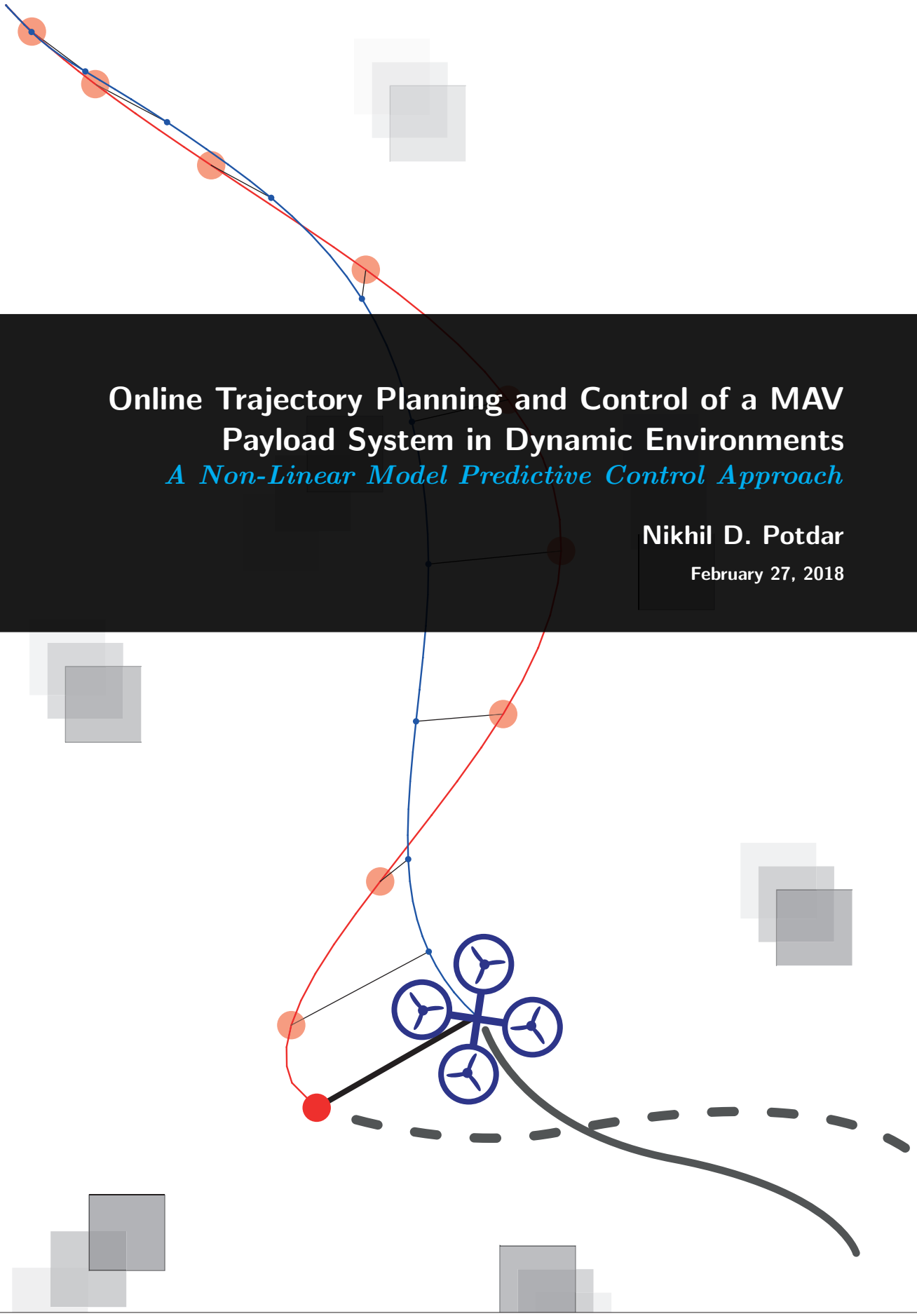
Faculty of Aerospace Engineering



Online Trajectory Planning and Control of a MAV Payload System in Dynamic Environments

A Non-Linear Model Predictive Control Approach

Nikhil D. Potdar

February 27, 2018

TU Delft Delft University of Technology

Challenge the future

Online Trajectory Planning and Control of a MAV Payload System in Dynamic Environments

A Non-Linear Model Predictive Control Approach

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

Nikhil D. Potdar

February 27, 2018

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Online Trajectory Planning and Control of a MAV Payload System in Dynamic Environments"** by **Nikhil D. Potdar** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: February 27, 2018

Readers:

Supervisor — Dr.ir. G. C. H. E. de Croon

Supervisor — Dr. J. Alonso-Mora

Committee — Dr.ir. C. C. de Visser

Preface

This thesis presents the culmination of my Masters of Aerospace Engineering studies at the Delft University of Technology. Reflecting back on the past few years, it has been a very demanding and challenging degree, however, this has made the study an exciting and stimulating experience. I am thankful for the acquaintances and friends I have made along the way and who have stood by me at every step.

I would like to extend my gratitude to my thesis supervisors Dr. Guido de Croon and Dr. Javier Alonso-Mora for giving me this opportunity to pursue my topic of interest, and for taking the time to support my learning process. Our fruitful discussions and meetings have been invaluable throughout the project and have given me insights into the field and research that only come with your combined years of experience. Your encouragement and help has motivated me to push the envelope throughout the thesis and I am proud of the outcome.

Furthermore, this thesis would not have been possible without the help of fellow students who have assisted me during the research with setting up and performing experiments; their help and input has been appreciated. I would also like to thank Tobias Naegeli at ETH Zürich for providing technical assistance in regard to getting the experimental code functional and hardware related matters. Also, I appreciate the support and cooperation received from staff at *embotech* who kindly arranged an academic license for the optimiser software used as part of this study.

Finally, I would like to thank my family and friends who have supported me throughout the thesis. To my parents and brother who have been pillars of strength to look up to and always been there through thick and thin. To my friends who have been a source of inspiration and motivation while being there for the lighter moments.

Thank you, and till our paths cross again.

Nikhil Potdar
Delft, February 27, 2018

Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
APF	Artificial Potential Field
CCC	Command, Control and Communication
ENU	East, North, Up reference frame
EOM	Equations of Motion
GUI	Graphical User Interface
KF	Kalman Filter
KKT	Karush-Kuhn-Tucker conditions
LBSI	Learning Based System Identification
LCP	Linear Complementarity Problem
LKF	Linear Kalman Filter
LQE	Linear Quadratic Estimator
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
LTI	Linear Time Invariant
LTV	Linear Time Variant
MAV	Micro Aerial Vehicle
MAVP	Micro Aerial Vehicle with swung Payload
MCS	Motion Capture System
NMPC	(Non-linear) Model Predictive Control
NRMSE	Normalised Root Mean Square Error
OCP	Optimal Control Problem
PID	Proportional Integral Derivative control
RHC	Receding Horizon Control
RMSE	Root Mean Square Error
ROS	Robotics Operating System

SDK	Software Development Kit
SMC	Switching Mode Control
UAV	Unmanned Aerial Vehicle
UAVP	Unmanned Aerial Vehicle with swung Payload
UKF	Unscented Kalman Filter

List of Symbols

Greek Symbols

$\phi,\ \phi_Q$	Roll angle of quadrotor (Euler ZYX convention)
ϕ_L	Payload suspension angle with $\{S\}$ frame y-axis of rotation
$\psi,\ \psi_Q$	Yaw angle of quadrotor (Euler ZYX convention)
$\theta,\ \theta_Q$	Pitch angle of quadrotor (Euler ZYX convention)
θ_L	Payload suspension angle with $\{S\}$ frame x-axis of rotation

Roman Symbols

A	State-space state matrix
B	State-space input matrix
C	State-space output matrix; Coriolis term matrix
c	Cost term for optimisation
D	State-space feed-forward matrix
e	Output error or error signal
F	Arbitrary (non-)linear function; Force vector
f	Arbitrary (non-)linear function; Force vector
G	Gravity term matrix
g	Nominal sea-level gravitational acceleration
H	Observation function for Kalman Filtering
J	Cost/Objective function for optimisation
K	Kinetic energy
k	Constant term; Time-step index

L	Lagrangian
l	Suspension cable length
M	Mass term matrix
m	Mass
N	Number of optimisation stages/horizon length
P	Potential energy
p	Position in Cartesian system
Q	Generalised force/torque term matrix; Process noise covariance
q	System configuration vector
R	Rotational transformation matrix; Observation noise covariance
r	Arbitrary vector in Cartesian system
S	Surface object
s	Slack for constrained optimisation
T	Thrust force
t	Time
Δt	Time-step
u	Input
V	Velocity
w	Cost weighting for optimisation
x	State; Cartesian x position
y	Output; Cartesian y position
z	Measurement; Cartesian z position

Special Symbols

$\{B\}$	Vehicle body frame
$\{E\}$	Vehicle-carried normal East-North-Up reference frame
$\{I\}$	Inertial East-North-Up reference frame
$\{L\}$	Payload reference frame
$\{S\}$	Rotated vehicle-carried East-North-Up reference frame
$\mathcal{U}$	Input space set
$\mathcal{X}$	State space set

Contents

Chapter 1

Introduction

In recent years, research and development into Micro Aerial Vehicles (MAVs) from the research community and industry alike has flourished. This is exemplified by the number of articles published related to the topic of MAVs as shown Figure 1-1.

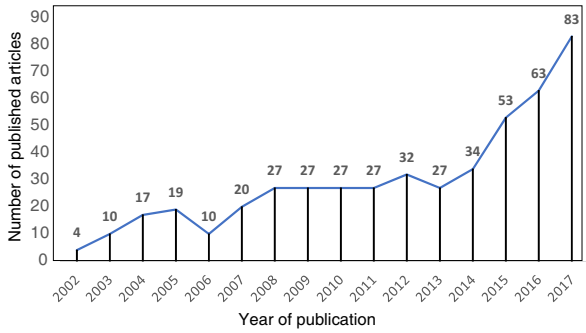


Figure 1-1: Number of published articles containing the keywords 'UAV' or 'MAV' or their full forms from 2002 to 2017 (Data obtained from Scopus on February 16, 2018)

The MAV is a disruptive technology that can bring about operational cost savings, enhanced functionality and outperform classical approaches to tasks including payload transportation (The Economist, 2012). The relatively small size, agility, ease of operation and low costs of mass-produced MAVs has led to their ubiquity in society as a consumer electronic. Those same qualities that make MAVs valuable are now being exploited for payload transport known as MAV Payload (MAVP) systems. These systems have the potential to be rapidly deployable and operable in situations where items need to be brought to areas that are inaccessible or dangerous for humans and conventional (aerial) vehicles. Current applications have seen MAVPs being used in search and rescue (Ryan & Hedrick, 2005), package/cargo delivery and even construction (T. Lee, 2018). A few exemplary situations where MAVPs could be useful include;

- delivering first-aid in flooded cities where ground vehicles can no longer operate,

- hoisting material up to the construction level of a structure,
- transporting small or even bulky items locally.

Till now most applications of MAVPs have been limited to wide, open spaces where the system is unhindered or has its own workspace. Operation of MAVPs in urban settings presents itself with notable challenges given the intricacies of the environment. See Figure 1-2 showing how a rural and urban setting compare exemplifying the complexities for urban MAVP operation.

(a) Rural scene (Credit: Llee Wu [1]) **(b)** Urban scene

Figure 1-2: Comparison of a open rural space and a moderately dense urban setting showing the differences in scene complexity. With urban settings there are moving and uncertain obstacles and the operating space is confined. Obstacle avoidance and proper planning is necessary in urban settings as the margins of error are much smaller.

Given the dynamic and uncertain nature of urban scenes, it is necessary to perform planning and control adaptively to account for unforeseen events in the surroundings. Current approaches to MAVP operation have mostly focused on pre-generating (agile) trajectories and then tracking them (the relevant article have been cited in the literature survey in Chapter 4). The main limitation of this approach is the reliance on the task-specific full motion planning making the behaviour inherently non-adaptive at run-time thus precluding the handling of uncertain, dynamic obstacles. Furthermore, generating lengthy trajectories with a priori knowledge requires considerable computational resources and is only tractable and amendable to demonstrative purposes in well defined spaces over relatively short durations. Therefore, the practicality of pre-generating trajectories for MAVPs in real-world applications is questionable.

To facilitate urban operation of MAVP systems while maintaining the ease of use and rapid deployability synonymous with MAVs, it is practical to integrate the planning aspect into the controller design to increase the system's level of autonomy. With this approach the human only needs to provide high-level planning objectives, such as the desired goal coordinates, and the MAVP is able to plan and execute the motion autonomously without requiring a pre-generated reference trajectory. Traditionally, the higher level of autonomy is achieved by extending the system's control design by an outer loop that uses a planning algorithm to regulate inner loops. In contrast, in this thesis the planning and control is unified in a Non-linear Model Predictive Control (NMPC) formulation that is able to predict a locally optimal system trajectory over a fixed time horizon, and in parallel generate control actions to realise the plan. Furthermore, the algorithm integrates feedback data in closed-loop form

[1] Llee Wu. "British rural scenery". Accessed 26 February, 2018. `https://flic.kr/p/cWA5Ys`

to facilitate online re-planning and control in dynamic environments. This approach results in a local motion planner and controller able to achieve trajectory re-planning in real-time amendable to autonomous operability. The thesis outlines the NMPC based unified planning and control approach, and shows with demonstration that it works in practice.

1-1 Thesis Content and Organisation

The study addresses the modelling, planning and control of a MAVP system in dynamic environments. The purpose of this thesis is captured in the following research objective;

> *'Demonstrate online, closed-loop, collision-free trajectory generation and control of a MAV-Payload (MAVP) system in dynamic environments using Non-Linear Model Predictive Control (NMPC)'*

The document has been divided into three parts reflecting the different stages of the study performed.

In Part I, the *Scientific Article* is presented outlining the main findings from the thesis in a succinct manner. The article contains a synopsis of the literature survey performed, the research contributions made, an outline of the trajectory planning and control algorithm, simulation and experimental results, and recommendations for future work. The thesis report supplements the article to elaborate on further studies conducted in support of the outcomes presented in the article. The scientific article is written in the IEEE Transactions style.

In Part II, the *Preliminary Study* is presented with supporting material required to develop the NMPC algorithm explained in the article. It was imperative that the physical model were accurate as NMPC's performance is model dependent, therefore, the preliminary study mainly addresses the derivation and testing of these models. The preliminary study is divided up into the following chapters;

- Chapter 2 - an introduction to the thesis topic and its relevance,
- Chapter 3 - the research outline presenting the research objective, aim and questions and putting the preliminary study into the context of the entire thesis project,
- Chapter 4 - a literature survey of the state-of-the-art approaches pertaining to MAVP system modelling, trajectory planning and control,
- Chapters 5, 6 and 7 - derivation of the physical models, and its verification and validation using simulated and experimental studies.

The conclusions from the preliminary study are presented in Chapter 9. An executive summary and abridged literature overview are presented in Appendix A and B.

In Part III, supplementary material to the scientific article is presented. This material provides a more in-depth analysis and the rationale behind the algorithms and methods used. The content contained within Part III include;

- Chapter 10 - the configuration of the Non-Linear Model Predictive Controller (NMPC) and a discussion on optimality and stability,

- Chapter 11 - the Cascaded Kalman Filter state estimator design, tuning and simulation based verification which were necessary to perform experimental trials,
- Chapter 12 - an outline of the software framework developed to perform the simulated and experimental studies during this research.

The appendices contain additional information, technical data and results that are referred to in the preliminary study and supplementary material. Appendix J is notable as it presents the primal-dual interior-point algorithm used to solve the optimisation problem for trajectory generation and control as implemented in the NMPC algorithm. It is advisable to read the *Scientific Article* first and then refer to the thesis report when clarification is required.

1-2 People Involved

The study was performed under the supervision of Dr. ir. G.C.H.E. de Croon (MAVLab department, Faculty of Aerospace Engineering), and Dr. J. Alonso-Mora (Cognitive Robotics department, Faculty of Mechanical, Maritime and Materials Engineering) at Delft University of Technology, Delft, The Netherlands.

Part I

Scientific Article

Online Trajectory Planning and Control of a MAV Payload System in Dynamic Environments using Non-Linear Model Predictive Control

Nikhil D. Potdar[1], Guido C.H.E. de Croon[2] and Javier Alonso-Mora[1]

***Abstract*— Micro Aerial Vehicles (MAVs) are increasingly being used for aerial transportation in remote and urban spaces where portability can be exploited to reach previously inaccessible and inhospitable spaces. Current approaches to MAV swung payload system path planning have primarily focused on pre-generating (agile) collision-free, or conservative minimal-swing trajectories in static environments. However, these approaches have failed to address the prospect of online re-planning in uncertain and dynamic environments which is a prerequisite for real-world deployability. This article describes a novel Non-Linear Model Predictive Controller (NMPC) for online, agile and closed-loop local trajectory planning and control addressing the limitations mentioned of contemporary approaches. We integrate the controller in a full system framework and demonstrate the algorithm's effectiveness in simulation and experimental studies. Results show the scalability and adaptability of our method to various dynamic setups with repeatable performance over several complex tasks which include flying through a narrow opening and avoiding moving humans.**

***Index Terms*— Autonomous Vehicle Navigation, Collision Avoidance, Optimization and Optimal Control.**

I. INTRODUCTION

The small size, agility and low upfront costs of Micro Aerial Vehicles (MAVs) could instigate their widespread use and quick deployment for payload transport in areas that are inaccessible or dangerous for humans and conventional (aerial) vehicles. Current applications for MAVs with slung payloads (*the MAVP system*) include search and rescue [1], package/cargo delivery and construction [2] primarily in large, rural, obstacle-free spaces.

Operation of MAVPs in urban settings presents itself with notable challenges given the complex and dynamic environment within which it would operate. While preserving the benefits synonymous with MAVs over traditional aerial vehicles, we require a system able to quickly, safely and autonomously navigate an obstacle-ridden space and adapt to different situations with no arduous system configuration. Carriage of a swinging payload vastly increases the system's spatial footprint making operation in restrictive spaces challenging. With no direct swing control and the system's causal nature, pre-emptive MAV trajectory planning and control is necessary to generate the desired swing motions to avoid potential collisions. Failing to acknowledge the system's future response when performing agile flight could result in inevitable collisions as by the time an obstacle is detected, the MAV is unable to divert the swinging payload away. Working around the problem, one may pre-generate trajectories with fully defined environments or actively minimise swing to reduce the system's dynamic response, however, as we will demonstrate, these undermine the real-world practicality of the approaches in dynamic environments.

Paper created February 27, 2018

[1]Nikhil D. Potdar (Student Aerospace Eng.) and Javier Alonso-Mora are with Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands nikhil.potdar94@gmail.com, j.alonsomora@tudelft.nl

[2]Guido C.H.E. de Croon is with the Micro Aerial Vehicle Lab, Delft University of Technology, 2629 HS Delft, The Netherlands G.C.H.E.deCroon@tudelft.nl

A. Contributions

Our main contribution is an online local motion planner and controller for safe, agile and collision-free flight of a MAVP system in dynamic environments. We base our approach on constrained optimisation using a finite-horizon Non-linear Model Predictive Control (NMPC) algorithm. A full system framework is outlined integrating the NMPC controller in a combined hardware and software based control loop. The proposed framework is used in simulated and experimental studies where we showcase our method's scalability, adaptability and performance over various complex tasks in static and dynamic environments. We compare against contemporary algorithms and emphasise our method's merits and limitations.

B. Related Work

Historically, studies of aerial vehicle control with suspended payloads involved helicopter systems with applications to load transportation [3] but with the advent of MAVs, research into MAVP systems has gained traction. This paper addresses MAVP motion planning for collision-avoidance which we broadly classify into two types, namely open-loop planning with feedback control, and unified closed-loop planning and control; our method contributes to the latter.

1) Open-Loop MAVP Trajectory Planning: Most contemporary approaches to collision-free trajectory planning for MAVP systems have addressed the tracking of pre-generated, possibly agile, trajectories in static workspaces. We refer to these as offline, open-loop planning approaches as there is no online (in-the-loop) dynamic re-planning of trajectories involved.

Using pre-generated trajectories, planar and spatial tracking of MAVP trajectories has successfully been demonstrated through accurate modelling and stabilisation of the vehicle [4], [5] sometimes utilising swing minimisation [6]–[8] to mitigate coupling disturbance effects. The latter

approach is energetically inefficient and over-conservative as the vehicle appropriates considerable control effort to reduce swing resulting in a sluggish system. To accomplish desirable yet feasible MAV and payload responses, the pre-generated trajectories are computed taking the MAVP system model into account. Algorithms to achieve this have included, amongst others, optimisation and Reinforcement Learning (RL) techniques. In the former, optimal trajectories are computed as a cost minimisation problem subject to the task objectives and the MAVP model, and encoded as full state evolutions [9], [10], or a reduced dimension state using differential-flatness [11], [12]. In RL, as used in [13]–[15], feasible action policies (the trajectory) are generated that enforce the MAVP model on state transitions.

The main limitation of pre-generating MAVP trajectories is the reliance on task-specific full motion planning which is consequently inherently non-adaptive at run-time thus precluding handling of uncertain, dynamic obstacles. Therefore, in the aforementioned studies mentioned only fully known environments with static obstacles were considered.

2) Closed-Loop MAVP Dynamic Trajectory Planning: Motion planning in dynamic environments requires re-planning at run-time to accommodate the changing environment. Closed-loop re-planning of full motion trajectories on a global level is intractable for a high-dimensional system, such as that of a MAVP, thus necessitating the use of local planners with finite time-horizons [16].

In [17], an agile and collision-free local trajectory generator and controller method was demonstrated in simulated and experimental setups with static obstacles using iterative Linear Quadratic Gaussian (iLQG) control. The optimal control iLQG method relies on a cost function that is minimised at every control step such that user-defined planning objectives are met; the result is a local trajectory satisfying the objectives and system dynamics. The iLQG's iterative algorithm is exploited to generate locally optimal linear feedback controls to achieve the real-time, closed-loop performance. Impressive manoeuvres including flight through a narrow opening as in [11] were demonstrated. However, dynamic environments were not considered and system inputs would saturate at run-time as the controller did not consider physical constraints. In contrast, our approach takes into account model constraints to ensure the physical feasibility of generated trajectories.

3) Non-Linear Model Predictive Control and Unified Planning and Control of MAV(P)s: Early studies have successfully demonstrated the use of (N)MPC for real-time MAV [18], [19] and MAVP [8], [20] simple trajectory tracking. Focussing on the latter, in [8] NMPC for MAVP trajectory tracking control was addressed with a comparison to LQR control. The results showed NMPC's superior physical constraint handling for feasibility guarantees, and larger attainable MAVP flight envelope from the non-linear MAVP model description. Overall NMPC outperformed LQR in simulated tasks involving swing minimisation and agile manoeuvres. In [20], studies from [8] were extended to an experimental setup validating the results. However, unlike in [17], both studies only addressed the control aspect of tracking pre-generated trajectories. In contrast, our method unifies online planning and control, and not just tracking of a pre-defined plan thus making it a higher level approach.

Traditionally, (N)MPC algorithms for unified motion planning and control of MAVs have seldom been studied as the required real-time re-planning was computationally intractable [21]. With today's improved computing capabilities, applications have been demonstrated for a MAV without swung payload [21], [22]. Building on the approach, in this work we demonstrate the viability of NMPC based unified motion planning and control for MAVPs.

C. Paper Organisation

We introduce preliminaries in Section II with notations and our system models. In Section III we describe our method for online and closed-loop unified motion planning and control with NMPC. For the simulated and experimental studies performed, we outline our system setup and framework in Section IV. In Sections V and VI we present and discuss our findings followed by concluding remarks in Section VII.

II. PRELIMINAIRES

A. Notation

The following notations are observed; scalars x, vectors/matrices $\boldsymbol{x}$, sets $\mathcal{X}$ and reference frames $\{X\}$. Time derivatives use dot accenting. Position vectors are denoted by $\boldsymbol{p} \in \mathbb{R}^3$. Unless otherwise stated, vectors are expressed in the East-North-Up (ENU) inertial frame $\{I\}$. For vector $\boldsymbol{x} \in \mathbb{R}^n$ and positive semi-definite $n{\times}n$ matrix $\boldsymbol{Q}$, the weighted squared norm is $\|\boldsymbol{x}\|_{\boldsymbol{Q}} \triangleq \boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x}$. Rotations from frame $\{A\}$ to $\{B\}$ are denoted by matrix $\boldsymbol{R}_A^B{\in}SO(3)$ and basic axial rotations around x by $\boldsymbol{R}_x{\in}SO(3)$.

B. Quadrotor with Swung Payload Model

The system is composed of a quadrotor of mass m_q and a suspended point mass m_l attached by a l length cable from the quadrotor centroid. Let $\boldsymbol{p}_q$, $\boldsymbol{p}_l$ be the quadrotor, load position and $\boldsymbol{r}_{ql} = \boldsymbol{p}_l - \boldsymbol{p}_q$. All reference frames are defined in Fig. 1. The load suspension angles θ_l, ϕ_l

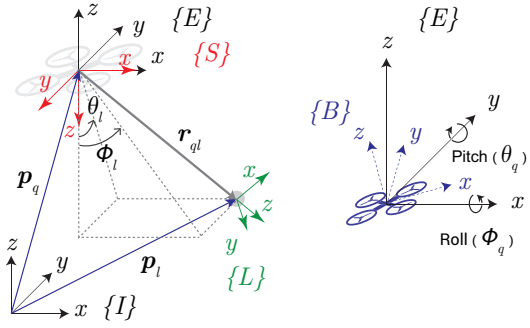


Fig. 1: Quadrotor-Payload system with the following references frames; $\{I\}$ inertial ENU, $\{E\}$ vehicle-carried ENU, $\{B\}$ vehicle body frame, $\{S\}$ is $\{E\}$ rotated by 180° about the $\{E\}$ x-axis and $\{L\}$ load frame with z-axis directed away from the cable's suspension point. Quadrotor and load positions and relative suspensions angles indicated. Euler angles ϕ_q, θ_q parametrise frame $\{B\}$ to $\{E\}$; constant yaw assumed.

parametrise the orientation of $\{L\}$ to $\{S\}$. Intermediary frame $\{S\}$ is used to avoid the singularity for a downward equilibrium load position when computing a rotation from $\{L\}$ to $\{E\}$ directly. Then let the MAVP configuration and its time derivative be given by variables

$$\boldsymbol{q} = \left[\boldsymbol{p}_q^\top,\ \theta_l,\ \phi_l\right] \in \mathbb{R}^5$$
$$\dot{\boldsymbol{q}} = \tfrac{d}{dt}\boldsymbol{q} = \left[\dot{\boldsymbol{p}}_q^\top,\ \dot{\theta}_l,\ \dot{\phi}_l\right] \in \mathbb{R}^5\ ,$$

and let θ_q, ϕ_q be the true quadrotor pitch and roll with yaw constant. The following additional model assumptions are adopted;

- rigid, massless cable with free suspension points,
- quadrotor centre of gravity and centroid coincide,
- no aerodynamic drag effects on the cable.

We first describe the quadrotor's input handling and the aerodynamic drag model. We then complete the model by derivation of the coupled quadrotor-load dynamics.

1) Quadrotor Inputs: As in [23], we abstract quadrotor motor inputs and assume fast attitude and motor control such that by actuating the quadrotor's pitch and roll, and setting a desired vertical velocity, we produce an inertial control force $\boldsymbol{F}_u$ in any desired direction for realising translational motion. Therefore, let the inputs be a desired quadrotor pitch, roll and vertical velocity defined in $\{E\}$ giving

$$\boldsymbol{u} = \left[\bar{\theta}_q,\ \bar{\phi}_q,\ \bar{w}_q\right] \in \mathbb{R}^3\ .$$

This input choice is consistent with our chosen Parrot Bebop 2[1] quadrotor that internally controls motors based on inputs $\boldsymbol{u}$ to achieve full spatial flight; the internal controller is schematised in Appendix A.

As the hardware-specific internal controller dynamics $\boldsymbol{u} \rightarrow \boldsymbol{F}_u$ are not documented, we empirically model the function. The quadrotor's true pitch, roll response and the vertical control force resulting from the vertical velocity input is given by

$$[\theta_q,\ \phi_q,\ F_q] = \left[h_\theta(\bar{\theta}_q),\ h_\phi(\bar{\phi}_q),\ h_F(\bar{w}_q)\right] \tag{1}$$

where, using the method presented in [24], $h_\theta,\ h_\phi,\ h_F$ are identified for the fast dynamics and decoupled as three linear second-order black-box models with model states and state transition

$$\begin{aligned}\boldsymbol{x}_c &= [x_{\theta,1},\ x_{\theta,2},\ x_{\phi,1},\ x_{\theta,2},\ x_{F,1},\ x_{F,2}] \in \mathbb{R}^6\\ \dot{\boldsymbol{x}}_c &= f_c(\boldsymbol{x}_c, \boldsymbol{u})\ .\end{aligned} \tag{2}$$

Note that with h_F we model $\bar{w}_q \rightarrow F_q$ directly as the internal vertical velocity stabiliser controls the vertical control force (in $\{E\}$) generated by the motors (See Appendix A). Then similar to [22], using outputs from (1) and based on equilibrium relations, the input control force is given by

$$\boldsymbol{F}_u = \left[m\tfrac{\tan(\theta_q)}{\cos(\phi_q)}g,\ \ -m\tan(\phi_q)g,\ F_q + mg\right] \in \mathbb{R}^3 \tag{3}$$

where $m = m_q + m_l$ and $g = 9.81\,\mathrm{m/s^2}$.

The full-form of (1) identified for the Parrot Bebop 2 quadrotor is provided in Appendix B.

[1]Parrot. `http://developer.parrot.com/docs/SDK3/`

2) Aerodynamic Drag Effects: We derive the drag induced forces on the MAVP system. As in [23], assuming relatively low quadrotor velocities $\dot{\boldsymbol{p}}_q$, we model a proportional linear drag force on the quadrotor with drag constant k_{Dq} giving

$$\boldsymbol{F}_{Dq} = k_{Dq}\dot{\boldsymbol{p}}_q\ . \tag{4}$$

Also as in [23], for the payload we only consider the rotational load motion relative to the quadrotor, hence, its drag force is assumed to always be perpendicular to the moment arm (the rigid cable). This simplifies the model complexity enabling us to model quadratic drag as an induced moment on the load suspension angles. Additionally, following from our free suspension point assumption, there are no payload drag induced reactive forces or moments on the quadrotor. Under these simplifications, the load's signed quadratic drag force, with drag constant k_{Dl}, is given by

$$F_{Dl} = k_{Dl}v^2\tfrac{v}{|v|} \equiv k_{Dl}l^2\omega^2\tfrac{\omega}{|\omega|} \tag{5}$$

where $v = \omega l$ for circular motion with v, ω the linear, angular load velocities and l the cable length. Substituting ω in (5) by the load's suspension angular rates and computing the induced moment at the suspension point we obtain

$$[\tau_\theta,\ \tau_\phi] = k_{Dl}l^3\left[\omega_\theta^2\tfrac{\omega_\theta}{|\omega_\theta|},\ \omega_\phi^2\tfrac{\omega_\phi}{|\omega_\phi|}\right] \tag{6}$$

where $\omega_\theta = \dot{\theta}_l$, $\omega_\phi = \dot{\phi}_l$ and τ_θ, τ_ϕ are the load's drag force induced moments affecting suspension angles θ_l, ϕ_l. With (4) and (6), the total exogenous system drag term is

$$\boldsymbol{D}(\dot{\boldsymbol{q}}) = \left[\boldsymbol{F}_{Dq}^\top,\ \tau_\theta,\ \tau_\phi\right]^\top\ . \tag{7}$$

3) System Kinematics and Dynamics: The MAVP Equations of Motion (EOMs) are derived in frame $\{I\}$ according to Lagrangian mechanics. With frame transformations

$$\boldsymbol{R}_L^S = \boldsymbol{R}_y(\phi_l)\boldsymbol{R}_x(\theta_l) \tag{8}$$
$$\boldsymbol{R}_S^E = \boldsymbol{R}_x(\pi)\ , \tag{9}$$

and $\boldsymbol{l} = [0,\ 0,\ l]^\top$ the rigid cable vector in $\{L\}$, we define the load position as

$$\boldsymbol{p}_l = \boldsymbol{p}_q + \boldsymbol{r}_{ql} = \boldsymbol{p}_q + \boldsymbol{R}_S^E\boldsymbol{R}_L^S\boldsymbol{l}\ . \tag{10}$$

The payload velocity is then given by

$$\dot{\boldsymbol{p}}_l = \tfrac{d}{dt}\boldsymbol{p}_l = \dot{\boldsymbol{p}}_q + \boldsymbol{R}_S^E\dot{\boldsymbol{R}}_L^S\boldsymbol{l}\ . \tag{11}$$

The Lagrangian in terms of the system kinetic and potential energies is

$$L = \underbrace{0.5\left\|[\dot{\boldsymbol{p}}_{\boldsymbol{q}},\ \dot{\boldsymbol{p}}_{\boldsymbol{l}}]^\top\right\|_{\boldsymbol{E}}}_{\text{kinetic energy}} - \underbrace{g\left(m_q\dot{\boldsymbol{p}}_q^\top\boldsymbol{e}_3 + m_l\dot{\boldsymbol{p}}_l^\top\boldsymbol{e}_3\right)}_{\text{potential energy}} \tag{12}$$

where $\boldsymbol{E} = \mathrm{diag}(m_{q(1\times3)},\ m_{l(1\times3)})$ and $\boldsymbol{e}_3 = [0,0,1]^\top$.

Using Lagrange's equations according to D'Alembert's principle, the non-linear EOMs describing the MAVP dynamics in compacted form are given by

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})\left(\boldsymbol{F} - \boldsymbol{D}(\dot{\boldsymbol{q}}) - \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{G}(\boldsymbol{q})\right) \tag{13}$$

with force $\boldsymbol{F} = [\boldsymbol{F}_u,\ 0,\ 0]^\top \in \mathbb{R}^5$, and mass $\boldsymbol{M}$, drag $\boldsymbol{D}$ from (7), Coriolis $\boldsymbol{C}$ and gravitational $\boldsymbol{G}$ matrix terms.

Equation (13) in full form is presented in [23]. Using (13), the system state and state transition is given by

$$\begin{aligned} \boldsymbol{x}_q &= [\boldsymbol{q},\ \dot{\boldsymbol{q}}] \in \mathbb{R}^{10} \\ \dot{\boldsymbol{x}}_q &= [\dot{\boldsymbol{q}},\ \ddot{\boldsymbol{q}}] = f_q(\boldsymbol{x}_q, \boldsymbol{F_u}) \ . \end{aligned} \tag{14}$$

4) Full MAVP Model: Combining the quadrotor input and system model from (2) and (14), we denote the full MAVP state and state transition by

$$\begin{aligned} \boldsymbol{x} &= [\boldsymbol{x}_c,\ \boldsymbol{x}_q] \in \mathbb{R}^{16} \\ \dot{\boldsymbol{x}} &= [\dot{\boldsymbol{x}}_c,\ \dot{\boldsymbol{x}}_q] = f(\boldsymbol{x}, \boldsymbol{u}) \ . \end{aligned} \tag{15}$$

Important MAVP model related variables and parameters that we often refer to are summarised in Table I.

C. Obstacle Model

Obstacles, with each position $\boldsymbol{p}_o$, are user-specified as cuboids and subsequently modelled by enclosing ellipsoids. Human obstacles are also specified as a cuboid of similar size. Ellipsoids create smooth convex bounding volumes for (non-convex) obstacles making them appropriate for representing objects including trees, humans and pillars. Additionally, computationally efficient collision checks against the ellipsoid's quadric exist [25] and is thus favourable for real-time applications.

1) Obstacle Ellipsoid Definitions: Let the ellipsoid semi-principal axes (a_o, b_o, c_o) be proportional to the specified cuboid dimensions (u_o, v_o, w_o) such that there is surface contact at all cuboid corners, hence

$$(a_o, b_o, c_o) = \tfrac{\sqrt{3}}{2}\,(u_o, v_o, w_o) \ .$$

We define two ellipsoids with buffers β as shown in Fig. 2;

1) the *bounding ellipsoid* S_o with dimensions $(a_o + \beta_o, b_o + \beta_o, c_o + \beta_o)$ models the obstacle against which collisions are checked,
2) the *expanded ellipsoid* S_e with dimensions $(a_o + \beta_e, b_o + \beta_e, c_o + \beta_e)$ represents a padding used for planning safer trajectories.

Note by setting β, a minimum cuboid to ellipsoid separation of β is warranted. Buffers β_o, β_e are used for collision-avoidance purposes as will become clear later.

2) Obstacle Motion Prediction: Static obstacle positions are assumed to be readily available for planning. As in [22], we assume a constant velocity model for dynamic obstacles and predict their future positions based on a velocity estimate produced by a linear Kalman Filter using measured obstacle position data.

TABLE I: MAVP system variables and parameters

Notation	Definition
$m_q, m_l; g \in \mathbb{R}$	Mass of quadrotor, load; Gravitational acceleration
$l; \theta_l, \phi_l \in \mathbb{R}$	Cable length; Payload suspension angles
$\boldsymbol{p}_q, \boldsymbol{p}_l \in \mathbb{R}^3$	Position of quadrotor, payload in $\{I\}$
$\boldsymbol{q}, \dot{\boldsymbol{q}} \in \mathbb{R}^5$	MAVP configuration, and its time derivative
$\boldsymbol{u} \in \mathbb{R}^3$	Quadrotor input commands
$\boldsymbol{F}, \boldsymbol{F_u} \in \mathbb{R}^n$	General, control input force in $\{I\}$
$\boldsymbol{x}_c, \boldsymbol{x}_q, \boldsymbol{x} \in \mathbb{R}^n$	Quad. input, system and full MAVP model state

Fig. 2: Cuboid obstacle (*left*) with fixed position $\boldsymbol{p}_o$ or dynamic (human) obstacle (*right*) with constant velocity $\boldsymbol{v}_o$ each modelled by bounding S_o and expanded S_e ellipsoid with dimensional buffers β.

D. MAVP-Obstacle Collision Avoidance Requirements

Imperative to collision avoidance is ensuring separation between the MAVP and obstacles. By quantifying the quadrotor, load and cable's proximity to an obstacle, we present mathematical requirements to guarantee a collision-free system.

1) Point to Ellipsoid Distance: The point to an ellipsoid signed distance is approximated as the true value cannot be expressed in closed form [25]. For a generic ellipsoid S with buffered dimensions $(a_o + \beta, b_o + \beta, c_o + \beta)$ and position $\boldsymbol{p}_o$, the approximate signed distance to a point $\boldsymbol{p}$ based on the ellipsoid equation is

$$d_o(\boldsymbol{p}, S) = \|\boldsymbol{p} - \boldsymbol{p}_o\|_\Omega - 1 \tag{16}$$

where $\Omega = \text{diag}(1/(a_o + \beta)^2, 1/(b_o + \beta)^2, 1/(c_o + \beta)^2)$.

When $\boldsymbol{p}$ is inside or on S, $d_o \leq 0$, and as $\boldsymbol{p}$ is further away from S, d_o increases from 0 to infinity.

2) Quadrotor and Payload Proximity: We model the quadrotor and payload each by a bounding sphere with an associated radius r_c. Without loss of generality, we assume an equal r_c for the quadrotor and payload. Consider the quadrotor; using the obstacle's bounding ellipsoid S_o, and setting $\beta_o > r_c$ and $\boldsymbol{p} = \boldsymbol{p}_q$, then using (16) we can guarantee the quadrotor does not collide with the cuboid shaped obstacle provided

$$d_o(\boldsymbol{p}_q,\ S_o) > 0 \ . \tag{17}$$

Similarly, considering the payload associated bounding sphere and position $\boldsymbol{p}_l$ gives

$$d_o(\boldsymbol{p}_l,\ S_o) > 0 \ . \tag{18}$$

3) Rigid Cable Proximity: Modelling the cable as a mobile finite line segment, we identify the cable's Closest Point of Approach (CPA) to S_o denoted by $\boldsymbol{p}_c^*$; this is the cable's most critical point for collisions. Given the cable cross-section dimensions are negligible, no buffer is required so $\beta_o = 0$. Using (16), $\boldsymbol{p}_c^*$ is computed by

$$\boldsymbol{p}_c^* = \underset{\boldsymbol{p}_c}{\arg\min}\,(d_o(\boldsymbol{p}_c,\ S_o)) \tag{19}$$

with $\boldsymbol{p}_c \in \{\boldsymbol{p} | \boldsymbol{p} = \boldsymbol{p}_q + s(\boldsymbol{p}_l - \boldsymbol{p}_q), s \in [0, 1]\}$. Appendix. C shows (19) is analytically solvable. Using (19) the cable is guaranteed to be collision-free with respect to the cuboid obstacle provided,

$$d_o(\boldsymbol{p}_c^*,\ S_o) > 0 \ . \tag{20}$$

Requirements (17-18,20) must be satisfied with respect to each obstacle to guarantee a collision-free MAVP system.

III. ONLINE AND CLOSED-LOOP MAVP TRAJECTORY GENERATION

A. Method Overview

The planning and control objective is to navigate the MAVP system from an initial position $\boldsymbol{p}_{\text{start}}$ to a user-definable goal position $\boldsymbol{p}_{\text{goal}}$ in a safe, agile and collision-free manner. To accommodate the dynamic environment, we perform dynamic and closed-loop local motion planning using NMPC which is a receding finite-horizon controller.

1) Receding Horizon Dynamic Planning: Denote by Δt the time-step, by k the stage index, and by N the finite planning horizon (number of stages). At every sampling instance t, we generate a local trajectory of duration $N\Delta t$ encoded as a sequence of $N+1$ states that includes the initial state $\boldsymbol{x}_0$, the transition states $\boldsymbol{x}_k$ and a terminal state $\boldsymbol{x}_N$ and is denoted by

$$\tilde{\boldsymbol{x}} := [\boldsymbol{x}_0, \dots, \boldsymbol{x}_N] \ . \tag{21}$$

For state realisation, the associated input sequence up to the terminal state is denoted by

$$\tilde{\boldsymbol{u}} := [\boldsymbol{u}_0, \dots, \boldsymbol{u}_{N-1}] \ . \tag{22}$$

Following execution of $\boldsymbol{u}_0$, the planning is receded by Δt to $t + \Delta t$. At the next sampling instance, the new obstacle positions and a new initial state estimate $\boldsymbol{x}_0$ is obtained. Then a local trajectory is re-generated by initialising the solver with a time-shifted version of the previous solution. This approach results in computationally efficient and closed-loop performance with robustness to model uncertainties [22]. We illustrate the process in Fig. 3.

2) Local Trajectory Generation: At every sampling instance we solve a constrained optimisation problem. The designer encodes the desired planning objectives in an *objective function* using *costs* to quantify the generated trajectory's performance. The costs are designed to lower with an increasing satisfaction of the objective. For every trajectory stage k, we evaluate an associated scalar cost giving a cost sequence

$$\tilde{\boldsymbol{c}} := [c_0, \dots, c_N] \ . \tag{23}$$

Within (23), the trajectory *stage* costs are given by

$$c_k = c_s(\boldsymbol{x}_k, \boldsymbol{u}_k, *_k), k \in [0, N-1] \tag{24}$$

where function c_s is evaluated on the predicted state, input and any additional variables (obstacle positions, navigation goal, slacks etc.) that we denote by $*$. The *terminal* cost is given by

$$c_N = c_t(\boldsymbol{x}_N, *_N) \tag{25}$$

where function c_t is evaluated on variables of the terminal stage. Terminal costs are used to achieve closed-loop stability of the finite-horizon planner [26].

We then quantify the full trajectory's performance by the *objective function* defined as

$$J = \sum\nolimits_{k=0}^{N} c_k \ . \tag{26}$$

Constraints are introduced to limit the solution space for the trajectory encoded in $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{u}}$ thus providing (feasibility) guarantees for the computed trajectory. To make sure the optimiser always returns a solution at run-time, we may tolerate minor constraint violations by introducing non-negative slack variables that *soften* the constraint [27]. Then the slack variables associated to the trajectory are encoded in

$$\tilde{\boldsymbol{s}} := [\boldsymbol{s}_0, \dots, \boldsymbol{s}_N] \ . \tag{27}$$

A *planning violation* occurs when the optimiser produces positive entries of $\tilde{\boldsymbol{s}}$. A *physical violation* only occurs when the real system breaches constraints; i.e., the current slack $\boldsymbol{s}_0$ of $\tilde{\boldsymbol{s}}$ is positive. By associating a high slack related cost in the optimisation objective function, we avoid positive entries of $\tilde{\boldsymbol{s}}$ and accordingly any planning and physical violations [27].

During optimal trajectory generation we minimise (26) respecting the constraints resulting in a $N\Delta t$ length locally feasible trajectory. In subsequent sections we introduce the costs and constraints after which we formalise the optimisation algorithm in Section. III-D.

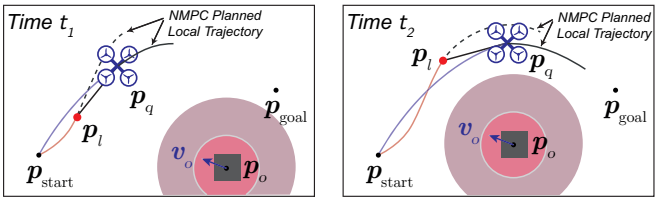


Fig. 3: System moves towards $\boldsymbol{p}_{\text{goal}}$ with $t_2 > t_1$; planned local trajectory (grey) at current time (*left*) that is updated in a future time (*right*) with the new state and obstacle data. Schematic projected top view with illustrative obstacle ellipsoids shown.

B. Costs

We introduce cost terms derived from our planning objectives presented in Section III-A. We use our weighted square norm definition from Section II-A with an $n \times n$ identity matrix denoted by $\boldsymbol{I}_n$ to make all cost terms scalar and positive.

1) Point-to-Point Navigation: For navigation, we minimise the displacement between the quadrotor position and goal $\boldsymbol{p}_{\text{goal}}$. Let $\boldsymbol{p}_{\text{start}}$ be the start position, then we normalise the cost to treat all start to goal distances equally. The cost term is given by

$$c_{\text{nav}} = \frac{\|\boldsymbol{p}_{\text{goal}} - \boldsymbol{p}_q\|_{\boldsymbol{I}_3}}{\|\boldsymbol{p}_{\text{goal}} - \boldsymbol{p}_{\text{start}}\|_{\boldsymbol{I}_3}} \ . \tag{28}$$

Making (28) a stage cost would mean the shortest path (straight line) is always preferred, this may result in deadlocks when it is necessary to go around an obtrusive obstacle. Therefore, we use (28) only as a terminal cost thus allowing curved paths to be generated such that locally and terminally the system reaches a more favourable position.

2) Potential Field based Obstacle Separation: For obstacle separation, we employ a MAVP to obstacle proximity related cost analogous to a reactive potential field [28]. We combine this with constraints to guarantee collision-free trajectories as will be presented in Section III-C.3. This two layered approach, similar to [22], enhances the operational safety by pro-actively reducing the collision risk, especially for unmodelled system and obstacle dynamics.

Let $\boldsymbol{p}$ be the quadrotor, load or cable's CPA [see (19)] position; for each we compute a cost. Let $\boldsymbol{p}_o$ be the obstacle's predicted position, then the potential cost term activates when $\boldsymbol{p}$ is in the obstacle's expanded ellipsoid S_e; i.e., using (16), $d_o(\boldsymbol{p},\ S_e) < 0$. We choose the S_e associated buffer β_e such that $\beta_e \gg \beta_o$. Observing that $|d_o(\boldsymbol{p},\ S_e)|$ increases from zero to one as point $\boldsymbol{p}$ moves from the ellipsoid surface towards its centre, by penalising a $\boldsymbol{p}$ more towards the centre, we naturally discourage $\boldsymbol{p}$ from getting closer to the smaller bounding ellipsoid S_o. Given S_o models the actual obstacle, using this method we promote obstacle separation. With this insight, and using (16), the cost is formalised as

$$c_{\text{pf}} = \begin{cases} \|d_o(\boldsymbol{p}, S_e)\|_{\boldsymbol{I}_1} & \text{, if } d_o(\boldsymbol{p}, S_e) < 0, \\ 0 & \text{, otherwise.} \end{cases} \tag{29}$$

3) Goal Directed Assistive Steering: Optionally we can augment collision-free trajectory generation for low planning horizons using assistive steering; the idea is inspired from Vector Field Histograms [29]. The quadrotor position and obstacle ellipsoids are projected onto the world horizontal plane P by the transformation $\boldsymbol{T}_P : \mathbb{R}^3 \to \mathbb{R}^2$ where $\boldsymbol{T}_P = \text{diag}(1, 1, 0)$. On P, we define a set of n_d candidate angular directions for steering

$$\mathcal{D} = \left\{ \gamma | \, \gamma = i\tfrac{2\pi}{n_d}, i \in \{1, \ldots, n_d\} \subset \mathbb{N} \right\}$$

originating from our projected quadrotor position $\boldsymbol{T}_P\boldsymbol{p}_q$. Checking all $\gamma \in \mathcal{D}$, we determine $\mathcal{D}_{\text{free}} \subseteq \mathcal{D}$ which are all the non-obstructed (free) directions in P up to a maximum omnidirectional range from $\boldsymbol{T}_P\boldsymbol{p}_q$. With γ_{goal} the heading of the goal position from the quadrotor position, the steering direction is chosen to minimise the angular offset to the goal as given by

$$\gamma^* = \arg\min_{\gamma} |\gamma - \gamma_{\text{goal}}|, \gamma \in \mathcal{D}_{\text{free}} \ . \tag{30}$$

With $\angle\boldsymbol{T}_P\dot{\boldsymbol{p}}_q$ the quadrotor's heading, the cost is evaluated as its deviation from γ^* by

$$c_{\text{steer}} = \|\angle\boldsymbol{T}_P\dot{\boldsymbol{p}}_q - \gamma^*\|_{\boldsymbol{I}_1} \ . \tag{31}$$

In our results we demonstrate the utility of steering only when using low planning horizons. As the $\mathbb{R}^2$ steering method is only amendable to planar obstacle avoidance, for $\mathbb{R}^3$ spatial avoidance we disable steering. Extension to $\mathbb{R}^3$ could be done analogously.

4) Input Magnitude Regulation: The input magnitude associated cost is given by

$$c_{\text{in}} = \left\|\boldsymbol{u}^\top\right\|_{\boldsymbol{I}_3} \ . \tag{32}$$

For our agile manoeuvres, we weight this cost low. Associating a high cost will improve the system's energy-efficiency by the conservative use of large inputs.

5) Payload Suspension Angles Regulation: The suspension angle associated cost is given by

$$c_{\text{swing}} = \left\|[\theta_l, \phi_l]^\top\right\|_{\boldsymbol{I}_2} \ . \tag{33}$$

For our agile manoeuvres, we weight this cost low. Associating a high cost will minimise the swing angles if desired.

C. Constraints

We derive constraints from our system and setup limits, and planning objectives.

1) MAVP Dynamics: The process model state transition given by (15) is discretised and enforced on the trajectory state evolution by an inter-stage equality constraint

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{34}$$

where k is the stage index.

2) State and Input Limits: The state and input values are bound to the system allowable ranges. Let $\mathcal{X}_{\text{min}}$, $\mathcal{X}_{\text{max}}$ and $\mathcal{U}_{\text{min}}$, $\mathcal{U}_{\text{max}}$ denote the state and input range limits, then the following inequalities must be satisfied

$$\mathcal{X}_{\text{min}} \leq \boldsymbol{x} \leq \mathcal{X}_{\text{max}} \tag{35}$$

$$\mathcal{U}_{\text{min}} \leq \boldsymbol{u} \leq \mathcal{U}_{\text{max}} \ . \tag{36}$$

We specify the hardware-specific limits in Section IV.

3) Collision-Free Planning: Collision-free trajectory planning is guaranteed by constraining the allowable system's spatial states. Let $\boldsymbol{p}$ be the quadrotor, load or cable's CPA [see (19)] position; for each we define a constraint. Adopting the requirements (17-18, 20) for a collision-free MAVP system as presented in Section II-D, and using (16), the associated constraint is formalised as

$$d_o(\boldsymbol{p}, S_o) + s_c > 0 \tag{37}$$

with the non-negative scalar slack s_c.

4) Workspace Limits: For confined (indoor) operation, the quadrotor and load position is limited to the workspace limits. Assume a cuboid workspace, then let $\mathcal{W}_{\text{min}}$, $\mathcal{W}_{\text{max}}$ denote the minimum and maximum workspace coordinates in frame $\{I\}$, and between which the cuboid's space diagonal is defined, then the following inequalities must be satisfied

$$\boldsymbol{p}_q + \boldsymbol{1}_3 s_q \geq \mathcal{W}_{\text{min}} \quad \text{and} \quad \boldsymbol{p}_q - \boldsymbol{1}_3 s_q \leq \mathcal{W}_{\text{max}} \tag{38}$$

$$\boldsymbol{p}_l + \boldsymbol{1}_3 s_l \geq \mathcal{W}_{\text{min}} \quad \text{and} \quad \boldsymbol{p}_l - \boldsymbol{1}_3 s_l \leq \mathcal{W}_{\text{max}} \tag{39}$$

with $\boldsymbol{1}_3 = [1, 1, 1]^\top$ and the non-negative scalar slacks s_q, s_l.When a constraint violation occurs, the slacks assume the highest value required to satisfy the associated workspace inequalities. Under workspace convexity, we also guarantee the rigid cable remains inside $\mathcal{W}$. Note that the inequalities are written in short form, however, for implementation each vector dimension would each have an individually defined inequality.

5) Scalability to Large Obstacle Rich Workspaces: We set a maximum omnidirectional obstacle detection range originating from the quadrotor position whereby we disregard any obstacles beyond the range for planning purposes. Therefore, the previously introduced obstacle related costs and constraints are dynamically implemented.

D. Optimisation Algorithm

Local trajectory generation is formulated as a constrained optimisation problem subject to the following costs and constraint definitions;

1) Costs: We define the stage and terminal cost functions based on the cost term definitions (28-29, 31, 32-33). Let w denote a user-definable weighting used to assign relative importance to costs and their associated objective, then the stage cost function is given by

$$c_s = w_{\text{in}}c_{\text{in}} + w_{\text{swing}}c_{\text{swing}} + w_{\text{steer}}c_{\text{steer}} + \boldsymbol{w}_{\text{pf}}^{\top}\boldsymbol{c}_{\text{pf}} + \boldsymbol{w}_{\text{slack}}^{\top}\boldsymbol{s} \quad (40)$$

where $\boldsymbol{w}_{\text{pf}}$ and $\boldsymbol{c}_{\text{pf}}$ are vectors of weights and costs equal in size to the number of obstacles, and $\boldsymbol{w}_{\text{slack}}^{\top}\boldsymbol{s}$ all the slacks associated cost. The terminal cost function is given by

$$\begin{aligned} c_t = w_{\text{nav}}c_{\text{nav}} + w_{\text{in}}c_{\text{in}} + w_{\text{swing}}c_{\text{swing}} + w_{\text{steer}}c_{\text{steer}} \\ + \boldsymbol{w}_{\text{pf}}^{\top}\boldsymbol{c}_{\text{pf}} + \boldsymbol{w}_{\text{slack}}^{\top}\boldsymbol{s} \ . \end{aligned} \quad (41)$$

2) Constraints: We impose the system dynamics, and state/input constraints, as introduced in Section III-C, on the optimiser. By function $\boldsymbol{g}$ we denote all additional inequality constraints defined in (37-39). The constraint associated slacks $\boldsymbol{s} = [s_c, s_q, s_l] \in \mathbb{R}^3$ must be non-negative.

Combining our previously introduced trajectory variables (21-22, 27), we denote the optimisation variable by

$$\begin{aligned} \tilde{\boldsymbol{z}} &= [\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N, \boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}, \boldsymbol{s}_0, \ldots, \boldsymbol{s}_N] \\ &\equiv [\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{s}}] \ . \end{aligned} \quad (42)$$

With the estimated initial state $\boldsymbol{x}_0$, we optimise $\tilde{\boldsymbol{z}}$ such that the objective function (26) is minimised resulting in a locally optimal and feasible trajectory. With costs and constraints stacked together over all stages and obstacles, the optimisation problem that is solved at every planning time instance t is formally defined as

$$\begin{aligned} \min_{\tilde{\boldsymbol{z}}} \quad & J = c_t(\boldsymbol{x}_N, *_N) + \Sigma_{k=0}^{N-1} c_s(\boldsymbol{x}_k, \boldsymbol{u}_k, *_k) \\ \text{s.t.} \quad & \boldsymbol{x}_0 = \boldsymbol{x}(t) && \text{(Initial Estimated State)} \\ & \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) && \text{(Discretised Dynamics)} \\ & g(\boldsymbol{x}_k, \boldsymbol{u}_k, *_k) \geq 0 && \text{(Inequality Constraints)} \\ & \boldsymbol{x}_k \in \mathcal{X} && \text{(State Constraints)} \\ & \boldsymbol{u}_k \in \mathcal{U} && \text{(Input Constraints)} \\ & \boldsymbol{s}_k \geq 0 && \text{(Slack Constraints).} \end{aligned} \quad (43)$$

E. Theoretical Analysis

1) Problem Dimensionality: Variable $\tilde{\boldsymbol{z}}$ is optimised at every planning time instance encoding the optimised local trajectory in its solution. As given by (42), $\tilde{\boldsymbol{z}}$ comprises a sequence of $N+1$ states $\boldsymbol{x} \in \mathbb{R}^{16}$, N inputs $\boldsymbol{u} \in \mathbb{R}^3$ and $N+1$ slacks $\boldsymbol{s} \in \mathbb{R}^3$, hence $\tilde{\boldsymbol{z}} \in \mathbb{R}^{22N+19}$.

2) Optimality and Feasibility: We use a fast non-linear programming based optimiser, namely FORCES PRO [30], on our non-convex optimal control problem. Consequently, the computed trajectories are only locally optimal over the planning horizon N with the possibility of deadlocks when the planned trajectory converges to any local optima in the solution space.

Planning feasibility is warranted over the full N stages when all optimised slacks $\tilde{\boldsymbol{s}}$ are zero. When full planning feasibility is not realised, provided that at least the current slack $\boldsymbol{s}_0$ is zero, the current system state and inputs will be feasible. Re-planning at a future instance can re-establish full planning feasibility.

A comprehensive overview of the optimality and stability of (N)MPC algorithms is available in [26].

IV. SYSTEM SETUP AND FRAMEWORK

We outline our particular implementation of the system model and NMPC controller for simulated and experimental studies. For the latter, we also present a state estimator.

A. System Properties and Hardware

The MAVP system properties used for all studies are given in Table II. The MAVP hardware is shown in Fig. 4.

B. Workspace

We perform studies in a simulated and real workspace measuring $6.0 \times 3.0 \times 2.6\,\text{m}$ (L×W×H). The real indoor workspace has a OptiTrack[2] Motion Capturing System (MCS) that can track markers for obtaining rigid-body pose measurements in $SE(3)$ at around 120 Hz.

C. Programmed Control System Framework

The control framework is schematised in Fig. 5; in Appendix A the on-board control component is expanded upon. The off-board components run on an Intel i7 Quad Core 3.3GHz processor PC, and is programmed in MATLAB with an efficient C language solver FORCES PRO performing the online NMPC computations [30]. All studies are performed on the same computer with at maximum one core being utilised by FORCES PRO at run-time. The on-board components run on the MAVP hardware; for simulated studies we replicate this with our system model. In experiments, communication between hardware is performed over a ROS

[2]OptiTrack Prime 17W. `http://www.optitrack.com`

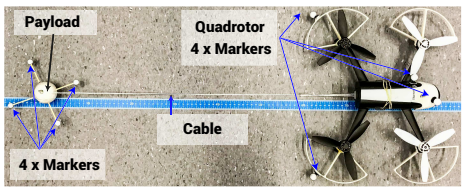


Fig. 4: Parrot Bebop 2 quadrotor with cable suspended load and attached tracking markers.

TABLE II: MAVP system properties as used for study

Quad. Mass	500 g	Quad. Drag Const. k_{Dq}	0.28
Load Mass	11 g	Load Drag Const. k_{Dl}	1.77×10^{-3}
Cable Length	0.77 m	Max. $\lvert\bar{\theta}_q\rvert$, $\lvert\bar{\phi}_q\rvert$ Input	15°
Detection Range	3.5 m	Max. $\lvert\bar{v}_q\rvert$ Input	1 m/s

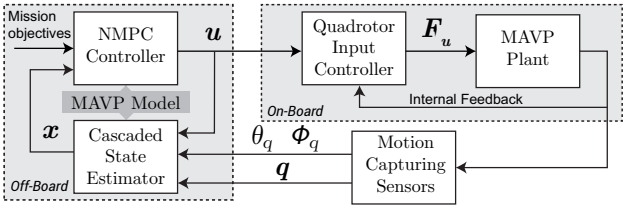


Fig. 5: Control system framework with the off-board NMPC controller and state estimator sharing a MAVP model, and the on-board MAVP system. Quadrotor input controller performs closed-loop low-level control. External motion capturing produces measurements necessary for state estimation.

based network. For simulation and controller design, Runge-Kutta 2nd order discretisation of the system model is used resulting in sufficient run-time stability and performance. The implemented NMPC cost weights are given in Table III.

D. Cascaded Kalman Filter State Estimator

Kalman Filtering (KF) based state estimation of $\boldsymbol{x}$ is performed using the system process model, and MCS based sub-millimetre measurements of the quadrotor and load's pose in $SE(3)$ [31]. As the Parrot Bebop's standard interface lacks a high frequency output of internal sensor measures to off-board clients, they were unusable for our state estimation routine. The MAVP system for which we present process models in Section II-B comprises (i) a quadrotor specific input controller, and (ii) the general non-linear MAVP system. We distribute the state estimation over two KFs permitting individual treatment of the subsystems and maintaining modularity.

A Linear KF is used to estimate state $\boldsymbol{x}_c$ of the quadrotor input model (2). The MCS provides measures of real pitch θ_q and roll ϕ_q for estimating states $x_{\theta_1}, x_{\theta_2}, x_{\phi_1}, x_{\phi_2}$. Lacking necessary measurements of the vertical control force F_q, states $x_{F,1}, x_{F,2}$ are only predicted without performing the KF measurement update step. Similar to [32], a non-linear Unscented KF is used to estimate the MAVP states $\boldsymbol{x}_q$ of the system model (14). Measures for state variable $\boldsymbol{q}$ are directly reconstructed from MCS data using the kinematic relations introduced in Section II-B.3. Using the process model, the Unscented KF is primarily tuned to provide noise reduced estimates of the time derivatives of $\boldsymbol{q}$ in $\boldsymbol{x}_q$. The Unscented KF directly uses the non-linear and observable process model for state prediction without performing linear approximations as traditionally required by the Extended KF thus usually improving the prediction accuracy [33]. The full cascaded KF based estimator design is schematised in Fig. 6.

V. SIMULATION STUDY

We showcase our method's scalability, robustness and performance in simulated studies. The following metrics are

TABLE III: Implemented default cost term weights

Navigation w_{nav}	1.0	Inputs w_{in}	0.01
Potential Field w_{pf}	1.2	Swing Angles w_{swing}	0.001
Steering w_{steer}	0.05	Slacks w_{slack}	10000

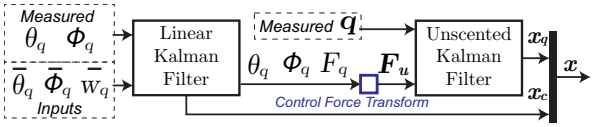


Fig. 6: Cascaded state estimation using Linear and Unscented KF and measured MCS data. Linear KF estimates $\boldsymbol{x}_c$ based on the quadrotor input model, measured true pitch, roll and inputs. Unscented KF estimates $\boldsymbol{x}_q$ based on MAVP model, measured MAVP configuration $\boldsymbol{q}$ and control force inputs computed by (3) using the Linear KF outputs.

used; let the system's *distance-to-goal* be defined as

$$d_{\text{goal}} = |\boldsymbol{p}_q - \boldsymbol{p}_{\text{goal}}| \ , \tag{44}$$

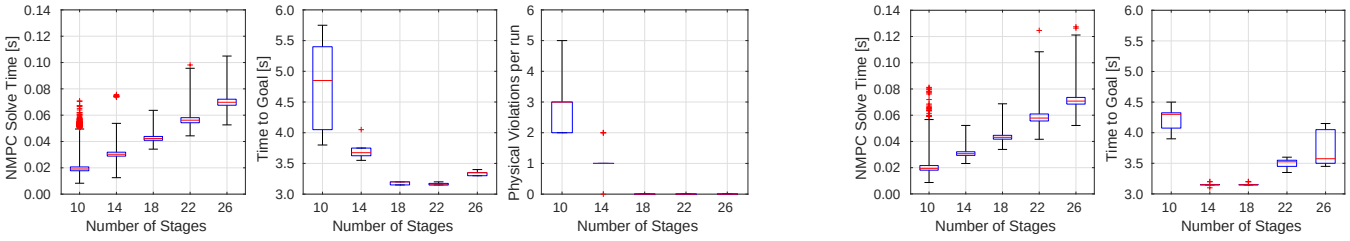
then the *time-to-goal* is the elapsed run-time such that d_{goal} strictly remains below 0.2 m.

A. Scalability of The Optimisation Problem

1) Scaling with Number of Planning Stages: The quadrotor, with a randomised initial swing θ_l, ϕ_l <10°, starts at $(-2.0, 0.0, 1.1)$ with the dynamic obstacle at $(2.0, 0.0)$. A collinear position swap is performed with the obstacle moving at 0.5 m/s such that the head-on paths critically tests the predictive planning behaviour. The number of planning stages N is increased from 10 by 4 to 26. Using $\Delta t = 0.05$ s, default cost weights and with assistive steering enabled or disabled (with 8 pre-defined steering directions n_d), we perform 16 runs per case.

Results in Fig. 7a show the scaling of the NMPC solve time with N using no assistive steering. It shows an increasing trend which is expected as every additional stage results in an increase of the optimisation variable given by $\tilde{z} \in \mathbb{R}^{22N+19}$ increasing the problem dimension. The major drawback of a low N are physical violations as the system is too late to respond to the incoming obstacle, leading to a collision. The late response means the attempted aggressive evasive behaviour causes the system to move far off-track, sometimes leading to workspace limit violations, and overall increasing the time-to-goal. With a higher N, collisions are averted with a quicker task completion by a smooth agile motion as depicted in Fig. 8a. However, with N=26, the time-to-goal increases as the planner increases the MAVP to obstacle separation lowering the potential field associated cost resulting in a more optimal route according to the objective function definition; this behaviour is tunable by the cost weights to achieve a different behaviour.

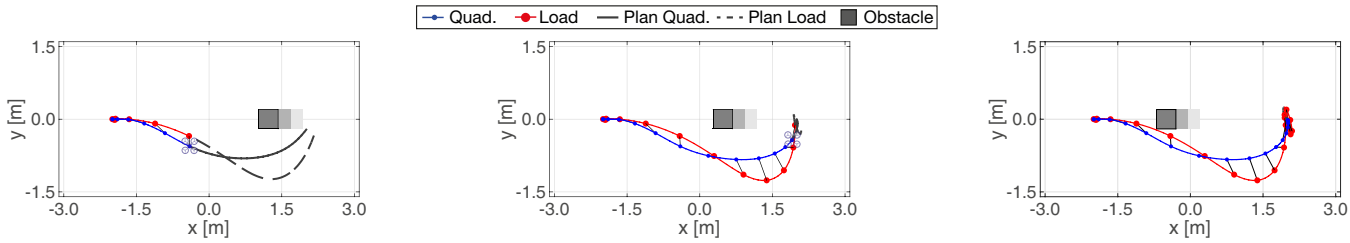
To address the underperformance with a low N, assistive steering is used to guide the MAVP towards obstacle-free regions without compromising on run-time performance. As shown in Fig. 7b, the NMPC solve times are minimally affected by enabling steering. Referring to Fig. 8b for the N=10 case, observe how the steering assisted trajectory is guided away from the obstacle resulting in a collision-free task completion. In general, application of the steering command over the entire planning length makes the path more conservative thus increasing the time-to-goal especially for the higher N when compared to no steering. The benefits of steering are greater for low stage (N) counts where the guidance is used to improve local planning. A low N



(a) No assistive steering

(b) With assistive steering (no violations)

Fig. 7: Simulated NMPC solve time, time-to-goal and physical violations (collisions or breach of workspace limits) per run with increasing planning stages, assistive steering enabled or disabled, and 16 runs per case.



$t = 1.5\,\text{s}$ $t = 3.0\,\text{s}$ $t = 5.0\,\text{s}$; task already completed

(a) $N = 26$ and no assistive steering; the long horizon pre-emptively guides the system away from the incoming obstacle's path.

$t = 1.5\,\text{s}$ $t = 3.0\,\text{s}$ $t = 5.0\,\text{s}$; task already completed

(b) $N = 10$ and with assistive steering; quadrotor's planned heading is steered towards obstacle free direction for avoidance.

Fig. 8: Point-to-point navigation with collinear dynamic obstacle; showing planned and executed trajectory and current quadrotor position. Top down view.

with steering is a viable method to maintain a reasonable run-time frequency and collision-free performance. In our simulation, we always used default cost weights, however, by fine-tuning the weights to accommodate a shorter/longer prediction length, better performance may be realised.

Based on the results and qualitative observations, $N = 18$ was used for all subsequent studies as it balances run-time and planning performance. As mentioned in Section III-B, assistive steering is limited to planar motion, therefore, to maintain the applicability of our results to spatial motion we have disabled steering for all subsequent studies.

2) Scaling with Number of Dynamic Obstacles: We perform a navigation task from $(-2.5, -1.0, 1.0)$ to $(2.5, 1.0, 1.0)$ amongst n_o randomly placed obstacles with randomised velocities $\leq 1\,\text{m/s}$. We increase n_o from 2 by 2 to 8 with $\Delta t = 0.05\,\text{s}$, N=18, default cost weights, no steering, and perform 16 runs per case.

Results in Fig. 9 indicate a positive trend in MPC solve time with n_o resulting from the additional cost and constraints introduced into the optimisation problem per additional obstacle. The time-to-goal shows an increasing spread with n_o as the obstacles are more likely to obstruct the system's path resulting in a lengthier route. In Fig. 10 we show one run demonstrating the MAVP's agile response

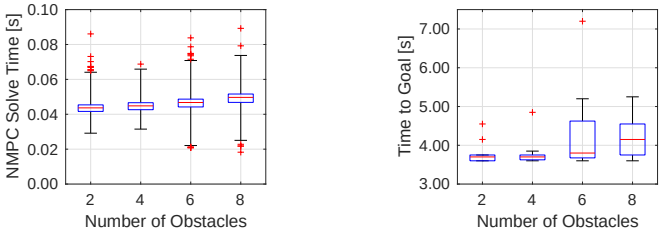
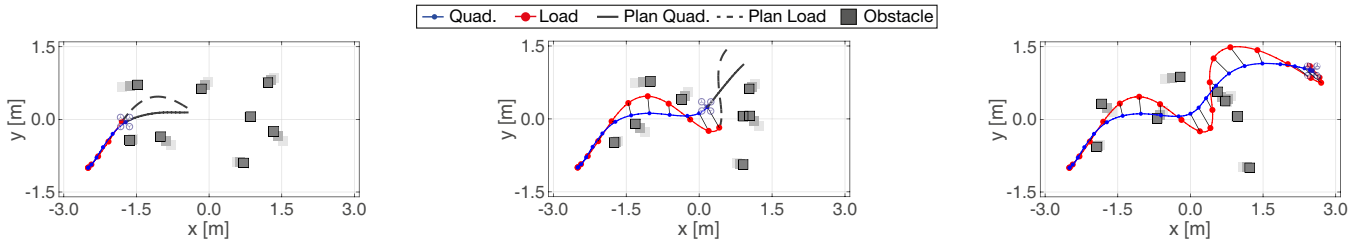


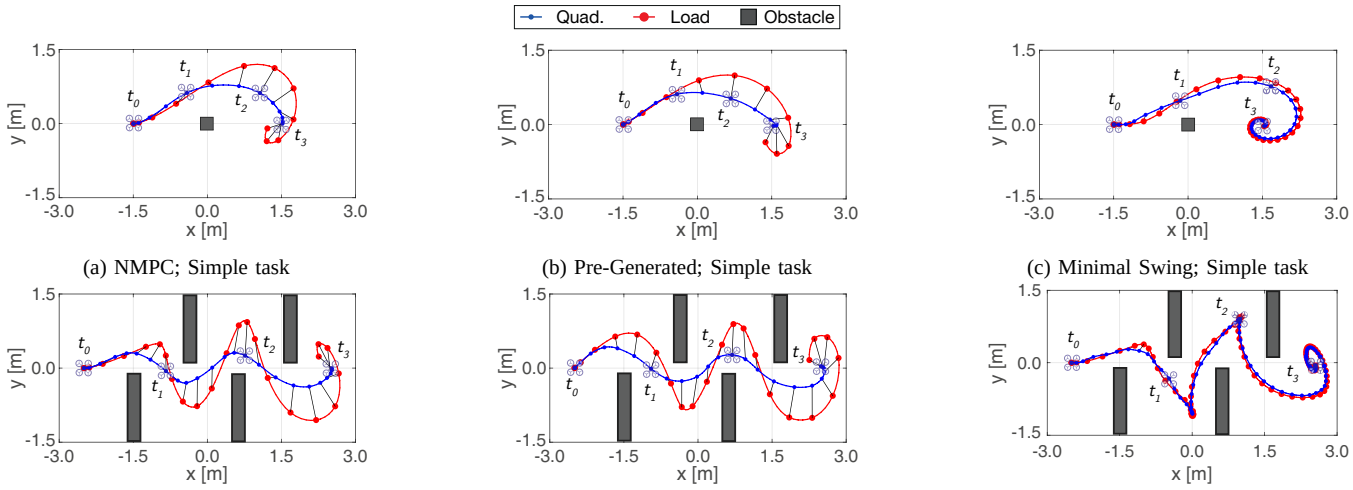
Fig. 9: Simulated NMPC solve time and time-to-goal with increasing number of dynamic obstacles using $N = 18$ and 16 runs per case. No violations/collisions occurred. Outlier for time-to-goal at 6 obstacles is for a run that temporarily entered deadlock resulting in a longer path.

amongst 8 dynamic obstacles. The outlier at six obstacles is the result of a temporary deadlock situation that is resolved by a lengthier planned route. As mentioned, NMPC is locally optimal, therefore, the deadlock situation arises from a local minimum of the objective function that occurs when several obstacles corner or obstruct the MAVP's path. In those cases, the planning may not be able to detour around the obstruction as the objective function over the planning length may only have a positive gradient. This local optimality is a limitation characteristic to local planning algorithms [34].



$t = 1.5\,\text{s}$ $t = 3.0\,\text{s}$ $t = 5.5\,\text{s}$; task already completed

Fig. 10: Point-to-point navigation using N=18 amongst 8 randomised dynamic obstacles moving at ≤ 1 m/s. The dynamically planned and agile executed trajectories of the quadrotor and payload are shown with the current quadrotor position indicated. Top down view.



(a) NMPC; Simple task (b) Pre-Generated; Simple task (c) Minimal Swing; Simple task

(d) NMPC; Difficult task (e) Pre-Generated; Difficult task (f) Minimal Swing; Difficult task

Fig. 11: Comparison of executed trajectories for manoeuvring around an obstacle (simple) and completing a slalom course (difficult) using NMPC with N=18, pre-generated and minimal swing planning and control. Observe that pre-generation leads to the smoothest, most optimal path resulting from the global planning scope. NMPC response resembles pre-generation and only initially reacts later to the presence of obstacles due to local planning. Minimal-swing response is sluggish as the turning motions are more suited for agile behaviour. Top down view with $t_3 > t_2 > t_1 > t_0$ shown.

B. Performance Comparison to Contemporary Approaches

We compare the total task completion time for three methods; (i) our NMPC (ii) pre-generated and (iii) minimal swing trajectory planning and control. A navigation task is performed for a simple static obstacle and a difficult slalom setup. For (i) we use N=18 and default cost weights, for (ii) we use our optimiser with N=200 for sufficient stages to pre-plan the entire trajectory and then simply track it, and for (iii) we use N=18 with a high swing cost $w_{\text{swing}} = 1$. We use $\Delta t = 0.05$, no steering and perform 4 repeated runs.

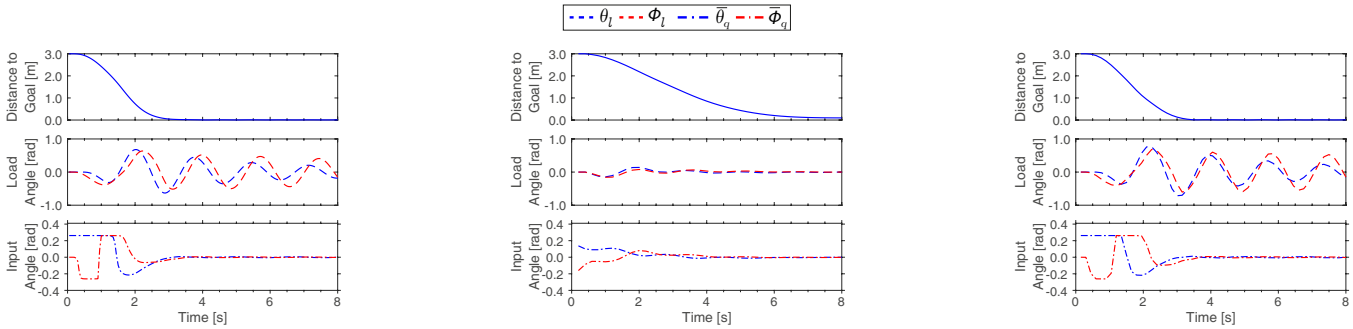
Table IV shows a comparison of the total task completion times (off-line computation and trajectory execution), and Fig. 11 depicts the executed trajectories using the three approaches. As expected, the pre-generated trajectory has the shortest time-to-goal for both tasks due to its highly optimised planning which requires large off-line computation times. The minimal swing approach results in sharp turns as the system accelerates and decelerates at the turning points making the motion slow and space inefficient as substantial effort is required to maintain a low swing angle through the turns. The NMPC based trajectory is marginally slower and less optimal than pre-generation, however, direct deployability means the simple task is completed within 2.65 s, a 48% reduction, and the difficult task within 5.35 s, a sizeable 65% reduction compared to pre-generation. Unlike pre-generation where a task-specific trajectory is generated, our NMPC method adapts to both tasks without any re-configuration. With increasing task complexity and duration, greater reductions can be realised making NMPC's scalability unparalleled. Furthermore, our NMPC method applies to dynamic scenarios.

TABLE IV: Comparison of NMPC to pre-generated and minimal swing approach for mean off-line computation, trajectory execution (time-to-goal) and total task completion time over 4 repeated runs.

Algorithm	Off-line [s]	Time-to-goal [s]	Total [s]
Simple task			
NMPC	N/A	2.65	2.65
Pre-Generated	2.91	2.25	5.16
Minimal Swing	N/A	7.10	7.10
Difficult task			
NMPC	N/A	5.35	5.35
Pre-Generated	10.54	4.85	15.39
Minimal Swing	N/A	18.55	18.55

C. Robustness to Change in Control Time Step and Lags

We demonstrate the robustness of our method by (i) increasing Δt from 0.05 s to 0.20 s to simulate a slower NMPC controller (on a less-powerful computer), and (ii) artificially adding a 0.1 s lag between NMPC generated input



(a) Δt=0.05 s, no lag (standard) (b) Δt=0.20 s, no lag (c) Δt=0.05 s, 0.10 s lag

Fig. 12: Distance-to-goal for a simple point-to-point navigation task, load suspension angles and NMPC generated pitch and roll inputs with $N = 18$ and an increased NMPC time-step from Δt=0.05 s to 0.20 s and control to execution time lag of 0.10 s. In (b), observe that NMPC compensates for the long time-step over which inputs are executed by reducing the input magnitudes resulting in a stable yet slower, less agile motion. Comparing (a) and (c), observe that even with a high time lag, the MAVP responds in a stable and agile manner.

commands and actually executing them. We use the simple task from Section V-B for analysis and N=18, default cost weights and no steering.

Comparing Fig. 12a and 12b with the different Δt, notice that NMPC automatically adjusts and reduces the computed input magnitudes for the longer time step resulting in a slower, less agile system; this is apparent from the distance-to-goal and load angles plots. With Δt=0.20 s, agile manoeuvres are inconceivable as large inputs over the long time-step would result in excessive accelerations with detrimental consequences on overall performance. Using the process model, NMPC is able to appropriately adapt its planning and control to the time-step size to realise the desired motion.

With a 0.10 s lag, notice in Fig. 12c that the system's distance-to-goal and load angles are similar to those with no lag in Fig. 12a. Due to our method's closed-loop setup, the true system behaviour is continually used to re-initialise the planning instance thus modelling errors do not accumulate. If pre-generated trajectories were used, any unaccounted lag would result in significant deviations of the real system from the planned path due to model mismatch. NMPC is therefore more robust to small modelling inaccuracies making it a safer and more practical method for real-world applications.

Increasing Δt further to 0.25 s and lag to 0.15 s destabilises the NMPC controller in simulation. We attribute this to several causes; first large time-steps used in combination with NMPC's discretised process model can result in prediction error divergence. Second, unmodelled time lags result in the prolonged execution of the large magnitude inputs required for agile flight resulting in excessive, destabilising accelerations; for short lags, the closed-loop control is able to prevent this from occurring. By acknowledging the presence of a long time-step and/or lag in the controller design, the method's prediction accuracy can be improved; this is future work.

VI. EXPERIMENTAL STUDY

We showcase complex, agile behaviour in static and dynamic experimental setups. The same distance/time-to-goal definitions as introduced in Section V are used. Videos of the experiments performed are at `https://youtu.be/2VtYjS3_6Gs`.

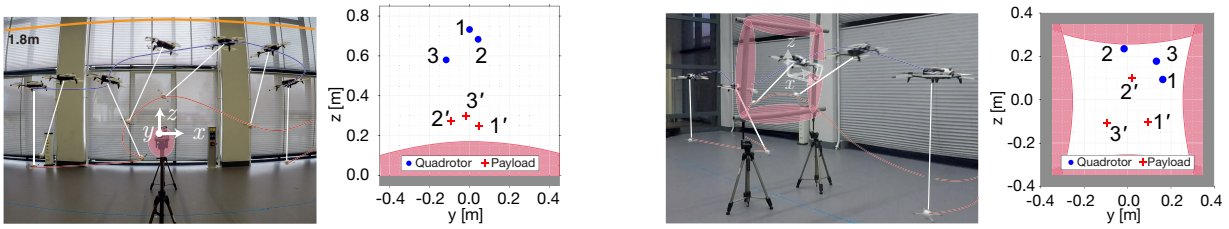
A. Agile Acrobatic Manoeuvres

Two complex agile manoeuvres are performed; (i) the MAVP must fly over a high bar at 0.95 m with a virtual ceiling of 1.8 m, and (ii) similar to [11] and [17], the MAVP must fly through a narrow 0.7×0.7 m opening. For both manoeuvres, three passes over/through the obstacle are performed in a rapid, successive and bidirectional manner. The tasks are impossible to execute without reducing the system's total vertical dimension (0.9 m when stationary) by swinging the load. The NMPC uses the real time-step, $N = 18$, default cost weights and no steering. For the narrow opening, the maximum pitch/roll input is increased to $20°$.

In Fig. 13 the two agile manoeuvres and the obstacle to MAVP clearance over all passes is shown. As the planning must excite the load's swing over a relatively short distance, large rapid inputs are commanded. Following the manoeuvre, the controller is able to stabilise the system at the goal position. As we do all computations online, and perform the passes in rapid succession, the clearances over the three passes differ while maintaining acceptable separation to the obstacle(s). For both manoeuvres, the entire system setup is identical with only the obstacles changed exemplifying our method's adaptability to different tasks.

Of the 48 tests performed over both tasks, 77% were successfully executed. In cases where the manoeuvre was not successful, the MAVP would either end up in a deadlock in front of the obstacle or make momentary contact with the obstacle. Flight was recoverable following the contact with only four tests where this was not the case. The likely culprits for the unsuccessful tests are the local planning approach, and inaccuracies in the model resulting in a sub-par prediction that leads to a discrepancy between the observed and planned motion. In a deadlock, the planning horizon is insufficient to plan a successful agile motion over/through the obstacle, however, increasing the horizon could rectify this. The obstacle contact was only observed when flying through the opening as the margins of error were small, so prediction inaccuracies have a noticeable effect on task performance.

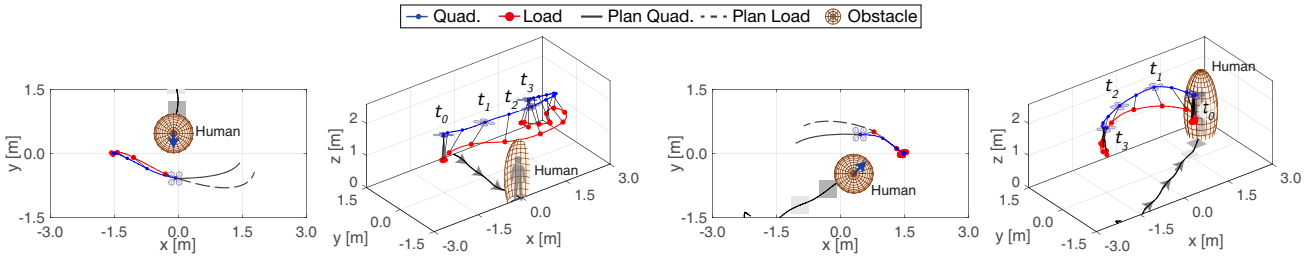
The setup was extended to the case of a moving high bar manoeuvre for agile dynamic obstacle avoidance (see video).



(a) Flight over high bar with virtual ceiling (orange line)

(b) Flight through narrow 0.7×0.7 m opening

Fig. 13: MAVP's agile acrobatic manoeuvre over a high bar and through a narrow opening. Snapshot of one pass shown (*image*). The quadrotor and payload clearance to obstacles (grey with pink enclosing ellipsoid) at the vertical obstacle plane (x=0 m) over three successive passes (*plot*) is shown and numbered 1,2,3 and 1′,2′,3′ for each pass. Observe the agile motion and that sufficient clearance is maintained over all passes.

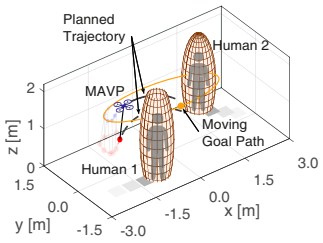


(a) Perpendicular path crossing (Human mean speed 1.0 m/s)

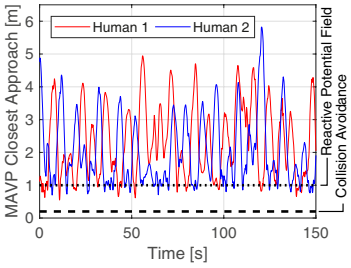
(b) Diagonal path crossing (Human mean speed 1.2 m/s)

Fig. 14: One planning instance (*left*) and full executed trajectory (*right*) with $t_3 > t_2 > t_1 > t_0$ for a perpendicular and diagonal MAVP-human path crossing. Smooth and agile planned and executed trajectories maintain a safe separation to the moving human obstacle (shown by the bounding ellipsoid).

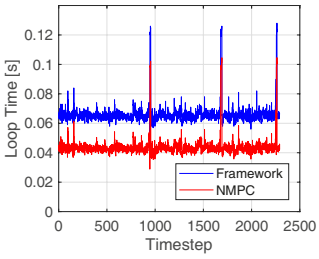
(a) Snapshot from experiment



(b) Schematic of MAVP, human obstacles and plan



(c) Separation between MAVP and ellipsoids representing humans



(d) Framework loop time includes the NMPC solve times

Fig. 15: MAVP follows an elliptical path avoiding two randomly walking human obstacles modelled as ellipsoids moving at a mean 0.5 m/s with max. 1.2 m/s. As shown in (c), the 0.2 m buffered collision avoidance limit is never violated. The system only intermittently enters the larger potential field ellipsoids with a 1.0 m buffer. In (d), observe the steady loop times for the full framework (mean 66 ms) which includes the NMPC controller (mean 44 ms); brief spiking arises from situations where significant re-planning was required.

B. MAVP Human Obstacle Avoidance

Obstacle avoidance performance is demonstrated amongst dynamic human obstacles with (i) test cases involving intersecting MAVP-human paths, and (ii) random motion in a shared MAVP-human space. The humans are represented by ellipsoids with buffers β_o=0.2 m, β_e=1.0 m, and are tracked to estimate their velocities for planning. The NMPC uses the real time-step, default cost weights, $N = 18$ and no steering. Note that we define the MAVP's closest approach to the human's associated ellipsoid as the smallest value of either the quadrotor to ellipsoid, or load to ellipsoid distance.

1) One Human with Crossing Paths: A human walks perpendicularly and diagonally on a path crossing the MAV performing a navigation task from $(-1.5, 0, 1.9)$ to $(1.5, 0, 1.9)$.

In Fig. 14 we show a snapshot and the full executed trajectory for both cases. Observe the MAVP's smooth, safe and agile execution of the task which includes the use of full spatial avoidance exploiting the available horizontal and vertical space around the obstacle (video shows this clearly). NMPC's predictive capability means load is actively swung away from the human's direction of motion to avoid a potential load-human collision. The minimum MAVP to human separations for the perpendicular and diagonal crossing task were 0.45 m and 0.61 m.

2) Two Humans Walking Randomly: Two humans walk for 150 s in random directions crossing the MAVP's path. The MAVP autonomously follows a goal position moving anti-clockwise with a 8 s period along an ellipse with semi-principal axes $(1.6,\ 0.4)$ m and a constant 1.4 m height.

In Fig. 15 we show a snapshot from our experiment alongside the system's closest approach to the humans and the framework/NMPC loop time. As shown in Fig. 15c, a safe distance is maintained by the MAVP from the humans with no collisions over the entire run; the minimum observed separation was 0.35 m from to the human's collision avoidance limit. Observe from Fig. 15d that the NMPC solve

time resembles the statistics obtained from the simulated study with 2 dynamic obstacles as shown in Fig. 9, therefore, the NMPC computation performance is preserved going from a simulation to the experimental setup. As the optimiser is initialised using the time-shifted previous solution, a roughly constant solve time is achieved. Spiking occurs when the optimiser's iterative solver requires more time to computed solutions which primarily occurs when considerable re-planning is required. Examples where we observed spikes included situations where the humans would inhibit the NMPC planner from feasibly planning a path to go to the goal position, or the MAVP would be trapped. The spikes only lasted one to two time-steps so observations showed the overall performance was not degraded. Specific to experiments is a mean 22 ms overhead (on top of NMPC solve time) associated with the framework's state estimation, communication and data parsing. The low overhead means controller's performance is not severely affected.

Thanks to its online and receding-horizon nature, our method can execute continuous manoeuvres and avoid dynamic obstacles. To the best of our knowledge, the experimentally demonstrated agile and safe MAVP manoeuvrability amongst humans in proximity is unprecedented. Our method is extendable to larger spaces with more humans/obstacles as we have already demonstrated in simulation with eight obstacles.

VII. CONCLUSION

In this paper, we presented a novel optimisation based unified motion planner and controller to accomplish online, closed-loop and agile flight of a Micro Aerial Vehicle slung payload system. We aptly formulated the optimisation objective function and constraints to achieve safe and collision-free flight in dynamic environments over various complex tasks including flying through a narrow opening and avoiding moving humans. With simulation and experimental studies we demonstrate the method's (i) scalability with the planning stages and the number of obstacles, (ii) robustness to different controller time-step durations and input execution lags, (iii) adaptability and repeatability over various complex tasks, and (iv) fast online performance in experimental conditions. For future studies we recommend the method's extension to non-rigid cables, improving the model's realism, accuracy and consequentially the NMPC prediction performance. Furthermore, a study involving variations of the model parameters would showcase the generality of the approach to different systems and setups. Also, due to our reliance on off-board NMPC control and motion capturing we limited our experiments to indoor spaces, however, with the controller frequency achieved off-board, we believe on-board computations would be feasible with hardware available today. Combining our method with contemporary obstacle detection, localisation and state estimation techniques could make urban MAVP operation a reality.

APPENDIX A
QUADROTOR ON-BOARD CONTROLLER

Figure. 16 shows an expanded schematic of the quadrotor's on-board controller. The pitch, roll input $\bar{\theta}_q$, $\bar{\phi}_q$ are tracked by the fast attitude controller resulting in longitudinal and lateral control forces F_x, F_y. From our observations, the Parrot Bebop quadrotor can perform level flight under a pitch/roll tilt suggesting the absence of an additional vertical force component. The quadrotor vertical velocity is stabilised by a controller based on reference input $\bar{w}_q$ resulting in vertical control force F_z trimmed for weight. All forces F are in the world East-North-Up inertial frame.

APPENDIX B
IDENTIFIED PARROT BEBOP 2 INPUT MODEL

The quadrotor pitch θ_q, roll ϕ_q and vertical control force F_q response to inputs pitch $\bar{\theta}_q$, roll $\bar{\phi}_q$ and vertical velocity $\bar{w}_q$ are identified using the MATLAB system identification toolbox. The linear second-order, state-space, black-box models, which we denote by $h_\theta,\ h_\phi,\ h_F$, are given by equations (45) (46) and (47) respectively;

$$\begin{aligned} \dot{\boldsymbol{x}}_{\boldsymbol{\theta}} &= \begin{bmatrix} -4.301 & -2.877 \\ 10.92 & -10.37 \end{bmatrix} \boldsymbol{x}_{\boldsymbol{\theta}} + \begin{bmatrix} -0.6893 \\ -16.32 \end{bmatrix} \bar{\theta}_q \\ \theta_q &= \begin{bmatrix} 1.763 & 4.586 \times 10^{-3} \end{bmatrix} \boldsymbol{x}_{\boldsymbol{\theta}} + \begin{bmatrix} 0 \end{bmatrix} \bar{\theta}_q \end{aligned} \tag{45}$$

$$\begin{aligned} \dot{\boldsymbol{x}}_{\boldsymbol{\phi}} &= \begin{bmatrix} -2.789 & -4.978 \\ 9.302 & -13.72 \end{bmatrix} \boldsymbol{x}_{\boldsymbol{\phi}} + \begin{bmatrix} -5.41 \\ -18.04 \end{bmatrix} \bar{\phi}_q \\ \phi_q &= \begin{bmatrix} 1.996 & 0.4657 \end{bmatrix} \boldsymbol{x}_{\boldsymbol{\phi}} + \begin{bmatrix} 0 \end{bmatrix} \bar{\phi}_q \end{aligned} \tag{46}$$

$$\begin{aligned} \dot{\boldsymbol{x}}_{\boldsymbol{F}} &= \begin{bmatrix} -6.767 & -6.546 \\ 3.031 & 0.311 \end{bmatrix} \boldsymbol{x}_{\boldsymbol{F}} + \begin{bmatrix} 38.75 \\ 1.841 \end{bmatrix} \bar{w}_q \\ F_q &= \begin{bmatrix} 0.310 & 2.03 \times 10^{-2} \end{bmatrix} \boldsymbol{x}_{\boldsymbol{F}} + \begin{bmatrix} -0.121 \end{bmatrix} \bar{w}_q \ . \end{aligned} \tag{47}$$

The states $\boldsymbol{x}_{\boldsymbol{\theta}} = [x_{\theta,1}, x_{\theta,2}]$, $\boldsymbol{x}_{\boldsymbol{\phi}} = [x_{\phi,1}, x_{\phi,2}]$ and $\boldsymbol{x}_{\boldsymbol{F}} = [x_{F,1}, x_{F,2}]$ are combined as $\boldsymbol{x}_{\boldsymbol{c}} = [\boldsymbol{x}_{\boldsymbol{\theta}}, \boldsymbol{x}_{\boldsymbol{\phi}}, \boldsymbol{x}_{\boldsymbol{F}}]$.

For system identification, we experimentally collected two datasets (estimation and validation) of the Parrot Bebop 2 response on a 5° amplitude 0.5 Hz square wave pitch/roll input over 15 s, and a 1 m/s pulse of width 1 s for the vertical control force. Table V shows the quadrotor input model's fit. We use NRMSE (MATLAB's definition) to facilitate comparison; a 100% NRMSE means a perfect fit.

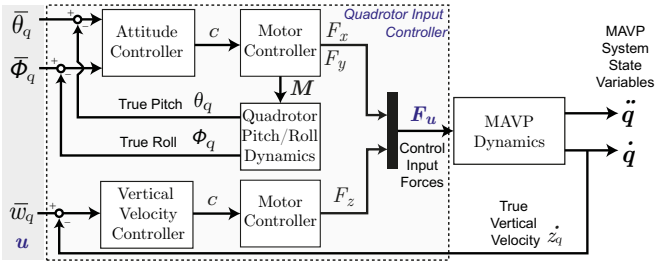


Fig. 16: On-board quadrotor input controller showing inner attitude and vertical velocity stabiliser loops. Motor inputs c are generated by controllers, resulting in forces F and moments M. Internally stabilised quadrotor attitude is perturbed by M; control force $\boldsymbol{F}_u$ affects MAVP dynamics.

TABLE V: Normalised Root Mean Squared Error (NRMSE) to estimation and validation dataset for empirically identified linear second-order quadrotor input control model.

System	Model fit as NRMSE [%]	
	Estimation dataset	Validation dataset
Pitch	94.59	93.81
Roll	91.09	89.09
Vertical Control Force	91.38	91.56

APPENDIX C
DERIVATION OF THE CLOSEST POINT OF APPROACH (CPA) OF A FINITE LINE SEGMENT TO AN ELLIPSOID

Consider an ellipsoid of dimensions (a, b, c), at position $\boldsymbol{p}_o$ and a parametrised finite line segment $\mathcal{L} = \{\boldsymbol{p} | \boldsymbol{p} = \boldsymbol{p}_q + s(\boldsymbol{p}_l - \boldsymbol{p}_q), s \in [0, 1]\}$ where $\boldsymbol{p}_q$ and $\boldsymbol{p}_l$ are the end-points. Let $[u, v, w]^\top \equiv \boldsymbol{p}_l - \boldsymbol{p}_q$ and $\boldsymbol{r}_{qo} = \boldsymbol{p}_o - \boldsymbol{p}_q$. Substituting $\mathcal{L}$ in the ellipsoid equation and expanding vectors into x, y, z components, we approximate the signed line to ellipsoid distance function by

$$d(s) = \frac{(x_{qo} + su)^2}{a^2} + \frac{(y_{qo} + sv)^2}{b^2} + \frac{(z_{qo} + sw)^2}{c^2} - 1 \ . \quad (48)$$

Minimising (48) with respect to s, we get the closest point to the ellipsoid along the infinite expansion of line $\mathcal{L}$

$$\hat{s} = \arg\min_{\hat{s} \in \mathbb{R}} d(s) = -\frac{x_{qo} u b^2 c^2 + y_{qo} v a^2 c^2 + z_{qo} w a^2 b^2}{u^2 b^2 c^2 + v^2 a^2 c^2 + w^2 a^2 b^2} \ . \quad (49)$$

Then on the finite line segment we obtain the Closest Point of Approach (CPA)

$$\boldsymbol{p}_c^* = [x_q + s^* u, y_q + s^* v, z_q + s^* w]^\top$$

where $s^* = \min\{\max\{\hat{s}, 0\}, 1\}$.

REFERENCES

[1] A. Ryan and J. Hedrick, "A mode-switching path planner for UAV-assisted search and rescue," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 1471–1476.

[2] T. Lee, "Geometric Control of Quadrotor UAVs Transporting a Cable-Suspended Rigid Body," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 255–264, jan 2018.

[3] L. S. Cicolani and G. Kanning, "Equations of motion of slung-load systems, including multilift systems," National Aeronautics and Space Administration, Moffett Field, Tech. Rep., 1992.

[4] I. H. B. Pizetta, A. S. Brandao, and M. Sarcinelli-Filho, "Modelling and control of a PVTOL quadrotor carrying a suspended load," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. Denver: IEEE, jun 2015, pp. 444–450.

[5] Y. Feng, C. A. Rabbath, and C.-Y. Su, "Modeling of a Micro UAV with Slung Payload," in *Handbook of Unmanned Aerial Vehicles*, K. P. Valavanis and G. J. Vachtsevanos, Eds. Dordrecht: Springer Netherlands, 2014, pp. 1257 – 1272.

[6] M. Bisgaard, A. Cour-Harbo, and J. Dimon Bendtsen, "Adaptive control system for autonomous helicopter slung load operations," *Control Engineering Practice*, vol. 18, no. 7, pp. 800–811, 2010.

[7] I. Palunko, R. Fierro, and P. Cruz, "Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload: A dynamic programming approach," in *Proceedings - IEEE International Conference on Robotics and Automation*. Saint Paul: IEEE, 2012, pp. 2691–2697.

[8] J. Trachte, F. Gonzalez, and A. McFadyen, "Nonlinear model predictive control for a multi-rotor with heavy slung load," in *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings*. Orlando: IEEE, 2014, pp. 1105–1110.

[9] I. Palunko, P. Cruz, and R. Fierro, "Agile load transportation : Safe and efficient load manipulation with aerial robots," *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 69–79, 2012.

[10] P. Foehn, D. Falanga, N. Kuppuswamy, R. Tedrake, and D. Scaramuzza, "Fast Trajectory Optimization for Agile Quadrotor Maneuvers with a Cable-Suspended Payload," in *Proceedings of Robotics: Science and Systems*. Boston: RSS Foundation, 2017, pp. 1–10.

[11] K. Sreenath, N. Michael, and V. Kumar, "Trajectory generation and control of a quadrotor with a cable-suspended load-A differentially-flat hybrid system," in *Proceedings - IEEE International Conference on Robotics and Automation*. Karlsruhe: IEEE, 2013, pp. 4888–4895.

[12] S. Tang and V. Kumar, "Mixed Integer Quadratic Program trajectory generation for a quadrotor with a cable-suspended payload," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 2216–2222, 2015.

[13] I. Palunko, A. Faust, P. Cruz, L. Tapia, and R. Fierro, "A reinforcement learning approach towards autonomous suspended load manipulation using aerial robots," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 4896–4901.

[14] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia, "Learning swing-free trajectories for UAVs with a suspended load," in *Proceedings - IEEE International Conference on Robotics and Automation*. Karlsruhe: IEEE, 2013, pp. 4902–4909.

[15] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia, "Automated aerial suspended cargo delivery through reinforcement learning," *Artificial Intelligence*, vol. 247, pp. 381–398, jun 2017.

[16] O. Brock and O. Khatib, "Real-time re-planning in high-dimensional configuration spaces using sets of homotopic paths," *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, no. April, pp. 550–555, 2000.

[17] C. De Crousaz, F. Farshidian, and J. Buchli, "Aggressive Optimal Control for Agile Flight with a Slung Load," in *IROS Workshop on Machine Learning in Planning and Control of Robot Motion*. Chicago: IROS, 2014.

[18] H. Kim, D. Shim, and S. Sastry, "Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, vol. 5. Anchorage: IEEE, 2002, pp. 3576–3581.

[19] A. Tzes, G. Nikolakopoulos, and K. Alexis, "Model predictive quadrotor control: attitude, altitude and position experimental studies," *IET Control Theory & Applications*, vol. 6, no. 12, pp. 1812–1827, 2012.

[20] F. Gonzalez, A. Heckmann, S. Notter, M. Zürn, J. Trachte, and A. McFadyen, "Non-linear model predictive control for UAVs with slung/swung load," in *ICRA Workshop on Aerial Robotics Manipulation and Load Transportation*. Seattle: ICRA, 2015.

[21] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm: IEEE, may 2016, pp. 1398–1404.

[22] T. Naegeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time Motion Planning for Aerial Videography with Dynamic Obstacle Avoidance and Viewpoint Optimization," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1696 – 1703, 2017.

[23] K. Klausen, T. I. Fossen, and T. A. Johansen, "Nonlinear control of a multirotor UAV with suspended load," in *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*. Denver: IEEE, 2015, pp. 176–184.

[24] I. Stanculeanu and T. Borangiu, "Quadrotor Black-Box System Identification," *International Journal of Mechanical and Mechatronics Engineering*, vol. 5, no. 6, pp. 1025–1028, 2011.

[25] A. Y. Uteshev and M. V. Goncharova, "Point-to-ellipse and point-to-ellipsoid distance equation analysis," *Journal of Computational and Applied Mathematics*, vol. 328, no. January, pp. 232–251, jan 2018.

[26] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[27] A. Zheng and M. Morari, "Stability of model predictive control with mixed constraints," *IEEE Transactions on Automatic Control*, vol. 40, no. 10, pp. 1818–1823, 1995.

[28] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[29] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, jun 1991.

[30] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, vol. 0, no. 0, pp. 1–17, 2017.

[31] N. Point, "Optitrack-optical motion tracking solutions." (http://optitrack.com/), 2009, [Website. Accessed Nov. 10 2017].

[32] M. Bisgaard, A. Cour-Harbo, and J. D. Bendtsen, "Full State Estimation for Helicopter Slung Load System," *Proceedings of AIAA Conference on Guidance, Navigation, and Control*, no. August, pp. 1–15, 2007.

[33] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Proc.SPIE 3068*, 1997, p. 12.

[34] S. M. LaValle, "Motion Planning," *IEEE Robotics & Automation Magazine*, vol. 18, no. 1, pp. 79–89, mar 2011.

Part II

Preliminary Study of the System Dynamics Model

Chapter 2

Introduction to the Preliminary Study

In recent years, there has been growing interest from the research community and industry alike in the Unmanned Aerial Vehicles (UAVs) potential of carrying payload/cargo. UAVs are a disruptive technology that can bring operational cost savings, enhanced functionality and outperform classical approaches to certain tasks including payload transportation (The Economist, 2012). UAV's relatively small form factor, agility and low upfront costs could result in widespread and simple deployment of UAVs for payload transport in areas that are inaccessible or dangerous for humans and/or conventional (aerial) vehicles. Current applications have seen UAVs being used in search and rescue (Ryan & Hedrick, 2005), package/cargo delivery (Jain, 2015) and construction (T. Lee, 2018). However, most applications have been limited to wide, open-spaces where the UAV is unhindered or confined to its own space.

Enabling UAVs to carry payloads in cluttered uncertain dynamic environments, such as a city, presents itself with many challenges. There are four high-level tasks that must occur to enable autonomous motion, namely *Perception*, to locate oneself, *Planning*, to have an objective, *Control*, to perform actions, and *Coordination*, to realise the objective (Pendleton et al., 2017). The focus of this entire research project will be to address the challenges related to path planning and control for collision/obstacle avoidance of UAV-swung Payload (UAVP) systems. Briefly stated, this thesis addresses the design of a closed-loop trajectory generation control architecture for a UAVP system through dynamic uncertain environments quickly and safely without complete environmental knowledge and with modelling imperfections. To achieve this final outcome, it is reasonable to start with a preliminary study of the UAVP dynamics as to understand the physical system characteristics and its physics. During this preliminary study, a thorough investigation of the UAVP system and its modelling was performed to garner insights about the system behaviour and fill knowledge gaps with simulation and experimental studies where necessary. The outcome of this study is a comprehensive system model and foundational framework to perform future simulation and experimental studies. The knowledge gained from preparing, setting up and experimenting with the hardware also provided insights into the possibilities and limitations of experimental studies as will be presented in this report. Summarising, the system modelling theory, simulation and

experimental setup, and results presented in this study establish the necessary groundwork for subsequent studies for the control of UAVP systems for collision/obstacle avoidance.

Chapter 3 introduces the preliminary study's content in detail, puts it into the context of the entire research project and presents the subsequent steps for completing the study. Chapter 4 provides an overview of literature relevant to the preliminary study performed. Chapter 5 presents the derivation of the UAVP system kinematics and dynamics using first-principles. Chapter 6 outlines the simulation and experimental methodology for verifying and validating the UAVP model. Additionally, details of the hardware and software used is presented for future duplicability. A discussion of the relevant results is presented in Chapter 7 with the intent of fully identifying, verifying and validating the UAVP dynamics model. Chapter 8 provides an overview of the research planning with an outlook for the remainder of the research project. Chapter 9 concludes and highlights the main research outcomes of the preliminary study.

Chapter 3

Research Outline

To put the research project into context, an overview of the research objective, questions and hypothesis is provided in Section 3-1. From this the main research themes are discussed in Section 3-2 identifying the research questions that are addressed in this preliminary study. In Section 3-3 an update of the latest research objective is provided.

3-1 Research Objective, Aim and Questions

The objective of the research thesis project is to:

> *'Demonstrate closed-loop collision-free planning, for an imperfectly modelled UAV-payload system, through dynamic uncertain environments by using Model Predictive Control (MPC) combined with Learning Based System Identification (LBSI)'*

To address and steer the thesis towards the research objective, several (sub-) research questions (RQ) are formulated that provide an indication of the knowledge required.

1. Can MPC be used for the closed-loop collision-free trajectory generation for a UAV-payload system?
   (a) Which theories are available for modelling the UAV-payload system dynamics while considering the limitation and assumptions made?
   (b) Which reactive collision avoidance methods have been researched in dynamic environments?
   (c) How should the MPC be setup such that the controlled system's motion is collision-free?
   (d) To what extent does simulation verify MPC's capability of closed-loop collision-free motion?

(e) To what extent do the simulation results translate to real-life experiments for validation?

2. Does the MPC with LBSI model show improved collision-avoidance performance in comparison to using the EOMs?
   (a) Which previous studies have explored the use of LBSI in the context of UAV-payload systems?
   (b) Which LBSI method is most suitable given the MPC control structure used?
   (c) How should the LBSI be trained such that it remains representative of the physical system?
   (d) To what extent do experimental results show collision-avoidance performance improvements made using LBSI?

The relevance of these research questions to the study are addressed in Section 3-2. The purpose of this research is to prove the following hypothesis. The research questions are used to guide the research towards addressing the hypothesis presented.

> *'The use of Model Predictive Control (MPC) combined with Learning Based System Identification (LBSI) enables closed-loop collision-free trajectory generation under system modelling uncertainties for UAV-Payload systems'*

3-2 Preliminary Study within Research Context

Three areas of knowledge are derived from this research objective where each will be progressively addressed during the studies conducted.

- UAV-Payload (UAVP) system dynamics modelling
- Model Predictive Control (MPC) for closed-loop collision-free planning and control
- Learning Based System Identification (LBSI) to learn for imperfectly modelled UAVP systems

This preliminary study contributes to the first area of knowledge; namely UAVP system dynamics modelling. This research will be necessary for the second knowledge area concerning the design of a model-based MPC controller. The performance of a model-based controller is dependent on the model's accuracy as control computations are derived from how the system is expected to respond based on predictions made using the model. The reason of deriving an accurate model becomes clear from understanding how MPC works.

Model Predictive Control (MPC) also referred to as *Receding Horizon Control (RHC)* is an optimal control theory in which system control inputs are optimised, with respect to costs and constraints, and executed in discrete time (Olsder, Woude, Maks, & Jeltsema, 2011). The cost function (usually quadratic) and constraints are carefully designed to promote a certain type of system behaviour that is feasible (constrained by physical limits). In MPC a constrained open-loop Optimal Control Problem (OCP) is solved over a set time horizon that generates a set of system inputs to achieve a locally optimal predicted system response. Only the first action from the generated optimal control sequence is performed after which

the entire time horizon is shifted by one time-step (recessed) and the process is repeated (J. H. Lee, 2011; Olsder et al., 2011). Due to this 'receding horizon' principle, closed-loop control is achieved as feedback is implicitly introduced by solving the OCP every time-step. For a more thorough explanation of MPC, refer to Appendix D.

Two time horizons are defined in MPC, the *control horizon* which is how far into the future the control inputs are to be optimised such that the predicted system response over the *prediction horizon* length, using the model, deviates the least from the reference response (J. H. Lee, 2011). The prediction horizon length drives the required model accuracy over sustained time periods as it require a model to predict the future UAVP states given only the actual measured initial state. It is expected that over longer time horizons, the predicted system response will increasingly deviate from the true response due to accumulation of errors. As the prediction horizon is generally in the order of 10^{-1} to 10^{0} seconds, the model only needs to capture the system's local behaviour accurately and over short time periods.

Having shown the relevance of this preliminary study for MPC, returning to the global research scope, this report addresses two research questions that were initially identified in the research planning. The first question requests the ways in which to identify and define the UAVP model while the second requests how to verify and validate the model.

- RQ1a *'Which theories are available for modelling the UAV-Payload system dynamics while considering the limitation and assumptions made?'*
- RQ1b *'To what extent do the simulation results translate to real-life experiments for validation?'*

To identify, verify and validate the UAVP system dynamics model, a preliminary study was performed using simulation and experiments results. The simulation was used to verify the derived system dynamics model and build a framework to perform simulated studies using the MPC high-level controller in subsequent stages of the research. Experimental data was collected for identification of unknown system dynamics and validation of the simulated responses. The multifaceted approach ensured that the obtained UAVP model was fully identified, verified and validated such that it would provide sufficient predictive performance for implementation in the model-based MPC.

3-3 Final Research Objective

Throughout the thesis, the research objective became clearer and focused. Following the preliminary study, it became apparent that adding the Learning Based System Identification on top of the closed-loop collision-free trajectory planning would be infeasible in the time available. Also, by concentrating on one aspect and evaluating the planning and control, a thorough and well-developed contribution could be made. Therefore, the research objective was reworded as follows;

> *'Demonstrate online, closed-loop, collision-free trajectory generation and control of a MAV-Payload (MAVP) system in dynamic environments using Non-Linear Model Predictive Control (NMPC)'*

Chapter 4

Literature Survey

The literature survey presents the state-of-the-art in concepts and techniques in the field of collision-avoidance for UAV control and planning combined with insights into modelling the Unmanned Aerial Vehicle - Payload (UAVP) dynamics. Within this scope, core concepts from trajectory planning, and UAVP system modelling and control are treated with reference to classical and contemporary techniques from literature and scholarly articles. The sections are structured to provide the readers with a core understanding of concepts diving into more detail, later on, to help substantiate the reason why the research proposed in this thesis is relevant. Note that to the best of the writer's knowledge, this is the state-of-the-art at the time of writing of August 2017. A condensed overview of this literature survey is presented in Appendix B.

Section 4-1 discusses the contemporary approaches to modelling the UAVP system highlighting the complexity and limitations of models that are necessary to design effective control techniques. Section 4-2 introduces core concepts from collision avoidance and highlights the difference between open- and closed-loop collision avoidance. Section 4-3 discusses contemporary control techniques for UAVP system with an extended treatment of Model Predictive Control (MPC) due to its relevance to this research's topic. Section 4-2 elaborates on collision avoidance algorithms as used for UAVP system with their implementation and results in empirical studies. Finally, Section 4-5 expands on adaptive control techniques by introducing learning based system identification schemes that support parametric uncertainties in the UAVP model.

4-1 UAV-Payload System Modelling

The modelling and control of aerial vehicles carrying suspended payloads was originally treated for helicopters (Cicolani & Kanning, 1992).The NASA Technical Paper from 1992 covered the Equations of Motion (EOMs) of slung-load systems including multi-lift systems (multiple vehicles) with different single/multiple suspension points with the intention of helicopter-like vehicles being the suspension point(s). The study helped in the understanding of aerial

vehicle assisted payload carriage and the disturbance effects that slung-payloads introduce to the carrying vehicle. However, the theory of helicopter-payload dynamics do not directly translate to UAVP dynamic systems; UAVs have fast, non-linear, unstable dynamics that are only complicated by introducing a swung payload resulting in external disturbances (Palunko, Fierro, & Cruz, 2012).

4-1-1 Types of Payload Attachment

When considering aerial vehicles such as a UAV with payload, there are two ways by which payload can be carried. The *stand-alone UAV* is the combination of all the components necessary for the UAV to fly independently; this includes the basic airframe and necessary battery resulting in the vehicle's Operating Empty Weight (OEW). The payload is defined as any additional, non-essential mass that is internally/externally attached to the UAV that does not serve any functional purpose related to flying the UAV. For the purpose of this research only externally attached payloads will be considered given that small-size UAVs will be used. An important assumption is that the payload can not directly contribute to additional thrust forces on the system will complicate the system dynamics significantly.

In literature, two problems for UAVP are studied namely, *grasped* loads whereby the payload is rigidly attached to the airframe (Palunko, Cruz, & Fierro, 2012) and more often *suspended* loads whereby the payload is (freely) swung under the vehicle (Feng, Rabbath, & Su, 2014). In the external grasping case, the payload can be seen as a protruding extension of the vehicle body itself that alters the vehicle's inertial and physical properties. Provided the payload is a rigid body, the grasped payload's position relative to the vehicle is unchanged during all flight manoeuvres (Palunko, Cruz, & Fierro, 2012). In the suspension case for a point load, the payload moves in $\mathbb{R}^3$ relative to the suspension point on the vehicle significantly increasing the complexity of modelling the dynamics.

With regards to this thesis' research, the first case of grasping a payload is not interesting as for the trajectory generation, the planning must only account for an expanded UAV shape with an altered dynamics model. In the suspended payload case, the trajectory generation problem will have to account for the UAV and relative payload dynamics which is more interesting.

4-1-2 Types of UAVs

Modelling the system dynamics involves understanding the carrier vehicle dynamics, therefore, a short introduction to UAVs is given highlighting the motivation for choosing the quadrotor type UAV. The Unmanned Aerial Vehicle (UAV) is a category of aerial vehicles that are piloted remotely or (semi-) autonomously with the absence of a human operator on-board. Within the UAV category, there are two main types of vehicles; commercially popular single/multi-rotor systems (e.g. helicopter, quadrotor/quadcopters, hexacopter, octocopters) and fixed-wing systems (e.g. General Atomics MQ-9 Predator (General Atomics Aeronautical, 2017)). More niche types of vehicles include the flapping-wing systems e.g. DelFly (De Croon et al., 2012), or hybrids of such systems (e.g. the ATMOS UAV Marlyn that combines a conventional quadrotor setup for take-off/landing with a fixed-wing flight mode (ATMOS UAV, 2017)).

This literature study will focus on modelling the UAV as a multi-rotor system and specifically a quadrotor.

A quadrotor is a four rotor Vertical Take-off and Landing (VTOL) vehicle. Quadrotors are able to perform stationary hovering which is in contrast to fixed-wing UAVs that require a continuous motion to generate lift which is not desirable especially in indoor conditions where space is limited. Consequently, a quadrotor is able to perform agile motions in confined spaces as it has the ability to fly without forward motion. The quadrotor is an under-actuated system (non-holonomic) as it has six Degrees of Freedom (DOF) and only four degrees of control/actuation (four rotors) (De Crousaz, Farshidian, & Buchli, 2014). Adding a suspended payload to the system increases the system's DOF by two (pendulum's polar coordinates in vehicle reference frame) and including a flexible rope by a further one (De Crousaz et al., 2014).

4-1-3 UAV-Payload Setup

The UAVP systems that have been treated in literature are either a single UAV or multiple UAVs cooperatively carrying a suspended payload. Both modelling cases will be addressed in this section.

Single UAV-Payload Setup

The basic UAVP setup consists of a single UAV with one suspended point payload. This problem is usually broken down into a two rigid-body problem where the bodies are coupled through interaction forces. The two body approach allows the UAV and payload dynamics to be treated separately coupling them through the interaction tensile force as shown in the left of Figure 4-1. Subsequently, cascaded control systems may be designed whereby the payload dynamics drive the quadrotor dynamics and vice versa as is done frequently in research (Gonzalez et al., 2015; Jain, 2015; Palunko, Cruz, & Fierro, 2012; Pizetta, Brandao, & Sarcinelli-Filho, 2015, 2016; Trachte, Gonzalez, & McFadyen, 2014). The single UAVP setup has been studied for both planar ($\mathbb{R}^2$) and Three Dimensional(3D) ($\mathbb{R}^3$) motion cases. The point load pendulum dynamics are analogous to a simple (in the case of planar motion) or spherical (three-dimensional case) pendulum attached to a moving suspension point. The Equations of Motion (EOM) are derived using the Newton-Euler or more often the Euler-Lagrange formulation due to more straightforward computation and resulting complexity of the EOMs.

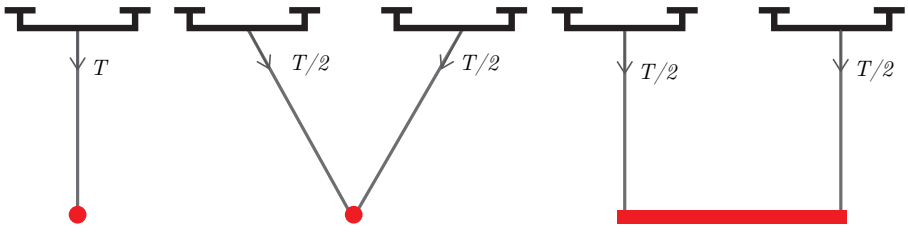


Figure 4-1: The UAVP configuration for single UAV - point load (left), multiple UAV - point (centre) and rigid (right) load.

Planar UAVP motion in the XZ plane studies the motion of a quadrotor with a cable suspended load (always in tension) where the system state/configuration will evolve in $SE(2) \times S^1$; for the 3D case, the state evolves in $SE(3) \times S^2$ (Sreenath, Michael, & Kumar, 2013). In (Pizetta et al., 2015), a dynamics model for the planar UAV-point payload model is derived using the Euler-Lagrange equations whereby the payload's effect on the vehicle is considered as external disturbances. The paper proposes the non-linear modelling and control strategy for the system through feedback linearisation for performing trajectory tracking. In the study the aim was not to make the payload follow a specific trajectory, but rather for a controller to be designed that allows the vehicle to counteract disturbances introduced by the pendulum such that the vehicle could perform trajectory tracking.

In (Feng et al., 2014), and in (Palunko, Cruz, & Fierro, 2012) the planar UAVP case is generalised to the 3D case for achieving perfect UAV trajectory tracking under payload disturbances as in (Pizetta et al., 2015). The EOMs are derived using the Euler-Lagrange formulation. In (Feng et al., 2014), the quadrotor UAV dynamics are also derived and interested readers can follow-up on the article. In (Feng et al., 2014), simulation studies were performed to show the UAVs tracking performance under the payload disturbance while (Palunko, Cruz, & Fierro, 2012) shows experimental studies with the inclusion of achieving swing-free payload motion. The studies demonstrate the model's validity by successfully being able to reject the payload disturbance forces and effectively control the vehicle.

In (Sreenath et al., 2013), a planning approach is proposed where the purpose was to design trajectories for the payload, rather than the vehicle, and then to control the vehicle (UAV) such that the payload would track its trajectory. This is in contrast to the works (Pizetta et al., 2015) and (Feng et al., 2014) where the payload was not actively controlled to follow a certain trajectory. Designing a payload trajectory is challenging as the pendulum dynamics are coupled to the vehicle dynamics and is only controlled through the UAV's inputs. Considering the planar XZ case, (Sreenath et al., 2013) establishes that the system is a *differentially flat hybrid system*. The approach in (Sreenath et al., 2013) enables the pendulum swing to be exploited for dynamic agile motions rather than just suppressing the disturbances introduced by the swing. Using this approach, the differential flatness property is used to generate a UAV trajectory that must be precisely tracked such that the payload follows the arbitrarily designed trajectory. The hybrid characteristic is required to address the case of wire slackening (coupling forces becoming zero) where the system dynamics switch; this will be explained in detail in Section 4-1-4. In (Sreenath et al., 2013), using the differentiable flatness property, it was successfully demonstrated that agile payload trajectory tracking is possible in the planar and 3D case.

The single UAVP system has been treated in (Trachte et al., 2014; Trachte, Toro, & McFadyen, 2015; Feng et al., 2014; Palunko, Cruz, & Fierro, 2012) and follows the same two rigid body approach to modelling the UAVP system arriving at same or similar models. The contributions of these papers relate more to the control aspect which will be discussed later. Later in Chapter 5, a complete derivation of the 3D UAV suspended point payload dynamics is provided based on the models presented in literature as referenced in this section.

Multiple UAV-Payload Setup

The payload suspension problem has also been extended to multiple UAVs cooperatively carrying a single payload. In this case, the problem is broken down into a $n + 1$ rigid body problem where n is the number of vehicles carrying the payload. For two UAVs carrying one payload, the configuration is similar to the schematic representation in Figure 4-1.

Extending from the single UAVP planar models, two UAVs cooperatively carrying a suspended point load was proposed in (Pizetta et al., 2016). As in their previous work (Pizetta et al., 2015), the purpose was to only design and track trajectories for the UAVs and reject all disturbances introduced by the payload. The tensile disturbance force on each UAV is computed by considering the payload suspension angle with respect to the UAV and the payload weight. In (Pizetta et al., 2016) also acknowledge the switching dynamics introduced by slackening cables and possible rope stretching. They also consider the suspension cable to be elastic and introduce a hybrid system model to account for the case when the cables become slack.

The idea of differential flatness for payload trajectory generation in (Sreenath et al., 2013) was extended to multiple UAVs cooperatively carrying a point or rigid body payload in (Sreenath & Kumar, 2013). In this case, instead of one UAV being affected by the tensile force induced by the suspended payload, it is split over the n vehicles depending on the current system configuration. The case of a rigid body payload suspended by n UAVs is also considered where the payload is suspended at n points on its body as in the right of Figure 4-1. Simulation and experimental studies showed that given the UAVs can precisely track their generated trajectories, the suspended payload will track its designed trajectory.

In (Bisgaard, 2008) and the publication (Bisgaard, Bendtsen, & Cour-Harbo, 2009), a comprehensive overview of modelling the single and multi-UAV payload system with full derivations of the dynamic EOMs and approaches for including wire slackening and drag effects is provided. For the purpose of this thesis' research, the concepts introduced in this section and the derivation provided in Appendix 5 for the single UAVP problem is sufficient for an initial literature study and further research can be performed should the scope of the research be expanded to the multi-UAVP problem.

4-1-4 Wire Slackening and Rigidity

As previously mentioned, the studies (Sreenath et al., 2013; Sreenath & Kumar, 2013) and (Pizetta et al., 2016) considered the switching dynamics introduced by wire slackening resulting in a *hybrid system dynamics model*. Initial studies of single UAVP systems including the work (Palunko, Cruz, & Fierro, 2012), (Feng et al., 2014) and others (Pizetta et al., 2015; Jain, 2015; Trachte et al., 2014) considered an always non-zero tensile force. For that modelling assumption to be true, the cable must always be taut (fully stretched) so the rope is comparable to a rigid link. Rope slackening fundamentally alters the system dynamics as the two bodies (UAV and payload) become independent in their motion as there is no coupling force, therefore, the uncontrolled payload enters a free-fall under gravity while the UAV is controllable. Consequently, the system EOMs are defined by a *hybrid system model* where the system EOMs switch between two equation sets; the non-zero and zero coupling tensile

force case. When considering slackening of the rope, the 3D dynamics no longer evolve in $SE(3) \times S^2$ but rather $SE(3) \times \mathbb{R}^3$ as the rope length becomes one DOF.

Alternative modelling approaches that do not require hybrid system models also exist. In (Dai, Lee, & Bernstein, 2014), a binary state of slack or taut is not used, but rather models the catenary curve formed by the rope going from taut to slack. This allows the EOMs to describe the continuous transition to a slack state eliminating the need for a hybrid system model. To achieve this, they model the suspension wire as a series of rigid mass links with an associated state q_i where each link's dynamics is modelled by an EOM. Combining the EOM for each link gives the system of EOMs describing the full wire's motion. This method is computationally expensive as the set of EOMs grows with increasing granularity in modelling the wire. Therefore, considering real-time motion planning, this method can more accurately capture the wire's behaviour at the high cost of increased computation.

4-1-5 Aerodynamic Drag

Besides wire slackening, aerodynamic drag has a prominent effect on the UAVP dynamics. Studies into UAVP motion have generally addressed only low-speed experiments such that the relative effect of aerodynamic drag when compared to the most significant external force, gravity, is negligible. However, when considering rigid body loads (large surface areas) and/or high-speed motions the drag force is significant as the force is proportional to the surface area perpendicular to the velocity vector and quadratically increases with the velocity magnitude (Bisgaard et al., 2009).

In (Bisgaard et al., 2009), the quadratic form of aerodynamic drag with velocity in $\mathbb{R}^3$ for the slung payload EOMs is considered. The quadratic form of drag for one axis is given by Eq. 4-1 where F_{D_x} is the drag force along axis x, C_D is the aerodynamic drag coefficient corresponding to the object's shape, ρ the air density, S_x is the surface area perpendicular to the velocity vector V_x.

$$F_{D_x} = C_{D_x} \cdot 0.5 \rho S_x V_x^2 \tag{4-1}$$

In (Klausen, Fossen, & Johansen, 2015), aerodynamic drag is also considered, however, due to the relatively slow motion of the system (even at what is considered high speed in indoor cases; around $\|V\| \leq 2m/s$), the quadratic function of V in this range can be estimated by the linear V function. Therefore, the drag force along axis x can be written in the form given in Eq. 4-2 where D_x is a combined drag coefficient that depends on the object's shape, the air density and surface area. As velocity, and not velocity squared, is usually a system state, the drag force equation can be directly included in the system EOMs using the kinematic relation (as provided in Appendix 5) that describes the payload position in polar coordinates with respect to the vehicle rather than Cartesian coordinates.

$$F_{D_x} = D_x \cdot V_x \tag{4-2}$$

4-2 Collision Avoidance Techniques

This section covers collision avoidance techniques for robotic systems in static, dynamic and uncertain (shared) workspace discussing the merits and drawbacks of different approaches to

collision avoidance. First, an introduction to generic motion planning is given followed by a discussion of global and local planning approaches which is further expanded upon with deliberative and reactive techniques.

4-2-1 Configuration Space

Enabling autonomous motion in obstacle rich environments requires collision-free path planning as the traversable space must be shared with other objects including vehicles. The concept of *configuration space (C-space)* is fundamental to understanding robot motion planning, it involves encoding the robot's configuration in a configuration vector q (Pan & Manocha, 2015). The *C-space* is the set of all attainable robot configurations; for example, for a 3D rigid-body robot with position and orientation encoded into q, *C-space* is the special Euclidean group $SE(3)$. For the 3D UAVP system with wire slackening, the configuration space is $SE(3) \times \mathbb{R}^3$ as previously discussed. Two subspaces can be identified in the *C-space* namely;

- C_{free} Free space - All configurations q that are not occupied by other object(s) in the *C-space*
- C_{obs} Obstacle Space - All configurations causing the robot to occupy the same configurations as another object/obstacle which is the C_{free} complement

The purpose of path planning is to compute a feasible, possibly optimal continuous curve in C_{free} from an initial q_I to goal q_G configuration as shown in Figure 4-2 (LaValle, 2006). Planning algorithms generally fall under two main approaches that use a discrete representation of the *C-space* connectivity (Pan & Manocha, 2015);

- *Combinatorial Planning* - "constructs structures in the *C-space* that discretely and completely capture all information needed to perform planning." (LaValle, 2006)
- *Sampling-based Planning* - "uses collision detection algorithms to probe and incrementally search the *C-space* for a solution, rather than completely characterizing all of the C_{free} structure." (LaValle, 2006)

For a comprehensive explanation of both concepts including examples refer to Appendix C.

4-2-2 Global and Local Planning

The purpose of this thesis is to perform closed-loop collision-free flight with a UAVP system and to achieve this it is imperative that the collision-free path planning is done in real-time. Path planning for real-time systems is generally performed using a hierarchical approach consisting of a *global* and *local* planner that perform planning at different frequencies to achieve real-time performance (Burgard, Stachniss, Bennewitz, & Arras, 2011).

- *Global planner* - Low frequency - "computes paths ignoring the kinematic and dynamic vehicle constraints" (Burgard et al., 2011)

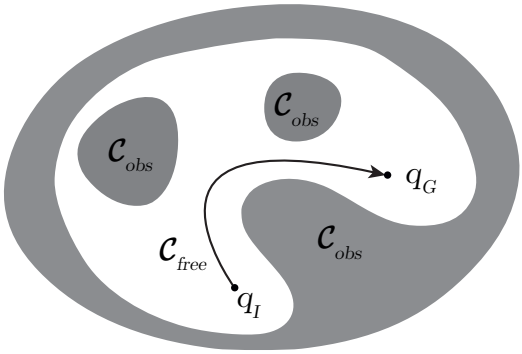


Figure 4-2: *C-space* showing obstacle free C_{free} and obstructed C_{obs} configurations and collision free path from q_I to q_G (LaValle, 2006)

- *Local planner* - High frequency - "accounts for the constraints and generates (sets of) feasible local trajectories (collision avoidance) " (Burgard et al., 2011)

The global planner uses a planning method, as introduced in Section 4-2-1, to generate an 'optimal' path from q_I to q_G. The path optimality is with respect to one or many user-defined objectives against which the resulting path is evaluated (LaValle, 2006). Depending on the objective(s) the path should satisfy, the global planner requires global information from the *C-space* to be able to produce a solution. Global information for a collision avoidance planner will generally include complete information of C_{obs} (Pan & Manocha, 2015). As the global planner must consider the global scope, the computational cost is high and any changes to the $C - Space$ usually requires a full re-computation of the global path (LaValle, 2011). This makes global planning impractical for highly dynamic environments as the computational resources required for (re-)planning means the algorithm can only run at low frequencies. Therefore, an off-line initial trajectory plan is pre-computed considering a static environment after which that is used as a reference for a local planner resulting in the hierarchical approach (Burgard et al., 2011).

A local planner is only concerned with a small subspace of the *C-space* that is locally relevant to the system's current configuration q_i. The local planner generates a path to get from q_i to q_{i+1} driven by the system dynamic constraints and pre-computed global path. As local planning only addresses a small sub-space of *C-space*, the computational cost is significantly reduced and the algorithm can run at much higher frequencies enabling its use for real-time planning (Burgard et al., 2011). As local planners run in-the-loop, any local changes to the *C-space* are accounted for in the local planning altering the pre-defined global path to accommodate any change. This quality is important and necessary for planning in highly dynamic environments; for avoiding collisions, the system is usually only concerned with changes in the immediate vicinity and not in the entire environment. Only local planning without global planning generally does not meet planning objectives as it is usually the case that the system must get from an initial to goal configuration that is far apart. To account for both terminal configuration, a global scope for planning is required. In the hierarchical setup, the local planner can complement the global planner improving the flexibility of the planning to unforeseen or changing environments.

As will be discussed in Section 4-4, the state-of-the-art for UAVP collision avoidance algo-

rithms generally leave out the local planner by pre-computing collision-free global paths in static environments, accounting for the system dynamics, and feeding the generated input sequence to a simple tracking UAV controller. For the remainder of the literature survey, an important distinction is made between open-loop and closed-loop collision-free trajectory generation. *Open-loop trajectories* are generated off-line (before performing any motion) and used as a reference trajectory to be precisely tracked by the robot. *Closed-loop trajectories* are generated online (while performing motion) and are continuously updated to serve as a dynamic reference trajectory that the robot must follow. Closed-loop planning is performed based on the hierarchical planning approach.

4-2-3 Deliberative and Reactive Collision Avoidance Planning

Planning for collision avoidance of robots in shared workspaces (single or multi-agent environments) has been approached using two approaches, the *deliberative* paradigm, or the *reactive* paradigm (Čáp, Gregoire, & Frazzoli, 2016). This section will refer to any robot or moving object in the workspace by the term *agent* for consistency with research papers addressing this field.

- *Deliberative* - Open-loop planning - **Globally** coordinated trajectories are pre-computed for all agents sharing the workspace from their initial to goal configuration. Consequently, the agents must precisely track these trajectories in space and time for the resulting motion to be collision free (Čáp et al., 2016). This problem quickly becomes intractable with an increasing number of agents and more complex environments.
- *Reactive* - Closed-loop planning - **Locally** resolves collisions by an agent observing the immediate surroundings and re-planning the immediate locally relevant trajectory (Čáp et al., 2016).

Deliberative

Deliberative collision avoidance theories postulate that collision-free motion is achieved given that trajectory plans are precisely followed in space and time (Čáp et al., 2016). Additionally, to pre-compute the globally coordinated trajectories the intended initial and goal configurations for all agents, their dynamics and full environmental knowledge must be known a priori which is impractical in many cases. For example, only considering homogeneous agents, all agent dynamics are equal, however, for non-homogeneous agents, the problem complexity can become intractable. To address this, heuristic approaches including prioritised planning are used whereby the path for agents are planned in decreasing order of priority (Čáp, Novak, Kleiner, & Selecky, 2015). In this approach, an agent with a given priority rank only considers the paths of all agents ranked higher in its planning. So, the highest priority agent path is first computed not considering any other agents' intentions. The second highest priority agent path is then computed considering the first agent's path for collision avoidance and so on. Given that the algorithm is able to compute feasible paths for all agents, the resulting globally coordinated trajectories are guaranteed to be collision-free.

Including unpredictable agents (such as a human) can make global coordinated planning impossible as it cannot be said that the unpredictable agent precisely follows their trajectory as

assumed in deliberative planning (Čáp et al., 2016). The assumption of a precise plan execution on which deliberative planning is based makes it impractical for real-world applications with dynamic, uncertain environments.

To handle unforeseen obstructions in deliberative planning, an *ALLSTOP* method can be used where the planning must be paused until the obstruction clears. For example, if any event obstructs an agent from proceeding with their pre-planned trajectory, all agents in the system must stop (pausing the planning). Once the event has passed, the plan can be resumed, with the planning still valid yet just shifted in time by the pause. However, if only the obstructed agent stops, the coordinated plan is no longer valid as each agent has an assigned trajectory in space and time; this could lead to potential collisions. In (Čáp et al., 2016), a strategy is introduced that is able to scale the planning time component for executing the coordinated trajectories to account for unpredictable changes in the environment. The strategy involves discretising the pre-planned collision-free trajectories by time into way-points after which conflicting way-points in space (not time) are identified. During the plan execution, if one agent stops, the remaining agent proceed with their trajectories. Once the stationary agent starts moving again the online planning will stop other agents as necessary such that no agent ever reaches a conflicting way-point in space. This strategy is a step towards including uncertainties in the environment, however, initially finding globally coordinated trajectories remains a computationally expensive and sometimes impossible task.

Reactive

In collision avoidance, acknowledging that the agents with which the workspace is shared are also decision-making, and therefore unpredictable, changes the required collision avoidance approach. If an agent A considers all other moving agents simply as moving obstacles, the collision avoidance scheme of A does not consider that the other agent will also re-plan based on A's motion. This can result in oscillatory behaviour and deadlock situations as agents wait for each-other (Berg, Guy, Lin, & Manocha, 2009). Therefore, the theory postulates that agents must be *reactive* to each-other in order to perform mutual collision avoidance.

A popular and widely used *reactive* collision avoidance method for multi-agent environments utilises the concept of *Velocity Obstacles (VO)* introduced in (Fiorini & Shiller, 1998). In VO, the complete set of feasible and physically attainable robot velocity inputs for each robot is partitioned into a subset that will result in a collision and its complement valid for a given time horizon. The complement is identified by considering the velocities of all objects and agent that are or will be reachable in the specified time frame for which the VO is computed. This VO is continuously re-computed to enable dynamic collision avoidance (Fiorini & Shiller, 1998). The limitation of VO is that collision-free motion is only guaranteed under specific conditions. For example, the VO space may become too restrictive for certain vehicles such as cars that are not as agile as a UAV resulting in no possible velocities that can be executed. In (Berg et al., 2009), the VO idea is extended to Optimal Reciprocal Collision Avoidance (ORCA) that includes optimal reciprocity that can always guarantee collision-free navigation even in highly dense environments. The agent is able to independently (without inter-agent communication) compute an optimal velocity to avoid collisions with all relevant agents over a fixed planning time frame (Berg et al., 2009). This property is very useful for collision-free navigation in multi-agent and obstacle rich environments where all the agents themselves are

also decision-making. Reciprocity also shares the collision avoidance task making the VO space less restrictive on an individual vehicle. However, reactive collision avoidance methods such as ORCA are prone to deadlocks where two or more agents wait for each other resulting in no conflict resolution (Čáp, Vokínek, & Kleiner, 2015).

Artificial potential fields (APF) as introduced in (Khatib, 1986) are another popular approach to reactive collision avoidance due to their conceptual simplicity and favourable characteristics for collision avoidance in continuous space. Each agent/obstacle in the environment generates a virtual repulsive force that drives agents away from each other and any static/moving obstacles (Shim, Kim, & Sastry, 2003). The closer two objects are, the stronger the virtual repulsive force for both objects so indirectly there is reciprocity in the collision avoidance behaviour similar to ORCA (Shim et al., 2003). Following from the superposition principle, the effect of all APFs originating from all objects in *C-space* are superimposed to give a resultant APF that causes the agent to move away from all other objects. However, APF approaches are prone to local minima, live-lock and deadlock situations when troughs (local-minima) are formed in the virtual potential fields (Shim et al., 2003). For example, consider two agent approaching and trying to pass each other in a narrow corridor; they will hinder each-others progress due to the repulsive force resulting in a deadlock situation. Also, no hard guarantees of a collision-free motion is provided by APF as the virtual repulsive force simply guides the agent's desired velocity unlike in ORCA where a set of guaranteed collision-free velocities are generated. To counteract such a situation, In (Kamel, Alonso-Mora, Siegwart, & Nieto, 2017), a hard constraint is introduced for the planned trajectories that guarantees no collision can occur by including a required minimum separation between objects. This approach has been successfully demonstrated in experiments performed in the work (Kamel et al., 2017) with two drones performing inter-collision avoidance. In (Naegeli, Alonso-Mora, Domahidi, Rus, & Hilliges, 2017) a collision avoidance involving a single/multiple drones with moving human obstacles is successfully demonstrated. In both studies, an optimal controller was used in combination with the APF collision technique to successfully demonstrate collision-free motion for UAVs provided that the position and velocity data from the agents/obstacles are available.

The discussion on deliberative and reactive collision avoidance approaches makes it clear that for performing collision avoidance in highly dynamic and uncertain environments, reactive methods are more suitable and tractable. The book (LaValle, 2006) provides an excellent overview of many other approaches to planning and collision avoidance, however, for brevity only the ideas that will be relevant to this thesis are presented.

4-3 Control Techniques for UAV-Payload Systems

The previous section covered the modelling of UAVP dynamics and introduced collision avoidance techniques. Control techniques that have been tested in simulation and experiments as presented in the literature are discussed in this section. A brief introduction to the UAV low-level system is given followed by a discussion of commonly used control techniques for UAVP systems. Control techniques to handle the UAVP hybrid system model and modelling uncertainties are also treated followed by a discussion of interesting vision-based techniques for

payload state estimations. Finally, a discussion on high-level control approaches for collision avoidance of UAVP systems is introduced.

4-3-1 UAV Low-Level Control

As low-level attitude control is not the primary objective of UAVP control design, it is generally assumed that there is low-level UAV control system available onboard. The high-level planning based controller generates system inputs u for the low-level controller as to achieve a certain objective such as trajectory tracking (Palunko, Fierro, & Cruz, 2012; Sreenath et al., 2013; Klausen et al., 2015; T. Lee, 2018). Figure 4-3 shows a simple schematic of the division of high and low-level control as presented in literature (Gonzalez et al., 2015; Trachte et al., 2014). The control inputs given to the quadrotor are generally the required thrust (to move position) and torques (to change orientation) in order to move the vehicle in $SE(3)$ space (Mahony, Kumar, & Corke, 2012).

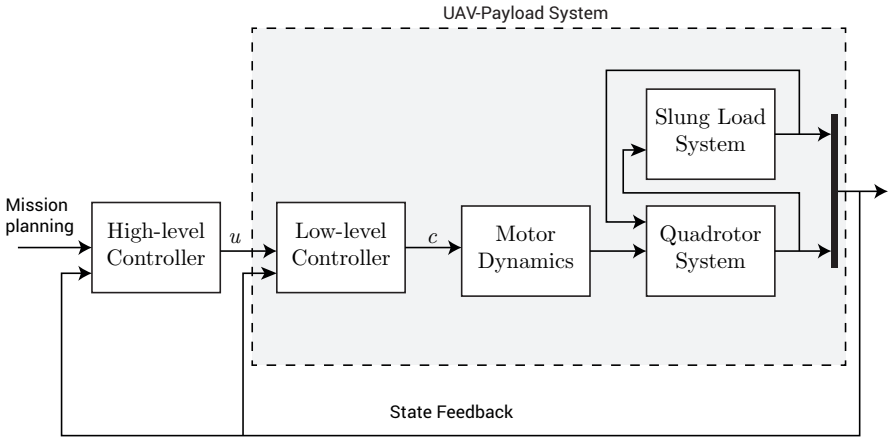


Figure 4-3: Schematic control architecture for UAV-payload system control

The research presented in this thesis treats only the design of the high-level controller. Therefore, the controllers discussed in the following sections discuss trajectory controllers rather than low-level attitude controllers for the UAV. For modelling the lower level systems (for simulation), simplified models are available and interested readers can refer to Mahony *et al.* work on multi-rotor aerial vehicle modelling, estimation and control in (Mahony et al., 2012).

4-3-2 Proportional-Integral-Derivative (PID) Control

Proportional-Integral-Derivative (PID) control is a classical feedback control technique for continuous dynamic systems where the objective is error minimisation with respect to a set-point/reference. PID control techniques have been around for many years and their simplicity enables high frequency computations for real-time applications. PID control has been used in UAVP systems to achieve trajectory tracking tasks provided a collision-free reference trajectory is pre-computed (Trachte et al., 2014; Dai et al., 2014; Pounds, Bersak, & Dollar, 2012).

PID control is conceptually simple and able to accurately perform tracking tasks even under (mild) disturbances. However, PID control must be carefully tuned and re-tuned every time the system changes to maintain/guarantee robustness and stability. PID tuning is a time intensive and difficult process requiring multiple design iterations before acceptable performance is achieved. Additionally, PID has no inherent constraint handling so control signals to the UAV system may get saturated resulting in subpar tracking performance. As highlighted in (Trachte et al., 2014) regarding UAVP systems, "significant feedback forces may be induced on the vehicle [UAV] during certain flight manoeuvres ... constant variation in the reference operating point, induced by the slung load, causes conventional controllers to demand increased control effort".

4-3-3 Linear Quadratic Gaussian (LQG) Control

Linear Quadratic Gaussian (LQG) control is an optimal control technique for linear time-(in)variant (LTV/LTI) systems that combine a Linear-Quadratic Estimator (LQE), such as a Kalman Filter (KF), as state observer and Linear-Quadratic Regulator (LQR) for control allocation. LQR can also be used in the absence of an LQE when the full state information is available as feedback.

LQR control has been successfully used for controlling the UAVP system by linearising the system dynamics about a nominal UAVP state, usually hover with the pendulum at the downward equilibrium position. The obtained LTI system is then used to design and tune the KF and LQR such that the control system is able to perform tracking tasks for the UAVP system. Studies demonstrating the use of LQG/LQR control for the UAVP system in simulation and experiments have been performed in several studies (De Crousaz et al., 2014; Omar, 2009; Bisgaard, 2008; Feng et al., 2014).

As LQG control relies on an LTI model, the controller is able to be run in real-time on systems with limited computational capabilities such as embedded micro-controllers. However, due to the reliance on linearised system dynamics, fast, agile manoeuvres far beyond the linearisation nominal state are not modelled accurately which affects the controller robustness and possible stability (Trachte et al., 2014). In comparison to PID control where gain tuning is necessary, LQG requires tuning of optimal control parameters that include the cost matrices. Cost matrices are more intuitive to tune compared to PID gains as they associate relative importance to different desired control objectives. However, just like PID control, LQG has no inherent constraint handling, therefore, the commanded systems inputs from LQG may not be attainable by the UAV leading to input saturation and consequent sub-par tracking performance.

4-3-4 Model Predictive Control (MPC)

Historically, MPC has been very popular in industrial process engineering due to the relatively slow process dynamics involved (conventional MPC solvers are computationally costly) and requirement to handle system constraints and non-linearities (Pendleton et al., 2017). However, with the advent of smaller, efficient and powerful micro-controllers, MPC is becoming more viable for controlling fast, non-linear, smaller systems such as quadrotors (Tzes,

Nikolakopoulos, & Alexis, 2012; Kunz, Huck, & Summers, 2013; Kamel et al., 2017; Trachte et al., 2015).

Model Predictive Control (MPC) also referred to as *Receding Horizon Control (RHC)* is an optimal control theory in which control is performed in discrete time (Olsder et al., 2011). Optimisation is performed with respect to a cost function (usually quadratic) that is carefully designed to promote a certain type of system behaviour. In MPC a constrained open-loop Optimal Control Problem (OCP) is solved over a set time horizon that generates a set of system inputs to achieve a locally optimal predicted system response. Only the first action from the generated optimal control sequence is performed after which the entire time horizon is shifted by one time-step (recessed) and the process is repeated (J. H. Lee, 2011; Olsder et al., 2011). Due to this 'receding horizon' principle, closed-loop control is achieved as feedback is implicitly introduced by solving the OCP every time-step. For a more thorough explanation of MPC, refer to Appendix D.

As LQR and MPC control are both optimal control schemes, comparative studies have been performed in (Gonzalez et al., 2015; Trachte et al., 2014, 2015). In (Trachte et al., 2014) the control performance of non-linear MPC to LQR is compared and improved performance for MPC was found "over a larger flight envelope, including aggressive manoeuvres and large slung load displacements". The simulation studies performed showed tests using MPC and LQR control for wind disturbance rejection, step responses and slung load damping. The study concluded that LQR is sufficient for small displacements from the equilibrium from which the LTI model is derived and provides a simpler and faster solution than MPC. However, MPC outperforms LQR for high load displacements at increased computational cost as the non-linear dynamics are considered. In (Gonzalez et al., 2015) and (Trachte et al., 2015) which build on the work of (Trachte et al., 2014), the importance of considering the non-linear system dynamics and constraints as done in MPC is highlighted demonstrating the robustness of MPC for time varying references and aggressive control. The studies demonstrate a clear improvement in tracking performance when using MPC rather than LQR in those cases.

The continuous optimisation in MPC makes it a relatively computationally expensive control method, especially when the system dynamics are complex and (highly) non-linear (Trachte et al., 2015). In contrast to PID and LQG control, the optimisation in MPC handles input and state constraints so that the generated control inputs and trajectories are also guaranteed to be physically feasible (Gonzalez et al., 2015). MPC is attractive due to its predictive nature enabling proactive control design and inherent constraint handling which is necessary for systems with limited control inputs.

4-3-5 Accounting for Cable Slackening using Switching Mode Control (SMC)

As introduced in Section 4-1-4, cable slackening results in a hybrid system model description of the UAVP dynamics. As the control techniques are model based, a hybrid system model also necessitates a Switching Mode Control (SMC) framework. In SMC two separate controllers are derived, one for each EOM set and the switching is performed by an algorithm that detects the wire slackening. An SMC formulation has been used by in (Sreenath et al., 2013) for the single UAVP case, and for the multi-UAVP case in (Sreenath & Kumar, 2013) and for performing aggressive control using LQG in (De Crousaz et al., 2014).

As introduced in Section 4-1-4, recent UAVP modelling techniques consider the continuous transition from taut to slack in one model thus eliminating the need for SMC. In (Foehn, Falanga, Kuppuswamy, Tedrake, & Scaramuzza, 2017) a novel approach to handling the hybrid system dynamics of the UAVP system is presented considering a Linear Complementarity Problem (LCP) in their optimal trajectory generation algorithm. The LCP approach is made possible by using constraints on the OCP which is possible also in MPC. Using LCP, hybrid models are avoided by considering a non-penetration/contact force that activates when two rigid bodies are in contact. The optimisation constraints are defined such that the contact force only becomes non-zero when a certain non-penetration constraint $\phi(q)$ (a function of configuration q) becomes zero. In the case of UAVP systems, the tensile force is analogous to the contact force for LCP and is non-zero only when the suspension rope is taut. For an in-depth explanation of the LCP formulation for general rigid-body dynamics refer to (Posa & Tedrake, 2013).

4-3-6 Adaptive Control Techniques for Parametric Uncertainties

Many control techniques including MPC are model-based techniques meaning their control performance is governed by the model accuracy. As presented in Section 4-1, for practical purposes simplified system dynamic models are derived. With simplified models, it is implied that the model complexity is limited to the significant physical phenomenon relevant to the swinging payload problem to make the dynamics simulation tractable. The majority of state-of-the-art literature on UAVP systems address the fundamental that include the inter-body kinematics and in some cases the rope slackening problem.

Adaptive control is any technique that account for parametric uncertainties in the model. Adaptive techniques for the following uncertainties have been addressed; a misplaced suspension point (Palunko, Cruz, & Fierro, 2012), changes in the system centre of gravity (Palunko, Cruz, & Fierro, 2012), unknown pendulum length (Bisgaard, Cour-Harbo, & Dimon Bendtsen, 2010) and unknown payload mass (Dai et al., 2014; Min, Hong, & Matson, 2011). System identification and parameter estimation techniques including the Kalman Filter (KF) have been used to identify the uncertain parameters in the model. In Section 4-5, learning based approaches to UAVP parametric uncertainties will be discussed as the state-of-the-art UAVP literature has not addressed this.

4-3-7 Vision Based State Estimation for Slung-Load

The state of the slung load is generally described relative to the UAV reference frame by two angular displacements in 3D and possibly a linear displacement when cable slackening is included. See also Appendix 5 for a derivation of the UAVP model. Most studies use external motion capturing systems to obtain the 3D Cartesian location of the payload which can be transformed into polar coordinates used in the state definition. Even though state estimation of the slung load is not the primary focus of this thesis, for interested readers vision based slung payload state estimators have been formulated.

In (Bisgaard et al., 2010) a downward facing pinhole camera on the UAV is used to estimate the load position by identifying the marked payload in the image. Using the payload

kinematics with respect to the UAV and the 2D positional information gathered from the image, the 3D payload position can be derived assuming the cable does not slack. In (Zürn et al., 2016), similar experimental studies were performed showing a mean error of only $\sim 3^{\circ}$ in the suspension angle when comparing the vision based method to external tracking for a $60cm$ pendulum . Both studies demonstrate that vision based state estimation is a promising method when compared to information gathered from an external motion capture system as long as the cable remains taut (as this is necessary to perform 3D state estimation).

4-4 Control Techniques for Collision Avoidance of UAV-Payload Systems

The high-level collision avoidance control techniques introduced in literature are of particular interest for the research to be performed in this thesis. It will be shown that for UAVP systems, the research has focussed on globally planning collision-free trajectories (open-loop) and utilising the control techniques presented for performing accurate trajectory tracking. This has been practical for the experiments researchers have performed in static environments and demonstrated impressive manoeuvres including 3D path following and flying through openings. However, as discussed before global planning only works when the environment is static or highly predictable enabling the use of deliberative planning techniques. In the case of uncertain, dynamic environments a reactive collision avoidance approach must be used for closed-loop trajectory generation. The following section will discuss the current approaches to open-loop collision-free trajectory generation of UAVP systems followed by the state-of-the-art in closed-loop collision-free trajectory generation. For completeness, some references to works made in previous sections of the literature study are repeated.

4-4-1 Open-loop Collision-free Trajectory Generation

In the open-loop case, reference trajectories are generated in C_{free} space given a known environment. UAVP system controllers have generally been designed to achieve swing-free manoeuvres with the intention of stabilising the payload under motion such that a reference trajectory is precisely followed (Trachte et al., 2015). Reference trajectories are pre-computed using methods which have included dynamic programming approach for swing and collision-free trajectories in (Palunko, Fierro, & Cruz, 2012; Palunko, Cruz, & Fierro, 2012) and reinforcement-learning approach for swing-free motion in (Faust, Palunko, Cruz, Fierro, & Tapia, 2013). These reference trajectories are then followed using tracking controller that minimise tracking error such as a PID (Palunko, Cruz, & Fierro, 2012). In (Trachte et al., 2015) the use of non-linear MPC and LQR for swing minimisation and trajectory tracking is demonstrated. Swing minimisation research has tackled the collision-free motion problem by combining accurate tracking controllers with algorithmically generated open-loop collision-free trajectories. In these studies, minimising swing is the main control objective so the overall UAVP motion is gradual as high-speed motion causes the payload to lag behind the vehicle or perform an aggressive motion.

Recent research has also looked at performing aggressive payload manoeuvres utilising, rather than suppressing, swing. As mentioned, the work (Sreenath et al., 2013) successfully

demonstrated in simulation and experiments a motion controller that accurately tracks the generated reference trajectories even under high swing loads. The motion controller and differential flatness property were also derived for the case of multi-UAV payload systems in (Sreenath et al., 2013).

The work (Foehn et al., 2017) demonstrates in simulation and experiments a fast trajectory optimisation for agile manoeuvres using UAVP systems. In their work, an optimal control formulation using LCP (as explained in Section 4-3-5) is formulated to generate an open-loop collision-free trajectory offline for multiple tasks including collision avoidance with a static obstacle. The UAVP system state encodes the UAV and payload's pose in 3D space so the generated trajectory includes a trajectory for both the UAV and payload. The generated UAV trajectory consists of a sequence of desired positions, velocities and accelerations that must be executed by the vehicle using a position controller that translates the trajectory data to UAV attitude and thrust inputs. However, this method still relies on an off-line trajectory generation that takes around 30s using 50 nodes for a single obstacle avoidance task(Foehn et al., 2017).

As highlighted in the preceding discussion and sections, the majority of UAVP research has focussed on pre-computing open-loop collision-free trajectories and building accurate motion tracking controllers. The main limitation is the absence of a feedback mechanism so any changes during runtime to the *C-space* are not considered for planning or control. For UAVP operation outside of a confined experimental setup, the *C-space* is dynamics and always changing which would render the pre-generated open-loop trajectory invalid causing a potential UAV collision. This highlights the necessity for closed-loop collision-free control of the UAVP system.

4-4-2 Towards Closed-loop Collision-free Control

Research involving only UAVs have already tackled the closed-loop collision-free control problem using various approaches, however, most have not been applied to a UAVP system. Collision avoidance in the presence of uncertain environments has been performed using a group of *reactive collision avoidance* techniques enabling trajectories to be shaped/modified on a local scale to perform quick avoidance manoeuvres (Čáp et al., 2016; Alonso-Mora, Naegeli, Siegwart, & Beardsley, 2015). In (Fiorini & Shiller, 1998) the concept of Velocity Obstacles (VO) space for performing reactive collision avoidance was introduced. In (Alonso-Mora et al., 2015), the VO idea has been extended to demonstrate, in simulation, collision-free motion for multiple agents sharing a workspace by using a potential function that discourages vehicles getting too close to each other. Recently in (Naegeli et al., 2017), a similar approach is demonstrated using potential functions to perform real-time collision avoidance for static and dynamic obstacles on a single quadrotor using MPC control. The UAV is driven away from collision by including an MPC cost term that increases based on the UAV's proximity to obstacles. Also in (Naegeli et al., 2017), position and velocity of moving obstacles are used to forward simulate obstacle motion allowing changes in C_{free} to be accounted for in the trajectory generation. Even though only an unaltered UAV system was considered, the research done shows promising applications for a UAVP system as proposed in this thesis.

Similar to this thesis proposal, the work (De Crousaz et al., 2014) has tackled the closed-loop UAVP trajectory generation and control optimisation using an iterative Linear Quadratic Gaussian (iLQG) optimal controller that adapts the system dynamics model for parametric uncertainty using Kalman Filtering . Adaptation is performed using a sampling based learning algorithm which works in parallel with the iLQG controller. The iterative aspect of iLQG allows it to "return locally optimal linear feedback controller able to work with arbitrary non-linear cost functions" (De Crousaz et al., 2014) as to optimise a sequence of control inputs accounting for changing system dynamics thus enabling more aggressive control beyond the nominal linearisation point. An SMC formulation must be used as they use a hybrid system model description for the UAVP system. Successful demonstration of trajectory generation and flight through a narrow opening with the UAV and payload was achieved by designing the iLQG cost function. As mentioned previously, this was also demoed in (Mellinger, Michael, & Kumar, 2012), however, in that case, the trajectory was designed beforehand rather than performing the generation online. In (De Crousaz et al., 2014) a high-cost term was associated with UAV configurations that would be in C_{obs} thus guiding the algorithm to collision-free trajectories. However, the study did not consider dynamic obstacles, which could have been an extension, and it must be noted that iLQG does not explicitly consider system input constraints so there were cases in their experimental study where the required UAV input was saturated so the response was not as expected. Input saturation is generally not a problem with MPC as it readily handles system constraints making it more robust, however, this makes MPC more computationally expensive. Also, it is important to remember that constraining the MCP too aggressively may result in frequent infeasible solutions. Compared to iLQG, the benefit of MPC is that it is predictive which enables collision-avoidance planning to be proactive; the predicted trajectory can be collision-checked over the set time-horizon so that evasive manoeuvres are performed as a contingent before the obstacle gets too close.

Summarising, similar to (Foehn et al., 2017) for open-loop and (De Crousaz et al., 2014; Naegeli et al., 2017) for closed-loop collision avoidance, the purpose of this thesis is to demonstrate MPC based control for active real-time collision-free control of a UAVP system in environments with uncertain/unknown static and dynamic obstacles. Contrary to the study (Foehn et al., 2017), the collision avoidance trajectory generation will also be done in closed-loop (real-time). Contrary to both (Foehn et al., 2017) and (De Crousaz et al., 2014), however as addressed by (Naegeli et al., 2017), dynamic obstacles for the UAVP system will be considered using an MPC control formulation using the MPC s predictive nature and constraint handling.

4-5 Learning Based System Identification

As mentioned in Section 4-3-6, researchers have looked at adaptive control techniques that involve a parameter estimation scheme to addresses parametric uncertainties in the system dynamical model. This section discusses learning techniques for handling parametric uncertainties using a Learning Based System Identification (LBSI) scheme.

4-5-1 Types of Learning

Three high-level approaches to learning are identified in literature with the possibility of sub-classifications for specific algorithms(Du & Swamy, 2013);

- *Supervised learning* - Given a set of inputs (independent variables) and associated outputs (dependent variables), the algorithm captures the functional mapping between the inputs and outputs
- *Unsupervised learning* - Given only the set of inputs, the algorithm attempts to auto-associate the information to find significant patterns or features
- *Reinforcement learning* - Generates a policy to perform certain actions sequentially to maximise a total expected reward. It is a form of supervised learning, due to the reward process, where the desired output is unknown

Supervised Learning is commonly used when a certain input-output data relation exists, however, the functional mapping must be identified. During learning, the algorithm's primary goal is to minimise the residual error between the predicted (learnt) and actual output given a certain input taken from the entire input set (Du & Swamy, 2013). Examples of supervised learning algorithms include regression analysis and neural network non-linear fitting/regression. Supervised learning is useful for identifying system dynamics models provided input-output data is obtainable from empirical studies.

Unsupervised Learning is used when a data set does not have an inherent relational construct (with an output) is available. The goal is to identify patterns and features in the data set that can result in data dimensionality reduction or reduce the total data to be processed (Du & Swamy, 2013). Examples of unsupervised learning algorithms include clustering using k-means and neural network based classification.

Reinforcement Learning (RL) is neither a supervised or unsupervised learning method, rather it uses a 'learning by interaction' approach. RL involves having a 'learner' that performs actions and observes rewards as to determine a sequence of actions to maximise the cumulative expected reward. As explained in (Sutton & Barto, 1998), different from supervised learning there is no example data set (input-output data) to be reproduced so the learner must explore uncharted territory to gain knowledge. Also, it is not unsupervised learning as there is a reward system associated with being in every state and performing a certain action which is guiding the learner's decision. RL generally involves a trial-and-error approach to identifying a policy (sequence of actions) resulting in the highest cumulative reward (Sutton & Barto, 1998). The trial-and-error nature means learning RL learning is generally performed online for slowly evolving systems or completely offline.

4-5-2 Learning for Parametric Uncertainties

The modelling of system dynamics involves knowledge of the model structures and parameters. System modelling categories are identified based on the information available and occurs on a

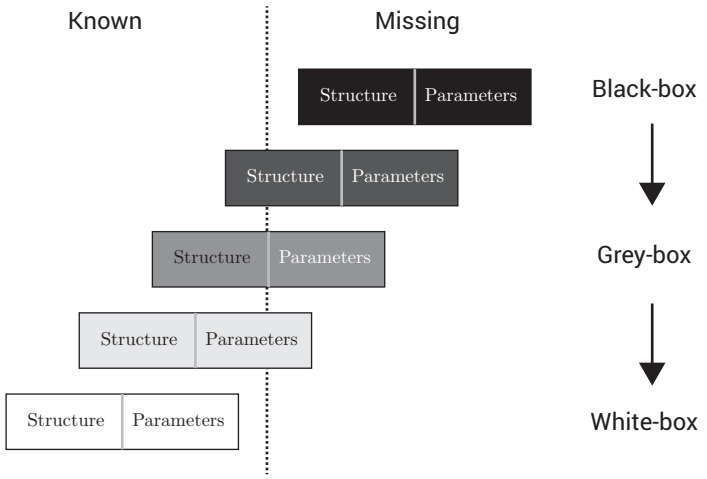


Figure 4-4: Model classification based on prior information available (Horváth, 2003)

spectrum and ranging from *black-box* to *white-box* modelling as shown in Figure 4-4 (Horváth, 2003). A model generally lies between the two extremes and is known as a *grey-box* model.

Non-parametric models (black-box models) are derived when no definite model structure can be obtained, however, considering the UAVP system, it was shown in Section 4-1 that the fundamental structure based on first-principles is derivable. It was shown that the model has a parametric dependency on the UAV and payload mass, and wire length. The full model is derived in Appendix 5. The UAVP modelling is generally white-box as it is assumed that all parameters are known or measurable. However, to include some modelling adaptability a grey-box approach is more appropriate. To address the uncertain parameters, a common parametric identifier that relies on the system structure is used which includes the (extended) Kalman Filter (Qin, Su, & McAvoy, 1992).

If there are system dynamics for which the model structure cannot be derived or is too complex, a black-box approach is generally appropriate. As shown in Section 4-1, the UAVP model is applicable on the basis of assumptions being valid which is necessary to limit the model's complexity. However, these assumptions only remain valid under certain UAVP system setups, for example, when the payload becomes a large flat plate, the point load and negligible drag assumptions become invalid. Also, if the payload contains a fluid substance, the dynamics modelling becomes complicated and mathematically complex. In these cases, a data rather than a physics driven model is more appropriate for which black-box modelling is required (Horváth, 2003; Qin et al., 1992). With data driven modelling, the system's input-output data is collected empirically and the functional mapping must be captured by the model. Among many black-box system identification approaches, Neural Networks (NN) have seen considerable application from the research community due to their "simple architecture [and] their universal approximation capability" (Horváth, 2003; Qin et al., 1992). Neural networks are "distributed information processing systems made up of a greater number of highly interconnected identical or similar simple processing units" (Horváth, 2003). Each 'processing unit', technically referred to as a *neuron*, produces an output which is a linear weighted combination of the neuron inputs processed through an activation function. Various activations functions exist which include (piece-wise) linear and logistic functions. Combining many of these neurons (with same/different activation functions) in parallel and series

and associating each inter-neuron connection with weights results in the NN. The process of modifying the NN weights enables the network to be configured for representing a functional mapping. For an in-depth explanation of the NN structure, refer to the work (Horváth, 2003).

4-5-3 Learning Implementation in Control Architectures

Neural networks, specifically for fitting, are data driven models that rely on supervised learning to recreate the input-output mapping observed from a dynamical system that takes the form of a non-linear function given by Equation 4-3 where $\mathbf{y}$ is the output, $\mathbf{x}$ is the state and $\mathbf{u}$ the input. Note that both the state and input vectors are inputs to the NN that models $\mathbf{f}$.

$$\mathbf{y} = f(\mathbf{x}, \mathbf{u}) \tag{4-3}$$

NNs demonstrate adaptability through learning which is a process by which the network weights are modified (Horváth, 2003). A global output error function is defined, usually in the form of a least squares error over the entire data set, that quantifies the model's accuracy with respect to the observed system dynamics (the training set). The error is computed using the difference between the network and actual output for a given input. The global error is then *back-propagated* to compute the error contribution of every neuron and recompute the inter-neuron weights iteratively (see (Qin et al., 1992) for details). After many iterations, the global error value will converge and the network will accurately, to a certain extent, reproduce the training set provided the network size is sufficient. In (Cybenko, 1989) the universal approximation theory is formulated stating that given a finite number of neurons in a feed-forward NN, the network can approximate any continuous function under certain mild conditions imposed on the activation functions. This is an important theorem as it proves that provided the network size is sufficient, the continuous modelling function, as given by Equation 4-3, can be accurately reproduced by the NN. This allows NNs to be used as a replacement for the system dynamics model required by model-based controllers. However, a commonly cited drawback of NN is its black-box nature, thus not exposing the structure and parameters, which prevents further analysis and hard guarantees of its stability and accuracy (Åkesson & Toivonen, 2006). Also, over-fitting must be checked which occurs when the network perfectly reproduces the training set, but given new 'unseen' input data, the output is poorly predicted. To ensure the model generalises to the full range of input data, over-fitting and validation checks are commonly performed by checking the network performance with new empirically collected data (Åkesson & Toivonen, 2006).

The NN identified model can be used in combination with a model-based control technique, as introduced in Section 4-3, to perform high-level UAVP control. This approach enables UAVP control when there are parametric uncertainties in the model and/or modelling the UAVP dynamics becomes too complicated due to the introduction of complex physical phenomenon. An NN model is able to adapt online using the back-propagation learning method whereby new data collected during runtime can be to adjust the network weights incrementally achieving adaptability to changing system dynamics. Figure 4-5 shows an expanded form of the UAVP control schematic, originally introduced Figure 4-3, with an NN identified UAVP system model. The schematic shows how the UAVP system output is used to compute an error and back-propagate that for learning the NN weights. The NN identified model is then used to update the controller's internal model to achieve adaptive control. The system state, as

obtained from sensors, and input generated by the controller is used by the NN model to compute a predicted model output y_{NN}.

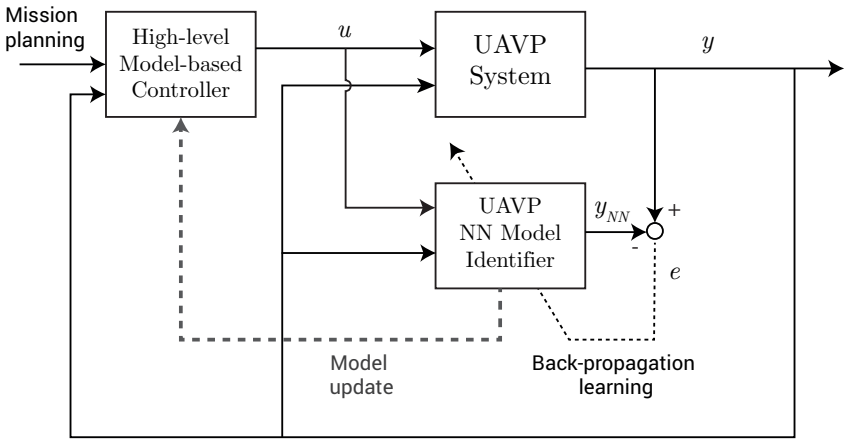


Figure 4-5: Schematic control architecture for UAV-Payload system with Neural Network identified model used for enabling adaptation

Chapter 5

Derivation of UAV-Payload System Kinematics and Dynamics

This chapter describes the derivation of the UAV-Payload (UAVP) model in detail. The derived model is not specific to the hardware used during the experiment so it generalises to other setups provided the assumptions remain valid. This model is derived from literature presented in published articles as introduced in Chapter 4.

Section 5-1 introduces the UAVP system and the reference frames used in the derivation. Section 5-2 introduces the relevant kinematic relations while Section 5-3 treats the derivations of the system dynamic equations. Section 5-4 then looks at extending the UAVP model with the inclusion of aerodynamic drag on the UAV and payload. Note that a general parametric model is derived; the actual parameter values are provided in Section 6-1 that discusses the experimental hardware used. Please note that the following subscripts are used interchangeably throughout the report; for quadrotor Q and q and for payload L and l.

5-1 System Model and Reference Frames

The UAVP system studied comprises a Vertical Take-off and Landing (VTOL) quadrotor UAV with a suspended point mass payload attached using a rigid massless link (the cable). The payload, modelled as a point mass, is attached to the cable's end with the other end being the suspension point attached at the quadrotor's center of mass. As presented later in this preliminary study, slackening behaviour of the cable was studied over the full range of manoeuvres and it was determined that the cable remains taut throughout the flight. Should slackening of the cable be observed in future experimental studies, the behaviour can be implemented in the model at a later stage. To limit the complexity of the derived UAVP model, the following assumptions are adopted;

- Payload is a point mass

- Quadrotor center of gravity and centroid coincide
- Suspension cable is a rigid massless link attached on a free suspension/pivot point coincident with the quadrotor's center of mass (origin of Vehicle carried normal Earth reference frame $\{E\}$)
- No aerodynamic drag effects on the suspension cable / rigid link due to massless assumption
- Standard sea level conditions for atmospheric and gravitational effects

The Equations Of Motions (EOMs) are derived in the three-dimensional inertial frame $\{I\}$, as shown in Figure 5-1, using Lagrangian mechanics theory. The generalised coordinates $\mathbf{q}$, is defined as

$$\mathbf{q} = [x_Q \;\; y_Q \;\; z_Q \;\; \theta_L \;\; \phi_L]^\top \tag{5-1}$$

The time derivative is given by,

$$\dot{\mathbf{q}} = \left[\dot{x}_Q \;\; \dot{y}_Q \;\; \dot{z}_Q \;\; \dot{\theta}_L \;\; \dot{\phi}_L\right]^\top \tag{5-2}$$

Then the system state is denoted by,

$$\mathbf{x_q} = [\mathbf{q}, \;\; \dot{\mathbf{q}}] \tag{5-3}$$

The load position $\mathbf{p_L} = [x_L \;\; y_L \;\; z_L]^\top$ is fully described by the Quadrotor position $\mathbf{p_Q} = [x_Q \;\; y_Q \;\; z_Q]^\top$ in combination with the suspension angles θ_L (x-axis $\{S\}$ frame) and ϕ_L (y-axis $\{S\}$ frame) defined in the Quadrotor attached $\{S\}$ frame as shown in Figure 5-1. The specific choice of coordinate frames, specifically the $\{S\}$ frame, ensures there is no singularity in the derived EOMs for the pendulum's equilibrium position (Jain, 2015). The derivation of the EOMs also closely follows that of Jain in (Jain, 2015; Klausen et al., 2015) with some modifications and extensions to include aerodynamic drag.

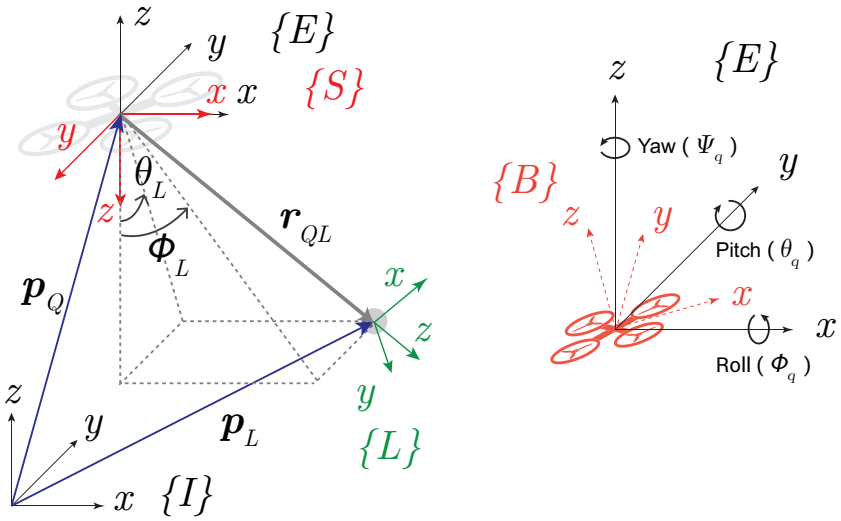


Figure 5-1: Schematic of UAV-Payload system and quadrotor showing reference frames and payload suspension angles

This system is modelled as a spherical pendulum under aerodynamic drag disturbance attached to a mobile pivot point on a three-dimensionally translatable vehicle. Inputs to the system are three forces $\mathbf{f_u} = [F_x \ F_y \ F_z]^\top$ defined in $\{E\}$ and acting directly on the quadrotor center. Reference frames are defined and presented in Figure 5-1 are itemised with an explanation;

- $\{I\}$ - Inertial East, North, Up (ENU) reference frame taken on Earth surface
- $\{E\}$ - Vehicle carried ENU reference frame; origin coinciding with the Quadrotor centre of gravity and x orientated with vehicle's front facing camera axis
- $\{S\}$ - The $\{E\}$ frame rotated by 180° degrees about the $\{E\}$ x-axis
- $\{B\}$ - The vehicle body frame which is the $\{E\}$ frame rotated by the yaw (ψ), pitch (θ) and roll (ϕ) Euler angles in that specific order.
- $\{L\}$ - Payload reference frame; origin coinciding with payload's position, z-axis aligned away from link's suspension point. In equilibrium the x and y axes are parallel to the x and y axes of frame $\{S\}$.

The Euler angles expressing the UAV's orientation (frame $\{B\}$) in frame $\{E\}$ is given in Figure 5-1 which shows the definition of yaw, pitch, roll. The ZYX rotation convention, i.e. first yaw, then pitch, then roll, is followed to describe the orientation of $\{B\}$ with respect to $\{E\}$. The transformation is mathematically expressed in terms of rotation matrix given by Eq. 5-4 where the rotation axis is sub-scripted.

$$\mathbf{R_E^B} = \mathbf{R_x}(\phi)\mathbf{R_y}(\theta)\mathbf{R_z}(\psi) \tag{5-4}$$

5-2 Kinematic Relations

The load position in absolute coordinates ($\{I\}$ frame) and as a function of the generalised coordinates q is given by Eq. 5-5. Vector $\mathbf{p_Q}$ is the UAV's position and vector $[0 \ 0 \ l]^\top$ is the cable link vector of length l defined in frame $\{L\}$.

$$\mathbf{p_L} = \mathbf{p_Q} + \mathbf{R_S^E}\mathbf{R_{S'}^S}(\phi_L)\mathbf{R_L^{S'}}(\theta_L)\begin{bmatrix} 0 \\ 0 \\ l \end{bmatrix} \tag{5-5}$$

The rotation matrices as used in Eq. 5-5 are given below. Note that the body attached frame $\{E\}$ has the same orientation as frame $\{I\}$, therefore no rotation transformation is necessary. The intermediary $\{S'\}$ frame is necessary to perform the payload suspension angle based transformation.

$$\mathbf{R_L^{S'}}(\theta_L) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_L & -\sin\theta_L \\ 0 & \sin\theta_L & \cos\theta_L \end{bmatrix} \tag{5-6}$$

$$\mathbf{R}^{\mathbf{S}}_{\mathbf{S}'}(\phi_L) = \begin{bmatrix} \cos\phi_L & 0 & \sin\phi_L \\ 0 & 1 & 0 \\ -\sin\phi_L & 0 & \cos\phi_L \end{bmatrix} \tag{5-7}$$

$$\mathbf{R}^{\mathbf{E}}_{\mathbf{S}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{5-8}$$

The load velocity, given by Eq. 5-9, is derived by taking the time derivative of Eq. 5-5 for $\mathbf{x_L}$.

$$\dot{\mathbf{p}}_\mathbf{L} = \dot{\mathbf{p}}_\mathbf{Q} + \mathbf{R}^{\mathbf{E}}_{\mathbf{S}}(\pi) \cdot \frac{d}{dt}\mathbf{R}^{\mathbf{S}}_{\mathbf{S}'}(\phi_L) \cdot \mathbf{R}^{\mathbf{S}'}_{\mathbf{L}}(\theta_L) \begin{bmatrix} 0 \\ 0 \\ l \end{bmatrix} + \mathbf{R}^{\mathbf{E}}_{\mathbf{S}}(\pi) \cdot \mathbf{R}^{\mathbf{S}}_{\mathbf{S}'}(\phi_L) \cdot \frac{d}{dt}\mathbf{R}^{\mathbf{S}'}_{\mathbf{L}}(\theta_L) \begin{bmatrix} 0 \\ 0 \\ l \end{bmatrix} \tag{5-9}$$

5-3 System Dynamics

The payload suspension cable is assumed to be a massless rigid link. Under this assumption, the attachment is a three-dimensional free pivot so there is no torque transfer from the payload to the quadrotor and vice versa. To derive the EOMs using Lagrangian mechanics, the total system kinetic and potential energy is derived from which the Lagrangian L can be computed.

The kinetic energy K of the system is given by Eq. 5-10 where $\text{diag}(m, 3)$ denotes a diagonal matrix of dimension 3×3 with entries m. Variables m_Q and m_L are the UAV and payload mass, respectively.

$$K = \frac{1}{2} \cdot [\dot{\mathbf{p}}_\mathbf{Q} \quad \dot{\mathbf{p}}_\mathbf{L}] \cdot \begin{bmatrix} \text{diag}(m_Q, 3) & 0 \\ 0 & \text{diag}(m_L, 3) \end{bmatrix} \cdot [\dot{\mathbf{p}}_\mathbf{Q} \quad \dot{\mathbf{p}}_\mathbf{L}]^\top \tag{5-10}$$

The potential energy P of the system, given by Eq. 5-11, includes only the UAV and payload's gravitational potential energy.

$$P = m_Q g z_Q + m_L g z_L \tag{5-11}$$

The Lagrangian L is given by Eq. 5-12 from which the EOMs are derived by the Euler-Lagrange equations in terms of the generalised coordinates.

$$L = K - P \tag{5-12}$$

Equation 5-13 is constructed for every generalised coordinate q_i of vector $\mathbf{q}$. f_i represents a generalised conservative external force or torque defined on the generalised coordinate directions q_i.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i \tag{5-13}$$

Let the general external input force vector be denoted by $\mathbf{f}$ given by Equation 5-14. No direct external force/torque affects the θ_L or ϕ_L angles, therefore, those entries in $\mathbf{f}$ are zero.

$$\mathbf{f} = [\ \mathbf{f_u} \quad 0 \quad 0\ ]^\top = [\ F_x \quad F_y \quad F_z \quad 0 \quad 0\ ]^\top \tag{5-14}$$

The resulting system of equations gives the EOMs and are in the form given by Eq. 5-15 which is rearranged to solve for $\ddot{\mathbf{q}}$ in Eq. 5-16. $\mathbf{M}$ represents the Mass matrix, $\mathbf{C}$ the Coriolis matrix and $\mathbf{G}$ the gravitational matrix.

$$\mathbf{f} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) \tag{5-15}$$

$$\ddot{\mathbf{q}} = \mathbf{M^{-1}}(\mathbf{q})(\mathbf{f} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{G}(\mathbf{q})) \tag{5-16}$$

The resulting EOMs are lengthy, therefore, they have not been included in this report in the expanded form. See (Klausen et al., 2015) for the full form. The EOMs have been derived using the symbolic manipulation toolbox in MATLAB following the derivation steps outlined in this report. The code and resulting computed EOMs are available digitally in MATLAB code form.

5-4 Drag Inclusion in Dynamics

Drag effects on the UAV and payload are included in the EOMs and this section outlines the theory and implementation procedure. The drag force on the UAV with velocities $\dot{\mathbf{p}}_\mathbf{Q}$ is given by the linear drag form Eq. 5-17. The drag force $\mathbf{f}_{D,Q}$ is given by the multiplication of drag constant $k_{D,Q}$, and $\dot{\mathbf{p}}_\mathbf{Q}$, the UAV's velocity. Assuming the UAV moves relatively slow, the linear drag formulation should sufficiently capture the drag dynamics on the vehicle (Klausen et al., 2015).

$$\mathbf{f}_{D,Q} = \tfrac{1}{2} k_{D,Q} \dot{\mathbf{p}}_\mathbf{Q} \tag{5-17}$$

5-4-1 Payload Drag in Quadratic Form

Payload drag for the preliminary study is computed using quadratic drag theory as the payload moves at higher velocities due to the combined translational and rotational velocity. The drag force always acts in the $-\dot{\mathbf{p}}_\mathbf{L}$ direction and will be estimated using the drag Equation 5-18 where $C_{D,L}$ is the drag coefficient, S the payload drag affected reference area and ρ the sea-level air density of $1.225 kg/m^2$. Simplifying the notation, all constant terms in Equation 5-18 are denoted by drag constant $k_{D,L}$.

$$\mathbf{f}_{D,L} = \frac{1}{2} C_D \rho S (\dot{\mathbf{p}}_\mathbf{L}^\top \dot{\mathbf{p}}_\mathbf{L}) \cdot \frac{\dot{\mathbf{p}}_\mathbf{L}}{|\dot{\mathbf{p}}_\mathbf{L}|} = -k_{D,L} (\dot{\mathbf{p}}_\mathbf{L}^\top \dot{\mathbf{p}}_\mathbf{L}) \cdot \frac{\dot{\mathbf{p}}_\mathbf{L}}{|\dot{\mathbf{p}}_\mathbf{L}|} \tag{5-18}$$

As the payload drag always acts perpendicular to $\mathbf{r}$ (the link vector), no component of $\mathbf{f}_{D,L}$ acts radially, therefore, the force is purely tangential. The payload drag is included in the

EOM as an external torque τ such that it acts on the system's states θ_L and ϕ_L; the computed torque value is given by Eq. 5-19.

$$\boldsymbol{\tau} = \mathbf{r_{QL}} \times \mathbf{f}_{D,L} \tag{5-19}$$

Vector $\mathbf{r}$ is the moment arm from the suspension point $\mathbf{p_Q}$ to the payload $\mathbf{p_L}$ and is given by Eq. 5-20.

$$\mathbf{r_{QL}} = \mathbf{p_L} - \mathbf{p_Q} \tag{5-20}$$

As the angles θ_L and ϕ_L are defined in the $\{S\}$ frame it is necessary to transform the computed torques from the $\{E\}$ frame to the $\{S\}$ frame using the transformation $\mathbf{R_S^E}$ given by equation 5-8.

$$\boldsymbol{\tau}^{\boldsymbol{S}} = \mathbf{R_S^E}^{-1}\boldsymbol{\tau} \tag{5-21}$$

5-4-2 Payload Drag Induced Moment

An alternative method is to describe the payload drag induced moment on the payload suspension angles. In the previous section, the quadratic drag was translated into a torque, however, using the linear to angular velocity relation, the torque, given by Equation 5-19 can be directly computed by Equation 5-22.

$$[\tau_x, \tau_y]^\top = k_{D,L} l^3 \left[\dot{\theta}_L^2 \frac{\dot{\theta}_L}{|\dot{\theta}_L|}, \dot{\phi}_L^2 \frac{\dot{\phi}_L}{|\dot{\phi}_L|}\right]^\top \tag{5-22}$$

This is a simplification as only the rotational dynamics of the payload relative to the quadrotor are considered and not the translational. Therefore, this approach is only valid for agile motions where the payload rotational dynamics are more prominent. If the entire system translates for sustained periods of time in one direction, the payload would lift up behind the quadrotor due to drag, this cannot be modelled using this approach. The benefit of this approach is that $\dot{\theta}_L$ and $\dot{\phi}_L$ are part of $\dot{\mathbf{q}}$ making the drag implementation in the EOMs computationally simple. For the quadratic drag implementation, the term $\dot{\mathbf{p}}_\mathbf{L}$ must be computed by Equation 5-9 from $\mathbf{q}, \dot{\mathbf{q}}$ making the EOMs more computationally expensive.

5-4-3 Drag Implementation in Equations of Motion

The external aerodynamic drag induced torque acting on the payload is divided into its components $\boldsymbol{\tau}^{\boldsymbol{S}} = [\tau_x \ \tau_y \ \tau_z]^\top$ which are then included in the EOMs. The torque τ_x acts on θ_L and τ_y on ϕ_L. Note that τ_z has been ignored as it is generally a very small torque component. In general τ_z is an order of magnitude smaller than τ_x or τ_y, therefore, it is assumed to have minimal effect on the motion. Therefore, τ_z is insignificant for the derived UAVP model. An appropriate way to include this in the dynamics may be considered in a future study.

The total system drag is abbreviated by the drag term $\mathbf{D}(\dot{\mathbf{q}})$ as defined below. The complete EOMs with drag is given by Equation 5-24.

$$\mathbf{D}(\dot{\mathbf{q}}) = [\ \mathbf{f}_{D,Q} \quad \tau_x \quad \tau_y\ ]^\top \tag{5-23}$$

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q})(\mathbf{f} - \mathbf{D}(\dot{\mathbf{q}}) - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{G}(\mathbf{q})) \tag{5-24}$$

Having defined the EOMs, the subsequent step involved verifying and validating the model through simulation and experimental studies. The necessary method and steps to achieve this will be outlined in Chapter 6.

Chapter 6

Simulation and Experimental Setup and Methodology

To verify and validate the UAVP model, both simulation and experimental data was collected to ensure the model is representative of observed UAVP behaviour. Section 6-1 describes the experimental hardware used throughout the study. Section 6-2 describes the simulation environment coded for the study while Section 6-3 explains the indoor workspace environment. Section 6-4 describes the experimental procedure in detail with the necessary theoretical content. Finally, Section 6-5 outlines how experimental data was collected and post-processed for performing the study.

6-1 Experimental Hardware and Control System

The hardware described in this section will be used throughout the study for both simulation and experiments. The methodology and setup described generalises to other hardware, with possibly some minor modifications, provided the interfaces between the hardware components is comparable.

6-1-1 Quadrotor with Suspended Payload

A Parrot Bebop 2 quad-rotor[1] with a suspended spherical payload was used with the physical characteristics outlined in Table 6-1. The only modifications made to the Bebop 2 hardware was the attachment of four light-weight rotor guards and four retro-reflective markers. Figure 6-1 shows the quadrotor and payload with the attached markers necessary for tracking the their pose using external motion capturing.

The quadrotor's drag constant was taken to be that of a blunt bullet head cylindrical object (Bebop's body is of similar shape) (Scott, Jeff, 2005) and increased to account for the rotors

[1]Parrot. Bebop 2. Accessed September 26 2017. `https://www.parrot.com/us/drones/parrot-bebop-2`

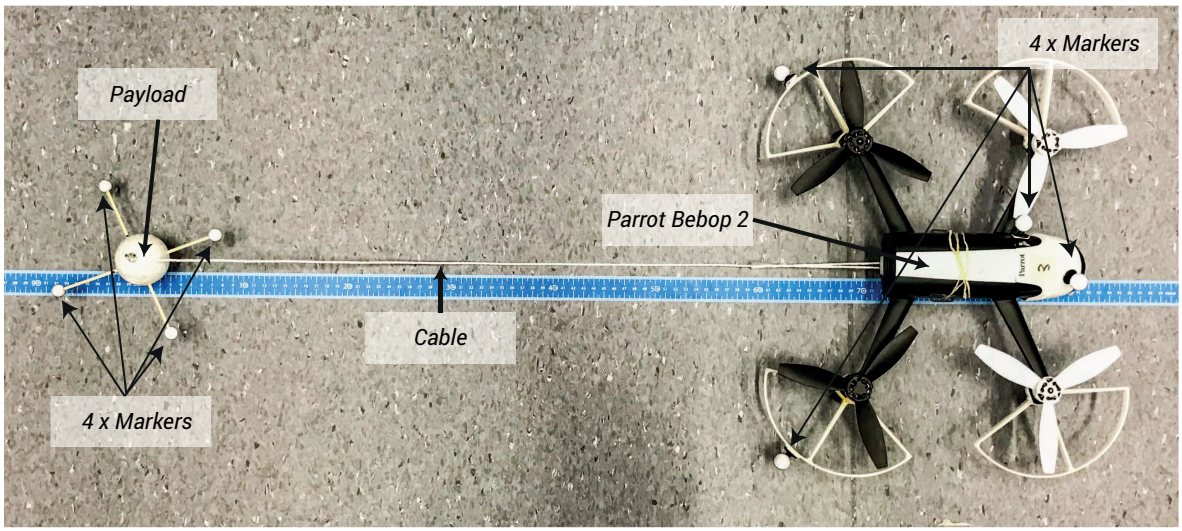


Figure 6-1: Photo of Parrot Bebop 2 and suspended payload with motion capturing retro-reflective marker locations indicated

and guards. The payload's drag constant is computed assuming a spherical object with a contact surface of 0.025 m^2 (cross-section dimension). The quadrotor's mass is taken from the published technical specifications and the payload mass was measured on a digital scale with a precision of 0.1 grams.

Table 6-1: Physical characteristics of Parrot Bebop 2 quadrotor and attached payload

System	Characteristic	Value
Parrot Bebop 2 Quadrotor	Mass	500 g
	Drag constant $k_{D,Q}$	0.28
Payload	Mass	11.1 g
	Massless Cable Length	0.77 m
	Drag constant $k_{D,L}$	1.77 $\times 10^{-3}$

The Parrot Bebop 2 runs on a proprietary Operating System (OS) and interfaces with other hardware over a Wi-Fi based network. The OS handles all low-level control and regulation tasks while accepting high-level commands sent to it processed through the Parrot Bebop's Software Development Kit (SDK). The SDK is the only available interface with the UAV, therefore, the user can only internally reconfigure the UAV to the extent the SDK allows it. The piloting functions included in the SDK allow a user to execute pitch θ_u, roll ϕ_u and vertical velocity $\dot{z}_u$ commands,

$$\mathbf{u} = [\theta_u, \phi_u, \dot{z}_u]^\top$$

Note: In the paper, the entries of $\mathbf{u}$ are denoted by equivalent notation $[\bar{\theta}_q, \bar{\phi}_q, \bar{w}_q]$.

Yaw commands may also be executed, however, the quadrotor is able to move in $\mathbb{R}^3$ without yaw so it help constant and not considered for control. For a comprehensive list of all configurable variables, functions and commands, refer to the Parrot SDK [2].

[2]Parrot. "ARDroneSDK3". Accessed September 26, 2017. `http://developer.parrot.com/docs/bebop/`

6-1-2 Full Control System Design

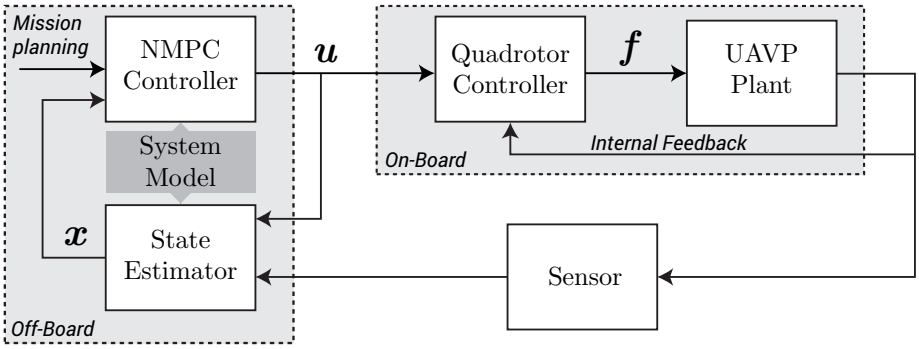


Figure 6-2: High-level schematic control architecture system showing division into subsystems, namely high-level (NMPC) controller, Quadrotor Controller, the UAVP plant and State Estimator

The full control system design is divided into four subsystems namely,

1. *High-Level Controller* - Non-linear Model Predictive Controller (NMPC) that generates commands $\mathbf{u}$ for the quadrotor that are sent via the SDK to the Parrot Bebop 2
2. *Quadrotor (Inputs) Controller* - accepts $\boldsymbol{u}$ commands that can then be transformed to control input force commands $\boldsymbol{f}$ acting on the UAVP system
3. *UAV-Payload (UAVP) System* - accepts control input force commands $\boldsymbol{f}$ that then are used to model the UAVP dynamics resulting in the UAVP state; $\boldsymbol{q}$ and its derivative.
4. *State Estimator* - using $\boldsymbol{u}$ and the state feedback from the system through the Motion Capturing System, a filtered/estimated state estimate $\boldsymbol{x}$ is computed

See Figure 6-2 schematising the control system.

The *High-Level Controller* generates the piloting command $\boldsymbol{u}$. The generation is based on optimising the predicted system response according to the mission planning and system model; this will be part of the subsequent study. The *Quadrotor Controller* generates motor inputs derived from commanded high-level inputs $\boldsymbol{u}$ such that a control force input $\boldsymbol{f}$ is generated acting on the UAVP system. The UAVP system's dynamic response to $\boldsymbol{f}$ is given by the system's state which is subsequently used as feedback to the high-level controller closing the loop. The off-board state estimator design was part of the subsequent study and is discussed in Chapter 11. Note that the internal low-level quadrotor controller generally runs at a very high frequency (order 10^1 to 10^2 Hz) allowing the high-level controller to run at lower frequencies as it only generates reference input commands without directly controlling the UAV rotors.

The quadrotor controller as implemented on the Parrot Bebop 2 has not been documented by the manufacturer, however, to be able to predict the UAVP system response it was necessary to derive a model for these internal dynamics. In the subsequent section the Parrot Bebop 2 specific controller model is briefly introduced. In the remainder of the preliminary study, the identification, verification and validation of the quadrotor and UAVP model from empirical data is discussed. Using the derived models, the subsequent study will discuss the high-level NMPC controller design.

6-1-3 Quadrotor (Inputs) Controller

In the most primitive form, a quadrotor may be modelled as four individually controlled thrust and torque inputs that transform to the quadrotor dynamics evolving in $SE(3)$. Readers interested in modelling with motor inputs can refer to (Mahony et al., 2012) for the quadrotor model. As mentioned, the quadrotor does not require any yaw commands to perform full $\mathbb{R}^3$ translational motion, hence it is set to a constant and assumed to be zero degrees.

Parrot Bebop 2 Internal Control Design

Considering only the quadrotor (no payload), this section describes a model for the internal controller adopted by the Parrot Bebop 2. This is the best guess of what the actual dynamics are considering there is no documentation available. Similar to (Chen & Wang, 2013), the model considers the attitude (pitch/roll) dynamics to be decoupled from the translational dynamics as the executed pitch/roll angles are relatively small and over short time periods. All inputs are trimmed to the equilibrium hover condition which is a zero pitch, roll and vertical velocity. Similar to (Klausen et al., 2015), assuming fast attitude dynamics, the UAV is controllable by roll, pitch and a vertical velocity commands in $\boldsymbol{u}$ such that a resulting control input force,

$$\boldsymbol{f}_u = [F_x, F_y, F_z]^\top \tag{6-1}$$

defined in the inertial world axes is generated for performing translational motion.

In Figure 6-3 the quadrotor's internal control design is shown. With the input pitch and roll commands θ_u and ϕ_u, motor commands c are generated that result in a moment M and forces F_x and F_y on the quadrotor. As observed from the system response in trials, the quadrotor maintains altitude when executing a pitch and/or roll command hence there is no contribution to F_z. The quadrotor's true pitch θ_q and roll ϕ_q response to moment M is internally used as feedback to compute the error signal sent to the attitude controller. With the input vertical velocity command $\dot{z}_u$ motor commands c are generated that results in a force F_z defined on the inertial z axis. As observed from the system response in trials, the system only ascends and descends, even under a pitch/roll angle, when $\dot{z}_u$ is non-zero. Therefore, this input only affects the quadrotor translational dynamics in the inertial z axis. The quadrotor's true vertical velocity response $\dot{z}_q$ to the generated F_z input is internally used as feedback to compute the error signal sent to the vertical velocity controller. The combined forces F_x, F_y and F_z are the control input forces by which the quadrotor translational dynamics evolve giving the system velocities $\dot{\boldsymbol{p}}_q$ and accelerations $\ddot{\boldsymbol{p}}_q$.

All internal feedback is derived from the Parrot Bebop's built-in sensors (Full list of sensors available in (PaparazziUAV, 2017)). As Parrot does not officially release the Bebop 2's internal controller design, this is the best guess of what the actual system looks like. To be able to perform predictions of the system response, it was necessary to identify the relation $\boldsymbol{u} \rightarrow \boldsymbol{f}_u$ which is discussed in the next section.

Derivation of Input Commands to Control Input Force Transformation

As observed from flying the Parrot Bebop 2, the quadrotor is able to perform horizontal manoeuvres while maintaining altitude, hence the total vertical thrust component must be

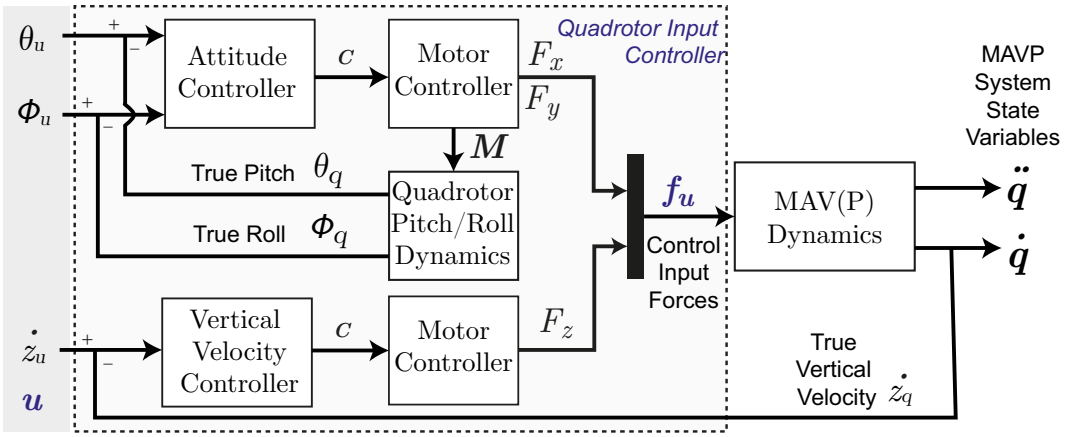


Figure 6-3: Parrot Bebop 2 specific internal control system design showing the handling of inputs $\boldsymbol{u}$ and the transformation to control input forces $\boldsymbol{f}_u$ resulting in the MAV(P) dynamics. The internal inner-loop quadrotor attitude controller with feedback is shown and is used to stabilise the quadrotor pitch and roll to the commanded inputs. Internal feedback of vertical velocity is used to stabilise the system's vertical velocity to the commanded input.

sufficient to counteract gravity. With the system mass m, transforming the thrust vector $\mathbf{T} = [0, 0, T]^\top$ by the quadrotor attitude given by $\mathbf{R}_\mathbf{E}^\mathbf{B}$ gives Equation 6-2 for the horizontal thrust components. Let the following notations for true pitch and roll be equivalent, $\theta \equiv \theta_q$ and $\phi \equiv \phi_q$

$$\begin{bmatrix} F_x \\ F_y \end{bmatrix} = \text{diag}(1, 1, 0)(\mathbf{R}_\mathbf{E}^\mathbf{B}\mathbf{T}) = T \begin{bmatrix} \sin(\theta) \\ -\cos(\theta)\sin(\phi) \end{bmatrix} \tag{6-2}$$

Additionally, Equation 6-3 must be satisfied for maintaining altitude.

$$(\mathbf{R}_\mathbf{E}^\mathbf{B}\mathbf{T})^\top [0, 0, 1]^\top - mg = 0 \tag{6-3}$$

Solving for T in Equation 6-3,

$$T = \frac{mg}{cos(\phi)cos(\theta)} \tag{6-4}$$

Substituting T in Equation 6-2, the horizontal force components are given in terms of the quadrotor's true roll ϕ and pitch θ angle.

$$\begin{bmatrix} F_x \\ F_y \end{bmatrix} = mg \begin{bmatrix} \tan(\theta)/\cos(\phi) \\ -\tan(\phi) \end{bmatrix} \tag{6-5}$$

For determining F_z, the commanded vertical velocity $\dot{z}_u$ results in the true system vertical velocity response which when differentiated gives the system's acceleration input response $\ddot{z}_u$. Equation 6-6 then describes the UAV's vertical input force in terms of the resulting vertical acceleration input $\ddot{z}_u$ and system mass m.

$$F_z = m(\ddot{z}_u + g) = F_q + mg \tag{6-6}$$

where $F_q = m\ddot{z}_u$. When there is no $\ddot{z}_u$ input, $F_z = mg$ to counteract the gravitational force.

Using the Eqs. 6-5 and 6-6 to define $\boldsymbol{f}$ as given by Eq. 5-14, the quadrotor controller model's output is defined. For the UAVP system considered $m = m_Q + m_L$ which is the sum of the

quadrotor and payload mass. This approach to modelling the inputs to force transformation was previously used in (Tobias, Alonso-mora, Domahidi, Rus, & Hilliges, 2017) for research also conducted using a Parrot Bebop 2 quadrotor.

Note that Eqs. 6-5 and 6-6 are in terms of the actual quadrotor pitch, roll and vertical acceleration. To transform the commanded inputs to the quadrotor's true response, the Parrot Bebop 2 specific quadrotor dynamics were identified as Linear Time Invariant (LTI) models denoted by Eqs. 6-7, 6-8 and 6-9. The states of the LTI models is denoted by $\boldsymbol{x}_c$. The specific LTI model state definition and structure as well as the black-box identification from empirical data collected using a Parrot Bebop 2 will be a major part of the preliminary study and is discussed in Section 6-4-1.

The attitude dynamic response for pitch and roll is given by Eqs. 6-7 and 6-8 (as mentioned before, yaw is not considered). The quadrotor's resultant vertical acceleration input $\ddot{z}_u$ is modelled as a LTI system with input $\dot{z}_u$ as defined by Eq. 6-9.

$$\theta = h_\theta(\theta_u) \tag{6-7}$$
$$\phi = h_\phi(\phi_u) \tag{6-8}$$
$$\ddot{z}_u = h_{\ddot{z}}(\dot{z}_u) \tag{6-9}$$

Substituting the quadrotor dynamics given by Eqs. 6-7 to 6-9 in Eqs. 6-5 and 6-6, the relation $\boldsymbol{u} \rightarrow \boldsymbol{f}$ is obtained,

$$\boldsymbol{f} = [F_x, F_y, F_z, 0, 0]^\top \quad \text{where} \quad \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = mg \begin{bmatrix} \tan(h_\theta(\theta_u))/\cos(h_\phi(\phi_u)) \\ -\tan(h_\phi(\phi_u)) \\ m(h_{\ddot{z}}(\dot{z}_u) + g) \end{bmatrix} \tag{6-10}$$

Note that for this study it is assumed that the definition of $\boldsymbol{f}$ given by Equation 6-10 is always substituted in the EOMs given by Equation 5-24. Unless otherwise stated when referring to the *UAVP system model* it is assumed to refer to both the quadrotor input controller model and UAVP system model.

6-2 Simulation Environment and Model Discretisation

Simulation studies of the UAVP dynamics were performed to verify the model for use in the model-based MPC that follows after this preliminary study. A simulated environment is built in MATLAB that is used to perform desk-based research using the derived model from Chapter 5. The simulation environment enables rapid development and testing of new concepts before they are implemented on a real-life setup as performing multiple runs and iterations in experimental setups is time consuming, costly and not always tractable. Also having a simulation setup allows comparative studies to be performed with experiments as to quantify how realistic a simulated model is.

To discretise the continuous-time dynamics, given by Equation 5-24, numerical integration is used. The next system state is explicitly solved using the forward Euler method. More

accurate methods such as Runge-Kutta are not used as they require multiple function evaluations of $\dot{\boldsymbol{x}}$ per simulation cycle while forward Euler only requires one. The drawback is that the first-order numerical procedure, such as forward Euler, start deviating from the actual solution quickly with a global error proportional to the step size used for simulation. However, considering the future application of the UAVP model in MPC, where the prediction horizon is generally in order of 1 second, the global error over the full horizon should be acceptable. Other integrator methods may be easily implemented with higher order Taylor expansions should the explicit Euler method not suffice in later studies. Equation 6-11 defines the forward Euler based computation of the next state $\mathbf{x}_{i+1}$ and derivative in discrete form where Δt is the simulation step size (set at 0.05 seconds to match the experimental control frequency).

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \dot{\mathbf{x}}_i \Delta t \tag{6-11}$$

The simulation may be initialised at any desired state vector $\mathbf{x}$. For most cases, the quadrotor position states x_q, y_q, z_q are pre-defined for the user with all other states zero to be in an equilibrium condition.

6-3 Experimental Environment and Workspace

Experimental studies were performed in the an indoor specially built environment with equipment used for commanding, controlling and communicating between different hardware. The experiments took place at the *Network Embedded Robotics Laboratory* at *TU Delft*.

6-3-1 Physical Space and Motion Capturing System

The indoor setup comprises a $3.0 \times 6.0 \times 2.6$m (width $\times$ length $\times$ height) usable workspace contained within an outer perimeter netting wall for safety. Figure 6-4 shows a schematic of the workspace dimensions and layout of the Motion Capture System (MCS) cameras. The MCS system comprises of 10 Optitrack Prime 17W[3] infrared based optical motion tracking cameras positioned overhead along the outer perimeter of the workspace. The maximum usable workspace is determined by all physical locations that can at least be captured by four cameras. This was determined by taking a retro-reflective marker and collecting a point cloud of tracked marker locations. Markers are tracked when at least three cameras have it in sight with the fourth camera necessary for redundancy. Figure 6-4 also shows the orientation and positioning of the inertial reference frame $\{I\}$ which is defined at the ground-level centroid of the usable workspace. The MCS provides rigid-body information in $SE(3)$ of the quadrotor, payload and any tracked obstacles. Prior to performing experiments the MCS was tuned and calibrated. The calibration results and error margins are presented in Appendix E.

6-3-2 Command, Control and Communication Architecture

To perform experiments a command, control and communication architecture was established enabling execution of controlled experiments and collection of data. The following section outlines some characteristics of the hardware and software used.

[3]Optitrack. "Prime 17W". Accessed September 19 2017. `http://optitrack.com/products/prime-17w/`

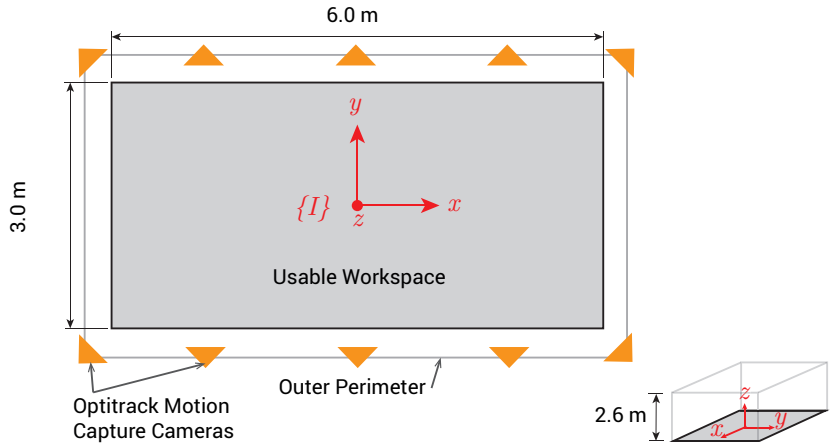


Figure 6-4: Schematic showing indoor usable workspace dimensions with inertial frame $\{I\}$ marked in center and the Motion Capture System cameras (orange triangles) laid out

Central Computer Hardware and Software

All commands, control computations and communications during the experiments were performed on a central host computer with the hardware and software itemised below;

- Hardware: Laptop with Intel Core i7-3610QM Quad-core at 2.30 GHz (3.30 GHz max), 8 GB RAM and 2.4/5 GHz capable Wi-Fi card
- Operating System and Software: Ubuntu 16.04, ROS Kinetic Kame, MATLAB R2017a

In addition to the software outlined above, several MATLAB toolboxes and ROS packages are required which have been itemised in Appendix F. An additional computer was necessary to run the Optitrack provided *Motive* software that controls and communicates with the MCS cameras to makes its data available on the ROS network.

System Architecture

Referring to Figure 6-5, all communications were hosted on the central computer. The MCS data outputs 3D rigid-body data which includes the quadrotor's pose (position and orientation), and the payload's position. The central computer was running the ROS master with packages loaded to enable the MCS rigid-body data to be read, and commands to be sent to the quadrotor. Control computations were done in real-time using a MATLAB script that published commands to a ROS topic that interfaced with the Bebop SDK. A physical game-console controller was used to facilitate manual control and override the MATLAB generated commands should that be necessary.

6-4 Experimental Procedure

Three experiments were conducted for the complete identification, verification and validation of the UAVP dynamics. Each experiment and its relevance will be outlined in more detail in its own section.

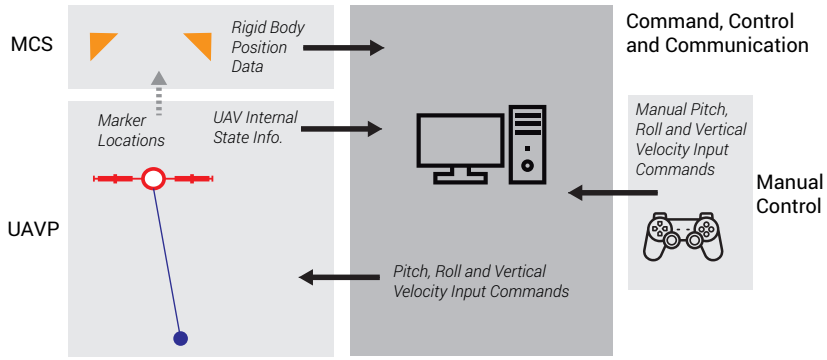


Figure 6-5: Data flow between digital systems in the experimental setup

- Identification of the UAV pitch, roll and vertical velocity dynamics
- Analysis of the cable slackening during manoeuvres
- Validation through comparison of simulated and experimental system dynamics responses

6-4-1 System Identification Method for Quadrotor Pitch, Roll and Vertical Velocity Dynamics

As mentioned in Section 6-1-3, to complete the model it is necessary to identify the UAV's attitude and vertical velocity dynamics as given by Eqs. 6-7, 6-8 and 6-9. System identification is performed using the MATLAB *System Identification Toolbox* that uses recorded input-output time traces to identify a process model, of arbitrary degree n, in an innovations state-space form as presented in Eq. 6-12. As the particular model structure implemented on the on-board controller is not known, a black box modelling approach is used where the primary model purpose is to fit the data. Therefore, the states of the identified models have no physical meaning, but rather represent abstract model parameters. The output measure is denoted by a placeholder notation $\Theta \in \mathbb{R}$ which in this experiment's case is either pitch, roll or vertical velocity. Let $u(t) \in \mathbb{R}$ be the reference pitch, roll or vertical velocity input. Input delay may be added to the system by setting a positive non-zero t_d. The intermediate state-space variables are encoded in vector $\mathbf{x}_{\boldsymbol{\Theta}} \in \mathbb{R}^n$ for which the size depends on the model order n. As experimental data is used, a disturbance component is included by signal $e(t)$ and the disturbance matrix given by K_Θ capturing dynamics unrelated to the process being identified.

$$\begin{aligned} \dot{\mathbf{x}}_{\boldsymbol{\Theta}} &= \mathbf{A}_{\boldsymbol{\Theta}}\mathbf{x}_{\boldsymbol{\Theta}}(t) + \mathbf{B}_{\boldsymbol{\Theta}}u(t-t_d) + \mathbf{K}_{\boldsymbol{\Theta}}e(t) \\ \Theta &= \mathbf{C}_{\boldsymbol{\Theta}}\mathbf{x}_{\boldsymbol{\Theta}}(t) + \mathbf{B}_{\boldsymbol{\Theta}}u(t-t_d) + e(t) \end{aligned} \tag{6-12}$$

The pitch, roll and vertical velocity models were identified individually using the same process. Two runs of the same experiment were selected to create a estimation and validation dataset. The former dataset is used by the toolbox to identify a (delayed) first, second and third order state-space model. The validation dataset is used to measure the fit/performance of the generated model through simulation of the model response to the recorded inputs in the dataset. The toolbox was configured to estimate the state-space models using *Prediction*

Error Minimisation (PEM) with a focus on *simulation* and *zero* initial state. Note that achieving a constant perfectly zero initial state is impossible due to the experimental nature of the data.

Only linear, low order models were identified as the resulting model accuracy was found to be sufficient while keeping the computational demand low for real-time implementation. A discrete state-space model would be more appropriate for simulation and control purposes due to the use of digital computers for performing both tasks, however, this requires the model sample time to be constant. This is achievable in simulation, however, for implementation purpose in a real-time system, the sample time constantly varies. Therefore, a continuous time model is identified which can then be forward simulated using numerical integrator methods.

6-4-2 Analysis of Cable Slackening During Manoeuvres

As mentioned in Chapter 5, it is necessary to verify whether cable slackening occurs when the UAV undergoes inputs under the full range. For the purpose of modelling the UAVP system, it is important to know whether the cable always remains taut (in tension) throughout the motion or there are instances where it becomes slack; if the cable switches between the slack and taut state, a switching UAVP dynamics model would be necessary. Consequently a switching-mode controller or modelling of the continuous cable dynamics would be required for proper modelling and validation of the UAVP dynamics.

It is presumed that a cable never slackens if the vector between its two end-points remains of a constant known length. From experimentally collected data, the cable length l is computed by Eq. 6-13 using the MCS provided payload and UAV positional data $\mathbf{p_L}$ and $\mathbf{p_Q}$ respectively. During experiments it was noted that the MCS collects the quadrotor's position at the centroid of the polygon formed by the four markers attached while it is assumed $\mathbf{p_Q}$ is the UAV's center of gravity. Also, the cable suspension point is under the UAV's actual center of gravity as it is physically impossible to attach the cable at the centre. To account for the offset between the measured MCS based UAV position and the actual suspension point, Eq. 6-13 is modified giving Eq. 6-14 where the offset is included in vector r_{off}.

$$l = \|\mathbf{p_L} - \mathbf{p_Q}\| \tag{6-13}$$

$$l = \|\mathbf{p_L} - (\mathbf{p_Q} + \mathbf{r}_{\text{off}})\| \tag{6-14}$$

Under the premise that a taut cable corresponds to a constant cable length, with some deviations for measurement errors, the results can show if slackening ever occurs.

6-4-3 Validation Through Comparison of Simulated and Experimental System Responses using Configuration State Reconstruction

To facilitate validation of the simulated system dynamics, it was necessary to reconstruct the state vector $\mathbf{q} = [x_Q \;\; y_Q \;\; z_Q \;\; \theta_L \;\; \phi_L]^\top$ from experimentally collected measurements. The first three states of $\mathbf{q}$ are the UAV's position which is directly available from the MCS. The

remaining states are the suspension angles ϕ_L and θ_L which are computed using Eqs. 6-15 and 6-16 respectively. The computation of these angles uses the MCS positional data of $\mathbf{p_L}$ and $\mathbf{p_Q}$. The equations are derived by inverting the kinematics describing the payload position $\mathbf{p_L}$ with respect to the UAV position $\mathbf{p_Q}$ as given by Eq. 5-5.

$$\phi_L = \tan^{-1}\left(\frac{-(x_L - x_Q)}{z_L - z_Q}\right) \tag{6-15}$$

$$\theta_L = \sin^{-1}\left(\frac{y_L - y_Q}{\|\mathbf{p_L} - \mathbf{p_Q}\|}\right) \tag{6-16}$$

The simulated and experimentally derived states time traces are then compared to identify whether the responses are consistent and/or whether there are any (significant) deviations.

6-5 Collection and Post-processing of Experimental Data

Data collected during experimentation was post-processed for noise and systematic errors to be usable for performing the system identification and validation described in the preceding sections. Both procedures are described in detail hereafter.

6-5-1 Data Collection

As presented in section 6-3-2, all communications lines in the experimental setup either start or end at the central computer, therefore, logging of all data occurred there. Data logging was performed during the experiment as part of the MATLAB control algorithm that was receiving data from ROS topics published by the MCS and quadrotor. As data was logged in-the-loop, the data sampling frequency was inherently linked to the high-level control loop frequency. This was not a problem considering the control frequency was consistently 19 Hz. Future iterations of the code will likely benefit from separating control and logging into two independent scripts running as two processes.

The type of odometry data and metrics collected included;

- Real-time clock samples (MATLAB on host computer)
- Quadrotor's pose which includes position and orientation in Euler angles (MCS)
- Payload's position (MCS)
- Quadrotor commanded pitch, roll and vertical velocity inputs (MATLAB on host computer)

Important to note is that commanded inputs are recorded at the time they are generated by the MATLAB control script running on the host computer. The UAV will execute this command once it is received and may have earlier commands buffered till then so there is a time lag component. Additionally, the UAV may skip inputs if they are received at a higher rate than can be physically executed. Unfortunately, it is currently not possible to obtain the timed UAV executed inputs due to limitations of the Bebop SDK. Therefore, from here on forward it is assumed that all inputs sent to the UAV are executed within a reasonable time

period. Additionally, the quadrotor's orientation in Euler angles may also be obtained from the built-in gyroscopes, however, the current transmission rate from the UAV is limited to 5 Hz using the SDK and ROS (Monajjemi, Mani, 2015). This sampling frequency is too low for reconstructing the continuous system dynamics for system identification purposes (as will become clear in Chapter 7).

6-5-2 Data Post-processing

Data collected was directly logged into MATLAB arrays that could readily be imported into the workspace and manipulated in MATLAB. First an analysis was performed to identify any missing data points or outliers resulting from to the digital nature of recording the metrics. Any issues were rectified by removing or estimating the missing value through interpolation using locally relevant data points. This step ensured that erroneous data was not used for the system identification and validation procedure adversely affecting the results obtained.

The data was then processed using low-pass filtering to remove high-frequency noise components. The data was filtered/smoothed using the MATLAB `smooth` function which is a simple moving average filter with a span of 5 measurement points. The simple filter was sufficient for the purposes of this study, however, one may explore more advanced low-pass filter constructions and/or finely tune the moving average span based on the expected frequency components in the data.

Additional de-trending post-processing was performed on the estimation and validation dataset for the UAV pitch and roll system identification. The details of this is outlined in Appendix G.

Chapter 7

Results and Discussion

Three sets of experiments were performed as described in Section 6-4. Section 7-1 presents the system identification results for the pitch, roll and vertical velocity dynamics of the Parrot Bebop 2 quadrotor. Section 7-2 looks at the results to answer whether cable slackening occurs for inputs over the allowable range. Lastly, Section 7-3 addresses the comparison of simulation and experimental studies of the UAVP response dynamics to validate the model.

7-1 Identified Quadrotor Pitch, Roll and Vertical Velocity Models

The identification of the Parrot Bebop UAV pitch, roll and vertical velocity dynamics using the method outlined in Section 6-4 is described concluding with the derived models. First the UAV step response is assessed to gain some insights into the system.

7-1-1 Step Response and System Bandwidth

To identify basic time-domain characteristics of the system pitch, roll and vertical velocity response, experiments involving step inputs were performed to collect system response data. The following experimental runs were performed:

- Pitch and Roll, 10° and 15° step inputs individually for 1 second duration
- Vertical velocity 1 m/s step input for 1 second duration

Due to limitations imposed by the available indoor workspace, it was not possible to perform step input experiments at higher pitch and roll angles as there was insufficient translational space available. Due to the same reasons, it was not possible to lengthen the input duration.

The experimentally derived rise time for the step response is summarised in Table 7-1. The rise time definition followed is the time taken for the response to go from 10% to 90% of the

final steady-state value which is set as the commanded input angle. As the quadrotor is not fully symmetrical along all axes, it is expected that the pitch and roll dynamics are similar but not equal. The results show that the pitch response rise time is more consistent than the roll response, however, on average the rise time of both pitch and roll are comparable at around 0.33s. The roll response is slightly slower which is consistent with the expectation as the quadrotor is generally designed to be most agile along the body x axis. From the relatively long pitch and roll rise times, it can be concluded that the pitch and roll inputs are not instantly tracked by the attitude stabilisation system of the UAV. Therefore, modelling the pitch and roll dynamics is necessary for developing an accurate UAVP model. The same conclusions can be drawn for the vertical velocity input response which has an average rise time of 0.42s which is relatively slow. The vertical velocity response is slow as it requires all four motors to increase their rotational velocity such that extra thrust is generated.

Table 7-1: Rise time of Parrot Bebop 2 quadrotor's pitch, roll and vertical velocity response for multiple runs

Run	**Step Response Rise Time [s]**				
	Pitch 10°	**Pitch** 15°	**Roll** 10°	**Roll** 15°	**Vertical velocity** 1m/s
1	0.32	0.33	0.34	0.30	0.42
2	0.34	0.33	0.32	0.40	0.44
3	0.34	0.30	0.46	0.29	0.41
Average	0.33	0.32	0.37	0.33	0.42

The rise times also give insight into the frequency response bandwidth of the pitch and roll attitude stabilisation system. As a rule of thumb, the step rise time relates to the system bandwidth BW by Eq. 7-1 (National Instruments, 2007). Therefore, the pitch and roll system bandwidth is 1.1 Hz (rounded to two significant figures). The bandwidth for the vertical velocity control system is 0.83 Hz. The bandwidth is important for designing the system identification input signals and setting the required sampling frequency to recreate the continuous-time dynamics from sampled data.

$$BW[\text{Hz}] = \frac{0.35}{\text{rise time [s]}} \tag{7-1}$$

As a continuous-time response is captured by discrete samples, it is also necessary to achieve a sampling rate greater or equal to the Nyquist rate. According to Nyquist-Shannon's sampling theorem, the minimum sampling rate is equal to two times the signal bandwidth, therefore, sampling must occur at a minimum of 2.2 Hz to capture the continuous time system dynamics. Given the pitch and roll system bandwidth is higher than for vertical velocity, achieving the stricter sampling rate automatically guarantees the sampling rate is sufficient for the latter system. From experimental trials, the minimum observed sampling rate using the workspace setup as described in Section 6-3 is 19 Hz which is well above the required sampling rate. Intuitively the sampling rate seems low, however, this rate is achieved as the higher frequency MCS data is communicated over the ROS network and processed in MATLAB before being logged into a data file. The in-the-loop processing also converts the external MCS data into relevant measures including the UAV Euler angles. Therefore, the control loop speed of MATLAB restricts the logging/sampling speed. Unfortunately, the high frequency MCS data

that is sent at around 60 Hz to MATLAB between logging instances is lost. For the purposes of the experiments performed, the current 19 Hz sampling rate is sufficient. To significantly increase the attained sampling rate, raw data from the MCS would need to be externally (not within MATLAB) be logged. The data will then need to be post-processed into the required measures.

7-1-2 System Identification of UAV Pitch and Roll Dynamics

Considering the pitch and roll system bandwidth of 1.1 Hz, a square wave input was designed to excite the system to collect experimental response data. Proper system identification requires excitation of the system at various input frequencies and a square wave achieves this as it is composed of multiple sine waves at various frequencies and amplitudes (this follows from its Fourier series decomposition). Three state-space systems of orders 1 to 3 in the form of Eq. 6-12 were identified on the estimation dataset using the MATLAB *System Identification Toolbox*. The estimation and validation dataset contain the system response to an input square wave with an amplitude of 5°, frequency of 0.5Hz and duration of 15 s. The input frequency was selected to be 0.5 Hz as to fall within the system bandwidth of 1.1 Hz. The data has been cut-off at the 15 s mark as to exclude the square wave signal's return to the 0° input; reason for this exclusion is that the quadrotor's attitude controller not only brings the pitch/roll back to 0°, but also triggers a position hold controller so there is no remaining translational motion. The position hold is achieved by generating a negative pitch/roll against the motion and this corrective action involves an extra outer loop to the attitude controller which is not currently being modelled.

Figures 7-1 and 7-2 show the designed input square wave with the corresponding simulated and experimental validation data responses. For both pitch and roll, three non-delayed model of increasing order and a delayed first-order model, with 0.1 s input delay t_d, were identified. Table 7-2 shows the quantified overall model fit, as a percentage, with the estimation and validation dataset. Percentage fit is the agreement between the model generated output and the experimental data output. The measure used is Normalised Root Mean Square Error (NRMSE) which is part of the MATLAB `goodnessOfFit` function. Additionally, the model quality is explored by considering the output residual auto-correlation and cross-correlation between the input and output residuals as given in Appendix I. Delayed second and third order state-space models were also identified, however, there was minimal to no improvement in the model fit as shown in Appendix H so they have been excluded from the results discussed here. As shown in Table 7-2, the delayed first-order model shows significant improvement in the model fit compared to the non-delayed version. Analysing Figures 7-1 and 7-2 and the associated model fits, models of all order are able to capture the system dynamics sufficiently well with at least a 80% fit. The model fits on pitch dynamics is better than roll which can be attributed to the more consistent pitch response compared to roll for the square wave's repetitive inputs as used for experiments. The same consistency issue with the roll response was observed in the results of Section 7-1-1. Additionally, note how the UAV's experimental pitch and roll response converges to a final value lower than the commanded input value of ±5° (±0.087 rad). This also means the overall gain of the derived models is less than 1. Over the repeated inputs given to the UAV in the square wave form, the UAV does not consistently converge to the same tilt angle, therefore, there is likely some error in the UAV's

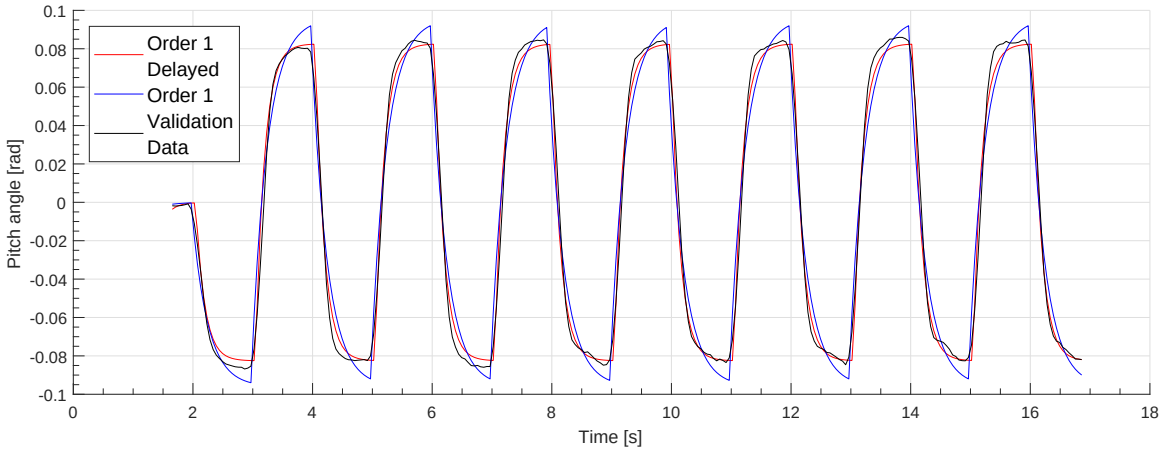
internal state observation. Another important conclusion to be drawn is that the model fit on the estimation and validation data is very similar indicating there is no apparent model over-fitting to the estimation data.

Further analysis shows that second order model is the most suitable pitch and roll model order considering the fit and computational load. The conclusions and observations made are relevant for both the pitch and roll system given how similar the systems respond to inputs. The residual analysis in Appendix I, shows that the second and third order model's quality is good given that the majority of the residuals fall within the 99% confidence intervals with some peaks beyond the intervals. The first order model performs well, however, it has a relatively low model fit at around 80-85% when compared to the second and third order model. Also considering the response shown in Figures 7-1 and 7-2, the first order model output overshoots the observed experimental response significantly. The delayed first order model fit is more comparable with the second and third order models, however, it slightly under-performs in terms of model fit. The improvement in model fit from the second to third order model is small while the additional order makes the third order model more computationally demanding. Following from the observations made, the second order pitch and roll model is selected for which the state-space representation is given by equations 7-2 and 7-3 respectively. The pitch and roll model inputs are given by θ_u and ϕ_u with the intermediate states encoded in vectors $\mathbf{x}_\theta$ and $\mathbf{x}_\phi$. These models address the earlier identified unknown UAV attitude dynamics functions formulated as Eqs. 6-7 and 6-8. To improve real-time performance of the script and simplify the model, the delayed first-order model may be considered in subsequent studies as it achieves a relatively good fit with the benefit of being one order lower (less computationally demanding).

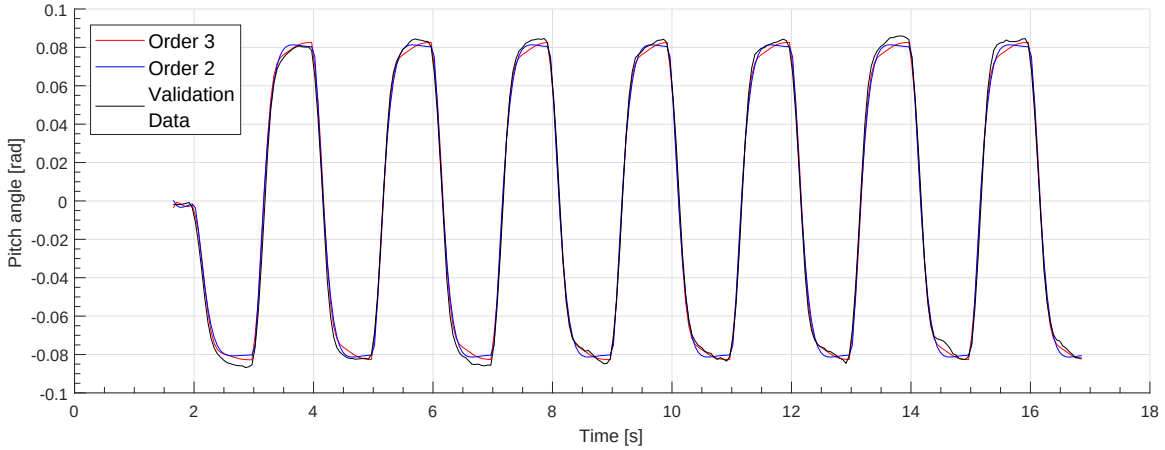
$$\begin{aligned} \dot{\mathbf{x}}_\theta &= \begin{bmatrix} -4.301 & -2.877 \\ 10.92 & -10.37 \end{bmatrix} \begin{bmatrix} x_{\theta,1} \\ x_{\theta,2} \end{bmatrix} + \begin{bmatrix} -0.6893 \\ -16.32 \end{bmatrix} \theta_u + \begin{bmatrix} 15.37 \\ 23.75 \end{bmatrix} e(t) \\ \theta &= \begin{bmatrix} 1.763 & 4.586 \times 10^{-3} \end{bmatrix} \begin{bmatrix} x_{\theta,1} \\ x_{\theta,2} \end{bmatrix} + [0]\,\theta_u + e(t) \end{aligned} \tag{7-2}$$

$$\begin{aligned} \dot{\mathbf{x}}_\phi &= \begin{bmatrix} -2.789 & -4.978 \\ 9.302 & -13.72 \end{bmatrix} \begin{bmatrix} x_{\phi,1} \\ x_{\phi,2} \end{bmatrix} + \begin{bmatrix} -5.41 \\ -18.04 \end{bmatrix} \phi_u + \begin{bmatrix} 13.57 \\ 1.152 \end{bmatrix} e(t) \\ \phi &= \begin{bmatrix} 1.996 & 0.4657 \end{bmatrix} \begin{bmatrix} x_{\phi,1} \\ x_{\phi,2} \end{bmatrix} + [0]\,\phi_u + e(t) \end{aligned} \tag{7-3}$$

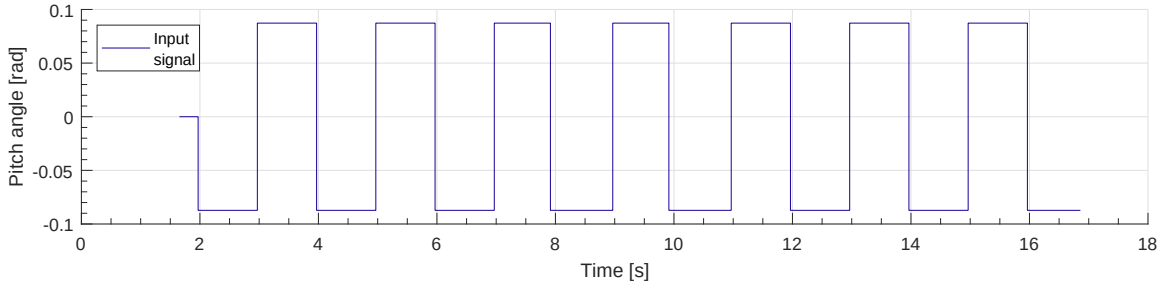
Table 7-3 shows the poles, zero, damping ratio and natural frequency of the resulting second order state-space systems. Both the pitch and roll models are under-damped with a damping ratio approaching critical damping, therefore, there is very little oscillatory behaviour in the response as also seen in Figures 7-1 and 7-2. This can also be deduced from the poles in the complex left-plane that also confirm the system is stable. The non-minimum phase zero introduces some undershoot behaviour, however, as it is very far right into the complex plane, it has minimal effect on the overall system behaviour as it does not cancel any effects from the more dominant poles (closer to the origin).



(a) First order models and validation data response to input signal

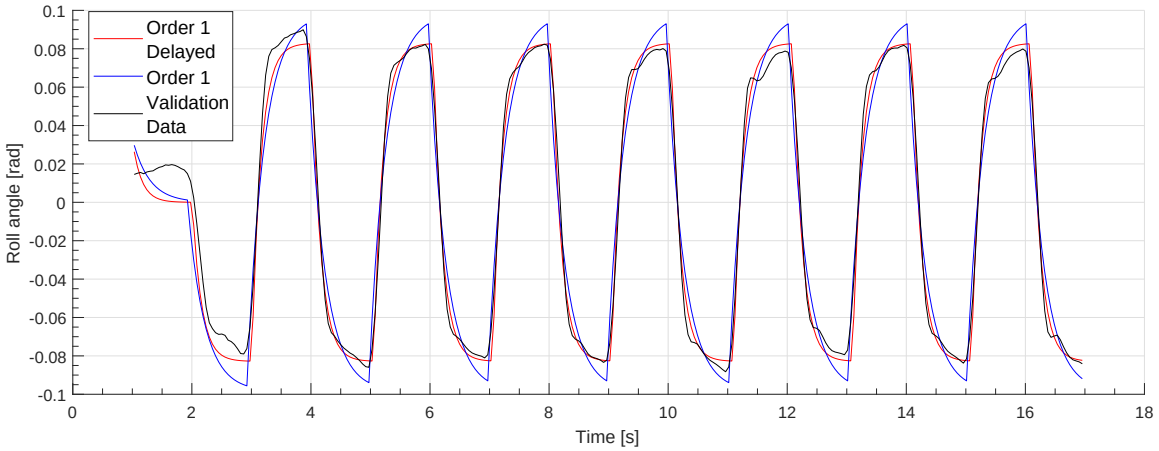


(b) Second and third order models, and validation data response to input signal

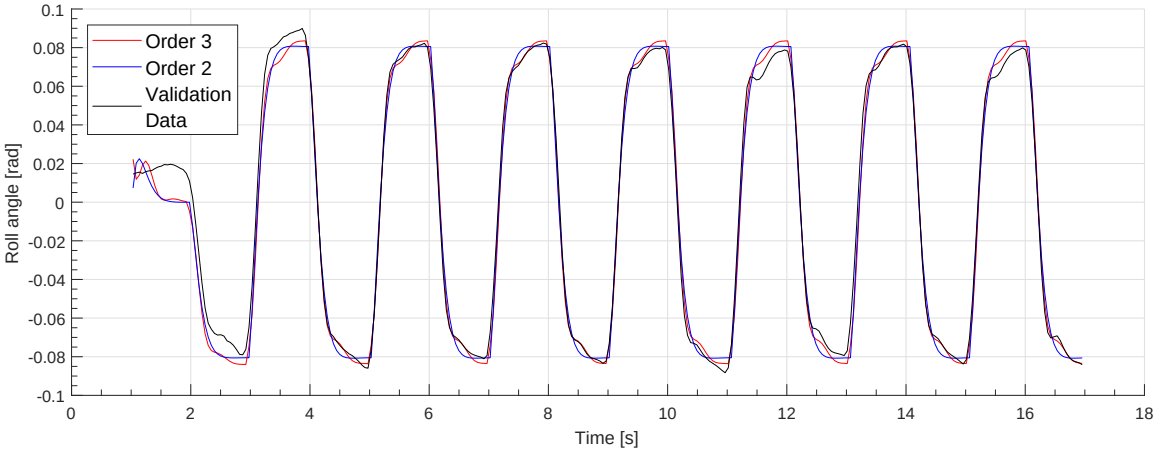


(c) Input signal square wave; amplitude: 5° (0.087 rad), frequency: 0.5 Hz, duration: 15 s

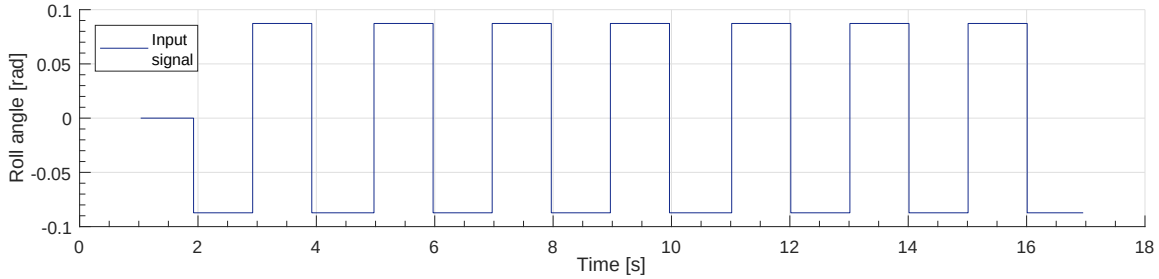
Figure 7-1: First (delayed), second and third order pitch state-space model response comparison to experimental validation data



(a) First order models and validation data response to input signal



(b) Second and third order models, and validation data response to input signal



(c) Input signal square wave; amplitude: 5° (0.087 rad), frequency: 0.5 Hz, duration: 15 s

Figure 7-2: First (delayed), second and third order roll state-space model response comparison to experimental validation data

Table 7-2: First, second and third order state-space model fit as NRMSE percentage to estimation (est.) and validation (val.) dataset. First-order input delays are 0.10 s for pitch/roll model, and 0.12 s for vertical velocity.

System	Dataset fit of identified state-space model [%]							
	Order 1		Order 1 delayed		Order 2		Order 3	
	Est.	Val.	Est.	Val.	Est.	Val.	Est.	Val.
Pitch	84.68	84.55	91.72	91.08	94.59	93.81	95.93	94.96
Roll	82.34	81.69	89.95	88.54	91.09	89.08	92.76	90.17
Vertical Velocity	77.12	76.42	87.52	89.01	91.38	91.56	91.19	92.19

Table 7-3: Poles and zero of identified pitch, roll and vertical velocity second order state-space system models

System	Zero	Pole(s)	Damping Ratio [-]	Natural Frequency [Hz]
Pitch	54.1	-7.34 ± 4.71i	0.841	1.39
Roll	32.6	-8.25 ± 4.05i	0.898	1.46
Vertical Velocity	93.0	-3.54 ± 3.07i	0.755	0.74

7-1-3 System Identification of UAV Vertical Velocity Dynamics

Following the same system identification process as for the pitch and roll dynamics model, the vertical velocity model is also identified. The estimation and validation dataset contain the system response to a pulse input of width 1s and an amplitude of 1m/s.

Figure 7-3 shows the designed input pulse and the corresponding simulated response using state-space models of different order. Table 7-2 shows the quantified overall model fit, as a percentage, with the estimation and validation dataset. A first order delayed model with a 0.13 s input delay t_d is included as there is significant improvement in the fit. As previously noted in Table 7-1, the vertical velocity system rise time is higher, therefore, it is consistent that the input delay is longer when compared to the pitch and roll system models. Looking at Table 7-2, the data fit on the estimation and validation dataset is comparable indicating that model over-fitting is not an issue. The first order model has a relatively poor fit at around 76-78% when compared to the second and third order models at 91-93%. Including the input delay in the first order model vastly improves the model fit making it comparable to the second and third order models. However, as seen in Figure 7-3, the (non-)delayed first order models show an overall good response amplitude, however, the response shape deviates from the experimental data. The second and third order models are able to capture the more gradual damped nature by which the vertical velocity response evolves. The delayed first order model may be considered to achieve real-time performance at the loss of some accuracy in subsequent studies as it is computationally less demanding due to the lower order. Comparing the second and third order model, the model fit is equivalent, therefore, the second order model is the all round better model. Considering the model quality, as presented in Appendix I, the third order model is better as the residuals fall more often into the confidence intervals, however, the second order model is only slightly worse. The second order vertical velocity state-space model is given by Eq. 7-4 where $\dot{z}$ is the UAV's actual vertical velocity.

$$\begin{aligned} \dot{\mathbf{x}}_{\dot{\mathbf{z}}} &= \begin{bmatrix} -4.875 & 1.848 \\ -6.062 & -2.203 \end{bmatrix} \begin{bmatrix} x_{\dot{z},1} \\ x_{\dot{z},2} \end{bmatrix} + \begin{bmatrix} 0.6029 \\ 3.681 \end{bmatrix} \dot{z}_u + \begin{bmatrix} 21.93 \\ 83.78 \end{bmatrix} e(t) \\ \dot{z} &= \begin{bmatrix} 4.029 & -0.7253 \end{bmatrix} \begin{bmatrix} x_{\dot{z},1} \\ x_{\dot{z},2} \end{bmatrix} + [0]\, \dot{z}_u + e(t) \end{aligned} \tag{7-4}$$

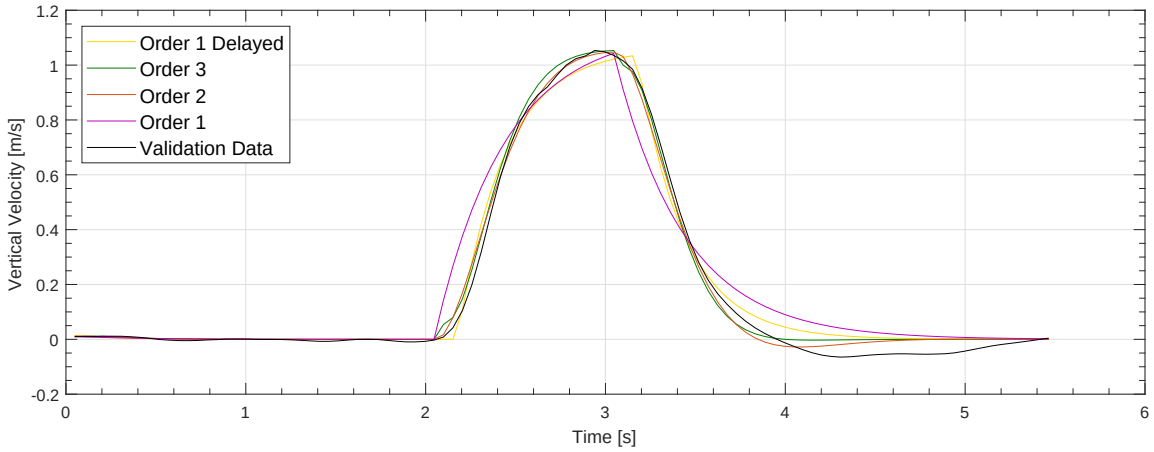
Table 7-3 shows the poles, zero, damping ratio and natural frequency of the resulting second order state-space system. The vertical velocity response is under-damped while more towards being critically damped, hence, there is little oscillatory response behaviour as seen in Figure 7-3. As the poles are in the complex left-plane, the system is stable, and the non-minimum phase zero being very far right in the complex plane has little effect on the behaviour resulting from the dominant poles.

A peculiar and interesting observation is made from Figure 7-3, the UAV responds to a 0 m/s vertical velocity input by flying down (negative vertical velocity). When given a 0 m/s vertical velocity command, the UAV control system controls the altitude rather than maintaining a zero velocity as it cannot directly sense velocities. Therefore, returning the input to the nominal state 0 m/s triggers an additional altitude hold control loop which is not accounted for in the model derived using this system identification scheme. Contrary to the roll and pitch angle which the UAV can obtain from built-in gyroscopes, sensor fusion and estimation must be performed to derive the UAV's current vertical velocity. To achieve this the Parrot Bebop 2 has a barometer, ultrasound based altimeter and accelerometers (PaparazziUAV, 2017). Therefore, due to the estimation involved, it is expected that the executed vertical velocity will be inconsistent over repeated input commands of the same magnitude.

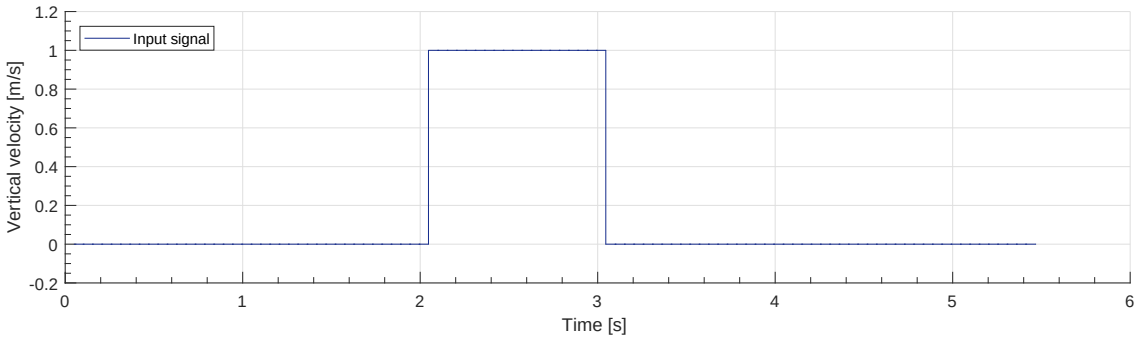
7-2 Analysis of Cable Slackening During Manoeuvres

To identify whether the cable slackens during foreseeable manoeuvres, flights were performed with gradual and aggressive inputs allowing the payload to swing freely. Using Eq. 6-14, the cable length can be computed over the duration of the experimental flight. From measurements taken from the UAVP system setup, the offset vector was determined to be $\mathbf{r}_{\text{off}}^{\mathbf{B}} = [0 \;\; 0 \;\; -0.04]^\top$ in meters in the $\{B\}$ (body) frame. As Eq. 6-14 requires the offset in the inertial frame $\{I\}$, the necessary transformation using the measured attitude Euler angles was performed to give $\mathbf{r}_{\text{off}}$. There was also an offset between the measured payload position and suspension point on the payload of 0.025 m, therefore, this value was subtracted from the computed cable length.

Figure 7-4 shows the computed cable length and suspensions angles for a section of flight recorded with the pitch, roll and vertical velocity inputs given by the third sub-plot. The flight was performed on level altitude with combined gradual and aggressive pitch and roll inputs. The maximum pitch/roll angle and vertical velocity amplitude was capped at 20° (0.35 rad) and 1 m/s respectively. As mentioned in Section 7-1-1, these inputs are a practical limit considering the limited traversable indoor workspace. This input range will also be used for constraining the MPC generated high-level commands in the subsequent study. Before performing the experiment, the UAVP was brought to a nominal hover state such that there was minimal residual payload swing. Ideally the payload would be stationary, however, this



(a) Models and validation data response to input signal



(b) Input signal pulse; amplitude: 1m/s, pulse width: 1s

Figure 7-3: First (delayed), second and third order vertical velocity state-space model response comparison to experimental validation data

is not practically achievable. As seen in Figure 7-4, the cable length varies during the run, however, it only varies about 1 cm from an average value of 0.755 cm, or about 1.3% of the nominal value. Also note that the measured cable length is 0.77 cm, as given in Table 6-1, however, here the cable on average appears to be shorter. The discrepancy likely results from the cable shortening after attaching the string with a knot on the UAV and payload suspension point. Also the offset vector measurement is not very accurate, so there is error introduced through that measure.

It cannot be conclusively said that the cable never becomes slack during the experiment, however, the variations in computed cable length is only around 1.3%, so that may be attributed to random errors, and biases introduced by the offsets. If there was significant slackening of the cable, the measured length between the UAV and payload would have more dramatically been affected, however, this was not the case. Therefore, assuming the cable remains taut is sufficient for the UAV command input ranges considered in this study. Slackening is generally observed when the payload swings aggressively (high suspension angles) combined with rapid acceleration of the vehicle to which the payload is attached. In this case, the motion of the two rigid bodies decouple causing the cable to slack. This sort of behaviour was not observed

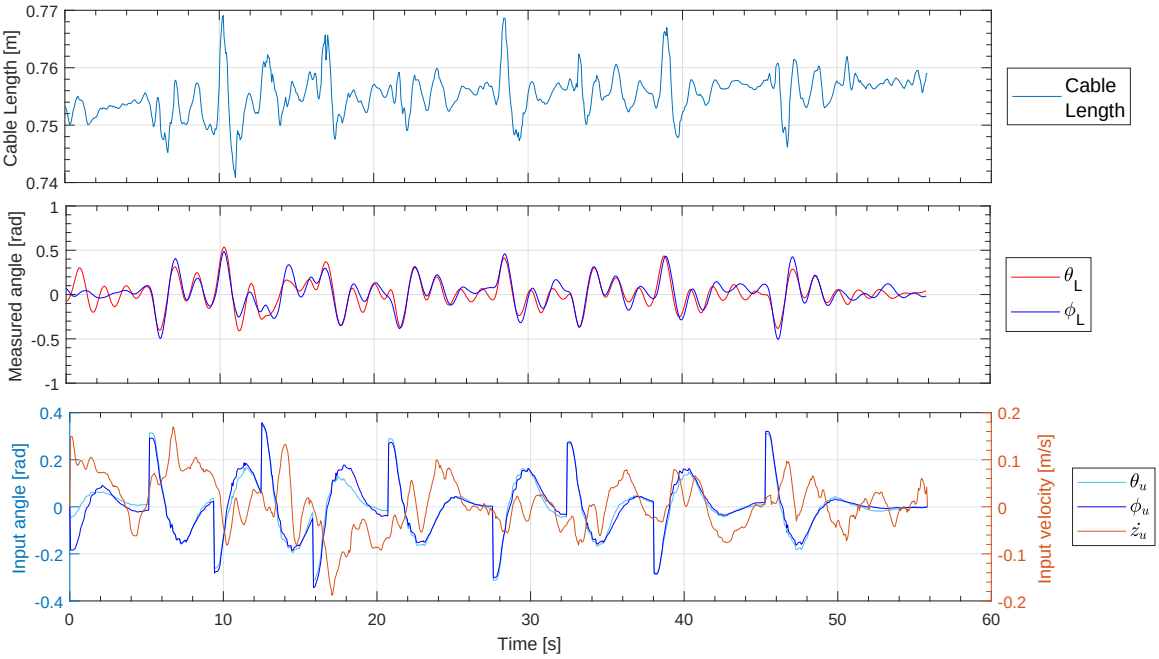


Figure 7-4: Payload suspension cable length, suspension angles and UAV commanded input pitch θ_u and roll ϕ_u angles, and vertical velocity $\dot{z}_u$ for gradual and aggressive manoeuvres

during the experiments performed as the vehicle is unable to perform such agile behaviour under the limited input range. Future studies can look at the slackening behaviour in more detail by measuring the actual tensile force in the cable which clearly indicates the cable becomes slack when there is zero force. Due to unavailability of such measurement capabilities, this study was not performed for this research.

7-3 Validation of UAV-Payload Model under Experimental Trials

Using results from the system identification of the pitch, roll and vertical velocity dynamics and the newly gained insights about the cable taut/slack dynamics, the final step involved validation of the obtained UAVP model using experimental data.

7-3-1 Quadrotor Input Control Model Implementation

The identified state-space models relating the commanded input to actual UAV state pitch, roll and vertical velocity were implemented in the simulation environment. As outlined in Section 6-2, the UAV's true pitch θ, roll ϕ and vertical acceleration input $\ddot{z}_u$ relate to the control forces $\boldsymbol{f}$ acting on the UAV system by Eq. 6-10. The true θ and ϕ states can now be modelled from the input θ_u and ϕ_u by Eqs. 7-2 and 7-3 respectively provided the initial UAV θ and ϕ state is known. The remaining equation is a relation between the vertical velocity $\dot{z}_u$ and acceleration $\ddot{z}_u$ inputs. Equation 7-4 currently models the vertical velocity dynamics provided the input and initial condition. To obtain the vertical acceleration input as output,

the differential output of Eq. 7-4 must be taken resulting in the state-space model as given by Eq. 7-5.

$$\begin{aligned}\dot{\mathbf{x}}_{\ddot{\mathbf{z}}} &= \begin{bmatrix} -6.767 & -6.546 \\ 3.031 & 0.311 \end{bmatrix} \begin{bmatrix} x_{\ddot{z},1} \\ x_{\ddot{z},2} \end{bmatrix} + \begin{bmatrix} 38.75 \\ 1.841 \end{bmatrix} \dot{z}_u \\ \ddot{z}_u &= \begin{bmatrix} 0.620 & 4.07 \times 10^{-2} \end{bmatrix} \begin{bmatrix} x_{\ddot{z},1} \\ x_{\ddot{z},2} \end{bmatrix} + [-0.241]\, \dot{z}_u \end{aligned} \tag{7-5}$$

Note that by substituting the output $\ddot{z}_u$ in Eq 6-6, the vertical control force $\boldsymbol{F}_q = m\ddot{z}_u$ with UAVP mass m can also be defined as the output of model Eq. 7-5. This model addresses the earlier identified unknown vertical acceleration input relation given by Eq. 6-9. The resulting system zero is at $s = 93.04$ and poles are at $s = -3.54 \pm 3.07\text{i}$ with damping ratio 0.755 and natural frequency of 0.74 Hz, therefore, it is an under-damped and stable system.

The three previously mentioned unknown functions given by Eqs. 6-7, 6-8 and 6-9 are now defined completing the Quadrotor Controller model. The three state-space models, Eqs. 7-2, 7-3 and 7-5 were implemented in simulation and the response simulated numerically using an (explicit) forward Euler algorithm to keep the computational complexity low. A Runge-Kutta numerical method has been implemented in code, however, it has not been used for the simulations performed and the results presented in this preliminary report. During trials it was determined that in experiments the loop step-size averaged around 0.05 s which was small enough that the forward Euler method did not deviate significantly from the actual continuous function value.

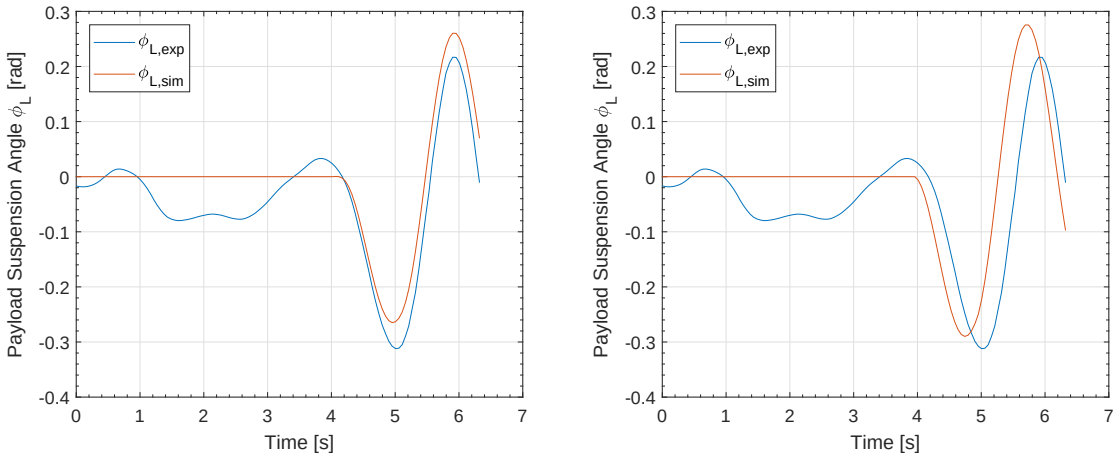
7-3-2 Comparison of Simulated to Experimental UAVP Response

The complete simulation with the identified UAV dynamics included was used to perform studies comparing the experimental and simulated UAVP response. As the payload motion is inherently linked to the UAV motion and vice versa, the improved UAV model should reflect in an improved modelling of the payload dynamics. To study this, simulated and experimental step response data was collected to study how the payload suspension angle evolves when the identified UAV dynamics are included and excluded in the simulated response.

For simulation purposes, the payload was assumed to be in a nominal equilibrium state (zero suspension angles) with the UAV stationary and hovering. These idealised conditions are not realisable in experimental conditions, however, effort was taken to bring the system as close to nominal as possible. Being a causal system, all the residual undesirable motion has an effect on the obtained results to a certain degree. Multiple runs were performed so that at least one run could be selected where the system was close to a nominal state.

The first experiment involved a step 10° (0.175 rad) pitch input on the UAV at 4 s resulting in the response seen in Figure 7-5. The simulated to experimental response fit is computed as the NRMSE starting from 4 s till the end of the run. The UAV only moves in the $\{I\}$ inertial frame's positive x direction so the payload ideally only responds with a ϕ_L angle. The experimentally measured θ_L angle remained below 2.3° (0.04 rad) throughout the experiment which is acceptable and has negligible effect on the ϕ_L response. The simulated responses presented in Figure 7-5a includes the identified UAV dynamics while in 7-5b they are excluded. Figure 7-5b shows how exclusion of the dynamics results in an immediate payload response to the input while in reality there is a slight delay before the payload responds which is captured

by including the damped dynamics as seen in Figure 7-5a. Overall the simulated response frequency is equivalent to experimental data with the response in Figure 7-5a showing a much better fit at 70.9% compared to 21.7%. As seen in Figure 7-5a, the simulated response is approximately a shifted version of the experimental response, this may be attributed to the residual oscillatory energy in the experimental system at the 4s mark when the step input is executed. Given the response appears shifted and does not oscillate about a zero value, there is likely a bias in the measurement which could be identified and removed. It cannot be said for certain if this is the case, therefore, the results are presented as-is without removing a bias.

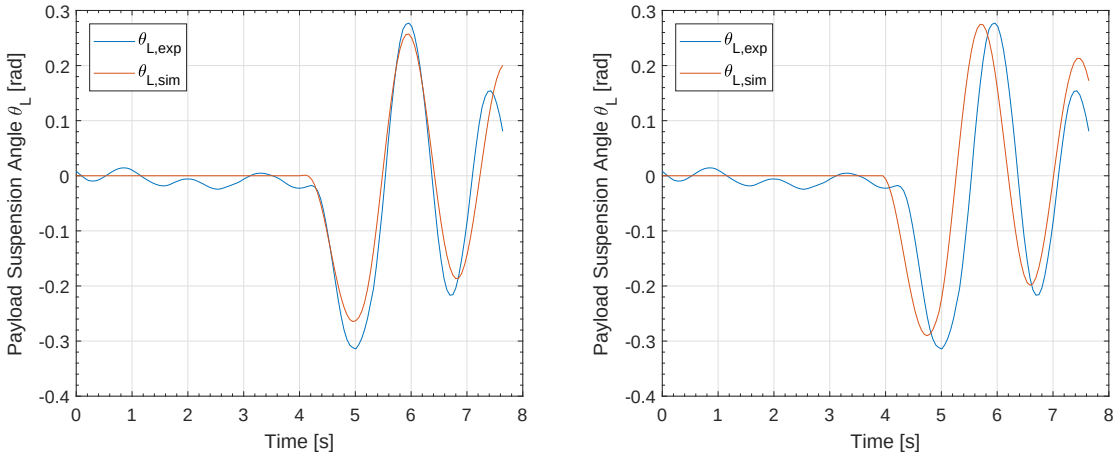


(a) With identified dynamics, 70.9% fit **(b)** Without identified dynamics, 21.7% fit

Figure 7-5: Experiment (exp) and simulation (sim) payload suspension angle for UAV pitch step input $\theta_u = 10°$

Similar to the first experiment, the second experiment involved a step 10° (0.175 rad) roll input on the UAV at 4 s resulting in the response seen in Figure 7-6. The same NRMSE fit was computed in this case. The UAV only moves in the $\{I\}$ inertial frame's positive y direction so the payload ideally only responds with a θ_L angle. The measured ϕ_L angle remained below 5.8° (0.1 rad) throughout the experiment which is acceptable. The simulated responses presented in Figure 7-6a includes the identified UAV dynamics while in 7-6b they are excluded. Again, Figure 7-6b shows how exclusion of the dynamics results in an immediate payload response to the input while in reality there is a slight delay before the payload responds which is captured by including the dynamics as seen in Figure 7-6a. Again, the simulated response frequency is equivalent to experimental data with Figure 7-6a showing a much better fit at 72.3% compared to 31.5%. As there was less residual oscillatory energy, the experimental amplitude response is more comparable. The simulation shown in Figure 7-6a shows a slightly weaker amplitude response while being less damped. This could be as the real UAV accelerates faster and the payload experiences more drag than modelled in simulation.

The experiments that have been discussed till now have tested the simulated response to discrete and simple step pitch or roll inputs. Figure 7-7 shows the simulation and experimental response of the UAVP to combined gradual and aggressive, roll and pitch inputs. The experimental data has previously been presented in Figure 7-4 which also shows the input



(a) With identified dynamics, 72.3% fit **(b)** Without identified dynamics, 31.5% fit

Figure 7-6: Experiment (exp) and simulation (sim) payload suspension angle for UAV roll step input $\phi_u = 10°$

commands. Again care was taken to bring the system to a nominal state before initiating the experiment as the simulation assumes a nominal initial state. Before analysis of the results, it is important to note that due to causality and the dead-reckoning UAVP dynamics simulation process, the simulated response is subject to cumulative errors. Given the experiment elapsed for 56 seconds, there is a large span over which accumulation of errors occurs.

Observing Figure 7-7, the initial impression is that the overall fit is decent with the simulation showing a comparably shaped simulation and experimental response. As also noted in the step response data, the response amplitudes do not always agree which is also the case here. On the other hand, the oscillatory frequency is equivalent. The simulated end experimental responses deviate the most from each other when there is minimal input to system such that the suspension angles are low ($< 6°$ or < 0.1 rad) which occurs for ϕ_L from 0 to 5 s, and for ϕ_L and θ_L from 50 to 56 s. This interesting observation indicates that the model is able to more accurately capture the UAVP motion when the inter UAV-Payload dynamics become dominant, i.e. when the UAV accelerates/decelerates significantly resulting in large payload swinging motions. When the UAV is idle or undergoing very small inputs, the model and actual response deviate which is expected. Taking the measurements from 0 to 5 s, looking back at Figure 7-4, notice how the roll input is significantly larger than the pitch input, this translates to larger θ_L angles than ϕ_L. Therefore, the simulation is able to capture the θ_L response much better than the ϕ_L response. When there are very small accelerations in a given direction, other unmodelled phenomenon become more dominant factors affecting the UAVP dynamics, consequently, there is greater disparity between the simulated and experimental response.

Given this observed behaviour, an experiment was set-up to perform flight under dominantly UAV pitch inputs with significantly smaller roll inputs. Figure 7-8 shows the simulation and experimental response of the UAVP to inputs of that nature. As the postulate states, it is expected that the experimental ϕ_L response will be captured by the simulation while there is more disparity in the θ_L response. This expected behaviour is also clearly observable in the

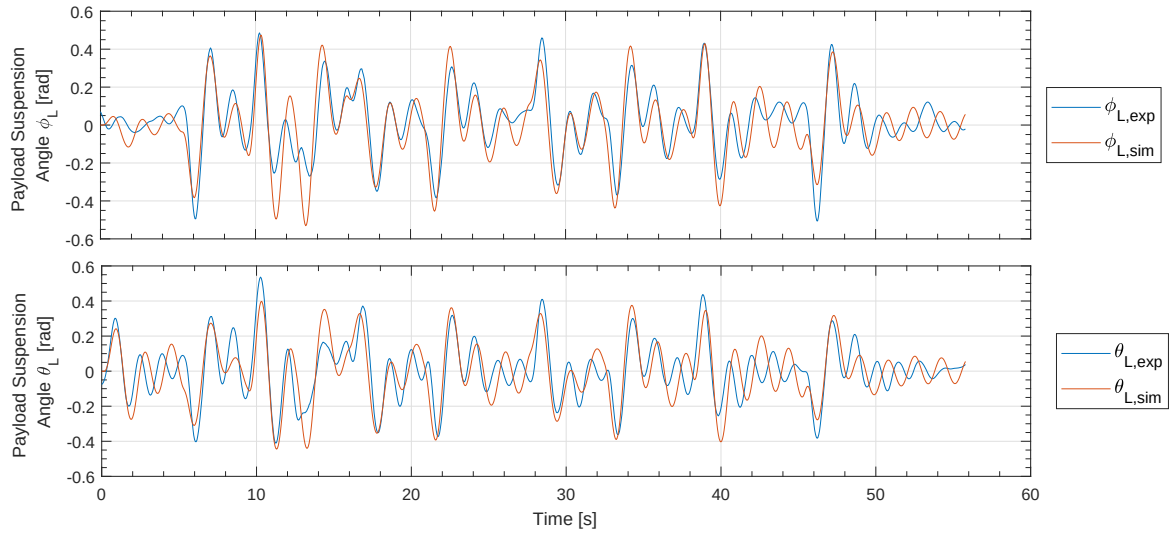


Figure 7-7: Payload suspension angles in experiment and simulation for combined pitch and roll UAV inputs, simulated ϕ_L fit of 41.0%, θ_L fit of 31.1%

collected response data in Figure 7-8. Note how the θ_L is almost always $< 6°$ or < 0.1 rad which are very small suspension angles. Compare this to ϕ_L which reaches angles around $35°$ (0.5 rad) showing that the inputs have a dominant effect on the UAVP dynamics observed here. Also the simulated response fit on ϕ_L is 28.9% while for θ_L it is -12.3%.

7-3-3 Completed Model and Outlook

Having identified the Quadrotor Controller Model, summarising, the derived model's state $\boldsymbol{x}_c$ combining the states of the identified second-order state space systems given by Eqs. 7-2, 7-3 and 7-5 is formally defined as,

$$\boldsymbol{x}_c = [\boldsymbol{x}_\theta, \boldsymbol{x}_\phi, \boldsymbol{x}_{\ddot{z}}]^\top \tag{7-6}$$

Forward-looking, the model disparity to the real response for different UAV inputs should be considered when using it for predicting the future UAVP motion in MPC. With lower magnitude, gradual inputs the model is less accurate so there is more output uncertainty, while for larger inputs, the model is more accurate with less output uncertainty. The MPC controller can be made more robust by dynamically adjusting for the modelling uncertainty depending on whether the inputs result in the UAV's gradual or aggressive motions. It should also be noted that unmodelled dynamics have an affect on the UAVP response, for example, the rotor wake usually causes the payload to swing even when the UAV is hovering. This effect is difficult to isolate and avoid as the payload is always suspended under the UAV.

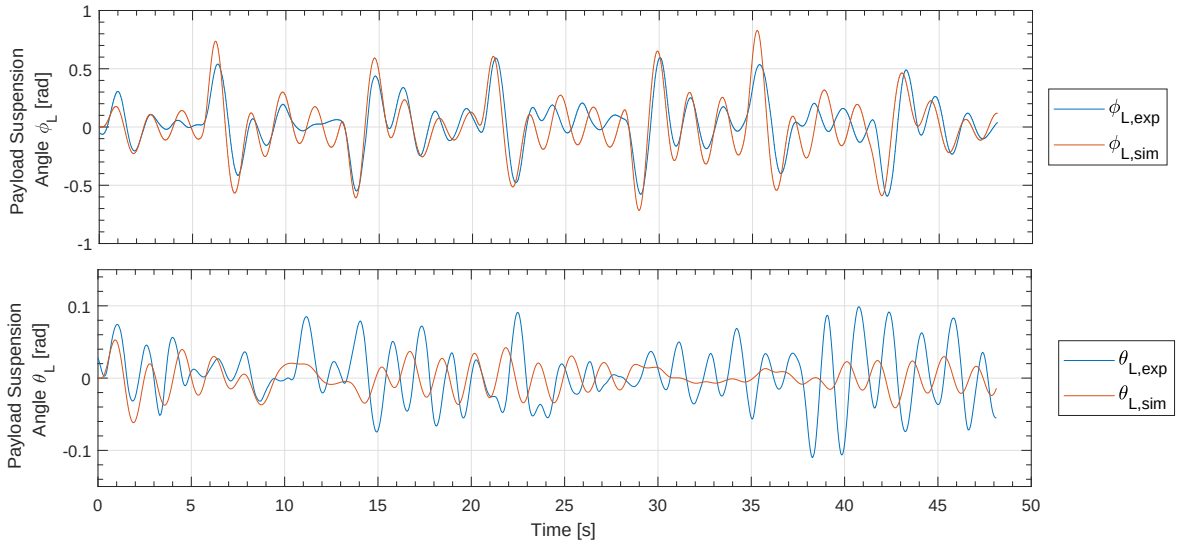


Figure 7-8: Payload suspension angles in experiment and simulation for dominantly pitch UAV inputs, simulated ϕ_L fit of 28.9%, θ_L fit of -12.3%

Chapter 8

Future Research Plan

As mentioned in Chapter 3, the identified, verified and validated UAVP model resulting from this preliminary study will be used to perform MPC control on the UAV. Having addressed research questions 1a and 1e as shown in Section 3-2, the remainder of this thesis will be concerned with the MPC control algorithm implementation. Section 8-1 outlines the necessary steps to be taken for the controller design and implementation in experiments. Section 8-2 then outlines the future extension with learning capabilities for controller adaptability.

8-1 Controller Design and Implementation

The implementation of MPC for closed-loop UAVP system control requires a definition of the MPC control properties, a verification of the expected collision-avoidance behaviour and finally validation of the results. These are the steps required to address the remaining research questions 1b, 1c and 1d from RQ1. The steps to be undertaken to achieve the research objective include the following with the estimated time for completion indicated;

1. Implementation of identified UAVP system model from preliminary study within MPC framework for simulation (1 month)
   (a) Definition of MPC objective and constraints to achieve closed-loop collision avoidance flight with dynamic obstacles in simulated and experimental environments
   (b) Develop simulation environment where MPC controller based on *FORCES PRO* solver can be utilised to generate control commands
   (c) Achieve capability to simulate the UAVP system under the derived dynamics, defined objective and constraints for multiple test case scenarios
2. Verification of the simulated MPC controlled UAVP response (1 month)
   (a) Use scenario test cases to demonstrate closed-loop collision avoidance in simulation
   (b) Evaluate run-time performance of controller to determine applicability on physical setup

   (c) Tune controller to achieve desired objective while performing under future required run-time considerations for physical experiments

   (d) Identify potential merits and limitations of implemented system to show the scenarios in which the MPC controller does (not) offer improved performance over current control techniques of UAVP systems in obstacle rich dynamic environments

3. Implementation of MPC controller in experimental setup (1 month)

   (a) Programming intensive phase involving the deployment of the controller within the hardware control framework as introduced in this preliminary study

   (b) Addressing issues arising from usage of real world data including noise, uncertainties and state estimation using, for example, Kalman Filtering

4. Testing and validation of MPC scheme in experimental setup to achieve real-time closed-loop collision-free flight (1-2 month)

   (a) Demonstrate that the simulated behaviour translates to the real-world under experimental conditions using the scenarios used during simulation

   (b) Identify limitations or performance affecting factors that are attributed to using the MPC controller in real-world scenarios

   (c) Evaluate the merits and limitations of the system demonstrating how the obtained control scheme compares to current UAVP control implementation in obstacle rich dynamic environments

The first and primary contribution of the research will be the demonstration of real-time closed-loop collision-free flight in dynamic uncertain environments which has not been addressed in research. As presented in the literature study in Chapter 4, collision-free UAVP flight has only been performed using off-line generated collision-free trajectories that are precisely tracked by the UAV controller. Demonstrating a real-time online generator enables the possibility for the UAVP system to account for unforeseen changes in the environment in real-time. This is important for dynamic and uncertain real world applications where it is intractable and impractical to pre-compute trajectories for extended periods of time and distance. As the desired implementation still relies on knowing the obstacle positions, there is a need for future research to look into combining the obstacle detection methods with the controller to be designed in this research.

8-2 Learning Extension

The second contribution that also addresses a practical aspect of UAVP system is the LBSI for model uncertainties that could include, amongst others, adaptation to unknown or roughly estimated payload mass and suspension cable length. This contribution is incremental in nature and only makes the obtained MPC framework robust to additional modelling uncertainties that are not already addressed by the model derived in this preliminary study.

As the primary contribution of this research project already requires a significant amount of time, the extension of using Learning Based System Identification (LBSI) for improving the

performance is conditional to the time planning. Planning of this stage will be performed should all steps from Section 8-1 be completed in time and a significant need for the learning extension is recognised. The research questions from RQ2 provides guidance on defining these steps. The possible avenues for improving the system dynamics modelling using learning is addressed in the Literature Survey presented in Appendix 4.

Chapter 9

Preliminary Conclusion

The preliminary study as presented in this report has outlined the derivation of the UAV-Payload (UAVP) system dynamics model followed up with a thorough simulation and experimental based verification and validation process. Chapter 3 presented an introduction to the preliminary study and its relevance in the scope of this research project. It was shown that understanding the system dynamics is imperative to be able to design a model based controller which is the topic of subsequent studies during this research. Chapter 4 gave an overview of relevant literature regarding UAVP modelling from which the model used for this preliminary study was derived. The remaining chapters presented outcomes of the preliminary study.

In Chapter 5, the full UAVP kinematics and dynamics were derived from first-principles using the Euler-Lagrange formulation for derivation of the Equations of Motion (EOMs). Building upon models presented in literature, the aerodynamically disturbed UAVP dynamics with a rigid suspension cable was derived. Chapter 6 outlined the research methodology with extensive detail of the MATLAB simulation environmental and experimental setup to enable duplicability of the results obtained. Inverse kinematic equations to obtain the system state from raw Motion Capturing Systems (MCS) were presented to facilitate comparison of simulation and experimental results. A detailed account of the data collection and post-processing procedures is provided as experimental results are extensively used throughout the preliminary study and such procedures are necessary for collecting valid, systematic error-free data.

Following the groundwork presented in preceding chapters, a discussion of the main results was presented in Chapter 9. System identification using experimentally obtained data was performed to identify the pitch, roll and vertical velocity dynamics of the Parrot Bebop 2 quadrotor. It was determined that three second-order state-space models, one for the pitch, roll and vertical velocity dynamics, was able to capture the quadrotor's real dynamics adequately. The models had a model fit of 94.59%, 91.09% and 91.38% (NRMSE) on the estimation dataset respectively and were of a decent quality according to the residual analysis. The models were determined to be stable with relatively high damping ratios in the range of 0.75-0.90 explaining the limited oscillatory response behaviour. As observed from experimental data,

the models generally converge to a steady-state value less than the commanded input value. The experimental response data also shows how the quadrotor does not consistently reach a particular steady-state value over repetitive commanded inputs so the internal measured angles likely suffer from an error range. Therefore, the overall gain of the response was less than 1.

The identified UAV dynamics completed the UAVP model after which a comparison of the simulated and experimental responses was performed. Short single step input runs as well as longer runs with combined inputs were performed to perform a comparison. For short step inputs of one second, the simulated and experimental suspended payload frequency and amplitude response was equivalent with an overall good fit of greater than 70% when the identified dynamics are included. Excluding the identified UAV dynamics showed a considerably degraded performance with a fit in the range of $20 - 30\%$. Undoubtedly there were discrepancies between simulation and experimental response data due to unmodelled dynamics and imperfectly estimated effects such as drag. During the longer experimental runs lasting around 50 seconds, it was observed that the simulated and experimental dynamics agreed the most when significant energy was introduced in a certain swing direction. For example, if the UAV pitched forward, the simulated and experimental ϕ_L angle would be equivalent. For swing directions where there was very minimal or no input energy introduced, the swing angle would remain relatively low (< 0.1rad or $< 6°$) and the model and observed behaviour diverge. This is expected as the residual and/or low amplitude oscillations are not driven by the main UAVP dynamics, but additional phenomenon not implemented in the currently derived UAVP model. An important effect is that of the rotor wake on the payload directly in it. Therefore, the current UAVP model works well when the UAV motion has a dominant effect on the payload dynamics such that significant swinging motion is observed. When the UAV is just hovering (no inputs) or moving very slowly, the payload motion is driven by unaccounted effects.

Finally, an analysis of the cable taut and slack state was performed on experimental runs involving gradual and aggressive inputs to identify whether switching UAVP dynamics was a prevalent phenomenon. The results showed a minimal change in the suspension cable length pointing to no switching in the cable state, however, it could not conclusively be said whether the cable never becomes slack due to lack of accurate cable measurements. Future studies can potentially measure the cable tensile force which will clearly indicate the slack state when the tensile force becomes zero.

The preliminary study highlights the merits and limitations of the derived UAVP model enabling its future implementation in the model based MPC. Given the scope of this research study, limitations which hinder the use of the model for the remaining project have been addressed. Limitations requiring further investigation have been noted and could be potentially addressed in future in-depth studies where it may be necessary to have more accurate models.

Part III

Supplementary Material to the Scientific Article

Chapter 10

Configuring the Non-Linear Model Predictive Controller

As a supplement to the paper, this chapter provides additional details regarding the Non-Linear Model Predictive Control (NMPC) algorithm. Section 10-1 provides an explanation of the NMPC concept with details regarding stability and optimality. Section 10-2 explains the rationale behind the chosen NMPC cost weightings.

10-1 NMPC Algorithm for Closed-Loop, Collision-Free Trajectory Generation

NMPC is a multi-stage optimal control strategy where control sequences are generated that respect a user-defined control objective function and possible constraints. In the Literature Survey of the preliminary study, Chapter 4, the theoretical foundations of MPC and the state-of-the-art applications pertaining to UAVP system has been extensively discussed. Also, the merits and limitations to other classical and optimisation based techniques including PID, LQR and iLQR were presented. Recapitulating, some major advantages of Non-Linear MPC specifically is the ability to use non-linear system dynamics descriptions and provide guarantees through constraint handling (Mayne, Rawlings, Rao, & Scokaert, 2000). In Appendix D, the algorithm is described in detail.

10-1-1 Costs and Constraints

The optimisation problem associated costs and constraints and their relevance are discussed.

Costs

Cost terms of the NMPC objective function are used to promote or discourage certain qualities of the generated trajectories. The main objectives of the planning problem addressed in this

study are;

- Perform point-to-point navigation from a start to goal position in three dimensional space
- Agility of the system's response in terms of payload swing and the commanded inputs
- Collision-free guarantees for the trajectories with respect to all obstacles
- Safe navigation amongst dynamic obstacles that includes humans

To achieve the objectives, relevant costs terms have been introduced and described in the scientific article.

Equality Constraints for System Dynamics

The NMPC generated trajectory is feasible with respect to the system dynamics through enforcement of the discretised state transition as an equality constraint $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$. The EOMs are non-linear ordinary differential equations, as presented in the preliminary study, Chapter 5, and are continuous-time functions. For implementation in NMPC which is a discrete-time controller, the EOMs are discretised using the Runge-Kutta approach such that a discrete time state transition function is derived. Requiring real-time performance and online implementation of the controller, only explicit discretisation was considered and after some mostly qualitative data analysis, the 2^{nd} order Runge-Kutta method was determined to provide a balance of speed, accuracy and stability. The simple explicit Euler method, though very fast, quickly becomes unstable for non-linear systems when the real sample time is relatively large (Burden & Faires, 2011). Higher order Runge-Kutta methods are slower but more stable, especially if the dynamics are highly non-linear, however, under experimental conditions that were performed, no loss of stability was qualitatively observed for the 2^{nd} order formulation. The benefit of the lower order is the higher control frequency.

A future comparative study of different discretisation methods on the real-time performance of NMPC would help quantitatively evaluate the discretisation's effects on control performance. This is currently beyond the scope of this research.

Inequality Constraints

Additional (inequality) constraints are defined to ensure practical feasibility of the obtained solution. They are also used to guarantee collision-free trajectories and compliance to physical workspace limits. For real-world deployability, it is generally favourable to allow minor violations of some uncritical constraints as to maintain run-time feasibility of the optimisation problem. This is achieved by *softening* the hard constraint whereby slack variables $\boldsymbol{s}$ are introduced to the (inequality) constraint that 'capture' the minor violations while keeping the (inequality) constraint valid (Mayne et al., 2000). For example, given a function $g(x) \geq 0$, the slack variable is introduced as,

$$g(x) + s \geq 0$$

In this case if $g(x) = -0.1$, then given $s \geq 0.1$ the inequality will remain valid. To make sure slacks are not abused by excessive use, a very high slack variable associated cost is included in the objective function. The cost discourages the optimiser from using slacks to

achieve constraint satisfaction. Under this cost, the soft-constraint becomes similar to hard-constraints while maintaining run-time feasibility (Zheng & Morari, 1995). Slack variables must be non-negative by definition as allowing full range of values would make the constraints redundant.

10-1-2 Optimality and Stability

The algorithm that solves the NMPC optimisation problem is discussed as well as a note on the solution optimality and the controller's closed-loop stability.

Real-Time Convex Optimisation Algorithms

The NMPC non-convex planning optimisation problem is solved using a constrained, convex optimiser. Traditionally, MPC type control was reserved for slowly evolving processes due to the relatively high computational load (Mayne et al., 2000), however, advances in the realm of hardware and software have greatly improved the controller's run-time frequency. In this study, an efficient state-of-the-art non-linear programming based solver FORCES Pro is used that is able to deliver real-time computations for optimisation problems (Zanelli, Domahidi, Jerez, & Morari, 2017). The solver uses a *Primal-Dual Interior Point Constrained Optimiser* to solve the optimisation problem at high frequencies of order 10^1 to 10^2 Hz depending on the problem complexity. In Appendix J the optimisation algorithm is fully outlined providing the reader a basic introduction to the optimiser used in this study.

Optimality of the Obtained Solution

Using convex programming on a non-convex problem means that the solution obtained is only locally optimal. Realising that NMPC uses a finite-horizon, hence only local planning is performed, there is a significant risk that the optimised trajectory will converge to a local optimum. Convergence to a local optimum may result in a deadlock situation where the system gets 'stuck' or trapped during its manoeuvre from the start to goal position. This is generally less of an issue for global trajectory planners as the optimisation algorithm is able to probe the entire optimisation space resulting in a global optimum as solution. To mitigate the effects of local optima convergence, it is necessary to carefully design the objective function and constraints. Additionally, as in reinforcement learning an exploration type step can be built in to allow temporary non-locally optimum solutions such that terminally a more favourable (global) optimum is reached (Sutton & Barto, 1998). This exploratory type behaviour is recommended for a future study.

Closed-Loop Stability of the NMPC Controller

NMPC is a finite-horizon controller that predicts the system's trajectory in open-loop and achieves closed-loop performance through the receding-horizon principle. However, the closed-loop performance is not necessarily stable and depends on the chosen horizon N and cost tuning (Mayne et al., 2000); it is possible that a tuned controller works in one situation but fails in others. (Re-)tuning the controller for every minor system change can be tedious,

therefore, approaches for closed-loop stability have been developed; methods include, amongst others, a terminal (in)equality constraint or cost, or both and the survey (Mayne et al., 2000) provides an excellent overview on stability issues surrounding MPC control.

10-1-3 Extra: Additional NMPC Stability Results

Extending the results in the Scientific Article, Appendix K provides an overview of the effects of control time-step size and lags on NMPC's closed-loop stability and performance.

10-2 Controller Implementation and Cost Weight Tuning

The primary approach to weight tuning was trialling different values in simulation, observing the system response and verifying whether the planning objectives as mentioned in Section 10-1-1 were fulfilled. The tuned weights for the costs introduced in the scientific article are presented in Table 10-1. A brief rationale is provided explaining the reasoning behind the chosen cost weighting.

Table 10-1: Tuned NMPC objective function cost associated weights as implemented for simulation and experimental studies

Cost Term	Value	Description
Navigation (Terminal)	$w_{\text{nav}} = 1.0$	High weight to guide towards, and stabilise the system at the goal position.
Obstacle Separation	$w_{\text{pf}} = 1.2$	High cost guides generated trajectories away from the obstacle for safer planning.
Inputs	$w_{\text{in}} = 0.01$	Low cost to only use large inputs when necessary without compromising on system agility.
Payload Swing	$w_{\text{swing}} = 0.001$	Very low cost promotes swing minimisation only when the system reaches the goal position. At the goal position, the swing associated cost is the only non-zero cost (assuming no obstacle is close by). As the cost is low, system agility is not compromised.
Assistive Steering	$w_{\text{steer}} = 0.05$	Steering for low-horizon planning is promoted to be assistive rather than driving the trajectory generation. By keeping a low weight, the navigation and collision avoidance costs are given more importance. Steering is beneficial when the MPC uses a low horizon length such that the system has short foresight; including steering guides the system to obstacle free zones improving collision avoidance performance.
Each slack variable	$w_{\text{slack}} = 1 \times 10^5$	Ensures the slack variables are only used to maintain MPC run-time feasibility without making the constraints fully redundant.

Chapter 11

Online State Estimation and Filtering using Cascaded Kalman Filters

As a supplement to the paper, the rationale behind developing the Cascaded Kalman Filter is discussed with the theoretical background and details regarding the filter tuning. Section 11-1 explains the theoretical methodology of the chosen state estimation approach. Section 11-2 describes the implementation and tuning of the Cascaded Kalman Filter setup. Finally, Section 11-3 presents an evaluation of the state estimator's performance. Note that some notation from this report may differ from the scientific article.

11-1 State Estimation Scheme

Referencing back to the control system design presented in Section 6-1-2, the system is divided into three subsystems (excluding the state estimator) namely 1) the high-level NMPC controller, 2) the Quadrotor (Input) Controller and 3) the UAVP system. For NMPC to predict the next system states, it must initialise the problem based on the current estimated state $\boldsymbol{x}_0$ which includes the quadrotor controller model and UAVP model states. As the quadrotor controller is specific to the Parrot Bebop 2 while the UAVP model is general, with the intention of keeping the state estimation scheme as modular as possible, the estimator for both subsystems is separated in a cascaded setup.

Two estimators are combined to construct the cascaded estimator;

1. The quadrotor controller estimator uses the linear model description of the Parrot Bebop 2 dynamics to estimate the state $\boldsymbol{x}_c = [\boldsymbol{x}_{\boldsymbol{\theta}},\ \boldsymbol{x}_{\boldsymbol{\phi}},\ \boldsymbol{x}_{\ddot{z}}]$ (See Section 7-3-3 for the state definition; in the paper $\boldsymbol{x}_F \equiv m\boldsymbol{x}_{\ddot{z}}$)

2. The UAVP system estimator uses the non-linear model description of the UAVP system to estimate the state $\boldsymbol{x}_q = [\boldsymbol{q},\ \dot{\boldsymbol{q}}]$ (See Section 5-1 for the state definition)

Each estimator is treated individually in the subsequent sections after which the combined estimator is discussed. This division of the estimation algorithm is possible as the dynamics associated to the $\boldsymbol{x}_c$ and $\boldsymbol{x}_q$ states are assumed to be decoupled as discussed in Section 6-1-2. The Kalman Filter (KF) formulation is used for both estimators. Other methods and their performance comparison to the Kalman filtering approach are beyond the study's scope and may be explored in future studies.

11-1-1 Linear Kalman Filter

The Linear KF is used to estimate state $\boldsymbol{x}_c$ for the quadrotor input controller model which is defined in terms of three linear second-order state-space systems by Eqs. 7-2, 7-3 and 7-5. Using the state estimates, the model's filtered outputs are computed. As previously introduced, the output of the linear models give the quadrotor's real pitch, roll angles and vertical control acceleration input.

The continuous time dynamics are discretised using zero-order hold with the time step Δt. With abuse of notation, the standard discrete-time linear system description for use with a Linear KF is given by Eq. 11-1 (Kalman, 1960).

$$\begin{aligned} \boldsymbol{x_k} &= F_k \boldsymbol{x_{k-1}} + B_k \boldsymbol{u_k} + \boldsymbol{w_k} \\ \boldsymbol{z_k} &= H_k \boldsymbol{x_k} + \boldsymbol{v_k} \end{aligned} \tag{11-1}$$

With time index k, state $\boldsymbol{x_k}$, input $\boldsymbol{u_k}$, measurement $\boldsymbol{z_k}$, discrete state matrix F_k, discrete input matrix B_k, discrete observation matrix H_k and process and observation noise random variables $\boldsymbol{w_k} \sim \mathcal{N}(0, Q_k)$ and $\boldsymbol{v_k} \sim \mathcal{N}(0, R_k)$ with covariances Q_k and R_k respectively.

11-1-2 Unscented Kalman Filter

The Unscented KF is used to estimate the state $\boldsymbol{x}_q$ with variables $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ using the non-linear dynamics given by the EOMs in Equation 5-24. The transformation from the quadrotor pitch, roll and vertical control acceleration to $\boldsymbol{f}$, given by Eq. 6-10, is included in the non-linear dynamics model.

As KFs work on discrete-time models, the UAVP dynamic model is discretised using the explicit Euler method as given by Eq. 6-11. With abuse of notation, the standard discrete-time non-linear system description for use with a Non-Linear KF is given by Eq. 11-2 (Wan & Van Der Merwe, 2000).

$$\begin{aligned} \boldsymbol{x_k} &= F_k(\boldsymbol{x_{k-1}}, \boldsymbol{u_k}) + \boldsymbol{w_k} \\ \boldsymbol{z_k} &= H_k(\boldsymbol{x_k}) + \boldsymbol{v_k} \end{aligned} \tag{11-2}$$

With time index k, state $\boldsymbol{x_k}$, input $\boldsymbol{u_k}$, measurement $\boldsymbol{z_k}$, discrete non-linear state transition function F_k, discrete non-linear observation function H_k and process and observation noise random variables $\boldsymbol{w_k} \sim \mathcal{N}(0, Q_k)$ and $\boldsymbol{v_k} \sim \mathcal{N}(0, R_k)$ with covariances Q_k and R_k respectively.

Rationale for choosing the Unscented Kalman Filter

The Unscented KF, as introduced in (Julier & Uhlmann, 1997), is chosen over the more traditional Extended KF for non-linear systems due to their same order of computational complexity with Unscented KF's benefit and ability to handle non-linearities in dynamics directly (Wan & Van Der Merwe, 2000). The Extended KF is a familiar non-linear state estimator that shares many similarities with the standard linear KF. In Extended KF, the non-linear state and observation equations are linearised at run-time by computing their associated Jacobians so that the state and observation transition function in discrete form can be obtained. The system is usually linearised around a nominal value of the state and input after which the state prediction is propagated giving a new state distribution. However, for highly non-linear systems using the linearised dynamics for state propagation can result in large errors in the estimated state distribution (Wan & Van Der Merwe, 2000). Additionally, the computations of the Jacobians is usually very expensive and not always possible if derivatives of the state and observation function are not available and/or computable. The Unscented KF is able to address these shortcomings by using a sampling based approach to propagating the state distribution as will be explained in the subsequent section. In (Bisgaard, 2008) an Unscented KF was also used for state estimation in the case of a traditional helicopter with a slung load system; similarly, the Unscented KF is used for this study.

The Unscented Kalman Filter Algorithm

A qualitative overview of the Unscented KF method is presented; for a comprehensive explanation of Unscented Kalman Filtering, the equations and algorithm please refer to (Julier & Uhlmann, 1997; Wan & Van Der Merwe, 2000). Figure 11-1 is included to visualise the algorithm's process supporting the explanation provided. Given the current true state distribution, shown by the blue dots in the top-left of Figure 11-1, there is an associated true mean and covariance. The Unscented KF algorithm captures this distribution by carefully selecting sigma points $\mathcal{X}$ that are representative of the distribution; this process is known as the *unscented transform* of the distribution. For a system of n state variables, there are $2n+1$ sigma points of which one is always at the mean state value. For the real system following application of a non-linear function $f(\boldsymbol{x})$ the resulting true state distribution has a new true mean and covariance. The Unscented KF algorithm applies the same non-linear function to the sigma points such that,

$$\boldsymbol{\mathcal{Y}} = f(\boldsymbol{\mathcal{X}})$$

The transformed sigma points $\boldsymbol{\mathcal{Y}}$ are then used to reconstruct the new predicted sample mean and covariance. Measurement data is subsequently used to further refine the prediction to a state estimate by computing a Kalman gain based on the prediction and measurement statistics. This is analogous to a standard Kalman filter measurement update. The result of this process is a state estimation with an associated distribution encoded in the covariance. See (Wan & Van Der Merwe, 2000) for the full algorithm.

11-1-3 Cascaded Kalman Filter and Data Measurement

The Linear and Unscented KF are combined in a *Cascaded KF* setup as schematised in Figure 11-2. The filtered states of the Linear KF are used to compute the true pitch, roll and

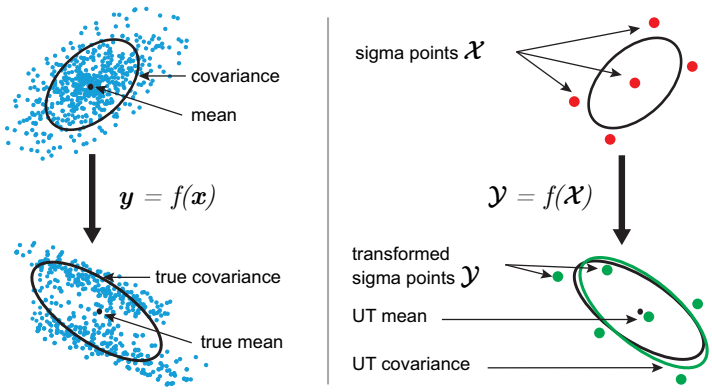


Figure 11-1: The unscented transform and propagation of a state distribution, shown by blue dots, for the Unscented KF algorithm

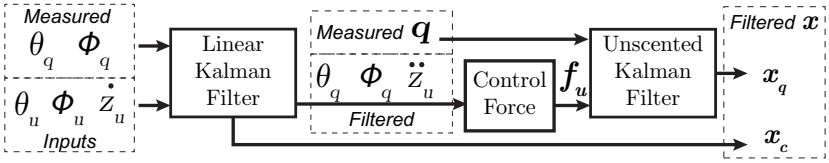


Figure 11-2: Cascaded Kalman Filtering setup for state estimation and filtering of the quadrotor controller model and UAVP system model states

vertical control acceleration. Then by Eq. 6-10, the control input forces $\boldsymbol{f}_u$ are computed as inputs for the UAVP model.

The Parrot Bebop 2 quadrotor that is used for experiments only transmits internal sensor data to external clients in the order of 10^0 Hz [1], this is too slow for state estimation of the highly dynamic UAVP system. As the designed Cascaded KF estimators runs off-board, it was ineffective to use any transmitted on-board sensor data for state estimation. Therefore, to perform the state estimation, measurement data was collected exclusively from the external Motion Capturing System (MCS); the MCS obtainable measurement data is given in Table 11-1.

As the high-fidelity MCS provides sub-millimetre accurate measurements of the quadrotor and payload's pose in $SE(3)$[2], the system configuration $\boldsymbol{q} = [x_q, y_q, z_q, \theta_l, \phi_l]^\top$ can readily be measured very accurately with low noise levels (See Appendix E showing the MCS calibration results). The time derivative of $\boldsymbol{q}$, namely $\dot{\boldsymbol{q}}$ is not directly measurable as the MCS cannot provide this type of data, therefore, the KF is primarily designed to produce low noise estimates of this.

Also note that within the quadrotor controller model there is the vertical control acceleration output, however, there are no measurements available for this. Therefore, for the vertical velocity to vertical control acceleration model state estimation, the Linear KF only performs the model prediction skipping the measurement update. As the study predominantly involved

[1] Mani Monajjemi. "bebop_autonomy". Accessed February 01, 2018. `http://bebop-autonomy.readthedocs.io/en/latest/reading`

[2] Natural Point. "OptiTrack Motive". Accessed January 18, 2018. `http://optitrack.com/products/motive/`

Table 11-1: Measurable system variables from Motion Capturing System data for use in state estimation scheme

Measurement	Notation	Notes
Quadrotor Pitch	θ_q	Directly obtained from MCS
Quadrotor Roll	ϕ_q	Directly obtained from MCS
Quadrotor Position	x_q, y_q, z_q	Directly obtained from MCS
Payload Suspension Angles	θ_l, ϕ_l	Computed with Equations 6-15 and 6-16 using MCS positional data of the quadrotor and payload

navigation in the horizontal plane, using only the state prediction did not result in any qualitatively observable detrimental effects.

11-2 Kalman Filter Implementation and Tuning

The observability of the system models were verified to ensure the KFs would converge to an estimation when implemented. Additionally, the filters have been tuned to produce system state estimates provided the noise levels in the MCS data available (See Appendix E showing the MCS calibration results).

11-2-1 System Model Observability

Using the methods outlined in (Hermann & Krener, 1977), observability of the quadrotor controller model and UAVP system model are verified to satisfy the necessary criteria for having a convergent Kalman Filter. Only an overview of the procedure and results is provided; the state observability was verified using the Symbolic toolbox of MATLAB.

Quadrotor Controller Model and Linear Observability

The observability rank criterion based on the linear observability matrix, as defined by Eq. 11-3, is used to verify the quadrotor controller model observability (Olsder et al., 2011).

$$O_{\text{linear}} = \left[C, CA, CA^2, \ldots, CA^{n-1}\right]^\top \tag{11-3}$$

The criterion states that given a system with n states, the rank of $O_{\text{linear}} = n$ for the system to be fully observable. The variable A and C are the discrete-time model state and output matrix, respectively.

Combining the quadrotor controller linear systems given by Eqs. 7-2, 7-3 and 7-5 into one linear state-space formulation, the resulting state state is $\boldsymbol{x}_c \in \mathbb{R}^6$, and A, C matrices are

given below and are zero everywhere except where indicated.

$$A_{(6\times6),\text{continuous}} = \begin{bmatrix} -4.301 & -2.877 & & & \cdots & 0 \\ 10.92 & -10.37 & & & \ddots & \vdots \\ & & -2.789 & -4.978 & & \\ & & 9.302 & -13.72 & & \\ \vdots & \ddots & & & -6.767 & -6.546 \\ 0 & \cdots & & & 3.031 & 0.311 \end{bmatrix}$$

$$C_{(3\times6),\text{discrete}} = \begin{bmatrix} 1.763 & 4.586\times10^{-2} & & & \cdots & 0 \\ \vdots & \ddots & 1.996 & 0.4657 & \ddots & \vdots \\ 0 & \cdots & & & -0.620 & 4.070\times10^{-2} \end{bmatrix}$$

For digital implementation, the matrices are discretised with $\Delta t = 0.05$ s (experimentally measured average control period). The discrete form of A is given by Eq. 11-4 using the zero-order hold method. The continuous and discrete form of C is equivalent.

$$A_{(6\times6),\text{discrete}} = e^{A_{(6\times6),\text{continuous}}\Delta t} \tag{11-4}$$

The resulting discretised matrices A and C are time and state invariant.

Substituting the discretised A and C in Eq. 11-3 with $n = 6$, then $\text{rank}(O_{\text{linear}}) = 6$ so the system is fully observable.

UAVP System Dynamics and Non-Linear Observability

The observability rank criterion based on the system Lie derivatives is used to verify the non-linear system observability. For the Unscented KF, the system state is $\boldsymbol{x}_q \in \mathbb{R}^{10}$ so the numbers of states $n = 10$, then when the rank of the observability matrix given as $\text{rank}(O_{\text{non-linear}}) = 10$ the system is observable (Hermann & Krener, 1977). For more details about definitions of non-linear observability refer to (Hermann & Krener, 1977). The state transition function is $f(\boldsymbol{x}_q, \boldsymbol{u}) = [\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}]^\top$, where $\ddot{\boldsymbol{q}}$ is defined by Eq. 5-24. The estimator's observation function $h(\boldsymbol{x}_q) = \text{diag}(1,1,1,1,1,0,0,0,0,0)\boldsymbol{x}_q \equiv \boldsymbol{q}$.

The observability matrix definition requires the use of Lie derivatives based on the method presented in (Hermann & Krener, 1977). The Lie derivative of the observation function h with respect to the state dynamics f and its subsequent derivatives is given by,

$$L_f h = \partial_x h \cdot f \tag{11-5}$$

$$L_f L_f h = \partial_x (L_f h) \cdot f \tag{11-6}$$

The non-linear observability matrix is defined by,

$$O_{\text{non-linear}} = \begin{bmatrix} \partial_x h, & \partial_x (L_f h), & \partial_x (L_f L_f h), & \ldots, & \partial_x (\underbrace{L_f \ldots L_f}_{n-1} h) \end{bmatrix}^\top \tag{11-7}$$

Replacing f and h by the definitions presented in the preceding text, for the UAVP non-linear model $\text{rank}(O_{\text{non-linear}}) = 10$ hence the system is observable.

Table 11-2: Linear and Unscented Kalman Filter noise covariance matrices as implemented in controller framework

Kalman Filter	Tuned Process and Observation Noise Covariance Matrices
Linear	$Q_{\text{LKF}} = \text{diag}\left(\underbrace{0.01, 0.01}_{\theta_q}, \underbrace{0.01, 0.01}_{\phi_q}, \underbrace{0, 0}_{\ddot{z}_u}\right)$
	$R_{\text{LKF}} = \text{diag}\left(\underbrace{0.01}_{\theta_q}, \underbrace{0.01}_{\phi_q}\right)$
Unscented	$Q_{\text{UKF}} = \text{diag}\left(\underbrace{0.1, 0.1, 1, 0.1, 0.1}_{\boldsymbol{q}}, \underbrace{10, 10, 1, 0.1, 0.1}_{\dot{\boldsymbol{q}}}\right)$
	$R_{\text{UKF}} = \text{diag}\left(\underbrace{1 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-4}}_{\boldsymbol{q}}\right)$

11-2-2 Kalman Filters Tuning

The definition and tuning of the KFs is performed using the noise statistics of the MCS provided data and user-defined values.

Linear Kalman Filter

The tuned process and observation noise covariances are given in Table 11-2. The Q_{LKF} and R_{LKF} matrix entries are chosen such that the Linear KF equally relies on the model prediction and measurement data to produce the best estimate. As discussed in Chapter 7, the linear second-order model fits are very good (See Appendix H for model fit) so the KF can use the prediction for aggressively removing noise from the measured data. As the high-fidelity MCS provides very accurate measures of θ_q and ϕ_q, their associated covariances in R_{LKF} are relatively low. The Q_{LKF} entries of θ_q and ϕ_q are not set to higher values as this lets more noise pass through to the output of the KF. As mentioned, there are no measures of $\ddot{z}_u$ and only the prediction step of the Linear KF is performed to obtain the associated state estimates.

Unscented Kalman Filter

Tuning of the Unscented KF was performed to obtain accurate estimates of $\dot{\boldsymbol{q}}$ using measurements of $\boldsymbol{q}$ without excessive noise amplification. The tuned process and observation noise covariances are given in Table 11-2 with the variables to which the covariance are associated.

The Q_{UKF} and R_{UKF} matrix entries are chosen such that the Unscented KF relies more on the high-fidelity MCS measurements to produce good estimates of $\dot{\boldsymbol{q}}$. As the full $\boldsymbol{q}$ vector is directly measured, all associated entries in R_{UKF} have very low covariances. The $\boldsymbol{q}$ related entries in Q_{UKF} have relatively low covariances to still allow the predicted $\boldsymbol{q}$ values to be used for noise reduction in the state estimate. For entries in Q_{UKF} associated to $\dot{x}_q$, $\dot{y}_q$ and $\dot{z}_q$

(first three elements of $\dot{\boldsymbol{q}}$), the covariance is high as the model predicted values are likely to be less accurate due to unaccounted dynamics. For the payload suspension angular rates $\dot{\theta}_l$ and $\dot{\phi}_l$, the Q_{UKF} covariance is relatively low as there is no direct measure of the angular rates and it needs to be predicted using the UAVP model. The reliance on the process model to obtain $\dot{\boldsymbol{q}}$ means noise will inevitably be amplified when using only the available $\boldsymbol{q}$ measures. However, the Unscented KF filtering reduces the noise amplification to acceptable values as will be shown in the subsequent section.

11-3 Evaluation of State Estimation Performance

The state estimator was evaluated in a simulated setup with artificial noise introduced to the process model and measurements. Experimental evaluation of the state estimator is beyond the study's scope and is recommended for future studies.

The standard deviation of the MCS based measurements is chosen to be approximately 3×10^{-3} m and 3×10^{-3} degrees. As no process noise statistical information was directly available, a value of 0.01 for the standard deviation was used to approximate some process noise. A comprehensive study of the process and its noise statistical properties is outside the scope of this study and is recommended for future studies; this may improve the estimation performance of the designed KFs.

The UAVP system response was simulated for 15 s with a simulation step size of $\Delta t = 0.05$ s. The UAVP system was initialised with the quadrotor positioned at $(3,\ -3,\ 2)\ m$ and the suspensions angles $\theta_l = 20°$ and $\phi_l = 20°$. The Linear and Unscented KF state variables were each initialised from the zero-mean Gaussian distribution $\mathcal{N}(0, 0.2)$. The input signals are given below with time t

$$\boldsymbol{u}(t) = \begin{bmatrix} \theta_u \\ \phi_u \\ \dot{z}_u \end{bmatrix} = \begin{bmatrix} 0.175 \sin(2\pi \cdot 0.2 \cdot t) \\ 0.175 \cos(2\pi \cdot 0.2 \cdot t) \\ 0.5 \sin(2\pi \cdot 0.2 \cdot t) \end{bmatrix}$$

The inputs θ_u and ϕ_u are 0.2 Hz sinusoidal signals with an amplitude of 0.175 rad ($\approx 10°$). The input $\dot{z}_u$ is a 0.2 Hz sinusoidal signal with an amplitude of 0.5 m/s.

The resulting trajectory of the UAVP system is shown in Figure 11-3.

The estimator's performance is evaluated using the Root Mean Squared Error (RMSE) and Normalised Root Mean Squared Error (NRSME) of the true to estimated/filtered KF output as presented in Table 11-3. Within first 20 data points (1 second) the KFs estimates are still converging to the true signal, so to not affect the fit statistics they have been omitted in the fit calculation. The NRMSE facilitates the comparison of the estimator's performance on the different scales of the data produced; the MATLAB definition of NRSME is used for normalisation [3]. The subsequent sections discuss the performance of the Cascaded KF.

[3] MATLAB. "goodnessOfFit". Accessed 10 January, 2018. `https://nl.mathworks.com/help/ident/ref/goodnessoffit.html`

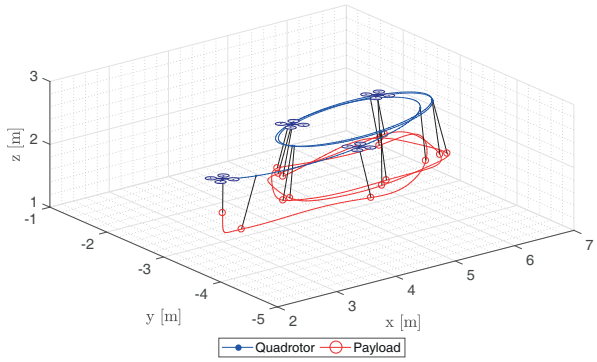


Figure 11-3: Simulated 20 s UAVP flight trajectory of initially non-equillibrium system with sinusoidal inputs $\boldsymbol{u}$

Table 11-3: Fit as (N)RMSE for estimated variables is using Linear and Unscented Kalman Filter for simulated UAVP system flight

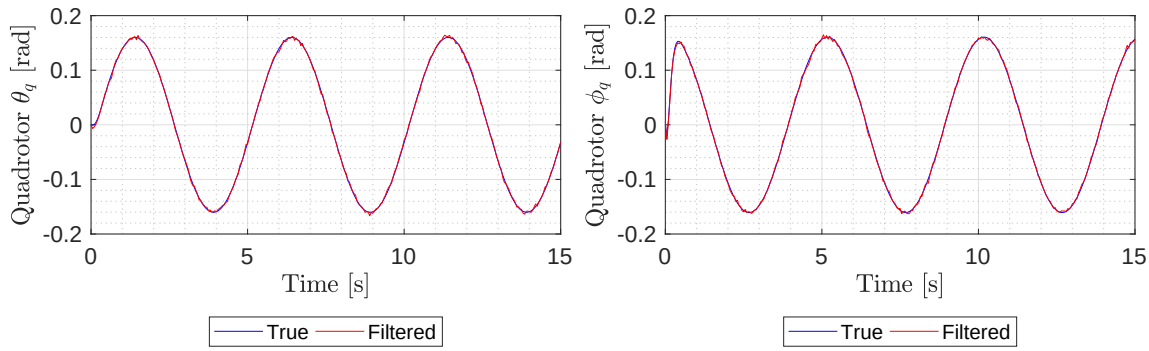
Esimtated variables Linear Kalman Filter	**RMSE**	**NRMSE [%]**
θ_q	2.30×10^{-3} rad	98.0
ϕ_q	2.40×10^{-3} rad	97.8

Estimated variables Unscented KF	**RMSE**	**NRMSE [%]**
x_q	2.90×10^{-3} m	99.6
y_q	2.90×10^{-3} m	99.6
z_q	1.07×10^{-5} m	98.9
θ_l	2.90×10^{-3} rad	98.2
ϕ_l	2.90×10^{-3} rad	98.5
$\dot{x}_q$	5.52×10^{-2} m/s	93.7
$\dot{y}_q$	5.50×10^{-2} m/s	93.2
$\dot{z}_q$	2.20×10^{-3} m/s	87.4
$\dot{\theta}_l$	1.91×10^{-1} rad/s	53.1
$\dot{\phi}_l$	1.21×10^{-1} rad/s	79.0

11-3-1 Linear Kalman Filter

The true to estimated/filtered output of the Linear KF is shown in Figure 11-4. The estimator's fit to true data as (N)RMSE is given in Table 11-3.

Figures 11-4a and 11-4b indicate that due to the availability of very accurate measurements of θ_q and ϕ_q (which would be directly obtainable from the low-noise MCS measurements in experimental conditions) the fit is very good, with an NRMSE greater than 97%, with very low noise. The KF has also been tuned to rely heavily on the measurements with the process model used to reduce the noise. The fit for the $\ddot{z}_u$ is not shown as the value is computed using only the model prediction skipping the measurement update of the KF. Therefore, the fit is as good as the model, and as the model is used for the simulated study, computing a fit is meaningless.



(a) Quadrotor pitch angle **(b)** Quadrotor roll angle

Figure 11-4: Comparison of true to the estimated quadrotor pitch and roll for the simulated UAVP flight with Δt=0.05 s

11-3-2 Unscented Kalman Filter

The true to estimated/filtered states of the Unscented KF is shown in Figures 11-5 and 11-6. The estimator's fit to true data as (N)RMSE is given in Table 11-3. The estimation error, specifically for the $\dot{\boldsymbol{q}}$ estimations, is shown in Figure 11-7 for the first 4.0 s of simulation.

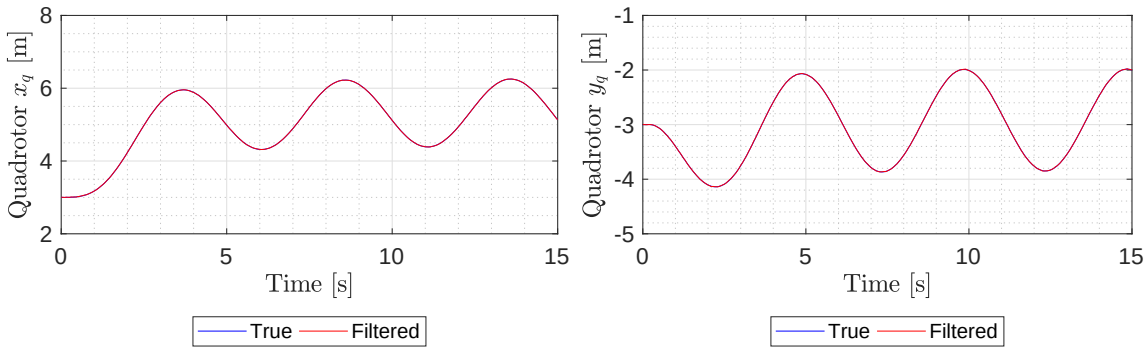
From Figure 11-5, it is again apparent that the low noise measurements already produces accurate measures of $\boldsymbol{q}$. Therefore, as shown in Table 11-3, the fit for the estimated variables in $\boldsymbol{q}$ is very good with a NRMSE of at least 98%. One of the main reason for using the Unscented KF was to produce good estimates of $\dot{\boldsymbol{q}}$. As shown in Figure 11-6 there is more discrepancy between the estimated/filtered and the true value which is expected given there are no direct measures available for any variables in $\dot{\boldsymbol{q}}$. Therefore, the reliance on the UAVP process model to obtain the time derivatives results in some noise propagation from the $\boldsymbol{q}$ measures which are then differentiated (hence amplified) in the KF prediction step. However, using the filtering process the noise amplification is acceptable and a good fit is obtained.

The estimation of $\dot{x}_q$ and $\dot{y}_q$ as shown in Figures 11-6a and 11-6b is very good with the NRMSE of 93.7% and 93.2% respectively. As shown in Figures 11-7a and 11-7b the estimator converges within two time steps to the true value. For $\dot{z}_q$, the estimation as shown in Figure 11-6c is good with an NRMSE of 87.4%. The error convergence for the $\dot{z}_q$ estimate as shown 11-7c is comparable to that of $\dot{x}_q$ and $\dot{y}_q$.

Estimation of the suspension angles rates $\dot{\theta}_l$ and $\dot{\phi}_l$ have the lowest fit compared to all other estimated states as they are highly coupled to other UAVP system dynamics through the EOMs. Therefore, the process noise introduced on the state transition in simulation significantly affects the model predicted suspension angle rates. Given that no measurements of the suspension angle rates are available, then as with other $\dot{\boldsymbol{q}}$ states, the KF must rely on the predictions to produce the estimates. From Figures 11-7d and 11-7e the error convergence, especially for $\dot{\phi}_l$ is fast, however, it is markedly slower than the estimation convergence for $\dot{x}_q$, $\dot{y}_q$ and $\dot{z}_q$. Overall the true and estimated response shapes as shown in Figures 11-6d and 11-6e are comparable with differences in amplitude being the significant discrepancy.

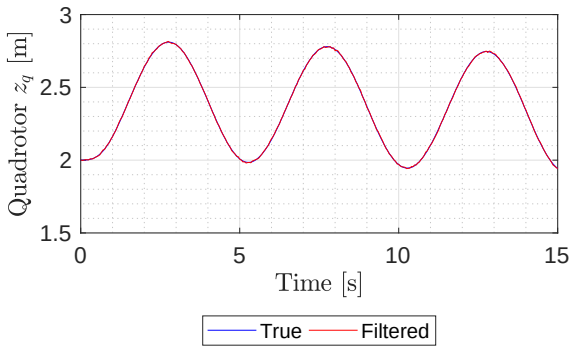
Concluding, the limitation of the currently implemented Cascaded KF is the high reliance on accurate MCS data for feeding measurements to the Linear and Unscented KF. Additionally, due to lack of availability of any time rate variable measurements, which include the entries

of $\dot{\boldsymbol{q}}$, the KFs rely heavily on the process models for predicting these values. The KFs have only been verified in simulation, however, experimental validation can permit a more comprehensive analysis of the KFs performance. This is currently beyond the scope of this research and it is recommended that future studies follow up on this.

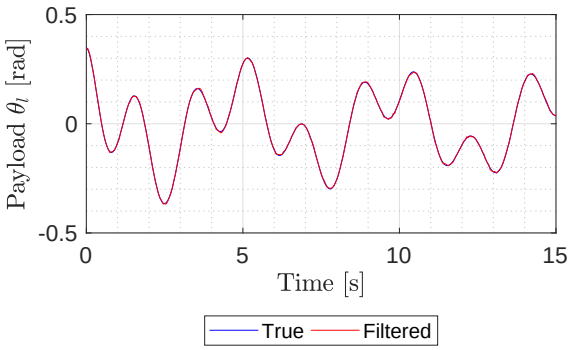


(a) Quadrotor position x

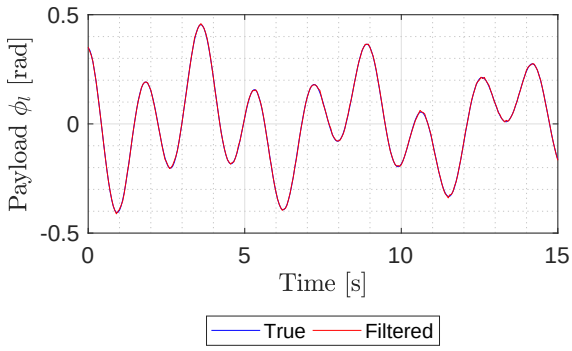
(b) Quadrotor position y



(c) Quadrotor position z

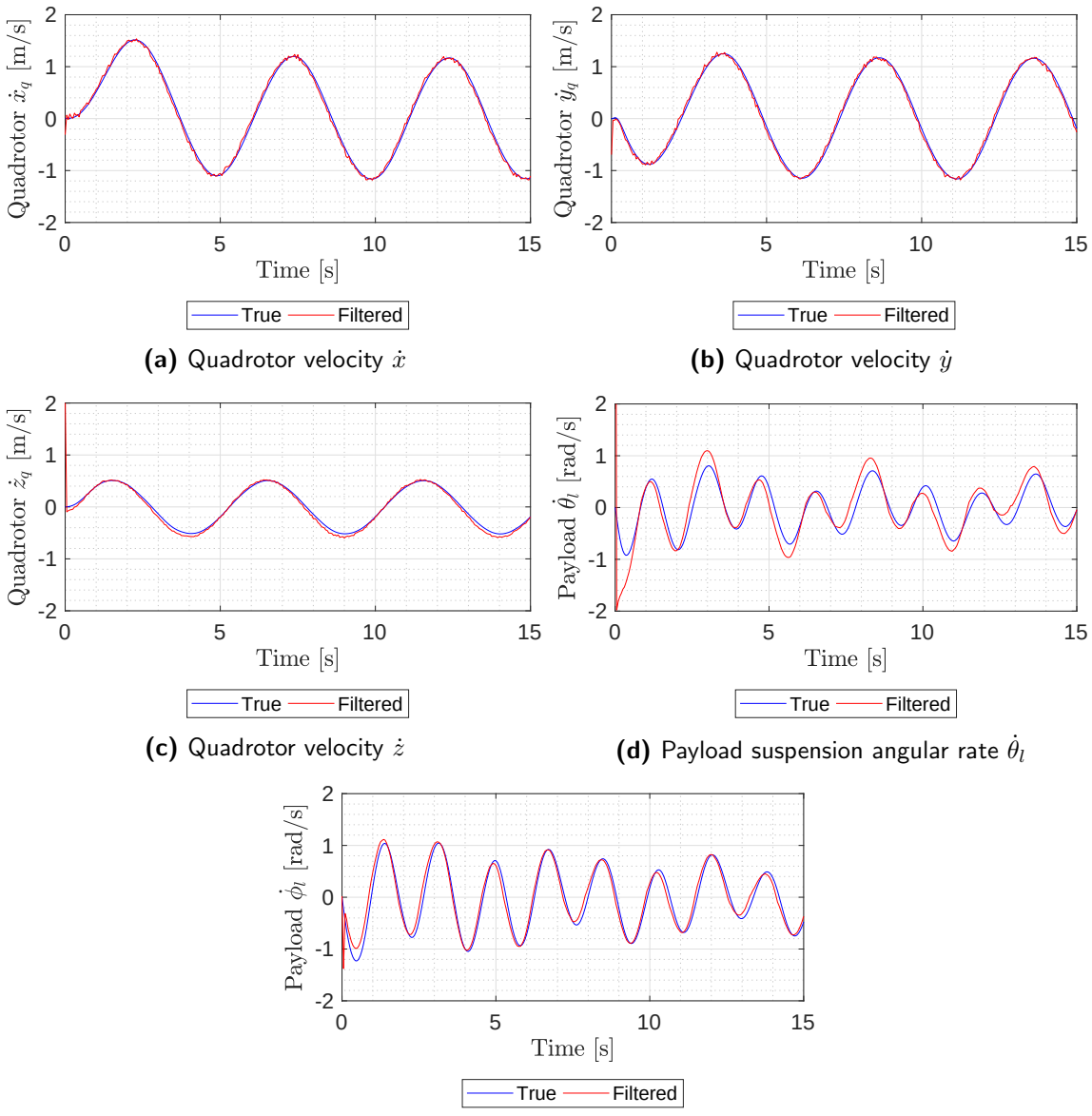


(d) Payload θ_l suspension angle



(e) Quadrotor ϕ_l suspension angle

Figure 11-5: Comparison of true to the estimated UAVP model $\boldsymbol{q}$ states for a simulated UAVP system flight with Δt=0.05 s



(a) Quadrotor velocity $\dot{x}$

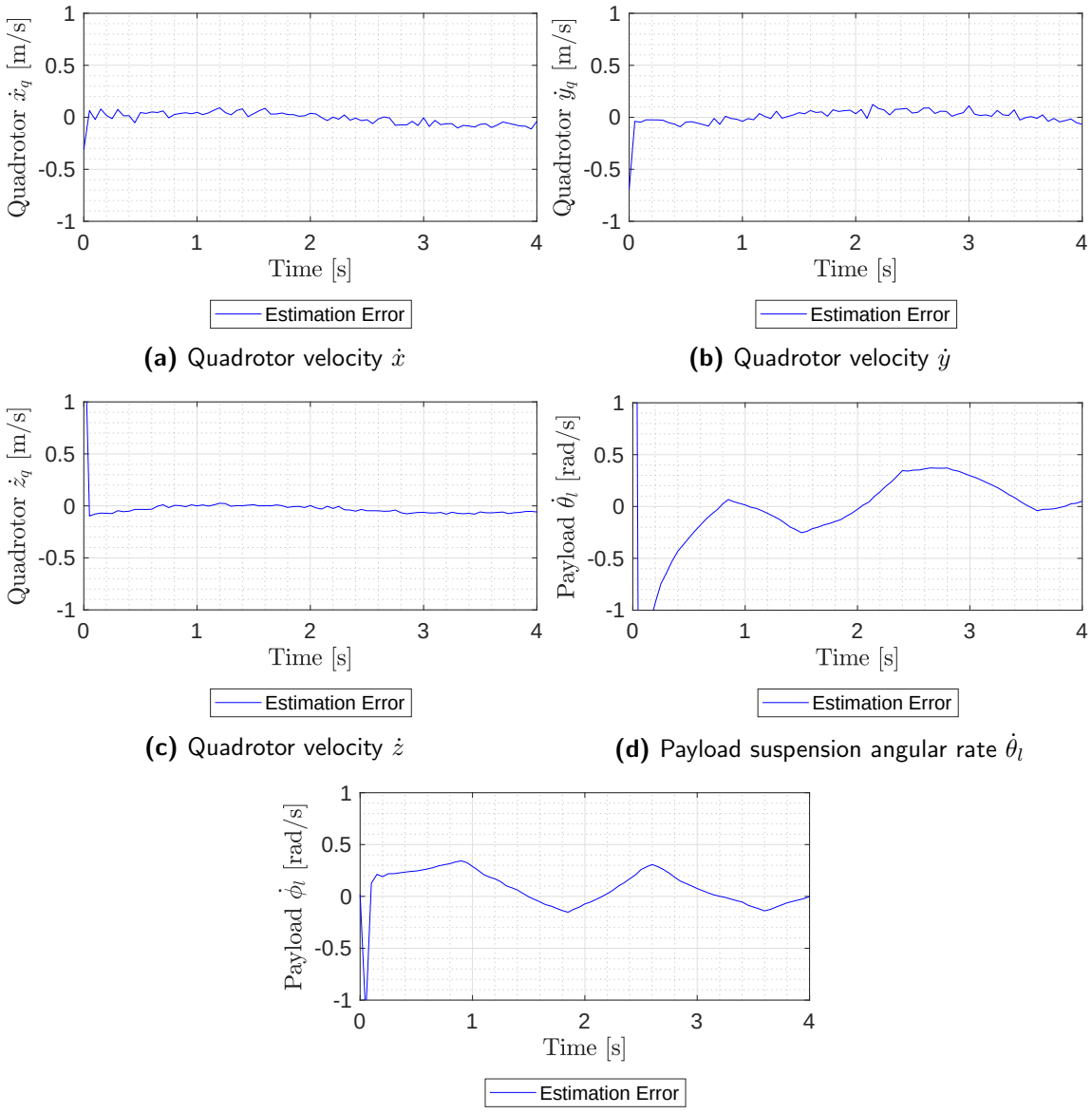
(b) Quadrotor velocity $\dot{y}$

(c) Quadrotor velocity $\dot{z}$

(d) Payload suspension angular rate $\dot{\theta}_l$

(e) Quadrotor suspension angular rate $\dot{\phi}_l$

Figure 11-6: Comparison of true to the estimated UAVP model $\dot{\boldsymbol{q}}$ states for a simulated UAVP system flight with Δt=0.05 s



(a) Quadrotor velocity $\dot{x}$

(b) Quadrotor velocity $\dot{y}$

(c) Quadrotor velocity $\dot{z}$

(d) Payload suspension angular rate $\dot{\theta}_l$

(e) Quadrotor $\dot{\phi}_l$ suspension angular rate

Figure 11-7: Convergence of estimation error within first 4.0 s for the UAVP model $\dot{\boldsymbol{q}}$ states for a simulated UAVP system flight with Δt=0.05 s

Chapter 12

Real-Time Software based Control System Framework

As a supplement to the paper, the simulation and experimental framework is further described in detail. A custom real-time control software and framework was developed as part of this thesis to enable the fully autonomous operation of an Unmanned Aerial Vehicle with suspended Payload (UAV(P)). This framework took a significant amount of programming, testing, verification and validation effort over multiple months and is planned to be used by other students in their research and theses. In Section 12-1, the programmed real-time controller is outlined with technical details. Sections 12-2 describe the operation of the software framework, and Section 12-3 presents contributions and extensions.

12-1 Real-Time Control System Software Architecture

The framework has been written to allow both simulated and experimental studies in one integrated package. This full stack allows simulation trials to readily and quickly be deployed for experimental trials with no recoding work. The discussion of this software package picks up where the preliminary framework in Sections 6-2 and 6-3 left off outlining the physical workspace and hardware used. Readers are advised to familiarise themselves with those sections before proceeding with this chapter.

12-1-1 Software Overview

The MATLAB based software is split up in two main scripts that run in parallel on possibly a multi-computer setup,

- *run_main.m* - Controller that handles all commands generated and sent to the quadrotor (simulated or real) and performs all inter-hardware communication

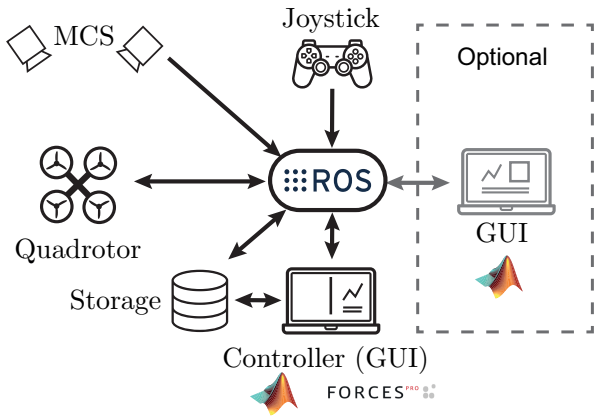


Figure 12-1: Hardware and software data connections and interface enabled through ROS based communication network. ROS network is hosted on the controller computer. The GUI may either run locally on the controller computer, or any external client on the ROS enabled local area network.

- *run_visual.m* - (Optional) Graphical User Interface (GUI) for user interaction, UAVP system and trajectory visualisation

A ROS network is used to interface to all physical hardware which includes the OptiTrack Motion Capture System (MCS), the Parrot Bebop 2 quadrotor [1], the manual gaming joystick and potentially any client computers on which the GUI is running. This network is schematised in Figure 12-1

Even though this software has been designed with the Parrot Software Development Kit (SDK)[2] to interface with the quadrotor, it is general enough to exchange out for other SDKs for other quadrotor hardware. For a list of ancillary software packages, refer to Appendix F.

12-1-2 System and Scenario Initialisation

The system definition and workspace is initialised in a file *initialise.m* containing variables that are user configurable before run-time. These variables include, amongst others, the quadrotor and payload mass, the suspension cable length, drag coefficients and workspace limits. A scenario file then defines the initial start and goal point of the quadrotor and the dimensions of any virtual or real obstacles in the workspace.

Obstacles in the workspace may either be virtual or real, and static or dynamic. For real obstacles in the experimental mode the MCS can capture real-time obstacle positions and relay those to the controller for dynamic trajectory planning. Additionally, to enable studies with static and dynamic obstacles in simulation, or when physical objects are not available, virtual obstacles may be introduced into the experimental workspace. Summarising, the following scenarios are possible with the framework;

- Simulation mode: virtual static or dynamic obstacles

[1] Parrot. "Parrot Bebop 2". Accessed January 18, 2018. `https://www.parrot.com/global/drones/parrot-bebop-2`

[2] Parrot. "ARDroneSDK3". Accessed September 26 2017. `http://developer.parrot.com/docs/bebop/`

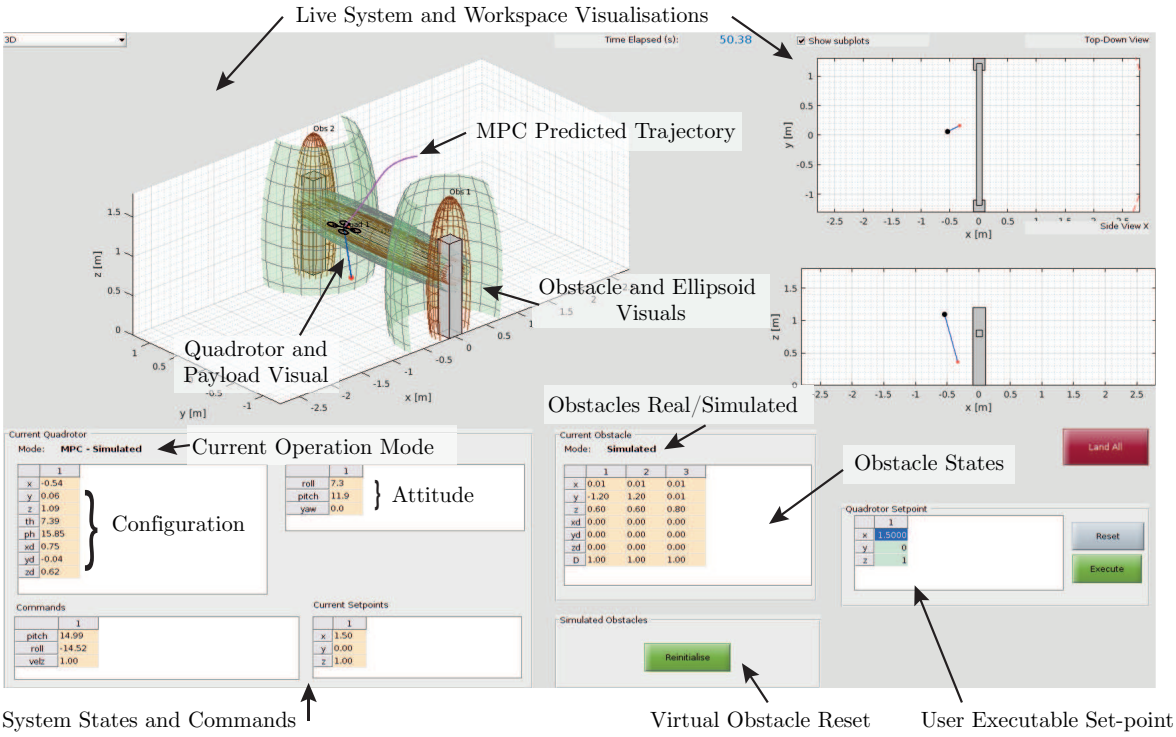


Figure 12-2: Graphical User Interface (GUI) in MATLAB used to dynamically interact with the real-time controller framework

- Experimental mode: virtual static or dynamic obstacles, or, real static or dynamic obstacles

12-1-3 Interacting with the Graphical User Interface

The Graphical User Interface (GUI) allows users to interact with the controller and see a real-time visualisation of the UAVP system response in simulation or experiments. Figure 12-2 shows the GUI with the various features labelled. Most importantly, the GUI allows the user to set the UAV goal position (set-point) to interactively command the controller to execute the trajectory planning to the new goal. For debugging purposes, values of several key variables including the estimated and filtered UAVP system configuration and time derivatives are displayed.

The GUI runs on a parallel MATLAB script to the controller; the division is necessary such that the graphic processing does not affect the more critical real-time controller performance. The communication with parallelisation is achieved through ROS based asynchronous bi-directional message exchanges allowing the two scripts to run independently.

12-1-4 Logging Data for Post-Processing and Replay

Post-processing and replay of the simulation/experiment is possible from data that is logged in-the-loop to a MATLAB log data file and a ROS rosbag file. These two logging approaches

provides the data to re-visualise the entire simulation/experiment in the GUI as well as use the run-time data to perform post-processing to gather statistics and generate plots.

12-1-5 Controller Computational Performance on Standard Computers

The scientific article presents NMPC solve times results and as these are entirely computer hardware dependent, for clarification, additional details of the computer used are provided. The MATLAB based controller script runs on a laptop[3] with an Intel Core i7-3610QM Quad-Core processor running at 2.30 GHz (with a maximum 3.30 GHz); the processor frequency is dynamically adjusted according to demand. Additionally, the laptop had 8 GB of random access memory and Quadro K1000m graphics processing unit. As introduced in Section 10-1-2, the optimisation problem is solved using a C based solver FORCES Pro (Domahidi & Jerez, 2014) that is called from MATLAB as a MEX file. When running the controller, the MATLAB instance utilises at maximum 25% of the processor effectively only using one out of four processing cores. This makes the controller amendable to small microcontrollers that increasingly have very powerful processors such as an ODROID[4]. The graphics processing unit is not used in the optimisation computations. As the control framework is divided over two scripts running on independent MATLAB instances, it is advisable to use a dual core processor at the minimum so each MATLAB instance has sufficient resources.

12-2 Operation of the Framework

When used for experiments, the framework has two main modes of operation, namely *manual* and *autonomous* (NMPC based) control. Following take-off, the two modes can be switched according to the finite state machine schematic as shown in Figure 12-3.

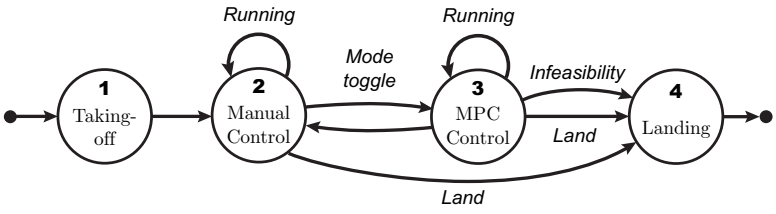


Figure 12-3: Finite State Machine for quadrotor operation in experimental studies from take-off to a landing with toggling between manual and MPC control.

The standard procedure for performing a flight is as follows; first the quadrotor takes-off after which it enters the manual control mode. The MPC control mode is enabled by the user through the joystick and may be terminated at any point to switch back into manual mode.

[3]Hewlett Packard. "HP EliteBook 8570w Mobile Workstation". Accessed 12 January, 2018. `https://support.hp.com/us-en/product/hp-elitebook-8570w-mobile-workstation/5257502/`

[4]Hardkernel. "ODROID". Accessed 22 January, 2018. `http://www.hardkernel.com/`

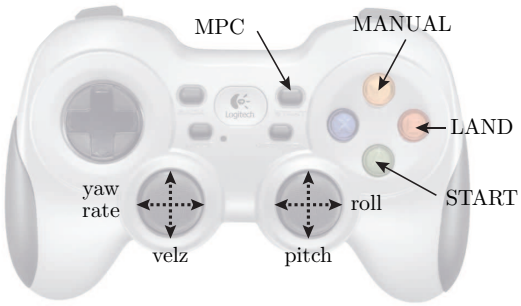


Figure 12-4: Schematic of Logitech F710 joystick controller showing manual control commands and mode switching buttons. The two rockers allow the continuous range of inputs for pitch, roll, vertical velocity (velz) and yaw rate (disabled for this study) to be sent. Buttons are used to enable modes of operations.

The MPC mode continues running indefinitely with the following exceptions, 1) the optimiser encounters an infeasibility in the problem, or 2) the quadrotor battery simply runs out. When this occurs, for safety reasons, the quadrotor is commanded to land on-the-spot.

12-2-1 Manual Control Mode

When performing manually controlled flight, the human operator replaces the MPC controller in the control loop. Manual control actions are commanded through the joystick[5] after which the commands are sent through MATLAB which acts as a bridge. The wireless transmissions and signal processing results in some lag, however, in experimental trials it was found that these lags are sufficiently small to not be noticeable to the user. The commands that can be sent from the joystick are displayed in Figure 12-4

12-2-2 NMPC Autonomous Control Mode

For NMPC based autonomous control, the human is removed from the control loop and only provides high level objectives through desired goal positions. The user may define a navigation goal that the GUI will verify to be within the workspace before execution. It is possible that by providing a list of navigation points in successive order, trajectories can be followed in a set-point tracking manner.

In the unlikely event that the NMPC controller or quadrotor destabilises, manual control can be reclaimed or a high-priority emergency command to land the quadrotor can be sent through the GUI or joystick.

12-3 Code Acknowledgments and Extensions

Considerable time and effort was put into programming this complete software framework, then testing and debugging it. The software uses snippets of code from a framework written

[5]Logitech. "F710 Wireless Gamepad". Accessed January 18, 2018. `https://www.logitechg.com/en-us/product/f710-wireless-gamepad`

by Tobias Naegeli at ETH Zürich and released as part of an article (Naegeli et al., 2017). The snippets used pertain to ROS functions for communicating with the Parrot Bebop 2, obtaining OptiTrack data and general communication over ROS. The developed framework is written with object-oriented programming methods resulting in a modular code enabling its use in various studies involving (multiple) UAVs including the UAVP system.

Part IV

Appendices

Appendix A

Executive Summary of the Preliminary Study

A synopsis of the literature study highlights the state-of-the-art approaches to UAV and Payload (UAVP) system dynamic modelling, collision avoidance techniques, controller design and learning for model adaptability to structural/parametric uncertainties. It is shown that current implementations of UAVP controller design is limited to open-loop trajectory design and precise tracking with seldom any approaches that account for planning under uncertainty in closed-loop. This has often led to inefficient, slow and conservative UAVP planning in largely static environments with full environment knowledge. In the pursuit of addressing this standing issues, this preliminary study is concerned with the identification, verification and validation of the Unmanned Aerial Vehicle (UAV) and swung payload dynamics to enable its implementation in a model based Model Predictive Controller (MPC) in the subsequent study.

The Equation of Motion (EOMs), a set of non-linear Ordinary Differential Equations (ODEs), are derived from first-principles using the Lagrangian mechanics approach. Assumptions are made which include the treatment of the suspension cable as a rigid link and the payload as a point mass simplifying the system definition and mechanics. To enable studies on the model, a MATLAB simulation environment and physical indoor experimental workspace was established. Experimentation was performed on a Parrot Bebop 2 quadrotor that was attached with a light suspended payload. All inter-hardware communications was performed over a ROS based communication network and external real-time UAV and payload pose data was acquired from an optical Motion Capturing System (MCS).

To enable the identification, verification and validation of the model, three experiments were performed. The first experiment involved the identification of the functional relation between the commanded pitch, roll and vertical velocity commands and the actual observed UAV response. It was determined that assuming immediate and exact execution of input commands by the UAV is unrealistic and that a (delayed) response model would need to be identified to complete the UAVP model. Experimental runs were performed from which three second-order

state-space models, one for each input command, were acquired that were able to adequately capture the quadrotor's attitude and vertical velocity dynamics. The models demonstrated an exceptional fit of 94.59%, 91.09% and 91.38% (NRMSE) on the estimation dataset respectively and were of an acceptable quality according to a residual analysis performed. The dynamic response resulting from the models is stable with relatively high damping ratio in the range of 0.75-0.90. Additionally, observations showed that the UAV pitch and roll dynamics did not consistently converge to a fixed steady state value over repeated runs due to the internal state feedback nature by which the Parrot Bebop executes attitude commands. The state feedback is used to perform attitude stabilisation, and due to possible errors in this, the response is not consistent. Additionally, the overall UAV attitude response gain is not one resulting in an inexact input execution.

During definition of the UAVP model, it was assumed that the suspension cable was rigid. To validate the assumption, a cable slackness test was performed using a UAV with suspended payload. The second experiment involved flying the UAV using gradual and aggressive inputs over the UAV's allowable input ranges as to observe whether the cable ever became slack. The cable would never slacken under the presumption that the suspension cable length remains constant (fully stretched) throughout the experiment. The results showed minimal change in the suspension cable length pointing to no switching in the cable state, however, it can not conclusively be said whether the cable never becomes slack. This is partly due to a lack of accurate cable measurements and future studies can potentially measure the cable tensile force which will clearly indicate the slack state when the tensile force becomes zero.

The final experiment involved validating the full UAVP model. Short single step input runs as well as longer runs with combined inputs were performed to perform a comparison. For the short step inputs of about 1 second, the simulated and experimental suspended payload frequency and amplitude response was equivalent with a good overall fit exceeding 70% when the identified dynamics are included. Excluding the identified UAV dynamics in the model resulted in a poor performance with a fit in the range of $20 - 30\%$. Undoubtedly there were discrepancies between the simulated and experimental responses due to un-modelled dynamics and imperfectly estimated effects such as drag. During the longer experimental runs lasting around 50 seconds it was observed that the simulated and experimental dynamics agreed the most when significant energy was introduced in a certain swing direction. For swing directions where there was very minimal or no input energy introduced, the swing angle would remain relatively low (< 0.1rad or $< 6^\circ$) and the model and observed behaviour diverged significantly. This is expected as the residual and/or low amplitude oscillations in that direction are not driven by the main UAVP dynamics, but additional phenomenon not implemented in the currently derived UAVP model. A possible significant contribution to these residual oscillations is the generated rotor wake that encapsulates the payload hanging under the UAV. Therefore, the current UAVP model works well when the UAV motion has a dominant effect on the payload dynamics such that significant swinging motion is observed. Over short simulation periods, the model fit is good and as expected the accumulation of errors causes the long term response to deviate significantly from what is actually observed.

The preliminary study highlights the merits and limitations of the derived UAVP model enabling its future implementation in the model based MPC. The simulation and experimental study provides sufficient evidence to show that the obtained model performs sufficiently well and is able to replicate the actual UAVP system response under the assumptions and conditions presented. Steps for further research to be conducted are discussed.

Appendix B

Literature Overview

This literature overview is an abridged version of the comprehensive *Literature Survey* presented in Chapter 4. Note that to the best of the writer's knowledge, this is the state-of-the-art at the time of writing of August 2017.

The modelling and control of aerial vehicles carrying suspended payloads was originally treated for helicopters (Cicolani & Kanning, 1992). Cicolani *et al.* published a comprehensive NASA Technical Paper in 1992 covering the Equations of Motion (EOMs) of slung-load systems including multi-lift systems (multiple vehicles) with different single/multiple suspension points with the intention of helicopter-like vehicles being the suspension point(s). The study helped in the understanding of aerial vehicle assisted payload carriage and the disturbance effects that slung-payloads introduce to the carrying vehicle. However, the theory of helicopter-payload dynamics do not directly translate to UAVP dynamic systems; UAVs have fast, non-linear, unstable dynamics that are only complicated by introducing a swung payload resulting in external disturbances (Palunko, Fierro, & Cruz, 2012). Therefore, this literature overview provides a brief summary of state-of-the-art UAV-Payload (UAVP) system modelling techniques.

Section B-1 discusses the types of UAVs currently found on the market and their applicability for payload carriage. Section B-2 gives an overview of approaches to payload attachment on UAVs. Section B-3 discusses the UAVP problem and the main ideas in literature regarding its modelling. Finally Section B-4 and B-5 provide more details regarding cable slackening and aerodynamic effects on the system.

B-1 Types of Unmanned Aerial Vehicles (UAVs)

Modelling the UAVP system dynamics involves understanding the carrier vehicle dynamics, therefore, a short introduction to UAVs is given highlighting the motivation for choosing the quadrotor type UAV. The Unmanned Aerial Vehicle (UAV) is a category of aerial vehicles that are piloted remotely or (semi-) autonomously with the absence of a human operator

on-board. Within the UAV category, there are two main types of vehicles; commercially popular single/multi-rotor systems (e.g. helicopter, quadrotor/quadcopters, hexacopter, octocopters) and fixed-wing systems (e.g. General Atomics MQ-9 Predator (General Atomics Aeronautical, 2017)). More niche types of vehicles include the flapping-wing systems (e.g. DelFly Micro/Explorer (Micro Aerial Vehicles Laboratory TU Delft, 2017)) or hybrids of such systems (e.g. the ATMOS UAV Marlyn that combines a conventional quadrotor setup for take-off/landing with a fixed-wing flight mode (ATMOS UAV, 2017)). This literature study will focus on modelling the UAV as a multi-rotor system and specifically a quadrotor.

A quadrotor is a four rotor Vertical Take-off and Landing (VTOL) vehicle. Quadrotors are able to perform stationary hovering which is in contrast to fixed-wing UAVs that require a continuous motion to generate lift which is not desirable especially in indoor conditions where space is limited. Consequently, a quadrotor is able to perform agile motions in confined spaces as it has the ability to fly without forward motion. The quadrotor is an under-actuated system (non-holonomic) as it has six Degrees of Freedom (DOF) and only four degrees of control/actuation (four rotors) (De Crousaz et al., 2014). Adding a suspended payload to the system increases the system's DOF by two (pendulum's polar coordinates in vehicle reference frame) and including a flexible rope by a further one (De Crousaz et al., 2014). In literature, a quadrotor's pose dynamics (position and orientation) is usually derived in terms of the body's inertial properties with four rotational velocity inputs, one for each rotor attached motor (Mahony et al., 2012).

B-2 Approaches to Payload Attachment

The *UAV* is the combination of all the components necessary for the vehicle to fly independently; this includes the basic airframe and necessary battery resulting in the vehicle's *Operating Empty Weight*. The payload definition adhered to during this research is as any additional, non-essential mass that is externally attached to the UAV that does not serve any functional purpose related to the flying and/or structure of the UAV. The payload does not directly contribute to additional thrust forces on the system; this would complicate the system dynamics significantly.

When considering aerial vehicles such as a UAV with payload, there are two ways by which payload can be carried, *grasped* loads whereby the payload is rigidly attached to the airframe (Palunko, Cruz, & Fierro, 2012) and more often *suspended* loads whereby the payload is (freely) swung under the vehicle (Feng et al., 2014). In the external grasping case, the payload can be seen as a protruding extension of the vehicle body itself that alters the vehicle's inertial and physical properties. Provided the payload is a rigid body, the grasped payload's position relative to the vehicle is unchanged during all flight manoeuvres (Palunko, Cruz, & Fierro, 2012). In the suspension case for a point load, the payload moves in $\mathbb{R}^3$ relative to the suspension point on the vehicle significantly increasing the complexity of modelling the dynamics.

With regards to this thesis' research, the first case of grasping a payload is not interesting as for the trajectory generation, the planning must only account for an expanded UAV shape with an altered dynamics model. In the suspended payload case, the trajectory generation

problem will have to account for the UAV and relative payload dynamics which is more interesting.

B-3 UAV-Payload Setup and Modelling

The UAVP systems that have been treated in literature are either a single UAV or multiple UAVs cooperatively carrying a suspended payload. Only the single UAV case will be considered for this preliminary study.

The basic UAVP setup consists of a single UAV with one suspended point payload. This problem is usually broken down into a two rigid-body problem where the bodies are coupled through interaction forces. The two body approach allows the UAV and payload dynamics to be treated separately coupling them through the interaction tensile force. Subsequently, cascaded control systems may be designed whereby the payload dynamics drive the quadrotor dynamics and vice versa as is done frequently in research (Gonzalez et al., 2015; Jain, 2015; Palunko, Cruz, & Fierro, 2012; Pizetta et al., 2015, 2016; Trachte et al., 2014). The single UAVP setup has been studied for both planar ($\mathbb{R}^2$) and Three Dimensional (3D) ($\mathbb{R}^3$) motion cases. The point load pendulum dynamics are analogous to a simple (in the case of planar motion) or spherical (three-dimensional case) pendulum attached to a moving suspension point. The Equations of Motion (EOM) are derived using the Newton-Euler approach, or more often the Euler-Lagrange formulation due to more straightforward computation and resulting complexity of the EOMs.

Planar UAVP motion in the XZ plane studies the motion of a quadrotor with a cable suspended load (always in tension) where the system state/configuration will evolve in $SE(2) \times S^1$; for the 3D case, the state evolves in $SE(3) \times S^2$ (Sreenath et al., 2013). In (Pizetta et al., 2015), Pizetta *et al.* derive a dynamics model for the planar UAV-point payload model using the Euler-Lagrange equations whereby the payload's effect on the vehicle is considered as external disturbances. The paper proposes the non-linear modelling and control strategy for the system through feedback linearisation for performing trajectory tracking. In the study the aim was not to make the payload follow a specific trajectory, but rather for a controller to be designed that allows the vehicle to counteract disturbances introduced by the pendulum such that the vehicle could perform trajectory tracking.

Feng *et al.* in (Feng et al., 2014), and Palunko *et al.* in (Palunko, Cruz, & Fierro, 2012) generalises the planar UAVP case to the 3D case for achieving perfect UAV trajectory tracking under payload disturbances as in Pizetta *et al.* work. The EOMs are derived using the Euler-Lagrange formulation. In (Feng et al., 2014), the quadrotor UAV dynamics are also derived and interested readers can follow-up on the article. In (Feng et al., 2014), simulation studies were performed to show the UAVs tracking performance under the payload disturbance while (Palunko, Cruz, & Fierro, 2012) shows experimental studies with the inclusion of achieving swing-free payload motion. The studies demonstrate the model's validity by successfully being able to reject the payload disturbance forces and effectively control the vehicle.

Sreenath *et al.* in (Sreenath et al., 2013) proposed a planning approach where the purpose was to design trajectories for the payload, rather than the vehicle, and then to control the vehicle

(UAV) such that the payload would track its trajectory. This is in contrast to the work of Pizetta *et al.* and Feng *et al.* where the payload was not actively controlled to follow a certain trajectory. Designing a payload trajectory is challenging as the pendulum dynamics are coupled to the vehicle dynamics and is only controlled through the UAV's inputs. Considering the planar XZ case, Sreenath *et al.* establishes that the system is a *differentially flat hybrid system*. Sreenath *et al.* approach enables the pendulum swing to be exploited for dynamic agile motions rather than just suppressing the disturbances introduced by the swing. Using this approach, the differential flatness property is used to generate a UAV trajectory that must be precisely tracked such that the payload follows the arbitrarily designed trajectory. The hybrid characteristic is required to address the case of cable slackening (coupling forces becoming zero) where the system dynamics switch. Using the differentiable flatness property, it was successfully demonstrated in (Sreenath et al., 2013) that payload trajectory tracking is possible in the planar and 3D case.

The single UAVP system has been treated in (Trachte et al., 2014, 2015; Feng et al., 2014; Palunko, Cruz, & Fierro, 2012) and follows the same two rigid body approach to modelling the UAVP system arriving at same or similar models. The contributions of these papers relate more to the control aspect which will be discussed later.

Bisgaard's PhD thesis (Bisgaard, 2008) and the publication (Bisgaard et al., 2009) by Bisgaard *et al.* provide a comprehensive overview for modelling the single and multi-UAV payload system with full derivations of the dynamic EOMs and approaches for including cable slackening and drag effects. Further research can be performed should the scope of the research be expanded to the multi-UAVP problem.

B-4 Cable Slackening of Payload Suspension Link

As previously mentioned, studies by Sreenath *et al.* (Sreenath et al., 2013; Sreenath & Kumar, 2013) and Pizetta *et al.* (Pizetta et al., 2016) considered the switching dynamics introduced by cable slackening resulting in a *hybrid system dynamics model*. Initial studies of single UAVP systems including the work of Palunko *et al.* (Palunko, Cruz, & Fierro, 2012) and Feng *et al.* (Feng et al., 2014) and others (Pizetta et al., 2015; Jain, 2015; Trachte et al., 2014) considered an always non-zero tensile force. For that modelling assumption to be true, the cable must always be taut (fully stretched) so the rope is comparable to a rigid link. Rope slackening fundamentally alters the system dynamics as the two bodies (UAV and payload) become independent in their motion as there is no coupling force, therefore, the uncontrolled payload enters a free-fall under gravity while the UAV is controllable. Consequently, the system EOMs are defined by a *hybrid system model* where the system EOMs switch between two equation sets; the non-zero and zero coupling tensile force case. When considering slackening of the rope, the 3D dynamics no longer evolve in $SE(3) \times S^2$ but rather $SE(3) \times \mathbb{R}^3$ as the rope length becomes one DOF.

Alternative modelling approaches that do not require hybrid system models also exist. In (Dai et al., 2014), Dai *et al.* do not consider a binary state of slack or taut, but rather models the catenary curve formed by the rope going from taut to slack. This allows the EOMs to describe the continuous transition to a slack state eliminating the need for a hybrid system

model. To achieve this, Dai *et al.* model the suspension cable as a series of rigid mass links with an associated state where each link's dynamics is modelled by an EOM. Combining the EOM for each link gives the system of EOMs describing the full wire's motion. This method is computationally expensive as the set of EOMs grows with increasing granularity in modelling the cable. Therefore, this method can more accurately capture the cable's behaviour at the expensive of the high cost associated with increased computation. This is not ideal when real-time execution is of high concern in the to be designed algorithm.

B-5 Aerodynamic Drag on UAV and Payload

Besides cable slackening, aerodynamic drag has a prominent effect on the UAVP dynamics. Studies into UAVP motion have generally addressed only low-speed experiments such that the relative effect of aerodynamic drag when compared to the most significant external force, gravity, is negligible. However, when considering rigid body loads (large surface areas) and/or high-speed motions the drag force is significant as the force is proportional to the surface area perpendicular to the velocity vector and quadratically increases with the velocity magnitude (Bisgaard et al., 2009).

In (Bisgaard et al., 2009), Bisgaard *et al.* consider the quadratic form of aerodynamic drag with velocity in $\mathbb{R}^3$ for the slung payload EOMs. In this case, the drag force is proportional to the velocity squared which is the classical formulation of the simple drag force equation. Klausen *et al.* also considers aerodynamic drag on a UAVP system in (Klausen et al., 2015), however, due to the relatively slow motion of the system (even at what is considered high speed in indoor cases; around $\|V\| \leq 2m/s$), the function V^2 in this range can be roughly estimated by the linear V function. The benefit of this that the drag force equation becomes linear and can be directly implemented into a linear formulation of the EOMs.

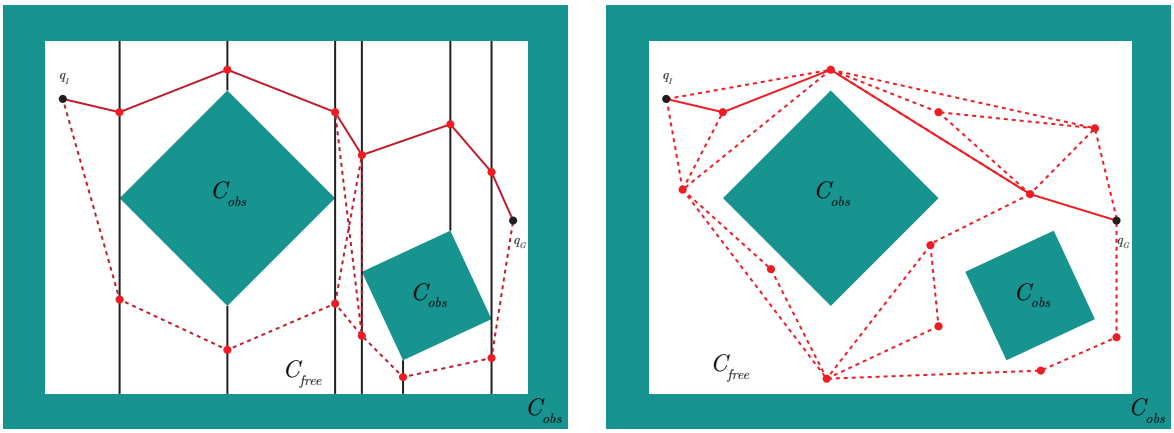
A summarised literature was presented in this chapter showcasing the state-of-the-art in the field of UAVP modelling.

Appendix C

Combinatorial and Sampling-based Planning

In *Combinatorial Planning* a road map or graph is produced in C_{free} with each vertex representing a configuration $q \in C_{free}$ with each edge between vertices also being collision free (contained fully within C_{free}). Under combinatorial planning there are many approaches, for example, a common approach known as *visibility graphs* is presented (Pan & Manocha, 2015). In the *visibility graph* approach the vertices of the C_{obs} space are identified after which straight links are formed between all vertices for which the link is contained fully within C_{free} as shown in Figure C-1a. Once the graph is generated, an optimisation algorithm is implemented to find the shortest route between q_I and q_G on the graph. Combinatorial planning techniques are computationally expensive and scale badly with the dimension of q as it uses the full configuration space knowledge for computing trajectories (Pan & Manocha, 2015).

In *Sampling-based Planning* instead of fully capturing the *C-space*, it is probed/sampled using various different methods to construct a *C-space* approximation. The probing identifies samples which include $q \in C_{obs}$ and $q \in C_{free}$ which are then used as vertices to generate graphs similar to combinatorial planning. A popular example of this approach is LaValle's Rapidly exploring Random Trees (RRTs) that probes the *C-space* by expanding incrementally from an initial configuration q_I forming a graph structure (LaValle, 1998). The visibility graph method can also be used here, as shown in Figure C-1b, whereby mutually visible sample points in C_{free} are joined and the shortest route is found. This method is quicker as an approximation of the *C-space* is used for planning, however, due to the sampling nature, some obstacles might not be captured and like combinatorial planning, it scales badly with the dimension of the *C-space* as more samples are required. Also with increasing scarcity of vertices, the computed trajectory becomes more jagged which is not favourable for smooth robot motions. Important is that sampling-based planning is probabilistically complete such that if there exists a solution, it will be found within infinite time (LaValle, 2006).



(a) Combinatorial planning using way-points defined by midpoints of vertical partitions on obstacle vertices (LaValle, 2011).

(b) Sampling-based planning using randomly identified way-points in C_{free} and visibility graph approach (Pan & Manocha, 2015)

Figure C-1: Example of combinatorial and sampling-based planning algorithm with the shortest route between q_I and q_G shown solid with other routes given as dashed

Appendix D

Model Predictive Control Theory

Model Predictive Control (MPC) solves an Optimal Control Problem (OCP) whereby the objective is to minimise an objective/cost function subject to the constraints (Trachte et al., 2015). To perform MPC control, the following information must be available;

- Mission planning objective(s) to define and tune the cost terms of the objective function that is optimised
- System model to perform the state prediction
- Control and process constraints to limit the range of feasible states and inputs
- Measurements from the system to set the initial condition using the actual system response

The OCP time horizon is discretised in N steps indexed by k with Δt being the time sample step size. The initial system state at $k = 0$ is encoded in $\boldsymbol{x}_0$. The stage cost function c_s is defined over the interval $k = 0 \rightarrow N - 1$ while c_t is a terminal cost function related to being at state x_N with input u_N at time N. The cost functions are constructed to achieve several control objectives, examples include amongst others minimising the tracking error to a reference trajectory or discouraging extreme control inputs (Kamel et al., 2017). The system dynamics are encoded in the discretised (non-)linear function f which is enforced through an equality constraint. State and inputs are constrained to feasible values which are derived from physical/mechanical space limitations denoted by $\mathcal{X}$ and motor/actuator limits by $\mathcal{U}$. Slack variables $\boldsymbol{s}$ may be introduced for soft-constraining the problem to facilitate run-time feasibility by allowing minor constraint violations (Neunert et al., 2016). A high cost is associated with slack variables being non-zero to make their use the last resort option to maintain feasibility of the optimisation problem.

The OCP is optimised with respect to the trajectory $\tilde{\boldsymbol{x}} = [\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N]$ with associated inputs $\tilde{\boldsymbol{u}} = [\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}]$ and slack variables $\tilde{\boldsymbol{s}} = [\boldsymbol{s}_0, \ldots, \boldsymbol{s}_N]$. Together these are the optimisation variable $\tilde{\boldsymbol{z}} = [\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{s}}]$ that are to be optimised. The mathematical definition of the

optimisation problem is given by Eq. D-1.

$$
\begin{aligned}
\min_{\tilde{\boldsymbol{z}}} \quad & J = c_t(\boldsymbol{x}_N, \boldsymbol{u}_N) + \Sigma_{k=0}^{N-1} c_s(\boldsymbol{x}_k, \boldsymbol{u}_k, *_k) \\
\text{s.t.} \quad & \boldsymbol{x}_0 = \boldsymbol{x}(t) && \text{(Initial Estimated State)} \\
& \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) && \text{(Discretised Dynamics)} \\
& g(\boldsymbol{x}_k, \boldsymbol{u}_k, *_k) \geq 0 && \text{(Inequality Constraints)} \\
& \boldsymbol{x}_k \in \mathcal{X} && \text{(State Constraints)} \\
& \boldsymbol{u}_k \in \mathcal{U} && \text{(Input Constraints)} \\
& \boldsymbol{s}_k \geq 0 && \text{(Slack Constraints)}
\end{aligned}
\tag{D-1}
$$

where $*$ represents all additional variables (obstacle positions, slacks etc.) used for cost and constraint definitions. From the optimised $\tilde{\boldsymbol{z}}$ only the first input $\boldsymbol{u}_0$ is executed and the time-horizon is receded by Δt (one time step) to a new planning instance repeating the process. Closed-loop performance is achieved as a result of this receding-horizon approach.

Figure D-1 provides a graphical representation of the MPC algorithm demonstrating the concept.

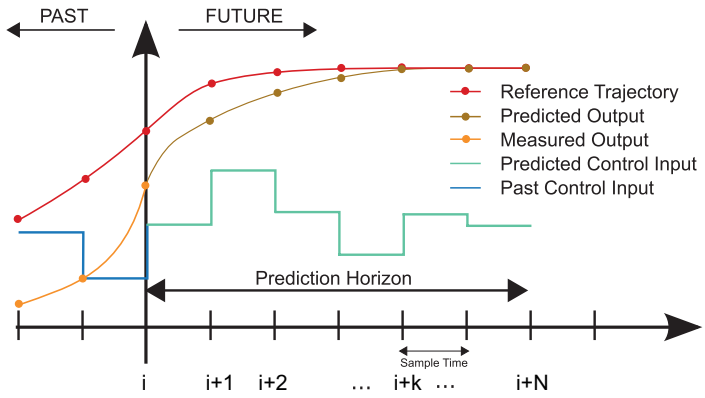


Figure D-1: Model Predictive Control algorithm for trajectory tracking showing the computation performed for one OCP (Adapted from (M. Behrendt, 2009)). The future predicted trajectory and control input is based on the current measured system response/output.

The OCP is solved using an optimisation method that falls into either of two classes, namely *shooting* or *direct* methods. In (Posa & Tedrake, 2013) these two approaches are described; "In shooting methods, the non-linear optimization searches over (a finite parametrization of) $u(t)$, using a forward simulation from $x(0)$ to evaluate the cost of every candidate input trajectory. In direct methods, the non-linear optimization simultaneously searches over parametrizations of $u(t)$ and $x(t)$; here no simulation is required and instead the dynamics are imposed as a set of optimization constraints, typically evaluated at a selection of collocation points".

Appendix E

Motion Capture System Calibration

The Optitrack optical infrared Motion Capture System (MCS) was calibrated prior to performing experiments using the *Motive* calibration software provided by the vendor. The itemised details below are from the calibration report showing the error in tracked two and three dimensional (2D/3D) position readings obtained from the MCS. To understand the provided report, please refer to the OptiTrack documentation[1].

All experimental data collected for the preliminary study was performed under the following MCS calibration;

- Overall Re-projection — Mean 3D Error: 0.326 mm, Mean 2D Error: 0.086 pixels (Exceptional)
- Worst Camera — Mean 3D Error 0.407 mm, Mean 2D Error 0.108 pixels (Exceptional)
- Triangulation — Recommended: 2.7 mm, Residual Mean Error: 0.3 mm
- Overall Wand Error — Mean Error: 0.220 mm (Exceptional)
- Ray length — Suggested Max: 11.0 m

Experimental data collected for the scientific article and supplements was performed under the following MCS calibration;

- Overall Re-projection — Mean 3D Error: 0.323 mm, Mean 2D Error: 0.080 pixels (Exceptional)
- Worst Camera — Mean 3D Error 0.357 mm, Mean 2D Error 0.108 pixels (Exceptional)
- Triangulation — Recommended: 2.7 mm, Residual Mean Error: 0.3 mm
- Overall Wand Error — Mean Error: 0.284 mm (Exceptional)
- Ray length — Suggested Max: 11.5 m

[1] NaturalPoint. "OptiTrack Calibration". Accessed 22 February 2018. `https://v20.wiki.optitrack.com/index.php?title=Calibration`

Appendix F

Framework's MATLAB and ROS Dependencies

To use the programmed control framework, the following software, MATLAB toolboxes and ROS packages are required;

- MATLAB R2017a Linux 64-bit version[1]
- ROS Kinetic Kame[2]
- MATLAB Toolboxes
  - Control System Toolbox (10.2)
  - Image Processing Toolbox (10.0)
  - Curve Fitting Toolbox (3.5.5)
  - Aerospace Toolbox (2.19)
  - Robotics System Toolbox (1.4)
- ROS Packages
  - bebop_autonomy (0.7.0)[3]
  - mocap_optitrack (GitHub latest master 23 May 2016)[4]

To perform the Model Predictive Control computations the FORCES Pro[5] package is required. An academic license is available upon request. **Note:** This package was not required for the preliminary study.

[1] MathWorks. "MATLAB". Accessed September 20 2017 `https://mathworks.com/products/matlab`

[2] ROS. "ROS Kinetic Kame". Accessed September 26 2017. `http://wiki.ros.org/kinetic`

[3] Mani Monajjemi. "bebop_autonomy". Accessed September 26 2017. `http://bebop-autonomy.readthedocs.io/en/latest/`

[4] Kathrin Gräve and Alex Bencz. "mocap_optitrack". Accessed September 20 2017. `http://wiki.ros.org/mocap_optitrack`

[5] embotech. "FORCES PRO". Accessed September 20 2017. `https://www.embotech.com/FORCES-Pro`

Appendix G

Quadrotor Model Identification Experimental Data Post-Processing Details

The estimation and validation dataset for the UAV pitch and roll step response used for system identification had to be de-trended to account for a bias in the recorded response data during the run. The offset value is subtracted from the recorded output signal in the dataset resulting in a de-trended dataset. The output signals are measured in radians and considering the offsets are in the 10^{-3} order, the actual bias is very small. The required offset is computed by comparing the positive and negative peak responses and adjusting for the average value such that it is zero (undisturbed UAV state).

- Pitch dataset de-trending:
  - Estimation dataset, output signal: -0.0090;
  - Validation dataset, output signal: -0.0063;
- Roll dataset de-trending:
  - Estimation dataset, output signal: -0.0061;
  - Validation dataset, output signal: -0.0059;

Appendix H

Identified Quadrotor Model Fit Accuracy

In addition to the first (delayed), second and third order Parrot Bebop 2 quadrotor state-space models identified and presented in the report, input delayed models of the second and third order were also identified. As the delayed second and third order state space models underperformed or showed minimal improvement with respect to their non-delayed counterparts, they were not considered for implementation in the UAVP model. The input delays were determined by trial-and-error through selection of various input delays in the range 0 to 0.2 seconds. The chosen input delay resulted in the best fitting model. Inputs delays longer than 0.2 seconds showed severe degradation of model fit indicating excessive input delay.

The delays in the first, second and third order models are presented in Table H-1. The Normalised Root Mean Square Error (NRMSE) fit of the model response when compared to the experimentally obtained estimation and validation dataset is provided in Table H-2. The estimation dataset included the recorded data used to identify the model while the validation dataset is a repeated experimental run used for checking over-fitting and generalisation.

Table H-1: First, second and third order delayed identified quadrotor state-space model input delays

State-space model	**Input delay** [s]		
	Order 1	**Order 2**	**Order 3**
Pitch	0.10	0.10	0.10
Roll	0.10	0.10	0.10
Vertical Velocity	0.12	0.10	0.14

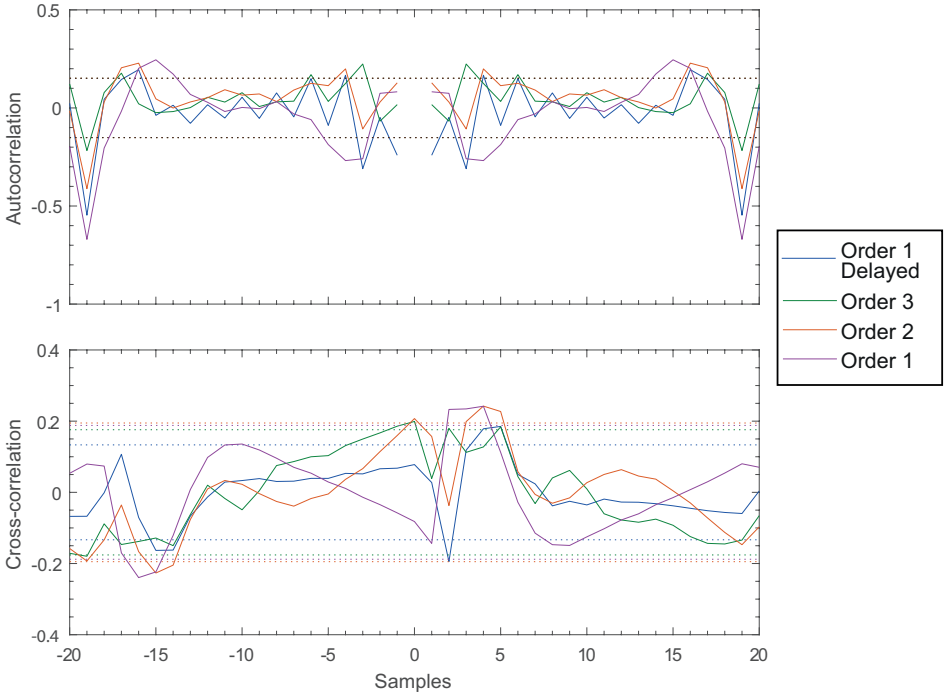
Table H-2: First, second and third order (delayed) identified quadrotor state-space model fit as NRMSE percentage to estimation (est.) and validation (val.) dataset.

System	Dataset fit of the identified state-space model [%]											
	Order 1		Order 1 delayed		Order 2		Order 2 delayed		Order 3		Order 3 delayed	
	Est.	Val.	Est.	Val.	Est.	Val.	Est.	Val.	Est.	Val.	Est.	Val.
Pitch	84.68	84.55	91.72	91.08	94.59	93.81	94.21	93.52	95.93	94.96	94.27	93.57
Roll	82.34	81.69	89.95	88.54	91.09	89.08	91.19	90.05	92.76	90.17	92.19	90.22
Vertical Velocity	77.12	76.42	87.52	89.01	91.38	91.56	91.25	91.48	91.19	92.19	91.39	90.43

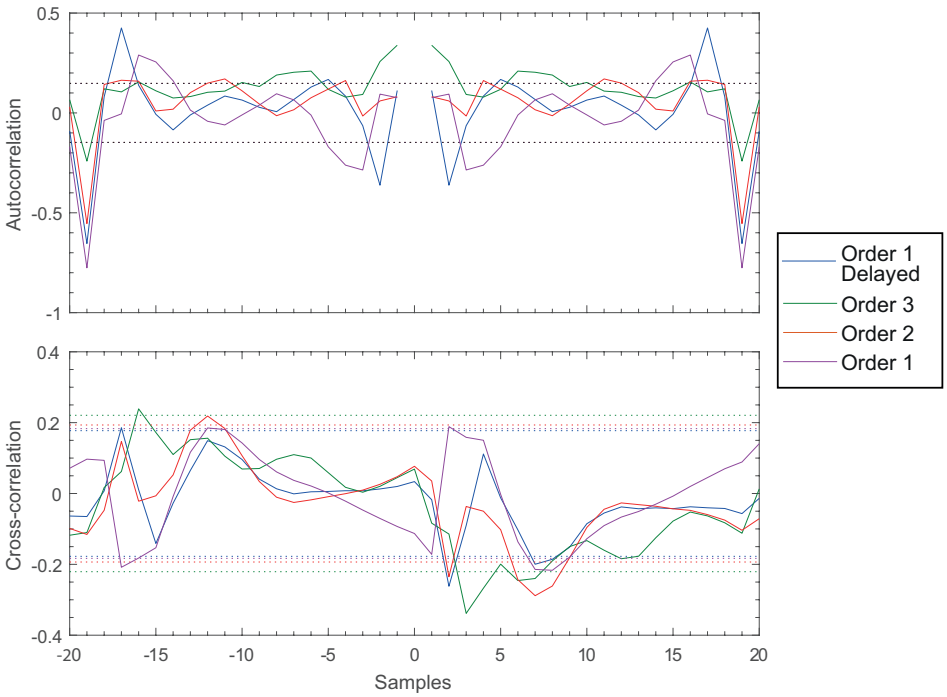
Appendix I

Identified Quadrotor Model Residual Analysis for Fit Quality

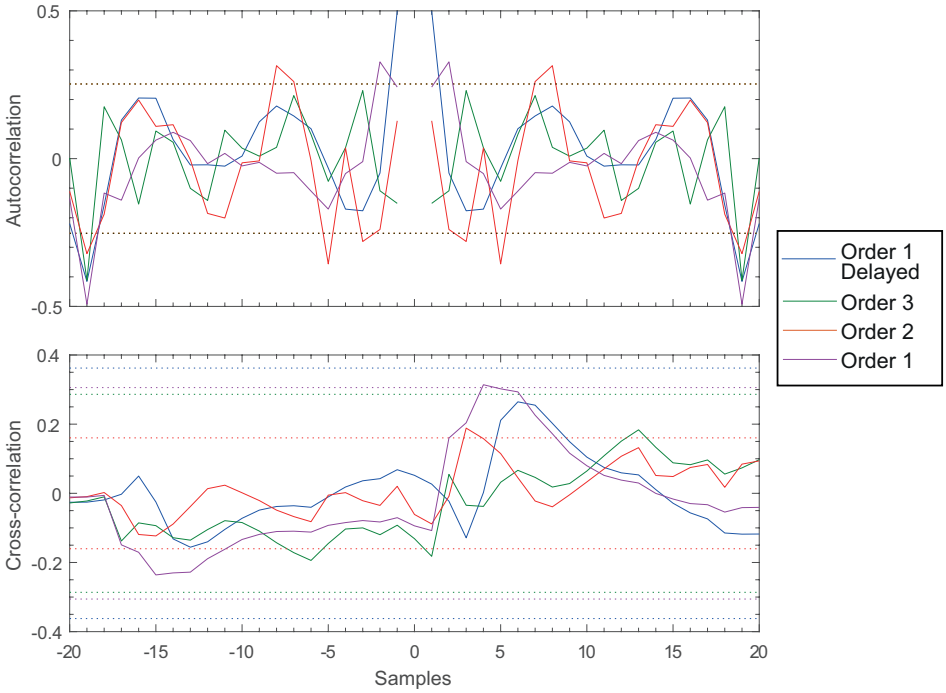
Residual analysis is performed on the identified Parrot Bebop 2 quadrotor state-space models to evaluate their quality. "Residuals are differences between the one-step-predicted output from the model and the measured output from the validation data set" (MathWorks, 2017). The report only mentions the model fit, however, a residual analysis is necessary to explore the portion of the validation data that cannot be explained by the identified model. Figure I-1 shows the autocorrelations of output residuals (*whiteness test*) and cross-correlation of input with output residuals (*independence test*) for the different order state-space models. For a high quality model, the residual function should fall within the 99% confidence interval identified by the dotted line which matches the colour of the model order. Please refer back to the report for a discussion of the model quality. A more thorough residual analysis is necessary to understand the quality of the obtained models, however, this is currently beyond the scope of this research so future studies can perform such an analysis.



(a) Pitch models



(b) Roll models



(c) Vertical velocity models

Figure I-1: Identified pitch, roll and vertical velocity model residuals with 99% confidence intervals marked with corresponding coloured dash line; Top: Autocorrelation of output residuals, Bottom: cross-correlation of input with output residuals

Appendix J

Primal-Dual Interior-Point Constrained Optimisation Algorithm Overview

The Model Predictive Controller (MPC) algorithm is a receding and finite-horizon, constrained optimisation problem that solves for quasi-optimal solutions. The inclusion of 'quasi-' follows from the fact that only a guaranteed global optimum is found for convex problems under specific conditions. For non-convex problems, the optimiser cannot be guaranteed to converge to the global optimum but rather any minima/maxima, local or global.

A Newton-type, optimisation scheme is implemented in the software package FORCES Pro that is used to achieve real-time computational performance (Domahidi & Jerez, 2014). The specific algorithm is called *Barrier Interior-Point Optimisation* for which a short theoretical exposition is provided based on the derivation provided in (Vanderbei, 2012). The notation convention may differ from the main report and is made clear from the context or by explicit definition in this appendix.

J-1 Optimisation Problem Definition

Take the cost function minimisation optimisation problem with total cost J, and with $f : \mathbb{R}^n \to \mathbb{R}$ subject to p inequality and $q - p$ equality constraints as defined by Eq. J-1.

$$\begin{aligned} J = \min_{\boldsymbol{x} \in \mathbb{R}^n} & f(\boldsymbol{x}) \\ & g_i(\boldsymbol{x}) \geq b_i \ \text{ for } \ i = 1, \ldots, p \in \mathbb{N} \\ & g_i(\boldsymbol{x}) = b_i \ \text{ for } \ i = p+1, \ldots q \in \mathbb{N} \end{aligned} \tag{J-1}$$

Rewrite an equivalent problem using slack variables s_i to convert the inequality constraints to equality constraints.

$$\begin{aligned} J = \min_{\boldsymbol{x} \in \mathbb{R}^n} & f(x) \\ & c_i(\boldsymbol{x}, \boldsymbol{s}) = 0 \ \text{ for } \ i = 1, \ldots, q \in \mathbb{N} \\ & s_i \geq 0 \ \text{ for } \ i = 1, \ldots, p \in \mathbb{N} \end{aligned} \tag{J-2}$$

$$c(\boldsymbol{x}, \boldsymbol{s}) = [g_i(\boldsymbol{x}) - b_i - s_i \ , \ldots , \ g_p(\boldsymbol{x}) - b_k - s_p, \ g_{p+1}(\boldsymbol{x}) - b_{p+1} \ , \ldots , \ g_q(\boldsymbol{x}) - b_q]^\top$$

The non-negativity constraint on the slack variables is rewritten as a logarithmic barrier function cost term giving Eq. J-3 which gives the equivalent *barrier problem.*

$$\begin{aligned} B = \min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}) - \mu \sum_{i=1}^{p} \ln(s_i) \\ c_i(\boldsymbol{x}, \boldsymbol{s}) = 0 \ \text{ for } \ i = 1, \ldots, q \in \mathbb{N} \end{aligned} \tag{J-3}$$

A small positive *barrier parameter* μ is introduced such that for $\lim_{\mu \to 0} \min B = \min J$, the new optimisation problem is equivalent to the original. By definition, the natural logarithmic is undefined for $s_i < 0$ thereby implicitly satisfying the inequality $s_i \geq 0$. Finally, incorporate the equality constraints into the objective function using Lagrange multipliers $\boldsymbol{\lambda}$ giving the Lagrange function Eq. J-4. The *primal* variable is $\boldsymbol{x}$ with the dual variable $\boldsymbol{\lambda} \in \mathbb{R}^q$.

$$L(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s})^\top \boldsymbol{\lambda} - \mu \sum_{i=1}^{p} \ln(s_i) \tag{J-4}$$

J-2 Newton-Raphson Iterative Root Finding Search

The optimisation problem is solved iteratively with a Newton-Raphson gradient based search direction algorithm. The gradients of Eq. J-4 is defined by Eqs. J-5 to J-7 which are set to equal zero. These three necessary conditions constrained optimisation problems are referred to as the *Karush-Kuhn-Tucker (KKT)* conditions (Kuhn, 2014).

$$\nabla_x L \equiv \quad \nabla_x f(\boldsymbol{x}) - \nabla_x \boldsymbol{c}^T(x, s)\boldsymbol{\lambda} = 0 \tag{J-5}$$

$$\nabla_s L \equiv \quad -\mu S^{-1} e + Y\boldsymbol{\lambda} = 0 \tag{J-6}$$

$$\nabla_\lambda L \equiv \quad \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s}) = 0 \tag{J-7}$$

$$\text{with} \quad S = \begin{bmatrix} s_1 & & 0 \\ & \ddots & \\ 0 & & s_p \end{bmatrix} \quad , \quad Y = \begin{bmatrix} I_{(p \times p)} & 0_{(p \times q-p)} \end{bmatrix}$$

Rewriting, we get the set of Eqs. J-8 to J-10.

$$\nabla_x f(\boldsymbol{x}) - \nabla_x \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s})^\top \boldsymbol{\lambda} = 0 \tag{J-8}$$

$$\Lambda S \boldsymbol{e} = \mu \boldsymbol{e} \tag{J-9}$$

$$\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s}) = 0 \tag{J-10}$$

$$\text{with} \quad \Lambda = \text{diag}(Y\boldsymbol{\lambda}) = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_p \end{bmatrix} \quad , \quad \boldsymbol{e} = [1 \ , \ldots , \ 1] \in \mathbb{N}^p$$

Applying the Newton-Raphson method for root-finding to the set of Eqs. J-8 to J-10, the system of equations, as given by Eq. J-11, is obtained with the Δ search steps. The left matrix is the Jacobian to the set of Eqs. J-8 to J-10.

$$\begin{bmatrix} W & 0 & -\nabla_x \boldsymbol{c}(\boldsymbol{x},\boldsymbol{s})^\top \\ 0 & \Lambda & S \\ \nabla_x \boldsymbol{c}(\boldsymbol{x},\boldsymbol{s}) & -Z & 0 \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{s} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = - \begin{bmatrix} \nabla_x f(\boldsymbol{x}) - \nabla_x \boldsymbol{c}^\top(\boldsymbol{x},\boldsymbol{s})\boldsymbol{\lambda} \\ \Lambda S \boldsymbol{e} - \mu \boldsymbol{e} \\ \boldsymbol{c}(\boldsymbol{x},\boldsymbol{s}) \end{bmatrix} \tag{J-11}$$

$$\text{with} \quad Z = \begin{bmatrix} I_{(p\times p)} & 0 \\ 0 & 0_{(q-p\times q-p)} \end{bmatrix}$$
$$W = \nabla^2_{xx} L(\boldsymbol{x},\boldsymbol{s},\boldsymbol{\lambda})$$

Computing the search direction given by the vector of Δ is the most computationally expensive step of the algorithm (Domahidi, Zgraggen, Zeilinger, Morari, & Jones, 2012). In practice this is achieved through various methods, however, this is beyond the scope of this explanation.

Then the variables $\boldsymbol{x},\ \boldsymbol{s},\ \boldsymbol{\lambda}$ are iteratively found by taking step of size α in the search directions identified in Eq. J-11 resulting in the update Eq. J-12. The step-size is regulated to achieve a converging line-search.

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{s} \\ \boldsymbol{\lambda} \end{bmatrix}_{(k+1)} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{s} \\ \boldsymbol{\lambda} \end{bmatrix}_{(k)} + \alpha_{(k)} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{s} \\ \Delta \boldsymbol{\lambda} \end{bmatrix}_{(k)} \tag{J-12}$$

J-3 Algorithm Convergence and Extensions

As the Newton-Raphson method is iterative in nature, the KKT conditions defined by Eqs. J-8 to J-10 are set to be satisfied when a certain tolerance ϵ is satisfied.

$$\begin{aligned} \max \left| \nabla_x f(\boldsymbol{x}) - \nabla_x \boldsymbol{c}(\boldsymbol{x},\boldsymbol{s})^\top \boldsymbol{\lambda} \right| &\leq \epsilon_1 \\ \max |\Lambda S \boldsymbol{e} - \mu \boldsymbol{e}| &\leq \epsilon_2 \\ \max |\boldsymbol{c}(\boldsymbol{x},\boldsymbol{s})| &\leq \epsilon_3 \end{aligned}$$

As the Hessian W is not analytically defined in FORCES Pro, an approximation method is used resulting in a quasi-Newton method known as the *Broyden-Fletcher-Goldfarb-Shanno* algorithm (Domahidi & Jerez, 2014); the specific's of which are beyond the scope of this report. The standard Newton method as presented in this appendix is sufficient to understand the general method utilised to solve the constrained optimisation problem.

Note in (Karmarkar, 1984), Karmarkar first showed that interior-point methods are polynomial time algorithms hence the number of algorithmic steps is $\mathcal{O}(n^k)$ for a non-negative k and n size input. Furthermore, for interested readers, the article (Forsgren, Gill, & Wright, 2002) provides a comprehensive overview of interior-point methods for non-linear optimisation.

Appendix K

NMPC Robustness to Time-Step Size and Execution Lag

The effects of various controller time-step size and command execution lags on the NMPC performance are presented. The results in this appendix expand on the results already presented in the scientific article.

The UAVP system performs a point-to-point navigation tak starting from (-1.5,0,1) to a goal (1.5,0,1) position in meters with a small obstacle positioned at (0,0,0) with an infinite height. As the obstacle is in the direct path from the start to the goal point, the planning must be executed with a curvature as shown in Figure K-1. For all cases the planning stages $N = 18$ and default cost definitions and weights are used with assistive steering disabled. In Figure K-2 the effects of different NMPC planning and control time-steps (controller frequency) on the generated inputs and manoeuvres is presented. In Figure K-3, the NMPC controller uses a fixed Δt=0.05 s planning time-step to generate the predicted trajectory, however, the execution of commands is delayed by 0 to 0.15 seconds to artificially model lag. The lag demonstrates the effects on performance generally seen in real-world controller deployment.

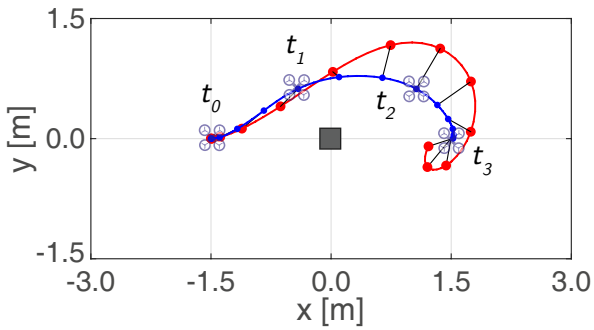
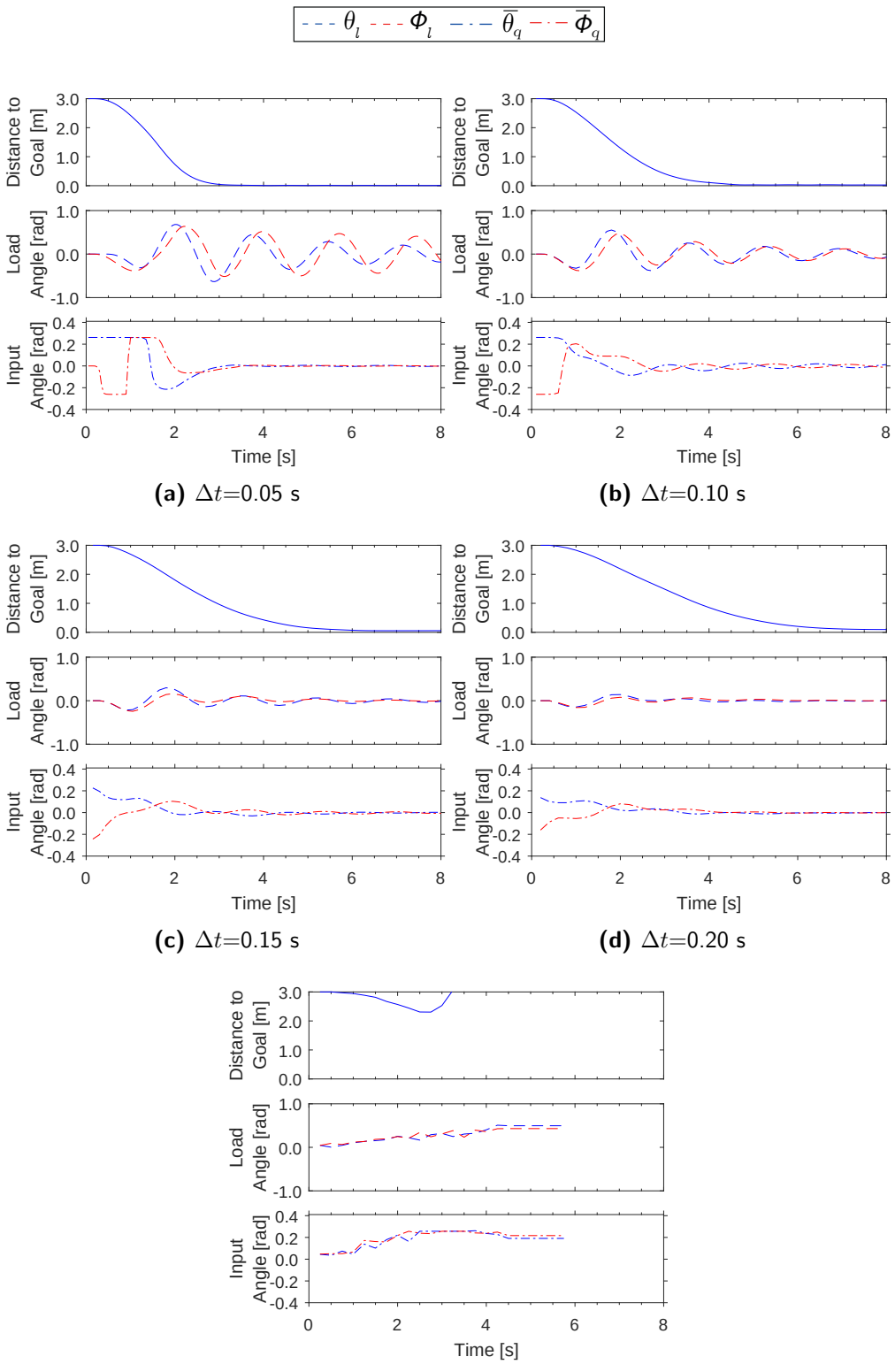


Figure K-1: Example executed point-to-point navigation task that is performed using the NMPC controller with $N = 18$

In Figure K-2 notice how as the Δt becomes longer, the controller compensates for the lower control frequency by making the manoeuvre less agile i.e. smaller suspension angles. Hence, the systems takes longer to reach the goal with a longer Δt. The magnitude of generated input commands is also reduced with increasing Δt to ensure system stability with larger time-steps. These characteristics are all achieved without any re-tuning of the NMPC and is an inherent feature of the NMPC closed-loop planning approach. There is a limit to how large Δt can be which is determined by the discretisation of the non-linear dynamics used. In this study, a Runge-Kutta 2$^{\text{nd}}$ order method is used, so if Δt becomes too large the linearised dynamics may result in error divergence and eventually controller instability. With Δt=0.25 s (4 Hz), the controller destabilises, however, this is already a very low controller frequency.

In Figure K-3, notice how the delay has a minimal effect on the executed trajectory which was performed under the control inputs generated by the NMPC using a time-step Δt=0.05 s. Note that the controller is set up such that it will continue executing the last received command until a new command is received. Therefore, as the NMPC generates commands that should be executed for 0.05 s, in reality they are executed for 0.20 s due to the time delay (lag). Therefore, notice how for the 0.15 s delay case, the payload suspension angles start amplifying and eventually the entire system destabilises. This occurs as the controller tries to reduce the suspension angles when the goal is reached, however, those inputs have the adverse resonating effect when executed over the longer real time-step. In simulation, the total time lag under which the system remains stable is also affected by the discretisation method used for the simulated system model. The Runge-Kutta 2$^{\text{nd}}$ order method is used by both the NMPC controller and to simulate the UAVP dynamics and higher-order methods may be more stable for larger time-steps and lags. In experimental setups, the total allowable lag may be greater, however, this study falls beyond the scope of this research.

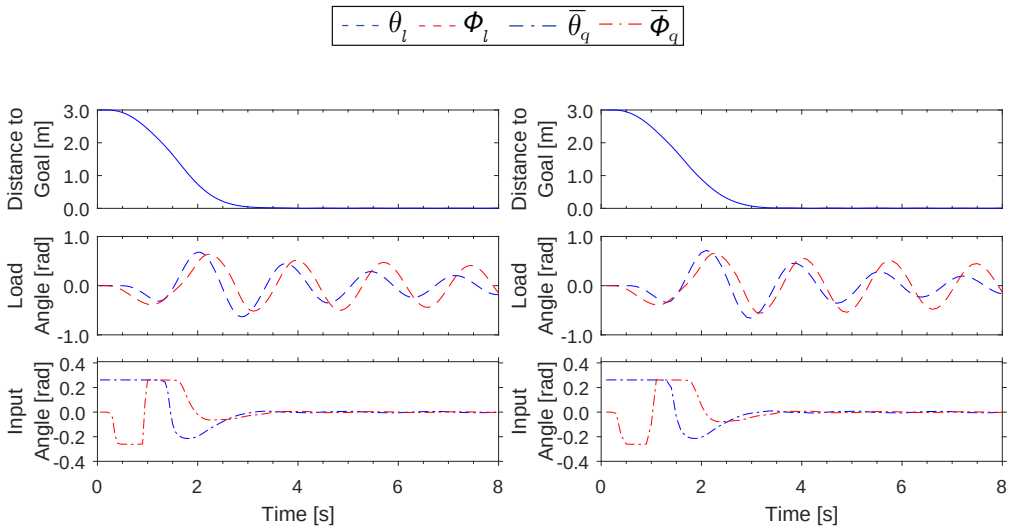


(a) Δt=0.05 s **(b)** Δt=0.10 s

(c) Δt=0.15 s **(d)** Δt=0.20 s

(e) Δt=0.25 s

Figure K-2: Comparison of distance-to-goal, payload suspension angles and NMPC generated inputs for different control time-steps



(a) Δt=0.05 s, no delay

(b) Δt=0.05 s, 0.05 s delay

(c) Δt=0.05 s, 0.10 s delay

(d) Δt=0.05 s, 0.15 s delay

Figure K-3: Comparison of distance-to-goal, payload suspension angles and NMPC generated inputs for different planning to execution time delays

Bibliography

Åkesson, B. M., & Toivonen, H. T. (2006). A neural network model predictive controller. *Journal of Process Control*, *16*(9), 937–946. Available from http://dx.doi.org/10.1016/j.jprocont.2006.06.001

Alonso-Mora, J., Naegeli, T., Siegwart, R., & Beardsley, P. (2015). Collision avoidance for aerial vehicles in multi-agent scenarios. *Autonomous Robots*, *39*(1), 101–121. Available from http://dx.doi.org/10.1007/s10514-015-9429-0

ATMOS UAV. (2017). *Dutch start-up launches next generation mapping drone.* Available from http://www.atmosuav.com/2017/05/dutch-start-launches-next-generation-mapping-drone/ ([Press release. Accessed Jun. 10 2017])

Berg, J. V. D., Guy, S. J., Lin, M., & Manocha, D. (2009). Reciprocal n -body Collision Avoidance. *International Symposium on Robotics Research*, 1–16. Available from http://dx.doi.org/10.1007/978-3-642-19457-3_1

Bisgaard, M. (2008). *Modeling , Estimation, and Control of Helicopter Slung Load System.* Unpublished doctoral dissertation, Aalborg University. Available from http://vbn.aau.dk/files/14412238/thesis_final.pdf

Bisgaard, M., Bendtsen, J. D., & Cour-Harbo, A. (2009). Modeling of Generic Slung Load System. *Journal of Guidance, Control, and Dynamics*, *32*(2), 573–585. Available from http://arc.aiaa.org/doi/10.2514/1.36539

Bisgaard, M., Cour-Harbo, A., & Dimon Bendtsen, J. (2010). Adaptive control system for autonomous helicopter slung load operations. *Control Engineering Practice*, *18*(7), 800–811. Available from http://dx.doi.org/10.1016/j.conengprac.2010.01.017

Burden, R., & Faires, J. (2011). *Numerical Analysis* (9th ed.). Boston: Cengage Learning.

Burgard, W., Stachniss, C., Bennewitz, M., & Arras, K. (2011). *Robot Motion Planning* (No. July). Available from http://ais.informatik.uni-freiburg.de/teaching/ss11/robotics/slides/18-robot-motion-planning.pdf

Čáp, M., Gregoire, J., & Frazzoli, E. (2016). Provably Safe and Deadlock-Free Execution of Multi-Robot Plans under Delaying Disturbances. *arXiv preprint arXiv:1603.08582*, 5113–5118. Available from http://ieeexplore.ieee.org/document/7759750/

Čáp, M., Novak, P., Kleiner, A., & Selecky, M. (2015). Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots. *IEEE Transactions on Automation Science and Engineering*, *12*(3), 835–849. Available from http://ieeexplore.ieee.org/document/7138650/

Čáp, M., Vokínek, J., & Kleiner, A. (2015). Complete Decentralized Method for On-Line Multi-Robot Trajectory Planning in Well-Formed Infrastructures. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling* (pp. 324–332). Jerusalem: AAAI. Available from https://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10504

Chen, L., & Wang, G. (2013). Attitude Stabilization for a Quadrotor Helicopter Using a PD Controller. *IFAC Proceedings Volumes*, *46*(20), 236–239. Available from http://dx.doi.org/10.3182/20130902-3-CN-3020.00163

Cicolani, L. S., & Kanning, G. (1992). *Equations of motion of slung-load systems, including multilift systems* (Tech. Rep.). Moffett Field: National Aeronautics and Space Administration. Available from https://ntrs.nasa.gov/search.jsp?R=19930003627

Cybenko, G. (1989). Approximation by Superpositions of a Sigmoidal Function*. *Mathematics of Control, Signals and Systems*, *2*(4), 303–314. Available from https://doi.org/10.1007/BF02551274

Dai, S., Lee, T., & Bernstein, D. (2014). Adaptive control of a quadrotor UAV transporting a cable-suspended load with unknown mass. In *Proceedings of the IEEE Conference on Decision and Control* (Vol. 53, pp. 6149–6154). Los Angeles: IEEE. Available from http://ieeexplore.ieee.org/document/7040352/

De Croon, G. C., Groen, M. A., De Wagter, C., Remes, B., Ruijsink, R., & Van Oudheusden, B. W. (2012). Design, aerodynamics and autonomy of the DelFly. *Bioinspiration and Biomimetics*, *7*(2), 1–36.

De Crousaz, C., Farshidian, F., & Buchli, J. (2014). Aggressive Optimal Control for Agile Flight with a Slung Load. In *IROS Workshop on Machine Learning in Planning and Control of Robot Motion.* Chicago: IROS.

Domahidi, A., & Jerez, J. (2014, July). *FORCES Professional.* embotech GmbH (http://embotech.com/FORCES-Pro).

Domahidi, A., Zgraggen, A. U., Zeilinger, M. N., Morari, M., & Jones, C. N. (2012, dec). Efficient interior point methods for multistage problems arising in receding horizon control. In *2012 ieee 51st ieee conference on decision and control (cdc)* (pp. 668–674). Maui: IEEE. Available from http://ieeexplore.ieee.org/document/6426855/

Du, K.-L., & Swamy, M. N. S. (2013). Fundamentals of Machine Learning. In *Neural Networks and Statistical Learning* (pp. 15–65). London: Springer London. Available from https://doi.org/10.1007/978-1-4471-5571-3_2

Faust, A., Palunko, I., Cruz, P., Fierro, R., & Tapia, L. (2013). Learning swing-free trajectories for UAVs with a suspended load. In *Proceedings - IEEE International Conference on Robotics and Automation* (pp. 4902–4909). Karlsruhe: IEEE. Available from `http://ieeexplore.ieee.org/document/6631277/`

Feng, Y., Rabbath, C. A., & Su, C.-Y. (2014). Modeling of a Micro UAV with Slung Payload. In K. P. Valavanis & G. J. Vachtsevanos (Eds.), *Handbook of Unmanned Aerial Vehicles* (pp. 1257 – 1272). Dordrecht: Springer Netherlands. Available from `https://link.springer.com/referencework/10.1007/978-90-481-9707-1/`

Fiorini, P., & Shiller, Z. (1998). Motion Planning in Dynamic Environments Using Velocity Obstacles. *The International Journal of Robotics Research*, *17*(7), 760–772. Available from `http://ijr.sagepub.com/cgi/doi/10.1177/027836499801700706`

Foehn, P., Falanga, D., Kuppuswamy, N., Tedrake, R., & Scaramuzza, D. (2017). Fast Trajectory Optimization for Agile Quadrotor Maneuvers with a Cable-Suspended Payload. In *Proceedings of Robotics: Science and Systems* (pp. 1–10). Boston: RSS Foundation.

Forsgren, A., Gill, P. E., & Wright, M. H. (2002, jan). Interior Methods for Nonlinear Optimization. *SIAM Review*, *44*(4), 525–597. Available from `http://epubs.siam.org/doi/10.1137/S0036144502414942`

General Atomics Aeronautical. (2017). *Predator B RPA*. Available from `http://www.ga-asi.com/predator-b` ([Website. Accessed Jul. 31 2017])

Gonzalez, F., Heckmann, A., Notter, S., Zürn, M., Trachte, J., & McFadyen, A. (2015). Non-linear model predictive control for UAVs with slung/swung load. In *ICRA Workshop on Aerial Robotics Manipulation and Load Transportation.* Seattle: ICRA. Available from `https://eprints.qut.edu.au/72665/`

Hermann, R., & Krener, A. J. (1977). Nonlinear Controllability and Observability. *IEEE Transactions on Automatic Control*, *22*(5), 728–740.

Horváth, G. (2003). Neural Networks in System Identification. In *Nato Science Series Sub Series III Computer And Systems Sciences 185* (185th ed., pp. 43–78). Amsterdam: IOS Press.

Jain, R. P. K. (2015). *Transportation of Cable Suspended Load using Unmanned Aerial Vehicles: A Real-time Model Predictive Control approach.* Available from `http://resolver.tudelft.nl/uuid:4c6b4a94-4f15-4e67-8c30-eb8156aab406`

Julier, S. J., & Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In *Proc.SPIE 3068* (p. 12). Available from `http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.280797`

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, *82*(1), 35. Available from `http://fluidsengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1430402`

Kamel, M., Alonso-Mora, J., Siegwart, R., & Nieto, J. (2017). *Nonlinear Model Predictive Control for Multi-Micro Aerial Vehicle Robust Collision Avoidance.* Available from `http://arxiv.org/abs/1703.01164`

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, *4*(4), 373–395.

Khatib, O. (1986). Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, *5*(1), 90–98. Available from dx.doi.org/10.1177/027836498600500106

Klausen, K., Fossen, T. I., & Johansen, T. A. (2015). Nonlinear control of a multirotor UAV with suspended load. In *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015* (pp. 176–184). Denver: IEEE. Available from http://ieeexplore.ieee.org/document/7152289/

Kuhn, H. (2014). Nonlinear programming: A historical view. In G. Giorgi & T. Kjeldsen (Eds.), *Traces and Emergence of Nonlinear Programming* (p. 393-414). Basel: Birkhuser.

Kunz, K., Huck, S. M., & Summers, T. H. (2013). Fast Model Predictive Control of Miniature Helicopters. In *European Control Conference* (pp. 1377–1382). Zurich: IEEE. Available from http://ieeexplore.ieee.org/document/6669699/

LaValle, S. M. (1998). *Rapidly-exploring random trees: A new tool for path planning* (Tech. Rep.). Ames: Iowa State University. Available from http://msl.cs.illinois.edu/~lavalle/papers/Lav98c.pdf

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge: Cambridge University Press. Available from https://www.cambridge.org/core/product/identifier/9780511546877/type/book

LaValle, S. M. (2011, mar). Motion Planning. *IEEE Robotics & Automation Magazine*, *18*(1), 79–89. Available from http://msl.cs.illinois.edu/~lavalle/papers/Lav11b.pdfhttp://ieeexplore.ieee.org/document/5751929/

Lee, J. H. (2011). Model Predictive Control and Dynamic Programming. In *2011 11th International Conference on Control, Automation and Systems* (pp. 1807–1809). Gyeonggi-do: IEEE. Available from http://ieeexplore.ieee.org/document/6106171/

Lee, T. (2018, jan). Geometric Control of Quadrotor UAVs Transporting a Cable-Suspended Rigid Body. *IEEE Transactions on Control Systems Technology*, *26*(1), 255–264. Available from http://ieeexplore.ieee.org/document/7843619/

M. Behrendt. (2009). *A basic working principle of Model Predictive Control.* Available from https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg ([Image Online. Accessed Jul. 03 2017])

Mahony, R., Kumar, V., & Corke, P. (2012). Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. *IEEE Robotics & Automation Magazine*, *19*(3), 20–32. Available from http://ieeexplore.ieee.org/document/6289431/

MathWorks. (2017). *What is Residual Analysis?* Available from https://mathworks.com/help/ident/ug/what-is-residual-analysis.html ([Website. Accessed September 16 2017])

Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, *36*(6), 789–814.

Mellinger, D., Michael, N., & Kumar, V. (2012). Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, *31*(5), 664–674. Available from `http://journals.sagepub.com/doi/abs/10.1177/0278364911434236`

Micro Aerial Vehicles Laboratory TU Delft. (2017). *History of the DelFly project.* Available from `http://www.delfly.nl/history/` ([Website. Accessed Jul. 10 2017])

Min, B., Hong, J., & Matson, E. (2011, oct). Adaptive Robust Control (ARC) for an altitude control of a quadrotor type UAV carrying an unknown payloads. In *2011 11th International Conference on Control, Automation and Systems* (pp. 1147–1151). Gyeonggi-do: IEEE. Available from `http://ieeexplore.ieee.org/document/6106100/`

Monajjemi, Mani. (2015). *bebop autonomy.* Available from `http://bebop-autonomy.readthedocs.io/en/latest/index.html` ([Website. Accessed Sep. 16 2017])

Naegeli, T., Alonso-Mora, J., Domahidi, A., Rus, D., & Hilliges, O. (2017). Real-time Motion Planning for Aerial Videography with Dynamic Obstacle Avoidance and Viewpoint Optimization. *IEEE Robotics and Automation Letters*, *2*(3), 1696 – 1703. Available from `http://ieeexplore.ieee.org/document/7847361/`

National Instruments. (2007). *Relationship Between Bandwidth and Rise Time.* Available from `https://goo.gl/7YCWTC` ([Website. Accessed Sep. 18 2017])

Neunert, M., Crousaz, C. de, Furrer, F., Kamel, M., Farshidian, F., Siegwart, R., et al. (2016, may). Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking. In *2016 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1398–1404). Stockholm: IEEE. Available from `http://ieeexplore.ieee.org/document/7487274/`

Olsder, G., Woude, J. van der, Maks, J., & Jeltsema, D. (2011). *Mathematical Systems Theory* (4th ed.). Delft: VSSD.

Omar, H. (2009). New fuzzy-based anti-swing controller for helicopter slung-load system near hover. In *Computational Intelligence in Robotics and Automation (CIRA), 2009 IEEE International Symposium on* (pp. 474 – 479). Daejeon: IEEE. Available from `http://ieeexplore.ieee.org/document/5423159/`

Palunko, I., Cruz, P., & Fierro, R. (2012). Agile load transportation : Safe and efficient load manipulation with aerial robots. *IEEE Robotics and Automation Magazine*, *19*(3), 69–79. Available from `http://ieeexplore.ieee.org/document/6299175/`

Palunko, I., Fierro, R., & Cruz, P. (2012). Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload: A dynamic programming approach. In *Proceedings - IEEE International Conference on Robotics and Automation* (pp. 2691–2697). Saint Paul: IEEE. Available from `http://ieeexplore.ieee.org/document/6225213/`

Pan, J., & Manocha, D. (2015, mar). Efficient Configuration Space Construction and Optimization for Motion Planning. *Engineering*, *1*(1), 46–57. Available from http://linkinghub.elsevier.com/retrieve/pii/S2095809916300443

PaparazziUAV. (2017). *Bebop Sensor.* Available from https://wiki.paparazziuav.org/wiki/Bebop#Sensors ([Website. Accessed Sep. 26 2017])

Pendleton, S., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y., et al. (2017, feb). Perception, Planning, Control, and Coordination for Autonomous Vehicles. *Machines*, *5*(1), 6. Available from http://www.mdpi.com/2075-1702/5/1/6

Pizetta, I. H. B., Brandao, A. S., & Sarcinelli-Filho, M. (2015, jun). Modelling and control of a PVTOL quadrotor carrying a suspended load. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 444–450). Denver: IEEE. Available from http://ieeexplore.ieee.org/document/7152321/

Pizetta, I. H. B., Brandao, A. S., & Sarcinelli-Filho, M. (2016). Cooperative quadrotors carrying a suspended load. In *2016 International Conference on Unmanned Aircraft Systems, ICUAS 2016* (pp. 1049–1055). Arlington: IEEE. Available from http://ieeexplore.ieee.org/document/7502605/

Posa, M., & Tedrake, R. (2013). Direct Trajectory Optimization of Rigid Body Dynamical Systems Through Contact. In *Algorithmic Foundations of Robotics X* (Vol. 86, pp. 527–542). Berlin: Springer Berlin. Available from http://link.springer.com/10.1007/978-3-642-36279-8

Pounds, P. E., Bersak, D. R., & Dollar, A. M. (2012). Stability of small-scale UAV helicopters and quadrotors with added payload mass under PID control. *Autonomous Robots*, *33*(1-2), 129–142. Available from https://link.springer.com/article/10.1007/s10514-012-9280-5

Qin, S. Z., Su, H. T., & McAvoy, T. J. (1992). Comparison of four neural net learning methods for dynamic system identification. *IEEE Transactions on Neural Networks*, *3*(1), 122–130. Available from http://ieeexplore.ieee.org/document/105425/

Ryan, A., & Hedrick, J. (2005). A mode-switching path planner for UAV-assisted search and rescue. In *Proceedings of the 44th IEEE Conference on Decision and Control* (pp. 1471–1476). IEEE. Available from http://ieeexplore.ieee.org/document/1582366/

Scott, Jeff. (2005). *Drag of Cylinder and Cones AerospaceWeb.* Available from http://www.aerospaceweb.org/question/aerodynamics/q0231.shtml (Website. Accessed Oct. 1 2017)

Shim, D., Kim, H. J., & Sastry, S. (2003). Decentralized nonlinear model predictive control of multiple flying robots. In *42nd IEEE International Conference on Decision and Control* (Vol. 4, pp. 3621–3626). Mau: IEEE. Available from http://ieeexplore.ieee.org/document/1271710/

Sreenath, K., & Kumar, V. (2013, jun). Dynamics, Control and Planning for Cooperative Manipulation of Payloads Suspended by Cables from Multiple Quadrotor Robots. In *Robotics: Science and Systems IX* (Vol. 9, pp. 661–664). Berlin: RSS Foundation. Available from http://www.roboticsproceedings.org/rss09/p11.html

Sreenath, K., Michael, N., & Kumar, V. (2013). Trajectory generation and control of a quadrotor with a cable-suspended load-A differentially-flat hybrid system. In *Proceedings - IEEE International Conference on Robotics and Automation* (pp. 4888–4895). Karlsruhe: IEEE. Available from `http://ieeexplore.ieee.org/document/6631275/`

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction* (Vol. 1) (No. 1). Cambridge: MIT Press.

The Economist. (2012, December). *An internet of airborne things.* Available from `https://www.economist.com/news/technology-quarterly/21567193-networking-enthusiasts-dream-building-drone-powered-internet-carry-objects` (Technology Quarterly. Accessed Jul. 04 2017)

Tobias, N., Alonso-mora, J., Domahidi, A., Rus, D., & Hilliges, O. (2017). Real-time Motion Planning for Aerial Videography with Dynamic Obstacle Avoidance and Viewpoint Optimization. *IEEE Robotics and Automation Letters*, *2*(3), 1696 – 1703.

Trachte, J., Gonzalez, F., & McFadyen, A. (2014). Nonlinear model predictive control for a multi-rotor with heavy slung load. In *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings* (pp. 1105–1110). Orlando: IEEE. Available from `http://ieeexplore.ieee.org/document/6842363/`

Trachte, J., Toro, L. F. G., & McFadyen, A. (2015). Multi-rotor with suspended load: System Dynamics and Control Toolbox. In *IEEE Aerospace Conference* (pp. 1–9). Big Sky: IEEE. Available from `http://ieeexplore.ieee.org/document/7119210/`

Tzes, A., Nikolakopoulos, G., & Alexis, K. (2012). Model predictive quadrotor control: attitude, altitude and position experimental studies. *IET Control Theory & Applications*, *6*(12), 1812–1827. Available from `http://ieeexplore.ieee.org/document/6397107/`

Vanderbei, R. J. (2012). *Interior Point Methods and Nonlinear Optimization.* La Palma: Princton. Available from `http://www.princeton.edu/~rvdb/tex/talks/MLSS_LaPalma/LaPalma3.pdf`

Wan, E. A., & Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. *Technology*, *v*, 153–158. Available from `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=882463`

Zanelli, A., Domahidi, A., Jerez, J., & Morari, M. (2017). FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, *0*(0), 1–17. Available from `https://doi.org/10.1080/00207179.2017.1316017`

Zheng, A., & Morari, M. (1995). Stability of model predictive control with mixed constraints. *IEEE Transactions on Automatic Control*, *40*(10), 1818–1823. Available from `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=467664`

Zürn, M., Morton, K., Heckmann, A., McFadyen, A., Notter, S., & Gonzalez, F. (2016). MPC controlled multirotor with suspended slung Load: System architecture and visual load detection. *IEEE Aerospace Conference Proceedings*, *2016-June*.