurement i and tracking gait j
D_{obj}	The Down position of an object in the UAV's local NED frame [m]
D_{uav}	The Down position of the UAV in the UAV's local NED frame [m]
D_x, D_y	Metric scaling factors describing the size of a grid cell in the occupancy grid map [m/grid cell]
$\mathbf{D}$	Distance matrix between all n measurements to all m tracking gaits
$\epsilon(\lambda)$	The spectral emissivity at wavelength λ of a thermal source
$\boldsymbol{\eta}$	The UAV's position and attitude vector in frame {e} [m, rad]

$\tilde{\boldsymbol{\eta}}$	The UAV's position and attitude vector relative frame {e} using the course angle χ [m, rad]
η_{pq}	The $(p+q)$th normalized central moments of an image region
$\mathbf{E}$	Noise model for a linear motion model
E_{obj}	The East position of an object in the UAV's local NED frame [m]
E_{uav}	The East position of the UAV in the UAV's local NED frame [m]
f	Focal length of a camera [mm]
$\mathbf{F}$	The state-transition matrix for a linear motion model
$\mathbf{g}$	An $n \times n$ Gaussian kernel
$\mathbf{G}$	Extrinsic orientation parameters of a camera
$\mathbf{G_I}$	Gradient image of a smoothed thermal image
$\boldsymbol{G}_{NE}$	Partial extrinsic parameter matrix for the camera
$\boldsymbol{\Gamma}$	Tunable weighting matrix for the distance between two feature vectors
γ	Tunable weighing factor for the feature vector in the distance metric
$\mathbf{h}(\mathbf{z}_k)$	Distance between the UAV and objects of interest, as well as gimbal attitude [m, rad]
$\mathbf{h}_d$	The desired distance between the UAV and objects of interest, as well as the desired gimbal attitude [m, rad]
h_{ref}	Reference height for a local NED frame [m]
$\mathbf{I}$	A thermal image
$\mathbf{I_s}$	A smoothed thermal image
J_{t+T}	MPC Cost function for the predict and control horizon $[t, t+T]$
k	Time step k
k_1, k_2, k_5	Radial distortion coefficients

k_3, k_4	Tangential distortion coefficients
k_u, k_v	Pixel sizes along a camera's image axes [m/pixel]
$l_{i,t}$	The logarithm of the probability of grid cell i being occupied at time step t
$l_{i,t-1}$	The logarithm of the probability of grid cell i being occupied at time step $t-1$
λ_{ref}	Reference longitude for a local NED frame [rad]
λ_1	Lower wavelength limit of radiation that a thermal camera is sensitive to [μm]
λ_2	Upper wavelength limit of radiation that a thermal camera is sensitive to [μm]
$M(\lambda, T_s)$	Spectral blackbody radiation of wavelength λ emitted by thermal source at temperature T_s
$\mathbf{m}_{i,t}$	Element i in the occupancy grid map at time step t
m_t	An occupancy grid map at time step t
m_{pq}	The $(p+q)$th order moment of an image region
$\boldsymbol{\nu}$	The UAV's velocity and angular rate vector in frame {u} relative frame {e} [m/s, rad/s]
$\boldsymbol{\nu}_j^{obj}$	Object j's velocity vector in frame {e} [m/s]
$\tilde{\boldsymbol{\nu}}$	The UAV's velocity and angular rate vector in frame {u} relative {e} based on the course angle χ [m/s, rad/s]
$n_{k,t}$	A partial measurement model for pixel k at time step t for the update of the occupancy grid map
N_{obj}	The North position of an object in the UAV's local NED frame [m]
N_{uav}	The North position of the UAV in the UAV's local NED frame [m]
$\mathbf{O}$	A binary image region
$\mathbf{O_p}$	Set of pixels in the detected object blob in the binary image

P_1	Tuning parameter for the partial measurement model $n_{k,t}$
P_2	Tuning parameter for the partial measurement model $n_{k,t}$
p_u, p_v	An object's location in an image [pixels]
p'_u, p'_v	Scaled object location in the image frame [pixels]
ϕ	The UAV's pitch angle [rad]
ϕ_1	The first invariant Hu moment
ϕ_{ref}	Reference latitude for a local NED frame [rad]
ψ	The UAV's yaw angle [rad]
ψ_{gb}, ϕ_{gb}	The gimbal's pan and tilt angles [rad]
$\mathbf{p}^{img}$	Location of a grid cell's center in the image frame [pixels]
$\tilde{\mathbf{p}}^{img}$	Segmented pixel in the image frame [pixels]
$\mathbf{p}^{m}$	Homogeneous coordinates of a grid cell in the occupancy grid map given in the map frame
$\tilde{\mathbf{p}}^{m}$	Position of segmented pixel $\tilde{\mathbf{p}}^{img}$ in the mapping frame
$\mathbf{P}$	The Prewitt operator matrix
q_k, r_k	Gaussian white noise terms representing noise in the measurement of an object's position
$\mathbf{q}^{m}$	Coordinates of the center of grid cell $\mathbf{p}^{m}$ given in the map frame
$\mathbf{q}^{ned}$	Coordinates of the center of grid cell $\mathbf{p}^{m}$ given in the NED frame [m]
$\mathbf{Q}$	Diagonal weight matrix for the dynamic state vector in the MPC cost function
$\mathbf{r}$	The UAV's position vector in frame {e} [m]
$\mathbf{r}_j^{obj}$	Object j's position vector in frame {e} [m]
r_χ	The UAV's course rate in frame {e} [rad/s]
$\mathbf{R}$	Diagonal weight matrix for the controlled variables in the MPC cost function

$\boldsymbol{R}_{ned}^{camera}$	The rotation matrix from the NED frame to the camera frame
$\boldsymbol{R}_{body}^{ned}$	The rotation matrix from the UAV's body frame to the NED frame
$\boldsymbol{R}_{body}^{mount}$	The rotation matrix from the UAV's body frame to the gimbal's mount frame
$\boldsymbol{R}_{camera}^{body}$	The rotation matrix from the camera frame to the UAV's body frame
$\mathbf{R}_{m}^{ned}$	The rotation matrix from a map frame to a local NED frame
$\boldsymbol{R}_{ecef}^{ned}$	The rotation matrix from ECEF frame to the UAV's local NED frame
σ	The UAV's crab angle [rad]
σ_g	The standard deviation of Gaussian kernel $\mathbf{g}$
$s_{k,t}$	A binary value indicating if pixel k is segmented into foreground or background at time step t
$\mathbf{S}_j$	Covariance matrix for prediction of object j's position
$\boldsymbol{\Theta}$	The UAV's roll, pitch and yaw angles relative frame {e} [rad]
$\dot{\boldsymbol{\Theta}}$	The UAV's roll, pitch and yaw rates relative frame {e} [rad/s]
θ	The UAV's roll angle [rad]
T	The MPC predict and control horizon [s]
T_g	A threshold to segment a gradient image
$\hat{T}$	Perceived temperature of area covered by pixel x, y in a thermal image [C°]
T_i	Threshold value to segment image into ice and non-ice regions
T_s	Temperature of a thermal source [C°]
T_{avg}	The average perceived temperature across an entire object [C°]
T_{min}, T_{max}	The minimum and maximum temperature observed with the thermal camera [C°]
$\mathbf{u}$	The controlled variables in the MPC formulation

$u \times v$	Width and height of an image [pixels]
u_0, v_0	Principal point of a camera [pixels]
u_d, v_d	A distorted image point [pixels]
u_u, v_u	An undistorted image point [pixels]
V	The UAV's airspeed [m/s]
$V_{x,k}^{obj}, V_{y,k}^{obj}$	An object's velocity in NE coordinates [m/s]
$w \times h$	Width and height of a camera's image sensor [mm]
w_k, z_k	Gaussian white noise representing change in velocity of an object
w_s	Scaling factor for homogeneous pixel coordinates
$\bar{x}, \bar{y}$	Centroid of binary image region [pixels]
x_k^{obj}, y_k^{obj}	An object's position in NE coordinates [m]
$\mathbf{X}$	A detected object's feature vector
$\bar{\mathbf{X}}$	The average of an object's first 5 calculated feature vectors
$\mathbf{X}_m^{img}$	Perspective transformation between the map frame and the image frame
X^e, Y^e, Z^e	A position in the ECEF frame [m]
x_t	The UAV's position and attitude at time step t [m, rad]
$x_{1:t}$	The UAV's position and attitude for all time steps from 1 to t [m, rad]
X_{GB}	The gimbal's position in the UAV's body frame's x-axis [m]
$\boldsymbol{y^{obj}}_i$	Object position measurement i [m]
$\hat{\boldsymbol{y}}^{\boldsymbol{obj}}{}_j$	A priori predicted position of object j [m]
$y_{x,k}^{obj}, y_{y,k}^{obj}$	Detected object's position at time step k in NE coordinates [m]
Y_{GB}	The gimbal's position in the UAV's body frame's y-axis [m]
$\mathbf{z}_k$	Dynamic state vector in MPC cost function

Z_{GB}	The gimbal's position in the UAV's body frame's z-axis [m]
$z_{1:t}$	All measurements of the occupancy grid map's state for all time steps from 1 to t
z_t	A measurement of the state of the occupancy grid map at time step t

CHAPTER 1

Introduction

This monograph represents the work conducted during the development of an object detection and tracking framework for small unmanned aerial vehicles (UAVs). It includes a full description of the developed framework and its implemented modules in its current form, in addition to experimental results, discussions, and conclusions thereof.

This chapter will first give background information that will serve as a base for the work presented throughout the thesis. A brief overview of the objectives for the conducted work will then be given, before a list of relevant publications that the work presented in this thesis is based on is stated. Finally, the chapter concludes by describing the structure and contributions of the remaining monograph.

1.1 Background

Over the last decade UAVs have played an increasingly prominent role in military applications all around the world. Surveillance and reconnaissance missions and battle damage assessment are only two examples of a wide range of applications where increasingly capable UAVs are useful [21]. In recent years, there has also been a drastic increase in the commercial availability of small UAVs for civil and commercial applications. These applications are as of yet not as well developed as the military applications, but the field of UAVs for civil applications has gained an increasing research interest. This, in addition to technological advances, has expanded the capabilites of UAVs and has lead to the use of UAVs in applications such as inspections of structures, search and rescue, monitoring and surveillance of oil spills and autonomous border control [74]. In this regard, UAV related research often emphasize the UAV's capabilities to increase safety and efficiency of a task or operation. Moreover, the UAV's capabilities as a remote sensing plat-

form is often vital to many of its applications. The remote sensing capabilities of UAVs are enabled by having the UAV carry remote sensing sensors such as a radio detection and ranging (RADAR), light detection and ranging (LIDAR), automatic identification system (AIS) receiver and (thermal) cameras [33]. An example of a search and rescue application where a UAV has been equipped with a thermal camera in order to locate humans floating in the water is illustrated in Figure 1.1.

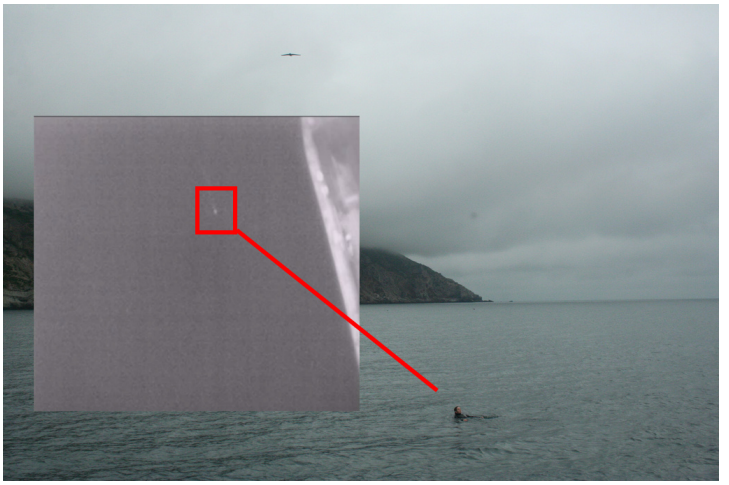

Figure 1.1: A thermal camera image taken from a UAV of a human floating in sea water and the corresponding view from the ground. In the top of the picture the UAV can be seen flying.

Using UAVs as remote sensing platforms will in many applications generate an overwhelming amount of raw sensor data. However, a UAV data analyst could be interested in only a small subset of the gathered data. Using a search and rescue operation as an example, the operator is likely most interested in knowing possible locations for the person(s) that are searched for. Hence, equipping UAVs with a processing unit capable of analyzing the sensor data for the data which is useful for the data analyst could save the data analyst valuable time. Furthermore, by placing a processing unit on-board the UAV, the autonomy of the UAV could be increased by having the processing unit make the UAV sense and react to the gathered sensor data. Increasing the autonomy of UAVs is a crucial part of extending its areas of application, as more autonomy will allow for more extensive operations such as operating beyond line of sight (BLOS). In such operations the UAV might be required to be fully autonomous for extended periods of time [53].

In recent years, computer hardware and embedded systems have become smaller, lighter, more power efficient and more powerful. This has lead to the possibility of implementing sophisticated data processing algorithms on-board small light weight UAVs [40, 17], as well as the need to develop new processing algorithms suitable for real-time use during a UAV operation. The scope of this monograph is focused on using fixed-wing UAVs equipped with a processing unit and a thermal camera as a remote sensor in order to increase the UAVs usefulness and autonomy in various applications such as search and rescue, surveillance and reconnaissance and sea ice management. More specifically its focus is on autonomous object detection and tracking and the applications where this is considered useful.

Using a UAV equipped with a thermal camera for object detection and tracking can generally be divided into two subproblems, where the first problem is that of controlling the path and orientation of the UAV in such a way that the UAV is able to capture useful images with the on-board thermal camera. This should ideally be done in such a way that the gathered thermal images are of use to the on-board processing unit and the UAV data analyst, and maximizes the gained information about the environment. The second problem is that of developing algorithms that enables the on-board processing unit to interpret the gathered thermal images, and based on this interpretation be able to automatically detect, and track, objects of interest found in these images. The first subproblem is considered a problem of path planning, while the second subproblem is considered a problem of machine vision (MV). These two subproblems are described in further detail in Section 1.1.1 and Section 1.1.2.

1.1.1 Path Planning and Object Tracking Using UAVs

While operating a low-cost and light-weight UAV has become easier in the recent years, the endurance of the vehicle is often still very limited. This makes UAV flight time a scarce resource, which should be carefully spent. In the means of using a thermal camera as a remote sensing instrument on-board a UAV, this amounts to maximizing the information gained from the gathered thermal images.

When UAVs are used in applications such as border patrol as well as search and rescue, there is often a number of (moving) objects of interest in the environment that it is vital that their location and velocity is known. Assuming that the UAV is supplied with information about the position and velocity of these objects of interest, the task of the UAV path planner is to control the UAV path in such a way that the UAV can use its remote sensing capabilities in order to keep monitoring the whereabouts of said objects. In order to do this, the objects of interest will have to

be revisited from time to time in order to verify if they have moved away from their last known positions and to update their velocity estimates, while at the same time giving all of the tracked objects enough camera time to have a sufficiently good estimate of the movement of all of the objects. One objective in such a scenario could also be to observe the objects to perform object classification, recognition and other vision-based analysis, which in is something the path planner should take into account.

The process of monitoring multiple moving objects is often referred to as multi-object tracking, and many different approaches to the problem is found in the literature. This problem is not restricted to UAVs, as there exists many different remote sensing platforms. However, finding or adapting solutions that can be applied for cameras in UAVs is still useful and necessary since e.g methods for radar-based multi-object tracking are not directly applicable. Approaches found to the problem of path planning and multi-(object) tracking in UAVs are covered in Chapter 3. In short, the different approaches to the problem of multi-object tracking can be divided into two categories; resource allocation object tracking, and information driven object tracking [45]. In resource allocation tracking the tracking problem is formulated as a visitation problem, i.e finding the path that has the UAV visit all tracked objects in the shortest time frame possible. For the information driven approach, the path is often planned dynamically based on the information reward of visiting a tracked object. I.e, this could be based either on the time duration since the last visit, the uncertainty of a tracked object's position and velocity estimate, or the information gained from the analysis of other observations of the tracked object.

Note that when the UAV is equipped with a gimbal (a pointing device covered in Section 1.1.4), the gimbal control could also be a part of the path planning modules. In these scenarios the path planner has to generate a physical path for the UAV, as well as control set points for the gimbal's rotation angles. Furthermore, although not within the scope of this thesis, the path planner could also be developed to incorporate a search component in the generated UAV path. That is, the path planner can be responsible not only for making the UAV track already detected and known objects, but also for making the UAV gather thermal images from locations which are likely to contain unknown objects of interest.

1.1.2 Object Detection and Tracking Using Machine Vision

In the context of object detection and tracking in machine vision, objects are characterized by their salient features, such as color, shape, texture, size and other

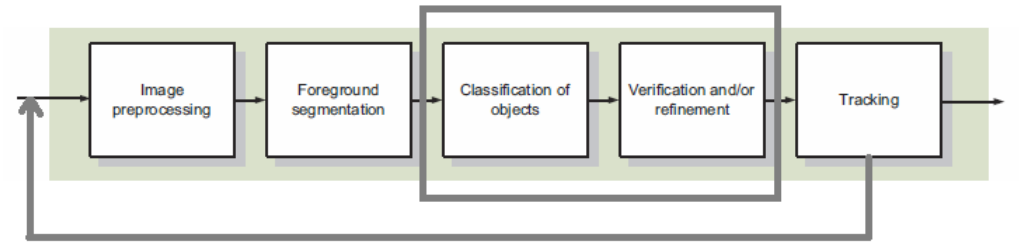

Figure 1.2: Typical processing sequence in a detection and tracking system. The two modules inside the grey box and the feed back loop might or might not be included in the overall system, depending on the purpose of the object detection and tracking system.

similar traits [28]. The problem of object detection is then telling whether an image contains a defined object, and if so, finding its position in the image. This is referred to as a 2-class classification problem [28], as one can think of it as segmenting an image into foreground (object) and background (non-object) pixels.

Object detection can be a challenging task due to the fact that the appearance of an object depends on many different factors. Just to mention a few aspects; viewing angle, distance, illumination, camera system, occlusion, deformation and background clutter are all characteristics that play an important part in relation to how an object is projected onto the lens of a camera. For an object detection algorithm there are mainly two things that can go wrong [28]. The first is that it produces a false positive. That is, the object detection algorithm is reporting finding a searched for object in an image where in fact no such object is present. The second fault is a false negative. This is when the object detection module fails to report finding an object when in fact a searched for object is located in the image. Hence, using methods that are robust against such errors is crucial, although it is often found that reducing the fault rate of one of these problems increases the occurrence of the other. That is, decreasing the number of false positives an object detection algorithm produces will often come at the cost of increasing the number of false negatives and vice versa.

For the object tracking part, the problem becomes that of following (tracking) a specific object over the duration of a video sequence [28]. Since the object detection module will not be perfect, it is important that the object tracking algorithm is robust against missed detections (false negatives) and can avoid tracking "objects"

originating from false positives. Furthermore, when an object leaves and then re-enters the image frame, it is favourable if the machine vision algorithm is able to perform recognition. I.e, ideally it should be able to decide if an object entering the image frame has been detected in the image frame at an earlier point in time, or if it is a newly discovered object. This is connected to the problem of data association, which arises when tracking multiple objects simultaneously. Data association in this context is the problem of associating object detections either with objects already being tracked, or as a novel object entering the image frame if the former is not possible or likely to be true.

When discussing the topic of object tracking, the concept of ground truth also has to be introduced [28]. The ground truth is the directly measured position of the searched for objects in an image. The performance of tracking algorithms are often evaluated on how far from the ground truth the tracked position of the object is, and is a metric often used in MV object detection and tracking systems [28].

Figure 1.2 illustrates the different processing steps in a typical object detection and tracking system that utilize machine vision. Furthermore, an overview of different existing object detection and tracking methods that is suitable for use in on-board processing of images captured with a UAV is given in Chapter 4.

1.1.3 Thermal Camera

Similarly to how UAVs have become increasingly more available for civil and commercial use, the same is the case for thermal cameras [104]. Furthermore, technological advances has made thermal cameras not only cheaper, but also smaller, lighter and equipped with better image resolution. Hence, the use of thermal cameras in small UAVs have now become more viable, and is in many applications a useful remote sensor on such platforms. Since thermal radiation is outside of the visible spectrum (thermal radiation has longer wavelengths), a visible light source (e.g daylight) is not necessary in order to acquire thermal images that can contain important information about the UAVs environment. This makes thermal cameras ideal for night time operations with UAVs, as well as being a complementary sensor to for instance regular cameras sensitive to the visible spectrum.

A thermal camera measures the thermal emission from the environment and objects contained within. The measured thermal emission from an object depends on the object's temperature and emissivity [104]. More specifically, for a given thermal source, the total gray level output of a thermal camera at the pixel location

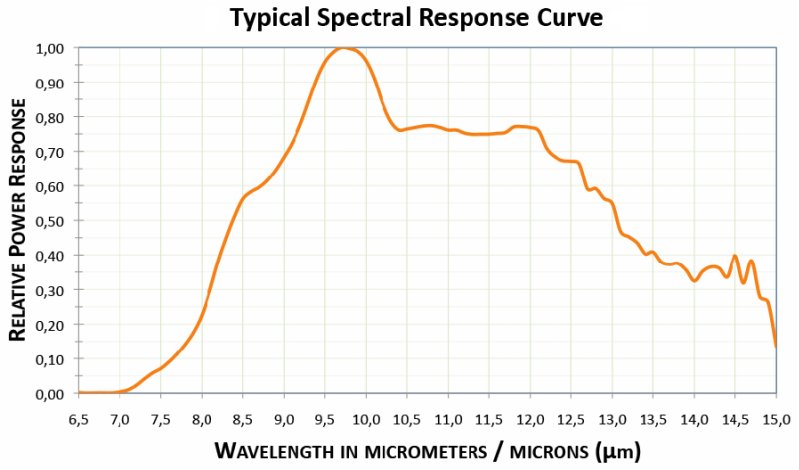

Figure 1.3: An example of the response curve of a thermal camera.

of said thermal source will be given by [94]

$$g(T_s) = \int_{\lambda_1}^{\lambda_2} M(\lambda, T_s) S(\lambda) \epsilon(\lambda) d\lambda \tag{1.1}$$

where λ_1 and λ_2 are the lower and upper wavelengths for the radiation which the camera is sensitive to and $M(\lambda, T_s)$ is the spectral blackbody radiation emitted by a thermal source at temperature T_s. $S(\lambda)$ is the sensitivity and response characteristics of the thermal camera used, and $\epsilon(\lambda)$ is the spectral emissivity at wavelength λ of the thermal source. Figure 1.3 illustrates an example sensitivity and response characteristic for a thermal camera.

The thermal camera measures $g(T_s)$ for every pixel location on its image sensor and creates a $w \times h$ greyscale image by normalizing these values for the entire image into for instance an 8 bit resolution where a pixel value of 0 will be equal to the lowest measured gray level output and $2^8 - 1 = 255$ will be equal to the highest measured grey level output. $w \times h$ is the number of pixels that the thermal image consists of. An example greyscale image captured with a thermal camera located on-board a UAV is illustrated in Figure 1.4. Once the on-board processing unit receives the captured thermal image from the thermal camera, the processing unit can treat the image as any other greyscale image captured for instance by a camera sensitive to the visible spectrum. The benefits of using a thermal camera when observing objects at the sea surface is readily seen to be that the water surface

appears as a uniform greyscale value in the thermal image, and that objects such as boats clearly stand out from the water surface.

1.1.4 Gimbal

A gimbal is a pivoted support that allows the rotation of an object about a single axis. In UAVs a set of typically two or three gimbals, one mounted on the other with orthogonal pivot axes such that each gimbal offers one degree of freedom, can be used in order to rotate an on-board sensor to point at a specific location in the environment. An example of a three degrees of freedom gimbal device made for use in UAVs is illustrated in the two lower images in Figure 1.4.

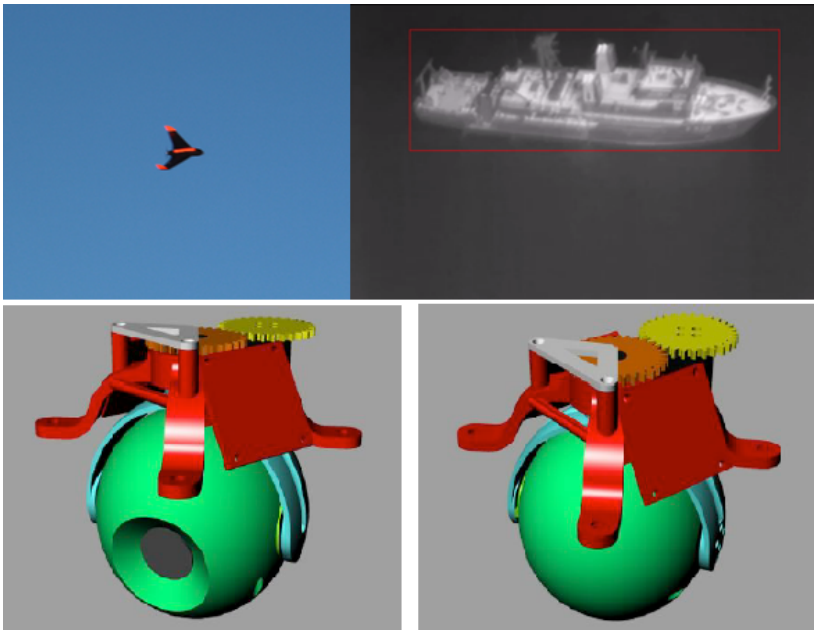

Figure 1.4: A gimbal device and its use in fixed-wing UAVs. The gimbal device shown here was modelled and developed in [95].

When dealing with the problem of object detection and tracking with UAVs using a thermal camera, the purpose of using such a gimbal device is two-fold. First, by using a gimbal in an object detection and tracking system with a thermal camera the gimbal can be enabled to stabilize the camera, which will result in sharper thermal images. This is not only convenient for the UAV data analyst, but many machine vision algorithms can break down when blurry images are supplied to

the machine vision system. Second, using a gimbal device will enable the UAV path planner to utilize completely different and more convenient tracking patterns than a system that does not utilize a gimbal. This is because the alternative is to have the thermal camera located at a fixed position in the UAV payload, effectively causing a direct coupling of the UAV path and attitude with the images captured. Hence, in order to capture images of specific locations in the environment, the UAV attitude as well as position would have to be controlled accordingly. Using a gimbal the path planner has more degrees of freedom, and the gimbal can be used to counteract the effects of changes in the UAV's attitude when wanting to image a specific location. Another advantage of using a gimbal is a combination of the two benefits previously stated, where the gimbal can be used to stabilize the camera in such a way that the points of interest in the UAV's environment are centred in the gathered thermal images. Although the gimbal controller has to be either incorporated in the path planner or made as a stand-alone controller, the effort of doing so is undoubtedly beneficial to the application of object detection and tracking in UAVs.

1.2 Objectives

The literature is rich on approaches to the object detection and tracking problem using UAVs, as can be seen in Chapter 3, 4 and 5, and the references therein. However, all of the approaches are ultimately UAV platform specific and in many cases implemented ad-hoc. Furthermore, most of the approaches are application specific, which combined with the platform specific and ad-hoc implementation constitutes a rigid system which is hard to adapt and configure for use in other scenarios and applications than what the system was intended for.

Therefore, a large effort was put into the development and implementation of an object detection and tracking framework for UAVs, which is more agile and adaptable than the approaches found in the literature. More specifically, the developed object detection and tracking framework and system should be agile and adaptable in a way to ensure that it has the following benefits:

Configurable: The object detection and tracking framework should be easily configurable to facilitate the specific object detection and tracking requirements for a given application. For instance, automated border patrol is concerned with locating e.g illegal trespassing of a border. In this scenario the path planner should be able to guide the UAV along the border while the machine vision module detects objects of interest (humans on foot, cars etc.) and reports its findings to the UAV data analyst. This detection and tracking behaviour is very different from the ideal

behaviour for a UAV used to keep track of and follow a boat passing through an ice infested region. In this scenario, the path planner should be able to keep the UAV in the vicinity of the boat while at the same time using machine vision in order to assess if the boat is moving in a direction which pose an immediate threat.

Modularized: The framework should be highly modularized in the sense that if for instance the implemented path planner solution is changed, the new path planner solution could be easily made compatible with the remainder of the system. This indicates the need to be very specific in which type of information and in what format each module should communicate with the other modules in the framework. Furthermore, the requirement for the framework to be highly modularized is also arguably necessary in order to have a highly configurable system. This is because two different object detection and tracking applications might for instance require two completely different machine vision modules in order to facilitate the requirements of the specific object detection and tracking tasks of the application.

Hardware Independent: The framework should include an abstraction layer for the hardware located on-board the UAV. Since advances in the field of UAV technology and hardware is happening at such a high rate, new UAV platforms and autopilots are constantly available. In addition, the processing unit placed on-board the UAV payload is quickly outdated and should be replaced by a new, more power efficient and more powerful processing unit at regular intervals. Hence, the object detection and tracking framework should be developed in such a way that it provides an abstraction layer between the framework and the hardware components. By doing this, changes in hardware will only affect the module responsible for interacting with the hardware component that is changed.

The objective of this thesis is investigating methods to increase the agility and adaptability of object detection and tracking with UAVs in all parts of such a system. This includes the development of an object detection and tracking framework and payload software which is as independent as possible of the UAV platform and hardware (Chapter 2), as well as the development of an adaptable and configurable path planner (Chapter 3) and machine vision (Chapter 4 and 5) modules.

1.3 Publications

The following publications describe the main results of this thesis:

- [62] F. S. Leira, T. A. Johansen, and T. I. Fossen. Automatic detection, classification and tracking of objects in the ocean surface from UAVs using a

thermal camera. In *Proceedings of the IEEE Aerospace Conference*, pages 1–10, 2015

- [64] F. S. Leira, K. Trnka, T. I. Fossen, and T. A. Johansen. A ligth-weight thermal camera payload with georeferencing capabilities for small fixed-wing UAVs. In *Proceedings of the IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 485–494, 2015
- [96] E. Skjong, S. A. Nundal, F. S. Leira, and T. A. Johansen. Autonomous search and tracking of objects using model predictive control of unmanned aerial vehicle and gimbal: Hardware-in-the-loop simulation of payload and avionics. In *Proceedings of the IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 904–913, 2015
- [63] F. S. Leira, E. Skjong, S. A. Nundal, T. A. Johansen, and T. I. Fossen. Autonomous unmanned aerial vehicle object tracking framework using model predictive control and camera gimbal. *AIAA Journal of Guidance, Control, and Dynamics*, 2017 (*Submitted*)
- [60] F. S. Leira, T. I. Fossen, and T. A. Johansen. Object detection, recognition and tracking from UAVs using a thermal camera. *Journal of Field Robotics*, 2017 (*Submitted*)
- [61] F. S. Leira, T. I. Fossen, and T. A. Johansen. A UAV ice tracking framework for sea ice management. *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017 (*Submitted*)

Other published papers not part of this thesis

- [34] M. Faria, J. Pinto, F. Py, J. Fortuna, H. Dias, R. Martins, F. S. Leira, T. A. Johansen, J. Sousa, and K. Rajan. Coordinating UAVs and AUVs for oceanographic field experiments: Challenges and lessons learned. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6606–6611, 2014
- [49] H. H. Helgesen, F. S. Leira, T. A. Johansen, and T. I. Fossen. Tracking of marine surface objects from unmanned aerial vehicles with a pan/tilt unit using a thermal camera and optical flow. In *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 107–117, 2016

1.4 Structure of the Thesis and Main Contributions

The thesis is a collection of papers, which makes most of the chapters self contained. However, Chapter 2 is an exception, and contains the design and imple-

mentation of the system and payloads, which was used to conduct all flight experiments described throughout the thesis.

Chapter 2: This chapter describes the overall architecture for the object detection and tracking system developed throughout this thesis. This includes all aspects of the system, from the description of how the sub-modules of the system interact with each other, to the design and implementation of the UAV payloads utilized for flight experiments where different parts of the system have been tested. The main contribution of this chapter is the design and implementation details of a UAV payload based around a thermal camera and a gimbal, which is suitable for use in multiple object detection and tracking scenarios.

Chapter 3: (Based on [63]) In this chapter, a model predictive control (MPC) path and gimbal controller suitable for use with the overall object detection and tracking framework developed in Chapter 2 is described. The controller can be implemented on any UAV platform, with the only requirement being that the UAV use an autopilot that is able to compensate for wind disturbances and allows for external path commands. The MPC is configurable in-flight, and is designed to track either a single object or a group of objects in the proximity of each other, keeping a configurable distance to the object(s) so that they can be observed by appropriate gimbal control. The MPC is tested in three different scenarios using two different UAV platforms. In this chapter, all field tests are conducted using virtual object positions and velocities. All of the conducted field tests resulted in successful object tracking by the UAV. The main contribution in this chapter is the verification of using the described MPC in the overall object detection and tracking framework by demonstrating its object tracking capabilities in field tests.

Chapter 4: (Based on [60]) This chapter describes a multiple object detection, recognition and tracking module suitable for use with the overall object detection and tracking framework developed in Chapter 2. The module can be implemented on any UAV platform, with the main requirement being that the UAV has a suitable on-board computational unit and a camera. Using machine vision to automatically detect objects in the camera's image stream combined with the UAV's navigation data, the on-board computer is able to georeference each object detection in order to measure the location of the detected objects' in a local North-East coordinate frame. A tracking algorithm which uses a Kalman filter and a constant velocity motion model utilize the object's position measurements found using the object detection algorithm in order to track and estimate an object's position and velocity. Furthermore, a global nearest neighbour algorithm is applied for data association.

This is achieved using a measure of distance that is based not only on the physical distance between an object's estimated position and the measured position, but also how similar the objects appear in the camera image. The main contributions of this chapter is both the development and implementation of the entire multiple object detection, recognition and tracking module, as well as verifying the functionality of the module by conducting two flight experiments.

Chapter 5: (Based on [61]) This chapter describes how the object detection and tracking framework can be adapted to be used in an ice management scenario where the UAV should detect and track the movement of icebergs and ice floes in an Arctic environment. This is achieved by using an occupancy grid map and a locations of interest generator coupled with the total object detection and tracking framework. The main contribution of this chapter is interfacing an occupancy grid map with the remainder of the object detection and tracking framework, by generating locations of interest based on the occupancy grid map established during an initial search period. These locations of interest are then used by the MPC in order to track objects at these locations. Furthermore, the chapter verifies the use of the machine vision algorithm developed in Chapter 4 for detecting ice bergs and ice floes from thermal images, as well as verifying the developed framework in an ice management scenario by conducting two flight experiments.

Chapter 6: Concluding remarks and future perspectives are presented in the final chapter.

CHAPTER 2

UAV System and Payload

This chapter presents the UAV system and payload developed and implemented to test the object detection and tracking algorithms presented in this thesis. First an overview of the total system is given in Section 2.1, before the software and hardware developed and/or used for the UAV flight tests conducted in this work is presented. The chapter concludes with Section 2.4, which covers the actual implementation of the payload described in this chapter on two different UAV platforms.

2.1 System Overview

The overall proposed object detection and tracking system is illustrated in Figure 2.1. The path controller is an MPC, which seeks to find an optimal turning rate or waypoint for the UAV, as well as the optimal gimbal attitude given by the pan and tilt angles. The control outputs found by the MPC algorithm are given to the UAV's flight controller (autopilot), which in turn has its own controllers to control the low-level dynamics of the UAV. In this work, the UAV flight control system is assumed to be already implemented to stabilize the UAV's dynamics. The output of the MPC is optimal in the sense that it tries to optimize the path of the UAV and the gimbal attitude with respect to camera time. Camera time is here defined as the time a tracked object is in the field of view of the camera, that is, the MPC is responsible for sending the control inputs which maximize a tracked object's camera time to the UAV's autopilot.

When performing tracking of multiple objects simultaneously, the Object Handler is the module that is responsible for prioritizing which object(s) that should be tracked at any given time. The inner workings of this module may vary depending on the nature of the objects that are being tracked. Often discrete optimization

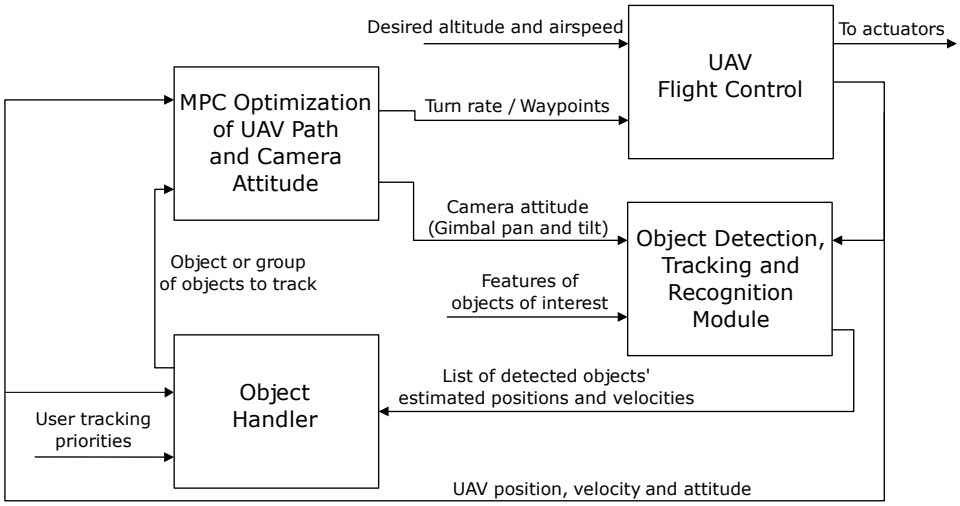

Figure 2.1: Overall object tracking system description. A path planner module (MPC) is combined with an object detection, tracking and recognition module in order to detect and track objects using a UAV platform. The UAV's flight controller is assumed already implemented in the present work.

such as graph search (Travelling Salesman Problem [18]) and Mixed Integer Linear Programming (MILP) [14] can be used as an alternative to find an optimal prioritization of the visiting sequence by some predefined criteria (e.g shortest total flight distance). However, a simple approach where the module choose to track the object closest to the current position of the UAV can also be effective in many applications. The main concept of this system is that after the MPC algorithm has given the object received by the Object Handler a set amount of camera time, the Object Handler will decide the next object (or group of objects) that the UAV should prioritize for tracking.

The UAV object detection, tracking and recognition module, also included in Figure 2.1, is a machine vision (MV) module running on-board the UAV supplying the path planner with estimates of object's position and velocity. These estimates can be found using different methods. The most autonomous method is to have an on-board computer analyse the images coming from the on-board camera, automatically detecting, recognizing and tracking objects of interest over a series of images. An example of such a system can be found in [59] and [64]. This approach is applied without any prior knowledge of the location and number of objects of interest, but the user will typically input to the MV module some

features of the objects that are of interest (e.g size, anticipated temperature and shape). Another scenario is the case where one already knows the location of the objects that one want to track, but still want to do some verification and/or surveillance of the objects. In this case the object detection, tracking and recognition module is initialized with a list of the positions and velocities of all objects, and the path planner (MPC module) will then make sure that these objects will be observed by the UAV. This approach can be combined with the previous mentioned autonomous approach, making the object detection, tracking and recognition module update an object's position and velocity when the object is in the image frame and the MV module is automatically detecting the object, effectively estimating its current position and velocity. It is also possible with semi-autonomous operations where object detection or recognition is, if needed, assisted by a UAV data analyst. This could in many cases increase the accuracy of the MV module and yield an overall better object detection and tracking performance.

2.2 Software Toolchain

This section will describe the software toolchain used in order to realize the system outlined in the section above in two different UAV platforms. The software toolchain is based on a control architecture developed at the Laboratório de Sistemas e Tecnologia Subaquática (LSTS), which is a robotic lab at the University of Porto. Their control architecture is based on their open-source Dune-IMC-Neptus software toolchain [80]. The toolchain, and how it has been customized for the object detection and tracking system developed in this thesis, will be described in this section.

2.2.1 Inter-Module Communication

Inter-Module Communication (IMC) is a communications protocol that defines a common set of control, sensor and actuator messages that is understood by all types of nodes in a network of unmanned vehicles and ground stations. This provides an abstraction layer that will enable sensor information and control input to be communicated between for instance a ground control station and a UAV platform, independent of the actual sensor package and autopilot carried by the UAV. The main concept here is that each node in the network places information it wants to communicate to the other nodes in the network inside predefined IMC messages. The protocol is described in detail in [68], and its use in the object detection and tracking system presented in this thesis is further explained in the following subsections.

2.2.2 DUNE

DUNE Uniform Navigation Environment [78] (DUNE) is a system software used on-board the UAV platforms presented in this chapter. It provides an architecture independent C++ programming environment which is based on using reactive tasks in a modular fashion. That is, a 'task' is here defined as a module within the DUNE environment that has a very specific purpose. For instance, if the UAV is equipped with an on-board camera, there would be a task within DUNE responsible for reading the video stream from said camera and conveying this video feed to other tasks or nodes in the network for further processing. Similar single-purpose tasks would be found in the areas of control, navigation and actuators. Note that similar to how each node in the network communicates using the IMC protocol, the very same protocol is used in order to convey information across different tasks within the DUNE environment. As an example, the next section will describe what tasks were established in order to implement the object detection and tracking system described in Figure 2.1, and how each of these tasks communicates with each other.

Machine Vision, MPC and Object Handler as DUNE Tasks

The implementation of the overall object detection and tracking system within the DUNE environment is illustrated in Figure 2.2. The IMC bus is organized in DUNE with a subscriber and provider methodology. Note that a task can be both a provider and a subscriber to certain IMC messages simultaneously, enabling the information flow illustrated. The provider and subscriber methodology means that each task can subscribe to specific types of IMC messages, and once these messages are provided (by other tasks) on the IMC bus, the task will be notified and the IMC message containing the wanted information will be delivered.

The MPC task runs the MPC described in Chapter 3 internally, and requires both the telemetry data it receives from the autopilot task and the object visiting sequence it receives from the Object Handler task in order to perform its calculations. These calculations will in turn result in a control input to the autopilot consisting of a waypoint and a set point for the gimbal servo angles.

The autopilot task is a module with the purpose of reading the autopilot's sensors and data such as position, velocity and attitude, and provide it to the other tasks subscribed to the telemetry IMC message via the IMC bus. Furthermore, the autopilot task is subscribed to IMC messages containing set points for waypoints for the UAV path and gimbal angles. Although not illustrated in Figure 2.2, the auto-

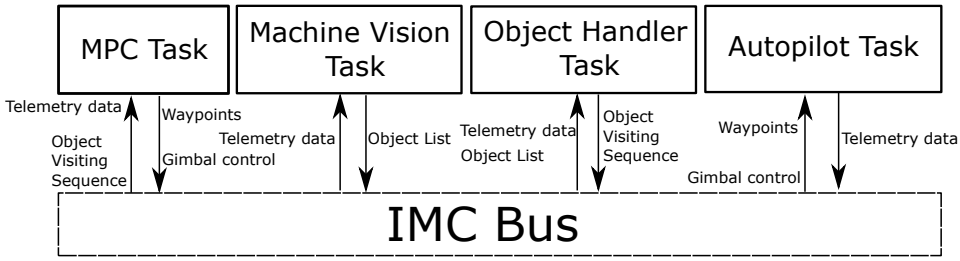

Figure 2.2: IMC bus example

pilot task also has an interface directly to the autopilot itself, enabling it to read the autopilot's sensor values and configure and send control input to the autopilot based on the IMC messages that the autopilot task receives.

The Machine Vision task runs the object detection, recognition and tracking algorithms developed in Chapter 4. These algorithms results in an 'object list', which is an IMC message with a list of all detected objects which are currently being tracked. The list is structured so that each object get a unique identifier, and the current position and velocity estimates for each object is also contained within the same IMC message. A new 'object list' IMC message is provided by the Machine Vision task every 5 seconds, given that the list has changed since the previous message was provided. Note that the Machine Vision task is also subscribing to the autopilot taks's telemetry data IMC message, and that the autopilot task only has to provide one IMC message to the IMC bus in order for the MPC task, the Object Handler task and the Machine Vision task to receive it. Note that in the same way that the autopilot task has an interface directly to the autopilot itself, the MV task also has an interface directly to the thermal camera video feed. This interface is not included in Figure 2.2.

The Object Handler task subscribes to telemetry data and object list IMC messages, enabling it to prioritize which objects to focus the camera (placed in a gimbal) on, and in what sequence. This can be done by simply focusing on the object closest to the UAV that has not been in focus already, at any given moment. A more sophisticated Object Handler is briefly discussed in Chapter 5.

Implementing the MPC, Object Handler and MV modules as separate tasks in DUNE, as well as letting DUNE communicate with the autopilot provides a hardware abstraction. This is beneficial because if, for instance, one were to change the autopilot in the UAV system, the only other change that would have to be made

to the complete object detection and tracking system is to change the way DUNE is communicating with the autopilot. This is done by specifying (or alternatively creating if it is not already available) another task in DUNE which is responsible for communication with the new autopilot. In Figure 2.2, this is the equivalent of changing the autopilot task. The other parts of the system would remain untouched.

However, implementing the object detection and tracking framework's modules in a distributed architecture introduces some challenges regarding the synchronization in the time domain of data communicated between the modules. In DUNE this is handled by including a UNIX timestamp (the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970) in each IMC message. Each DUNE task then has to read the timestamp data field of the IMC messages it receives, and based on this timestamp, synchronize the IMC message with its internal data and variables.

2.2.3 Neptus

Neptus is an open-source Command and Control Center [79], which can be used in operations with any type of vehicles, sensors and human operators. Neptus can be used for a variety of tasks, such as world representation and modelling, mission planning, simulation, execution control and supervision, data logging, and post-

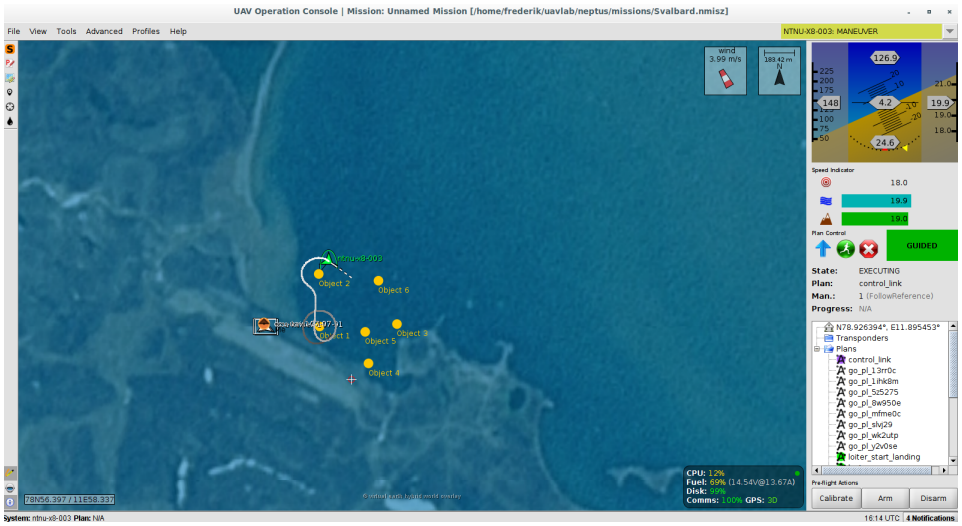

Figure 2.3: The Neptus Command and Control GUI

mission analysis. While DUNE is the on-board control software, Neptus is it's counter-part on the ground station. DUNE and Neptus communicates using the IMC protocol. An example of a Neptus console is shown in Figure 2.3.

Since Neptus is an open-source project, it is possible to incorporate customized plugins into the Command and Control Center. A plugin showing yellow circles at the locations that the on-board machine vision algorithm detects objects was implemented into Neptus, and the result can also be seen in Figure 2.3. In this scenario, the UAV (green arrow) has detected six objects (yellow circles) and has started tracking them. The yellow circles are visual representations of the items in the 'object list' IMC message provided by the on-board Machine Vision task described in the previous subsection.

2.2.4 MPC Ground Station Software

The MPC has an off-board component, which is a graphical user interface (GUI) developed in [95]. This GUI makes it easy for the operator to observe what the MPC is currently planning. Furthermore, the GUI allows the user of the system

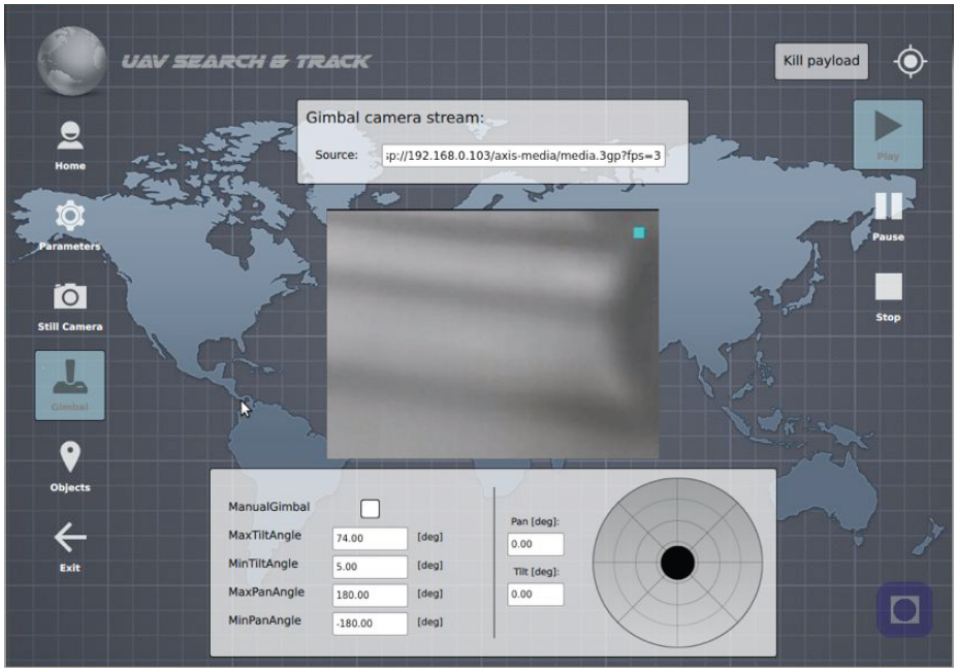

Figure 2.4: The MPC ground station software

to see what objects are being tracked, where they are estimated to be located, and also view a real-time video stream directly from the UAV. Moreover, an important functionality of the MPC GUI is that the operator can use it to change some of the MPC's configuration parameters on-line during operation. Parameters such as the optimal distance from the UAV to the tracked object, the amount of camera time per object and how smooth the gimbal movements should be can all be configured on-line during UAV operations from this GUI. Furthermore, if something goes wrong, the GUI can be used to cut the power to the payload of the UAV. This will only keep the most crucial components such as the autopilot and the UAV's servos alive.

Figure 2.4 shows how one of the screens in the MPC GUI are designed. This screen shows how the MPC GUI can be used in order to view the real-time video feed from the UAV, and also set manual set points for the on-board gimbal. Exactly how this is done is described in the next section. Note that also the MPC GUI uses the IMC protocol in order to communicate with the other nodes (UAVs and ground stations) in the network.

2.2.5 Gimbal Control

In order to make the gimbal control architecture as independent from the specific hardware used in the UAV payload as possible, a control hierarchy as shown in Figure 2.5 was implemented. Note that the Gimbal task illustrated in this figure was omitted for simplification of Figure 2.2. The control architecture is centralized around a dedicated gimbal control task implemented in DUNE, which task is to receive set points for the gimbal's pan and tilt angles. Furthermore, this task works as a state machine, where the control input given to the task is only valid if the gimbal controller task is currently in the state that the control input is originating from. The gimbal controller was implemented with three different states, 'manual', 'stabilized' and 'MPC'. When the gimbal controller is put into manual state, input from both the joystick (connected via Neptus) and manually set gimbal angles from the MPC GUI is accepted as valid control input. The 'stabilized' state is intended for use when the UAV is flying in a loiter pattern. When the gimbal controller is in this state, the task will automatically control the gimbal to point directly towards the center of the loiter pattern in a stabilized manner. That is, the controller will compensate for changes in for example the roll and pitch angles. The 'MPC' state is a state where the gimbal controller will accept control input from the MPC when the MPC is controlling the UAV. Note that the ground station part of the gimbal control architecture communicates with the on-board DUNE

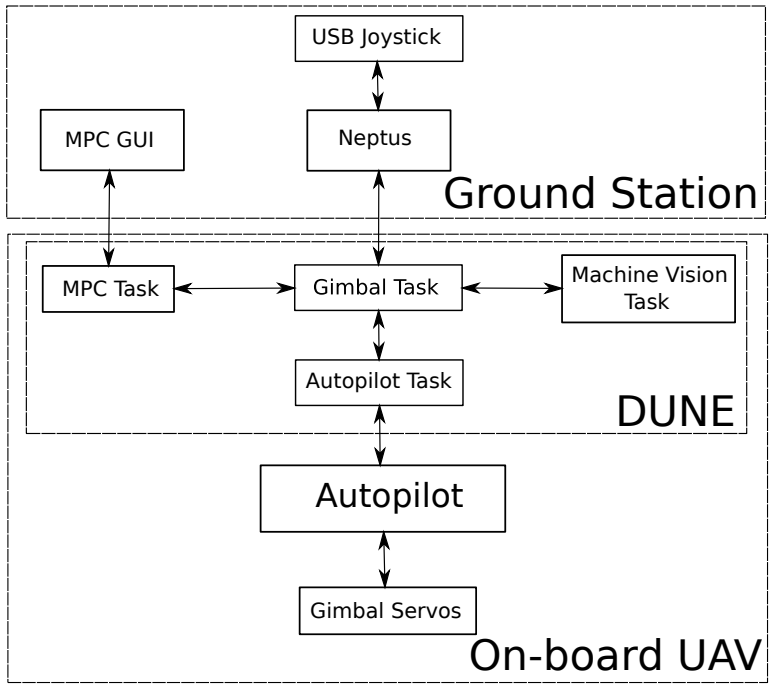

Figure 2.5: Overview of the gimbal control architecture.

tasks with IMC messages sent via the UAV payload's radio link. Also note that the different tasks implemented in the control architecture also use IMC messages to communicate between each other on-board the UAV. Although not fully implemented yet, the Machine Vision task is also allowed to take control of the gimbal pan and tilt angles, and the intention is that machine vision can be used in order to make small adjustment to the gimbal's pan and tilt angles in order to keep an object centred in the image frame.

In Figure 2.5 it is illustrated that the gimbal controller is communicating with the autopilot task within the DUNE environment. This may seem unnecessary at first, but is implemented this way in order to make the gimbal task as generic as possible. By implementing the gimbal control architecture as shown in Figure 2.5, changing the autopilot in any UAV payload becomes an isolated change, where the only requirement for the new autopilot and corresponding autopilot task is that the task can translate IMC messages coming from the gimbal task to servo set points for the gimbal servos. That is, the changes are limited to the autopilot itself and the autopilot task.

The illustrated gimbal control architecture requires that the autopilot has at least two available Pulse-Width Modulation (PWM) outputs in order to control the pan and tilt servos of the gimbal. If this is not the case, an alternative solution would be to use available PWM outputs on the on-board single board computer (SBC). By doing this, the autopilot task would be replaced by a SBC PWM controller task instead.

2.3 Hardware

In order to implement the system depicted in Figure 2.1, a suggested UAV payload hardware architecture is shown in Figure 2.6. The system consists of an Ethernet switch functioning as a communication hub between the on-board components and the ground station through the radio link. The suggested camera system includes a long-wave infrared (LWIR) camera together with a frame grabber making the camera images digitally available to the UAV system's network. There are many choices for a camera system, both in regard to the camera itself and the method used to make the camera's images available to the other UAV system components. The camera should be chosen on the basis of the task at hand, and make the (potential) autonomous object detection part as easy and robust as possible, hence the LWIR camera is just an example to illustrate a typical UAV system for object tracking. Furthermore, the UAV system has an on-board single board computer, enabling the UAV to perform real time on-board image processing and path planning.

The total architecture for the ground station is shown in Figure 2.7. This is based on the proposed ground station architecture and the UAV operations philosophy presented in [108]. The ground station will often have a dedicated ground control station for the autopilot. It is not needed for the autonomous UAV object detection and tracking operation, but can be included for other flight modes and safety reasons. In any case, the autopilot's dedicated ground control station will operate on a separate communication channel, directly connecting the ground station to the autopilot, effectively making this communication link independent on the remainder of the UAV system, even though it can (optionally) be connected to the remainder of the system through the switch. Furthermore, the proposed ground station setup has a computer running the off-board MPC and Neptus.

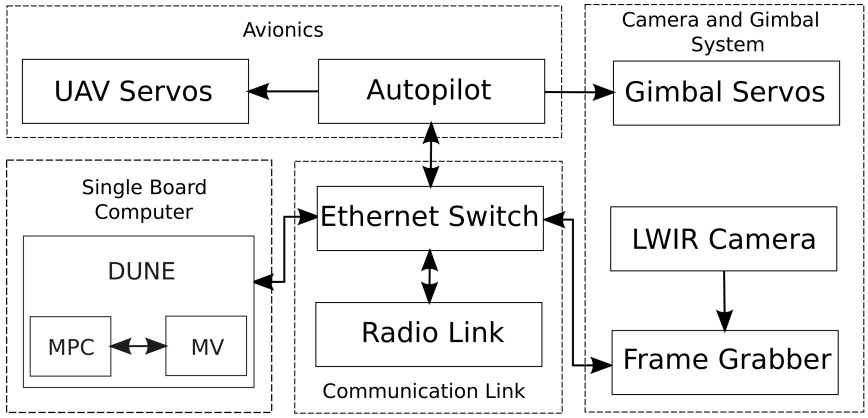

Figure 2.6: An example of a UAV payload architecture that can be used in conjunction with the proposed system for object detection and tracking. Both the MPC and MV modules are implemented as sub-modules of the on-board UAV control software DUNE

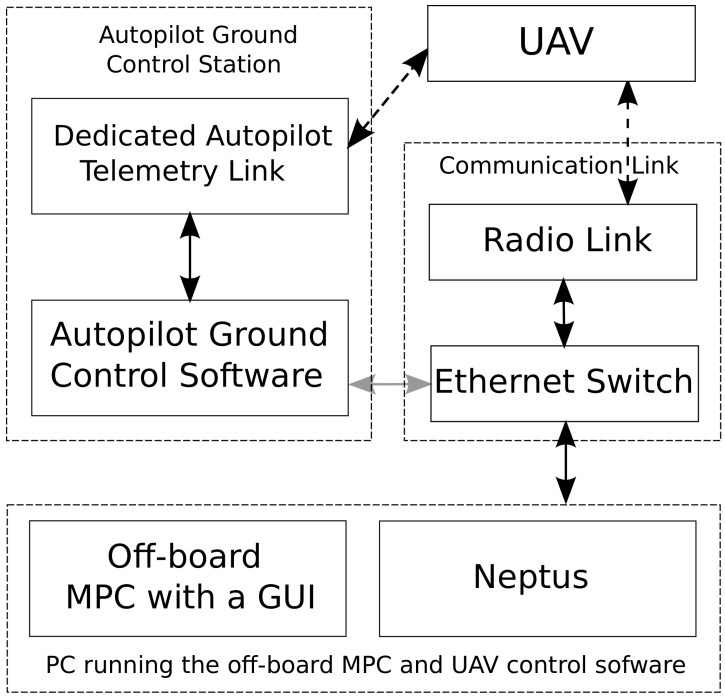

Figure 2.7: The architecture of the ground station. Neptus is the UAV Command & Control Center, while the MPC is the off-board component of the MPC algorithm.

2.3.1 Thermal Camera

A suitable thermal camera to be used as a visual sensor in the object detection and tracking system described in this chapter was chosen based on the research done in [59] and the following criteria

- Long-wavelength infrared (LWIR) sensitive ($8 - 15\mu m$)
- Low weight (maximum 200 g)
- Small in size (less than $10 \times 10 \times 10$ cm^3)
- Low-power requirement (max 5 Watt)
- Middle to high resolution (for thermal cameras this means at least 320×240 pixels)

This resulted in using the FLIR Tau2 thermal camera core [2]. It has a sampling frequency of 9 frames per second (upsampled to 30 frames per second for analog output) and is sensitive to the LWIR spectral band ($7.5-13.5\mu m$) with a sensitivity of $< 50mK$. The camera was acquired in two versions (FLIR Tau2 336 and FLIR Tau2 640) where the former has a resolution of 336×256 pixels and the latter has a resolution of 640×512 pixels. They both weigh only 70 grams. A more detailed description of their characteristics can be found in Appendix B.

Another choice that has to be made is which focal length to use. The focal length of an optical system is a measure of how strongly the system converges or diverges light, where a longer focal length will lead to a higher magnification and a narrower field of view when the camera is used on-board a UAV. Table 2.1 summarize the field of view for the FLIR Tau2 thermal camera for different focal lengths.

The optimal choice for the focal length is of course entirely dependent on the scenario that the thermal camera is to be used for. That is, the UAV's operating altitude, the image sensors resolution and the size of the objects of interest to be detected are all factors which affect the ideal choice for the camera's focal length. Further, the level of required automation for the scenario and application also affects the choice of focal length. Johnson's criteria for detection and identification [55] states that the number of object pixels (pixels occupied by an object) required for automated detection is less than that of for instance automated identification and recognition. Hence, if only detecting objects are sufficient for an application a short focal length (less magnification) can be used, compared to scenarios where object identification and recognition is required. Johnson's criteria then states that more information (more object pixels) are required for these applications, hence a

Table 2.1: A table listing focal length vs. field of view. The values were retrieved from [2].

Camera \Focal Length	**7.5mm**	**9mm**	**13mm**	**19mm**
Tau2 640	$90° \times 69°$	$69° \times 56°$	$45° \times 37°$	$32° \times 26°$
Tau2 336	$45° \times 35°$	$35° \times 27°$	$25° \times 19°$	$17° \times 13°$

longer focal length (more magnification) has to be utilized.

In the flight tests conducted throughout this thesis the thermal camera was placed inside a gimbal, effectively putting less importance on having a large field of view as the camera's field of view can simply be moved to focus on an area of interest. Hence, for testing done with the Tau2 640 model a focal length of 19 mm was used, and for the Tau2 336 a focal length of 9 mm was used. This results in a field of view of approximately $35° \times 27°$ for both of the Tau2 versions. This choice also yielded sufficiently magnification to easily automatically detect objects of interest at the ground surface, as well as keeping the field of view big enough to have several objects within the camera's image frame simultaneously.

Since the FLIR Tau2 camera uses an analog output, it was necessary to use a frame grabber in order to read the analog output from the camera and convert it into a digital format which could be read by the on-board single board computer for processing. The Axis M7001 frame grabber [1] was found to be suitable for use with the FLIR Tau2 thermal camera, and provides a digital video stream to both the on-board computer and the ground station through a real-time streaming protocol (RTSP) video stream. More details on the Axis M7001 frame grabber is listed in Appendix D.

Extension For Future Development

It should be noted that retrieving the thermal images from the RTSP stream, which is a continuous video feed made available by the Axis M7001 frame grabber, is not an ideal solution. This is because the frame grabber introduces a time delay between the time that the FLIR Tau2 captures an image and it is available on the on-board video stream. Furthermore, it has no guarantee that each image is read from the FLIR Tau2's analog output and converted to a digital format compatible with the video stream in the exact same time. This means that one image can arrive some milliseconds later than the expected $\frac{1}{9}$ second spacing (the FLIR Tau2 camera has 9 frames per second, and the next might arrive some milliseconds before. Hence, knowing exactly when an image read from the video stream is

captured is impossible. This makes it very difficult to synchronize the on-board telemetry data with the captured thermal image data, which is required in order to georeference the pixels in the images. A solution to this would be to implement the a digital frame grabber proposed in [89], which is a custom made printed circuit board with a micro-controller which is able to read the digital output of the FLIR Tau2 directly from its digital output pins. This digital signal is converted to images which the on-board single board computer can receive through a USB connection. When the digital output of the FLIR Tau2 is used instead of the analog, the time of capture for each image is also available, enabling the UAV payload to more or less perfectly synchronize the thermal image data with the on-board telemetry data. This would increase the accuracy of the georeferencing process, and in turn increase the performance of the entire object detection and tracking framework.

Camera Calibration and Distortion Model

When using a thermal camera as a visual sensor for object detection and tracking, it is important to relate the location of detected objects in the image frame to their actual location in geodetic coordinates. To do this as accurately as possible, the camera's intrinsic parameter matrix, $\boldsymbol{A}$, has to be either calculated or estimated by a process referred to as camera calibration. The intrinsic parameter matrix is a matrix used in order to transform image coordinates (given in pixels) to camera coordinates (world units). The exact calculations for this is given in Chapter 4. However, the process of finding the values for the FLIR Tau2 camera's intrinsic parameter matrix is presented in this section.

Firstly, $\boldsymbol{A}$ is completely dependent of the camera characteristics, and is defined as:

$$\boldsymbol{A} := \begin{bmatrix} k_u f & 0 & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

where (u_0, v_0) is the principal point of the camera, k_u, k_v are the pixel sizes along the two image axes and f is the focal length of the camera. This matrix can be found in two ways. The first is to use the camera specification sheet to find the focal length f of the lens, the image sensor size $(w \times h)$ of the camera and its pixel resolution $(u \times v)$. The following is then an estimate for $\boldsymbol{A}$:

$$\hat{\boldsymbol{A}} = \begin{bmatrix} \frac{u}{w} f & 0 & \frac{u}{2} \\ 0 & \frac{v}{h} f & \frac{v}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

The reason that this is only an estimate of $\boldsymbol{A}$ is that the manufacturer of the camera will not be able to make the camera exactly as specified. E.g the camera's principal point might be off by a small margin, making $(\frac{u}{2}, \frac{v}{2})$ a poor estimate for the principal point.

The camera intrinsic parameter matrix is based on an ideal pinhole camera model. In reality, this is not correct for actual cameras. This results in distortions both along the radial and tangential direction of the image frame, hence this needs to be corrected. In order to correct for distortion, we will use the following simplified model for the distortion in a camera [27]:

$$\begin{aligned} u_d &= u_u(1 + k_1 r^2 + k_2 r^4 + k_5 r^6) + 2k_3 u_u v_u + k_4(r^2 + 2u_u^2) \\ v_d &= v_u(1 + k_1 r^2 + k_2 r^4 + k_5 r^6) + k_3(r^2 + 2v_u^2) + 2k_4 u_u v_u \end{aligned} \tag{2.3}$$

where

$$r = \sqrt{u_u^2 + v_u^2} \tag{2.4}$$

(u_d, v_d) is the distorted image point as projected in the camera frame by the camera lens, while (u_u, v_u) is the undistorted image point as projected by an ideal pin-hole camera. That is, (u_d, v_d) is the position in the camera frame that we observe, while (u_u, v_u) is the (undistorted) position that we want to find. k_1, k_2, k_5 are radial distortion coefficients, while k_3 and k_4 are tangential distortion coefficients.

Now, in order to find the undistorted camera frame coordinates (u_u, v_u) of an object given its pixel location (p_u, p_v) in an image, we use the following relation

$$\begin{bmatrix} u_d \\ v_d \\ 1 \end{bmatrix} = \boldsymbol{A}^{-1} \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} \tag{2.5}$$

Having found the object's distorted position (u_d, v_d), we now solve for (u_u, v_u) in (2.3). This is non-trivial because of the nonlinearity of the problem, and the reader is referred to [48] for a more in-depth take on the solution.

In order to find a better estimate of $\boldsymbol{A}$ than (2.2), in addition to estimating the distortion coefficients for the thermal camera, we go through a process referred to as camera calibration [99]. In our work, the *OpenCV Calibration Tool* [25] was used in order to calibrate the camera. Usually the calibration is performed by using a chessboard pattern on a sheet of paper and capturing images of this pattern with the camera from several different angles.

However, this is not possible with a thermal camera, as the chessboard pattern will not show up on the paper because the temperature of the paper is uniform. To

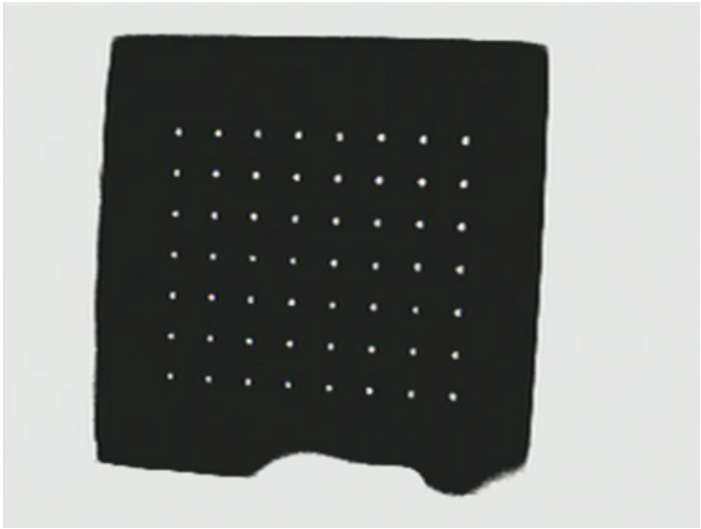

Figure 2.8: An image used for calibration of the thermal camera. By cooling a plate with a circular pattern down to $\approx 0^{\circ}$ it is seen that the pattern is clearly visible and ready for automatic circle detection.

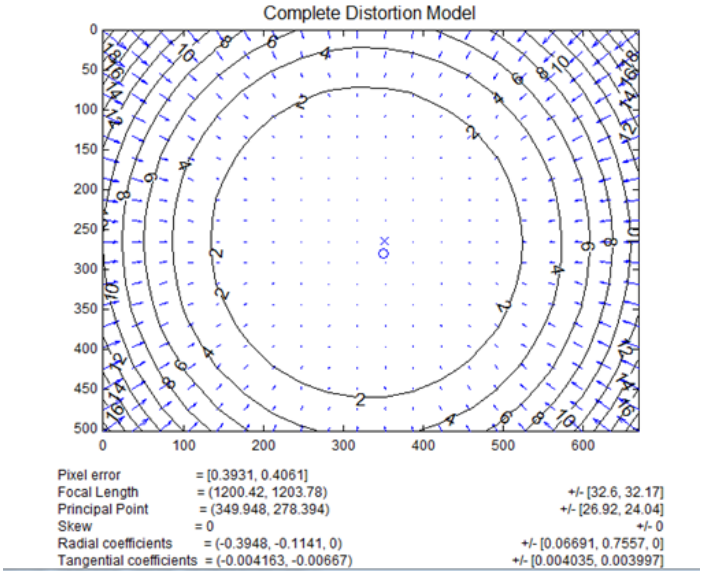

Figure 2.9: Distortion model for the 9 mm focal length FLIR Tau2 thermal camera.

make the calibration process more in line with that of calibrating a regular camera, a $2mm$ thick plate with 8×7 holes that was $3mm$ wide and $2cm$ apart from each other was made. After cooling the plate down to $\approx 0°C$, several images of the plate from different angles and positions were taken in room temperature ($\approx 24°C$). An example image is shown in Figure 2.8. The thermal camera was configured to be sensitive to the temperature range $5° - 15°C$, effectively creating an almost binary like image, hence making the plate pattern easily detectable. The result (estimated $\boldsymbol{A}$ matrix and distortion coefficients) from the calibration process can be seen in Figure 2.9.

2.3.2 Gimbal

In choosing a gimbal to use with the object detection and tracking system described in this chapter, gimbals were evaluated by the following criteria

- Light Weight
- Small in size
- Should be retractable
- Compatible with the FLIR Tau2 thermal camera
- At least two degrees of freedom (pan and tilt)

A gimbal that was found to satisfy all of these criteria was the retractable R-BTC88 gimbal [4]. A more detailed description of the gimbal and its characteristics can be found in Appendix C.

Note that a gimbal with three degrees of freedom (pan, tilt and roll) is especially useful when the camera should be stabilized pointing directly downwards toward the ground during UAV flight. A pan and tilt gimbal (two degrees of freedom) will for instance not be able to stabilized the camera to point directly downwards when the UAV is flying in a straight line. This is because it can not be used to compensate for small perturbations in the UAV's roll angle in this position. However, the object detection and tracking system outlined in this thesis does not utilize algorithms that requires the camera to be perfectly stabilized pointing directly downwards, hence a two degrees of freedom gimbal is sufficient for the use in the overall object detection and tracking system.

2.4 Implementation on UAV Platforms

The object detection and tracking system was implemented in two different UAV platforms with the setup shown in Figure 2.6 as a basis, together with a ground

station with Figure 2.7 as a basis. The software and hardware for these two different UAV platforms and ground stations only have some minor differences between each other, illustrating that the described system is agile and easily adapted to different UAV platforms. A more detailed description of the specifics of the implementation process for each UAV platform is given in Sections 2.4.1 and 2.4.2.

To have enough computational power to run an MPC algorithm on-board the UAV is in many cases not straight forward, especially for the more light-weight UAVs. For such systems, a possibility is to let a ground station computer do the MPC optimization, and then send the resulting control input to the UAV's autopilot for execution. This would entail that the ground station MPC would have to receive real time information about the UAV's current position, heading and turn rate. A way to achieve this and incorporate an off-board MPC component into the object tracking system is illustrated in Figure 2.10. A task in DUNE communicating with

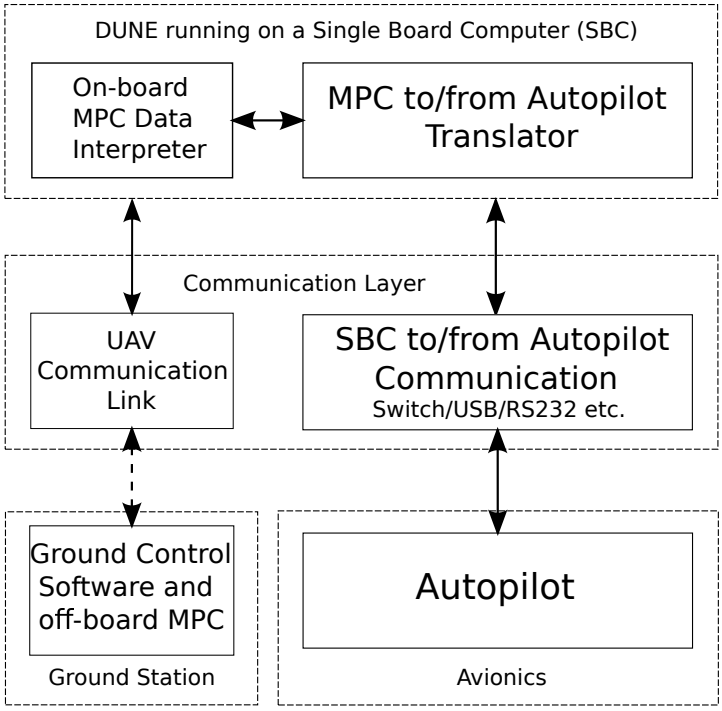

Figure 2.10: Illustration of the data flow when an off-board MPC running on a powerful computer in the ground station is included in the object tracking system's architecture. The on-board MPC data interpreter and the MPC to/from autopilot translator are both implemented in DUNE as separate tasks.

the autopilot through a suitable interface (e.g this could be a Ethernet, USB or RS232 connection between the autopilot and the on-board single board computer) receives the telemetry data and translates this to messages that the "On-board MPC Data Interpreter" module can process. The on-board MPC data interpreter is also implemented as a task in DUNE, which in turn works as a relay between the on-board hardware and the off-board MPC algorithm. The on-board MPC data interpreter sends the current UAV telemetry data, as well as the tracked object's newest position and velocity estimates to the off-board MPC. The off-board computer then finds the optimal control input for the UAV, which in turn sends this back to the on-board MPC data interpreter. The control input is then transmitted back to the autopilot using the implemented translator task.

The ground station network is communicating with the UAV system through a 5.8 GHz radio link, which is identical to the one found on-board the UAV platform. This is the communication channel the off-board MPC uses to send and receive data to and from the on-board MPC data interpreter module, enabling remote control of the UAV.

2.4.1 Skywalker X8

The Skywalker X8 (X8 for short) is a very light-weight UAV made out of styrofoam with the focus of supplying a cheap expendable aircraft for short-duration ($\leq$ 1 hour) UAV flights [12]. The characteristics of the X8 UAV platform can be summarized as follows

- 212 cm wing span
- 60 cm length
- ≈ 60 min flight
- ≈ 1 kg of payload (excluding batteries)

The communication link module of the payload is based on the communication architecture described in [108]. This entails using an Ubiquity Rocket M5 airMAX modem [5] as the 5.8 Ghz communication link between the UAV payload and the ground station. Furthermore, the payload consists of the retractable R-BTC88 pan-tilt gimbal, a FLIR Tau2 thermal camera and an on-board single board computer capable of performing real-time image processing. The single board computer used in the X8 payload is mainly limited by the X8's relatively restrictive maximum payload weight. The ODROID-U3 [6], which is an ARM-based single board computer and has a 1.7 GHz quad core CPU and 2 GB RAM was found to

Figure 2.11: The Skywalker X8 UAV platform

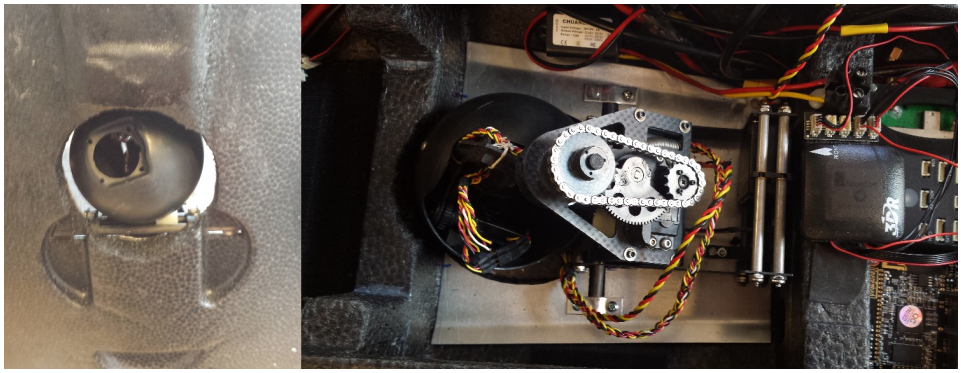

Figure 2.12: Integration of a retractable gimbal with the Skywalker X8

have sufficient computational power as well as being within the weight limits of the X8. A detailed description of the ODROID-U3 can be found in Appendix E.

The autopilot used in this payload is the open-source ArduPilot [7] autopilot (more specifically the Pixhawk PX4 [11]), and the autopilot ground station software shown in Figure 2.7 is the open-source software APM Planner 2 [8].

The physical integration of the hardware into the X8 fuselage was a somewhat challenging task, since the space is limited and weight of the hardware components has to be distributed so that the center of gravity is placed approximately beneath the wings. The retractable gimbal was placed in the middle of the fuselage (see Figure 2.12), and the remainder of the components was placed according to Figure 2.13. The battery, a lithium polymer (LiPo) battery consisting of 4 cells (1 cell has 3.7 V) in a series (4S), was placed in the front of the fuselage. Note that since the LiPo battery supplies a variable voltage ($14-17$ V depending on how much power is left in the battery), the payload has to utilize voltage regulators in order to supply

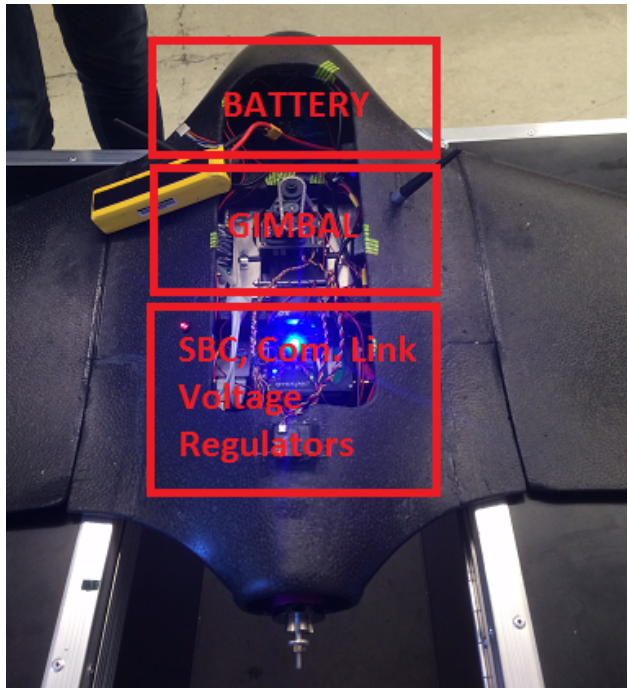

Figure 2.13: The placement of hardware components in the X8 payload.

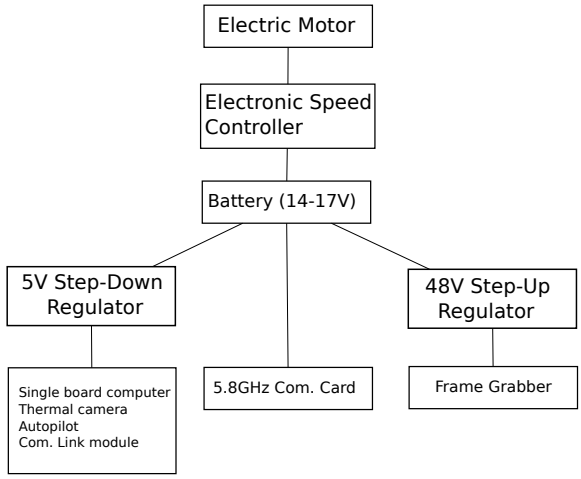

Figure 2.14: The X8 payload's power scheme.

a steady voltage to components that need a specific voltage in order to function and operate. For the X8, the ODROID-U3, FLIR Tau2 thermal camera and some parts of the communication link module requires a steady 5V power source, while the Axis M7001 frame grabber requires a steady 48 V power source. The Ubuiquiti Rocket M5 can handle a variable voltage ($10-24$ V), hence was connected directly to the on-board battery. This power scheme is illustrated in Figure 2.14. Notice that the battery is connected to an electrical speed controller (ESC), which is a control unit responsible for varying the thrust (speed) of the electrical propulsion.

2.4.2 UAV Factory Penguin-B

The Penguin-B is a more advanced and bigger platform than the Skywalker X8, and is designed for long-duration (≥ 5 hours) UAV flights [10]. It has the following characteristics

- 330 cm wing span
- 200 cm length
- $10+$ hours flight
- 3 kg of payload (excluding fuel)

Figure 2.15: The UAV Factory Penguin-B platform (left) and its universal payload mount. The payload is mounted in the front bottom of the Penguin-B UAV.

The architecture of this system is not based on [108], but still uses the Ubiquity Rocket M5 airMAX modem [5] as the 5.8 Ghz communication link between the UAV payload and the ground station. This is convenient because it enables the communication link in the payload and ground station when operating the Penguin-B UAV platform to remain identical to the communication link used when operating the Skywalker X8. However, the autopilot used for this UAV platform is not the open-source ArduPilot, but the industry standard autopilot Piccolo made by CloudCap Technology [9]. Hence, the autopilot ground control software is not the

Piccolo Command Center, which is the ground control station software developed by the same company for use with the Piccolo autopilot. The Piccolo autopilot has an internal 2.4 GHz antenna for communication between the Piccolo autopilot and the Piccolo Command Center. Furthermore, since the Penguin-B is able to carry more payload weight than the Skywalker X8, an extended payload including a camera sensitive to the visible light spectrum (electro-optical (EO) camera) was used for this platform. This camera was not placed inside a gimbal, but rather placed in a strap-down position.

The single board computer used for the Penguin-B payload was the PandaBoard [3]. The PandaBoard is an ARM-based single board computer with a 1.2 GHz dual core processor and 1 GB RAM. A detailed description of the PandaBoard can be found in Appendix E. The remainder of the payload and ground station is similar to the setup described in the previous subsection.

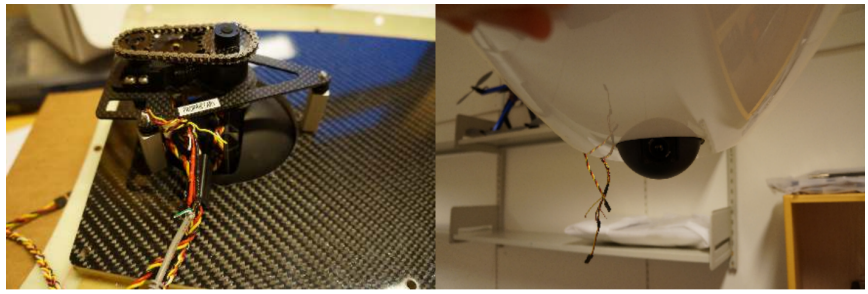

Figure 2.16: Gimbal integration on the Penguin-B UAV Platform

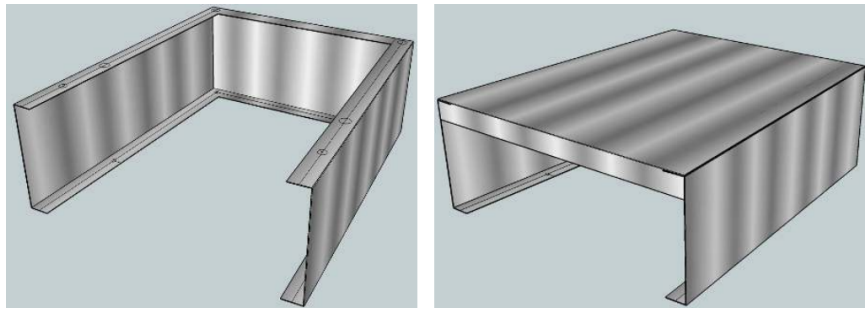

Figure 2.17: The Penguin-B payload housing without (left) and with a lid (right).

The physical integration of the hardware into the Penguin-B UAV platform is based on the universal payload mount that comes with the Penguin-B platform. This payload mount is illustrated in Figure 2.15. The mount is designed so that changing

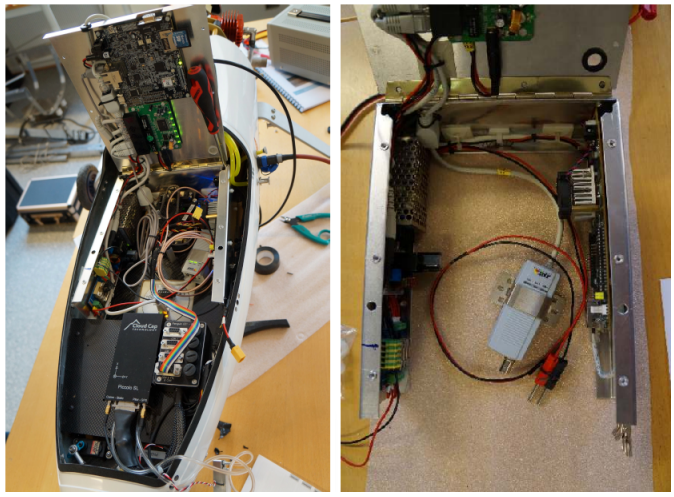

Figure 2.18: The Penguin-B payload. The payload is mounted on the payload housing (right) and the housing itself is then mounted on the universal payload mount (left).

the complete payload of a Penguin-B UAV can be done in a matter of minutes, since the complete payload should be integrated on the payload mount. In order to fit all the hardware required for the complete object detection and tracking system on one of these mounts, a payload housing was designed. The designed payload housing is illustrated in Figure 2.17, and is a housing of aluminium which the hardware components can be mounted on. Figure 2.18 show how the components have been mounted on the payload housing, as well as how the EO camera and the thermal camera (in a gimbal) has been mounted on the payload mount itself. The integration of the BTC-88 gimbal into the universal payload mount is shown in Figure 2.16. The electrical power scheme for the Penguin-B payload is identical to the power scheme for the Skywalker X8 payload, with the exception that the Penguin-B payload also has a power line directly from the battery to the EO camera.

CHAPTER 3

MPC Path Planning

This chapter is based on [63] (F. S. Leira, E. Skjong, S. A. Nundal, T. A. Johansen, and T. I. Fossen. Autonomous unmanned aerial vehicle object tracking framework using model predictive control and camera gimbal. *AIAA Journal of Guidance, Control, and Dynamics*, 2017, *Submitted*).

3.1 Introduction

In a scenario where one has a number of (moving) objects of interest that one wants to keep track of using a single UAV equipped with a video camera, the resource allocation of the video camera will be crucial. Objects will have to be revisited from time to time in order to verify if they have moved away from their last known positions and to update their velocity measurements, while at the same time giving all of the tracked objects enough camera time to have a sufficiently good estimate of the movement of the objects. One objective in such a scenario could also be to observe the objects to perform object recognition and other vision-based analysis.

Many different approaches to the problem of multi-object tracking is found in the literature. This problem is not restricted to UAVs, as there exists many different remote sensing platforms. However, finding or adapting solutions that can be applied for cameras in UAVs is still useful and necessary since e.g methods for radar-based multi-object tracking are not directly applicable. The next section covers the recent efforts in the field of (multi-)object tracking with UAVs.

3.1.1 Related Work

The different approaches to the problem of multi-object tracking can be divided into two categories; resource allocation object tracking, and information driven object tracking [45]. In resource allocation tracking the tracking problem is for-

mulated as a visitation problem, i.e finding the path that has the vehicle visit all tracked objects in the shortest time frame possible. For the information driven approach, the path is often planned dynamically based on the information reward of visiting a tracked object. I.e, this could be based either on the time duration since the last visit, the uncertainty of a tracked object's position and velocity estimate, or the information gained from the analysis of other observations of the tracked object.

[57] successfully implemented a resource allocation based multiple object tracking algorithm minimizing the total mission time required for n UAVs to collect a specified amount of information about m objects. Although the algorithm works for any given number of UAVs, it does not take into account moving objects. [67] is a similar solution, but accounts for objects moving with a constant velocity. [41] defines the multi-object tracking problem as an optimization problem with non-convex constraints using MILP. The non-convex constraints are defined to make the UAV avoid collisions and be fuel efficient. Furthermore, there is a constraint requiring communication capacity, making sure that the UAV's path is made in such a way that it will be able to communicate with the ground station. However, all of the above mentioned path planners are run as a one-time-calculation, off-line, having the entire path planned prior to the UAV flight.

Information driven approaches have also proven useful as a solution to the object tracking problem. [86] implements an object tracking framework based on partially observable Markov decision processes (POMDPs). Using a kinematic formulation of the UAV, they state the tracking problem as an optimization problem to minimize the mean-squared error between the estimated and actual object position. I.e, the tracking algorithm prioritizes the tracking of objects whose position estimate is uncertain. The approach considers all tracked objects simultaneously when calculating an optimal path, which greatly increases the complexity of the optimization problem with increasing numbers of tracked objects. To solve the optimization problem they use an approximation method called nominal belief-state optimization (NBO) and a "look-ahead" horizon. Using a powerful desktop computer and a total of 2 objects, they find an approximate solution to the optimization problem in an average of 350 ms. Although this is similar to the computation time required for the solution presented in this chapter, the "look-ahead" horizon of 6 timesteps is relatively short, and increasing it would increase the required computation time. In addition, since the approach presented in [86] considers all tracked objects simultaneously when solving the optimization problem, it is also reasonable to assume that their approach would have to be modified in order to perform

real time tracking of several objects simultaneously. [45] formulates the object tracking problem as an optimal control problem. It implements a model predictive controller (MPC) which seeks to find the flight path that minimizes the weighted traces of the tracked objects' predicted state estimation covariance matrices. Although highly effective for object tracking purposes, it has a high one-time computational cost, computing a long tracking path consisting of UAV waypoints. This makes the algorithm vulnerable to abrupt changes in the tracked object's motion dynamics. If the tracked objects do not follow the initially guessed motion model, the proposed tracking algorithm may quickly fail to track the moving objects. [75] presents an approach to path planning based on Lyapunov vector field guidance. A vector field is generated using the object location (or average object location in case of a cluster of objects) and a desired stand-off distance. The desired stand-off distance is made so that if the UAV has this stand-off distance all objects currently tracked are within the UAV sensor's field of view. This is an effective approach, but requires that the algorithm manages how to cluster objects together, and constantly changes the stand-off distance in order to include all objects in the sensor's field of view simultaneously.

The above mentioned methods describe approaches to implement high-level object tracking behaviour in UAVs. However, one also needs low level object tracking control. That is, the low-level control is here defined as the process of generating the actual flyable paths for the UAV. In some cases, the high-level and low-level UAV controls are interconnected in such a way that solving one problem yields both the high-level and low-level control simultaneously (e.g [45]). However, in general this is not the case, and one needs one approach to do the high-level control and another approach to do the low-level path generation for the UAV. In the literature one finds different general approaches to the low-level path generation; roadmap [43], cell decomposition [19], potential field [22], probabilistic [13] and optimal control [82][47]. An in-depth analysis of the different low-level path generators is not within the scope of this work, hence the reader is referred to [103] for an extensive overview of the existing approaches.

In order to make object tracking using fixed-wing UAVs more efficient, such systems are often equipped with a gimbal. In [84] a fixed-wing UAV is equipped with a pan and tilt unit in order to make the process of tracking objects easier. However, the gimbal control is not completely automated, as the gimbal position is dependent on feedback supplied by the UAV operator. [29] presents an elegant solution for automated gimbal control coupled with a path planner for object tracking, but only utilize the pan mechanism of their gimbal. Both [85] and [91] presents auto-

mated gimbal controllers for object tracking with fixed-wing UAVs using both a pan and a tilt mechanism, but the gimbal controllers are decoupled from the path planner. This means that the UAV, unless explicitly considered by the path planner, might end up flying in such a manner that there does not exist a solution which will point the camera directly towards the tracked object. Therefore, this approach might be limiting for the chosen path planner algorithm.

3.1.2 Contribution

In this chapter, the focus is on the development, implementation and testing of an MPC algorithm to be used with the object detection and tracking framework presented in Chapter 2. Chapter 3 extends the work presented in [96] by describing the implementation details necessary to bring the MPC and multi-object tracking framework from a Hardware-In-the-Loop (HIL) test bed to experiments on actual UAV platforms. The MPC algorithm can be described as a hybrid between a resource allocation based and an information driven based object tracking approach. It uses a resource allocation based method in order to find which order the UAV should track objects of interest. However, once the object(s) to track are decided, the framework switches to an information driven approach, where an MPC is used to generate an optimal object tracking path with respect to a chosen cost function, which in turn will maximize the accuracy of the object's position and velocity estimations. The proposed solution in this chapter has its main focus on being able to track an arbitrary and changing number of moving objects simultaneously, while at the same time being flexible enough to quickly adapt for potential changes in the position and velocity of the tracked objects. Although not illustrated in the results of this chapter, the object tracking framework is able to adapt to changes in the position and velocity of the tracked objects by the use of an object detection, tracking and recognition module which continuously estimates and updates a motion model for each tracked object when they are observed by the UAV. Such an object detection and tracking module is developed in Chapter 4. Note that this chapter's focus is on the tracking process, i.e keeping track of already detected objects, and is not concerned with the process of searching for new, undetected objects. Although the framework is designed to allow to incorporate such behaviour (searching) with the tracking process, this is not investigated in this chapter. Search algorithms that could be incorporated with the MPC algorithm and the object detection and tracking framework presented in this thesis can be found in e.g [38] and [87].

The MPC developed in this chapter is responsible for steering the UAV towards regions of interest, as well as optimally controlling the gimbal attitude. By using

a gimbal in the UAV payload system, the goal is to place objects of interest in the center of the camera image for as long as possible. Placing the control of the UAV's path and the gimbal's attitude trajectories in the same controller enables enhancements in the performance and functionality of the system. This could be proven advantageous when the quality of the object tracking is to be increased, or the tracking behaviour has to be altered. Consequently, the main contributions of this chapter are in the description, formulation, implementation and testing of an MPC algorithm to be used as a path planner an gimbal controller in the overall object detection and tracking system proposed in this thesis. The MPC algorithm's only requirement for the UAV is that the UAV has an autopilot that can send telemetry data and receive commands to and from an external source.

3.1.3 Overview of the Chapter

The remainder of the chapter is organized as follows. First, an in-depth description of the MPC used to enable the UAV to perform object tracking is described. Second, challenging aspects regarding the integration of the proposed MPC algorithm into the general object detection and tracking framework is addressed. Third, the performance of the total object tracking system is evaluated by implementing the system in two UAV platforms and conducting a total of three field tests. Finally, the chapter is summarized and concluded.

3.2 Model Predictive Control

An autopilot is used to provide the low-level control for maintaining the UAV's air speed and altitude, as well as the UAV's attitude. In this thesis it is assumed that the UAV has an autopilot that takes waypoints as control input, and compensates for wind disturbances. Furthermore, it is assumed that the UAV platform has a two-axes gimbal system installed in the UAV's payload. The gimbal's rotation along these two axes (pan and tilt), is hereby denoted β and α, respectively. The control objective for the MPC is to control the UAV's path, dependent on detected (and identified) objects to be tracked, and place objects of interest within the image frame of the camera embedded in the gimbal. Hence, the control objective is to provide optimal UAV waypoints and gimbal attitude (β and α) depending on the position of objects being tracked. The main requirements related to the object detection, tracking and recognition module and the autopilot are:

Requirement 1

A machine vision module provides an object list with object positions and velocities. The initialization procedure of each MPC update selects a geographically referenced object, or a cluster of objects, from the object list to track within a finite time horizon. The objects to be tracked at a given time horizon are selected based on the UAV's and the objects' positions and velocities, as well as priorities related to previous visitations and user specifications.

Requirement 2

The autopilot controls the UAV relative to the MPC's optimal trajectory calculations. The autopilot is also responsible for maintaining a desired air speed and altitude in the presence of wind disturbances.

Before presenting the MPC's mathematical formulation, the UAV and gimbal system dynamics are addressed in the two following sub-sections.

3.2.1 UAV Kinematics

A local body frame fixed to the UAV's coordinate origin (CO) denoted by {u} and an East-North-Up (ENU) frame fixed to the Earth denoted by {e} are considered as the two main coordinate frames. In addition a third coordinate frame, {g} is defined, which is a body frame fixed to the center of the gimbal. The different coordinate frames used in this work are represented in Fig. 3.1. For the autonomous UAV system, only the horizontal plane motions are considered, as constant altitude is assumed. This is a fair assumption as the UAV system is used for recognizance, and, due to safety reasons, a change in altitude should usually be determined and executed by the system's operator in agreement with airspace regulation and air traffic control. With these assumptions in mind the UAV's horizontal position can be simplified and expressed in the {e} frame by[1]

$$\boldsymbol{\eta} = \begin{bmatrix} x, & y, & \psi \end{bmatrix}^\top, \tag{3.1}$$

which represents a 3-DOF system where x and y are North and East positions, respectively, relative to a predetermined origin, and ψ is the yaw angle. Since only the horizontal plane motions are considered, the matrix to rotate a vector from the {u} frame to the {e} frame can be simplified as a single rotation of ψ about the {e} and {u} frame's coinciding z-axes, i.e $\boldsymbol{R}_z(\psi)$ (as defined in [39]). Hence, assuming that the associated generalized velocities of the UAV given in the {u}

[1] The notation is adopted from [39].

frame is,

$$\boldsymbol{\nu} = \left[\begin{array}{ccc} \nu_x^u, & \nu_y^u, & r \end{array} \right]^\top \tag{3.2}$$

we have that

$$\dot{\boldsymbol{\eta}} = \mathbf{R}_z(\psi)\boldsymbol{\nu} \tag{3.3}$$

The UAV's course angle χ is the sum of the crab angle σ and the yaw angle ψ [21],

$$\chi = \psi + \sigma, \tag{3.4}$$

where the crab angle is given by [21]

$$\sigma = \arctan\left(\frac{\nu_y^u}{\nu_x^u}\right). \tag{3.5}$$

Since ν_y^u is small, (3.3) can be approximated by using the course angle χ. Moreover,

$$\dot{\tilde{\boldsymbol{\eta}}} = \mathbf{R}_z(\chi)\tilde{\boldsymbol{\nu}}, \tag{3.6}$$

where $\tilde{\boldsymbol{\eta}} = [x, y, \chi]^\top$, $\tilde{\boldsymbol{\nu}} = [\nu^u, 0, r_\chi]^\top$, $\dot{\chi} = r_\chi = r$ and $\nu^u = \sqrt{(\nu_x^u)^2 + (\nu_y^u)^2}$ is the speed over ground. The representation (3.6) may be used instead of (3.3) since its implementation does not require a compass or other heading sensor. Table 3.1 summarizes the symbols used for the MPC formulation.

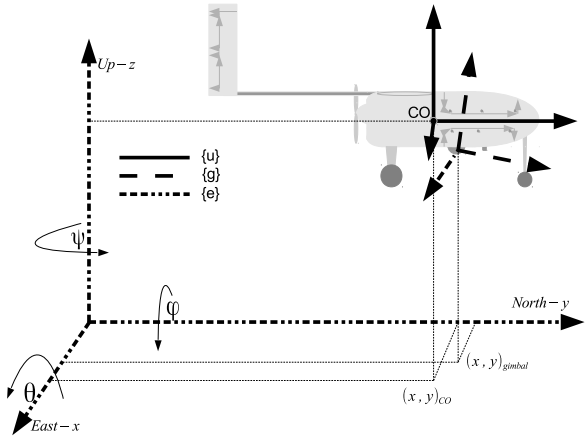

Figure 3.1: Representation of the ENU and body frames.

Table 3.1: Symbols and notation related to the MPC formulation.

Symbol	Explanation	Unit
$\mathbf{r} = [x, y, z]^\top$	UAV's position vector in frame {e}	m
$\boldsymbol{\eta} = [x, y, \psi]^\top$	UAV's position and attitude vector in frame {e}	m rad
$\boldsymbol{\nu} = \left[\nu_x^u, \nu_y^u, r\right]^\top$	UAV's velocity and angular rate vector in frame {u} relative {e}	$\frac{m}{s}$ $\frac{rad}{s}$
χ	UAV's course angle relative frame {e}	rad
r_χ	UAV's course rate relative frame {e}	$\frac{rad}{s}$
σ	Crab angle	rad
$\tilde{\boldsymbol{\eta}} = [x, y, \chi]^\top$	UAV's position and attitude vector relative frame {e} using the course angle χ	m rad
$\tilde{\boldsymbol{\nu}} = [\nu^u, 0, r_\chi]^\top$	UAV's velocity and angular rate vector in frame {u} relative {e} based on the course angle χ	$\frac{m}{s}$ $\frac{rad}{s}$
$\boldsymbol{\Theta} = [\phi, \theta, \psi]^\top$	UAV's roll, pitch and yaw angles relative frame {e}	rad
$\dot{\boldsymbol{\Theta}} = [p, q, r]^\top$	UAV's roll, pitch and yaw rates relative frame {e}	$\frac{rad}{s}$
$\mathbf{r}_j^{obj} = \left[x_j^{obj}, y_j^{obj}, z_j^{obj}\right]^\top$	Object j's position vector in frame {e}	m
$\boldsymbol{\nu}_j^{obj} = \left[\nu_{x,j}^{obj}, \nu_{y,j}^{obj}, \nu_{z,j}^{obj}\right]^\top$	Object j's velocity vector in frame {e}	$\frac{m}{s}$
$[\alpha, \beta]$	Gimbal's tilt and pan angle relative frame {u}	rad β

3.2.2 Gimbal Kinematics

The gimbal system has two DOF, tilt (α), which is rotation about the gimbal frame's x-axis, and pan (β), which is rotation about the gimbal frame's z-axis. The MPC should provide the autopilot with optimal pan and tilt angles placing the object(s) of interest in the center of the camera's image frame. Assuming an object j with position $\mathbf{r}_j^{obj} = [x_j^{obj}, y_j^{obj}, z_j^{obj}]^\top$ expressed given in {e} and the position of the center of the camera's image frame, $\mathbf{r}^c = [x^c, y^c, z^c]^\top$ also expressed in {e}), the desired tilt and pan angles placing the object j in the middle of the camera image can be stated as a mapping dependent on the UAV's position and attitude and object j's position,

$$\begin{aligned} \alpha_{d,j} &= \lambda_g \left(\mathbf{r}^c, \mathbf{r}_j^{obj}, \phi, \theta, \psi \right) \\ \beta_{d,j} &= \mu_g \left(\mathbf{r}^c, \mathbf{r}_j^{obj}, \phi, \theta, \psi \right), \end{aligned} \tag{3.7}$$

where θ and ϕ are the UAV's pitch and roll angles, respectively. λ_g and μ_g are functions that yields a desired gimbal attitude. These mappings are stated in (3.9) and (3.10). The desired gimbal attitude is calculated based on placing object j in the middle of the camera's image frame, and is calculated based on the UAV's attitude, which rotates the object coordinates to level the UAV. This can be seen as rotating the Earth frame about the UAV's center of origin, to compensate for the UAV's attitude. The rotated object coordinates are used to calculate the desired pan and tilt angles with a simple trigonometric consideration. The rotation of the object coordinates relative to the UAV's CO is illustrated in Fig. 3.2. For object j the rotation, hence calculation of the new coordinates, can be expressed by

$$\hat{\mathbf{r}}_j^{obj} = \mathbf{R}_z(\psi)\mathbf{R}_y(-\phi)\mathbf{R}_x(-\theta)\mathbf{R}_z^\top(\psi) \left(\mathbf{r}_j^{obj} - \mathbf{r} \right) + \mathbf{r}. \tag{3.8}$$

Before proceeding with the calculations of the desired gimbal angles, the orientation of the gimbal attitude must be defined. In this work, using a two-axes pan-tilt unit (PTU), the gimbal's tilt angle is bounded within $\alpha \in [\alpha_{min}, \alpha_{max}]$, where $\alpha = 0$ is defined when the camera is pointing down perpendicular to the longitudinal body axis, and $\alpha = \pi/2$ when the camera is pointing parallel to the plane given by the x and y axes in the {u} frame. The gimbal's pan angle is bounded within $\beta \in [\beta_{min}, \beta_{max}]$, where $\beta = 0$ when the gimbal (camera) is pointing towards the UAV's nose, assuming zero tilt angle. The positive direction of rotation is counter-clockwise when seeing the gimbal from above.

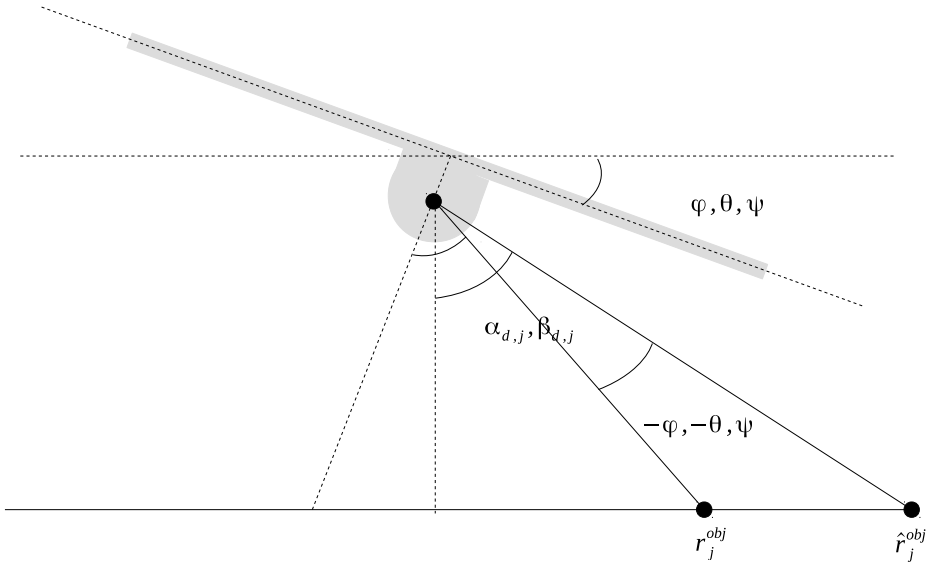

Figure 3.2: Rotating object j coordinates to counteract the UAV's attitude.

With a simple trigonometric consideration the desired gimbal attitude placing object j in the center of the camera's image frame can be derived,

$$\alpha_{d,j} = \arctan\left(\frac{\sqrt{(x^c - \hat{x}_j^{obj})^2 + (y^c - \hat{y}_j^{obj})^2}}{z^c - \hat{z}_j^{obj}}\right), \tag{3.9}$$

$$\beta_{d,j} = \begin{cases} -\arctan\left(\frac{\hat{x}_j^{obj}-x_c}{\hat{y}_j^{obj}-y_c}\right) - \psi & \forall x_c \leq \hat{x}_j^{obj} \wedge y_c < \hat{y}_j^{obj}, \\ -\pi + \arctan\left(\frac{\hat{x}_j^{obj}-x_c}{y_c-\hat{y}_j^{obj}}\right) - \psi & \forall x_c \leq \hat{x}_j^{obj} \wedge y_c > \hat{y}_j^{obj}, \\ \pi - \arctan\left(\frac{x_c-\hat{x}_j^{obj}}{y_c-\hat{y}_j^{obj}}\right) - \psi & \forall x_c \geq \hat{x}_j^{obj} \wedge y_c > \hat{y}_j^{obj}, \\ \arctan\left(\frac{x_c-\hat{x}_j^{obj}}{\hat{y}_j^{obj}-y_c}\right) - \psi & \forall x_c \geq \hat{x}_j^{obj} \wedge y_c < \hat{y}_j^{obj}, \\ -\frac{\pi}{2} & \forall x_c < \hat{x}_j^{obj} \wedge y_c = \hat{y}_j^{obj}, \\ \frac{\pi}{2} & \forall x_c > \hat{x}_j^{obj} \wedge y_c = \hat{y}_j^{obj} \end{cases} \tag{3.10}$$

Note that equations (3.9) and (3.10) are used only to find a desired value for α and β for a given object j. That is, $\alpha_{d,j}$ and $\beta_{d,j}$ are calculated only once for each step in the predict horizon of the MPC, and are treated as constants in the MPC cost function given in Section 3.2.4.

3.2.3 Object Dynamics

The object detection, tracking and recognition module is responsible for identifying objects. From the camera's images, objects are detected and object positions and velocities are estimated by tracking them for some time. The estimated object information is delivered to the MPC in the form of an object list, which is used by the MPC to predict the objects' position throughout the MPC horizon $[t, t+T]$. For object j and time step Δ_k the discrete-time dynamic equations are written as

$$\begin{aligned} x_{j,t+T}^{obj} &= x_{j,t}^{obj} + \sum_{k=t+1}^{t+T} \nu_{x,j,t}^{obj} \Delta_k \\ y_{j,t+T}^{obj} &= y_{j,t}^{obj} + \sum_{k=t+1}^{t+T} \nu_{y,j,t}^{obj} \Delta_k. \end{aligned} \tag{3.11}$$

The last term in (3.11) shows that the future object velocity is assumed constant and equal to the instantaneous velocity at the start of the MPC horizon, i.e. $\nu_{j,k}^{obj} =$

$\nu_{j,t}^{obj} \quad \forall k \in [t, t+T]$. Clearly, the future object positions predicted by the MPC will drift from the true positions without updated object positions and velocities as feedback from the object detection, tracking and recognition module. Hence, the coupling between the object detection module and the MPC is crucial for placing the objects of interest within the camera's image frame, especially if the objects are moving with changes in velocity.

3.2.4 Cost Function

To achieve high accuracy and performance of the object tracking, one requires coordination of the UAV and the gimbal system to place the objects of interest within the camera's image frame. However, this has to be achieved within the limitations on both the UAV's path (turn radius and roll angle) and the gimbal system's attitude. The objective, to place the objects as close to the center of the camera's image frame as possible, can be formulated as a Lagrangian problem with integral quadratic cost,

$$\begin{aligned} J_{t+T} =& \frac{1}{2} \sum_{k=t}^{t+T} \left[\left(\mathbf{h}(\mathbf{z}_k) - \mathbf{h}_d \right) \mathbf{Q} \left(\mathbf{h}(\mathbf{z}_k) - \mathbf{h}_d \right)^\top \right] \\ &+ \frac{1}{2} \sum_{i=t}^{t+T} \left[\mathbf{u}_k \mathbf{R} \mathbf{u}_k^\top \right], \end{aligned} \tag{3.12}$$

where the dynamic state vector $\mathbf{z}_k$ is given by

$$\mathbf{z}_k = \left[x_k, y_k, \chi_k, \alpha_k, \beta_k, x_{1,k}^{obj}, y_{1,k}^{obj}, \ldots . x_{n,k}^{obj}, y_{n,k}^{obj} \right]^\top, \tag{3.13}$$

Here n is the number of objects detected by the machine vision module and k is the time instance. $\mathbf{Q} \in \mathcal{R}^{3n \times 3n}$ and $\mathbf{R} \in \mathcal{R}^{3 \times 3}$ are diagonal weight matrices. The controlled variables' vector, $\mathbf{u}$ at time step k is given by

$$\mathbf{u}_k = \left[\dot{\alpha}_k, \dot{\beta}_k, r_{\chi,k} \right]^\top. \tag{3.14}$$

The predict horizon, T, defines how far into the future the cost function for the system should be considered. A control horizon is also defined, and decides for how many time steps the control input should be optimized. Although not always the case, for this thesis we assume that the predict horizon is the same length as the control horizon, i.e T steps.

As can be seen, rates $\dot{\alpha}_k$ and $\dot{\beta}_k$, are chosen as controls instead of α_k and β_k. This is because minimization of angular rates, which is represented in the last term of

(3.12), would give a smoother gimbal motion which could improve the camera image's quality. Also the UAV's course rate is chosen as control instead of the course angle to reduce rapid motions impairing the quality of the object tracking.

The distance between the UAV and the objects of interest, together with the gimbal attitude, are given by the $3n$-dimensional mapping

$$\mathbf{h}(\mathbf{z}_k) = \begin{bmatrix} \sqrt{(x_k - x_{1,k}^{obj})^2 + (y_k - y_{1,k}^{obj})^2} \\ \vdots \\ \sqrt{(x_k - x_{n,k}^{obj})^2 + (y_k - y_{n,k}^{obj})^2} \\ \alpha_k \\ \beta_k \\ \vdots \\ \alpha_k \\ \beta_k \end{bmatrix}^\top . \tag{3.15}$$

The desired optimal gimbal attitude and distance between the UAV and the object(s) to be tracked can be stated as

$$\mathbf{h}_d = \left[\delta \mathbf{1}^{n\times 1}, \alpha_{d,1,k}, \beta_{d,1,k}, \ldots, \alpha_{d,n,k}, \beta_{d,n,k}\right]^\top , \tag{3.16}$$

where the UAV's desired turning radius is given by δ. $\alpha_{d,1..n,k}$ and $\beta_{d,1..n,k}$ are the desired gimbal tilt and pan angles given by (3.7). The turning radius should be carefully chosen relative to the UAV's maximum bank angle (and thus roll angle [66]) and the gimbal's maximum tilt angle due to the fact that the UAV's roll angle will at some level counteract the gimbal's tilt angle in a circular loitering motion around the object(s) to be tracked.

An interesting extension to the cost function is proposed in [95] where a collision avoidance algorithm is used, which projects the camera's image frame corners down on the ground relative to the UAV's and gimbal's position and attitude, and uses the projected image corners to determine whether objects are within the camera's image frame. If the objects to be tracked are not within the projected frame, an additional penalty to the cost function is introduced.

3.2.5 Constraints

The UAV's manoeuvrability is directly coupled to the UAV's physical limitations. As the autopilot is assumed to maintain the UAV's airspeed and altitude, only

changes in the turn radius (which is dependent on the course rate r_χ) will have an impact on the calculation of the UAV's path. That is, assuming level flight, we have from [21] that

$$\delta = \frac{V^2}{g \cdot \tan(\theta)} \tag{3.17}$$

where δ is the turn radius, g is the standard acceleration due to gravity, V is the airspeed of the UAV and θ is the UAV's roll angle. Furthermore, from [76] we have that

$$r_\chi = \frac{g}{V}\tan(\theta) \tag{3.18}$$

Now, assuming the course rate is bounded, i.e. $r_{k,\chi} \in [r_{\chi,min}, r_{\chi,max}]$, and that the UAV's airspeed is given by ν_c, the minimum turning radius, δ_{min}, can be calculated as follows by combining (3.17) and (3.18)

$$\delta_{min} = \frac{\nu_c}{\max\left(\left|r_{\chi,min}\right|, \left|r_{\chi,max}\right|\right)}. \tag{3.19}$$

Hence, the desired turn rate should be decided according to

$$\delta \geq \delta_{min} \tag{3.20}$$

However, if the turn radius is chosen too small, the UAV's manoeuvrability could be impaired, as its ability to correct for wind disturbances may be limited, which could lead to a low accuracy of the object tracking.

The gimbal's attitude is bounded relative to the gimbal's physical, or operational, limitations, i.e. $\dot{\alpha}_k \in [\dot{\alpha}_{min}, \dot{\alpha}_{max}]$ and $\dot{\beta}_k \in \left[\dot{\beta}_{min}, \dot{\beta}_{max}\right]$. To compensate for disturbances, like wind gusts, the gimbal may require a fast response. However, a fast response could impair the image quality since the camera may then be moving while it is capturing an image. Hence, care must be taken when choosing the dynamic limitations. To summarize, the MPC constraints for the object tracking system can be stated as

$$r_{\chi,min} \leq r_{\chi,k} \leq r_{\chi,max}, \quad \frac{rad}{s} \tag{3.21}$$

$$\alpha_{min} \leq \alpha_{d,j,k} \leq \alpha_{max}, \quad rad \tag{3.22}$$

$$\beta_{min} \leq \beta_{d,j,k} \leq \beta_{max}, \quad rad \tag{3.23}$$

$$\dot{\alpha}_{min} \leq \dot{\alpha}_k \leq \dot{\alpha}_{max}, \quad \frac{rad}{s} \tag{3.24}$$

$$\dot{\beta}_{min} \leq \dot{\beta}_k \leq \dot{\beta}_{max}, \quad \frac{rad}{s} \tag{3.25}$$

3.2.6 MPC Implementation Aspects

Note that even though the dynamics are linear, the non-quadratic cost demands a nonlinear MPC and optimization. The ACADO toolkit[2] [52] is used to implement the MPC formulation using the ACADO C++ API. The MPC problem is discretized into 1 second steps, with a total prediction and control horizon of 20 steps (20 seconds). Although the dynamics of the UAV's attitude changes are faster than the 1 second discretization, we argue that the average course rate stays more or less constant during this time interval, making the motion model given by (3.6) valid. Furthermore, choosing a sufficiently large predict horizon relative to the system's time constants will according to [42] ensure asymptotic stability for a non-linear MPC. A predict horizon of 20 seconds proved to yield stable results for the MPC in the field tests described in Section 3.4. However, note that as the system ultimately relies on the autopilot to perform the low level control of the UAV, the MPC would not be able to make the UAV unstable at any point. This is because the autopilot has its own controllers to ensure stability during flight.

The discretization method is multiple shooting [23], which transforms the MPC problem to a generic Nonlinear Programming (NLP) problem. Furthermore, it does so in such a way that an NLP solver can utilize parallelization in order to speed up the computation time needed to find an optimal solution. ACADO solves this NLP problem by redefining the problem using Quadratic Programming (QP) subproblems, which in turn are solved using qpOASES [35]. The QP solver qpOASES is open source and already interfaced with the ACADO software. To support online changes of constraint limits, the changes will only be executed in the initialization process of the MPC before each new control update calculation. This means that introducing slack variables [73] to prevent the optimization problem from becoming infeasible is not needed.

The MPC problem formulation does not include any disturbance models, e.g. wind models. This is assumed to be a fair simplification due to the closed loop feedback provided within the autopilot. New UAV and gimbal measurements for the initialization of the MPC predictions provides trajectory corrections.

3.3 System Implementation Challenges

Because the MPC is more computationally demanding than the processing power available on the UAV payloads as they were implemented and integrated in Chapter 2, it was decided to implement the MPC on an off-board computer as described in

[2] `www.acadotoolkit.org`

Section 2.4. However, having the main computational component of the MPC located off-board in a ground station introduces some challenges regarding communication delay and/or fault. In the work presented in this chapter, loss of communication between the off-board and the on-board MPC data interpreter is handled by having the autopilot configured so that it will make the UAV loiter around the last received waypoint calculated by the off-board MPC algorithm. This will also happen if there is a large delay in communication between the ground station and the UAV, so that the new control input for the UAV is not received before the UAV has already reached its previously received control input/waypoint. An alternative solution would be to have the off-board MPC module transfer the whole predict and control horizon (20 seconds of control input) to the on-board MPC data interpreter. By doing this, the on-board MPC component could feed the autopilot new waypoints based on the predicted optimal trajectory in the case that communication with the ground station is lost or delayed.

Because of the challenges that arise regarding communication by introducing the off-board MPC component to the system, the ideal solution would be to have the off-board component located on-board. With the recent developments on embedded systems, especially the fact that the CPU of these systems now typically consist of several cores, this could be possible even for some of the more lightweight UAVs. This might require that parallelization is used in order to utilize all CPU cores of such systems. In addition, the predict horizon of the MPC could be shortened (to some degree) for embedded systems running the MPC on-board. In the case that the off-board MPC component is placed on-board, the only difference in the implementation is that the "On-board MPC Data Interpreter" in Figure 2.10 is replaced by a general "MPC" module (also implemented in DUNE), that includes both the core MPC and also functions as the MPC data interpreter between the MPC and the remainder of the system.

3.4 Field Tests

In order to test the performance of the object tracking framework described in the previous sections, several field tests were conducted with both the Skywalker X8 and the Penguin-B UAV platforms and payloads described in Chapter 2. Note that even though the X8 was using the open-source ArduPilot [7] autopilot and the Pengiun-B was equipped with the CloudCap Piccolo [9] autopilot, the only difference between the two systems was the task within DUNE responsible for communicating/translating autopilot messages. For all the tests conducted in this research, it was the performance of the path planning and tracking modules of the system

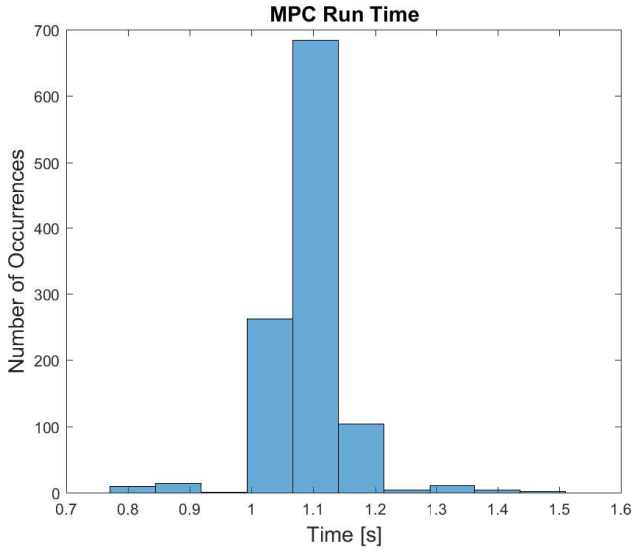

Figure 3.3: A histogram showing the distribution of the run time for (initializing and solving) the MPC algorithm. The average run time is 1.09 seconds, while the minimum and maximum run time observed is 0.7 and 1.6 seconds, respectively. The histogram is based on 1100 runs of the MPC (1100 control outputs calculated), and the bin size is 0.07 seconds.

that was emphasized, hence all objects were assumed to have already known positions and velocities. Moreover, the computer vision module was not doing any real-time on-board image processing in order to give the path planner estimates of the updated position and velocities of the objects. All of the tests was performed using an Intel Core i7 (4800MQ, 2.7GHz), 8GB RAM (DDR3 1600MHz) computer, with ACADO implemented and run using the configuration and parameters described in Section 3.2.5. Fitting a normal distribution to the computation time needed for the MPC to initialize and solve the optimization problem, it was found that the expected time between each calculated control input by the MPC was 1.09 seconds, with a standard deviation of 0.06 seconds. A histogram of the run times for 1100 MPC optimizations is shown in Fig. 3.3.

The field tests were divided into three different scenarios, all of which will be described in detail in this section.

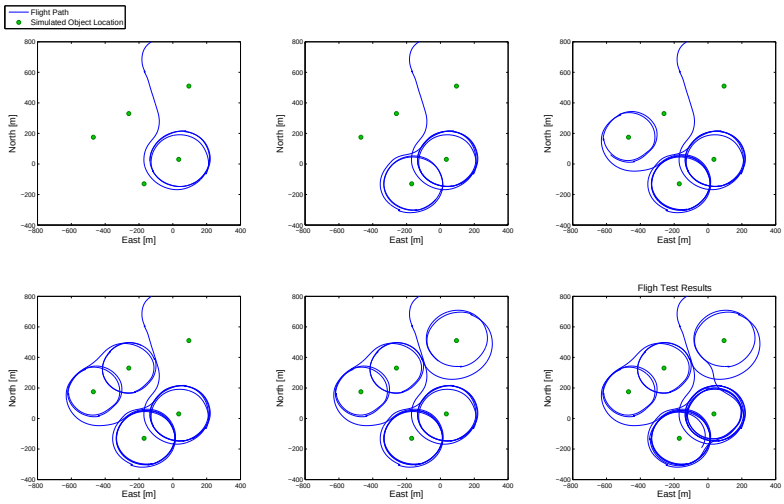

Figure 3.4: Field test conducted using the Penguin-B UAV platform. Five simulated stationary objects of interest were successfully tracked by the UAV. The blue line is the flight path, while the green dots represents the objects.

3.4.1 Five Stationary Objects

In the first field test, the Penguin-B UAV platform was used in order to track five simulated stationary objects. The objects were positioned as illustrated by the green dots in Fig. 3.4, and the system setup was as described in Section 2.1. An off-board MPC was used in order to do the most computationally demanding calculations, which implied sending the controller input calculated by the MPC from the ground station to the autopilot on-board. From Fig. 3.4 it is readily seen that once the off-board MPC is activated, the UAV starts to fly around the simulated stationary objects. The UAV loiters above one object until that object has received the allotted camera time, before it decides to move on to the next object. It is also observed that the sequence in which the objects are visited is chosen by selecting the next object to track to be the object closest to the one currently being tracked. Also note that once the UAV has 'visited' all five objects, it moves on to track the first object, hence starting the sequence over again. Furthermore, looking closely at the flight path, it can be observed that the UAV is spending more time tracking the first two objects than the last three. This is because the operator modified the camera time required on each object before moving on to the next using the ground station MPC GUI. Being able to adjust the system parameters in-

flight is very useful, as the operator can make adjustments to the UAV's tracking behaviour based on current event information and tracking needs. The estimated wind velocity was in the range of $1 - 5$ m/s throughout the test flight.

3.4.2 One Moving Object

The second field test was conducted using the Skywalker X8 UAV platform. During this test, a single moving object was simulated, and the UAV was flying at an altitude of 200 m. The speed of the object was set to be 1 m/s, and the direction of the speed was set to $65°$ South-West.

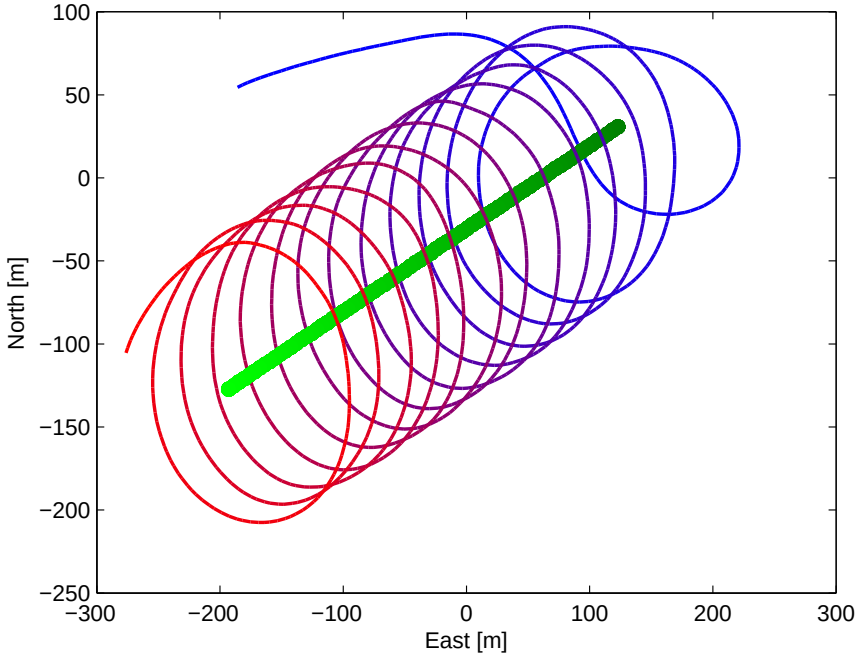

Figure 3.5: Flight test using the Skywalker X8 UAV platform and simulating one moving object. The tracking was successful, as the UAV is flying in circles around the object, slowly adjusting for the object's change in position. Blue represents the UAV's position at $t = 0$ and red represents the UAV's position at $t = end$. Dark green represents the object's position at $t = 0$, and the light green represents the object's position at $t = end$

From the field test it is seen that the MPC path planner makes the UAV loiters above the object, slowly adjusting for the object's change in position. From Fig.

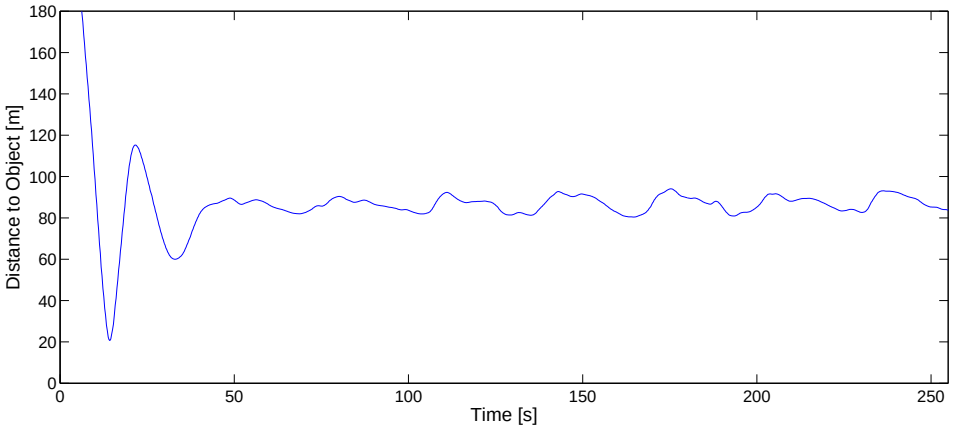

Figure 3.6: The distance from the UAV to the tracked moving object in the second field test.

3.6 the UAV's distance to the tracked object during the tracking sequence is also shown. The ideal distance was chosen to be 80 m, and it is observed that although the distance to the object is varying to some degree, the average distance to the target is ~ 80 m. It was observed that the UAV's distance to the object is varying between $75 - 90$ m. The oscillations around 80 m can be explained partly by disturbances such as wind (measured to be ~ 4 m/s), but also the fact that the motion model for the UAV used by the MPC path planner does not necessarily fit perfectly with the behaviour of the autopilot, forcing the MPC path planner to adjust its course when the UAV begins to drift closer to or further away from the tracked object. Also, the ideal distance of 80 m is not a strict requirement, and is implemented in the MPC path planner as a soft constraint.

In Fig. 3.7 the result of the gimbal control is illustrated. This is done by using the on-board telemetry data (the UAV's GPS position, altitude, attitude etc.) to calculate the distance of the camera's principal point projected onto the ground plane to the tracked object. As seen from Fig. 3.7, this distance sometimes reach ~ 50 m. However, most of the time (94% of duration of the test flight) the tracked object is within the camera's image frame. The reason for the distance between the camera's principal point projected onto the ground plane and the tracked object not being equal to 0 m is two-fold. First, the MPC path planner calculates the gimbal control input based on the position and attitude that the UAV has at the beginning of the optimization step. Since the optimization is not instant, the UAV has already changed its position and attitude when the optimization is completed, hence the

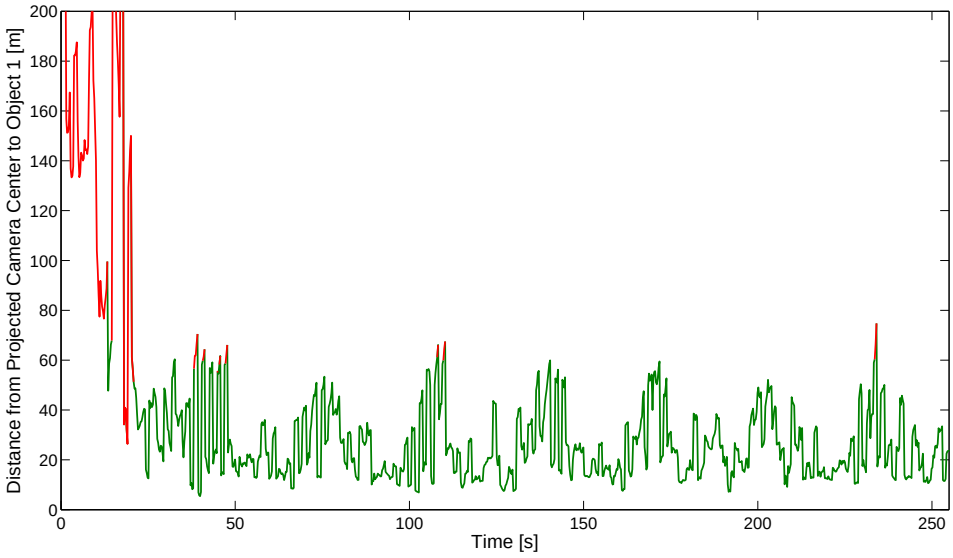

Figure 3.7: The distance from the position of the tracked object to the camera's center point projected onto the ground plane during the second field test. A green line implies that the tracked object is visible in the image frame, and a red line means that the object is outside of the image frame. It can be seen that in an initial period where the UAV is trying to assume the loitering pattern, the object is not visible in the image frame. However, after the UAV settles into a loiter pattern, the object is consistently captured in the image frame by the MPC module.

optimal gimbal set point is no longer optimal. Second, when the gimbal control input is calculated and set by the MPC path planner, it stays constant while the next control input is calculated by the path planner even though the UAV's position and attitude is changing. This causes the projection of the camera's principal point to follow the movement of the UAV. A method to improve on this behaviour would be to let the MPC path planner output a point on the ground that the camera's principal point should be projected onto, and let an internal gimbal controller stabilize the gimbal. The internal controller could utilize the on-board telemetry data to constantly adjust the gimbal pan and tilt angles in such a way that the camera's principal point is projected exactly onto the tracked object's estimated position.

3.4.3 Two Moving Objects

The third field test was also conducted using the Skywalker X8 UAV platform. However, during this test, two moving objects were simulated. The speed of both

objects were set to be 1 m/s; but one was set to move as in the second field test (65° South-West), while the other was set to move 65° North-West. The resulting tracking sequence is illustrated in Fig. 3.8.

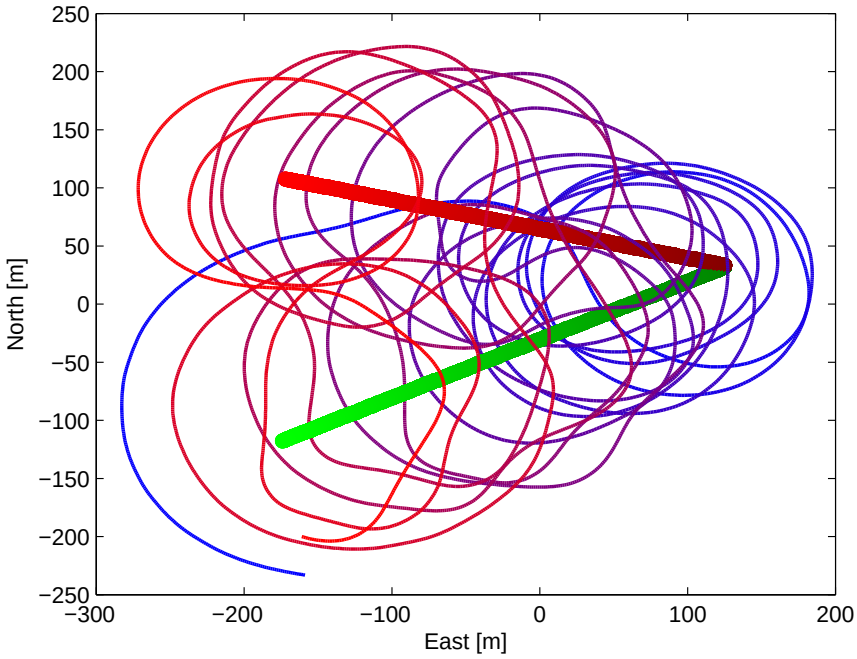

Figure 3.8: Flight test using the Skywalker X8 UAV platform and simulating two moving objects. The tracking was successful, as the UAV starts out flying in circles around both objects, slowly adjusting for the objects' change in position. When the distance between the two objects reaches a limit of 50 m, the system decides to focus on each object individually while tracking. Every 40 seconds, the path planner switch between which object it is currently tracking. Blue represents $t = 0$, and red represents $t = end$.

Since the objects were initially set to be at the same position, the Object Handler module told the MPC module to regard these two objects as one group, and try to track both at the same time. It was observed that the UAV started out circling around both objects. At this point, the MPC path planner is controlling the gimbal in such a way that both object's are to be within the camera's image frame. However, the interesting part is that once the objects were more than 50 m apart from each other (a configurable parameter), the Object Handler decides that the objects

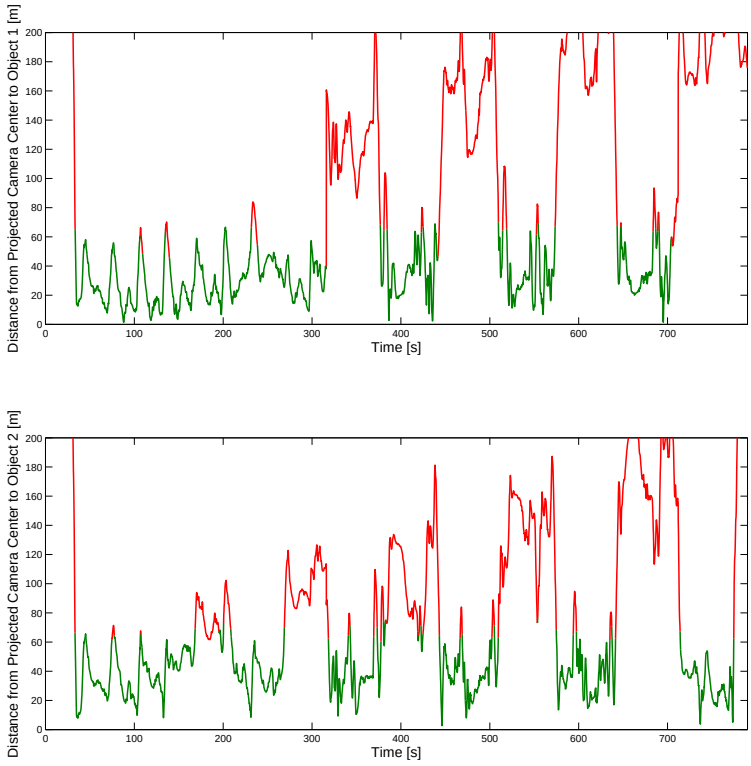

Figure 3.9: The distance from the position of the tracked objects to the camera's principal point projected onto the ground plane during the second field test. A green line implies that the tracked object is visible in the image frame, and a red line means that the object is outside of the image frame. It can be seen that in the initial tracking phase (when the objects are within 50 m of each other, the path planner is tracking both objects simultaneously, keeping both objects within the camera's image frame. However, as the object's move further apart, the MPC path planner begins to point the camera at only the object it is currently tracking. This can be seen by the fact that when one object is outside of the image frame (red line), the other object is typically within the image frame (green line).

should be tracked individually. Hence, at this moment, the Object Handler tells the MPC to switch from tracking both objects at the same time, to focusing on only one of them. The Object Handler then chose the one object of the two that the UAV is currently closest to, and continues to track this object. Once this object get

the required amount of camera time (40 seconds), the UAV will switch object and revisit the object moving in the different direction.

The results of the gimbal control are shown in Fig. 3.9. Here we see that the MPC path planner was controlling the UAV and the gimbal in such a way that at least one of the objects that were being tracked was inside the camera's image frame for 88% of the duration of the flight test. Considering that a majority of the remaining 12% of the flight time was due to limitations on the gimbal control making it infeasible to have the objects within the camera's field of view, this gives each object a large amount of camera time. Furthermore, the object illustrated with a green color in Fig. 3.9 was in the camera's image frame for 56% of the time, and the object illustrated with a red color was in the camera's image frame for 55% of the time. This also illustrates the path planner's ability to distribute the time resource correctly.

3.5 Conclusions

In this chapter an object tracking system framework has been developed. The framework can be implemented on any UAV platform, with the only requirement being that the UAV uses an autopilot that allows for external control of the aircraft. The framework also includes the possibility of having an object detection, tracking and recognition module, which typically uses an on-board camera mounted in a two-axes gimbal to detect new objects and/or verify the position of already known objects. Once implemented, the total system is easily adaptable to new or alternative UAV platforms as the system is highly modularized. The framework includes a detailed description of a highly (in-flight) configurable MPC, which is designed to follow and track any number of objects of interest. The MPC gives an intuitive way to tune the UAV's tracking behaviour. Having intuitive control of a complex object tracking system is important, as different tracking scenarios will require different tracking behaviours. E.g, if the UAV is tracking hundreds of people swimming in the water, the camera time per object that the operator wants may be in the order of only a couple of seconds. On the other hand, if the UAV is doing surveillance of an oil platform and some vessels floating next to it, the operator may want to have a camera time per object in the order of minutes. These are two object and tracking scenarios which require two different behaviours of the aircraft, both of which can be enabled by the presented framework using the correct configuration.

The presented object tracking framework was tested in three different scenarios using two different UAV platforms. All field tests were conducted using simulated

object positions and velocities, effectively assessing the MPC's tracking performance unaffected by the performance of the computer vision module. All of the conducted field tests resulted in successful object tracking by the UAV. The MPC managed to control the UAV to keep a configurable distance to the target, with only some small oscillations around the set distance. Furthermore, it was observed that since the MPC control loop is much slower than the aircraft dynamics, the gimbal system could benefit from having an internal controller stabilizing the gimbal in-between the MPC set-points. More specifically, the MPC could calculate a point on the ground that should be the center of the image, based on the gimbal control input resulting from the optimization step. The internal gimbal controller is then responsible for having the gimbal point towards this ground location until the next control input is calculated by the MPC. This would result in a more stable image stream, and give the operator an easier job at observing the tracked objects.

CHAPTER 4

Object Detection, Recognition and Tracking

This chapter is based on [60] (F. S. Leira, T. I. Fossen, and T. A. Johansen. Object detection, recognition and tracking from UAVs using a thermal camera. *Journal of Field Robotics*, 2017, *Submitted*).

4.1 Introduction

In the multiple object detection and tracking scenario described in the previous chapter, a way to further increase the automation of the overall UAV object detection and tracking framework is to make the UAV payload able to automatically detect and track multiple objects during flight using machine vision algorithms. It further improve the UAV's automation if the UAV payload is able to perform object recognition in order to distinguish between the UAV re-visiting an already detected object and the UAV detecting a novel object.

The process of automatically detect and track objects of interest using machine vision algorithms is a well known problem, and many different approaches to this are found in the literature. This problem is not restricted to UAVs as there exists many different remote sensing platforms. However, finding or adapting solutions that can be applied for thermal cameras in UAVs is still useful and necessary since e.g radar-based multi-object tracking is not directly applicable. The next section covers the recent efforts in the field of (multi-)object detection and tracking using machine vision in UAVs.

4.1.1 Related Work

The problem of automated multi-object detection and tracking using a UAV with an on-board computer and camera can be separated into three different sub-problems. First, in order for the process to be automated, robust and reliable image processing techniques for automatic object detection needs to be developed and implemented. Second, filters or estimators are needed in order to estimate the position (and possibly the velocity) of the detected objects in a given coordinate frame. Note that in some cases the detection and tracking module can be implemented as one and the same, as an algorithm might do these two steps simultaneously. Third, there is the problem of data association, i.e how to associate new measurements of objects' position with objects already being tracked. This section will cover some of the recent work on these three different sub-problems.

The problem of object detection is the problem of having a computer automatically segment an image into object and non-object regions. That is, using a machine algorithm, the computer should be able to say which regions of an image that contains objects of interest. This problem is well studied, and several very different approaches are described in the literature. In [83] a template matching with zero mean normalized cross correlation is used to identify objects of interest in an image. This approach requires that the machine vision algorithm is supplied with an image (template) of the object of interest prior to searching novel images for said object. This is a fast and reliable method if the appearance of the object or type of objects of interest is not changing over time and/or changing with the angel the object is viewed from. However, if the object can take on different appearances, and is non-symmetric, the number of templates needed to perform the object detection increases quickly. The computation time needed to detect objects using this approach is directly linear to the number of different templates used for an object. Furthermore, in some scenarios this approach is not viable due to the fact that the appearance of the object is not exactly known prior to detecting the object. [101] use a combination of color segmentation (filtering an image based on color) with an Continuously Adaptive Meanshift (CAMshift) algorithm. The CAMshift algorithm was first introduced in [26], and is an algorithm that adaptively finds the position and orientation of an object of interest in an image. However, the algorithm requires that the image already is segmented into object and non-object regions, hence the object detection step itself in [101] is completely dependent on the accuracy of the color segmentation step. Segmenting an image based on color and/or intensity is suitable for many applications, but often requires manual and individual tuning for specific scenarios and scenery. Furthermore, this technique will

easily generate false positives (segmenting parts of the image as an object although there is no object to be found) because it treats everything in a scene above a certain threshold as an object of interest. Some variation of the color segmentation technique for object detection in UAVs can also be found in [93] and [92]. These approaches handles false positives by having a robust object tracking algorithm that will filter out the false positives. This could for instance be done by observing that the detections of actual objects are persistent, whereas detections of false positives often are not. Another approach to object detection is based on optical flow in a sequence of images, and examples where this is utilized can be found in [88] and [49]. This is an effective but often computationally heavy approach, and is mainly used for detection of moving (i.e not stationary) objects.

The detection and tracking framework presented in [56], Tracking-Learning-Detection (TLD), is an open-source framework which, given a region containing an object in an initial image frame, detects the location of said object throughout the duration of the image sequence (for as long as the object is within the image frame). This is done by decomposing the long-term tracking process into three sub-problems: tracking, learning and detection. That is, the tracker follows the object from frame to frame. The detector localizes all appearances that have been observed so far and corrects the tracker if necessary [56], and the learning module estimates the detector's errors and updates it to avoid these errors in the future. The major weakness of the TLD framework is that the tracking results are greatly affected by the initially chosen region for the object to track, hence care has to be taken as to how this region is (automatically) chosen. [77] presents a successful implementation of the TLD framework on a small multi-rotor UAV, but their system requires the operator to manually select an object to track and follow.

For the object tracking problem, several solutions for object tracking using UAVs have been proposed. [20] use a ground station computer for image processing and object detection in combination with a recursive least square filter in order to estimate an object's position. The filter converges in ≈ 40 measurements, which is $\approx 5m$ from the object's actual position. In addition to estimating an object's position, they simultaneously estimate the bias offset angles of the mounting of the camera gimbal. This is done by flying in circles around the object while solving a minimization problem. However, the system is assuming that there is only one single object to track, and that said object is stationary in the duration of the flight. That is, the object's velocity is not estimated.

[30] proposes a solution for single object tracking using UAVs and off-board im-

age processing. A non-linear parametrically varying (NLPV) filter is applied to estimate both the object's position and velocity. The NLPV filter combine the advantages of a parametrically varying model structure and the NARMAX (nonlinear autoregressive exogenous moving average) algorithm. An accuracy of 5 m in the positional estimate, and 0.1 m/s in the velocity estimate is reported. Although the proposed solution is able to track a moving object, a method to expand this system to the case of multiple object tracking is not presented.

[81] performs object tracking by using an extended Kalman filter (EKF) to estimate an objects position, velocity and heading based on measurements of the object's position. These estimates are then used to calculate an optimal predicted object movement for a short time horizon into the future. This is an effective approach in order to enable the UAV to plan its future movement, however, the work presented in [81] is tested in simulation. Furthermore, it is only tested in the case where there exists continuous measurements of the tracked object's position throughout the whole tracking process.

[49] uses a Kalman filter and a linear motion model to estimate an object's position and velocity based on measurements of both the object's position and velocity (acquired using optical flow). Initial results show that the estimated position of an object is within 15 m of it's actual position. However, the object's velocity estimate is prone to noise in the UAV's attitude measurements.

It should be noted that the performance of an object tracking algorithm is not only measured by the distance between the ground truth and the estimated position, but that this is a convenient performance metric when not comparing the specifics of each tracking method. See e.g [51] for two more examples of performance metrics in object tracking systems.

When multiple objects are tracked simultaneously, the problem of data association arises. This is a problem where object detections has to be associated with objects already being tracked, or as a novel object entering the image frame. [71] and [72] proposes a multiple object tracking framework which performs tracking and data association simultaneously based on the Random Sample Consensus (RANSAC) algorithm. RANSAC is a powerful algorithm used in many computer vision problems [71], and is based on the concept that a dataset contains so-called "inliers" and "outliers". Inliers is data points which can be explained by some set of model parameters. The outliers are data points that does not fit with the set of model parameters, hence should ideally not be used when trying to estimate the model parameters. Typically algorithms would estimate the model parameters us-

ing the total data set, RANSAC however, assumes that given a small set of inlier data points, there exists a procedure which can estimate the parameters of a model that optimally explains or fits the data. [71] and [72] presents how this can be a powerful tool when performing multiple object tracking of an unknown number of dynamic objects. The strength of the approach is in its ability to handle a huge amount of false positives and false negatives from the object detection module. However, the weakness of the system is that knowing exactly how many objects currently is being tracked is difficult, as one object may have several plausible tracks, and as a consequence, several possible model parameters.

In [106], the traditional "detection-tracking" approach is replaced by stating the object detection, tracking and data association problem as a single objective function. The main concept of the approach is to have a robust object detector mark separate blobs in each image frame in a video sequence, and then stating the data association as a network flow problem where the goal is to optimize the flow going through the network. The strength of this approach is that the whole dataset is considered simultaneously when solving the data association and tracking process. However, this is also its weakness as the computational burden becomes large for longer video sequences. Furthermore, the approach would have to be modified to deal with false positives in the object detection algorithm.

The nearest neighbor (NN) method is regarded as one of the most straight forward approaches to data association [58]. Given several possible detections for the position of an object, the associated detection is assumed to be the detection "closest" to the estimated or predicted position of the tracked object. The term "closest" is based on a predefined measurement of the distance to the object, e.g this could be the distance in image pixels between the object measurement and the estimated object position in the image frame. However, the NN method is prone to end up in non-optimal solutions for the data association problem when multiple objects are being tracked simultaneously. This is because the order of associating measurements with tracked objects can affect the overall result [58]. To address the shortcomings of the NN method a global nearest neighbor (GNN) algorithm can be applied [58]. The GNN algorithm seeks to find the global minimum solution in the case of multiple object tracking, effectively avoiding the non-optimal solution which the NN method can possible generate. This solution is optimal in the way that the total distance between the measured object positions and the estimated position of the tracked objects are minimized. The GNN algorithm is a more computationally heavy approach, but is otherwise very similar to the NN algorithm.

4.1.2 Contribution

In this chapter, the focus is on the development and implementation of an automatic object detection and tracking module intended to be used in the general UAV object detection and tracking framework described in this thesis. The main concept is that an object detection, recognition and tracking module detects, recognizes and tracks the position and velocity of objects of interest, which is supplied to a path controller described in the previous chapter which is calculating a flight path and gimbal control that will enable the UAV to autonomously keep the objects of interest within the on-board camera's image frame for extended periods of time. This means that the module must be able to run on-board and in real-time in order to make the UAV able to detect, recognize and track objects during flight. Moreover, the module combines methods from both object detection, tracking and data association in order to be able to simultaneously detect and track multiple stationary and moving objects.

This chapter extends the automatic object detection, recognition and tracking algorithms presented in [64] and [62] by improving the tracking module's ability to recognize objects on the UAV's re-visitation of the objects it is tracking. Furthermore, the tracking process is extended from tracking objects only in the image frame to the case where the UAV is able to track the position and velocity of objects of interest in world coordinates.

4.1.3 Overview of the Chapter

The remainder of this chapter is organized as follows. First, an MV algorithm for automatically detecting objects in real-time from a processing unit placed on-board the UAV is presented. Second, a tracking algorithm developed to track the position and velocity of the automatically detected objects is presented. This approach covers both the process of estimating an object's position and velocity based on image based detections, as well as data association, i.e, associating new detections with objects already being tracked. Section 4.4 covers the details of a field test conducted in order to test the object detection and tracking algorithms presented in this chapter, while Section 4.5 covers the results of the conducted flights. Finally, the chapter is summarized and concluded.

4.2 Object Detection and Recognition

The Object Detection, Tracking and Recognition module described in the previous section use images from an on-board camera to automatically do segmentation

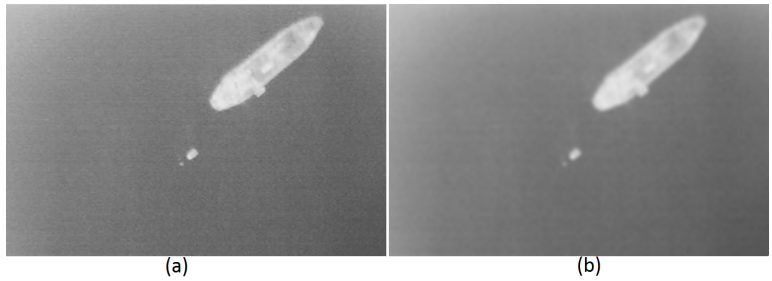

Figure 4.1: Before (a) and after (b) smoothing of the original image. The image is showing a large boat (length of 55m), a rigid-inflatable boat and a small buoy.

of the images. That is, using machine vision, the module's task is to segment images into foreground (object) and background. In order to automatically detect objects of interest, the thermal image, $\mathbf{I}$, is first smoothed. The motivation for this is to reduce the thermal noise present in the image. This was found to make the edge detector more robust than when performed on the sharp thermal images. The smoothing is done by convolving (denoted by $*$) the image with a Gaussian kernel $\mathbf{g}$. $\mathbf{g}$ is a $n \times n$ kernel approximating a Gaussian distribution with a standard deviation of σ_g, i.e

$$\boldsymbol{I_s}[x,y] = (\boldsymbol{I} * \boldsymbol{g})[x,y] = \sum_{k=-\frac{n-1}{2}}^{k=\frac{n-1}{2}} \sum_{m=-\frac{n-1}{2}}^{m=\frac{n-1}{2}} \boldsymbol{I}[x-m, y-k]\boldsymbol{g}[m,k] \qquad (4.1)$$

$\boldsymbol{I}$ and $\boldsymbol{I_s}$ are $w \times h$ matrices, where w and h is the width and height of the original thermal image. $[x, y]$ are integers representing a pixel coordinate in images $\boldsymbol{I}$ and $\boldsymbol{I_s}$. $[m, k]$ are integers representing a coordinate in the Gaussian kernel approximation $\boldsymbol{g}$. The result of smoothing an image showing a big boat (length of 56m), a rigid-hulled inflatable boat (RHIB) and a small buoy can be seen in Figure 4.1. Notice that the upper left corner has slightly brighter pixels than the rest of the image, due to inaccurate camera calibration. Now, to detect the edges in the resulting smoothed image $\boldsymbol{I}_s$, the gradient image, $\boldsymbol{G}$, of $\mathbf{I}_s$ is calculated. The gradient image of $\boldsymbol{I}_s$ is found by the following calculation

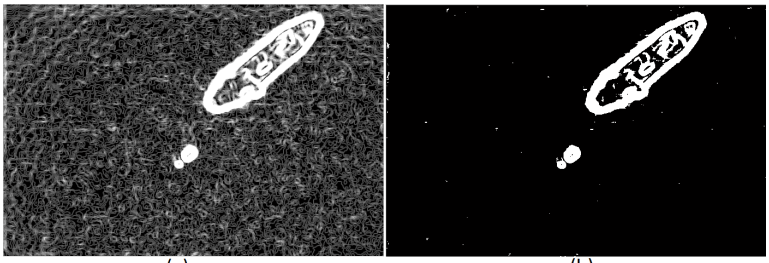

Figure 4.2: Before (a) and after (b) gradient magnitude thresholding.

$$
\begin{aligned}
\boldsymbol{G}_{\boldsymbol{I}x}[x,y] &= (\boldsymbol{I}_s * \boldsymbol{P})[x,y] = \sum_{k=-1}^{k=1}\sum_{m=-1}^{m=1} \boldsymbol{I}_s[x-m, y-k]\boldsymbol{P}[m,k], \\
\boldsymbol{G}_{\boldsymbol{I}y}[x,y] &= (\boldsymbol{I}_s * \boldsymbol{P}^T)[x,y] = \sum_{k=-1}^{k=1}\sum_{m=-1}^{m=1} \boldsymbol{I}_s[x-m, y-k]\boldsymbol{P}^T[m,k], \\
\boldsymbol{G}_{\boldsymbol{I}}[x,y] &= \sqrt{\boldsymbol{G}_{\boldsymbol{I}x}^2[x,y] + \boldsymbol{G}_{\boldsymbol{I}y}^2[x,y]}
\end{aligned}
\tag{4.2}
$$

$\mathbf{P}$, also referred to as the Prewitt operator [97], is defined as the 3×3 matrix

$$
\mathbf{P} := \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.3}
$$

The resulting gradient image $\mathbf{G_I}$ can be seen in Figure 4.2a. It is seen that the big boat, the RHIB and the small buoy is clearly visible after these image processing operations. However, it is apparent that the waves and ripples in the ocean in addition to some of the noise in the image are still visible, albeit smaller in magnitude than the objects of interest in the image. Because of this, removing them can be done by using a threshold value for the magnitude of the gradients. That is, all pixels in the gradient image that have a magnitude less than a certain threshold T_g can be removed. This is achieved by the following operation

$$
\boldsymbol{G}_{\boldsymbol{I}}(x,y) = \begin{cases} maxValue & \text{if } \boldsymbol{G}_{\boldsymbol{I}}(x,y) \geq T_g \\ 0, & \text{otherwise} \end{cases} \tag{4.4}
$$

Where $maxValue$ is the maximum brightness value a pixel in image $\mathbf{G_I}$ can have. From Figure 4.2b it is readily seen that, post processing, it is mostly objects with a distinct heat signature that are left in the image.

Figure 4.3: Before (a) and after (b) removing blobs which are either too small or too large to be an object of interest.

Looking at Figure 4.3a, it is obvious that some of the blobs clearly do not originate from any object of interest (i.e, the small dots scattered across the image in Figure 4.3a), and therefore have to be filtered out. To filter out the unwanted blobs from the image, a connected component algorithm [100] is used to group and label components together in blobs. Furthermore, the area of each blob is then calculated, and blobs with a smaller or larger area than what is expected from a blob originating from an object of interest are then removed from the image. The result of this process is seen in Figure 4.3. The resulting image (Figure 4.3b) is hereby referred to as the binary image, $\mathbf{B}$, of the original image $\mathbf{I}$.

After applying the image analysis methods just described and finding the bounding boxes for each detected object, the detected objects can be seen in Figure 4.4a. However, it is seen that big objects which have some texture in them can trigger detections within the interior of the actual object. This is because the texture of the object shows up in the edge detector. In order to make every detection of an object only show up as one unique detection, bounding boxes completely contained in a larger bounding box are removed. The result of this process is seen in Figure 4.4b.

Looking at Figure 4.4b, it is apparent that the three detected objects are of further interest. The detected objects are now ready for classification. The center positions of the remaining blobs are calculated in both the image frame and in the world frame, and then passed on to the tracking module as measurements.

The detection step provides the position of objects of interest in the image. However, since the detector is using edge detection, the areas of an image that is highlighted as interesting will often only contain the exterior edges of an object. When performing for instance recognition based on characteristics such as size, average

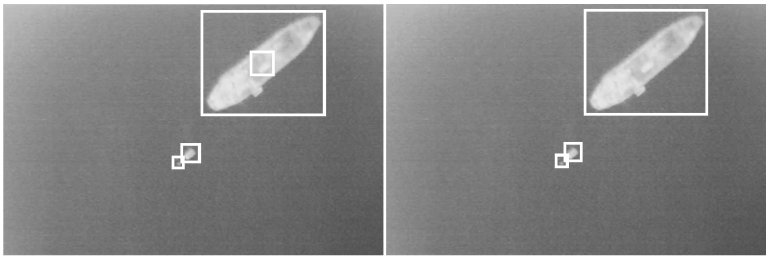

Figure 4.4: Before (a) and after (b) removing detections completely contained in the interior of other detections.

temperature and overall form it is crucial that the whole object is evaluated. To expand the detections to also include the interior of the objects of interest, an algorithm that seeks to fill holes in the binary image [97] shown in Figure 4.4b is applied. The result of applying this algorithm can be seen in Figure 4.5.

Figure 4.5: Before (a) and after (b) filling the interior holes in the detected objects.

Using the location of the bright pixels in the binary image seen in Figure 4.5b, the pixels that make out the object in the original image (Figure 4.1a) can be analysed. The image with filled contours is hereby denoted $\mathbf{B}_{filled}$.

In this thesis, the object characteristics used for recognition are the observed object area, perceived average object temperature and one of the scale, rotation and translation invariant moments proposed by Hu [54]. Using scale, rotation and translation invariant features when describing objects observed from the air is very important, as the altitude, angle and orientation that the object is viewed from is constantly changing. Note that both the observed object area and the perceived average object temperature will be, to some extent, invariant to the scale, rotation and translation of the object. Furthermore, keep in mind that the recognition

method presented can be used for a large variety of other object characteristics and be modified to include other object features.

The invariant moments presented in [54] is based on the following moment function

$$m_{pq} = \sum_{x,y \in \boldsymbol{O}} x^p y^q \boldsymbol{B}(x,y), \quad p,q = 0,1,... \tag{4.5}$$

where m_{pq} is referred to the $(p+q)$th order moment of the image region $\boldsymbol{O}$. Here, $\boldsymbol{O}$ is defined as the set of pixels inside the object's bounding box. $\boldsymbol{B}$ is here a binary image. Note that since $\boldsymbol{B}$ is a binary image, the 0th moment (m_{00}) simply becomes the number of positive pixels in the image region. This in turn can be interpreted as the pixel area of the object. This is an effective parameter to use when classifying objects, especially when pixel area is converted to the metric area of the object through the use of the on-board altimeter's and attitude and heading reference system's (AHRS) measurements. The metric area of an object can give good estimates of the object's type and inertia. Assuming that the UAV is flying approximately straight forward (low roll and pitch values), the metric area of an object can be found by multiplying the pixel area of the object (m_{00}) with a factor $a(h)$. $a(h)$ is defined as square meters per pixel when the camera is at altitude h. This factor is given by the thermal camera lens and characteristics. Hence, the metric area of an object can be approximated by

$$A \approx a(h) m_{00} \tag{4.6}$$

Central moments are also used in the calculation of Hu's invariant moments, and are given as

$$\begin{gathered} \mu_{pq} = \sum_{x,y \in \boldsymbol{O}} (x-\bar{x})^p (y-\bar{y})^q \boldsymbol{B}(x,y), \quad p,q = 0,1,... \\ \bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}} \end{gathered} \tag{4.7}$$

Where $(\bar{x}, \bar{y})$ is the centroid of the image region. $\boldsymbol{O}$ is still defined as in (4.5). Note that the central moments are invariant to translation. This is readily seen by observing that the central moment is just m_{pq} shifted to the centroid of the image region. Now, to get scale invariant moments, the normalized central moments are introduced. The normalized central moments are given as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}, \quad \gamma = (p+q+2)/2, \quad p+q = 2,3,... \tag{4.8}$$

Using the normalized central moments, Hu introduced seven moments invariant to rotation, translation and scale. However, research has shown that for objects represented with a low pixel amount of pixels (less than 100×100 pixels), these moments may vary when the image is scaled and/or rotated. Furthermore, [37] show that the higher order moments (ϕ_{2-7}) vary much more than the lower order moment (ϕ_1) for images with small resolution. Since the resolution of most thermal imaging cameras are still quite poor, objects of interest will not be represented by a lot of pixels. Because of this, only the first invariant Hu moment is included in the feature vector.

$$\phi_1 = \eta_{20} + \eta_{02} \tag{4.9}$$

To calculate the perceived average object temperature, the settings of the thermal imaging camera is utilized. That is, the camera can be set to capture temperatures in a range from a minimum and a maximum temperature (T_{min} and T_{max}). Furthermore, assuming that the output from the camera is an image where each pixels is represented by one byte, each pixel can take on a value between 0 and $2^8 - 1 = 255$. This means that the perceived temperature of an area covering only 1 pixel will be

$$\hat{T} = \frac{I(x,y)}{255}(T_{max} - T_{min}) + T_{min} \tag{4.10}$$

Expanding this to calculate the average perceived temperature across a detected object, we get

$$T_{avg} = \frac{\frac{\sum_{x,y \in \boldsymbol{O_p}} I(x,y)}{m_{00}}}{255}(T_{max} - T_{min}) + T_{min} \tag{4.11}$$

Where

$$\boldsymbol{O_p} = \{x, y \in \boldsymbol{O} | \boldsymbol{B_{filled}}(x,y) = 1\}$$

$\boldsymbol{O}$ is the set of pixels in the object's bounding box. Hence, the image region $\boldsymbol{O_p}$ is given by the set of pixels in the detected object blob in the binary image $\mathbf{B}_{filled}$. Note that the perceived temperature is found from pixel intensities in the original image $\mathbf{I}$.

Combining the perceived average object temperature with the invariant moments, we can represent any group of pixels in an image using the feature vector $\mathbf{X}$

$$\mathbf{X} = \begin{bmatrix} A \\ T_{avg} \\ \phi_1 \end{bmatrix} \tag{4.12}$$

In the case that a new object appears in the image frame and is not matched to any of the objects currently being tracked, the object's feature vector $\mathbf{X}$ is calculated for the first 5 images where the whole object is visible. The average of these 5 feature vectors, $\bar{\mathbf{X}}$ is then stored on-board the UAV for future reference, and used when detected objects are associated with objects already being tracked. This process is described in Section 4.3.2.

4.3 Object Tracking

The object tracking module is responsible for estimating and keeping track of the position and velocity of the detected objects. This is done by using Kalman filters to estimate and predict the position and velocity for each object. That is, for each uniquely detected object, a Kalman filter instance is created. Further detections of the tracked object is then used as measurements in the Kalman filter in order to estimate the object's position and velocity. This means that the tracking module also has to be able to associate new detections with already existing tracked objects. This is done by associating object detections to the most likely among the tracking gaits. A tracking gait is defined as the complete state history of an object, i.e the history of its positions and velocities. If an object detection is not likely to originate from any of the objects currently being tracked, the tracking module assumes that a novel object has been detected, and creates a new Kalman filter instance associated with this object.

4.3.1 Discrete-Time Kalman Filter

The Kalman filter implemented in the detection, recognition and tracking module is based on [16], and utilizes a motion model based on the assumption that the UAV is tracking objects located on a flat surface (e.g the sea surface) and moving with a constant velocity. This yields the following linear equations of motion:

$$\begin{aligned} x_{k+1}^{obj} &= x_k^{obj} + \Delta t V_{x,k}^{obj} + \frac{1}{2}\Delta t^2 w_k \\ y_{k+1}^{obj} &= y_k^{obj} + \Delta t V_{y,k}^{obj} + \frac{1}{2}\Delta t^2 z_k \\ V_{x,k+1}^{obj} &= V_{x,k}^{obj} + \Delta t w_k \\ V_{y,k+1}^{obj} &= V_{y,k}^{obj} + \Delta t z_k \end{aligned} \tag{4.13}$$

where x_k^{obj}, y_k^{obj}, $V_{x,k}^{obj}$ and $V_{y,k}^{obj}$ is the position and linear velocity of an object in North-East coordinates (x is North and y is East) at time step k and Δt is the time passed from time step k to $k+1$. w_k and z_k is Gaussian white noise representing

change in velocity of an object. This yields the following model in state space form

$$\begin{aligned} \mathbf{x}_{k+1}^{obj} &= \boldsymbol{F}\mathbf{x}_{k}^{obj} + \boldsymbol{E}\mathbf{w_k} \\ \mathbf{y}_{k}^{obj} &= \boldsymbol{C}\mathbf{x}_{k}^{obj} + \mathbf{v}_k \end{aligned} \tag{4.14}$$

The matrices $\mathbf{F}$, $\mathbf{E}$ and $\mathbf{C}$ are

$$\begin{aligned} \boldsymbol{F} &= \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{E} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \\ \boldsymbol{C} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{aligned} \tag{4.15}$$

and we have that

$$\mathbf{w_k} = \begin{bmatrix} w_k \\ z_k \end{bmatrix} \quad \mathbf{v_k} = \begin{bmatrix} q_k \\ r_k \end{bmatrix} \tag{4.16}$$

w_k and z_k are here as previously defined, while q_k and r_k are Gaussian white noise terms which represent noise and errors in the measurement of an object's position.

For the two dimensional motion described in (4.13), the state vector $\mathbf{x}_k^{obj}$ and the measurement $\mathbf{y}_k^{obj}$ are equal to

$$\mathbf{x}_k^{obj} = \begin{bmatrix} x_k^{obj} \\ y_k^{obj} \\ v_x^{obj} \\ v_y^{obj} \end{bmatrix} \quad \mathbf{y}_k^{obj} = \begin{bmatrix} y_{x,k}^{obj} \\ y_{y,k}^{obj} \end{bmatrix} \tag{4.17}$$

where $y_{x,k}^{obj}, y_{y,k}^{obj}$ is the detected object's position at time step k. Note that $y_{x,k}^{obj}, y_{y,k}^{obj}$ is the measured position of the object in North-East (NE) coordinates, while the detection step described in Section 4.2 only yields the centroid of an object in image frame coordinates (pixel location of the centroid). Hence, in order to find the North-East coordinates that this pixel location corresponds to we need to project the pixel location onto the North-East plane. The following approach to this problem is based on the work found in [107] and [50].

Now, before we are able to compute anything in a North-East-Down (NED) frame, a reference point for the NED frame has to be set using geographic (latitude and longitude) coordinates. To define a NED frame at a reference geographic position,

λ_{ref} (longitude), ϕ_{ref} (latitude) and h_{ref} (height), we first find the Earth-Centered Earth-Fixed (ECEF) coordinates of the position using the following equations [39]

$$\begin{aligned} X^e &= (N+h)\cos(\phi_{ref})\cos(\lambda_{ref}) \\ Y^e &= (N+h)\cos(\phi_{ref})\sin(\lambda_{ref}) \\ Z^e &= \left[N(1-e^2)+h_{ref}\right]\sin(\phi_{ref}) \\ N &= \frac{a}{\sqrt{1-e^2\sin^2(\phi_{ref})}} \\ f &= \frac{(a-b)}{a} \\ e^2 &= 2f - f^2 \end{aligned} \tag{4.18}$$

(X^e, Y^e, Z^e) is a position in the ECEF frame, a is the semi-major earth axis (6378137 m) and b is the semi-minor earth axis (635752.3142 m). These values for a and b are coherent with the ellipsoid model used for the Earth by modern GPS devices [70], named World Geodetic System 84 (WGS84). We then apply the following rotation to the ECEF coordinates in order to align it with the North, East and Down axis [39]

$$\boldsymbol{R}_{ecef}^{ned} = \begin{bmatrix} -\mathrm{s}\lambda_{ref}\mathrm{c}\phi_{ref} & -\mathrm{s}\lambda_{ref}\mathrm{s}\phi_{ref} & \mathrm{c}\lambda_{ref} \\ -\mathrm{s}\phi_{ref} & \mathrm{c}\phi_{ref} & 0 \\ -\mathrm{c}\lambda_{ref}\mathrm{c}\phi_{ref} & -\mathrm{c}\lambda\mathrm{s}\phi_{ref} & -\mathrm{s}\lambda_{ref} \end{bmatrix} \tag{4.19}$$

where *s* and *c* are the trigonometric functions $\sin$ and $\cos$.

Now, to find the georeferenced location of an object (in our case the object's location in the NED frame), the pixel coordinates of the object's center in the image is calculated. Using these coordinates, combined with the UAV's attitude and altitude data and the gimbal camera orientation, the location of the object in NED frame coordinates can be related to the image coordinates as follows [107]

$$\begin{bmatrix} p'_u \\ p'_v \\ w_s \end{bmatrix} = \boldsymbol{AG} \begin{bmatrix} N_{obj} \\ E_{obj} \\ D_{obj} \\ 1 \end{bmatrix} \tag{4.20}$$

where

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{R}_{ned}^{camera} & -\boldsymbol{R}_{ned}^{camera}\boldsymbol{C} \end{bmatrix} \tag{4.21}$$

Here, w_s is a scaling factor; p'_u and p'_v define the scaled object location in the image frame (pixel coordinates) and N_{obj}, E_{obj} and D_{obj} are the NED frame coordinates

of the object. In order to find the non-scaled (actual) pixel coordinates (p_u, p_v) we calculate the normalized homogeneous form of the vector found in (4.20), i.e

$$\begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{p'_u}{w_s} \\ \frac{p'_v}{w_s} \\ \frac{w_s}{w_s} \end{bmatrix} \tag{4.22}$$

Furthermore, $\boldsymbol{A}$ is the intrinsic parameter matrix of the camera and $\boldsymbol{G}$ is the extrinsic orientation parameters of the camera (rotation and translation). $\boldsymbol{C}_{cam,ned}$ is the position of the camera given in the NED frame [50], i.e

$$\mathbf{C}_{cam,ned} = \begin{bmatrix} N_{uav} \\ E_{uav} \\ D_{uav} \end{bmatrix} + \boldsymbol{R}^{ned}_{body} \begin{bmatrix} X_{GB} \\ Y_{GB} \\ Z_{GB} \end{bmatrix} \tag{4.23}$$

where $N_{uav}, E_{uav}, D_{uav}$ define the UAV's position in the NED frame, and X_{GB}, Y_{GB}, Z_{GB} is the gimbal position in the body frame. The relation between the coordinates systems are illustrated in Figure 4.6.

The rotation matrix $\boldsymbol{R}^{camera}_{ned}$ can easily be calculated from the following equation

$$\begin{aligned} \boldsymbol{R}^{camera}_{ned} &= (\boldsymbol{R}^{ned}_{camera})^{-1} \\ &= (\boldsymbol{R}^{ned}_{body}\boldsymbol{R}^{body}_{camera})^{-1} \\ &= \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{bmatrix} \end{aligned} \tag{4.24}$$

where $\boldsymbol{R}^{ned}_{body}$ is defined as [39]

$$\boldsymbol{R}^{ned}_{body} = \begin{bmatrix} \mathrm{c}\theta\mathrm{c}\psi & -\mathrm{c}\phi\mathrm{s}\psi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{c}\psi & \mathrm{s}\phi\mathrm{s}\psi + \mathrm{c}\phi\mathrm{s}\theta\mathrm{c}\psi \\ \mathrm{c}\theta\mathrm{s}\theta & \mathrm{c}\phi\mathrm{c}\psi + \mathrm{c}\phi\mathrm{s}\theta\mathrm{s}\psi & -\mathrm{s}\phi\mathrm{c}\psi + \mathrm{c}\phi\mathrm{s}\theta\mathrm{s}\psi \\ -\mathrm{s}\theta & \mathrm{s}\phi\mathrm{c}\theta & \mathrm{c}\phi\mathrm{c}\theta \end{bmatrix} \tag{4.25}$$

θ, ϕ and ψ is the roll, pitch and yaw, respectively, of the aircraft. Furthermore, letting 0° pan and 0° tilt be the position in which the gimbal (hence also the camera) is pointing directly downwards, the rotation matrix describing the rotation between the body frame and the mounted frame ($\boldsymbol{R}^{mount}_{body}$) is equal to a 90 degrees clockwise rotation about the z-axis. That is,

$$\boldsymbol{R}^{mount}_{body} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.26}$$

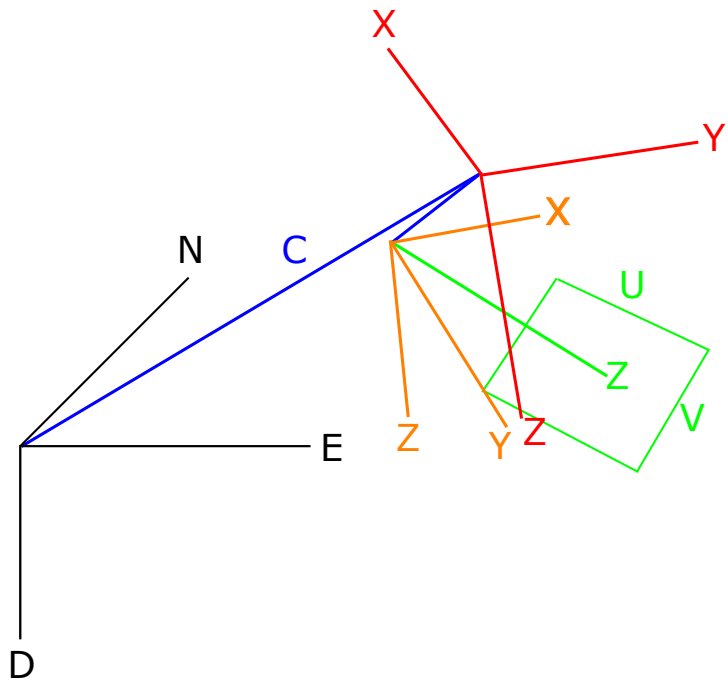

Figure 4.6: Illustration of the three different coordinate systems. The NED (black) frame, the body (red) frame of the UAV, the mount frame (orange) and the camera (green) frame. The blue line is the translation of the camera represented in the NED frame.

Now, in the mounted frame, the gimbal's pan and tilt movement corresponds to a rotation along the mounted z-axis and the mounted x-axis, respectively. Given a gimbal position ψ_{gb} (pan) and ϕ_{gb} (tilt), we have the following rotation matrix to relate the mounted frame to the camera frame.

$$\boldsymbol{R}_{mount}^{camera} = \begin{bmatrix} \mathrm{c}\psi_{gb} & \mathrm{s}\psi_{gb} & 0 \\ -\mathrm{s}\psi_{gb}\mathrm{c}\phi_{gb} & \mathrm{c}\psi_{gb}\mathrm{c}\phi_{gb} & \mathrm{s}\phi_{gb} \\ \mathrm{s}\phi_{gb}\mathrm{s}\psi_{gb} & -\mathrm{s}\phi_{gb}\mathrm{s}\psi_{gb} & \mathrm{c}\phi_{gb} \end{bmatrix} \tag{4.27}$$

Finally, in order to find $\boldsymbol{R}_{camera}^{body}$ we use the following relation

$$\begin{aligned} \boldsymbol{R}_{camera}^{body} &= (\boldsymbol{R}_{body}^{camera})^{-1} \\ &= (\boldsymbol{R}_{mount}^{camera}\boldsymbol{R}_{body}^{mount})^{-1} \end{aligned} \tag{4.28}$$

Now, we will assume, without loss of generality, that every object is located on the ground surface. This implies that D_{obj} in (4.20) will be equal to 0. This yields the

following relationship between the object's image frame coordinates (p_u, p_v) and its NED coordinates $(N_{obj}, E_{obj}, 0)$

$$w_s \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} = \boldsymbol{A}\boldsymbol{G}_{NE} \begin{bmatrix} N_{obj} \\ E_{obj} \\ 1 \end{bmatrix} \tag{4.29}$$

where

$$\boldsymbol{G}_{NE} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & -\boldsymbol{R}_{ned}^{camera}\boldsymbol{C} \end{bmatrix} \tag{4.30}$$

w_s is still a scaling factor as defined in (4.20) and (4.22). If, in general, $D_{obj} \neq 0$, the system can be adapted to use electronic maps. This would involve a search for the intersection between D_{obj} and the ray of light going from the object to the camera center. This would increase the complexity of the calculation by a small amount.

From (4.29) it is readily seen that if the pixel coordinates of an object (p_u, p_v) is known, the object's position in the NED frame can be found by the following calculation

$$\frac{1}{w_s} \begin{bmatrix} N_{obj} \\ E_{obj} \\ 1 \end{bmatrix} = \boldsymbol{G}_{NE}^{-1}\boldsymbol{A}^{-1} \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} \tag{4.31}$$

At each time step k the matrices $\boldsymbol{R}_{ned_k}^{camera}$ and $\boldsymbol{C}_{cam,ned_k}$ (hence also $\boldsymbol{G}_{NE,k}$) can be calculated using the navigation data received from the on-board autopilot, which is the output of an Extended Kalman Filter estimating the UAV's position, velocity and attitude. Note that since (4.31) will simultaneously depend on data from both the autopilot's navigation data and pixel location of a detected object from the camera's image data, the two data sources have to be time synchronized as accurately as possible.

4.3.2 Data Association

When tracking objects, one of the most crucial parts is to be able to match new object detections to objects which the system is already tracking. This problem is often referred to as data association. In the present system, a global nearest neighbour approach similar to the one found in [58] is utilized to perform data association. This involves using the following distance metric for the distance between measurement i and tracking gait j

$$D_{i,j} = (1-\gamma)\tilde{\boldsymbol{y}}_{i,j}^T \boldsymbol{S}_j(k)^{-1}\tilde{\boldsymbol{y}}_{i,j} + \gamma\tilde{\mathbf{X}}_{i,j}^T\boldsymbol{\Gamma}\tilde{\mathbf{X}}_{i,j} \tag{4.32}$$

where

$$\tilde{\boldsymbol{y}}_{i,j} = \left[\boldsymbol{y^{obj}}_i - \hat{\boldsymbol{y}}^{\boldsymbol{obj}}{}_j \right] \quad and \quad \tilde{\mathbf{X}}_{i,j} = \left[\mathbf{X}_i - \bar{\mathbf{X}}_j \right]$$

Here, $\boldsymbol{y^{obj}}_i$ is measurement i (given as a measurement of an object's position in NE coordinates), $\hat{\boldsymbol{y}}^{\boldsymbol{obj}}{}_j$ is the a priori predicted position of object j, and $\mathbf{S}_j$ is the prediction's associated covariance matrix. Both the predicted position and the covariance matrix is given by the Kalman filter estimating the position of tracking gait j. In this paragraph the time index k is dropped for simplicity of notation. γ is $\in (0, 1)$ (tunable parameter) when the whole object (all edges) is contained within the image frame and 0 otherwise. This is because the detected object's feature vector should only be calculated when the whole object is visible. $\mathbf{X}_i$ is the feature vector corresponding to measurement i, and $\mathbf{X}_j$ is the previously stored feature vector associated with object j. $\mathbf{\Gamma}$ is a tunable weighting matrix, allowing the difference between each measured and stored object feature to affect the total distance between measurement i and object j differently.

Conceptually there are two reasons to include the object features in the distance measurement. The first reason is to avoid associating measurements of a novel object with an object that is already being tracked. If object features were not included in the distance measurement, this could happen if the novel object is located on the same location as the predicted position (not the actual position) of an object that is already detected. However, using object features it is (sometimes) possible to detect that the object is not the object that was expected to be at said location, hence registering it as a new object in the object database. The second reason is that if there are two or more objects in the same image, noise in the measured position of the objects could easily cause the data association algorithm to mix up the association process. However, if the objects differ in appearance, using the object features to help perform the data association is powerful, as the appearance of an object is not as prone to noise as the measured position.

To associate a measured object position with the most likely among the tracking gaits, a matrix $\mathbf{D}$ expressing the distances from all n measurements to all m tracking gaits is calculated. Hence, the matrix $\mathbf{D}$ takes on the following form

$$\mathbf{D} = \begin{bmatrix} D_{1,1} & D_{1,2} & \cdots & D_{1,m} \\ D_{2,1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ D_{n,1} & \cdots & \cdots & D_{n,m} \end{bmatrix}. \tag{4.33}$$

If the distance $D_{i,j}$ exceeds some threshold $d > 0$, $D_{i,j}$ is set to infinity. This is because in the case of

$$D_{i,j} \geq d \tag{4.34}$$

the measurement i is not very likely to be a measurement originating from the object in tracking gait j. Hence, if all values along a column j are equal to infinity, it is assumed that a measurement for tracking gait j is not present. To cope with this, column j should be removed from $\mathbf{D}$, and the predicted object position should be used as the best available estimate for tracking gait j. Furthermore, if all values along a row i is equal to infinity, it is assumed that there exists no probable tracking gait for measurement i. In other words, this is most likely a measurement originating from a new, not yet tracked object. Hence, row i should be removed from the matrix $\mathbf{D}$, and a new tracking gait should be instantiated with measurement i as the initial position.

After calculating $\mathbf{D}$ and removing superfluous rows and columns, the data association problem is a matter of finding the combination of distances $D_{i,j}$ that yields the global minimum distance. This implies that the combination which is selected assigns exactly one measurement to exactly one tracking gait, in a way such that the total distance between all measurements and their assigned tracking gaits is the shortest achievable distance. This is a well studied problem, and can be solved by applying the Kuhn-Munkres algorithm [24] to the modified matrix $\mathbf{D}$.

4.4 UAV and Field Test

In order to test the performance of the object tracking framework described in the previous sections, several field tests were conducted. The platform and payload used for these tests was the X8 platform described in Section 2.4.1.

For the tests conducted in this research it was the performance of the detection, recognition and tracking modules of the system that was emphasized, hence the UAV was either remotely controlled by a UAV pilot or the path planning was done manually pre-flight. I.e, the path planner was not connected to the real-time feedback loop illustrated in Figure 2.1. Hence, during the field tests the path planning module was not doing any real-time on-board calculations, and the autopilot was simply following a predefined flight path.

All of the tests were carried out operating from a boat in the open sea just outside Horta in the Azores in the summer of 2015. The data set consists of two test flights where the captured thermal video stream contains one large boat (≈ 80 m long) and one small RHIB. The position and velocity of each of the two boats

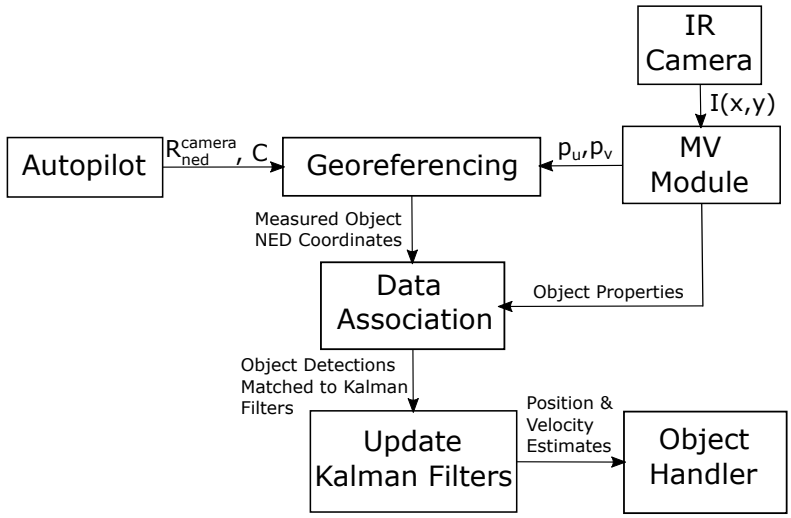

Figure 4.7: An overview of the tracking process at a given time step k. The CV module receives an infrared image $\boldsymbol{I}(x, y)$ from the infrared camera, and detects the centroid (p_u, p_v) of objects of interest within the image. This data is combined with navigation data from the autopilot (camera attitude, $\boldsymbol{R}_{ned}^{camera}$ and position $\boldsymbol{C}_{cam,ned}$) in order to georeference the object's pixel location. Having found the object's measured location in the NED frame, the object is associated with a specific Kalman filter based on the measured position and the object's features (size, temperature etc.). The Kalman filter associated with the measurement is then used to update the Kalman filter estimates.

were being tracked by a GPS device located in each boat, effectively supplying a relatively accurate measurement of their true position and speed (referred to as the "ground truth"). During the test flights the thermal video data, as well as the on-board telemetry data, were logged. However, when the UAV was loitering, the gimbal was controlled to point directly towards the center of the circle (using the pan servo), while the tilt servo was used to compensate for changes in the UAVs roll angle. This was done in order to stabilize the camera and make it point towards the ocean surface at the center of the loiter. The following results are the results of the detection and tracking algorithm tested in an offline real-time simulation using data gathered during the two test flights. However, computational tests show that the on-board computer use on average 0.07 seconds to perform the processing needed for each time step (image analysis, data association and updating Kalman filters). This means that running the object detection, recognition and tracking module online and in real-time are expected to yield near-identical results as the findings presented below.

Figure 4.8: Launch of the Skywalker X8, operating from a boat in the open sea outside Horta, Azores.

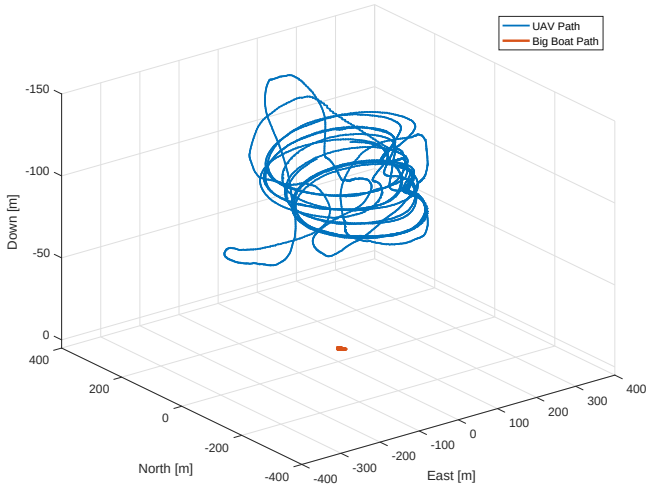

Figure 4.9: Flight path for the X8 during the first flight test.

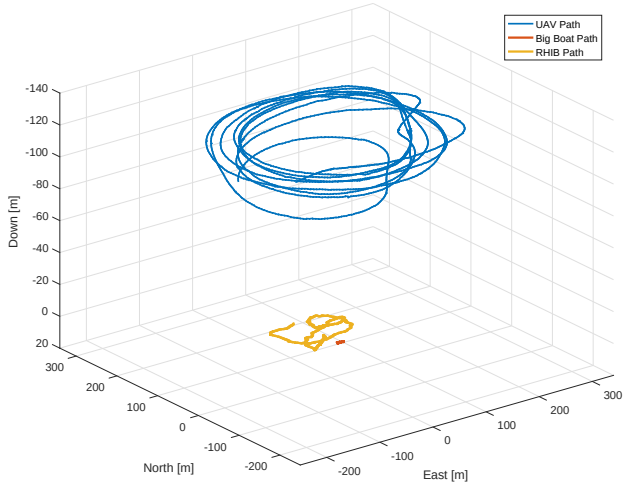

Figure 4.10: Flight path for the X8 during the second flight test.

4.5 Results

During the two test flights, the UAV was airborne for a total of 38 minutes. 16879 thermal images were collected, with 1165 of the images containing a large boat, 800 images of a small RHIB and 43 images where the RHIB and the large boat are visible simultaneously. The data set also includes 41 images of an autonomous underwater vehicle (AUV) resting at the ocean surface.

Setting the threshold, T_g, described in Section 4.2 to 230 and removing every detection consisting of less than 100 pixels, the object detection algorithm successfully detected the objects of interest (large boat, RHIB and AUV) in 99.25% of the images containing such objects. Furthermore, with the exception of 15 false detections (7 due to the horizon being visible, and 8 due to ripples in the water), the detection algorithm did not report any false positives. A false positive is when the object detection algorithm reports finding an object detection when there is, in fact, no object in the scene viewed by the camera.

For the tracking process, the Gaussian white noise variables q and r (measurement noise) were set to have a standard deviation of 45 m. Ideally, this is a parameter that should be scaled with the UAV's altitude. However, both flight tests was conducted at an altitude of $80 - 120$ m, where experimental data show that 45 m is

a reasonable choice for the measurement noise standard deviation. The Gaussian white noise variables w and z (motion model noise), will affect how quickly the Kalman filter adapts to actual changes in the velocity of the object being tracked. Hence, this parameter should ideally be chosen based on the type of object being tracked (classification). For the big boat being tracked in the conducted flight tests, the standard deviation of the motion model noise was set to 0.03 m/s. The thermal camera supplies the on-board computer with a new image 10 times per second, hence $\Delta t = 0.1$. The Kalman filter was initialized with a covariance matrix set to 45 m^2 for the positional estimates, and 0.3 m^2/s^2 for the velocities. For the soft start case (Kalman filter initiated with the measured average position using the 50 first measurements), the initial covariance matrix was set to 15 m^2 for the positional estimates and 1 m^2/s^2 for the velocities.

Following is a more detailed view on the performance of the object detection and tracking module for each flight.

4.5.1 Flight 1

During the first flight, the big boat was resting at a fixed position, heading approximately 45° North-East for the whole duration of the flight. Having the UAV loiter around the point where the big boat was located, as illustrated in Figure 4.10, the big boat was automatically detected a total of 362 times. The UAV was flying at altitudes varying from 80 m to 120 m. For each of registered big boat detection, a North-East position of the big boat was calculated using (4.31). This yielded a set of measurements of the big boat's position, which was distributed as shown in Figure 4.11. Using a Kalman filter as described in Section 4.3 with these position measurements yielded the tracking gait illustrated in Figure 4.12. It is observed that the initial position estimate is at -40 m East and 20 m North, approximately 80 m from the center of the boat's actual position. However, as more measurements of the big boat's position is put through the Kalman filter, the position estimate quickly (≈ 30 measurements) reaches the boat's interior. Further adding measurements to the Kalman filter improves the accuracy of the estimate even more, finally converging towards an area within $5 - 15$ m from the boat's actual position. Looking at the estimate of the boat's velocity, it was observed that given the nominal adjustments in the position estimate, the initial estimated velocity is relatively high (up to 15 m/s). However, as the position estimate converges towards the boat's center, the estimated velocity approaches 0 m/s, and stabilize around 0.3 m/s after ≈ 200 measurements. Note that from the 200th measurement mark, the boat's positional and velocity estimate is not further improved, but is

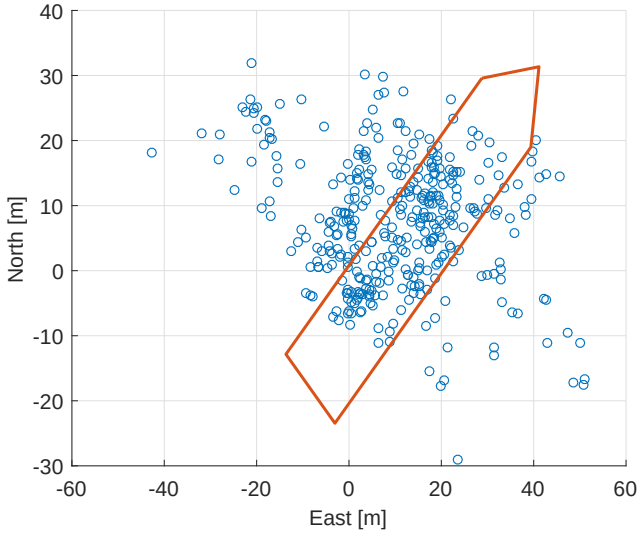

Figure 4.11: The distribution of the North-East position of the automatic detections of the big boat during the first flight.

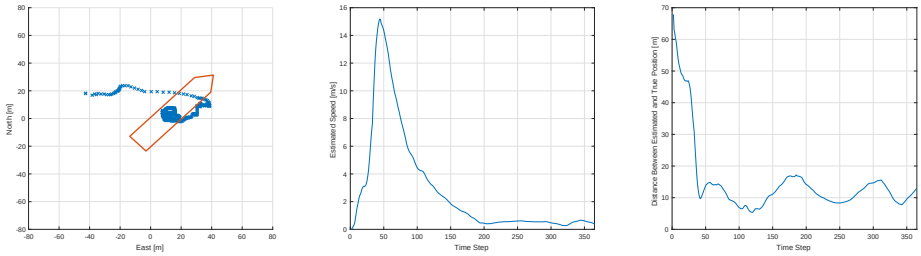

Figure 4.12: Kalman filter track of the big boat tracked during the first flight. The orange lines show the outline of the boat, and the blue line shows the estimated position of the boat for the duration of the flight.

maintained close to the true position and velocity throughout the duration of the flight.

In order to cope with the high standard deviation in the measurements of the big boat's position, a soft start of the Kalman filter was also tested. This implies that the Kalman filter is initiated not at the first detection, but at the average of the first 50 detections. Doing this resulted in the tracking performance illustrated in Fig

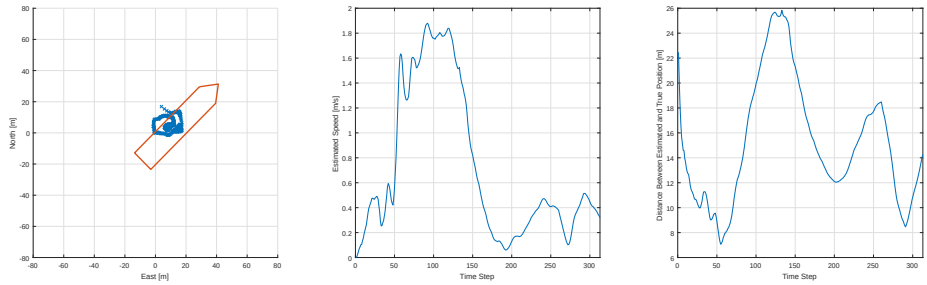

Figure 4.13: Soft started Kalman filter track of the big boat tracked during the first flight. The orange lines show the outline of the boat, and the blue line shows the estimated position of the boat for the duration of the flight.

4.13. It is readily seen that the tracking gait reaches the interior of the big boat almost immediately. Furthermore the peak of the estimated velocity is now reduced from 15 m/s to 1.9 m/s. However, it is seen that it is still $\approx$ 200 measurements (if the 50 measurements making up the initial position estimate is included) required to bring the estimated velocity to a stable value close to zero. The estimated speed stabilized around 0.3 m/s in this case as well. This means that if the UAV has to rely on predictions in order to know where to go to re-visit the big boat, the estimate would drift 18 m each minute, i.e almost 4 minutes before the estimate has drifted a full boat length (70 m).

4.5.2 Flight 2

In the second flight test, the big boat was still at a fixed position, but was now facing 60° North-West. During this flight, the big boat was detected in a total of 804 images, with each image giving a measurement of the boat's position in the NED frame. The distribution of the 804 measurements are shown in Figure 4.14. Furthermore, the estimated position and speed of the big boat based on these measurements are illustrated in Figure 4.15. Similar to the tracking performance observed during the first flight test, the initial position estimate is $\approx$ 30 m off of the boat's actual position. This estimate is however quickly adjusted (at the expense of a high speed estimate with a peak of 13 m/s), before both the position and the speed estimate stabilize around the true values (10 m North, 23 m East and 0 m/s speed) at the $\approx$ 250th measurement mark. As seen in Figure 4.15, a drift in the position estimate (the distance between the estimated and true object position increases) of the big boat from its true position is also observed in the period from approximately the 75th measurement to the 140th measurement. This

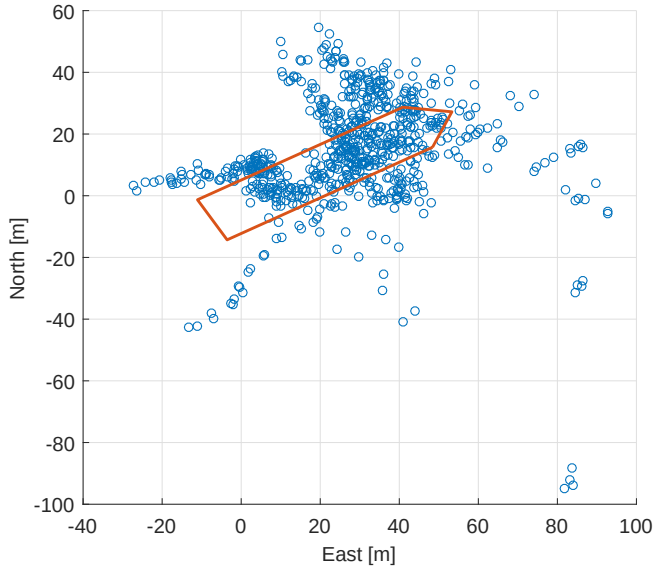

Figure 4.14: The distribution of the North-East position of the automatic detections of the big boat during the second flight.

can be explained by the fact that the big boat is only partly visible in the image frame during this period, effectively shifting the boat's measured centroid found by the image processing away from the true centroid of the boat. This could be avoided by using image processing to estimate the orientation of the object which is only partly visible, and combining this with an estimate of the object's size in order to calculate a more accurate centroid location for the object.

Using a soft start for the Kalman filter, where the average of the 50 first position measurements is used as the initial estimate for the boat's position, gives a similar tracking performance. However, the initial position estimate is more accurate, and the peak in the speed estimate is lower (5 m/s instead of 13 m/s).

During the second flight test, the RHIB was also moving slowly in the vicinity of the big boat. A GPS device on-board the RHIB made the ground truth position of the boat available. Figure 4.17 shows the RHIB's actual position (orange) in the periods where the RHIB was visible the image frame, together with the automatic RHIB detections and the measured RHIB position (blue). Unfortunately, the the maximum continuous sequence that the RHIB stayed in the image frame was only

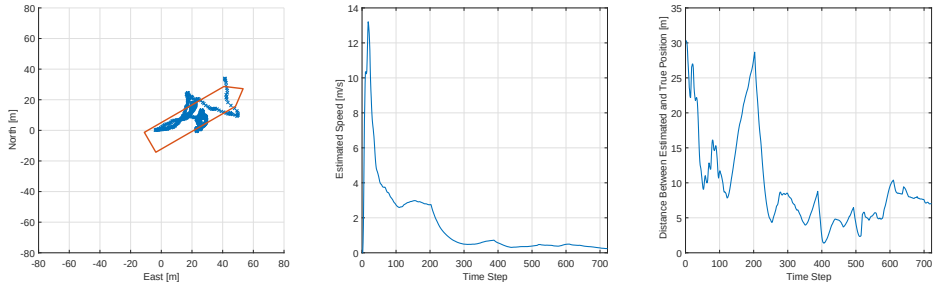

Figure 4.15: Kalman filter track of the big boat tracked during the second flight. The orange lines show the outline of the boat, and the blue line shows the estimated position of the boat for the duration of the flight.

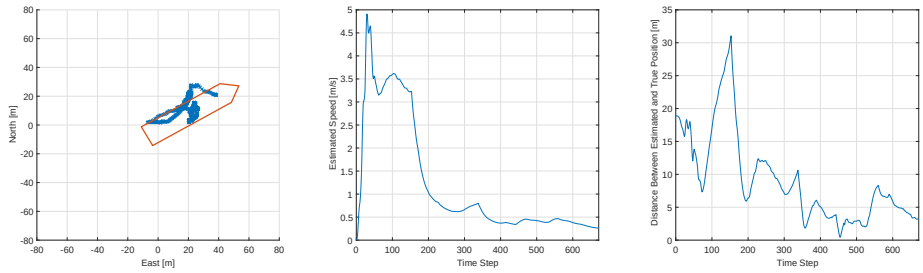

Figure 4.16: Soft started Kalman filter track of the big boat tracked during the second flight. The orange lines show the outline of the boat, and the blue line shows the estimated position of the boat for the duration of the flight.

106 images, with an average of 56 images. This was not sufficient to achieve reasonable tracking results for the boat's velocity using the methods described in this chapter. However, comparing the average measured position with the average RHIB position during the recorded sequences, the RHIB's position was estimated to be $\leq$ 10 m of it's actual position for every sequence. The actual position for a tracking sequence is here defined as the average position of the RHIB during the time the RHIB is visible in the image frame in a given sequence. Table 4.1 shows a more detailed overview of this for each captured sequence.

4.6 Conclusions

In this chapter, an automatic object detection, recognition and tracking algorithm has been developed. The algorithm is intended to be used in the generic object

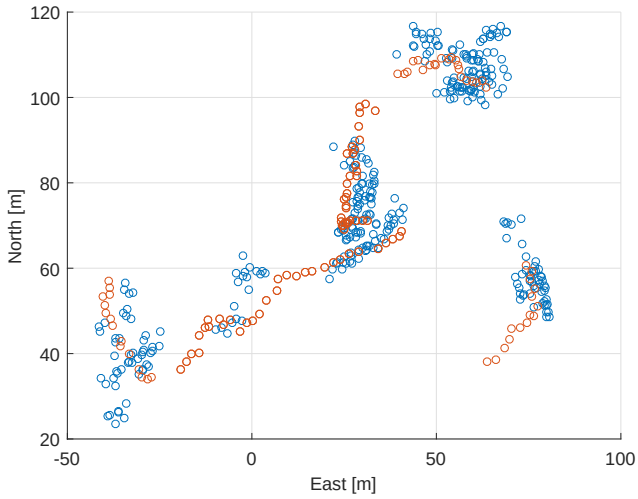

Figure 4.17: The RHIB's actual position (orange) and the measured position (blue) during the second flight test.

tracking system framework for UAVs presented in this thesis, in order to enable UAVs to perform multi-object tracking and situational awareness in real-time during a UAV operation. The detection algorithm uses a combination of edge detection and thresholding, together with dilatation/eroding and finding connected components in order to perform real-time automatic object detection from a thermal camera's video stream. Using on-board navigation data to get the UAV's and camera's attitude and altitude, the on-board computer is able to georeference each object detection to measure the location of detected objects in a local North-East-Down coordinate frame. Furthermore, the tracking algorithm uses a Kalman filter and a constant velocity motion model to perform object tracking based on the position measurements found using the object detection algorithm. To decide whether an object detection is a detection of an object already being tracked, or if the measurement originated from a novel object which has not been observed previously, the Kuhn-Munkres algorithm is applied for data association. This is achieved using a measure of distance that is based not only on the physical distance between an object's estimated position and the measured position, but also how similar the objects appear in the thermal image.

The presented object detection, recognition and tracking algorithm was tested in two different UAV flight tests conducted from a boat in the open sea. It was found

Table 4.1: Mean Position Estimates for Moving RHIB

	North Error	East Error	Total Error	Number of Detections
Sequence 1	5.7m	1.9m	6.0m	50
Sequence 2	8.4m	3.1m	8.9m	49
Sequence 3	1.0m	5.9m	6.0m	106
Sequence 4	3.6m	1.6m	4.0m	67
Sequence 5	1.3m	0.5m	1.4m	40
Sequence 6	5.4m	0.3m	5.4m	19
Average Error	4.2m	2.2m	5.3m	

that the system was able to successfully track the position of a big boat with an accuracy of $5 - 15$ m, and estimated the speed of the boat to be ≈ 0.4 m/s when the boat was staying at a fixed position. The results show that although the tracking was successful, the system required ≈ 50 measurements of the big boat's position in order for the position estimate to settle close to the ground truth (actual position), and $100 - 150$ measurements in order to have a sufficiently accurate estimate of the object's velocity. The tracking algorithm was also tested on a RHIB, moving in a random pattern in the vicinity of the big boat. Since the image sequences capturing the location of the RHIB was limited to segments of $19 - 106$ images, the velocity of the RHIB could not be successfully estimated. However, using a moving average estimate for the position of the RHIB, its location could be estimated with an accuracy of $5 - 15$ m for all the captured sequences containing images of the RHIB.

CHAPTER 5

Ice Management Application

This chapter is based on [61] (F. S. Leira, T. I. Fossen, and T. A. Johansen. A UAV ice tracking framework for sea ice management. *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, *Submitted*).

5.1 Introduction

In the object detection, recognition and tracking module described in the previous chapter, the algorithms implemented in the module were designed to detect, recognize and track rigid objects such as boats at the ocean surface. This is useful when the number of objects to detect, recognize and track is limited and there is a certain distance between each object.

An interesting application for the object detection and tracking framework developed in this thesis is in the application of sea ice management, where ice management is the sum of all activities where the objective is to reduce or avoid actions from any kind of ice features [31]. This could for instance be the detection, tracking and forecasting of sea ice, ice ridges and icebergs using UAVs. The difference in the scenario of ice management to the scenario of for example detecting and tracking boats is that the ice management system has to be able to deal with potentially hundreds or thousands of "objects" (icebergs or ice floes etc.) simultaneously. Furthermore, the size of an iceberg could be so big that the whole iceberg could not be captured in a single image frame of the thermal camera when the UAV is operating at a typical altitude of $100 - 150$ m. Another problem of utilizing the object detection, recognition and tracking module developed in the previous chapter for ice management is that it would require a Kalman filter in order to track each iceberg or ice floe individually. This would in many scenarios not be feasible. This calls for an alteration of the object detection, recognition and

tracking approach when the overall object detection and tracking framework is to be used in the area of ice management.

The use of aerial sensors and UAV platforms in ice management has become more and more popular in recent years, and some proposed solutions already exists. The next section covers the recent efforts in the field of ice management using UAVs.

5.1.1 Related Work

[46] presents a possible structure for a general ice observer/ice management system. The presented ice observer system structure is made to be able to collect, analyse and employ ice intelligence during operations in ice management, and UAVs are presented as a viable sensor platform for these operations. Moreover, it illustrates how UAVs can be equipped with sensors that provide information about the environment, including wind velocity, iceberg and ice floe distribution and ice velocity. This is also demonstrated in [65] where a small UAV is equipped with a camera and an ice monitoring experiment is conducted. Using a simple thresholding technique, images from a UAV flight is segmented into ice and non-ice regions, effectively demonstrating the UAVs usefulness in ice management settings.

[32] presents an algorithm that is able to use stereoscopic optical imaging from UAVs in order to image sea ice surface 3D structure. The algorithm uses structure from motion [105] in order to generate a 3D model of the sea ice structure, and the resulting model can be utilized in the process of finding a safe passage for a boat through a region with thick ice. Structure from motion is also used in [90] to generate 3D models of a glacier by the use of image data capture with a UAV. Although the structure from motion algorithm is too computationally expensive to be executed on-board and in real-time, this work further proves that the UAV is a viable and useful platform in the area of ice management.

[98] and [15] both presents path planning algorithms for UAVs in ice management based on optimization techniques. [98] develops an algorithm to find a path pattern that covers a predefined region of interest. The region is chosen based on the location of a boat and an ice flow estimate, yielding a region where potential icebergs and ice floes headed directly towards the boat are expected to be found. The region is divided into a grid of cells and the UAV's visitation sequence for the grid cells is found by solving a MILP problem seeking to minimize the time required to cover the total grid. [15] assumes that a set of n icebergs have been detected and located, and that the UAV is given an estimate of each of the iceberg's positions. It is further assumed that the uncertainty of the position estimate for the location of each

iceberg is reset to 0 when the UAV is in its vicinity in able to observe its position. A MILP problem is then defined and solved, yielding a visitation sequence that minimize a weighted cost function of total distance travelled by the UAV, and the total uncertainty in the position estimates of the icebergs. Both of these algorithms present useful applications of UAVs in the area of ice management, but does not describe the remainder of the components, such as a detailed description on incorporating sensor measurements into their solutions, that would be necessary in order to have a completely autonomous ice management system for UAVs.

[44] presents an ice concentration and flow monitoring system that is able to create feasible and collision-free paths for a set of UAVs, where the paths are generated in such a way as to minimize the uncertainty in the state estimate of the ice flow. The problem is formulated as an optimal control problem which use a kinematic model for UAVs and a finite element discretization of a uniformly drifting sea ice concentration field. The system's measurements are segmented images of the sea ice captured from a camera on-board the UAVs, which then maps the concentration of pixels segmented as sea ice onto a tessellated map in world coordinates. The approach is very useful for ice management, as it can be used to perform automated mapping of sea ice and ice flow in a predefined region. However, simulations show that the algorithm becomes increasingly computationally demanding as the search region increases in size. A way to deal with the scalability issue would have to be implemented in order to enable the algorithms developed in [44] to be applicable in a realistic ice management scenario for UAVs.

A similar but less computationally demanding approach to the problem of mapping the ice concentration in a predefined region is presented in [36]. This algorithm uses an occupancy grid map method for mapping a predefined search region for ice features based on images captured from a UAV and image processing. The occupancy grid map is a collection of grid cells where each cell is given a probability of being occupied by an object of interest. The occupancy probability for each grid cell is then updated by segmenting images into ice and non-ice regions. Generating an occupancy grid map is useful in many ice management scenarios, for instance when a boat is to safely traverse a region with icebergs and ice floes. However, this would require the UAV to not only generate an occupancy grid map, but the UAV should also be able to keep track of the movement of the ice features found which are in the vicinity of the boat's planned route.

5.1.2 Contribution

In this chapter, the focus is on the development, implementation and integration of an alternative object detection and tracking module and an alternative object handler module intended to be used in the general UAV object detection and tracking framework described in this thesis. The main difference between the object detection and tracking module and the object detection, recognition and tracking module presented in the previous chapter is that the module developed in this chapter is intended to be used when the objects of interest does not consists of a few rigid objects, but either entire regions of interest or too many objects of interest to track each one individually. This is achieved by using the algorithm developed in [36] in order to create an occupancy grid map of an area of interest, and extending the work presented in [36] by bridging the gap between the occupancy grid map and the path planner module in the object detection and tracking framework. In order to do this, an algorithm uses the occupancy grid map to generate locations of interest is developed. This algorithm which functions as an object handler module, and the generated locations of interest are generated in a way such that they are easily passed to the MPC developed in Chapter 3, such that the MPC can make the UAV keep track of these locations. This chapter also verifies through some initial testing the applicability of the object detection and tracking framework in a setting where UAVs are used for ice management.

5.1.3 Overview of the Chapter

The remainder of this chapter is organized as follows. First, an occupancy grid map algorithm coupled with a measurement update model is presented. Second, a locations of interest generation algorithm which uses an occupancy grid map as input is developed. Section 4.4 covers the details of a field test conducted in order to test the object detection and tracking framework with the occupancy grid map and the locations of interest generator presented in this chapter, while Section 4.5 covers the results of the conducted flights. Finally, the chapter is summarized and concluded.

5.2 Occupancy Grid Map

Occupancy grid maps are world fixed grid maps that tries to estimate the probability of whether the grid locations in the map are occupied or free. This is a powerful approach to mapping objects or regions of interest in a predefined search area, especially in scenarios such as ice management where there often are too many objects to track the position and movement of each object of interest indi-

vidually. A brief overview of an occupancy grid map algorithm developed in [102] and expanded to include the possibility of a dynamic map in [36] is given in this section.

When using an occupancy grid map, the purpose is to estimate the posterior probability density over all possible maps given past measurements and states, i.e

$$p(m_t|z_{1:t}, x_{1:t}) \tag{5.1}$$

where m_t is the map representation, $z_{1:t}$ are all measurements/observations of the state of the occupancy grid map (described in more detail in Section 5.2.1) and $x_{1:t}$ are the UAV's position and attitude for all time steps from 1 to t.

Now, representing the map as a 2D grid map

$$m_t = \{\mathbf{m}_{i,t}\} \tag{5.2}$$

where $\mathbf{m}_{i,t}$ are grid elements with an associated (binary) occupancy value, we are interested in knowing the probability of $p(\mathbf{m}_{i,t} = \text{occupied})$ and $p(\mathbf{m}_{i,t} = \text{not occupied})$ for each grid cell $\mathbf{m}_{i,t}$. The term 'occupied' will in our case mean that at least one iceberg or ice floe occupies grid cell i. An assumption made in [102] in order to calculate Equation (5.1) is that the grid cell densities are independent to each other. That is,

$$p(m_t|z_{1:t}, x_{1:t}) = \prod_i p(\mathbf{m}_{i,t}|z_{1:t}, x_{1:t}) \tag{5.3}$$

In some cases this could be an inaccurate assumption as the probability of locating icebergs or ice floes in one grid could indicate a higher probability of finding icebergs or ice floes in neighbouring grid cells. However, incorporating this conditional probability would greatly increase the complexity of the problem, while not necessarily significantly increase the occupancy grid map's accuracy [102].

Furthermore, assuming that $\mathbf{m}_{i,t}$ and x_t are Markov, i.e

$$p(\mathbf{m}_{i,t}|z_{1:t-1}, x_{1:t-1}) = p(\mathbf{m}_{i,t}|\mathbf{m}_{i,t-1}) \tag{5.4}$$

combined with the fact that estimating the state of an individual grid cell is a binary estimation problem (occupied or not occupied), it is possible to estimate the entire grid state by using a binary Bayes filter to estimate the state of each grid cell individually [102].

The reader is referred to [36] for a detailed description of the development of the binary Bayes filter, but the main concept is to exploit that the probability of grid cell $\mathbf{m}_{i,t}$ not being occupied, $p(\neg \mathbf{m}_{i,t})$, can be given as

$$p(\neg \mathbf{m}_{i,t}) = 1 - p(\mathbf{m}_{i,t}) \tag{5.5}$$

and defining the following odds ratio

$$\frac{p(\mathbf{m}_{i,t}|z_{1:t}, x_{1:t})}{p(\neg \mathbf{m}_{i,t}|z_{1:t}, x_{1:t})} \tag{5.6}$$

[36] then develops an additive update equation for each grid cell $\mathbf{m}_{i,t}$ by taking the logarithm of Equation (5.6). This yields the following additive update equation

$$l_{i,t} = \log_{10} \frac{p(\mathbf{m}_{i,t}|\mathbf{m}_{i,t-1})}{1 - p(\mathbf{m}_{i,t}|\mathbf{m}_{i,t-1})} + \log_{10} \frac{p(\mathbf{m}_{i,t}|z_t, x_t)}{1 - p(\mathbf{m}_{i,t}|z_t, x_t)} - \log_{10} \frac{p(\mathbf{m}_{i,t})}{1 - p(\mathbf{m}_{i,t})} \tag{5.7}$$

where the first term will be $l_{i,t-1}$ if the map is static. Using for instance the system developed in [69], an estimate of ice drift can be found by analysing synthetic aperture radar (SAR) images from a satellite. This estimate could be supplied to the UAV ice management system and incorporated in the derivation of this update equation, resulting in that the first term would also include a component caused by the map dynamics (ice drift). $p(\mathbf{m}_{i,t}|z_t, x_t)$ is referred to as the inverse measurement model and is covered in the next subsection, while the last term can be interpreted as the value returned by the inverse measurement model when no information is supplied by a measurement.

Note that in order to get the actual grid map probabilities we have to calculate

$$p(\mathbf{m}_{i,t}|z_{1:t}, x_{1:t}) = \frac{exp(l_{i,t})}{1 + exp(l_{i,t})} \tag{5.8}$$

5.2.1 Measurement Update

Looking at the additive logarithmic update equation developed in the previous section, i.e (5.7), it is readily seen that the inverse measurement model has to be decided in order to calculate $l_{i,t}$. That is, the term

$$p(\mathbf{m}_{i,t}|z_t, x_t) \tag{5.9}$$

has to be defined. This term is the probability of the map grid cell i being occupied given an image z_t and a UAV position and attitude x_t. There are many

possible choices for this probability function, with the only requirement being that $p(\mathbf{m}_{i,t}|z_t, x_t) \mapsto [0, 1]$.

Ideally this probability function should be found in a systematic way using empirical data, such as images of the objects or regions of interest captured by an UAV. However, since such a data set was not available, a simple probability function assuming that an image has been segmented into foreground (objects or regions of interest) and background (non-interesting regions) was used for demonstration purposes. More specifically, assuming that each pixel $s_{k,t}$ of the image z_t is segmented into foreground and background, we define that

$$n_{k,t} = \begin{cases} \log_{10} \frac{P_1}{1-P_1} & s_{k,t} = \text{Object of Interest} \\ \log_{10} \frac{P_2}{1-P_2} & s_{k,t} = \text{Non-interesting Region} \end{cases} \tag{5.10}$$

where $n_{k,t}$ is a partial measurement model. Note that P_1 should be a parameter reflecting the object detection algorithm's robustness against false positives. That is, if the object detection algorithm has a low percentage of false positives, P_1 could be set close to 1 because in such cases a pixel segmented into the "object of interest" category means that this pixel is very likely to originate from an actual object of interest. If the object detection algorithm has a high percentage of false positives, P_1 should reflect this by being set to a lower, non-zero value. Similarly, P_2 reflects the object detection module's percentage of false negatives. I.e, if the object detection module rarely fails to detect objects of interest if they are actually present, P_2 could be set to 0 since a pixel segmented into the non-interesting region is in such cases very likely to actually be a non-interesting region. On the other hand, if the object detection module regularly fails to detect objects of interest when they are present, P_2 should be set higher than 0 to reflect that a pixel segmented into the non-interesting region not necessarily originated from a non-interesting region but in fact an object of interest.

To have a complete measurement model, the UAV's position and attitude x_k has to be used in order to map a pixel coordinate in the image frame to a grid cell in the map frame. The occupancy grid map can be represented by a $N \times M$ matrix, and each cell in the grid map can be represented with homogeneous coordinates $\mathbf{p}^m = [x^m, y^m, 1]$. $x^m \in \{0, ..., N-1\}$ denotes the index of the cell in the horizontal direction, and $y^m \in \{0, ..., M-1\}$ denotes the index of the cell in the vertical direction. Defining the grid map to be in the positive quadrant of the $x-y$

plane in the map frame, the center of a grid cell in the map frame can be found by

$$\mathbf{q}^m = \begin{bmatrix} D_x & 0 & \frac{1}{2}D_x \\ 0 & D_y & \frac{1}{2}D_y \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^m \\ y^m \\ 1 \end{bmatrix} = \mathbf{M}\mathbf{p}^m \tag{5.11}$$

where $\mathbf{q}^m$ is the center of grid cell $\mathbf{p}^m$ given in map frame coordinates, and D_x and D_y are metric scaling factors describing the size of a grid cell in the x and y direction. Now, the center of a given grid cell $\mathbf{p}^m$ can found in a NED coordinate frame with the following equation

$$\begin{aligned} \mathbf{q}^{ned} &= \mathbf{R}_m^{ned}\mathbf{M}\mathbf{p}^m - \mathbf{R}_m^{ned}\mathbf{C}_{ned,m} \\ &= (\mathbf{R}_m^{ned}\mathbf{M} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & -\mathbf{R}_m^{ned}\mathbf{C}_{ned,m} \end{bmatrix})\mathbf{p}^m := \mathbf{P}_m^{ned}\mathbf{p}^m \end{aligned} \tag{5.12}$$

where $\mathbf{R}_m^{ned}$ is the rotation matrix from the map frame to the NED frame and $\mathbf{C}_{ned,m}$ is the position of the origin of the NED frame given in the map frame. Having found the location of a grid cell's center in the NED frame, the location of a grid cell's center in the image frame, $\mathbf{p}^{img}$, can be calculated. That is

$$\begin{aligned} w_s\mathbf{p}^{img} &= \mathbf{A}(\mathbf{R}_{ned}^{cam}\mathbf{P}_m^{ned}\mathbf{p}^m - \mathbf{R}_{ned}^{cam}\mathbf{C}_{cam,ned}) \\ &= \mathbf{A}(\mathbf{R}_{ned}^{cam}\mathbf{P}_m^{ned} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & -\mathbf{R}_{ned}^{cam}\mathbf{C}_{cam,ned} \end{bmatrix})\mathbf{p}^m := \mathbf{X}_m^{img}\mathbf{p}^m \end{aligned} \tag{5.13}$$

where $\mathbf{R}_{ned}^{cam}$ is the rotation matrix from the NED frame to the camera frame as defined in (4.24) and $\mathbf{C}_{cam,ned}$ is the position of the camera given in the NED frame. w_s is still a scaling factor as described in (4.22) and $\mathbf{A}$ is the intrinsic parameter matrix for the on-board camera as defined in (2.1). Hence, $\mathbf{X}_m^{img}$ is the perspective transformation between the map frame and the image frame. This means that a segmented pixel $\tilde{\mathbf{p}}^{img}$ in image z_t can be mapped to a grid cell coordinate in the map frame, $\tilde{\mathbf{p}}^m$, by the following equation

$$\frac{1}{w}\tilde{\mathbf{p}}^m = (\mathbf{X}_m^{img})^{-1}\tilde{\mathbf{p}}^{img} = \mathbf{X}_{img}^m\tilde{\mathbf{p}}^{img} \tag{5.14}$$

Using this equation with the partial measurement model in (5.10) and the additive update rule in (5.7), the occupancy grid map can be updated to reflect the observations made by the object detection algorithm for every segmented image.

5.3 Generating Locations of Interest

An initial occupancy grid map for a given region can made be by having a UAV search through a pre-defined search region in a systematically manner using the

Figure 5.1: An example occupancy grid map.

occupancy grid map method with the proposed measurement update described in the previous section. However, the initial occupancy grid map could also be generated from other types of data, such as for instance data from a SAR satellite. An example of an initial occupancy grid map resulting from such methods is shown in Figure 5.1. The map shows a total of 6 regions of interest where the occupancy grid map has cells with a probability of being occupied larger than 0. In order to make this occupancy grid map useful for the object detection and tracking framework developed in this thesis, a method to generate locations for the MPC to track has to be established. There are many possible algorithms for generating these locations, however, in this thesis the focus is on an approach which is directly compatible for the overall object detection and tracking framework.

The general approach to incorporate the occupancy grid map into the system is to use the map to generate locations which the MPC should track. The first step in the process of generating these locations is to perform a thresholding technique on the occupancy grid map (Figure 5.1) using a certain threshold $T_i \in [0, 1]$. Note that this thresholding step is identical to the thresholding technique described in Chapter 4 and (4.4). The reason for doing this is that when automatic image processing is used in order to perform the measurement update step of the occupancy grid map, false positives could cause the occupancy grid map to have (small) regions (a small number of cells) which has non-zero probability, while still not being occupied by any objects of interest. Since the false positives usually are not as consistent as the true positive detections made by the machine vision algorithm, the idea is that regions with a non-zero probability caused by false positives will

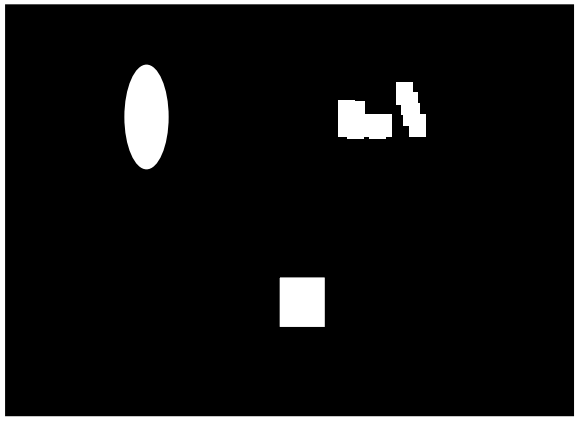

Figure 5.2: An example thresholded occupancy grid map. Note that one region has been removed because its size was too small (far left), and one region has been removed since non of the grid cells in this region had a value higher than the threshold value T_i.

still have a smaller probability than regions with a non-zero probability caused by true positives. Thresholding the occupancy grid map will yield a binary map where each cell is either a 1 or a 0. This binary map can optionally be filtered to remove blobs which are larger or smaller than a certain size, depending on what kind of regions of interest the system is looking for. An example of a binary map made by thresholding and filtering the blobs based on their size from the original occupancy grid map shown in Figure 5.1 is illustrated in Figure 5.2.

Furthermore, in some object detection and tracking applications, it could be beneficial if regions which are estimated to be occupied by objects of interest which are close to each other are viewed as one connected region. This is especially useful for the scenario of ice management, where it is useful to look at several icebergs and ice floes simultaneously when trying to estimate their general velocity and movement. In order to make regions of interest close to each other appear as one connected region, the binary map generated in the previous step can undergo a process called dilation. This is a process which probes and expands the shapes contained in an input image (in this case a thresholded occupancy grid map) using a predefined geometric form (for instance a circle). The result of dilating the binary map in Figure 5.2 is shown in Figure 5.3. Note how the previously disconnected regions on the right side of the map now is connected. It should also be noted that since that by dilating an image and expanding the shapes contained in it, the dilated thresholded occupancy grid map will set grid cells which are not likely

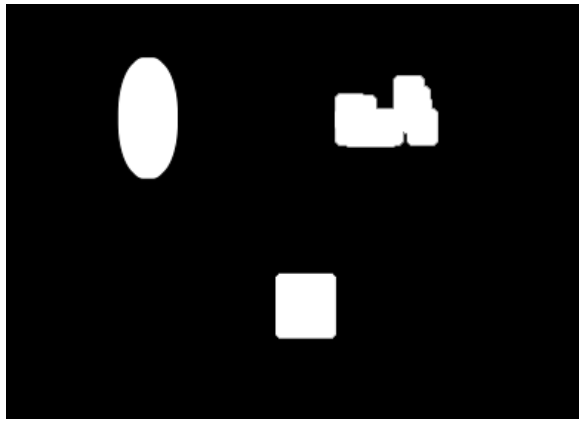

Figure 5.3: An example dilated thresholded occupancy grid map. Note how the previously disconnected regions on the right side now is connected to each other. Also note that the two other regions have been expanded as well.

to contain any ice features as occupied cells, care should be taken when the UAV is sent to investigate these regions. That is, the dilated thresholded occupancy grid map can be used in order to decide where the UAV should go and in which order each region should be visited, but ultimately the tracking process (e.g the control of the gimbal) should be based on the originally thresholded occupancy grid map shown in Figure 5.8.

Having generated a binary occupancy grid map of an area of interest, the locations for the MPC to track is simply generated by finding the centroid of each of these regions. Feeding the Object Handler and the MPC with these points will work as a replacement for an estimate of the position of rigid objects, as the object detection, recognition and tracking module from the previous chapter was implemented to supply to the remainder of the system. Using the centroid of the regions found to be occupied by objects of interest is effective as long as the regions of interest can be completely captured within one image frame. However, if this is not the case, a more complex algorithm generating several locations per occupied region might be necessary. An example of an algorithm which could be used to generate several loitering points that will completely cover such regions can be found in [38].

5.4 UAV Field Test

An initial field test of the occupancy grid map and the locations of interest generator integrated with the overall object detection and tracking system was tested

Figure 5.4: Launch of the X8 UAV at Ny-Aalesund. Picture courtesy of Kjell Sture Johansen.

with the X8 UAV payload at Ny-Aalesund, Svalbard in the spring of 2016. Figure 5.4 shows how the X8 UAV platform is launched at the airport of Ny-Aalesund using a catapult.

Two field tests were conducted, where the purpose of the first field test was to gather example thermal video footage of icebergs and ice floes floating in the fjord just outside Ny-Aalesund. This was necessary in order to see if the segmentation algorithm developed in Chapter 4 could be used to segment an image into ice and non-ice regions required for the measurement update step described in Section 5.2.1. More specifically, this means that both the threshold value T_g from (4.4) and the size of the icebergs and ice floes that were to be detected had to be tuned according to the thermal video data gathered during the first flight test.

After having tuned the ice detection machine vision algorithm according to the actual thermal video data gathered, the second flight test could be conducted. The purpose of this flight test was to illustrate how the overall object detection and tracking system can be used as a UAV ice management platform. Unfortunately, at the day of the testing there was only one iceberg within the reach of the X8 UAV platform, hence a full scale test of the system for ice management has to be tested at a later time.

In order for the occupancy grid map to be established and contain the iceberg located within the reach of the X8, the flight plan illustrated in Figure 5.5 was utilized. The iceberg was estimated (by human observation) to be somewhere inside the red square marked on the map. As seen from Figure 5.5, using the shown

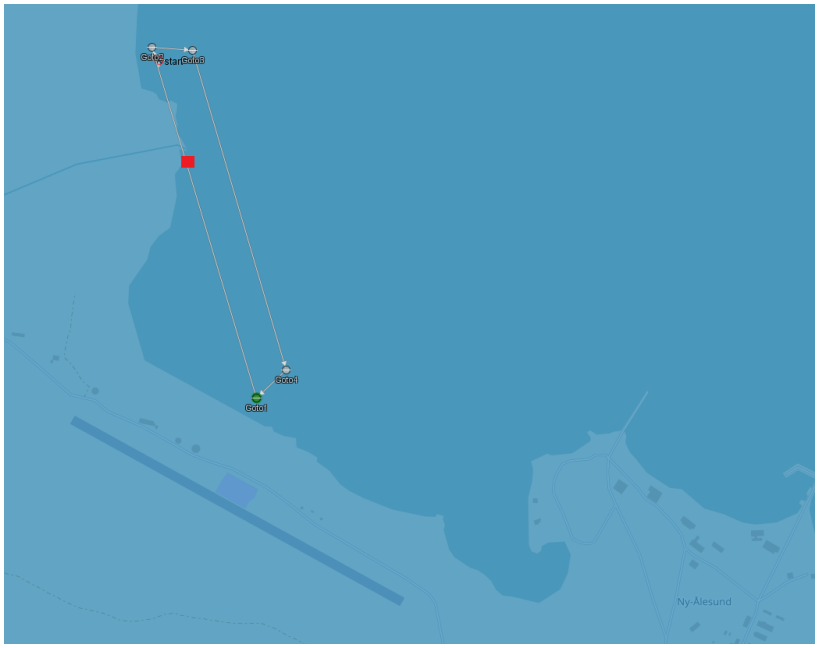

Figure 5.5: The flight plan for iceberg detection and tracking.

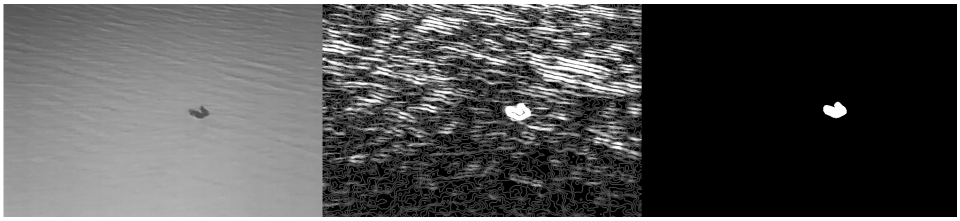

Figure 5.6: The iceberg detection process using machine vision.

flight plan to find an occupancy grid map will not yield a complete occupancy grid map over the fjord outside Ny-Aalesund. However, since there only was one iceberg within the reach of the UAV and the flight time of the X8 is somewhat limited, it was decided that finding an occupancy grid map for the search pattern illustrated was sufficient.

The following is the results of the conducted flight tests, and is the first test where the machine vision algorithm directly interacts with the MPC module in the presented object detection and tracking framework. Note however that once the MPC was activated with a generated location for the detected iceberg, the occupancy grid map was not further updated based on new detections of the tracked iceberg.

5.5 Results

Using the object detection algorithm developed in Section 4.2 on the thermal video data gathered during the first flight test proved to be very successful at segmenting the thermal images into iceberg/ice floe regions and non-interesting regions. An example of such a segmentation is shown in Figure 5.6. This figure is an example on how, when choosing appropriate values for the threshold T_g and the size of blobs that should be filtered, an iceberg is segmented into foreground while the rest of the ocean surface is segmented into background. $T_g = 230$ was found to be a threshold value that yielded a robust iceberg and ice floe detector. Furthermore, filtering out all blobs that consisted of less than 100 pixels was found to remove small blobs which could occur from time to time because of waves and ripples in the ocean surface.

Having tuned the iceberg and ice floe segmentation algorithm, the second flight test could then be conducted and the occupancy grid map algorithm and the locations of interest generator could be tested with the overall system. The occupancy grid map was initialized as a 600×1200 m^2 map where each cell was 10×10 m^2 in size. The initial logarithmic probability, $l_{i,0}$ was set to 0, which means that each grid cell was

assumed to have an equal probability of being occupied or not. P_1 was set to 0.8 and P_2 was set to 0.2. Flying 1 round of the flight path shown in Figure 5.5 while updating the occupancy grid map based on the captured segmented thermal images yielded the development of the occupancy grid map that is shown in Figure 5.7. Note that for simplification purposes, the gimbal on-board the X8 UAV platform was set to point directly downwards in the UAV body frame. This was done to reduce the uncertainty in the projection of pixels into the map frame as described in Section 5.2.1.

Using the occupancy grid map shown in Figure 5.7, a location of interest was generated by thresholding this map with $T_i = 0.8$ after the UAV had completed 1 round of the flight plan. This yielded the thresholded occupancy grid map shown in Figure 5.8, with the centroid of the blob located at -225 m North and 60 m East in the North-East plane. This point was then autonomously given to the MPC developed in Chapter 3, and the MPC was activated with a reference loiter radius of 125 m. The resulting iceberg tracking behaviour is illustrated in Figure 5.9, while the distance from the X8 UAV to the estimated location of the iceberg is shown in Figure 5.10.

Looking at Figure 5.9 it is readily seen that the MPC is successful at keeping the UAV in the wanted tracking range of ~ 125 m to the located iceberg for most of the time. However, on the right hand side of the loiter a more unstable tracking pattern is observed. The unstable tracking pattern is also visible in Figure 5.10, where it is observed that the UAV's distance to the estimated iceberg location occasionally dips $10 - 30$ m below the 125 m reference range. Although the data of the UAV's signal strength to the ground station is not available, a temporary loss of signal was sometimes observed when the UAV was traversing the right hand side of the loiter. Since the MPC was implemented in the ground station as described in Section 2.4, a loss of communication between the UAV and the ground station will cause the ground station to not receive updated telemetry data from the UAV. Hence, the ground station MPC believes that the UAV is standing still during the time that the communication is lost since it will assume the most recent received telemetry message is the current state of the UAV. This causes the MPC to generate a waypoint which is compliant with the belief that the UAV is at a different position than what it really is. When the communication link between the UAV and the ground station is re-established, this wrongly calculated waypoint is the first waypoint the UAV receives. However, the ground station will quickly receive a telemetry update from the UAV, which enables the ground station MPC to calculate a more reasonable waypoint, correcting the UAV course so that it can

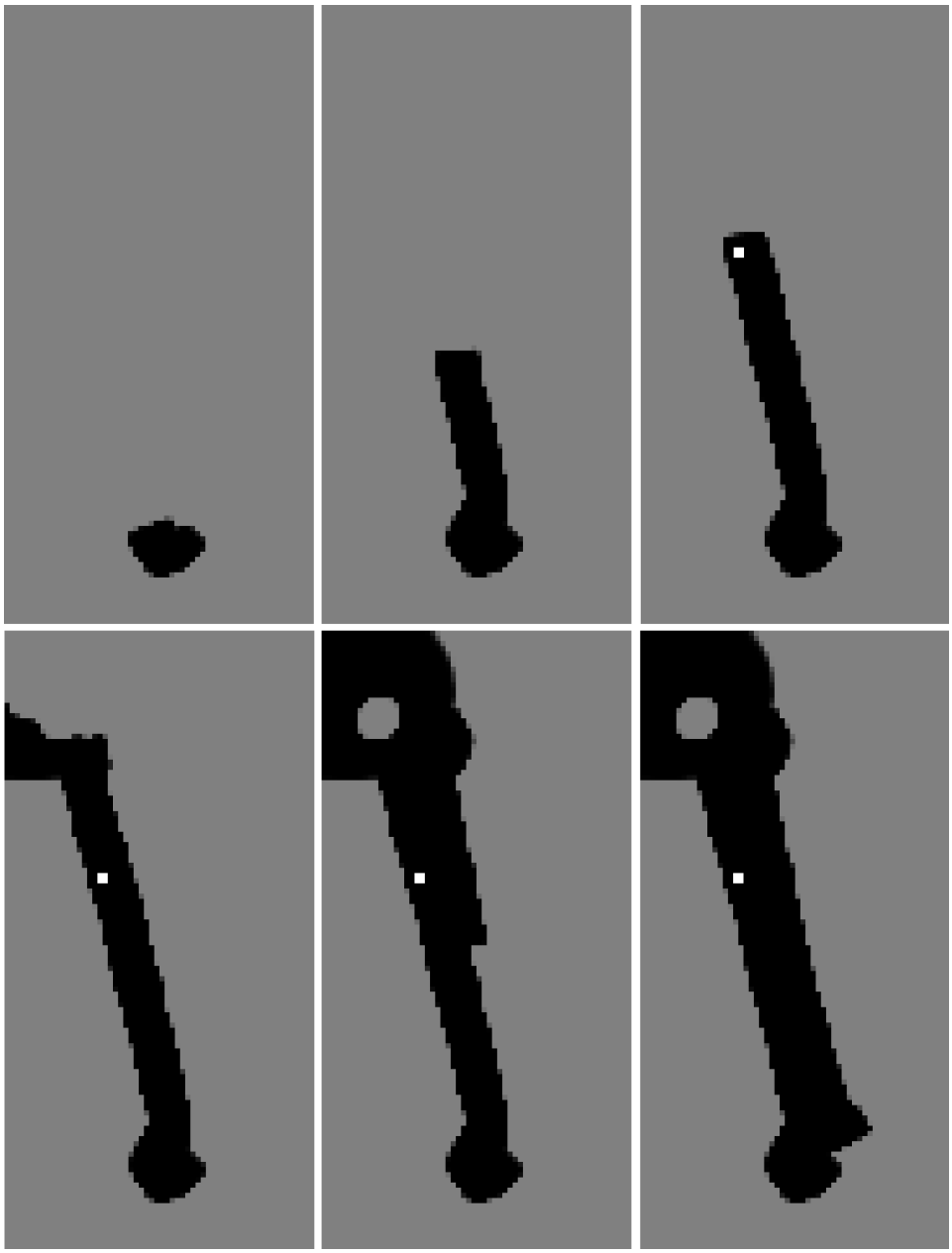

Figure 5.7: The development of the occupancy grid map during the second flight test conducted at Ny-Aalesund.

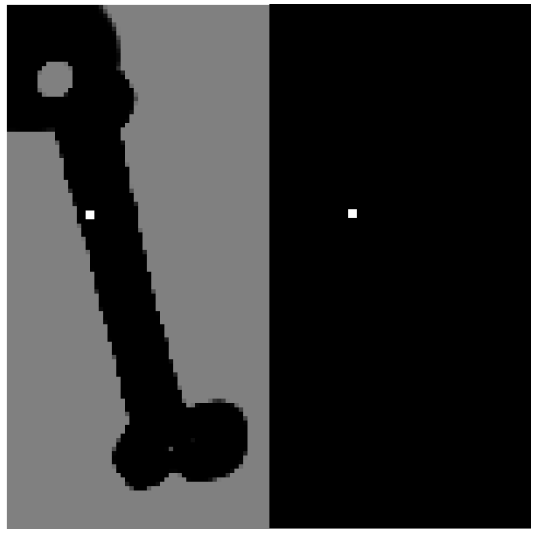

Figure 5.8: The complete (left) and thresholded (right) occupancy grid map for the second flight test.

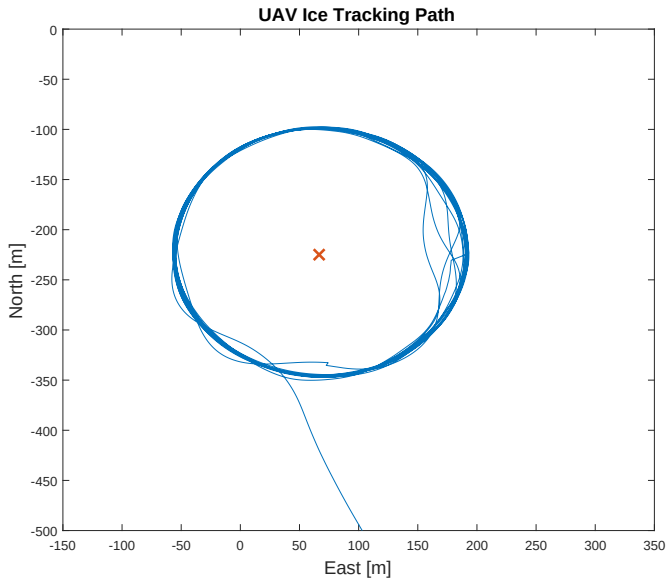

Figure 5.9: UAV path while using an MPC to track an iceberg.

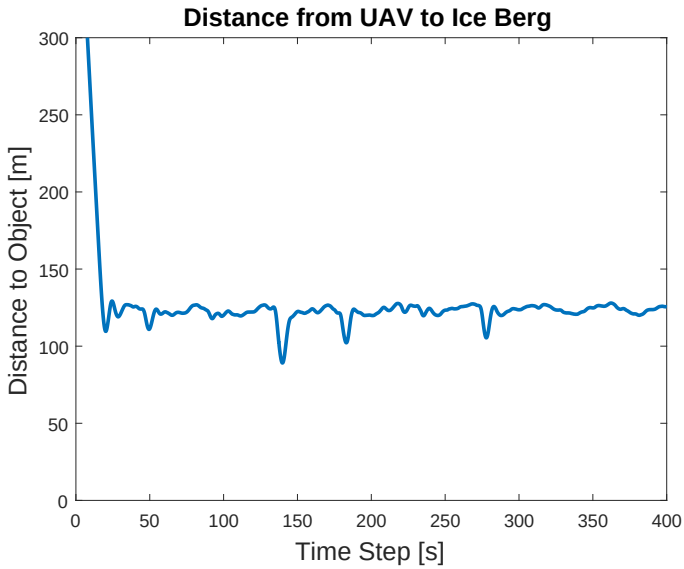

Figure 5.10: Distance from the UAV to the iceberg.

continue to follow the ideal loiter pattern. This problem could be mitigated by implementing the MPC completely on-board and getting rid of the MPC's ground station component.

5.6 Conclusions

In this chapter, an alternative machine vision module to the one presented in Chapter 4 was developed. The module is intended to be used in the object tracking system framework presented in this thesis, in order to enable UAVs to perform tasks such as for instance real-time ice management. The detection algorithm developed in the previous chapter was proven to be successful in segmenting the UAV's thermal images into ice and non-ice regions by tuning the algorithms parameters. Furthermore, using on-board navigation data to get the UAV's and camera's attitude and altitude, the on-board computer is able to automatically generate and update an occupancy grid map over an area of interest, where each cell in the grid map is given a probability of being occupied by an iceberg or ice floe. The generated occupancy grid map can in turn be used in order to generate locations of interest, which are locations that the UAV should investigate and keep track of. The object detection and tracking framework was found to be easily integrated with the occupancy grid map and the locations of interest generator.

The presented machine vision module was integrated with the overall object detection and tracking system presented in this thesis, and was tested in a UAV flight test conducted in the fjord outside Ny-Aalesund at Svalbard. It was found that the system was able to successfully create an occupancy grid map based on automatically segmenting thermal images into ice and non-ice regions. Furthermore, using this occupancy grid map, a single region of interest was detected and its centroid was calculated and passed to the MPC path planner developed in Chapter 3. The MPC was successful at keeping the UAV in the tracking range of the estimated iceberg location for most of the time. However, some challenges regarding communication delay and having the MPC implemented off-board in the ground station was observed. Although not a major issue, this definitely emphasize the need to implement the MPC in a more robust way, e.g completely on-board, for ice management missions to be conducted in the future.

Although nominal testing indicates that the proposed framework is suitable for ice management scenarios, a more complete test where there are more icebergs and ice floes should be conducted in order to better assess the performance of the object detection and tracking framework as a UAV ice management platform.

CHAPTER 6

Concluding Remarks

This chapter provides the main conclusions inferred based on the work presented in this monograph. Furthermore, Chapter 6 includes possible avenues for future work for all aspects of the developed object detection and tracking framework.

6.1 Conclusion

The motivation for the thesis was to develop an object detection and tracking framework intended for use in UAVs. The focus was to make the framework as modularized as possible, in order to make the framework agile and adaptable. The reason for focusing on making the framework agile and adaptable was to enable the object detection and tracking system to be easily configured to accommodate the needs and prioritizations of a wide span of different scenarios, ranging from search and rescue operations to ice management applications.

Chapter 2 presented a relatively general and modularized object detection and tracking framework, consisting of a path planner module, a machine vision module and an object handler module. Each module solved a specific sub-problem of the overall object detection and tracking problem, and each module has a wide range of applicable solutions which could be implemented and mixed with the remainder of the system. The importance of Chapter 2 is how each module interacts with each other, and what type of data each module is required to supply to the remainder of the system regardless of the chosen solution for each module. Chapter 2 further described a software toolchain which provides a bridge and abstraction layer for the object detection and tracking framework's architecture and the UAV's onboard hardware components. This makes the software toolchain compatible with the idea of easily being able to configure the individual modules and hardware of the overall UAV object detection and tracking system. This is exemplified by

the implementation of the hardware and software modules of the object detection and tracking framework in two different UAV platforms. Although the two UAV platforms are not using the same autopilot system, the only change required in the software is the module responsible for reading the autopilots telemetry data and supplying the remainder of the system with said data, as well as receive command controls from other modules in the system and conveying these to the autopilot.

Chapter 3 develops a solution for the path planner module based on an MPC compatible with the object detection and tracking framework outlined in Chapter 2. The MPC supplies the on-board autopilot with waypoints and gimbal control input which enables the UAV to track an object or group of objects, and at the same time making sure the gimbal is pointing directly towards the tracked object's estimated position. The MPC is demonstrated through field experiments in Chapter 3 and 5 to be able to successfully make both of the UAV platforms developed in this thesis able to track objects. However, the results indicate the need for a gimbal controller which can stabilize the gimbal to keep pointing at a specific location on the ground in-between the MPC iterations. The developed MPC has a set of configurable parameters, which it is possible for a UAV operator to change online and in real-time. This ensures that also the implemented path planner module is adaptable and agile, making the operator able to adapt the path planner to events that arise during a UAV object detection and tracking scenario. An example of this is shown in the results of Chapter 3 where the MPC is used to track 5 stationary objects. During this experiment, the parameter telling the MPC for how long it should track each object is changed, and the MPC is then observed to spend less time tracking each of the objects that followed in the visitation sequence. Other parameters that could be changed online includes for instance the gimbal's responsiveness, i.e how quickly it can change attitude, and the ideal distance between the UAV to the tracked objects.

Chapter 4 developed a solution for the machine vision module as an object detection, recognition and tracking module compatible with the object detection and tracking framework. The module enables the UAV to perform on-board image analysis of thermal images in order to automatically segment the images into either background or objects of interest. Using the on-board telemetry data indicating the UAVs position and attitude coupled with the segmented images, the module is able to use a Kalman filter to track and estimate the position and velocity of the detected objects of interest. Furthermore, by using a global nearest neighbor algorithm for data association, the module is able to track several objects simultaneously. The data association also takes the three features (size, temperature and

shape) of objects in the thermal image into account, effectively giving the system recognition abilities. More specifically, this approach ensures that a tracked object is of a consistent type. I.e, a Kalman filter set to track a large boat will not confuse simultaneous detections of a small boat and a large boat. Furthermore, the object detection step and the object tracking step are separated in such a way that each step can be modified without changing the other. This enables a UAV data analyst to chose an object detection algorithm that is suitable for detecting the type of objects which is of interest to a specific operation, while not having to change the remaining parts of the module. The operator could also change the tracking algorithm from a Kalman filter to for instance a particle filter for better tracking of non-linearly moving objects, without having to change the object detection part of the module.

Chapter 5 introduces an alternative solution to the machine vision module by using an occupancy grid map instead of Kalman filters to keep track of objects of interest. In addition, it also introduces an alternative solution to the object handler module by generating locations of interest based on the occupancy grid map instead of the position estimates of the Kalman filter. The nominal tests of the occupancy grid map and the locations of interest generator described in Chapter 5 was successful, although not extensive enough to assess the effectiveness of the approach.

A full scale test of the object detection and tracking framework where all of the modules of the framework are activated and interacting autonomously as described in Chapter 2 is still a test that has to be conducted. However, the final flight test conducted in Chapter 5 is a test where all three modules (path planner, machine vision and object handler) interacts with each other. This flight test illustrates how the object detection and tracking framework enables the UAV to search a predefined region for objects of interest (machine vision), then deciding which location to track (object handler), and finally making the UAV track this location using the implemented MPC algorithm (path planner). This demonstrates the viability of the total system, and the only part of the system not tested is the feedback loop between the machine vision module and the path planner. That is, the feedback loop where the path planner controls the UAV in such a manner that the machine vision module can get a better estimate of an objects position and velocity, which in turn improves the path planner's ability to track said object.

6.2 Future Work

This section will cover some of the future perspectives for each part of the presented object detection and tracking framework and its modules. The section is or-

ganized in accordance with the structure of the monograph.

6.2.1 UAV System and Payload

For future developments of the UAV system and payload described in Chapter 2 the main focus should be to integrate as much of the object detection and tracking framework's functionality into the core software components, that is into the Dune-IMC-Neptus toolchain. One example is that the MPC GUI presented in Chapter 2 should be implemented as a plugin in Neptus instead of as a stand-alone software. This is important in order to keep the object detection and tracking system as agile and adaptable as possible, as using a stand-alone software for each possible path planner module is not scalable and is inconvenient.

The developed object detection and tracking system's usability could also be considerably increased for the UAV operator and data analyst. It would for instance be highly beneficial to the UAV data analyst if the developed plugin in Neptus that shows the location of detected objects was extended to also include example footage of these objects. Further, this plugin could also be expanded to enable the UAV data analyst to remove objects that are no longer of interest from the list of objects being tracked. The possibility of manually adding new object locations, or possibly just locations of interest that should be included in the tracking process would further expand the usefulness of the system. This would enable the UAV data analyst to influence the object detection and tracking process, instead of being a passive observer once the pre-flight configuration of the object detection and tracking framework's modules was completed. Furthermore, having the possibility of the UAV data analyst to intervene if the object detection and tracking system does something wrong (e.g begins to track a location that does not contain an object of interest) is important since most object detection and tracking systems cannot be expected to operate flawlessly.

6.2.2 Path Planner

For future work on the developed MPC path planner, a high priority is to make the MPC completely on-board. The motivation for this is two-folded. First, it will remove the time delays caused by having to communicate telemetry data to the ground station and sending control inputs from the ground station to the autopilot. Second, and more importantly, it will allow the UAV to continue the object following and tracking process, fully autonomously, even in the event of (temporarily) losing communication with the ground station. This greatly extends the range and usability of the system. Another priority will be to improve computa-

tional efficiency of the optimization, that is, from the field tests it was apparent that the time needed to calculate the optimal control input directly affected the UAV's object tracking performance. Hence, an important part of improving the framework's tracking performance will be to minimize the time the on-board computer needs in order to calculate the next control input. Traditionally, this would be solved by using a computer with enough processing power. However, because of the weight and power limitations of the UAV, the computational power will be limited. Hence, improving the MPC algorithm is a more viable solution. In the future, the on-board computer should employ multi-core computations in order to reduce the time needed to find the next optimal control input. Another option to reduce the required computation time is to let the MPC calculate only the optimal tracking path, and employ a simpler gimbal control algorithm instead of using the MPC. This algorithm could for instance be as simple as letting the gimbal always point towards the object closest to the UAV.

The MPC's strength is that it is easily configurable through the modification of its cost function, hence it can be adapted to satisfy different requirements for different UAV applications. However, since the MPC could end up being challenging to implement completely on-board, other types of path planners could also be considered to be used with the object detection and tracking framework. It could for instance be interesting to evaluate methods that combine the object handler and path planner modules into one single algorithm. For this to be the case, the path planner has to be able to consider all detected objects simultaneously, and still be able to generate a tracking pattern that makes sense on an object-to-object basis.

The field of artificial intelligence also has some interesting developments that could be useful, not only in the tracking process but also in the work of incorporating multiple UAVs and/or a search component in the system. Being able to incorporate a search component in the path planner would enable the UAV object detection and tracking system to facilitate a larger set of applications. Using multiple UAVs would also greatly increase the effectiveness of a UAV object detection and tracking system.

6.2.3 Object Detection, Recognition and Tracking

To further improve on the object detection, recognition and tracking module developed in Chapter 4, a high priority is to make the object detection algorithm able to self-tune. That is, instead of setting the threshold T_g and the minimum object size manually, this should be set automatically or dynamically. The minimum object size could be based on the operator's choice for what type of objects that

should be detected and tracked. Furthermore, this could be further improved by adjusting the minimum object size based on the UAV's altitude, as a boat of a given size will for instance appear smaller the higher the UAV is flying. The threshold T_g could either be set on-line based on an initial calibration phase, or potentially chosen to be dynamically set based on the current scene.

From the results presented in Chapter 4 it is also readily seen that the tracking performance would benefit from improving the technique to calculate the centroid of an object when the object is only partly visible. As of now, the centroid is calculated as the center of the part of the object that is visible, but by combining classification of objects and estimation of object orientation it will be possible to find a more accurate measurement of the object's centroid. This will improve the accuracy of the tracking process significantly for big objects when using image data where only parts of an object is visible. Another priority will be to include on-line and real-time adjustments of the Gaussian white noise variables found in the motion model for the Kalman filter. Adjusting the measurement noise model based on the UAV's altitude, as well as choosing the motion model noise based on what type of object the UAV is tracking would make the tracking process more efficient and better at estimating the true position and velocity of the objects being tracked.

Another avenue to research is the use of other position and velocity estimators in the machine vision module. [49] investigates the use of a machine vision algorithm that generates a measurement of not only an object's position, but also its velocity. This is an alternative measurement which can be used directly with, or possibly in conjunction with, the Kalman filter developed in Chapter 4 and might yield a better tracking performance. Other avenues to research is for instance the use of an extended Kalman filter or a particle filter for estimation purposes, especially for the smaller boats which are often moving in a non-linear pattern.

6.2.4 Ice Management Application

To further improve the object detection and tracking framework's performance in a realistic ice management scenario, a more sophisticated way to generate the locations of interest for the path planner should be investigated. More specifically, a way to handle large regions of connected grid cells which has a high probability of being occupied has to be developed, since a single location of interest will not suffice to cover the whole region in these cases. Furthermore, a more realistic partial inverse measurement model should be established based on gathered thermal video footage of icebergs and ice floes. Both of these additions to the system will

increase the system's overall effectiveness in an ice management scenario.

To further extend the usefulness of the object detection and tracking framework in a sea ice management setting, current models generated by analysing SAR satellite data and weather data should also be incorporated in the update algorithm for the occupancy grid map. Furthermore, functionality that stores the occupancy grid map generated during a UAV operation in the ground control station is useful, as this enables the ground station to continue to update the occupancy grid map post-flight. The occupancy grid map would then be updated according to the ground station's estimated map dynamics (based on the current model). For the next UAV operation, the UAV could then be soft-started with the occupancy grid map stored in the ground station in order to make the UAV visit the regions which are more likely to contain icebergs and ice floes of interest first.

APPENDIX A

Autopilots

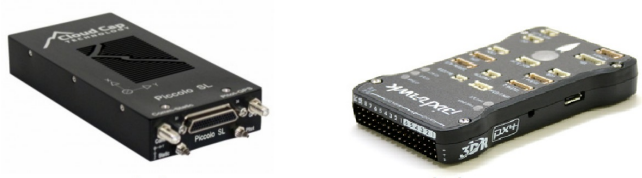

Figure A.1: The CloudCap Piccolo (a) and the Pixhawk PX4 (b) autopilots

Overview	**CloudCap Piccolo Autopilot**
Payload Interface	RS232
General Interface	CAN Bus
Digital/Analog I/O	14 configurable GPIO lines
Data Link	Integrated radio frequency data link ($900 - 1700$MHz)
Sensors	GPS, Pressure Sensor, Intertial Sensors
Physical Attributes	
Weight	110g
Dimensions	$13.1 \times 5.7 \times 1.9 cm^3$
Envoirmental and Power	
Operating Conditions	$-40^\circ - 80^\circ$ C
Power	$4.5 - 28$V (4W)

Overview	**Pixhawk PX4 Autopilot**
Payload Interface	UART or microUSB
General Interface	CAN and I2C Bus
Digital/Analog I/O	14 configurable GPIO lines
Data Link	External data link
Sensors	Gyroscope, Accelerometer, Barometer
Physical Attributes	
Weight	38g
Dimensions	$15.5 \times 8.1 \times 5.0 cm^3$
Envoirmental and Power	
Operating Conditions	$-40^\circ - 80^\circ$ C
Power	$4.8 - 5.4$V (4W)

APPENDIX B

Thermal Imaging Camera

Figure B.1: The FLIR Tau2 Thermal Imager

Overview	**FLIR Tau 640**
Analog Video Display Formats	640 x 480 (NTSC); 640 x 512 (PAL)
Pixel Pitch	$17\mu m$
Spectral Band	$7.5 - 13.5\mu m$ (LWIR)
Frame Rates	7.5 Hz NTSC; 8.3 Hz PAL
Sensitivity	$\leq$ 50 mK at f/1.0
Scene Range	-25°C to +100°C
Physical Attributes	
Size (w/o lens)	1.75" x 1.75" x 1.18"
Weight	72g
Interfacing	
Input Power	4.0 - 6.0 VDC
Power Dissipation, steady state	$\sim$ 1.2 W
Environmental	
Operating Temperature Range	-40°C to +80°C
Temperature Shock (5°/min)	Yes
Operational Altitude	Up to 12km
Humidity	Non-condensing between 5% - 95%
Vibration	4.3g three axis, 8 hr each
Shock	200g shock pulse w/ 11 msec sawtooth

Overview	**FLIR Tau 336**
Analog Video Display Formats	640 x 480 (NTSC); 640 x 512 (PAL) (Up-sampled from 336×256)
Pixel Pitch	$17\mu m$
Spectral Band	$7.5 - 13.5\mu m$ (LWIR)
Frame Rates	7.5 Hz NTSC; 8.3 Hz PAL
Sensitivity	$\leq$ 50 mK at f/1.0
Scene Range	-25°C to +100°C
Physical Attributes	
Size (w/o lens)	1.75" x 1.75" x 1.18"
Weight	72g
Interfacing	
Input Power	4.0 - 6.0 VDC
Power Dissipation, steady state	$\leq$ 1.0 W
Environmental	
Operating Temperature Range	-40°C to +80°C
Temperature Shock (5°/min)	Yes
Operational Altitude	Up to 12km
Humidity	Non-condensing between 5% - 95%
Vibration	4.3g three axis, 8 hr each
Shock	200g shock pulse w/ 11 msec sawtooth

APPENDIX C

Retractable Gimbal

Figure C.1: The R-BTC88 Retractable Gimbal

Overview	**Retractable BTC-88**
Degrees of Freedom	2 (pan and tilt)
Azimuth Range	360° ($180^\circ, -180^\circ$)
Elevation Range	80°
Gimbal Speed	$\geq 200^\circ$/second
Physical Attributes	
Weight	275g
Dimensions	$12.3 \times 8.9 \times 9.1 cm^3$
Interfacing	
Power	5V
Connectors (Input)	Servo input (PWM)
Compatibility	FLIR Tau2 and Sony FCB-EX11D Camera
Envoirmental	
Operating Conditions	$-30^\circ - 80^\circ$

APPENDIX D

Frame Grabber

Figure D.1: The Axis M7001 frame grabber

Overview	**Axis M7001**
Video Compression	H.264 or MJPEG
Resolutions	720×576 to 76×144
Frame Rate	30/25 (NTSC/PAL)
Video Streaming	Both H.264 and MJPEG
Supported Protocols	IPv4/v6, HTTP, HTTPS, FTP, SMTP, UPnP, TCP, UDP, RTCP, DHCP ++
Physical Attributes	
Weight	82g
Dimensions	$10,1 \times 8.6 \times 3.7cm^3$
Interfacing	
Power	Power over Ethernet (Class $2 \sim 5W$)
Connectors (Input)	Analog composite video (BNC input)
Connectors (Output)	Ethernet (RJ45)
Envoirmental	
Operating Conditions	$0° - 50°$ C
Humidity	20-80%

APPENDIX E

Single Board Computers

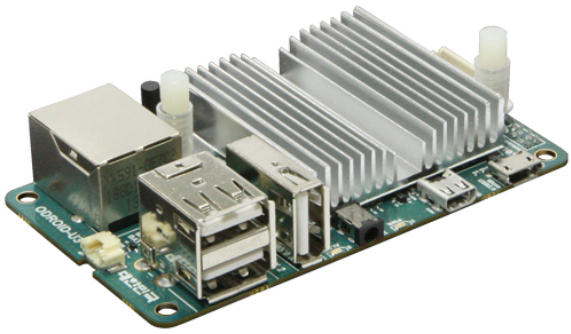

Figure E.1: The ODROID-U3.

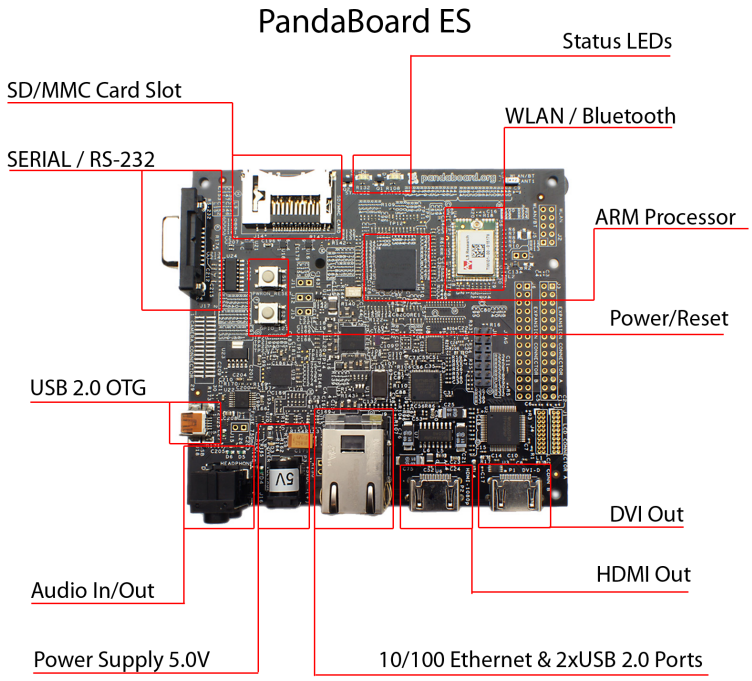

Figure E.2: The PandaBoard and its most important specifications.

Overview	**PandaBoard ES**
CPU	1.2GHz Dual Core
Processor Type	ARM Cortext-A9
RAM	1GB
GPU	304 MHz PowerVR SGX540
Physical Attributes	
Size	$11.4cm \times 10.1cm \times 3cm$
Weight	82g
Interfacing	
Display Port	HDMI
Power	DC Jack / USB OTG - 5V $\sim$ 1A
Storage	SD Card
Ethernet	RJ45 and WiFi
Environment	
Operating Temperature	$-20° - 70°$
Humidity	N/A

Overview	**ODROID-U3**
CPU	1.7GHz Quad Core
Processor Type	ARM Cortext-A9
RAM	2GB
GPU	Mali-400 Quad Core 440MHz
Physical Attributes	
Size	$8.3cm \times 4.8cm \times 2cm$
Weight	48g
Interfacing	
Display Port	Micro-HDMI
Power	DC Jack - 5V $\sim$ 2A
Storage	Micro SD, eMMC
Ethernet	RJ45
Environment	
Operating Temperature	$-20° - 70°$
Humidity	N/A

Bibliography

[1] Axis m7001, 2011. http://www.axis.com/products/cam_m7001/.

[2] Flir tau2 640, 2011. http://www.flir.com/cores/display/?id=54717.

[3] Pandaboard es, 2011. http://www.pandaboard.org/.

[4] Retractable BTC-88, 2011. http://microuav.com/btc-88.html.

[5] Ubiquiti rocket m5, 2011. http://www.ubnt.com/airmax/rocketm/.

[6] ODROID U3, 2015. http://www.hardkernel.com/main/products/prdt_info.php?g_code=g138745696275.

[7] APM autopilot suite, 2016. http://www.ardupilot.com/.

[8] APM planner 2, 2016. http://ardupilot.org/planner2/.

[9] Cloudcap piccolo autopilot, 2016. http://www.cloudcaptech.com/products/auto-pilots/.

[10] Penguin b UAV platform, 2016. http://www.uavfactory.com/.

[11] Pixhawk PX4, 2016. https://pixhawk.org/modules/pixhawk.

[12] X8 skywalker, 2016. http://www.airelectronics.es/products/solutions/x8/.

[13] A. P. Aguiar, J. P. Hespanha, et al. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE Transactions on Automatic Control*, 52(8):1362–1379, 2007.

[14] A. Albert and L. Imsland. Mobile sensor path planning for iceberg monitoring using a MILP framework. In *Informatics in Control, Automation and Robotics (ICINCO), IEEE 12th International Conference on*, volume 1, pages 131–138, 2015.

[15] A. Albert and L. Imsland. Mobile sensor path planning for iceberg monitoring using a milp framework. In *Informatics in Control, Automation and Robotics (ICINCO), IEEE 12th International Conference on*, volume 1, pages 131–138, 2015.

[16] A. Ali and K. Terada. A general framework for multi-human tracking using Kalman filter and fast mean shift algorithms. *Journal of Universal Computer Science*, 16(6):921–937, mar 2010.

[17] P. P. Angelov, C. Gude, P. Sadeghi-Tehran, and T. Ivanov. ARTOT: autonomous real-time object detection and tracking by a moving camera. In *6th IEEE International Conference on Intelligent Systems, IS 2012, Sofia, Bulgaria, September 6-8, 2012*, pages 446–452, 2012.

[18] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The traveling salesman problem: a computational study*. Princeton university press, 2011.

[19] S. S. Baek, H. Kwon, J. A. Yoder, and D. Pack. Optimal path planning of a target-following fixed-wing UAV using sequential decision processes. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 2955–2962, 2013.

[20] D. B. Barber, J. D. Redding, T. W. McLain, R. W. Beard, and C. N. Taylor. Vision-based target geo-location using a fixed-wing miniature air vehicle. *Journal of Intelligent and Robotic Systems*, 47(4):361–382, 2006.

[21] R. W. Beard and T. W. McLain. *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

[22] A. Benghezal, R. Louali, A. Bazoula, and T. Chettibi. Trajectory generation for a fixed-wing UAV by the potential field method. In *Control, Engineering & Information Technology (CEIT), IEEE 3rd International Conference on*, pages 1–6, 2015.

[23] L. T. Biegler. *Nonlinear programming : concepts, algorithms and applications to chemical processes*. MOS-SIAM series on optimization. Society for industrial and applied mathematics, Philadelphia, 2010.

[24] F. Bourgeois and J.-C. Lassalle. An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Commun. ACM*, 14(12):802–804, Dec. 1971.

[25] G. Bradski. *Dr. Dobb's Journal of Software Tools*, 2000.

[26] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. 1998.

[27] D. C. Brown. Decentering distortion of lenses. *Photometric Engineering*, 32(3):444–462, 1966.

[28] B. Cyganek. Object detection and recognition in digital images: theory and practice, 2013.

[29] V. N. Dobrokhodov, I. I. Kaminer, K. D. Jones, and R. Ghabcheloo. Vision-based tracking and motion estimation for moving targets using small UAVs. In *IEEE American Control Conference*, pages 6–pp, 2006.

[30] V. N. Dobrokhodov, I. I. Kaminer, K. D. Jones, and R. Ghabcheloo. Vision-based tracking and motion estimation for moving targets using small UAVs. In *IEEE American Control Conference*, pages 6–pp, 2006.

[31] K. Eik. Review of experiences within ice and iceberg management. *Journal of Navigation*, 61(04):557–572, 2008.

[32] T. Eltoft, A. P. Doulgeris, C. Brekke, S. Solbø, S. Gerland, and A. Hanssen. Imaging sea ice structure by remote sensing sensors. Proceedings of the 23rd International Conference on Port and Ocean Engineering under Arctic Conditions June 14-18, 2015 Trondheim, Norway, 2015.

[33] J. Everaerts et al. The use of unmanned aerial vehicles (UAVs) for remote sensing and mapping. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37:1187–1192, 2008.

[34] M. Faria, J. Pinto, F. Py, J. Fortuna, H. Dias, R. Martins, F. S. Leira, T. A. Johansen, J. Sousa, and K. Rajan. Coordinating UAVs and AUVs for oceanographic field experiments: Challenges and lessons learned. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6606–6611, 2014.

[35] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. qpOASES: a parametric active-set algorithm for quadratic programming. *Math. Program. Comput.*, 6(4):327–363, 2014.

[36] A. L. Flåten. Experimental monitoring of sea ice using unmanned aerial systems, June 2015. Master Thesis, NTNU, Department of Engineering Cybernetics.

[37] J. Flusser. Moment invariants in image analysis. *World Academy of Science, Engineering and Technology*, 11:376–381, 2005.

[38] E. J. Forsmo, T. I. Fossen, T. A. Johansen, et al. Optimal search mission with unmanned aerial vehicles using mixed integer linear programming. In *Unmanned Aircraft Systems (ICUAS), IEEE International Conference on*, pages 253–259, 2013.

[39] T. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 2011.

[40] A. Gaszczak, T. Breckon, and J. Han. Real-time people and vehicle detection from UAV imagery. In *Proc. SPIE Conference Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*, volume 7878, 2011.

[41] E. I. Grøtli and T. A. Johansen. Path planning for UAVs under communication constraints using SPLAT! and MILP. *Journal of Intelligent and Robotic Systems*, 65(1-4):265–282, 2012.

[42] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control*. Springer London, London, 2011.

[43] C. Hanson, J. Richardson, and A. Girard. Path planning of a Dubins vehicle for sequential target observation with ranged sensors. In *IEEE American Control Conference (ACC)*, pages 1698–1703, 2011.

[44] J. Haugen. Autonomous aerial ice observation. 2014. Doctoral Thesis, NTNU, Department of Engineering Cybernetics.

[45] J. Haugen and L. Imsland. Monitoring moving objects using aerial mobile sensors. *IEEE Transactions on Control Systems Technology*, 24(2):475–486, 2016.

[46] J. Haugen, L. Imsland, S. Løset, R. Skjetne, et al. Ice observer system for ice management operations. In *The Twenty-first International Offshore and Polar Engineering Conference*. International Society of Offshore and Polar Engineers, 2011.

[47] Z. He and J.-X. Xu. Moving target tracking by UAVs in an urban area. In *Control and Automation (ICCA), 10th IEEE International Conference on*, pages 1933–1938, 2013.

[48] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, pages 1106–, Washington, DC, USA, 1997. IEEE Computer Society.

[49] H. H. Helgesen, F. S. Leira, T. A. Johansen, and T. I. Fossen. Tracking of marine surface objects from unmanned aerial vehicles with a pan/tilt unit using a thermal camera and optical flow. In *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 107–117, 2016.

[50] E. M. Hemerly. Automatic georeferencing of images acquired by uavâĂŹs. *International Journal of Automation and Computing*, 11(4):347–352, 2014.

[51] G. Hendeby and R. Karlsson. Target tracking performance evaluation-a general software environment for filtering. In *IEEE Aerospace Conference*, pages 1–13, 2007.

[52] B. Houska, H. Ferreau, M. Vukov, and R. Quirynen. ACADO Toolkit User's Manual. http://www.acadotoolkit.org, 2009–2013. Last accessed: 2015-03-6.

[53] V. E. Hovstein, A. Sægrov, and T. A. Johansen. Experiences with coastal and maritime uas blos operation with phased-array antenna digital payload data link. In *Unmanned Aircraft Systems (ICUAS), IEEE International Conference on*, pages 261–266, 2014.

[54] M.-K. Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, February 1962.

[55] J. Johnson. Analysis of image forming systems. In *Selected papers on infrared design. Part I and II*, volume 513, page 761, 1985.

[56] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *Pattern Analysis and Machine Intelligence*, 2012.

[57] A. T. Klesh, P. T. Kabamba, and A. R. Girard. Path planning for cooperative time-optimal information collection. In *American Control Conference*, pages 1991–1996, 2008.

[58] P. Konstantinova, A. Udvarev, and T. Semerdjiev. A study of a target tracking algorithm using global nearest neighbor approach. In *Proceedings of the 4th International Conference Conference on Computer Systems and Technologies: E-Learning*, CompSysTech '03, pages 290–295, New York, NY, USA, 2003. ACM.

[59] F. S. Leira. Infrared object detection & tracking in UAVs, June 2013. Master Thesis, NTNU, Department of Engineering Cybernetics.

[60] F. S. Leira, T. I. Fossen, and T. A. Johansen. Object detection, recognition and tracking from UAVs using a thermal camera. *Journal of Field Robotics*, 2017.

[61] F. S. Leira, T. I. Fossen, and T. A. Johansen. A UAV ice tracking framework for sea ice management. *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017.

[62] F. S. Leira, T. A. Johansen, and T. I. Fossen. Automatic detection, classification and tracking of objects in the ocean surface from UAVs using a thermal camera. In *Proceedings of the IEEE Aerospace Conference*, pages 1–10, 2015.

[63] F. S. Leira, E. Skjong, S. A. Nundal, T. A. Johansen, and T. I. Fossen. Autonomous unmanned aerial vehicle object tracking framework using

model predictive control and camera gimbal. *AIAA Journal of Guidance, Control, and Dynamics*, 2017.

[64] F. S. Leira, K. Trnka, T. I. Fossen, and T. A. Johansen. A ligth-weight thermal camera payload with georeferencing capabilities for small fixed-wing UAVs. In *Proceedings of the IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 485–494, 2015.

[65] I. Lesinskis and A. Pavlovics. Monitoring of ice conditions in the gulf of riga using micro class unmanned aerial systems. *Marine Navigation and Safety of Sea Transportation: Navigational Problems*, page 167, 2013.

[66] S. Leven, J.-C. Zufferey, and D. Floreano. A minimalist control strategy for small UAVs. In *International Conference on Intelligent Robots and Systems (IROS). IEEE/RSJ*, pages 2873–2878, 2009.

[67] J. R. Looker. Minimum paths to interception of a moving target when constrained by turning radius. Technical Report DSTO-TR-2227, Defence Science and Technology Organisation, Canberra, Australia, 2008.

[68] R. Martins, P. Dias, E. R. B. Marques, J. Pinto, J. Sousa, and F. L. Pereira. IMC: A communication protocol for networked vehicles and sensors. In *OCEANS 2009-EUROPE*. IEEE, IEEE, 2009.

[69] S. Muckenhuber, A. A. Korosov, and S. Sandven. Open-source feature-tracking algorithm for sea ice drift retrieval from sentinel-1 sar imagery. *The Cryosphere*, 10(2):913–925, 2016.

[70] National Imagery and Mapping Agency. Department of defense world geodetic system 1984: its definition and relationships with local geodetic systems. Technical Report TR8350.2, National Imagery and Mapping Agency, Jan. 2000.

[71] P. C. Niedfeldt and R. W. Beard. Recursive RANSAC: Multiple signal estimation with outliers. *IFAC Proceedings Volumes*, 46(23):430–435, 2013.

[72] P. C. Niedfeldt and R. W. Beard. Multiple target tracking using recursive ransac. In *IEEE American Control Conference*, pages 3393–3398, 2014.

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

[74] K. Nonami. Prospect and recent research & development for civil use autonomous unmanned aircraft as UAV and MAV. *Journal of system Design and Dynamics*, 1(2):120–128, 2007.

[75] H. Oh, S. Kim, H.-s. Shin, and A. Tsourdos. Coordinated standoff tracking of moving target groups using multiple UAVs. *IEEE Transactions on Aerospace and Electronic Systems*, 51(2):1501–1514, 2015.

[76] J. Osborne and R. Rysdyk. Waypoint guidance for small UAVs in wind. *AIAA Infotech@ Aerospace*, 193(1-4):1–12, 2005.

[77] J. Pestana, J. L. Sanchez-Lopez, S. Saripalli, and P. Campoy. Computer vision based general object following for GPS-denied multirotor unmanned vehicles. In *IEEE American Control Conference*, pages 1886–1891, 2014.

[78] J. Pinto, P. Calado, J. Braga, P. Dias, R. Martins, E. Marques, and J. Sousa. Implementation of a control architecture for networked vehicle systems. In *Proceedings of the IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles*. IFAC, 2012.

[79] J. Pinto, P. S. Dias, R. Gonçalves, E. R. B. Marques, G. M. Gonçalves, J. B. Sousa, and F. L. Pereira. Neptus: a framework to support a mission life cycle. In *Proc. IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC)*. IFAC, 2006.

[80] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. Marques, and J. Sousa. The lsts toolchain for networked vehicle systems. In *OCEANS-Bergen, 2013 MTS/IEEE*, pages 1–9, 2013.

[81] C. G. Prevost, A. Desbiens, and E. Gagnon. Extended Kalman filter for state estimation and trajectory prediction of a moving object detected by an unmanned aerial vehicle. In *IEEE American Control Conference*, pages 1805–1810, 2007.

[82] C. G. Prévost, O. Thériault, A. Desbiens, E. Poulin, and E. Gagnon. Receding horizon model-based predictive control for dynamic target tracking: a comparative study. In *AIAA Guidance, Navigation, and Control Conference, Chicago, Illinois, USA*, 2009.

[83] A. Qadir, J. Neubert, and W. Semke. On-board visual tracking with unmanned aircraft system (UAS). In *Proc. of AIAA Infotech@ Aerospace*. St. Louis, Missouri, USA. St. Louis, Missouri, USA., 2011.

[84] M. Quigley, M. A. Goodrich, S. Griffiths, A. Eldredge, and R. W. Beard. Target acquisition, localization, and surveillance using a fixed-wing mini-UAV and gimbaled camera. In *Robotics and Automation (ICRA). Proceedings of the IEEE International Conference on*, pages 2600–2605, 2005.

[85] F. Rafi, S. Khan, K. Shafiq, and M. Shah. Autonomous target following by unmanned aerial vehicles. In *Defense and Security Symposium*, pages 623010–623010. International Society for Optics and Photonics, 2006.

[86] S. Ragi and E. K. Chong. UAV path planning in a dynamic environment via partially observable Markov decision process. *IEEE Transactions on Aerospace and Electronic Systems*, 49(4):2397–2412, 2013.

[87] J. R. Riehl, G. E. Collins, and J. P. Hespanha. Cooperative search by UAV teams: A model predictive approach using dynamic graphs. *IEEE Transactions on Aerospace and Electronic Systems*, 47(4):2637–2656, 2011.

[88] G. R. Rodríguez-Canosa, S. Thomas, J. del Cerro, A. Barrientos, and B. MacDonald. A real-time method to detect and track moving objects (DATMO) from unmanned aerial vehicles (UAVs) using a single camera. *Remote Sensing*, 4(4):1090, 2012.

[89] E. Roy. System integration of unmanned aerial vehicle with thermal camera, January 2016. Master Thesis, NTNU, Department of Engineering Cybernetics.

[90] J. C. Ryan, A. L. Hubbard, J. E. Box, J. Todd, P. Christoffersen, J. R. Carr, T. O. Holt, and N. Snooke. UAV photogrammetry and structure from motion to assess calving dynamics at store glacier, a large outlet draining the greenland ice sheet. *The Cryosphere*, 9(1):1–11, 2015.

[91] R. Rysdyk. Unmanned aerial vehicle path following for target observation in wind. *Journal of guidance, control, and dynamics*, 29(5):1092–1100, 2006.

[92] R. Sengupta, J. Connors, B. Kehoe, Z. Kim, T. Kuhn, and J. Wood. Final report - autonomous search and rescue with scaneagle. September 2010.

[93] M. Shah, A. Hakeem, and A. Basharat. Detection and tracking of objects from multiple airborne cameras. *SPIE Newsroom*, 2006.

[94] H. Sheng, H. Chao, C. Coopmans, J. Han, M. McKee, and Y. Chen. Low-cost UAV-based thermal infrared remote sensing: platform, calibration and applications. In *Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on*, pages 38–43, 2010.

[95] E. Skjong and S. A. Nundal. Tracking objects with fixed-wing UAV using model predictive control and machine vision, 2014. Master Thesis, NTNU, Department of Engineering Cybernetics.

[96] E. Skjong, S. A. Nundal, F. S. Leira, and T. A. Johansen. Autonomous search and tracking of objects using model predictive control of unmanned aerial vehicle and gimbal: Hardware-in-the-loop simulation of payload and

avionics. In *Proceedings of the IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 904–913, 2015.

[97] P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2 edition, 2003.

[98] A. Stalmakou. UAV/UAS path planning for ice management information gathering, June 2011. Master Thesis, NTNU, Department of Engineering Cybernetics.

[99] G. P. Stein. Lens distortion calibration using point correspondences. In *CVPR*, pages 602–608. IEEE Computer Society, 1997.

[100] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[101] C. Teuliere, L. Eck, and E. Marchand. Chasing a moving target from a flying UAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4929–4934, 2011.

[102] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[103] A. Tsourdos, B. White, and M. Shanmugavel. *Cooperative path planning of unmanned aerial vehicles*, volume 32. John Wiley & Sons, 2010.

[104] M. Vollmer and K.-P. Möllmann. *Infrared thermal imaging: fundamentals, research and applications*. John Wiley & Sons, 2010.

[105] M. Westoby, J. Brasington, N. Glasser, M. Hambrey, and J. Reynolds. Structure-from-motion photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, 179:300–314, 2012.

[106] Z. Wu, A. Thangali, S. Sclaroff, and M. Betke. Coupling detection and data association for multiple object tracking. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1948–1955, 2012.

[107] H. Xiang and L. Tian. Method for automatic georeferencing aerial remote sensing (RS) images from an unmanned aerial vehicle (UAV) platform. *Biosystems Engineering*, 108(2):104 – 113, 2011.

[108] A. Zolich, T. A. Johansen, K. Cisek, and K. Klausen. Unmanned aerial system architecture for maritime missions. design & hardware description. In *IEEE Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 342–350, 2015.