Aerial navigation in obstructed environments with embedded nonlinear model predictive control

Small, E., Sopasakis, P., Fresk, E., Patrinos, P., & Nikolakopoulos, G. (2019). Aerial navigation in obstructed environments with embedded nonlinear model predictive control. In *European Control Conference 25/06/2018 → 28/06/2019 Naples, Italy* (pp. 3556-3563). Institute of Electrical and Electronics Engineers Inc.. https://doi.org/10.23919/ECC.2019.8796236

Published in:
European Control Conference 25/06/2018 → 28/06/2019 Naples, Italy

Document Version:
Peer reviewed version

Aerial navigation in obstructed environments with embedded nonlinear model predictive control

Elias Small, Pantelis Sopasakis, Emil Fresk, Panagiotis Patrinos and George Nikolakopoulos

***Abstract*— We propose a methodology for autonomous aerial navigation and obstacle avoidance of micro aerial vehicles (MAVs) using non-linear model predictive control (NMPC) and we demonstrate its effectiveness with laboratory experiments. The proposed methodology can accommodate obstacles of arbitrary, potentially non-convex, geometry. The NMPC problem is solved using PANOC: a fast numerical optimization method which is completely matrix-free, is not sensitive to ill conditioning, involves only simple algebraic operations and is suitable for embedded NMPC. A `C89` implementation of PANOC solves the NMPC problem at a rate of $20\,\mathrm{Hz}$ on board a lab-scale MAV. The MAV performs smooth maneuvers moving around an obstacle. For increased autonomy, we propose a simple method to compensate for the reduction of thrust over time, which comes from the depletion of the MAV's battery, by estimating the thrust constant.**

I. Introduction

A. Background and motivation

The need for safe aerial navigation and increased micro aerial vehicle (MAV) autonomy nowadays poses all the more relevant and pressing research questions, as drones make their appearance in numerous application domains, such as the inspection of critical or aging infrastructure [1], surveying of underground mines [2], visual area coverage for search-and-rescue operations [3], precision agriculture [4] and many others. In the majority of these applications, MAVs have to navigate in obstructed environments, with static or moving obstacles of arbitrary geometry in known, or partially unknown surrounding environments.

Several methods have been proposed for navigation and collision avoidance, such as potential field methods [5], [6] and graph search methods [7]. Alongside these methods, nonlinear model predictive control (NMPC) is becoming popular for the autonomous navigation of various MAVs including fixed-wing aircrafts [8], [9] and multi-rotor vehicles [10]. NMPC uses a nonlinear dynamical model of the system dynamics to predict position and attitude trajectories from its current position to a reference point, while avoiding all obstacles on its way and minimizing a certain energy/cost function. This way, a nonconvex optimization problem needs to be solved at every sampling time instant in a receding horizon fashion. Another approach to obstacle avoidance is described in [11] where a high-level path planner generates collision-free trajectories which are followed by an MPC controller.

In [12], sequential quadratic programming (SQP) is used to solve the NMPC problem for the navigation of a multi-rotor MAV with a slung load, where the authors demonstrated the effectiveness of NMPC, however, provided neither evidence of the solution quality or solver performance, nor an experimental verification. NMPC was used in [13] for solving obstacle and collision avoidance for several MAVs flying in formation, however again, only simulations were done and the computation time was addressed.

Clearly, the presence of obstacle/collision avoidance constraints makes the MPC problems particularly hard to solve. SQP is the method of choice in the literature [12], [13], [14]. Its main disadvantage is the fact that it requires the solution of a quadratic program (QP) at every iteration of the algorithm, which requires inner iterations. SQP also requires computing and storing of the Jacobian matrices of the dynamics, and sometimes the Hessians when the Hessian of the Lagrangian is used in the QPs. Furthermore, the gradient descent method has been used to solve nonlinear MPC problems for aerial navigation [14]. This method, however, is sensitive to bad conditioning — indeed, problems with long horizons tend to become ill conditioned — while the convergence is expected to be slow.

B. Contributions

In this article we propose a control methodology for the autonomous navigation of MAVs in obstructed environments. We allow for the obstacles to have arbitrary nonconvex shapes and, contrary to distance-based methods [15], we do not require that the distance function between the MAVs and each obstacle is available.

The NMPC optimization problem is solved with PANOC [16], [17] — a recently proposed algorithm for nonconvex optimization problems, which is suitable for embedded NMPC as it requires only simple and cheap linear operations (mainly inner products of vectors) and exhibits fast convergence. Unlike SQP, PANOC is matrix-free and only requires the computation of Jacobian-vector products, which can be

P. Sopasakis is with Queen's University Belfast, School of Electronics, Electrical Engineering and Computer Science, Centre For Intelligent Autonomous Manufacturing Systems (i-AMS) p.sopasakis@qub.ac.uk.

E. Small, E. Fresk and G. Nikolakopoulos are with the Robotics Team at Luleå Technical University, Luleå SE-97187, Sweden elias.small, emil.fresk, geonik@ltu.se

P. Patrinos is with the Department of Electrical Engineering (ESAT-STADIUS), KU Leuven, Kasteelpark Arenberg 10, 3001, Leuven, Belgium. panos.patrinos@esat.kuleuven.be

This work was partially funded by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No.730302 – SIMS. P. Sopasakis was supported by European Union's Horizon 2020 research and innovation programme (KIOS CoE) under Grant No. 739551. P. Patrinos was supported by: FWO projects: No. G086318N; No. G086518N; Fonds de la Recherche Scientifique — FNRS, the Fonds Wetenschappelijk Onderzoek — Vlaanderen under EOS Project No. 30468160 (SeLMA) and Research Council KU Leuven C1 project No. C14/18/068.

computed very efficiently by backward (adjoint) automatic differentiation. PANOC has been shown in [16], [17], [18] to significantly outperform both SQP and interior-point methods. To the best of our knowledge, this is the first time that a fast NMPC optimization problem is demonstrated on an aerial platform, paving the way for future developments in aerial robotics.

Our modeling approach has the strong merit of being independent of the mass of the MAV, in contrast to existing approaches that require the knowledge of the mass and other parameters of the MAV [10], [11], [13]. Our approach is, instead *plug and play* and easy to use and generalize as it can be used without measuring the mass or tuning the available thrust.

Evidence of the solution quality is provided by physical laboratory experiments. A MAV is flown completely autonomously in a laboratory equipped with a VICON motion capture system. The proposed method uses a full position and attitude model of the MAVs and is able to run onboard, using 8-15% CPU of a single core on an Intel Atom Z8350. As shown in Section V, the onboard controller is able to successfully navigate the MAV around an obstacle running at a sampling rate of 20 Hz and a prediction horizon of 2 s.

II. MAV DYNAMICS

A. MAV kinematics

The model of a quadrotor MAV, discussed in [9], assumes that there exists a low-level controller of roll, pitch, yaw rate and thrust. This convention is common in MAV flight controllers such as PixHawk, [19] and ROSFlight, [20]. The high-level kinematics of the MAV is given by

$$\dot{p}(t) = v(t), \tag{1a}$$

$$\dot{v}(t) = R(\theta_r, \theta_p)\begin{bmatrix} 0 \\ 0 \\ T_d \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} - \begin{bmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & A_z \end{bmatrix} v(t), \tag{1b}$$

$$\dot{\theta}_r(t) = {}^1\!/_{\tau_r}(K_r\theta_{r,d}(t) - \theta_r(t)), \tag{1c}$$

$$\dot{\theta}_p(t) = {}^1\!/_{\tau_p}(K_p\theta_{p,d}(t) - \theta_p(t)), \tag{1d}$$

where $p(t) = (p_x(t), p_y(t), p_z(t)) \in \mathbb{R}^3$ and $v(t) \in \mathbb{R}^3$ are the position and velocity of the MAV in the global frame of reference, and $\theta_r \in \mathbb{R}$ and $\theta_p \in \mathbb{R}$ are the roll and pitch angles, while $\theta_{r,d} \in \mathbb{R}$ and $\theta_{p,d} \in \mathbb{R}$ are the reference angles sent to the low-level controller. Furthermore, $T_d \in \mathbb{R}_+$ is the z-axis thrust acceleration, while A_x, A_y, and A_z are the linear drag coefficients. The lower layer — the attitude control system — is modeled by simple first-order dynamics with time constant τ_r and τ_p and gains K_r and K_p for the roll and pitch, respectively. Lastly, $R(\theta_r, \theta_p) \in \mathrm{SO}(3)$ describes the MAV's attitude and is defined by the classical Euler angles in rotation matrix form as

$$R(\theta_r, \theta_p) = R_y(\theta_p)R_x(\theta_r),$$

with

$$R_x(\theta_r) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_r) & -\sin(\theta_r) \\ 0 & \sin(\theta_r) & \cos(\theta_r) \end{bmatrix},$$

$$R_y(\theta_p) = \begin{bmatrix} \cos(\theta_p) & 0 & \sin(\theta_p) \\ 0 & 1 & 0 \\ -\sin(\theta_p) & 0 & \cos(\theta_p) \end{bmatrix}.$$

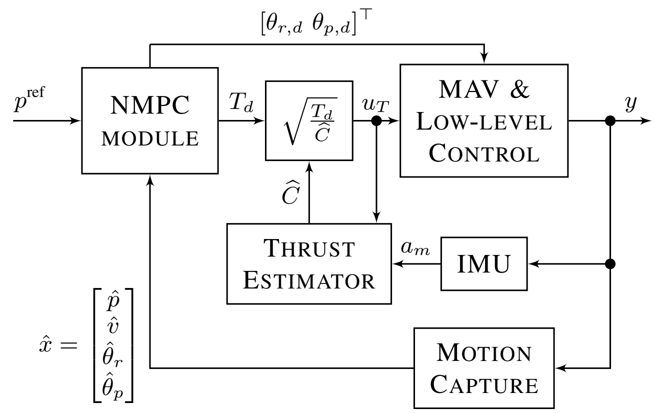

Fig. 1. This diagram represents the complete MAV system: p^{ref} is the reference position sent to the NMPC controller, which calculates the desired angles, $\theta_{r,d}$, $\theta_{r,d}$, and thrust, T_d. The thrust estimator uses the measured acceleration from the IMU, a_m, to yield an estimate of the thrust constant, $\widehat{C}$, which is used to obtain the thrust control signal u_T. x, is estimated by the motion capture system and measured by the IMU which produces $\hat{p}, \hat{v}, \hat{\theta}_r, \hat{\theta}_p$, which are sent to the NMPC module, and the linear acceleration a_m which is sent to the thrust estimator.

Note that yaw is absent in this rotation matrix, as this model operates in a yaw-compensated global frame, and the position control of the MAV is therefore independent of its yaw. Moreover, it is important to note that we have chosen the acceleration, T_d, to be the manipulated variable of the system, rather than the corresponding force, for the model to be mass-free. This has the strong merit of making the controller robust to changes in the mass of the MAV, the available thrust from the motors, and the loss of thrust over time due to the decline of battery voltage.

B. Adaptive acceleration control

In order for our design to be independent of the physical characteristics that determine the available thrust acceleration, a simplified version of [21] is used to continuously estimate the MAV's maximum available thrust. Following [21], the force, F, that is exercised by the propellers, is given by

$$F = C_T u_T^2, \tag{2}$$

where C_T is the *thrust constant* and $u_T \in [0,\ 1]$ is a unitless normalized thrust control signal. The thrust constant may change during flight due to, for instance, the battery drain and the proximity of the MAV to the ground. This is why identifying a constant is not sufficient to track thrust references. The issue is that there is no sensor in the system, which measures the generated force, however, the IMU can provide a measurement of the linear acceleration, albeit noisy. Then, by dividing the thrust model by the mass of the MAV, m, the resulting model is based on acceleration:

$$a = \frac{F}{m} = \frac{C_T}{m}u_T^2 = Cu_T^2, \tag{3a}$$

where $C \triangleq {}^{C_T}\!/_{m}$ is the *special thrust constant* of the vehicle. Now the acceleration is measurable, together with noise, and u_T is what is sent to the low-level controllers, hence now we merely need to design an observer to estimate C. Since C is a slow moving parameter, we use the simple model

$$\dot{C} = \sigma_C^2 w, \tag{3b}$$

where w is a zero-mean white noise[1]. Equations (3a) and (3b) define a nonlinear dynamical system with state variable C, input u_T and output $y(C, u_T) = C u_T^2$. We estimate C by means of an extended Kalman filter (EKF). EKF is chosen because it is simple to tune, it allows to specify an initial estimated variance, and converges fast in the first few iterations.

In addition, we employ an outlier rejection scheme based on bounds of the *direct estimate* $\widetilde{C} \in [1g,\ 10g]$, which is defined as

$$\widetilde{C} = \frac{a_m}{u_T^2}. \tag{4}$$

This is calculated for each IMU acceleration measurement a_m, which implies that each acceleration measurement is inspected to enforce that no outliers are allowed to update the filter. These bounds result from the fact that an MAV must be able to generate at least $1g$ of thrust to take off and it is assumed that it cannot generate more than $10g$ of thrust. The bounds on $\widetilde{C}$ are inherited by the estimates $\widehat{C}$ yielding a simple constrained estimation scheme.

Once the thrust constant is estimated, an acceleration reference can be converted to the thrust control signal u_T, by solving equation (3a) for u_T, resulting in

$$u_T = \sqrt{\frac{T_d}{\widehat{C}}}, \tag{5}$$

A depiction of how the thrust constant estimation is tied to the overall scheme can be found in Fig. 1.

C. Overall system dynamics

The state of the controlled system is defined to be $x(t) = (p(t), v(t), \theta_r(t), \theta_p(t))$ and the manipulated input is $u(t) = (T_d(t), \theta_{r,d}(t), \theta_{p,d}(t))$. The system is equipped with a VICON motion capture system which measures the full odometry of the system and provides the corresponding estimates of the full state of the MAV as $\hat{x} = (\hat{p}(t), \hat{v}(t), \hat{\theta}_r(t), \hat{\theta}_p(t))$. Overall, the system dynamics can be concisely written as

$$\dot{x}(t) = f(x(t), u(t)), \tag{6}$$

where f is implicitly defined via (1).

III. Nonlinear MPC for obstacle avoidance

A. Navigation in obstructed environments

We assume that an MAV needs to navigate towards a reference position $p^{\mathrm{ref}} \in \mathbb{R}^3$, while avoiding a set of $q(t) \in \mathbb{N}$ moving obstacles $\{O_j(t)\}_{j \in \mathbb{N}_{[1,q(t)]}}$.

We select n_{mav} *corner points* on the MAV and position a ball with radius r_{ball} centered at each such point so that the whole vehicle is contained in the union of these balls. We assume that the coordinates of the corner points in the global frame of reference are given by $c_\iota(p(t))$, for $\iota \in \mathbb{N}_{[1,n_{\mathrm{mav}}]}$.

[1] Equation (3b) is a stochastic differential equation which is meant in the sense $\mathrm{d}C = \sigma_C^2 \mathrm{d}B_t$, where B_t is the standard Brownian motion [22, Sec. 5.1].

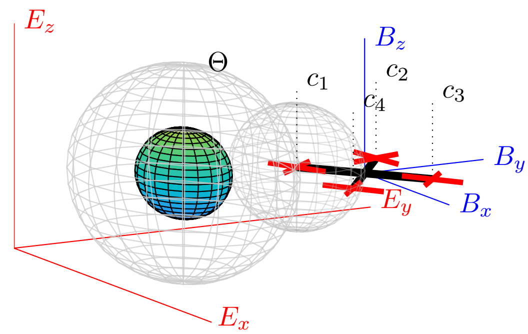

Fig. 2. A quadrotor and an spherical obstacle $O(t)$ (colored solid ball) and its enlargement $\Theta(t)$. We have selected four corner points, c_1, c_2, c_3, c_4 on the MAV. The red lines indicate the earth-fixed frame of reference, (E_x, E_y, E_z), and the blue ones the body-fixed frame, (B_x, B_y, B_z).

In order for the MAV to not collide with the obstacles, we shall require that

$$c_\iota(p(t)) \notin \Theta_j(t) := \underbrace{O_j(t) + \mathcal{B}(r_{\mathrm{ball}})}_{\text{enlarged obstacle}}, \tag{7}$$

for all $j \in \mathbb{N}_{[1,q(t)]}$, $\iota \in \mathbb{N}_{[1,n_{\mathrm{mav}}]}$, where $\mathcal{B}(r_{\mathrm{ball}})$ is a ball centered at the origin with radius r_{ball}. The set $\Theta_j(t)$ is an enlarged version of the original obstacle $O_j(t)$. The concept is illustrated in Fig. 2.

It is assumed that the selected corner points, $c_\iota(p(t))$, for all $\iota \in \mathbb{N}_{[1,n_{\mathrm{mav}}]}$ are such that the set

$$\bigcup_{\iota \in \mathbb{N}_{[1,n_{\mathrm{mav}}]}} c_\iota(p(t)) + \mathcal{B}(r_{\mathrm{ball}})$$

contains the whole MAV.

We introduce the stage cost function $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}_+ \to \mathbb{R}_+$ and the terminal cost function $\ell_f : \mathbb{R}^{n_x} \times \mathbb{R}_+ \to \mathbb{R}_+$ which penalize the deviation of the system state from a reference state. Typical choices are

$$\ell(x, u, t) = \|x - x^{\mathrm{ref}}(t)\|_Q^2 + \|u - u^{\mathrm{ref}}(t)\|_R^2, \tag{8a}$$
$$\ell_f(x, t) = \|x - x^{\mathrm{ref}}(t)\|_{Q_f}^2, \tag{8b}$$

where $Q \in \mathbb{R}^{n_x \times n_x}$, $R \in \mathbb{R}^{n_u \times n_u}$ and $Q_f \in \mathbb{R}^{n_x \times n_x}$ are positive semi-definite matrices and x^{ref} is the reference state which has the form $x^{\mathrm{ref}} = [p^{\mathrm{ref}}\ 0_{1 \times n_x - 3}]^\top$.

The nonlinear model predictive control problem for navigation in an obstructed environment consists in solving the following problem

$$
\begin{aligned}
&\textbf{minimize } J = \ell_f(\bar{x}(T), T) + \int_0^T \ell(\bar{x}(\tau), \bar{u}(\tau), \tau)\mathrm{d}\tau && \text{(9a)}\\
&\textbf{subject to } \bar{x}(0) = \hat{x}, && \text{(9b)}\\
&\qquad \dot{\bar{x}}(t) = f(\bar{x}(t), \bar{u}(t)), && \text{(9c)}\\
&\qquad \bar{u}(t) \in U(t), && \text{(9d)}\\
&\qquad c_\iota(\bar{p}(t)) \notin \Theta_j(t),\ j \in \mathbb{N}_{[1,q(t)]}, && \text{(9e)}\\
&\qquad\quad \iota \in \mathbb{N}_{[1,n_{\mathrm{uav}}]},\ t \in [0, T]
\end{aligned}
$$

where $\bar{u}(t) = (\bar{T}_d(t), \bar{\theta}_{r,d}(t), \bar{\theta}_{p,d}(t))$ and $\bar{x}(t) = (\bar{p}(t), \bar{v}(t), \bar{\theta}_r(t), \bar{\theta}_p(t))$, for $t \in [0, T]$ are the predicted input and state signals.

In this formulation we have assumed that the future trajectories of all obstacles are exactly known and independent of the trajectory of the controlled vehicle. If this is not the case, we have to formulate appropriate robust or stochastic variants of the above obstacle avoidance problem.

The control action is exercised to the system via a zero-order hold element, that is, $\bar{u}(t) = \bar{u}_k$ for $t \in [kT_s, (k+1)T_s)$, where T_s is the sampling period. We assume that $T = NT_s$ for some $N \in \mathbb{N}$. Then, the cost function in (9a) can be written as

$$
J = \ell_f(\bar{x}(T), T) + \sum_{k=0}^{N-1} \int_{kT_s}^{(k+1)T_s} \ell(\bar{x}(\tau), \bar{u}_k, \tau)\mathrm{d}\tau. \qquad (10)
$$

Since it is not possible to derive analytical solutions of the nonlinear dynamical system (9c), the system trajectories as well as the cost function J along these trajectories has to be evaluated by discretizing the system dynamics and integrals. By doing so, the system state trajectoriy $\bar{x}(t)$ is evaluated at points $\bar{x}_k = \bar{x}(kT_s)$ as follows

$$
\bar{x}_{k+1} \approx f_k(\bar{x}_k, \bar{u}_k),
$$

and

$$
J \approx \ell_N(\bar{x}_N) + \sum_{k=0}^{N-1} \ell_k(\bar{x}_k, \bar{u}_k).
$$

Any explicit integration method such as the fourth-order Runge-Kutta or Forward Euler lead to high quality approximations of MAV trajectories. This way, the original continuous-time optimal control problem is approximated by a discrete-time one which is solved at every time instant in a receding horizon fashion.

B. Penalty functions for obstacles of general shape

Each obstacle is described by a set of $m_j(t)$ nonlinear constraints of the form

$$
\Theta_j(t) = \{p \in \mathbb{R}^3 \mid h_j^i(p, t) > 0, i \in \mathbb{N}_{[1,m_j(t)]}\}, \qquad (11)
$$

where $h_j^i : \mathbb{R}^3 \times \mathbb{R}_+ \to \mathbb{R}_+$ are $C^{1,1}$ functions. This approach allows one to describe obstacles of very general convex or nonconvex shape. For example, by choosing functions h_j^i to be affine in p, we can model any polytopic object.

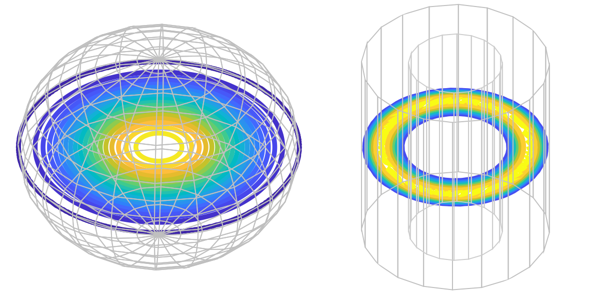

Fig. 3. Level sets of slices of the function $\psi_{\Theta(t)}$ on the plane $\{(p_x, p_y, p_z) \in \mathbb{R}^3 \mid p_z = 0\}$ for (Left) a ball-shaped obstacle and (Right) a nonconvex obstacle. The obstacles are circumscribed by light gray mesh lines.

Functions of the form $h_j(p, t) = 1 - (p - p_0(t))^\top M(t)(p - p_0(t))$ can be used to model ellipsoidal objects or elliptic cylindrical ones. Polynomial, trigonometric and other functions can be used to model more complex geometric shapes.

For simplicity, in this section we focus in the case where there is one obstacle, that is $q(t) = 1$, which we denote by $\Theta(t) = \{p \in \mathbb{R}^3 \mid h^i(p, t) > 0, i \in \mathbb{N}_{[1,m]}\}$. The constraint $c_\iota(p(t)) \notin O(t)$ is satisfied if and only if $h^{i_0}(c_\iota(p(t)), t) \leq 0$ for some $i_0 \in \mathbb{N}_{[1,m]}$, or equivalently, if

$$
\psi_{\Theta(t)}(c_\iota(p(t))) = 0, \qquad (12)
$$

for all $\iota \in \mathbb{N}_{[1,n_{\mathrm{uav}}]}$, where $\psi_{\Theta(t)} : \mathbb{R}^3 \to \mathbb{R}_+$ is the function defined as

$$
\psi_{\Theta(t)}(p) := \tfrac{1}{2} \prod_{i=1}^{m} [h^i(p, t)]_+^2. \qquad (13)
$$

Such functions are illustrated in Fig. 3. Function $\psi_{\Theta(t)}$ takes the value 0 outside the enlarged obstacle $\Theta(t)$ and increases in the interior of it as we move away from its boundary.

Function $\psi_{\Theta(t)}$ is differentiable with gradient

$$
\nabla \psi_{\Theta(t)}(p) = 1_{\Theta(t)}(p) \sum_{i=1}^{m} h^i(p, t) \prod_{j \neq i} (h^j(p, t))^2 \nabla_p h^i(p, t),
$$

where $1_{\Theta(t)}$ is the characteristic function of $\Theta(t)$ with $1_{\Theta(t)}(p) = 1$ if $p \in \Theta(t)$ and $1_{\Theta(t)}(p) = 0$ otherwise.

Functions $\psi_{\Theta(t)}$ can be used to impose the obstacle avoidance requirements as soft constraints. To this end, we eliminate the nonconvex constraints $c_\iota(\bar{p}(t)) \notin \Theta_j(t)$ and introduce the modified stage and terminal cost functions

$$
\tilde{\ell}(\bar{x}, \bar{u}, t) = \ell(\bar{x}, \bar{u}, t) + \sum_{\iota, j} \lambda_{j,\iota} \psi_{\Theta_j(t)}(c_\iota(\bar{p}(t))), \qquad (14a)
$$

$$
\tilde{\ell}_f(\bar{x}, \bar{u}, t) = \ell_f(\bar{x}, \bar{u}, t) + \sum_{\iota, j} \lambda_{j,\iota}^f \psi_{\Theta_j(t)}(c_\iota(\bar{p}(t))), \qquad (14b)
$$

where $\lambda_{j,\iota}$ and $\lambda_{j,\iota}^f$ are positive weight coefficients. The overall model predictive control (MPC) problem becomes

$$\underset{\bar{u}_0,\ldots,\bar{u}_{N-1}}{\text{minimize}}\ \tilde{\ell}_N(\bar{x}_N)+\sum_{k=0}^{N-1}\tilde{\ell}_k(\bar{x}_k,\bar{u}_k) \tag{15a}$$

$$\text{subject to } \bar{x}_0 = x \tag{15b}$$

$$\bar{x}_{k+1} = f_k(\bar{x}_k,\bar{u}_k),\ \ k\in\mathbb{N}_{[0,N-1]} \tag{15c}$$

$$\bar{u}_k \in U_k,\ \ k\in\mathbb{N}_{[0,N-1]} \tag{15d}$$

where $U_k = U(kT_s)$. The optimization is carried out over finite-dimensional vectors $\bar{u}=(\bar{u}_0,\ldots,\bar{u}_{N-1})\in\mathbb{R}^n$ with $n = n_u N$.

C. Single-shooting problem formulation

We shall cast optimization problem (15) in the following compact and simple form

$$\underset{\bar{u}\in U}{\text{minimize}}\ \phi(\bar{u};\,\hat{x},p^{\text{ref}}(\,\cdot\,)), \tag{16}$$

where $U = U_0\times U_1\times\ \ldots\ \times U_{N-1}$ and $\phi:\mathbb{R}^n\to\mathbb{R}_+$ is a $C^{1,1}$ function. To this end, we need to eliminate the state sequence in (15c). Let us introduce a sequence of functions $F_k:\mathbb{R}^n\to\mathbb{R}^{n_x}$ for $k\in\mathbb{N}_{[0,N]}$ defined recursively by

$$F_0(\bar{u}) = \hat{x}, \tag{17a}$$

$$F_{k+1}(\bar{u}) = f_k(F_k(\bar{u}),\bar{u}_k). \tag{17b}$$

This way, $\bar{x}_k = F_k(\bar{u})$. Then, problem (15) is written as in (16) with

$$\phi(\bar{u}) = \tilde{\ell}_N(F_N(\bar{u})) + \sum_{k=0}^{N-1}\tilde{\ell}_k(F_k(\bar{u}),\bar{u}_k). \tag{18}$$

This is known as the *single shooting* formulation [16].

IV. Fast online nonlinear MPC using PANOC

Problem (16) is in a form that can be solved by PANOC [16]. In particular, the gradient of ϕ can be computed using automatic differentiation [23] which is implemented by software such as CasADi [24]. PANOC finds a $\bar{u}^\star\in\mathbb{R}^n$ which solves the optimality conditions

$$R_\gamma(\bar{u}^\star) = 0, \tag{19}$$

where $R_\gamma:\mathbb{R}^n\to\mathbb{R}^n$ is the *fixed-point residual* operator with parameter $\gamma>0$ defined as

$$R_\gamma(\bar{u}) = \bar{u} - T_\gamma(\bar{u}), \tag{20}$$

where $T_\gamma:\mathbb{R}^n\to\mathbb{R}^n$ is the *projected gradient operator*

$$T_\gamma(\bar{u}) = \mathbf{\Pi}_U(\bar{u}-\gamma\nabla\phi(\bar{u})). \tag{21}$$

PANOC combines *safe* projected-gradient updates $\bar{u}^{\nu+1/2}$ with *fast* Newton-type directions d^ν computed by L-BFGS while it uses the forward backward envelope (FBE) function φ_γ as a merit function for globalization given by

$$\varphi_\gamma(\bar{u}) = \phi(\bar{u}) - \tfrac{\gamma}{2}\|\nabla\phi(\bar{u})\|^2 + \tfrac{1}{2\gamma}\,\mathbf{dist}^2_U(\bar{u}-\gamma\nabla\phi(\bar{u})). \tag{22}$$

The forward-backward envelope is an exact, continuous and real-valued merit function which shares the same (local/strong) minima with (16). That said, Problem (16) is reduced to the unconstrained minimization of φ_γ.

PANOC is shown in Algorithm 1. L-BFGS uses a buffer of length μ of vectors $s^\nu = \bar{u}^{\nu+1}-\bar{u}^\nu$ and $y^\nu = R_\gamma(\bar{u}^{\nu+1}) - R_\gamma(\bar{u}^\nu)$ to compute the update directions d^ν [16], [25, Sec. 7.2]. The computation of d^ν requires only inner products which amount to a maximum of $4\mu n$ scalar multiplications. In particular, following [26], the L-BFGS buffer is updated only if ${s^\nu}^\top y^\nu/\|s^\nu\|^2 > \epsilon_d\|R_\gamma(\bar{u})\|$.

Overall, PANOC uses exactly the same oracle as the projected gradient method, that is it only requires the invocation of $\mathbf{\Pi}_U$, ϕ and $\nabla\phi$. Lastly, owing to the FBE-based line search, PANOC converges globally, that is, from any initial guess, $\bar{u}^0$. The algorithm terminates once the infinity norm of the scaled fixed-point residual, $r^\nu := R_\gamma(\bar{u}^\nu)$, drops below a specified tolerance. Note also that a Lipschitz constant of $\nabla\phi$ does not need to be known in advance — a Lipschitz constant is evaluated with a simple backtracking.

Algorithm 1 PANOC algorithm for nonlinear MPC

Input: Initial guess $\bar{u}^0\in\mathbb{R}^n$, Current state $x\in\mathbb{R}^{n_x}$, Estimate $L>0$ of the Lipschitz constant of $\nabla\phi$, L-BFGS memory length μ, Tolerance $\epsilon>0$, Max. iterations $\nu_{\max}$
Output: Approximate solution $\bar{u}^\star$

Choose $\gamma\in(0,1/L)$, $\sigma\in(0,\frac{\gamma}{2}(1-\gamma L))$
for $\nu = 0,1,\ldots,\nu_{\max}$ **do**
  Compute $\nabla\phi(\bar{u}^\nu)$ // Using automatic differentiation
  $\bar{u}^{\nu+1/2}\leftarrow T_\gamma(\bar{u}^\nu)$ // Projected gradient step
  $r^\nu\leftarrow\bar{u}^\nu-\bar{u}^{\nu+1/2}$ // Fixed-point residual
  if $\|r^\nu\|_\infty<\epsilon$, exit // Termination criterion
  // Evaluate a Lipschitz constant for $\nabla\phi$ (and γ, σ):
  while $\phi(\bar{u}^{\nu+1/2}) > \phi(\bar{u}^\nu) - \nabla\phi(\bar{u}^\nu)^\top r^\nu + \frac{L}{2}\|r^\nu\|^2$ **do**
    Empty the L-BFGS buffers
    $L\leftarrow 2L$, $\sigma\leftarrow\sigma/2$, $\gamma\leftarrow\gamma/2$
    $\bar{u}^{\nu+1/2}\leftarrow T_\gamma(\bar{u}^\nu)$
  $d^\nu\leftarrow -H_\nu r^\nu$ using L-BFGS
  $\bar{u}^{\nu+1}\leftarrow\bar{u}^\nu-(1-\tau_\nu)r^\nu+\tau_\nu d^\nu$, where τ_ν is the largest number in $\{1/2^i : i\in\mathbb{N}\}$ such that

$$\varphi_\gamma(\bar{u}^{\nu+1}) \le \varphi_\gamma(\bar{u}^\nu) - \sigma\|r^\nu\|^2$$

V. Experimental validation

For the experimental validation of the proposed control scheme, an inverted quadrotor using the ROSFlight [20] low-level controller was used for all trials, shown in Fig. 4. The onboard computer used is an Aaeon Up Board with an Intel Atom x5-z8350 processor with four 1.44 GHz cores and 2 GB of RAM. The board runs Ubuntu Server 16.04. The field robotics lab at Luleå University of technology is equipped with a Vicon motion capture system featuring 20 infrared cameras that track the odometry of the MAV; this data is used by the NMPC controller for navigation.

The NMPC module runs simple `C89` code which was generated by `nmpc-codegen` — an LGPLv3.0-licensed open-source code generation toolkit which is available at `github.com/kul-forbes/nmpc-codegen`.

An upright cylindrical obstacle, O, is placed so that its vertical symmetry axis runs through the origin $(0.0, 0.0, 0.0)$

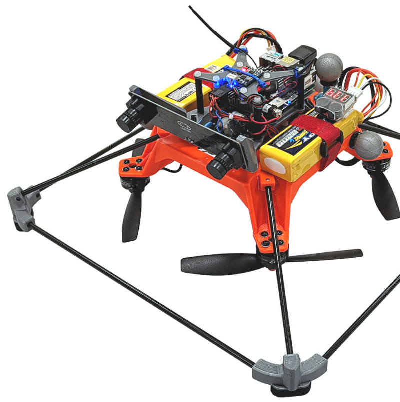

Fig. 4. The inverted quadrotor used in the experiments, which is specifically designed to have a small x/y footprint of $34\,\mathrm{cm}$ by $34\,\mathrm{cm}$, a height of $12\,\mathrm{cm}$, and weight of $1.02\,\mathrm{kg}$, to be suitable for indoor flight.

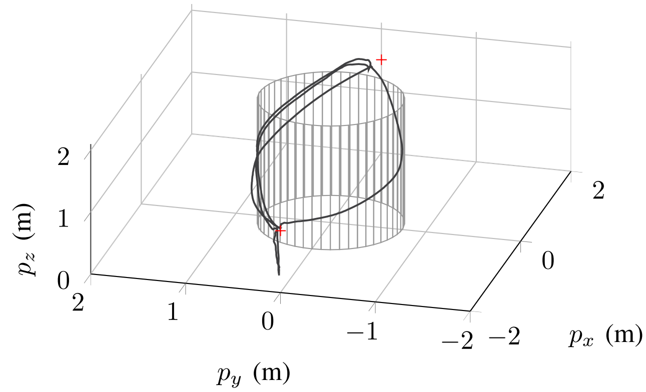

Fig. 5. Path of the UAV autonomously taking off, traveling between the two reference positions $p^{\mathrm{ref},1} = (-2, 0, 1)$ and $p^{\mathrm{ref},2} = (2, 0, 1.5)$, and landing. An upright cylindrical (enlarged) obstacle of radius $75\,\mathrm{cm}$ is positioned so that its axis runs through $(0, 0, 0)$.

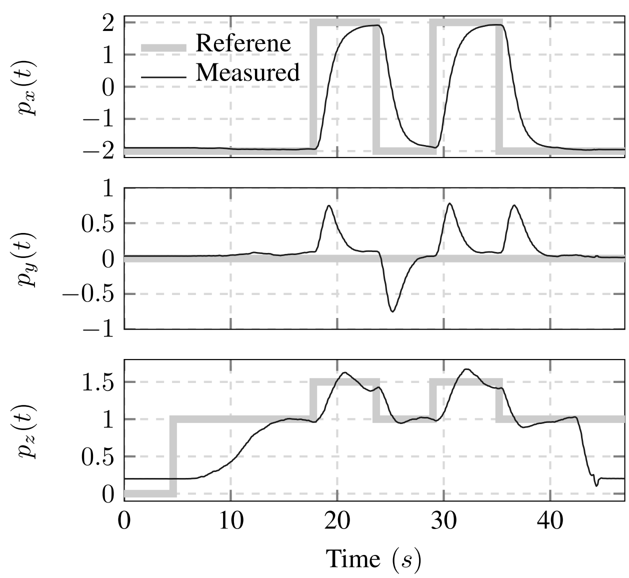

Fig. 6. Smooth navigation of the MAV in space: the position of the vehicle versus time. Positions are in m and time is in s.

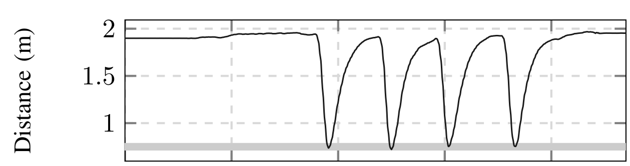

Fig. 7. Distance of the center of the MAV from the center of the enlarged cylinder, Θ. The solid grey line indicates the border of the cylinder at a radius of $75\,\mathrm{cm}$.

of the global coordinate frame in the flying arena at field robotics lab (FROST). The cylinder, O, has a radius of $r_{\mathrm{cyl}} = 0.45\,\mathrm{m}$ and height $z_{\mathrm{cyl}} = 2\,\mathrm{m}$. The obstacle is described by the functions $h^1(p,t) = r^2 - p_x^2 - p_y^2$, $h^2(p,t) = p_z$ and $h^3(p,t) = z_{\mathrm{cyl}} - p_z$. A single corner point is used which is positioned at the center of the MAV; the enclosing ball as in Fig. 2 has a radius of $r_{\mathrm{ball}} = 0.24\,\mathrm{m}$. In order to account for possible small constraint violations due to the fact that obstacle avoidance constraints are modeled via penalty functions, we consider an additional enlargement of $0.06\,\mathrm{m}$. As a result, the enlarged cylinder, $\Theta(t)$, has a radius of $0.75\,\mathrm{m}$ and height $2.3\,\mathrm{m}$. The weights of the obstacle constraints, $\lambda_{j,\iota}$ and $\lambda^f_{j,\iota}$, in Equation (14) were all set to 10000, and the continuous-time system was integrate with the forward Euler method.

The flight test performed for avoiding the obstacle consisted of alternating between two position references on opposite sides of the obstacle. The two position references given alternately were, in meters, $(-2.0, 0.0, 1.0)$ and $(2.0, 0.0, 1.5)$. These references were sent when the MAV was close to its previous reference position. The exact time the references are changed can be seen in Fig. 6.

NMPC runs at $20\,\mathrm{Hz}$ with a prediction and control horizon of 40 steps, meaning the solver predicts the states of the system $2\,\mathrm{s}$ into the future. The solver occupied between 8% and 15% of CPU on an Intel Atom Z8350 — an indication of the solver's computational efficiency.

Fig. 5 shows the actual path flown by the MAV during the test where the positioning data is taken from the motion tracking system and has sub-millimeter accuracy. The path is also shown in Fig. 6 where we plot the MAV's position versus time. The MAV does not have time to settle at the reference altitude as a new reference is sent to the controller before the position completely converges.

As the MAV passes the obstacle it violates the obstacle constraint, as shown in Fig. 7, which is expected from the penalty formulation. The maximum violation is $2.86\,\mathrm{cm}$, which is below the extra enlargement of $6\,\mathrm{cm}$ of the obstacle.

Fig. 8 shows the control signals (roll, pitch, and normalized thrust references) commanded by the NMPC. The roll and pitch angles are bound between $-0.5\,\mathrm{rad}$ and $0.5\,\mathrm{rad}$; these bounds are active as shown in Fig. 8. This further motivates the use of NMPC, allowing for bounds to be directly included in the problem formulation.

The control signals could be made less aggressive by penalizing the rate of change of the input in (15), that is, by adding a penalty of the form $\ell_\Delta = \|\bar{u}_k - \bar{u}_{k-1}\|^2_{R_\Delta}$ for a symmetric positive semidefinite matrix $R_\Delta \in \mathbb{R}^{n_u \times n_u}$. Nevertheless, the maneuvering of the MAV is smooth as shown in Figs. 5, 6 and 7 and a video of the experiment which can be found at https://youtu.be/E4vCSJw97FQ.

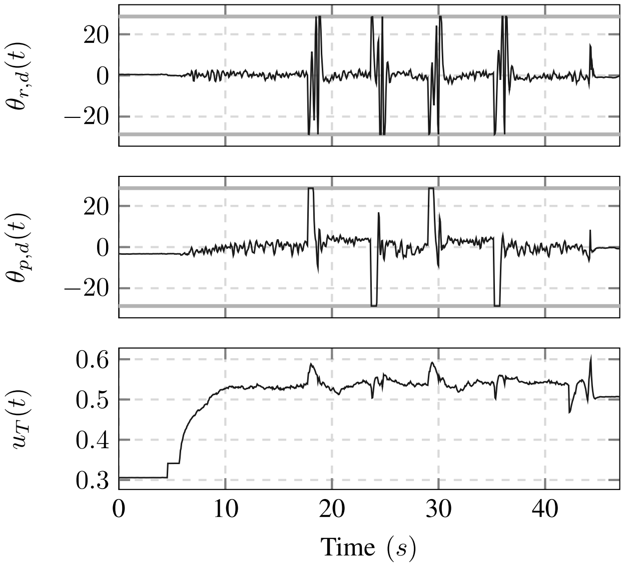

Fig. 8. Control signals sent from the solver to the low-level controller during the experiment. The angles are in degrees for ease of reading.

As shown in the second subfigure of Fig. 9, once the reference changes, the solver reaches the maximum number of iterations (200 iterations) and the solution it returns is of poor quality (fourth subfigure of Fig. 9). This happens because at each time instance, the solver is initialized with the previously computed optimal trajectory. Upon a reference change, the initial guess is rather far from optimal and this necessitates more iterations for convergence. Nonetheless, this inaccuracy is eliminated at the next time instant — 0.05 s later — where the solver is provided a good initial estimate and converges within the prescribed tolerance, $\epsilon = 10^{-3}$. This way, NMPC is executed at 20 Hz. As shown in the third subfigure of Fig. 9, at one time instant, the solution time exceeds the maximum allowed time. This is accommodated by delaying the dispatch of the control action by few ms and has no practical effect.

The infinity norm of the fixed-point residual is below ϵ at all time instants with the exception of four instants from the change of reference. Lastly, the average iteration time in every time step is shown in the third subfigure of Fig. 9, and ranges from 80 μs to 350 μs where the variability is because of the different number of line search iterations.

The parameters used in the dynamics of the MAV used in the experiment are shown in Table I. These values were chosen empirically (based on accurate values for other MAVs) and are not fine-tuned via experiments; this accentuates the fact that the closed-loop and the overall obstacle avoidance scheme is robust to errors in the determination of these parameters.

The tuning parameters used by the NMPC are $R = \mathbf{diag}(2, 10, 10)$, $Q = \mathbf{diag}(3I_2, 12, I_3, 3I_2)$, and $Q_f = 10\,Q$, and the prediction horizon is $T = 2$ s. For the EKF

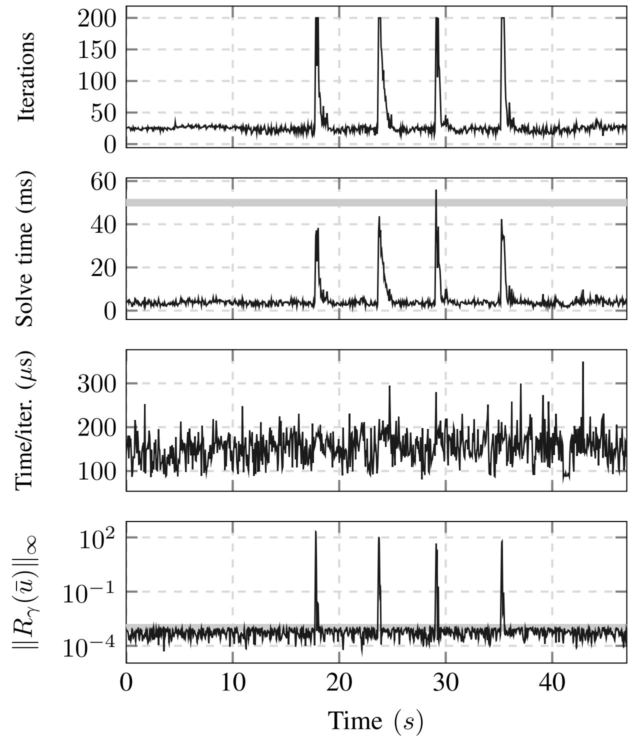

Fig. 9. Solver diagnostics: (Top) Number of iterations required for convergence. Observe that at reference changes, the initial guess is rather inaccurate and the solver requires more iterations, (Middle-top) time required by PANOC to find an optimal sequence of control actions, (Middle-bottom) average time taken per internal iteration, and (Bottom) infinity norm of the fixed-point residual, $\|R_\gamma(\bar{u})\|_\infty$, which serves as an indicator of the solution quality.

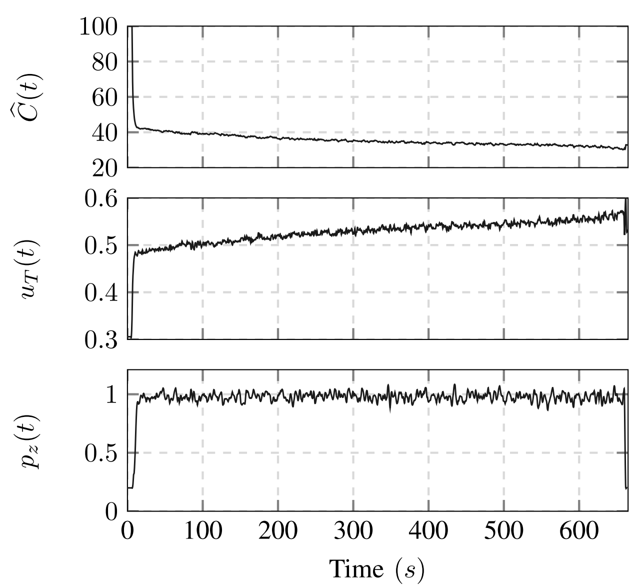

Fig. 10. Control signals sent from the solver to the low-level controller during the experiment.

for estimating the special thrust constant we have

$$P_0 = 100,\ Q_T = 10^{-3},\ R_T = 1,$$

where P_0 is the initial variance for $\widehat{C}$, Q_T is the process

TABLE I

MAV PARAMETERS — VALUES IN SI UNITS

parameter	value	parameter	value
A_x	0.1	τ_r	0.5
A_y	0.1	K_r	1
A_z	0.2	τ_p	0.5
		K_p	1

variance in (3b), and R_T is the measurement variance.

A separate experiment was carried out where the MAV was given a position reference to hold for as long as the battery could deliver power safely. This experiment was conducted to demonstrate the thrust constant estimation described in Section II-B and the results are presented in Fig. 10. As the battery drains, the special thrust constant is decreasing and the control signal is adapted to keep the MAV hovering at a constant altitude. A video from the experiments presented here is found at https://youtu.be/E4vCSJw97FQ.

VI. Conclusions

We presented an obstacle and collision avoidance methodology coupled with an adaptive thrust controller that leads to increased autonomy and context awareness for MAVs. Obstacle avoidance is addressed with an NMPC controller, which is solved using PANOC — a simple and fast algorithm, which involves simple algebraic operations and, unlike SQP, does not require the solution of linear systems at each step. Experiments were performed with the solver running onboard a MAV which maneuvered gently around a virtual obstacle with a smooth trajectory. The MAV passed the edge of the virtual obstacle with a minimal constraint violation, as expected from the solver.

Moreover, experiments were performed to demonstrate that our thrust estimation method successfully compensates for the reduction of thrust over time, making the control scheme applicable to any MAV platform. Future work will focus on experiments in presence of moving obstacles with uncertain trajectories using stochastic [27] and risk-averse [28], [29] variants of MPC.

References

[1] N. Metni and T. Hamel, "A UAV for bridge inspection: Visual servoing control law with orientation limits," *Automation in Construction*, vol. 17, no. 1, pp. 3 – 10, 2007.

[2] C. Kanellakis, S. S. Mansouri, G. Georgoulas, and G. Nikolakopoulos, "Towards autonomous surveying of underground mine using MAVs," in *Advances in Service and Industrial Robotics*, N. A. Aspragathos, P. N. Koustoumpardis, and V. C. Moulianitis, Eds., 2019, pp. 173–180.

[3] S. S. Mansouri, C. Kanellakis, G. Georgoulas, D. Kominiak, T. Gustafsson, and G. Nikolakopoulos, "2D visual area coverage and path planning coupled with camera footprints," *Control Engineering Practice*, vol. 75, pp. 1 – 16, 2018.

[4] C. Zhang and J. M. Kovacs, "The application of small unmanned aerial systems for precision agriculture: a review," *Precision Agriculture*, vol. 13, no. 6, pp. 693–712, Dec 2012.

[5] J. Minguez, F. Lamiraux, and J.-P. Laumond, *Motion Planning and Obstacle Avoidance*. Springer, 2016, pp. 1177–1202.

[6] O. Montiel, U. Orozco-Rosas, and R. Sepúlveda, "Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles," *Expert Systems with Applications*, vol. 42, no. 12, pp. 5177–5191, 2015.

[7] X. Chen and X. Chen, "The UAV dynamic path planning algorithm research based on Voronoi diagram," in *IEEE CCDC*, May 2014, pp. 1069–1071.

[8] T. J. Stastny, A. Dash, and R. Siegwart, "Nonlinear MPC for fixed-wing UAV trajectory tracking: Implementation and flight experiments," in *AIAA Guid., Navig. Contr. Conf.*, jan 2017.

[9] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, *Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system*. Springer, 2017, vol. 2, pp. 3–39.

[10] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463 – 3469, 2017, 20th IFAC World Congress.

[11] Y. Liu, S. Rajappa, J. M. Montenbruck, P. Stegagno, H. Bülthoff, F. Allgöwer, and A. Zell, "Robust nonlinear control approach to nontrivial maneuvers and obstacle avoidance for quadrotor UAV under disturbances," *Robotics and Autonomous Systems*, vol. 98, pp. 317–332, 2017.

[12] C. Y. Son, T. Kim, S. Kim, and H. J. Kim, "Model predictive control of a multi-rotor with a slung load for avoiding obstacles," *14th Int. Conf. Ubiq. Rob. Amb. Intel. (URAI)*, pp. 232–237, 2017.

[13] Z. Chao, S.-L. Zhou, L. Ming, and W.-G. Zhang, "UAV formation flight based on nonlinear model predictive control," *Mathematical Problems in Engineering*, vol. 2012, pp. 1–15, 2012.

[14] D. Shim and S. Sastry, "A situation-aware flight control system design using real-time model predictive control for unmanned autonomous helicopters," in *AIAA Guid., Navig. Contr. Conf.*, 2006.

[15] P. Wang and B. Ding, "A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance," *Int. J. Control*, vol. 87, no. 1, pp. 52–63, 2014.

[16] A. Sathya, P. Sopasakis, A. Themelis, R. V. Parys, G. Pipeleers, and P. Patrinos, "Embedded nonlinear model predictive control for obstacle avoidance using PANOC," in *IEEE ECC*, June 2018.

[17] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *IEEE Conference on Decision and Control (CDC)*, 2017, pp. 1939–1944.

[18] B. Hermans, P. Patrinos, and G. Pipeleers, "A penalty method based approach for autonomous navigation using nonlinear model predictive control," in 6th *IFAC Conf. Nonlin. Mod. Pred. Contr.*, 2018.

[19] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Autonomous Robots*, vol. 33, no. 1-2, pp. 21–39, 2012.

[20] J. Jackson, G. Ellingson, and T. McLain, "Rosflight: A lightweight, inexpensive MAV research and development tool," in *Int. Conf. Unman. Aircr. Sys. (ICUAS)*, June 2016, pp. 758–762.

[21] E. Fresk and G. Nikolakopoulos, "A generalized frame adaptive MPC for the low-level control of UAVs," in *European Control Conference, Cyprus, Limasson, 12-15 June, 2018*, 2018.

[22] B. Øksendal, *Stochastic Differential Equations: An Introduction with Applications*, 6th ed. Springer, 2006.

[23] J. C. Dunn and D. P. Bertsekas, "Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems," *J. Optim. Theory & Appl.*, vol. 63, no. 1, pp. 23–38, 1989.

[24] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathem. Progr. Comput.*, 2018.

[25] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer New York, 2006.

[26] D.-H. Li and M. Fukushima, "On the global convergence of the BFGS method for nonconvex unconstrained optimization problems," *SIAM Journal on Optimization*, vol. 11, no. 4, pp. 1054–1064, jan 2001.

[27] P. Patrinos, P. Sopasakis, H. Sarimveis, and A. Bemporad, "Stochastic model predictive control for constrained discrete-time Markovian switching systems," *Automatica*, vol. 50, no. 10, pp. 2504–2514, Oct 2014.

[28] P. Sopasakis, D. Herceg, A. Bemporad, and P. Patrinos, "Risk-averse model predictive control," *Automatica*, vol. 100, pp. 281–288, 2019.

[29] P. Sopasakis, M. Schuurmans, and P. Patrinos, "Risk-averse risk-constrained optimal control," in *IEEE ECC*, Naples, Italy, 2019.