Online trajectory planning and control of a MAV payload system in dynamic environments

Potdar, Nikhil D.; de Croon, Guido C.H.E.; Alonso-Mora, Javier

DOI
10.1007/s10514-020-09919-8

Publication date
2020

Document Version
Final published version

Published in
Autonomous Robots

Citation (APA)
Potdar, N. D., de Croon, G. C. H. E., & Alonso-Mora, J. (2020). Online trajectory planning and control of a MAV payload system in dynamic environments. *Autonomous Robots*, *44*(6), 1065-1089. https://doi.org/10.1007/s10514-020-09919-8

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Autonomous Robots (2020) 44:1065–1089
https://doi.org/10.1007/s10514-020-09919-8

Online trajectory planning and control of a MAV payload system in dynamic environments

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

Received: 25 December 2018 / Accepted: 9 June 2020 / Published online: 24 June 2020

Abstract
Micro Aerial Vehicles (MAVs) can be used for aerial transportation in remote and urban spaces where portability can be exploited to reach previously inaccessible and inhospitable spaces. Current approaches for path planning of MAV swung payload system either compute conservative minimal-swing trajectories or pre-generate agile collision-free trajectories. 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 paper describes an online method for agile and closed-loop local trajectory planning and control that relies on Non-Linear Model Predictive Control and that addresses the mentioned limitations of contemporary approaches. We integrate the controller in a full system framework, and demonstrate the algorithm's effectiveness in simulation and in experimental studies. Results show the scalability and adaptability of our method to various dynamic setups with repeatable performance over several complex tasks that include flying through a narrow opening and avoiding moving humans.

Keywords Micro aerial vehicle · Collision avoidance · Trajectory optimization · Optimal control · MAV-payload system · MPC · Motion Planning in dynamic environments

1 Introduction

The small size, agility, and low upfront costs of Micro Aerial Vehicles (MAVs) could instigate their widespread use and rapid 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 (Ryan and Hedrick 2005), package/cargo delivery, and construction (Lee 2018) 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 they would operate. A MAVP system is required to be able to quickly, safely, and autonomously navigate an obstacle-ridden space while adapting to different situations. Carriage of a swinging payload vastly increases the system's spatial footprint making operation in restrictive spaces challenging. In such situations MAV trajectory planning and control is necessary to generate the desired swing motions to avoid collisions with obstacles. Failing to acknowledge the system's future response when performing agile flight could result in inevitable collisions as by the time an obstacle is to be avoided, the MAV might be unable to divert the swinging payload away. Working around the problem, one may pre-generate trajectories for fully defined environments (and thus static), or actively minimise swing to reduce the system's dynamic response, however, we will demonstrate that these undermine the real-world practicality of the approaches in dynamic environments.

A video of the experimental results is available at https://youtu.be/9C7O34W1w8Y.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10514-020-09919-8) contains supplementary material, which is available to authorized users.

✉ Nikhil D. Potdar
nikhil.potdar94@gmail.com

Guido C. H. E. de Croon
G.C.H.E.deCroon@tudelft.nl

Javier Alonso-Mora
j.alonsomora@tudelft.nl

[1] Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands

[2] The Micro Aerial Vehicle Lab, Delft University of Technology, 2629 HS Delft, The Netherlands

1.1 Contributions

Our main contribution is an online local motion planner and controller for the safe, agile, and collision-free flight of a MAVP system in dynamic, multi-obstacle settings. Our method is formulated as a constrained optimisation using a finite-horizon Non-linear Model Predictive Control (NMPC). The optimisation problem can be efficiently solved thanks to contemporary fast solvers.

A full system framework is outlined integrating the NMPC controller in a combined hardware and software based control loop. The proposed framework is verified and validated 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 it to state of the art methods and discuss the effect of computation time on the resulting system performance. Unlike previous works, we successfully demonstrate the safe operation and readiness of our method in realistic settings involving a MAVP system operating amongst multiple moving humans.

1.2 Related work

Historically, studies of aerial vehicle control with suspended payloads involved helicopter systems with applications to load transportation as shown in Cicolani and Kanning (1992), however, with the advent of MAVs the 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. We introduce contemporary approaches for both followed by a general discussion of NMPC control for MAV(P) systems and its application to closed-loop planning and control.

1.2.1 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 in-the-loop dynamic re-planning of trajectories.

Using pre-generated trajectories, planar and spatial tracking of MAVP trajectories has successfully been demonstrated through accurate modelling and stabilisation of the vehicle (Feng et al. 2014; Pizetta et al. 2015) sometimes utilising swing minimisation (Bisgaard et al. 2010; Palunko et al. 2012b; Trachte et al. 2014) to mitigate coupling disturbance effects. The latter approach is energetically inefficient and over-conservative as the vehicle devotes considerable control effort to reduce swing thus 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 which are then encoded as full state evolutions (Foehn et al. 2017; Palunko et al. 2012a) or a reduced dimension state using differential-flatness (Sreenath et al. 2013; Tang and Kumar 2015). In RL, as used in Palunko et al. (2013), Faust et al. (2013) and Faust et al. (2017), 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. Additionally, any trajectory infeasibility at run-time is catastrophic as the planning method is unable to accommodate for this. Therefore, the aforementioned studies only address fully known environments with static obstacles limiting the practical application of these methods to limited specialised demonstrative cases. In contrast, our method is able to rapidly re-plan even if the local trajectories intermittently become infeasible due to prevailing conditions.

1.2.2 Closed-loop MAVP trajectory planning

Motion planning in dynamic environments requires re-planning at run-time to accommodate for 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 Brock and Khatib (2000). Additionally, local planners need to cope with infeasibility during run-time while still conforming towards a higher global planning objective.

In De Crousaz et al. (2014), 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. In a similar fashion to Tang and Kumar (2015), De Crousaz et al. (2014) apply iLQG to demonstrate impressive manoeuvres which included the flight through a narrow opening. However, the study by De

Crousaz et al. (2014) did not consider planning in a dynamic environment. Furthermore, required rotor thrust inputs were generated at run-time which could saturate as the method did not account for vehicle constraints during planning. The consequence of saturating inputs over sustained time periods is the potential system instability and/or significant deviation of the actual to planned trajectory. In contrast, our approach takes into account the vehicle and environmental constraints to ensure the physical feasibility of the generated trajectories.

1.2.3 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 (Kim et al. 2002; Tzes et al. 2012) and MAVP (Trachte et al. 2014; Gonzalez et al. 2015) simple trajectory tracking. Focussing on the latter, in Trachte et al. (2014) NMPC for MAVP trajectory tracking control was addressed with a comparison to LQR control; the results demonstrated 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 Gonzalez et al. (2015), studies from Trachte et al. (2014) were extended to an experimental setup validating the results, however, unlike in De Crousaz et al. (2014), both studies only addressed the control aspect of tracking pre-generated trajectories. In contrast, our method unifies the online planning and control extending the capability beyond simple tracking of a pre-defined plan 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 (Neunert et al. 2016). With today's improved computing capabilities and fast optimisation solvers applications have been demonstrated for a MAV without swung payload (Neunert et al. 2016; Naegeli et al. 2017). The method of Neunert et al. (2016) utilises the Sequential Linear Quadratic (SLQ) algorithm to solve an unconstrained optimal control problem that is paired with an external high-level planner that pre-generates feasible waypoints accounting for only static obstacles. Therefore, their method is only deployable to MAVs in static environments for traversing a specific pre-generated set of waypoints.

Building upon the NMPC planning algorithm introduced by Naegeli et al. (2017), we utilise an interior-point based algorithm from Domahidi and Jerez (2014) within a NMPC setting to solve a constrained optimal control problem that is subject to the constraints on the system dynamics and environment. Introduction of constraints in our method allows us to perform real-time obstacle avoidance while still providing guarantees on the trajectory feasibility. Thus, in this work we demonstrate the viability of real-time NMPC based unified motion planning and control for MAVPs in dynamic, multi-obstacle settings.

1.3 Paper organisation

We introduce preliminaries in Sect. 2 with our notations and system models. In Sect. 3 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 Sect. 4. In Sects. 5 and 6 we discuss our findings followed by concluding remarks in Sect. 7.

2 Preliminaries

2.1 Notation

The following notations are observed; scalars x, vectors $\boldsymbol{x}$, 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 $\boldsymbol{R}_x \in SO(3)$.

2.2 Quadrotor with swung payload model

The system comprises 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 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 remaining constant. The following additional model assumptions are adopted;

– rigid, massless cable with free suspension points,

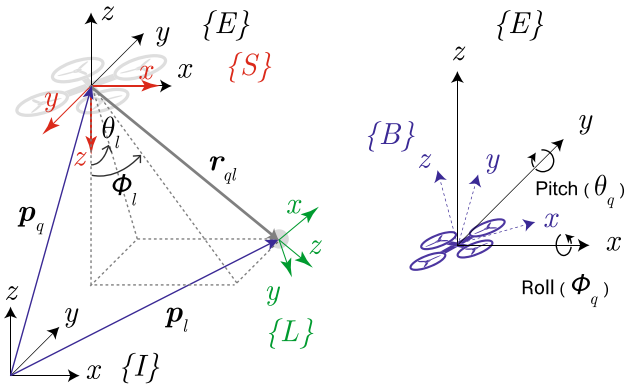


Fig. 1 Quadrotor-Payload system with the following references frames; $\{I\}$ inertial ENU, $\{E\}$ vehicle-carried ENU, $\{B\}$ vehicle body frame, $\{S\}$ is $\{E\}$ rotated by 180° about the $\{E\}$ x-axis and $\{L\}$ load frame with z-axis directed away from the cable's suspension point. Quadrotor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

– quadrotor centre of gravity, centroid, and cable suspension point coincide,
– no aerodynamic drag effects on the cable.

A non-rigid cable would introduce switching dynamics increasing the complexity of the system modelling. A preliminary study on the cable rigidity during agile manoeuvres, as shown in "Appendix A", shows the cable rarely slacks during flight allowing this assumption to be made. The second assumption regarding coincidence is made to simplify the model as the listed points tend to physically be in close proximity. Though the effects of the real point offsets on the system dynamics were not considered, for the purpose of performing predictions in the order of seconds, the effect was considered to be negligible. The assumption of no cable drag stems from the rationale that the string's exposed surface area to the flow is small resulting in negligible effects on the system dynamics.

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.

2.2.1 Quadrotor inputs

As in Klausen et al. (2015), 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 thrust command, 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 (radians), roll (radians), and thrust command (meters/second) 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 B". The input magnitude ranges are limited and hardware specific; these are accounted for in the design of the model predictive controller using constraints as further discussed in Sect. 3.3. We note that our method is not limited to the our chosen hardware and could easily be adapted to other quadrotors.

As the hardware-specific internal controller dynamics $\boldsymbol{u} \rightarrow \boldsymbol{F}_u$ are not documented, we empirically model the function. We define F_q as the purely vertical control force generated by the four rotors when commanded by input $\bar{w}_q$. The quadrotor's true pitch, roll response and the vertical control force F_q resulting from commanded inputs are given by

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

where, using the method presented in Stanculeanu and Borangiu (2011), $h_\theta,\ h_\phi,\ h_F$ are identified for the fast dynamics and decoupled as three linear second-order black-box models with model states and state transition

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

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 F_q (in $\{E\}$) generated by the motors based on the thrust command $\bar{w}_q$ (See "Appendix B"). Then similar to Naegeli et al. (2017), using outputs from (1) and based on equilibrium relations, the input control force is given by

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

where $m = m_q + m_l$ and $g = 9.81\,\text{m/s}^2$. See "Appendix C" for a derivation of this relation.

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

2.2.2 Aerodynamic drag effects

We derive the drag induced forces on the MAVP system; as in Derafa et al. (2006), assuming relatively low quadrotor velocities $\dot{\boldsymbol{p}}_q$ (up to 5 m/s) we model a proportional linear drag force on the quadrotor with drag constant k_{Dq} giving

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

As in Klausen et al. (2015), for the payload we only consider the rotational load motion relative to the quadrotor,

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

hence, its drag force is assumed to always be perpendicular to the moment arm (the rigid cable). The approximation introduces swing associated damping due to the relatively large rotational payload velocities. This allows the load's drag to be described in terms of the suspension angles and its derivative which are components of $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. We also avoid defining the drag in terms of load velocity as this term is not a variable available in $\dot{\boldsymbol{q}}$. Additionally, following from our free suspension point assumption, there are no payload drag induced reactive forces or moments on the quadrotor. We note that prolonged linear translation of the system would make the linear drag contribution to the load dynamics significant as the load would trail behind the quadrotor; this could be included in future studies. 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

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

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

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

2.2.3 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

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

Table 1 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 $\{I\}$
$\boldsymbol{x}_c, \boldsymbol{x}_q, \boldsymbol{x} \in \mathbb{R}^n$	Quadrotor input, system, and full MAVP model state

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

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

where $\boldsymbol{K} = \text{diag}(m_{q(1\times3)},\ m_{l(1\times3)})$ 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 $\boldsymbol{F} = [\boldsymbol{F}_u,\ 0,\ 0]^\top \in \mathbb{R}^5$, mass $\boldsymbol{M}$, drag $\boldsymbol{D}$ from (7), Coriolis $\boldsymbol{C}$ and gravitational $\boldsymbol{G}$ matrix terms. Equation (13) in its full form is presented in Klausen et al. (2015). Using (13), the system state and state transition are given by

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

2.2.4 Full MAVP model

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

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

Important MAVP model related variables and parameters that we often refer to are summarised in Table 1.

2.3 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 comparable size. Ellipsoids create smooth convex bounding volumes for (non-convex) obstacles making them appropriate for representing

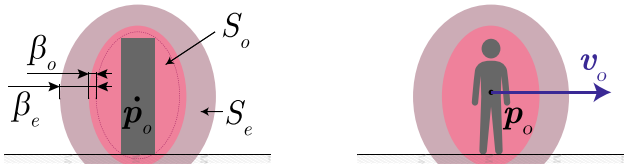


Fig. 2 Cuboid obstacle (*left*) with fixed position $\boldsymbol{p}_o$ or dynamic (human) obstacle (*right*) with constant velocity $\boldsymbol{v}_o$ each modelled by bounding S_o and expanded S_e ellipsoid with dimensional buffers β

objects including trees, humans and pillars. Additionally, computationally efficient collision checks against the ellipsoid's quadric exist (Uteshev and Goncharova 2018) making them favourable for real-time applications.

2.3.1 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 ellipsoid 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 identified as a high risk zone 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.

2.3.2 Obstacle motion prediction

Static obstacle positions are assumed to be readily available for planning. As in Naegeli et al. (2017), 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.

2.4 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 define mathematical requirements to guarantee a collision-free system.

2.4.1 Point to ellipsoid distance

The point to an ellipsoid signed distance is approximated as the true value cannot be expressed in closed form (Uteshev and Goncharova 2018). 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 $\Omega = \mathrm{diag}(1/(a_o + \beta)^2, 1/(b_o + \beta)^2, 1/(c_o + \beta)^2)$.

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.

2.4.2 Quadrotor and payload proximity

We model the quadrotor and payload individually 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

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

2.4.3 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 simplifying the check as a point to ellipsoid computation. Given the cable cross-section dimensions are negligible, no buffer is required so $\beta_o = 0$. Using (16), $\boldsymbol{p}_c^*$ is computed by

$$\boldsymbol{p}_c^* = \arg\min_{\boldsymbol{p}_c} \left(d_o(\boldsymbol{p}_c,\ S_o)\right) \tag{19}$$

with $\boldsymbol{p}_c \in \left\{\boldsymbol{p} \,|\, \boldsymbol{p} = \boldsymbol{p}_q + s(\boldsymbol{p}_l - \boldsymbol{p}_q), s \in [0, 1]\right\}$. "Appendix E" shows the problem (19) is expressible in closed-form and 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.

3 Online and closed-loop MAVP trajectory generation

3.1 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 for the dynamic environment, we perform dynamic and closed-loop local motion planning using NMPC which is a receding finite-horizon controller.

3.1.1 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 $\boldsymbol{x}_N$ thus giving

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

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

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

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 $\boldsymbol{x}_0$ are obtained. Subsequently, a local trajectory is re-generated by initialising the solver with a time-shifted version of the previous solution. The time-shift in simulation studies is a fixed simulation time step, and in real experiments the actual control loop time is used. This approach results in our method's computationally efficient closed-loop performance with robustness to model uncertainties (Naegeli et al. 2017); we illustrate this process in Fig. 3.

3.1.2 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

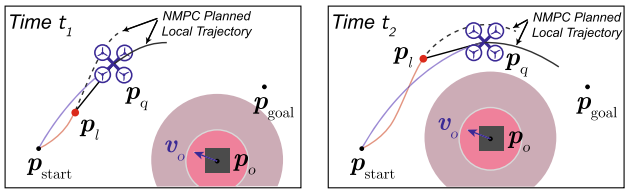


Fig. 3 System moves towards $\boldsymbol{p}_{\text{goal}}$ with $t_2 > t_1$; planned local trajectory (grey) at the current time (*left*) that is updated in a future time (*right*) with the new state and obstacle data. Schematic projected top view with illustrative obstacle ellipsoids shown

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

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 online environment 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 the variables of the terminal stage N. Terminal costs are used to achieve closed-loop stability of the finite-horizon planner (Mayne et al. 2000).

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 which is encoded in $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{u}}$ thus providing (feasibility) guarantees for the computed trajectory. To ensure 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 (Zheng and Morari 1995). We encode the slack variables associated with the trajectory in

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

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 (Zheng and Morari 1995).

During optimal trajectory generation we minimise (26) while 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 Sect. 3.4.

3.2 Costs

We introduce cost terms derived from our planning objectives presented in Sect. 3.1. We use our weighted square norm definition from Sect. 2.1 with an $n \times n$ identity matrix denoted by $\boldsymbol{I}_n$ to make all cost terms scalar and positive.

3.2.1 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

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

Making (28) a stage cost would mean the shortest path (straight line) is always preferred which may result in deadlock for cases where 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.

3.2.2 Potential field based obstacle separation

For obstacle separation, we employ a MAVP to obstacle proximity related cost analogous to a reactive potential field (Khatib 1986). We combine this with constraints to guarantee collision-free trajectories as is presented in Sect. 3.3.3. This two layered approach, similar to Naegeli et al. (2017), 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 position [see (19)]; for each object 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 separation from the obstacle. With this insight, and using (16), the cost is formalised as

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

3.2.3 Input magnitude regulation

The input magnitude associated cost is given by

$$c_{\text{in}} = \left\| \boldsymbol{u}^\top \right\|_{\boldsymbol{I}_3}. \tag{30}$$

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

3.2.4 Payload suspension angles regulation

The suspension angle associated cost is given by

$$c_{\text{swing}} = \left\| [\theta_l, \phi_l]^\top \right\|_{\boldsymbol{I}_2}. \tag{31}$$

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

3.3 Constraints

We derive planning constraints from our system and setup limits in addition to the global planning objectives.

3.3.1 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

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{32}$$

where k is the stage index.

3.3.2 State and input limits

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

$$\mathcal{X}_{\min} \leq \boldsymbol{x} \leq \mathcal{X}_{\max} \tag{33}$$

$$\mathcal{U}_{\min} \leq \boldsymbol{u} \leq \mathcal{U}_{\max}. \tag{34}$$

We specify the hardware-specific limits in Section 4.

3.3.3 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 position [see (19)]; for each we define a constraint. Adopting the requirements (17-18, 20)

for a collision-free MAVP system as presented in Sect. 2.4, and using (16), the associated constraint is formalised as

$$d_o(\boldsymbol{p}, S_o) + s_c > 0 \tag{35}$$

with the non-negative scalar slack s_c. Note that even though (35) is defined for every obstacle, only one slack s_c is defined and used in all obstacle associated inequality constraints per timestep in our optimisation problem definition. When any $d_o(\boldsymbol{p}, S_o)$ becomes negative, s_c assumes the highest value such that all obstacle associated inequality constraints are satisfied according to (35) for a specific timestep. Ideally s_c is zero hence the system is collision-free with respect to all obstacles. Provided a sufficiently high penalty cost associated with the slack, the feasible solution is recovered as shown in Kerrigan and Maciejowski (2000). Summarising, only one slack is necessary to differentiate between a system in collision or collision-free state with no necessary differentiation with respect to which obstacle that occurs.

3.3.4 Workspace limits

For confined (indoor) operation, the quadrotor and load position is limited to the workspace limits. Assume a cuboid workspace, then let $\mathcal{W}_{\min}$, $\mathcal{W}_{\max}$ 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

$$\boldsymbol{p}_q + \mathbf{1}_3 s_q \geq \mathcal{W}_{\min} \quad \text{and} \quad \boldsymbol{p}_q - \mathbf{1}_3 s_q \leq \mathcal{W}_{\max} \tag{36}$$

$$\boldsymbol{p}_l + \mathbf{1}_3 s_l \geq \mathcal{W}_{\min} \quad \text{and} \quad \boldsymbol{p}_l - \mathbf{1}_3 s_l \leq \mathcal{W}_{\max} \tag{37}$$

with $\mathbf{1}_3 = [1, 1, 1]^\top$ 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.

3.3.5 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.

3.4 Optimisation algorithm

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

3.4.1 Costs

We define the stage and terminal cost functions based on the cost term definitions (28–29, 65, 30-31). Let w denote a user-definable weighting used to assign relative importance to costs and their associated objective, then the stage cost function, as introduced in (24), is defined as

$$\begin{aligned} c_s(\boldsymbol{x}_k, \boldsymbol{u}_k, *_k) = w_{\text{in}} c_{\text{in}} + w_{\text{swing}} c_{\text{swing}} \\ + \boldsymbol{w}_{\text{pf}}^\top \boldsymbol{c}_{\text{pf}} + \boldsymbol{w}_{\text{slack}}^\top \boldsymbol{s}, k \in [0, N-1] \end{aligned} \tag{38}$$

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 where $\boldsymbol{s} = \left[s_c, s_q, s_l\right] \in \mathbb{R}^3$. The terminal cost function, as introduced in (25) is defined as

$$\begin{aligned} c_t(\boldsymbol{x}_N, *_N) = w_{\text{nav}} c_{\text{nav}} + w_{\text{in}} c_{\text{in}} + w_{\text{swing}} c_{\text{swing}} \\ + \boldsymbol{w}_{\text{pf}}^\top \boldsymbol{c}_{\text{pf}} + \boldsymbol{w}_{\text{slack}}^\top \boldsymbol{s}. \end{aligned} \tag{39}$$

3.4.2 Constraints

We impose the system dynamics, and state/input constraints, as introduced in Sect. 3.3, on the optimiser. By function $\boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, *_k)$ we denote all additional inequality constraints as defined in (35-37) which are functions of state, input and additional (time specific) variables/parameters, e.g., obstacle positions and workspace limits.

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

$$\begin{aligned} \tilde{\boldsymbol{z}} &= \left[\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N, \boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}, \boldsymbol{s}_0, \ldots, \boldsymbol{s}_N\right] \\ &\equiv \left[\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{s}}\right]. \end{aligned} \tag{40}$$

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

$$
\begin{aligned}
\min_{\tilde{z}} \quad & J = c_t(\boldsymbol{x}_N, *_N) + \Sigma_{k=0}^{N-1} c_s(\boldsymbol{x}_k, \boldsymbol{u}_k, *_k) \\
\text{s.t.} \quad & \boldsymbol{x}_0 = \boldsymbol{x}(t) && \text{(Initial Estimated State)} \\
& \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) && \text{(Discretised Dynamics)} \\
& g(\boldsymbol{x}_k, \boldsymbol{u}_k, *_k) \geq 0 && \text{(Inequality Constraints)} \\
& \boldsymbol{x}_k \in \mathcal{X} && \text{(State Constraints)} \\
& \boldsymbol{u}_k \in \mathcal{U} && \text{(Input Constraints)} \\
& \boldsymbol{s}_k \geq 0 && \text{(Slack Constraints).}
\end{aligned} \tag{41}
$$

3.5 Theoretical analysis

3.5.1 Problem dimensionality

Variable $\tilde{z}$ is optimised at every planning time instance encoding the optimised local trajectory in its solution. As given by (40), $\tilde{z}$ 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 $\tilde{z} \in \mathbb{R}^{22N+19}$.

3.5.2 Optimality and feasibility

We use a fast non-linear programming based optimiser, namely FORCES Pro (Zanelli et al. 2017), on our non-convex optimal control problem. FORCES Pro implements a primal-dual interior-point constrained optimisation algorithm which is outlined in "Appendix F". 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. When this occurs, our planner holds the MAVP at this locally optimal state until a solution becomes available as a result of the changing environment or external perturbation to the system.

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 Mayne et al. (2000).

4 System setup and framework

We outline our particular implementation of the system model and NMPC controller for simulated and experimental studies, including a state estimator.

4.1 System properties and hardware

The MAVP system properties used for all studies are given in Table 2. The maximum pitch, roll and thrust command inputs are derived from quadrotor's system limits as used in this study. The MAVP hardware is shown in Fig. 4.

Table 2 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}	0.00177
Cable length	0.77 m	Max. $\lvert\bar{\theta}_q\rvert$, $\lvert\bar{\phi}_q\rvert$ input	15°
Max. $\lvert\bar{w}_q\rvert$ input	1 m/s	Detection range	3.5 m

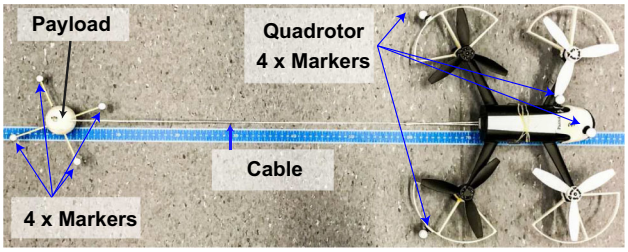


Fig. 4 Parrot Bebop 2 quadrotor with cable suspended load and attached tracking markers

4.2 Workspace

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

4.3 Programmed control system framework

The control framework is schematised in Fig. 5; in "Appendix B" the on-board control component is expanded upon. The off-board components run on an Intel i7-3610QM Quad Core 3.3GHz processor PC, and is programmed in MATLAB with an efficient C language solver FORCES Pro performing the online NMPC computations (Zanelli et al. 2017). 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 based network. As it is shown in Sect. 5, the controller loop time was on median in the order of 10^{-2} s, therefore, for simulation and controller design, explicit Runge–Kutta 2$^{\text{nd}}$ order integration of the system model is used. Runge–Kutta 2$^{\text{nd}}$ was chosen as a preliminary examination of the predicted system response showed it provided a good balance between loop time performance and dynamics approximation. An in-depth comparison of

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

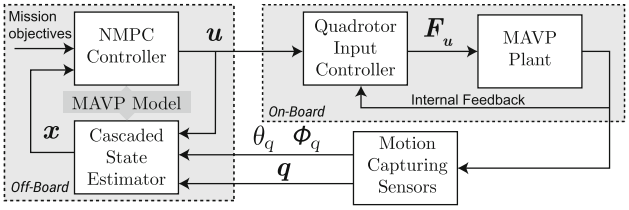


Fig. 5 Control system framework with the off-board NMPC controller and state estimator sharing a MAVP model, and the on-board MAVP system. Quadrotor input controller performs closed-loop low-level control. External motion capturing produces measurements necessary for state estimation

Table 3 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
Slacks w_{slack}	10000		

other discretisation methods is beyond the scope of this paper and should be investigate in future studies. The implemented NMPC cost weights are given in Table 3.

4.4 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 $SE(3)$ (Point 2009). 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 Sect. 2.2 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 Bisgaard et al. (2007), 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 Sect. 2.2.3. 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 (Julier and Uhlmann 1997). The full cascaded KF based estimator design is schematised in Fig. 6.

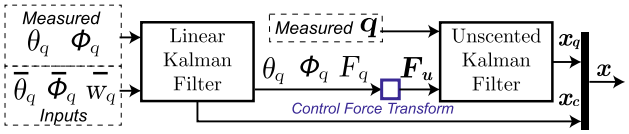


Fig. 6 Cascaded state estimation using Linear and Unscented KF and measured MCS data. Linear KF estimates $\boldsymbol{x}_c$ based on the quadrotor input model, measured true pitch, roll and inputs. Unscented KF estimates $\boldsymbol{x}_q$ based on MAVP model, measured MAVP configuration $\boldsymbol{q}$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

5 Simulation study

We showcase our method's scalability, robustness and performance in simulated studies. This is presented as a precursor towards our experimental results involving flight amongst multiple human obstacles and acrobatic manoeuvers.

The following metrics are used; let the system's *distance-to-goal* be defined as

$$d_{\text{goal}} = \left| \boldsymbol{p}_q - \boldsymbol{p}_{\text{goal}} \right|, \tag{42}$$

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

5.1 Scalability of the optimisation problem

The scalability of the optimisation problem is studied against the number of planning stages and separately the number of obstacles.

5.1.1 Scaling with number of planning stages

The quadrotor, with a randomised initial swing θ_l, $\phi_l < 10°$ 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 we perform 16 runs per case.

Results in Fig. 7 show the scaling of the NMPC solve time with N; it shows a positive correlation which is expected as the optimisation variable, given by $\tilde{z} \in \mathbb{R}^{22N+19}$, increases the problem dimension with a larger N. As shown in Fig. 8a where N=10 the system responds late to the incoming obstacle leading to a near-miss or collision (physical violation). The late response means the attempted aggressive evasive behaviour causes the system to move far off-track, sometimes

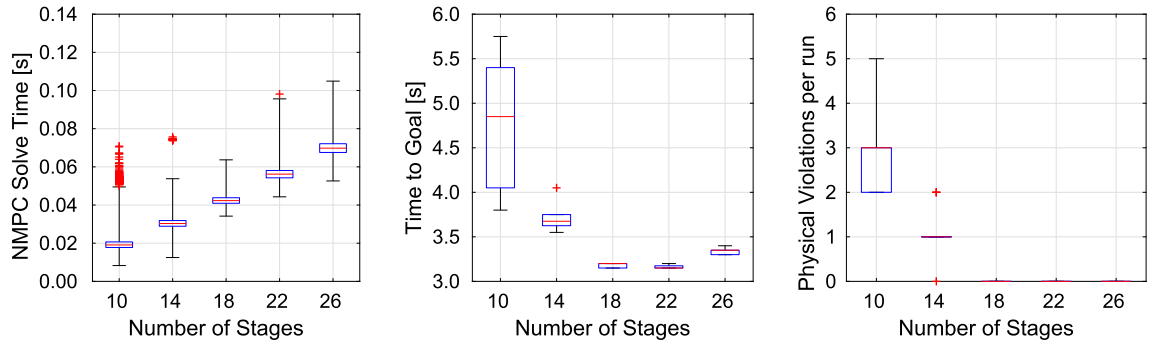
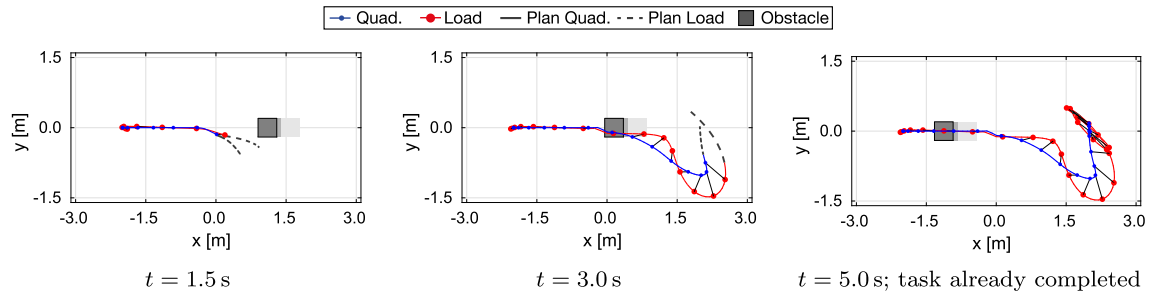
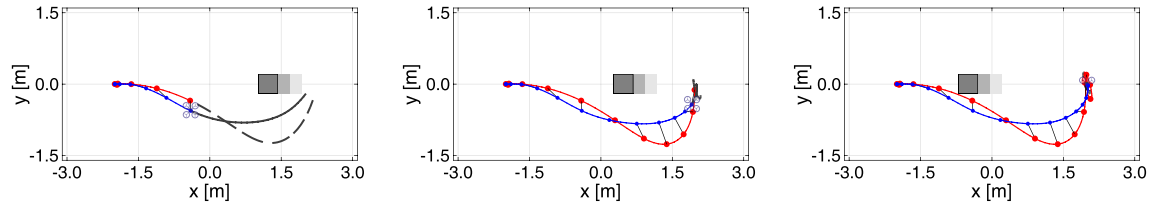


Fig. 7 Simulated NMPC solve time, time-to-goal and physical violations (collisions or breach of workspace limits) per run with increasing planning stages and 16 runs per case



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

(b) $N = 10$; the system is late to respond to the oncoming obstacle due to the short prediction horizon.



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

(b) $N = 26$; the long horizon pre-emptively guides the system away from the incoming obstacle's path.

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

leading to workspace limit violations, and overall increasing the time-to-goal. As depicted in Fig. 8b, with a higher N the collisions are averted and the system responds in a smooth agile motion. However, with $N = 26$, the planner favours a greater MAVP to obstacle separation over each generated trajectory resulting in a lower potential field associated cost of the objective function. As a result of the greater separation, the total distance covered from the start to goal is greater thus increasing time-to-goal. Important to note is that the generated planning remains tunable through modification of the cost function weights resulting in different system behaviours.

Based on the results and qualitative observations, $N = 18$ was used for all subsequent studies as it balances run-time and planning performance. To address the pitfalls of using a low N, a novel assistive steering approach can be used that guides the planned trajectory away from obstacles even when the obstacle is not within the prediction horizon; this method is outlined in "Appendix G". The results and benefits of assistive steering is discussed in "Appendix H". For short prediction horizons which are low N, the assistive steering aids the planning performance for obstacle avoidance. For N higher than 14 and the system speeds achieved in our experiments, the planner's lengthened predictive horizon makes the assistive steering redundant as no benefit are apparent. Therefore, in the experiments discussed in this paper, we do not employ assistive steering.

5.1.2 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

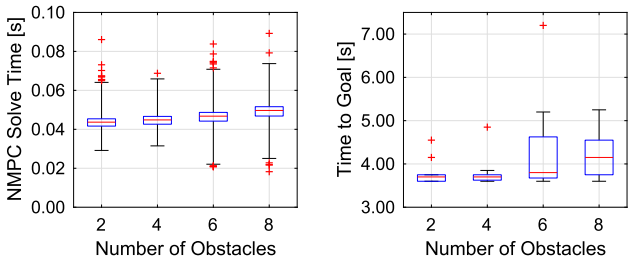


Fig. 9 Simulated NMPC solve time and time-to-goal with increasing number of dynamic obstacles using $N = 18$ and 16 runs per case. No violations/collisions occurred. Outlier for time-to-goal at 6 obstacles is for a run that temporarily entered deadlock resulting in a longer path

randomised velocities $\leq$1 m/s. We increase n_o from 2 by 2 to 8 with $\Delta t = 0.05$ s, N=18, default cost weights 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 most direct path to the goal thus requiring a re-route resulting in a lengthier route. In Fig. 10 we show one run demonstrating the MAVP's agile response 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 limiting characteristic of local planning algorithms (LaValle 2011). However, a benefit of our local planning method is that the system holds its position during deadlock and continues identifying solutions in the evolving solution space (due to dynamic obstacles). Therefore, in most cases it is capable of self-resolving the deadlock given sufficient time.

5.2 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 $\Delta t = 0.05$ and perform 4 repeated runs. Table 4 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

Table 4 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

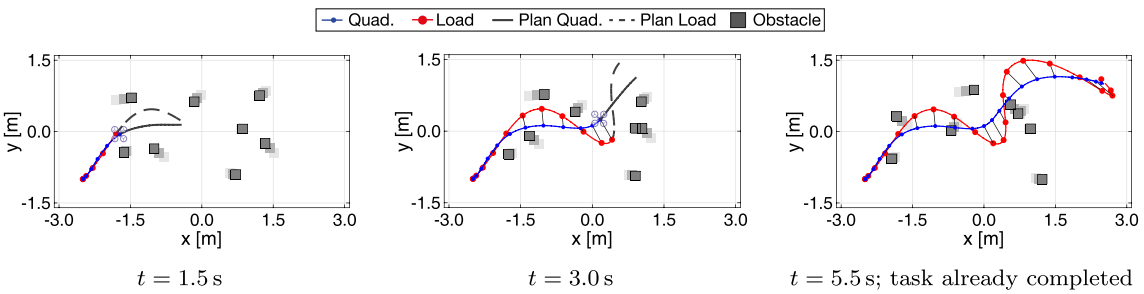


Fig. 10 Point-to-point navigation using N=18 amongst 8 randomised dynamic obstacles moving at $\leq$ 1 m/s. The dynamically planned and agile executed trajectories of the quadrotor and payload are shown with the current quadrotor position indicated. Top down view

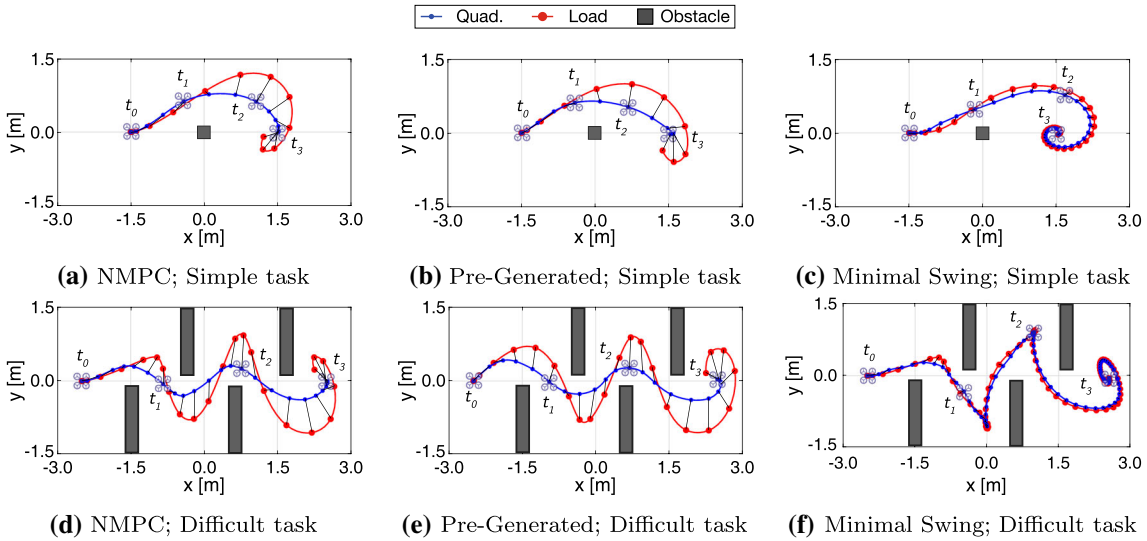


Fig. 11 Comparison of executed trajectories for manoeuvring around an obstacle (simple) and completing a slalom course (difficult) using NMPC with N=18, pre-generated and minimal swing planning and control. Observe that pre-generation leads to the smoothest, most optimal path resulting from the global 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

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 reconfiguration. With increasing task complexity and duration, greater reductions can be realised making NMPC's scalability unparalleled. Furthermore, our NMPC method applies to dynamic scenarios.

5.3 Robustness to change in control time step and lags

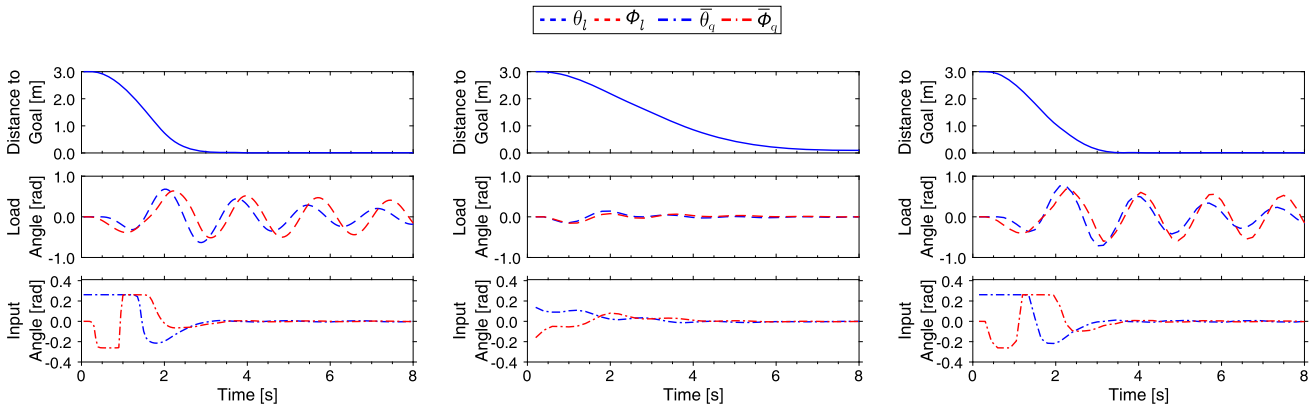
We demonstrate the robustness of our method by (i) increasing Δt from 0.05 s to 0.20 s to simulate a slower NMPC controller (on a less-powerful computer), and (ii) artificially adding a 0.1 s lag between NMPC generated input commands and actually executing them. We use the simple task from Sect. 5.2 for analysis and $N = 18$ and default cost weights. Given that N is a pre-configured design value of the MPC controller, and that Δt depends on the real loop-time, this study aims to show the affects of different Δt resulting from the use of hardware with varying computationally capability. Therefore, it is important to note that the lookahead horizon $N\Delta t$ is different in each case.

Comparing Fig. 12a, b 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.

In the simulation study we were able to select Δt for a pre-configured N and show its effects on the system dynamics approximation and system performance. In the experimental



(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

setup the Δt is equal to the real control loop-time which positively correlates to N as shown in Sect. 5.1. Therefore, in experimental studies it is important to tune N such that the obtained Δt is compatible with the discretisation method used on the system dynamics.

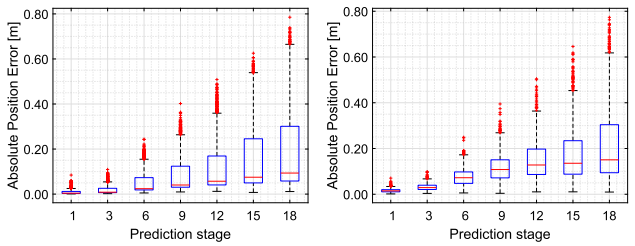
6 Experimental study

We showcase complex, agile behaviour in static and dynamic experimental setups. First we analyse the effects of prediction accuracy on our planner's performance as it is integral to our discussion of the experimental results. The same distance/time-to-goal definitions as introduced in Sect. 5 are used. Videos of the experiments performed are at https://youtu.be/9C7O34W1w8Y.

6.1 Planning prediction accuracy over prediction horizon

To demonstrate the effect of our local planner's prediction accuracy on system performance, we replicated the simulation as described in Sect. 5.1.1 in a real-world setup. Six identical runs were performed with the NMPC planner configured to use the real time-step, $N = 18$ and default cost weights.

As our method is online and local, closed-loop trajectory planning is achieved every cycle by using the estimated system and dynamic obstacle states, and a plant model for propagating these states over N planning stages. For each generated local trajectory $\tilde{\boldsymbol{x}}$, we compute the absolute error between the predicted and future true measured quadrotor/payload position for all N stages. With Δt our time-step,



(a) Quadrotor position **(b)** Payload position

Fig. 13 Absolute position error for the 18 predicted stages of quadrotor and payload position when compared to measured true values for a point-to-point navigation task with a collinear dynamic obstacle

for the k-th stage we compare the predicted position to the true measurements obtained for the system $k\Delta t$ in the future.

As shown in Fig. 13, our planner's prediction error of the quadrotor and payload position are in the order of magnitude 10^{-2} m up to 3 stages. As stated in Sect. 1, each local plan must remain feasible during execution for a full computation cycle, therefore, it is important that these first stages are accurately predicted. During one computation cycle of Δt, the system translates to the first predicted state of our plan before we re-plan, therefore, it is critical that this prediction is accurate to guarantee feasibility during execution. We demonstrate with metrics in Fig. 13 that our planner performs as desired for this critical prediction horizon.

Furthermore, as NMPC utilises plant model propagation over the N stage prediction we observe that the accumulation of inter-stage errors partially contributes to the reduction of prediction accuracy at the higher stage counts. Furthermore, with our local planning approach in a dynamic environment, planned trajectories can significantly differ from cycle-to-cycle as new routes become feasible at run-time. This can

skew prediction error results as a predicted trajectory may never actually be executed. These two factors results in an increase of absolute position error with number of prediction stages as seen in Fig. 13. We refer to this prediction accuracy analysis in the discussion of our experimental results to support explanation of observed system behaviour.

6.2 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 De Crousaz et al. (2014) and Tang and Kumar (2015) , 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, $N = 18$ and default cost weights. For the narrow opening, the maximum pitch/roll input is increased to 20°.

In Fig. 14 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 the rare cases when the manoeuvre was not successful, the system would either end up in a deadlock in front of the obstacle or make a momentary contact with the obstacle. Flight was recoverable following the contact with only four tests where this was not the case. For our unsuccessful tests, the primary cause was traced to our local planning approach or inaccuracies in the model resulting in the discrepancy between the observed and planned motion.

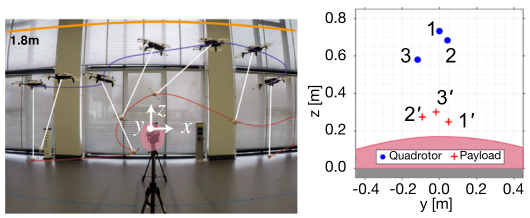
As addressed in Sect. 3.5, local planning method such as ours are prone to deadlocks. In this particular case, one reason could be an insufficient planning horizon necessary to compute a feasible trajectory over/through the obstacle. Furthermore, our NMPC optimisation algorithm utilises the time-shifted previous solution as an initial guess for the optimisation problem thus reduce computation time, however, this strategy contributes to an increased likelihood of deadlocks. The latter can be mitigated at the cost of few longer computation cycles by randomising the initial solution when deadlock situations arise. An automated and structural method to address these situations within our local planning framework is an interesting topic to be addressed in future work.

The cases involving obstacle contact were only observed in the flight through an opening experiment as the margins of error were small. Our method enables us to mitigate the effects of long horizon planning inaccuracies by using observed system states to re-plan rapidly. However, in this case, the prediction accuracy over longer horizons is critical to ensure the planner only initiates an agile manoeuvre that is feasible over the full planned trajectory. In the rare case the planner experiences an intermittent infeasibility during runtime, such as system-obstacle contact, it will quickly recover stable controlled flight once feasible trajectories become available.

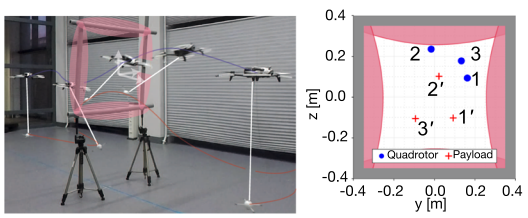
The setup was extended to the case of a moving high bar manoeuvre for agile dynamic obstacle avoidance (see video).

6.3 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

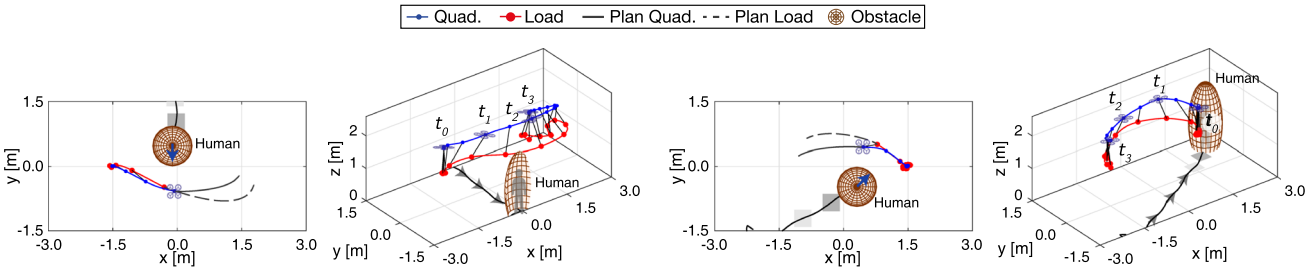


(a) Flight over high bar with virtual ceiling (orange line)



(b) Flight through narrow 0.7×0.7 m opening

Fig. 14 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. 15 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)

ellipsoids with buffers $\beta_o = 0.2$ m, $\beta_e = 1.0$ m, and are tracked to estimate their velocities for planning. The NMPC uses the real time-step, default cost weights and $N = 18$. 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.

6.3.1 One human with crossing paths

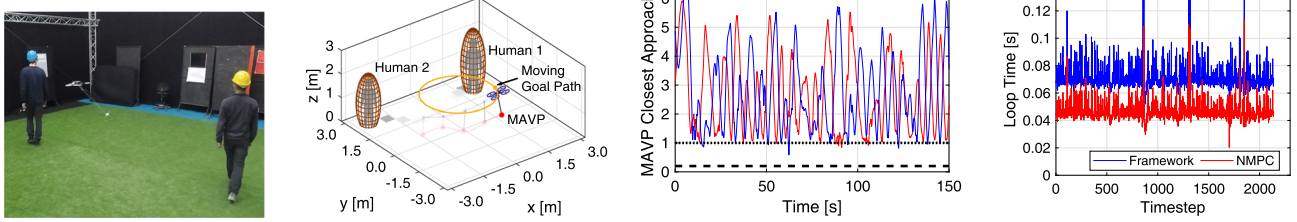
A human walks perpendicularly and diagonally on a path crossing the MAV performing a navigation task from $(-1.5, 0, 1.9)$ to $(1.5, 0, 1.9)$.

In Fig. 15 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.

6.3.2 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 7 s period along a circle with radius 1.5 m and a constant 1.4 m height. Experiment was conducted in a larger workspace measuring 3.2×3.2 m.

In Fig. 16 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. 16c, a safe distance is maintained by the MAVP from the humans with no collisions over the entire run; the minimum observed separation was 0.38 m from to the human's collision avoidance limit. In some cases the humans were apprehensive about the MAVP getting too close so they would perform a precautionary evasive motion, however, as shown in Fig. 16c, the MAVP still always maintains a safe separation. To address



(a) Snapshot from experiment **(b)** Schematic of setup with MAVP, human obstacles and moving goal **(c)** Separation between MAVP and ellipsoids representing humans **(d)** Framework loop time includes the NMPC solve times

Fig. 16 MAVP follows a circular path avoiding two randomly walking human obstacles modelled as ellipsoids moving at a mean 0.78 m/s with max. 1.45 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 71 ms) which includes the NMPC controller (mean 46 ms); brief spiking arises from situations where significant re-planning was required

observable closeness, it is possible to enlarge the obstacle associated buffers to increase the separation.

Observe from Fig. 16d that the NMPC solve time resembles the statistics obtained from the simulated study with two 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 compute 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 25 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 agile and safe continuous manoeuvres and avoid dynamic obstacles such as humans. Our method is extendable to larger spaces with more humans/obstacles as we have already demonstrated in simulation with eight obstacles.

7 Conclusion

In this paper, we presented an optimisation based unified motion planner and controller to accomplish online, closed-loop and agile flight of a Micro Aerial Vehicle slung payload system. We formulated the optimisation objective function, constraints and relied on a state of the NMPC solver to achieve 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.

Acknowledgements This work has been supported in part by Veni project 15916 of the Netherlands Organisation for Scientific Research (NWO), domain Applied and Engineering Sciences (TTW), the Amsterdam Institute for Advanced Metropolitan Solutions and ONRG-NICOP-grant N62909-19-1-2027.

A Analysis of cable slackening during maneuvers

A preliminary experimental study was performed to identify whether the cable slackens during flight maneuvers that were foreseen to be part of this study. The position of the cable suspension point and attachment to the load were recorded then the difference was used to compute the cable length. A flight was performed maintaining a close to constant altitude while combining (aggressive) pitch and roll inputs. The maximum pitch and roll angle was capped at 20° which is the limit set by the internal controller on the quadrotor. Fig. 17 shows the computed cable length, suspension angles and inputs to the system for a section of the flight.

As shown, the cable length varies by 0.01 m during the run, about 1.3% of the nominal value of 0.755 m. If slackening had occurred then deep troughs would have been visible in the cable length data, however, this was not the case therefore assuming the cable remains taut is a reasonable assumption. Slackening is likely when the vehicle suddenly accelerates in various direction not allowing sufficient time for the payload to respond, however, given the system performance limits this was not realised.

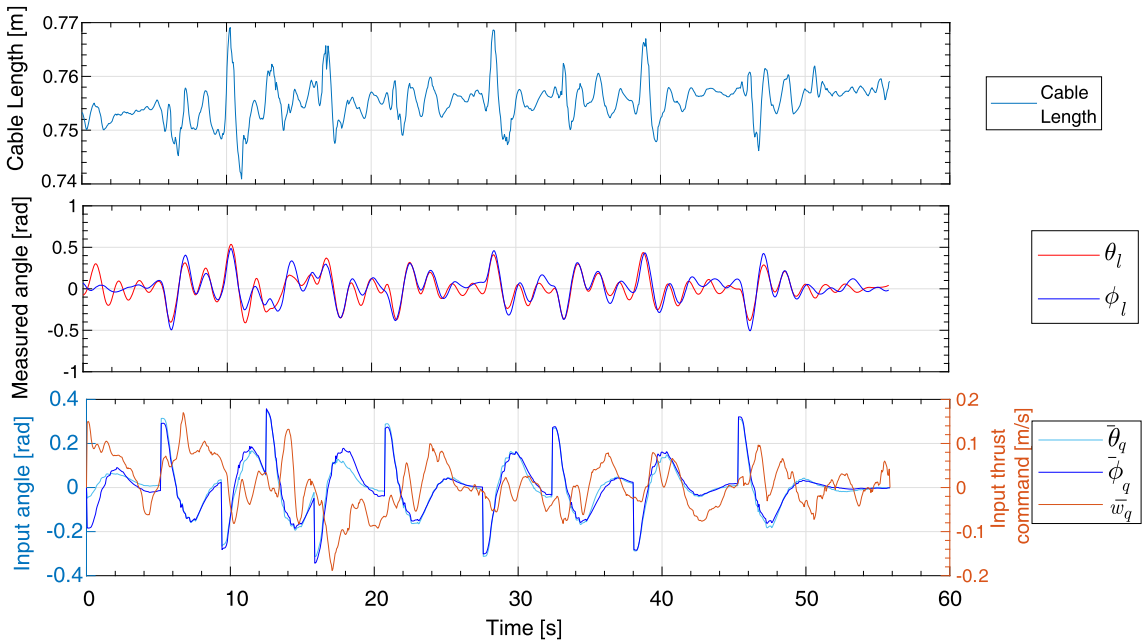


Fig. 17 Load suspension cable length, suspension angles and MAV commanded pitch, roll and thrust command inputs over an aggressively maneuvered flight section. Nominal cable length is 0.755 m when system is at equilibrium

B Quadrotor on-board controller

Figure 18 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 thrust command input $\bar{w}_q$ resulting in vertical control force F_q which when trimmed for the system weight gives F_z. All forces F are in the world East-North-Up inertial frame.

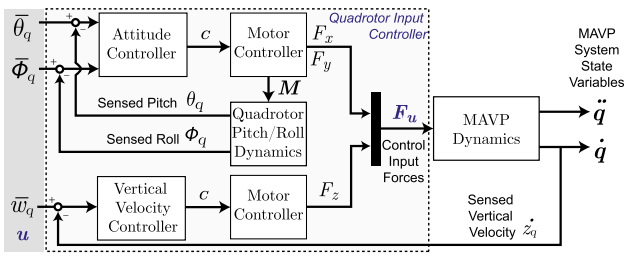


Fig. 18 On-board quadrotor input controller showing inner attitude and vertical velocity stabiliser loops. Motor inputs c are generated by controllers, resulting in forces F and moments M. Internally stabilised quadrotor attitude is perturbed by M; control force $\boldsymbol{F_u}$ affects MAVP dynamics. Sensed measures are internally fed-back on-board the quadrotor

C Input control force derivation

For modeling the input control force relations the premise is that the quadrotor is able to perform horizontal maneuvers while maintaining altitude, hence, the total vertical thrust component must always counteract the system weight when not changing elevation. The assumption made is that the horizontal and vertical input force components are decoupled. We also assume elevation changes only occur at negligible roll and pitch angles, so the system is in or near equilibrium. Let T be the total thrust force generated by the four rotors, $\boldsymbol{R}_E^B$ define the quadrotor's orientation and m the system mass, then the generated thrust force vector in frame $\{E\}$ is defined as

$$\boldsymbol{F_u} = \boldsymbol{R}_E^B\,[0,\ 0,\ T]^\top . \tag{43}$$

Additionally, as altitude is maintained then

$$\boldsymbol{F_u}^\top\,[0, 0,\ 1]^\top - mg = 0. \tag{44}$$

Solving for T in (44) gives,

$$T = \frac{mg}{cos(\phi)cos(\theta)}. \tag{45}$$

As we assume elevation changes only occur from a (near) equilibrium state, the vertical control input F_q directly controls the generated vertical thrust force component. Then using (44) and (45), the thrust force vector is defined as

$$\boldsymbol{F}_u = \left[m\frac{\tan(\theta_q)}{\cos(\phi_q)}g, \ -m\tan(\phi_q)g, \ F_q + mg \right] \in \mathbb{R}^3. \quad (46)$$

D 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 thrust command $\bar{w}_q$ are identified using the MATLAB system identification toolbox. The linear second-order, state-space, black-box models, which we denote by h_θ, h_ϕ, h_F, are given by equations (47) (48) and (49) respectively;

$$\begin{aligned}
\dot{\boldsymbol{x}}_{\boldsymbol{\theta}} &= \begin{bmatrix} -4.301 & -2.877 \\ 10.92 & -10.37 \end{bmatrix} \boldsymbol{x}_{\boldsymbol{\theta}} + \begin{bmatrix} -0.6893 \\ -16.32 \end{bmatrix} \bar{\theta}_q \\
\theta_q &= \begin{bmatrix} 1.763 & 4.586 \times 10^{-3} \end{bmatrix} \boldsymbol{x}_\theta + \begin{bmatrix} 0 \end{bmatrix} \bar{\theta}_q \qquad (47) \\
\dot{\boldsymbol{x}}_{\boldsymbol{\phi}} &= \begin{bmatrix} -2.789 & -4.978 \\ 9.302 & -13.72 \end{bmatrix} \boldsymbol{x}_{\boldsymbol{\phi}} + \begin{bmatrix} -5.41 \\ -18.04 \end{bmatrix} \bar{\phi}_q \\
\phi_q &= \begin{bmatrix} 1.996 & 0.4657 \end{bmatrix} \boldsymbol{x}_{\boldsymbol{\phi}} + \begin{bmatrix} 0 \end{bmatrix} \bar{\phi}_q \qquad (48) \\
\dot{\boldsymbol{x}}_{\boldsymbol{F}} &= \begin{bmatrix} -6.767 & -6.546 \\ 3.031 & 0.311 \end{bmatrix} \boldsymbol{x}_{\boldsymbol{F}} + \begin{bmatrix} 38.75 \\ 1.841 \end{bmatrix} \bar{w}_q \\
F_q &= \begin{bmatrix} 0.310 & 2.03 \times 10^{-2} \end{bmatrix} \boldsymbol{x}_{\boldsymbol{F}} + \begin{bmatrix} -0.121 \end{bmatrix} \bar{w}_q. \qquad (49)
\end{aligned}$$

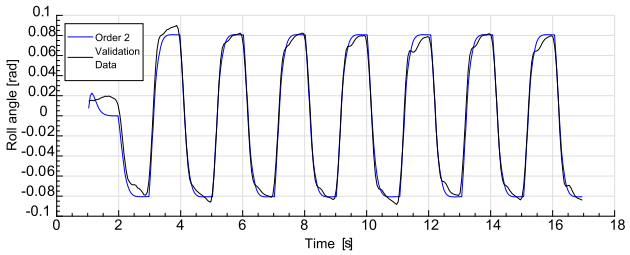
The states $\boldsymbol{x}_{\boldsymbol{\theta}} = [x_{\theta,1}, x_{\theta,2}]$, $\boldsymbol{x}_{\boldsymbol{\phi}} = [x_{\phi,1}, x_{\phi,2}]$ and $\boldsymbol{x}_{\boldsymbol{F}} = [x_{F,1}, x_{F,2}]$ are combined as $\boldsymbol{x}_{\boldsymbol{c}} = [\boldsymbol{x}_{\boldsymbol{\theta}}, \boldsymbol{x}_{\boldsymbol{\phi}},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

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 thrust command. Table 5 shows the quadrotor input model's fit. We use NRMSE (MATLAB's definition) to facilitate comparison; a 100% NRMSE means a perfect fit.

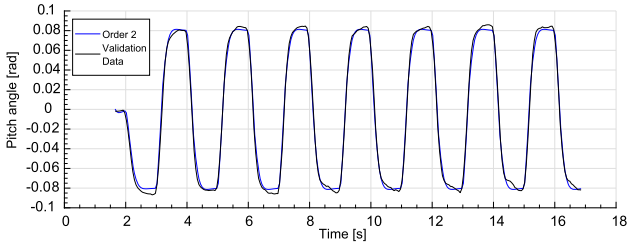
As the thrust command activates a velocity stabiliser loop, the commanded value will be tracked by the quadrotor hence a velocity input-output model can be identified. Using this relation, to obtain a model for the vertical force generated F_q, the velocity model's differential output is multiplied by the system mass. This approach to model the thrust command response is rudimentary so future studies would explore a better approach to modeling this.

Table 5 Normalised Root Mean Squared Error (NRMSE) to estimation and validation dataset for empirically identified linear second-order quadrotor input control model

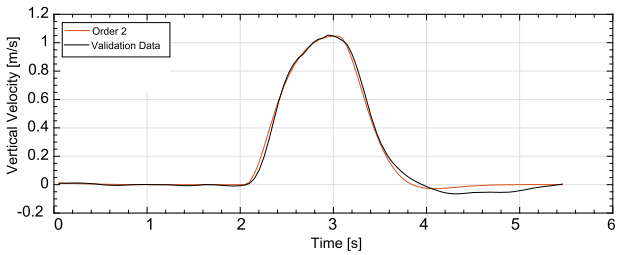
System	Model fit as NRMSE to dataset [%]	
	Estimation	Validation
Pitch	94.59	93.81
Roll	91.09	89.09
Thrust command	91.38	91.56



(a) Roll angle response



(b) Pitch angle response



(c) Vertical velocity response to thrust command

Fig. 19 Identified second order input model fit to experimentally collected system response validation data on the Parrot Bebop 2. For roll and pitch a 5°, 0.5 Hz square wave input was used over a duration of 15 s starting at 2 s. For the thrust command, a 1 m/s pulse of width 1 s was used starting at 2s

Figure 19 shows the second order models' fit to the validation data experimentally collected on the Parrot Bebop 2. For thrust command, the velocity input-output model response is shown (thus prior to taking the differential output and multiplying by the system mass).

As shown the identified model properly tracks the quadrotor's true pitch, roll and the vertical velocity response when given a pitch, roll and thrust command. This is also reflected in the NRMSE model fit shown in Table 5.

E 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

$$d(s) = \frac{(x_{qo} + su)^2}{a^2} + \frac{(y_{qo} + sv)^2}{b^2} + \frac{(z_{qo} + sw)^2}{c^2} - 1. \quad (50)$$

Minimising (50) with respect to s, we get the ellipsoid's closest point along the infinite expansion of line $\mathcal{L}$

$$\hat{s} = \arg\min_{\hat{s}\in\mathbb{R}} d(s) = -\frac{x_{qo}ub^2c^2 + y_{qo}va^2c^2 + z_{qo}wa^2b^2}{u^2b^2c^2 + v^2a^2c^2 + w^2a^2b^2}. \quad (51)$$

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

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

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

F Primal-dual interior-point constrained optimisation algorithm

The Model Predictive Controller (MPC) algorithm is a receding and finite-horizon, constrained optimisation problem that solves for quasi-optimal solutions. Global optima can be 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 and 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 paper and is made clear from the context or by explicit definition in this appendix.

F.1 Optimisation problem definition

Take the cost function minimisation optimisation problem with the total cost J, and with $f : \mathbb{R}^n \to \mathbb{R}$ subject to p inequality and $q - p$ equality constraints as defined by

$$\begin{aligned} J = &\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \\ &g_i(\boldsymbol{x}) \geq b_i \quad \text{for } i = 1, \ldots, p \in \mathbb{N} \\ &g_i(\boldsymbol{x}) = b_i \quad \text{for } i = p+1, \ldots q \in \mathbb{N}. \end{aligned} \quad (52)$$

Rewrite an equivalent problem using slack variables s_i to convert the inequality constraints to equality constraints.

$$\begin{aligned} J = &\min_{\boldsymbol{x}\in\mathbb{R}^n} f(x) \\ &c_i(\boldsymbol{x}, \boldsymbol{s}) = 0 \quad \text{for } i = 1, \ldots, q \in \mathbb{N} \\ &s_i \geq 0 \quad \text{for } i = 1, \ldots, p \in \mathbb{N}, \end{aligned} \quad (53)$$

where,

$$\begin{aligned} c(\boldsymbol{x}, \boldsymbol{s}) = \big[&g_i(\boldsymbol{x}) - b_i - s_i, \ldots, \\ &g_p(\boldsymbol{x}) - b_k - s_p,\ g_{p+1}(\boldsymbol{x}) - b_{p+1}, \ldots, \\ &g_q(\boldsymbol{x}) - b_q\big]^\top. \end{aligned}$$

The non-negativity constraint on the slack variables is rewritten as a logarithmic barrier function cost term giving

$$\begin{aligned} B = &\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) - \mu \sum_{i=1}^{p} \ln(s_i) \\ &c_i(\boldsymbol{x}, \boldsymbol{s}) = 0 \quad \text{for } i = 1, \ldots, q \in \mathbb{N}, \end{aligned} \quad (54)$$

which gives the equivalent *barrier problem*.

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

$$L(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s})^\top \boldsymbol{\lambda} - \mu \sum_{i=1}^{p} \ln(s_i) \quad (55)$$

where *primal* variable is $\boldsymbol{x}$ with the dual variable $\boldsymbol{\lambda} \in \mathbb{R}^q$.

F.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 (55) is defined by equations

$$\nabla_x L \equiv \nabla_x f(\boldsymbol{x}) - \nabla_x \boldsymbol{c}^T(x, s)\boldsymbol{\lambda} = 0 \quad (56)$$

$$\nabla_s L \equiv -\mu S^{-1} e + Y\boldsymbol{\lambda} = 0 \quad (57)$$

$$\nabla_\lambda L \equiv \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s}) = 0 \quad (58)$$

$$\text{with} \quad S = \begin{bmatrix} s_1 & & 0 \\ & \ddots & \\ 0 & & s_p \end{bmatrix}, Y = \left[I_{(p\times p)}\ 0_{(p\times q-p)}\right],$$

where the gradients are set to equal zero. These three necessary conditions constrained optimisation problems are

referred to as the *Karush–Kuhn–Tucker (KKT)* conditions Kuhn (2014).

Rewriting, the above we get the reformulated conditions

$$\nabla_x f(\boldsymbol{x}) - \nabla_x \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s})^\top \boldsymbol{\lambda} = 0 \tag{59}$$

$$\Lambda S \boldsymbol{e} = \mu \boldsymbol{e} \tag{60}$$

$$\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s}) = 0 \tag{61}$$

$$\text{with} \quad \Lambda = \text{diag}(Y\boldsymbol{\lambda}) = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_p \end{bmatrix}$$

$\boldsymbol{e} = [1, \ldots, 1] \in \mathbb{N}^p$.

Applying the Newton–Raphson method for root-finding to the set of Eqs. 59 to 61, the following system of equations is obtained with the Δ search steps;

$$\begin{bmatrix} W & 0 & -\nabla_x \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s})^\top \\ 0 & \Lambda & S \\ \nabla_x \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s}) & -Z & 0 \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{s} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} \tag{62}$$

$$= - \begin{bmatrix} \nabla_x f(\boldsymbol{x}) - \nabla_x \boldsymbol{c}^\top(\boldsymbol{x}, \boldsymbol{s})\boldsymbol{\lambda} \\ \Lambda S \boldsymbol{e} - \mu \boldsymbol{e} \\ \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s}) \end{bmatrix}$$

$$\text{with} \quad Z = \begin{bmatrix} I_{(p \times p)} & 0 \\ 0 & 0_{(q-p \times q-p)} \end{bmatrix}$$

$W = \nabla^2_{xx} L(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda})$.

The left matrix is the Jacobian to set of Eqs. 59 to 61.

Computing the search direction given by the vector of Δ is the most computationally expensive step of the algorithm Domahidi et al. (2012). In practice this is achieved through various methods, however, this is beyond the scope of this paper.

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. 62 resulting in the update

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{s} \\ \boldsymbol{\lambda} \end{bmatrix}_{(k+1)} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{s} \\ \boldsymbol{\lambda} \end{bmatrix}_{(k)} + \alpha_{(k)} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{s} \\ \Delta \boldsymbol{\lambda} \end{bmatrix}_{(k)}, \tag{63}$$

where the step-size is regulated to achieve a converging line-search.

F.3 Algorithm convergence and extensions

As the Newton–Raphson method is iterative in nature, the KKT conditions defined by Eqs. 59 to 61 are set to be satisfied when a certain tolerance ϵ is satisfied.

$$\max \left| \nabla_x f(\boldsymbol{x}) - \nabla_x \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s})^\top \boldsymbol{\lambda} \right| \leq \epsilon_1$$

$$\max |\Lambda S \boldsymbol{e} - \mu \boldsymbol{e}| \leq \epsilon_2$$

$$\max |\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s})| \leq \epsilon_3$$

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 and Jerez (2014); the specific's of which are beyond the scope of this paper. 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 et al. (2002) provides a comprehensive overview of interior-point methods for non-linear optimisation.

G Goal directed assistive steering

Collision-free trajectory generation for low planning horizons can be augmented using an assistive steering based cost term; the idea is inspired from Vector Field Histograms Borenstein and Koren (1991). 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

$$\gamma^* = \arg\min_{\gamma} \left| \gamma - \gamma_{\text{goal}} \right|, \gamma \in \mathcal{D}_{\text{free}}. \tag{64}$$

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

$$c_{\text{steer}} = \left\| \angle \boldsymbol{T}_P \dot{\boldsymbol{p}}_q - \gamma^* \right\|_{\boldsymbol{I}_1}. \tag{65}$$

Under a short planning horizon, evasive actions for obstacles are generally performed abruptly as there is limited planning foresight. Steering assists planning by guiding the system towards obstacle-free areas for smoother trajectories; we demonstrate the utility of steering 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.

H Effects of steering on system response

To address the underperformance of the system response to incoming obstacles using a low number of stages, assistive steering is used to guide the MAVP towards obstacle-free regions without compromising on run-time performance. Inclusion of the Goal Directed Assistive Steering as introduced in "Appendix G" is demonstrated by repeating the simulation study from Sect. 5.1. The steering associated cost weight is set to $w_{\text{steer}} = 0.05$, with a detection range of 3.5 m and 8 pre-defined uniformly space steering direction were used. As with the other cost weights, w_{steer} was found by manually tuning the weights. Figure 20 shows how the NMPC solve time and time-to-goal scales with the number of stages when using the assistive steering associated cost.

Comparing the NMPC algorithm solve times for the case when assistive steering is disabled and enabled, as shown in Figs. 7 and 20, respectively, demonstrates that there is a negligible delta. Referring to Fig. 21 for the N=10 case, observe how the steering assisted trajectory is guided away from the obstacle resulting in a collision-free task completion. Compare this behaviour to the N=10 case with no steering shown in Fig. 8a where clearly the steering guides the system preemptively. 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 more apparent for low stage (N) counts where the guidance is used to improve local planning. A low N 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.

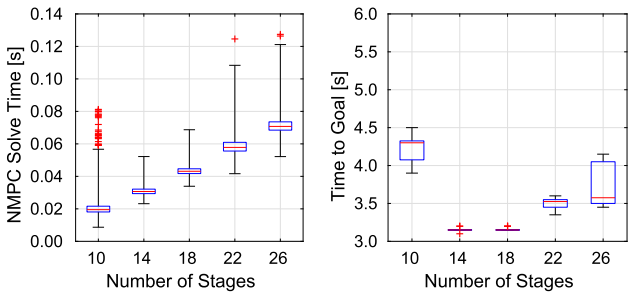


Fig. 20 Simulated NMPC solve time and time-to-goal per run with increasing planning stages using assistive steering and 16 runs per case

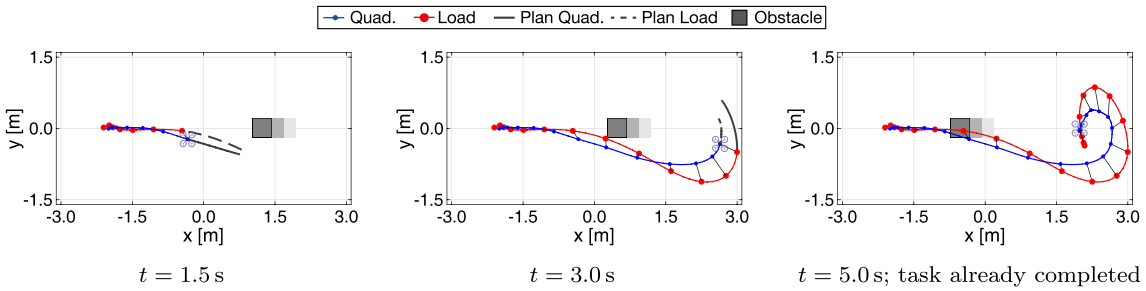


Fig. 21 Point-to-point navigation with collinear dynamic obstacle, $N = 10$ and assistive steering enabled; showing planned and executed trajectory and current quadrotor position. Notice how the steering guide the system away from the obstacle even when the MPC prediction horizon would not capture the oncoming obstacle. Top down view

References

Bisgaard, M., Cour-Harbo, A., & Bendtsen, J. D. (2007). Full state estimation for helicopter slung load system. *Proceedings of AIAA conference on guidance, navigation, and control*, August (pp. 1–15).

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.

Borenstein, J., & Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, *7*(3), 278–288.

Brock, O., & Khatib, O. (2000). 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 No00CH37065)* (Vol. 1, No. April, pp. 550–555).

Cicolani, L. S., & Kanning, G. (1992). Equations of motion of slung-load systems, including multilift systems. Technical Report, National Aeronautics and Space Administration, Moffett Field.

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

Derafa, L., Madani, T., Benallegue, A., & (2006) Dynamic modelling and experimental identification of four rotors helicopter parameters. In *2006 IEEE international conference on industrial technology*. Mumbai: IEEE.

Domahidi, A., & Jerez, J. (2014). FORCES Professional. embotech GmbH. http://embotech.com/FORCES-Pro.

Domahidi, A., Zgraggen, A. U., Zeilinger, M. N., Morari, M., & Jones, C. N. (2012). 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.

Faust, A., Palunko, I., Cruz, P., Fierro, R., & Tapia, L. (2013). Learning swing-free trajectories for UAVs with a suspended load. *Proceedings—IEEE international conference on robotics and automation* (pp. 4902–4909). Karlsruhe: IEEE.

Faust, A., Palunko, I., Cruz, P., Fierro, R., & Tapia, L. (2017). Automated aerial suspended cargo delivery through reinforcement learning. *Artificial Intelligence*, *247*, 381–398.

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.

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). RSS Foundation: Boston.

Forsgren, A., Gill, P. E., & Wright, M. H. (2002). Interior methods for nonlinear optimization. *SIAM Review*, *44*(4), 525–597.

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*, ICRA, Seattle.

Julier, S. J., & Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In: *Proc.SPIE 3068* (p. 12). arXiv:1011.1669v3.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, *4*(4), 373–395. 0604171.

Kerrigan, E. C., & Maciejowski, J. M. (2000). Soft constraints and exact penalty functions in model predictive control. *Proccedings of the UKACC international conference (Control 2000)*.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, *5*(1), 90–98. arXiv:1011.1669v3.

Kim, H., Shim, D., & Sastry, S. (2002). 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, pp. 3576–3581). IEEE, Anchorage.

Klausen, K., Fossen, T. I., & Johansen, T. A. (2015). Nonlinear control of a multirotor UAV with suspended load. *2015 International conference on unmanned aircraft systems, ICUAS 2015* (pp. 176–184). Denver: IEEE.

Kuhn, H. (2014). Nonlinear programming: A historical view. In G. Giorgi & T. Kjeldsen (Eds.), *Traces and Emergence of Nonlinear Programming* (pp. 393–414). Basel: Birkhüser.

LaValle, S. M. (2011). Motion planning. *IEEE Robotics & Automation Magazine*, *18*(1), 79–89.

Lee, T. (2018). Geometric control of quadrotor UAVs transporting a cable-suspended rigid body. *IEEE Transactions on Control Systems Technology*, *26*(1), 255–264.

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.

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.

Neunert, M., de Crousaz, C., Furrer, F., Kamel, M., Farshidian, F., Siegwart, R., et al. (2016). 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.

Palunko, I., Cruz, P., & Fierro, R. (2012a). Agile load transportation : Safe and efficient load manipulation with aerial robots. *IEEE Robotics and Automation Magazine*, *19*(3), 69–79.

Palunko, I., Fierro, R., & Cruz, P. (2012b). 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.

Palunko I, Faust A, Cruz P, Tapia L, Fierro R (2013) A reinforcement learning approach towards autonomous suspended load manipulation using aerial robots. In *2013 IEEE international conference on robotics and automation* (pp. 4896–4901).

Pizetta, I. H. B., Brandao, A. S., & Sarcinelli-Filho, M. (2015). 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.

Point, N. (2009). Optitrack-optical motion tracking solutions. Retrieved November 10, 2017, from http://optitrack.com/.

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.

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.

Stanculeanu, I., & Borangiu, T. (2011). Quadrotor black-box system identification. *International Journal of Mechanical and Mechatronics Engineering*, *5*(6), 1025–1028.

Tang, S., & Kumar, V. (2015). Mixed Integer Quadratic Program trajectory generation for a quadrotor with a cable-suspended payload. In *Proceedings—IEEE international conference on robotics and automation 2015*, June (pp. 2216–2222).

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.
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.
Uteshev, A. Y., & Goncharova, M. V. (2018). Point-to-ellipse and point-to-ellipsoid distance equation analysis. *Journal of Computational and Applied Mathematics*, *328*(January), 232–251.
Vanderbei, R. J. (2012). Interior point methods and nonlinear optimization.
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*, *93*, 13–29.
Zheng, A., & Morari, M. (1995). Stability of model predictive control with mixed constraints. *IEEE Transactions on Automatic Control*, *40*(10), 1818–1823.

Nikhil D. Potdar received his M.Sc. degree in Aerospace Engineering and specialised in ight control systems and operations at the Delft University of Technology, The Netherlands, in 2018. Mr. Potdar completed his master thesis, from which this article originates, under the supervision of Dr. de Croon (MAVLab) and Dr. Alonso-Mora (Cognitive Robotics) in Delft.

Dr. Guido de Croon is Associate Professor and scientific lead in the Micro Air Vehicle lab. He received his M.Sc. and Ph.D. in the field of Artificial Intelligence (AI) at Maastricht University, the Netherlands. His research interest lies with computationally efficient algorithms for robot autonomy, with a particular focus on computer vision. Since 2008 he has worked on algorithms for achieving autonomous flight with small and lightweight flying robots, such as the DelFly Explorer; a fully autonomous 20-gram apping wing MAV. In 2011-2012, he was a research fellow in the Advanced Concepts Team of the European Space Agency, where he studied topics such as optical flow based control algorithms for extraterrestrial landing scenarios. He is currently working at TU Delft again, where he is the Principal Investigator in various projects, e.g., the NWO project "Decentralized control of UAVs." and the STW project "As nimble as a bee".

Dr. Javier Alonso-Mora is an Assistant Professor at the Delft University of Technology, in the Department of Cognitive Robotics. Until October 2016 he was a Postdoctoral Associate at the Computer Science and Artificial Intelligence Lab CSAIL of MIT, working in the Distributed Robotics Lab. Dr. Alonso-Mora received his Ph.D. (2014) and M.Sc. (2010) degrees in robotics from ETH Zurich, where he worked in the Autonomous Systems Lab. Dr. Alonso-Mora holds a Diploma in Engineering (2010) and a Diploma in Mathematics (2008) from the Technical University of Barcelona. Until 2014 was also a member of Disney Research Zurich. Dr. Alonso-Mora is the recipient of a NWO Veni grant from the Netherlands Organisation for Scientific Research (2017), a best video award at the IEEE/ACM HRI (2014), a nomination for best student paper award at DARS (2010), a postgraduate scholarship from the Swiss Government and silver medals in the Spanish Physics and Mathematics Olympiads.