Optimal drone overtaking using Model Predictive Control

Akash John Subash
M.Sc. Embedded Systems Engineering
Albert-Ludwig's University Freiburg
MatrNr. 5253501

***Abstract*—This project report describes the formulation of an optimal overtake aerial manoeuvre computed in real time using Model Predictive Control (MPC). Limited implementation details in python with CasADi and the crazyFlie 2.1 drone are also mentioned.**

I. Introduction

The Crazyflie 2.1 quadcopter is a versatile open source flying development platform with expansion decks that make it a good candidate for university projects. The optimal controls computed offboard using MPC are fed to the crazyFlie, with the intention to overtake a second quadcopter (henceforth referred to as obstacle).

The structure of the report is as follows: Section II describes the model and the dynamics of the crazyFlie 2.1 quadcopter using Ordinary Differential Equations (ODE). Section III shows the optimal control problem, it's non-linear program and section IV, V detail the control topologies in open and closed-loop. Section VI presents the results of simulation and flight tests. Finally, section VII, VIII acknowledges the resources which were referred to, and the guidance received during the course of the project.

II. System Modelling

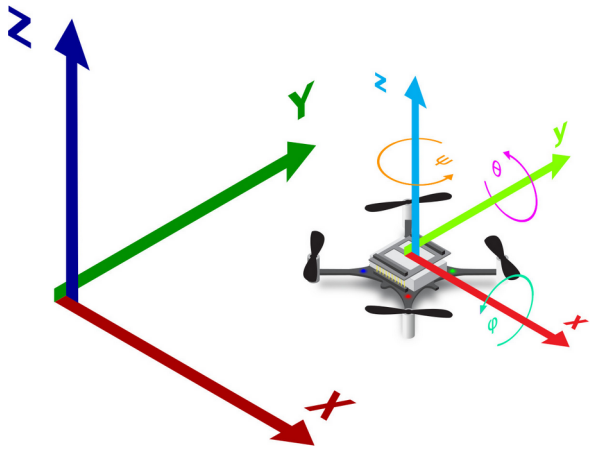

Fig. 1. Reference coordinate system [1]

As depicted in Fig. 1, a body frame located at the centre of mass of the quadcopter aligned with the East-North-Up inertial frame is used here, with φ, θ, ψ representing roll, pitch and yaw in the Euler notation respectively.

A. Physical model of the drone

The state $\zeta \in \mathbb{R}^{13}$ of the drone can be described by position $p = (x, y, z)^{\mathrm{T}}$, attitude in Quaternion notation $q = (q_w, q_x, q_y, q_z)^{\mathrm{T}}$, linear velocities $v = (v_x, v_y, v_z)^{\mathrm{T}}$, and angular velocities $\omega = (\omega_x, \omega_y, \omega_z)^{\mathrm{T}}$. It's non-linear dynamics are given by the ODE, which are adapted from [2] and [3], making changes to describe attitude in Quaternion instead of Euler notation.

$$\dot{\zeta} = f(\zeta, \mathrm{u}) = \begin{bmatrix} Sv \\ \frac{1}{2} q \times \omega \\ \frac{1}{m} F - S^{\mathrm{T}} g 1_z - q \times v \\ J^{-1}(M - \omega \times J\omega) \end{bmatrix}$$

The model parameters are described in Table I. The quaternion rotation matrix $S \in \mathbb{R}^{3\times3}$, external force matrix $F \in \mathbb{R}^{3\times1}$, external momentum matrix $M \in \mathbb{R}^{3\times1}$, and inertial matrix $J \in \mathbb{R}^{3\times3}$ are described below.

$$S = \begin{bmatrix} 2({q_w}^2 + {q_x}^2) - 1 & 2(q_x q_y - q_w q_z) & 2(q_w q_y + q_y q_z) \\ 2(q_w q_z + q_x q_y) & 2({q_w}^2 + {q_y}^2) - 1 & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_w q_x + q_y q_z) & 2({q_w}^2 + {q_z}^2) - 1 \end{bmatrix}$$

$$F = \begin{bmatrix} 0 \\ 0 \\ C_t({\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2) \end{bmatrix}$$

$$J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{1}{\sqrt{2}} d C_t(-{\Omega_1}^2 - {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2) \\ \frac{1}{\sqrt{2}} d C_t(-{\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 - {\Omega_4}^2) \\ C_d(-{\Omega_1}^2 + {\Omega_2}^2 - {\Omega_3}^2 + {\Omega_4}^2) \end{bmatrix}$$

Given the possibility to individually change the angular velocities of the propellers, the control vector is defined as: $u := (\Omega_1, \Omega_2, \Omega_3, \Omega_4)^{\mathrm{T}} \in \mathbb{R}^4$

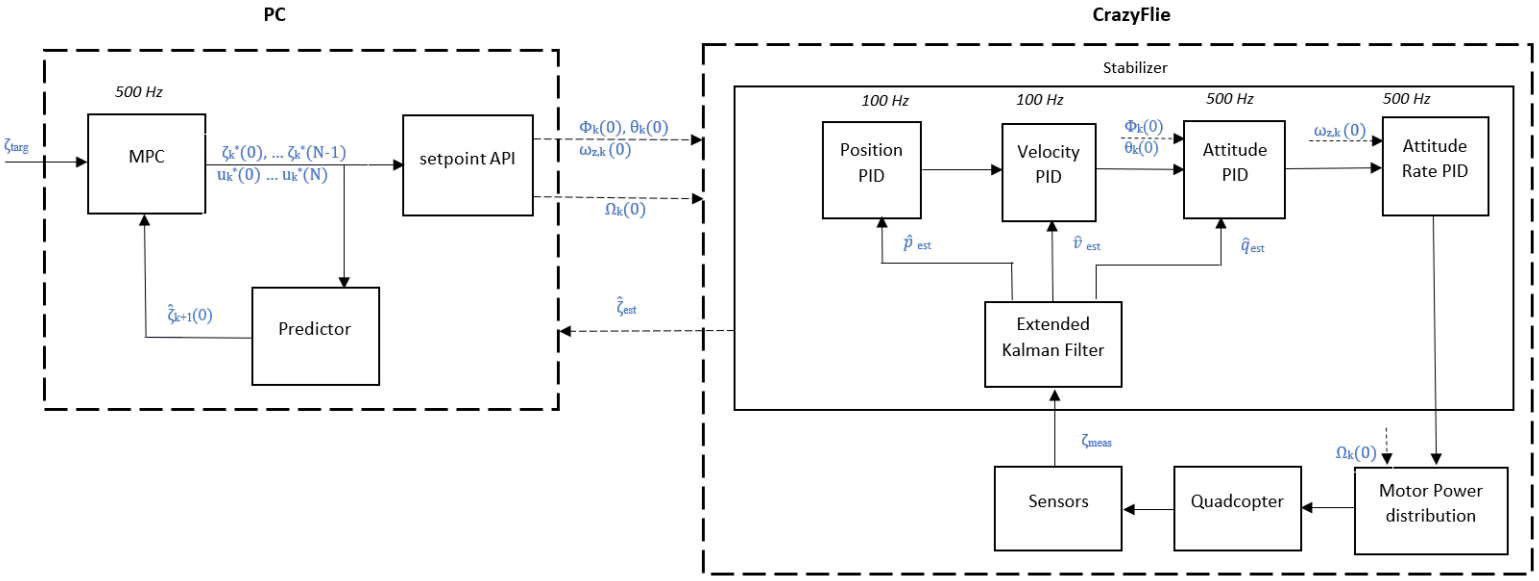

Fig. 2. Open loop control architecture

TABLE I
PARAMETERS OF THE MODEL

Symbol	Value	Unit	Description
m	31 x 10^{-3}	kg	Mass of the drone
d	46 x 10^{-3}	m	Length of the drone's arm
I_{xx}	1.395 x 10^{-5}	kg.m^2	Inertial moment around x-axis
I_{yy}	1.395 x 10^{-5}	kg.m^2	Inertial moment around y-axis
I_{zz}	2.173 x 10^{-5}	kg.m^2	Inertial moment around z-axis
C_d	7.93 x 10^{-12}	N.RPM^{-2}	Co-efficient of drag
C_t	3.25 x 10^{-10}	N.RPM^{-2}	Co-efficient of thrust
g	9.81	m.s^{-2}	Gravitational acceleration

B. State Predictor

A state predictor based on the explicit fourth order Runge-Kutta (RK-4) integrator was implemented to approximate the state evolution. Starting from the current measured state, forward iterations of the system dynamics are performed with an appropriate time step (τ_s) chosen to minimize the approximation error in the state evolution.

To solve the control problem in real time, it's important to consider the radio communication latency between the off-board processor running the MPC algorithm and the quad-copter. This impacts the delay in receiving measurements (τ_r) from the quadcopter and sending control commands to it (τ_t). Additionally, with an associated computational delay (τ_c) for calculating optimal solutions, the round trip delay of $\tau_{trr} = \tau_r + \tau_t + \tau_c$ is defined. While τ_r, τ_t are considered constants, the impact of variation of τ_c is discussed in the following chapters.

III. Problem Formulation

In this section, the discretized Optimal Control Problem (OCP) and it's constrained Non-Linear Program (NLP) are defined using direct multiple shooting [4] to discretize the underlying continuous time OCP.

A. Optimal Control Problem

The initial state is defined as the quadcopter hovering at a height z = 0.5 m, and the goal is to to reach a specified target state avoiding another quadcopter along the way, at position $p_{\mathrm{b}} = (x_{\mathrm{b}}, y_{\mathrm{b}}, z_{\mathrm{b}})^{\mathrm{T}}$

The discretized cost function is defined as :

$$L\left(\zeta_k, U_k\right) = \|\zeta_k - \zeta_{\mathrm{trg}}\|_Q^2 + \|u_k - u_{\mathrm{hov}}\|_R^2 \tag{1}$$

The OCP is then formulated as :

$$\min_{u_0,\zeta_0,..,u_{N-1},\zeta_N} \quad \sum_{k=0}^{N-1} L(\zeta_k, u_k) \tag{2}$$

$$\text{s.t} \quad \zeta_0 - \bar{\zeta_0} = 0 \tag{3}$$

$$\zeta_k - F(\zeta_k, u_k) = 0, \qquad k = 0, .., N-1 \tag{4}$$

$$3d \leqslant \|p_k - p_{\mathrm{b}}\| \leqslant \infty, \qquad k = 0, .., N-1 \tag{5}$$

$$\zeta_{\min} \leqslant \zeta_k \leqslant \zeta_{\max}, \qquad k = 0, .., N-1 \tag{6}$$

$$0 \leqslant u_k \leqslant U_{\max} \qquad k = 0, .., N-1 \tag{7}$$

$\bar{\zeta_0} = (0, 0, \frac{1}{2}, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T$
$\zeta_{\mathrm{trg}} = (1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T$
$\zeta_{\max} = (\frac{3}{2}, \frac{3}{2}, \frac{3}{2}, \infty, \infty, \infty, \infty, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{4})^T$
$\zeta_{\min}$ = -$\zeta_{\max}$
$U_{\max} = (22000, 22000, 22000, 22000)^T$
$p_{\mathrm{b}} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})^T$

Where $\zeta := (\zeta_0, .., \zeta_N)^T$, and $u := (u_0, .., u_{N-1})^T$ denote the state and control trajectories of the discrete time system. The discretized dynamics are given by the RK-4 integrator $F : \mathbb{R}^{n_\zeta} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\zeta}$ (Referred to as predictor in Figure 2). The horizon length and the estimated state of the system at the current time instant k are denoted by N, and ζ_k respectively.

$Q \in \mathbb{R}^{n_\zeta \times n_\zeta}$ and $R \in \mathbb{R}^{n_u \times n_u}$ are positive definite weighting matrices, tuned experimentally for the specific flight trajectory to be executed.

$Q = \text{diag}(120, 100, 100, 10^{-3}, 10^{-3}, 10^{-3}, 1, 1, 1, 1, 10^{-5}, 10^{-5}, 10^{-5})$
$R = \text{diag}(8 \times 10^{-1}, 8 \times 10^{-1}, 8 \times 10^{-1}, 8 \times 10^{-1})$

IV. Open-loop Control

The open-loop control of the drone without the obstacle to be avoided was implemented with a horizon of $N = 10$ in python using the CasADi [5] framework with IPOPT [6] as the interior point NLP solver. Using the next state predicted by the RK4 integrator from the current state and controls, the MPC computes the optimal solution iteratively. The optimal open-loop controls for each time could only be fed with minimal delay to the quadcopter, by solving the problem offline.

However, the large error propagation causing the quadcopter to quickly veer off the estimated trajectory, warrants the closed loop problem, and argues against placing the obstacle in the quadcopter's flight path in open loop. The appropriate set-point command API provided by the open source crazy-flie libraries allowed to communicate the controls as a set of roll, pitch, yaw rate, and thrust.

V. Closed-loop Control

Using the same solver framework, the estimated state from the Extended Kalman filter onboard the Crazyflie sampled at 50 Hz, was fed back to the MPC algorithm for closed loop control. To minimize radio communication latency arising from reception of multiple data frames, it was read in a compressed format via a single frame from the stabilizer module onboard and decoded on the PC. The sampling rate was determined by the lower bound on achievable computation time of the solver. Using MPC to control the system, an instance of (2) is solved iteratively at each sampling time point, with the current value of the state estimate ζ_k.

The closed loop architecture differs from Figure 2, only in the feedback path, where the predictor computes $\hat{\zeta}_{k+1}$ from $\hat{\zeta}_{\text{est}}$ instead of ζ_k^*. In closed loop, adding the obstacle to be avoided necessitated increasing the horizon to N = 20 for a smooth control, and state trajectory which led to a large increase in computation time (τ_c). In order to use the fedback state estimated, online iterations of the MPC was mandatory. As a consequence, the 500 Hz MPC computation rate could no longer be maintained and fell significantly to 50 Hz.

VI. Closed-loop Control Results

The most important factor to consider while moving from simulations to hardware flights were delays. The choice of τ_s was made to ensure the quadcopter's state, control trajectory evolution were plausible, while trying to maintain a minimal τ_c. A value of τ_s = 20 ms was found to provide a good trade-off between the two. The round trip delay τ_{trr} was found to be dominated by τ_c which is dependant on choice of the solver framework, and the integration step (τ_s) of the non-linear system dynamics, among other factors.

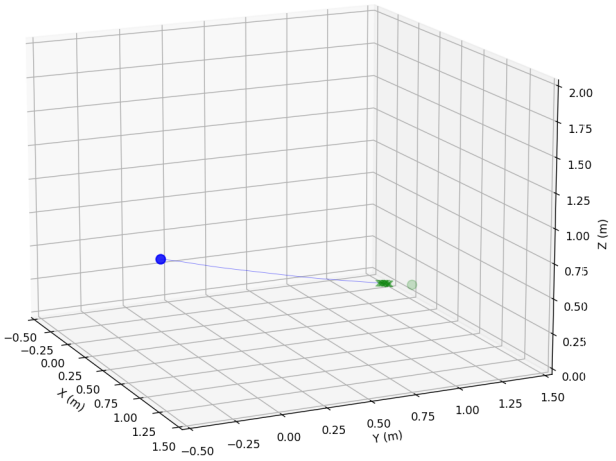

Fig. 3. Open loop trajectory without obstacle for N = 10, τ_s = 20ms

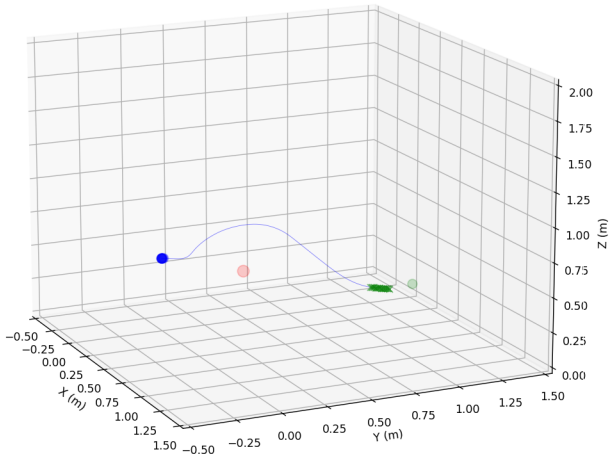

Fig. 4. Open loop trajectory with obstacle for N = 20, τ_s = 20ms

In the open, and closed loop flights without the obstacle, τ_{trr} was sufficiently small to observe stable flights. In closed loop, with the obstacle and N = 10, the quadcopter's attempt at aggressively avoiding the obstacle would cause the state estimated by the online stabilizer module to have large errors, arguing for a larger horizon length N. Choosing N = 20 provided a smooth control trajectories in simulation, but at the expense of a large τ_c which was no longer sufficient to control the quadcopter in a stable flight path.

The average iteration time achieved in closed-loop for computation without obstacle avoidance was τ_c = 40 ms, and τ_c = 100 ms with obstacle avoidance on an Intel i7-8750H processor @2.2Ghz running Windows. Future attempts could include reformulating the NLP as a sequential quadratic program (SQP) and using acados [7] with structure exploiting solvers to further reduce τ_c to achieve the overtake manoeuvre in real time.

VII. Acknowledgements

I would like to thank Mohammed Hababeh, Jakob Harzer, Andrea Ghezzi and Prof. Dr. Moritz Diehl for their supervision

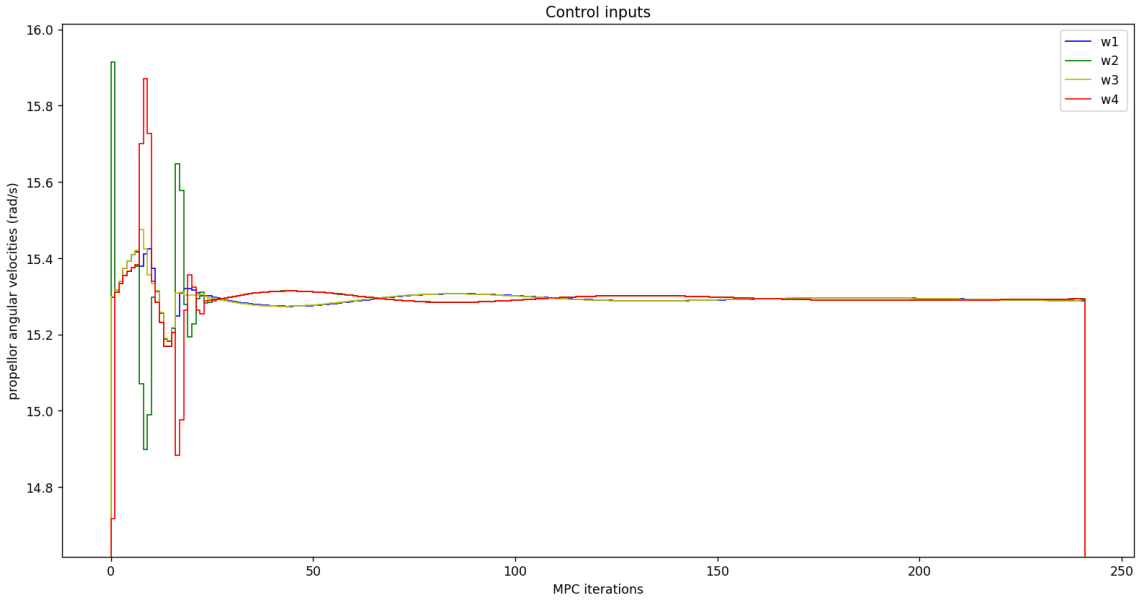

Fig. 5. Open loop controls without obstacle for N = 10, τ_s = 20ms, τ_c = 40ms

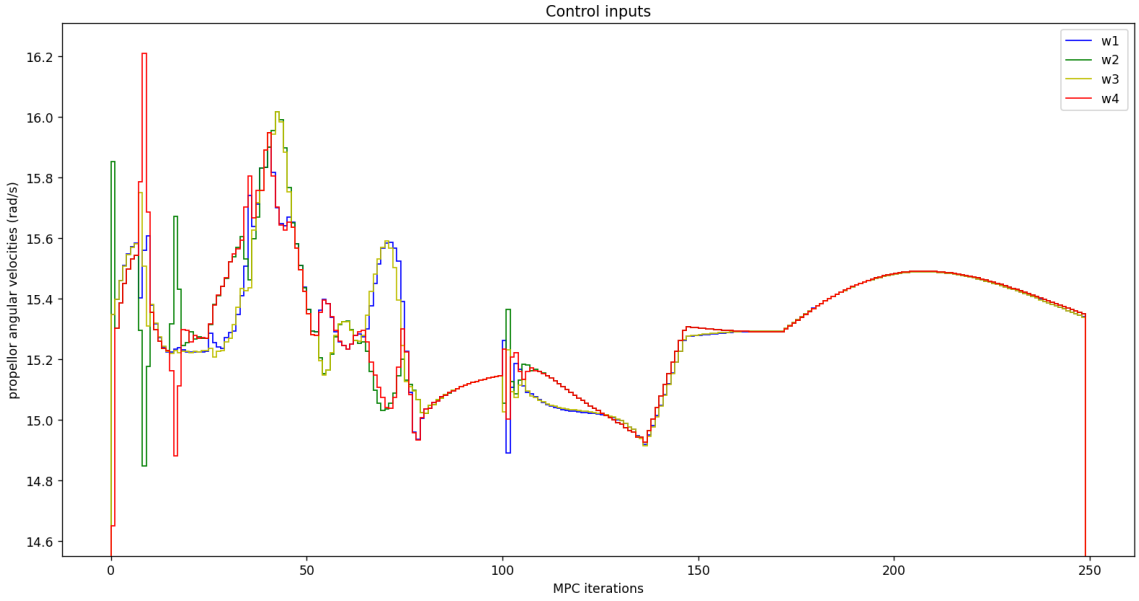

Fig. 6. Open loop controls with obstacle for N = 20, τ_s = 20ms, τ_c = 100ms

and guidance in this project.

VIII. References

[1] www.bitcraze.io/documentation/system/platform/cf2-coordinate-system/

[2] B. Barros Carlos, T. Sartor, A. Zanelli, G. Frison, W. Bugard, M. Diehl, G. Oriolo, "An Efficient Real-Time NMPC for Quadrotor Position Control under Communication Time-Delay" 2020. [Online]. Available:https://arxiv.org/abs/2010.11264

[3] C. Luis and J. L. Ny, "Design of a trajectory tracking controller for a nanoquadcopter," 2016. [Online]. Available: https://arxiv.org/abs/1608.05786

[4] H. G. Bock and K. J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," 9th IFAC World Congress, vol. 17, no. 2, pp. 1603–1608, 1984.

[5] Joel A E Andersson et al. "CasADi – A software frame- work for nonlinear optimization and optimal control". In: Mathematical Programming Computation 11.1 (2019), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[6] A. Wachter and L. T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large- scale

nonlinear programming". In: Mathematical Pro- gramming 106.1 (2006), pp. 25–57.

[7] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, B. Novoselnik, J. Frey, T. Albin, R. Quirynen, and M. Diehl, "acados: a modular open-source framework for fast embedded optimal control," 2019. [Online]. Available: https://arxiv.org/abs/1910.13753