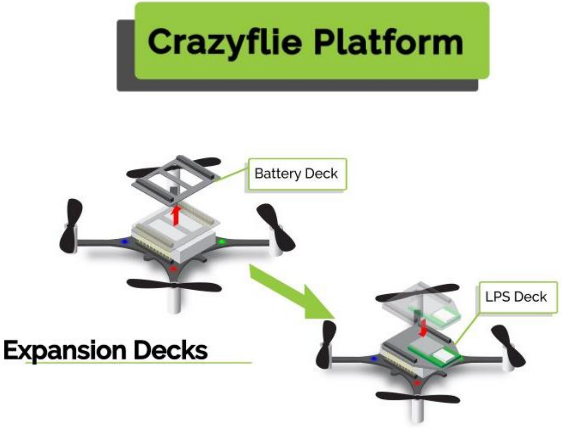
Obstacle Avoidance by Crazyflie drone using Model Predictive Control(MPC)

Albert-Ludwigs-Universität Freiburg

Presented By:
Vishal Sivakumar

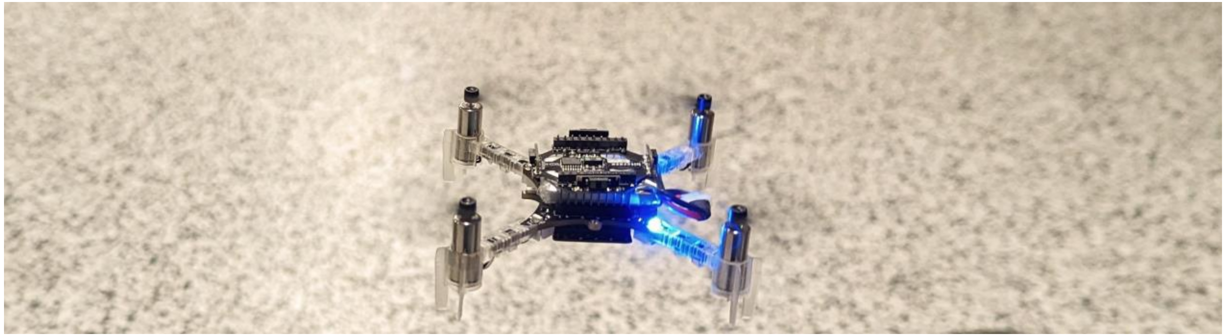
Contents

Drone overview



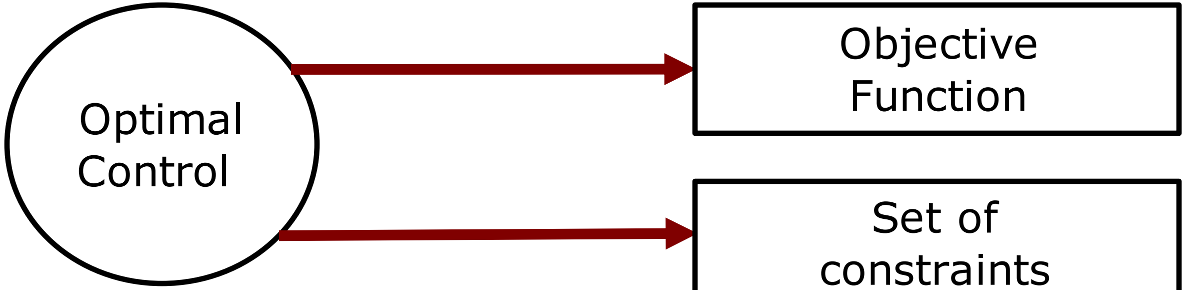
It consists of:

- STM32F4 that handles the main Crazyflie firmware with all the low-level and high-level controls.
- The NRF51822 handles all the radio communication and power management.



Approach using MPC

- **MPC** - A control strategy, predicts future behavior based on the model specified and provides optimized control inputs to achieve a desired performance.



Goal: It is to avoid obstacles based on set of constraints, provide optimal control inputs(velocity) to the drone and make it maneuver based on the state estimation using Lighthouse Positioning system.

Some definitions:

[Github link to this definition in code]

State vector:

$$\mathbf{x} = [x \; y \; z \; \psi \; v_x \; v_y \; v_z \; v_\psi]^T$$

Model:

$$\dot{x} = v_x cos(\psi) - v_y sin(\psi)$$
$$\dot{y} = v_x sin(\psi) + v_y cos(\psi)$$
$$\dot{z} = v_z$$
$$\dot{\psi} = v_\psi$$
$$\dot{v}_i = (-v_i + k_i u_i)/\tau_i, \quad i \in \{x, y, z, \psi\}$$

Objective functions

$$J^t = \frac{1}{2} \sum_{i=0}^{N-1} \|\mathbf{x}_i - \mathbf{x}_i^*\|_P^2 + \|\mathbf{x}_N - \mathbf{x}_N^*\|_Q^2$$

$$J^c = \frac{1}{2} \sum_{i=0}^{N-1} \|\mathbf{u}_i\|_R^2$$

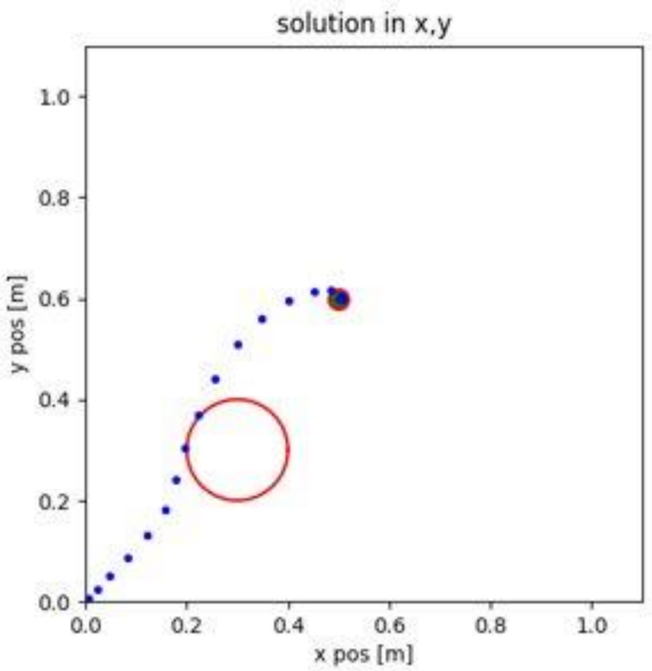
Static-Obstacle constraints

$$(p[0:2] - p0)^2 > (r0)^2 + 0.001$$

- Solver used: IPOPT (Interior Point OPTimizer)
- Control input limits: $-0.4 \leq u \leq 0.4$
- OCP horizon length: N = 5 - 15 control intervals
- Sampling time: dt = 0.1s or 100 msec
- Discretization method: multiple shooting
- Numerical integration method: Runge-Kutta 4th order

	k_i	τ_i (s)
x	1.0000	0.8355
y	1.0000	0.7701
z	1.0000	0.5013
ψ	$\pi/180$	0.5142

Simulation plot



Videos of Inferences

Movement from origin O(0, 0, 0) to a point A(0.5, 0.5, 0.5)

ClosedloopWithoutObstacle

Movement from origin O(0, 0, 0) to a point A(0.5, 0.5, 0.5) with an obstacle at (0.3, 0.2)

[ClosedloopWithObstacle]

Possible advancements

- Include Dynamic obstacle constraints.
- Use of different solvers and packages to produce a comparative results for better performance and latency, this could be for example SNOPT, KNITRO, APOPT, ACADOS and BONMIN.
- Inclusion of Utility theory(RL) considering similar finite horizon problems to have a reward based agent, environment interaction.
- Include multiple shooting with other techniques, such as direct collocation, to improve the accuracy and efficiency of the solution.

References

- Castillo-Lopez, M., Sajadi-Alamdari, S. A., Sanchez-Lopez, J. L., Olivares-Mendez, M. A., & Voos, H. (2018). Model Predictive Control for Aerial Collision Avoidance in Dynamic Environments. *Mediterranean Conference on Control and Automation*. https://doi.org/10.1109/med.2018.8442967

- meco-group. (n.d.). *GitHub - meco-group/rockit: Github Mirror of https://gitlab.mech.kuleuven.be/meco-software/rockit*. GitHub. https://github.com/meco-group/rockit

- *Robotics with MATLAB*. (n.d.). https://pab47.github.io/robotics.html

Discussion

Thank you !

Github link to this work: Flight-Control