Evasive Maneuvering for UAVs: An MPC Approach

Manuel Castillo-Lopez[1,2], Miguel A. Olivares-Mendez[2], and Holger Voos[2]

[1] University of Malaga, Malaga, Spain.
e-mail: manuelcloop@gmail.com,
[2] SnT-University of Luxembourg, Luxembourg, Luxembourg.
e-mail: miguel.olivaresmendez@uni.lu, holger.voos@uni.lu

Abstract. Flying autonomously in a workspace populated by obstacles is one of the main goals when working with Unmanned Aerial Vehicles (UAV). To address this challenge, this paper presents a model predictive flight controller that drives the UAV through collision-free trajectories to reach a given pose or follow a waypoint path. The major advantage of this approach lies on the inclusion of three-dimensional obstacle avoidance in the control layer by adding ellipsoidal constraints to the optimal control problem. The obstacles can be added, moved and resized online, providing a way to perform waypoint navigation without the need of motion planning. In addition, the delays of the system are cosidered in the prediction by an experimental first order with delay model of the system. Successful experiments in 3D path tracking and obstacle avoidance validates its effectiveness for sense-and-avoid and surveillance applications presenting the proper structure to extent its autonomy and applications.

Keywords: Unmanned Aerial Vehicle, Model Predictive Control, obstacle avoidance, sense and avoid, surveillance.

1 Introduction

Unmanned Aerial Vehicles (UAVs) are now a subject of active research due to its multiple applications such as traffic monitoring, load transportation and manipulation or search-and-rescue operations [1]. These applications often demand precise trajectories in a workspace populated by obstacles, increasing the importance of autonomous navigation. This is usually done by hierarchical decoupled planning and control, where a waypoint motion planner generates a collision-free path that the vehicle control system has to follow. A survey of different motion planning techniques applied to UAVs can be found in [2].

Unlike most classical methods, Model Predictive Control (MPC) make use of a model of the system to generate the control signal and the future trajectory at the same time by optimization. A given cost function is minimized over decision variables (including the current and the future states and controls over some prediction horizon) subjected to constraints on state and inputs. As the prediction horizon shifts forward in time, MPC is also called Receding Horizon Control (RHC). MPC is therefore able to naturally consider safety considerations and actuator saturations, as long as these can

be formulated as state and input constraints.

Trayectory generation for UAVs using MPC is becoming an attractive approach due to its potential to obtain the control signal considering the future state of the system, energy efficiency, obstacles and even uncertainty. In [3] an efficient MPC control scheme for quadrotors is presented to perform 3D path tracking. In [4] a robust predictive flight controller with obstacle avoidance capabilities is implemented. In [5] a Bayesian Policy Optimization with MPC is developed to provide stochastic collision avoidance for quadrotors. In [6] an Hybrid Predictive Controller is designed to interact phisically in inspection operations. Nonlinear Partial Enumeration with MPC is used in [7] to develop a fast MPC controller, tested in simulations. MPC is used in [8] for an aerial pick-and-place application with a manipulator attached to a quadrotor. Autonomous landing on a moving platform is developed in [9]. Reinforcement learning applied to MPC is used in [10] to reduce the computational cost when avoiding obstacles. In [11] a real time model predictive position control is implemented using sigmoid functions to model obstacles. Finally, in [12] Learning Based MPC is used to catch balls in the air and correct the ground effect.

In this work, a model predictive flight controller for UAVs is developed and tested using a Motion Capture System and a low-cost quadrotor helicopter tele-operated by a computer. The controller drives the UAV autonomously through collision-free trajectories to reach a given pose or follow a waypoint path. The prediction of the UAV behaviour integrates the system and communication delays using a First Order with Delay (FOD) model obtained by experimental identification. Unlike the previous work, the presented MPC controller performs three-dimensional avoidance of obstacles (modelled as ellipsoids) that can be added, moved and resized online. Successful experiments in 3D path tracking and obstacle avoidance validates its effectiveness for sense-and-avoid and surveillance applications, presenting the proper structure to extent its autonomy and applications.

This paper is structured as follows: Section 2 describes the hardware, software and methodology used for the experiments. Section 3 shows the experimental model used in chapter 4 to design the MPC approach. Finally, the results of the experiments are shown in section 5, including the discussing of the conclusions and future work in section 6.

2 Methodology and Equipment

The first phase is to formulate the MPC problem and create a program to solve it iteratively. In this work, ACADO Toolkit [13] is used to develop an MPC solver for UAVs. A computer[3] running ROS[4] is used to implement the MPC controller based on the previous MPC solver. The communications with the different targets (UAV, simulator,

[3] Lenovo Y50-70 laptop with Intel® i7-4710HQ CPU at 2.50 GHz and 8 GiB of DDR3 1600 MHz RAM.

[4] ROS Indigo [14] (The Robot Operating System) under Ubuntu 14.04

sensors, Motion Capture System, etc.) are implemented through a publisher/subscriber messagin pattern, making use of the *snt_ardrone_driver*[5] .

V-REP [16] is used to simulate the behaviour of the UAV using a physical model that interacts with the simulated environment (see Figure 1). Using the ROS bridge, it exanges information in real time with ROS as it would do in real experiments e.g. commands from the MPC controller, UAV and obstacles pose, etc.

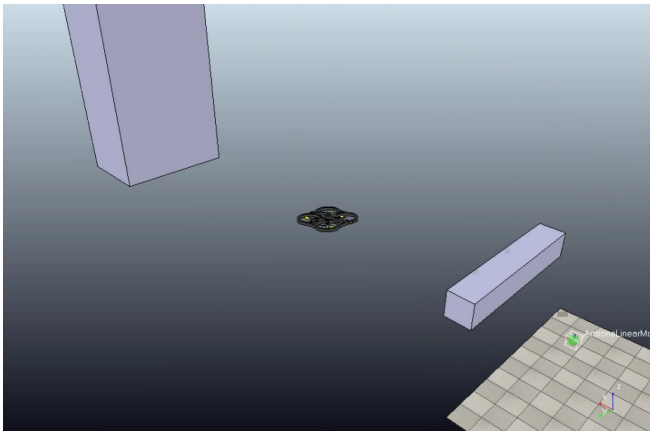
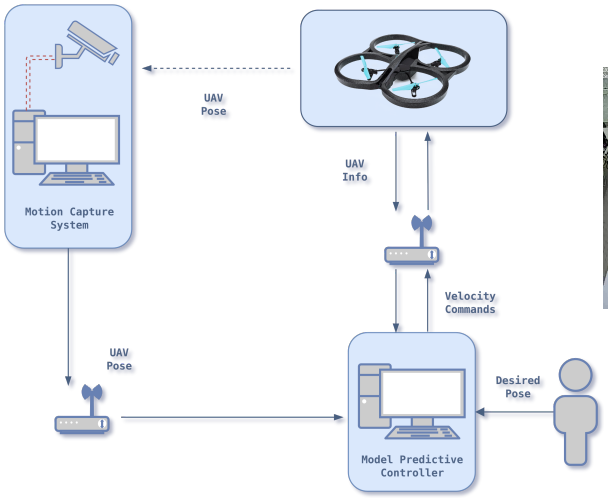


Fig. 1: VREP simulation environment used to test the MPC controller

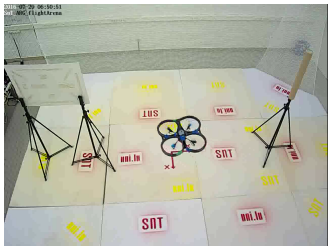
In this work, the chosen platform is the UAV Parrot® AR.Drone 2.0, i.e. a small quadcopter designed to be controlled remotely through WiFi. The UAV operates in a flight area limited by nets as shown in Figure 2a. Reflecting balls attached to the UAV are used to determine its position and orientation using the OptiTrack® Motion Capture System.

With the elements described before, the control architecture is designed as shown in Figure 2 to test the Model Predictive Controller developed in this work. The main computer runs the MPC Controller and is connected to the Motion Capture System and the UAV through Ethernet and WiFi respectively. The Motion Capture System publishes continuously the position and orientation of the UAV, while the MPC Controller process the data and obtains the control signal, which is sent to the UAV. The desired pose can be published manually or using a program to create the path to follow.

[5] A ROS package based on the official AR-Drone SDK to control the UAV [15]



(a) Control Architecture



(b) UAV Flight Arena

Fig. 2: Experimental testbed

3 UAV Model

Most quadrotors are designed with an internal controller that assures its stability while following velocity commands recieved from a pilot (automated or human). Typically, the input vector $\bar{u} = [u_f \ u_s \ u_u \ u_h]^T$ is made by four commands: forward, sideward, upward and heading velocities based on the body frame of the UAV. In this work, a right-handed frame is considered as shown in Figure 3.

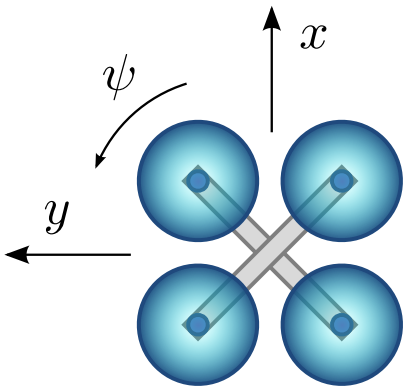


Fig. 3: Body frame of the UAV

As the inner controller and other specifications of the UAV may be unknown, an experimental first order with delay (FOD) model is defined in equation 1 to relate each component of the input vector $\bar{u}$ to the UAV velocity vector $\bar{v} = [v_f\ v_s\ v_u\ v_h]^T$, where K_i and d_i are the gain and the delay associated to v_i and u_i.

$$\dot{v}_i(t) = (-v_i(t) + K_i u_i(t - d_i))/\tau_i \tag{1}$$

The selected state vector is composed by the UAV cartesian coordinates (x, y, z), its orientation at hovering ψ, and the velocity vector $\bar{v}$ as shown in equation 2.

$$\bar{x}(t) = \left[x(t)\ y(t)\ z(t)\ \psi(t)\ v_f(t)\ v_s(t)\ v_u(t)\ v_h(t)\ \right]^T \tag{2}$$

Then, the state space model can be written as

$$\dot{\bar{x}}(t) = \bar{\bar{M}}\bar{x}(t) + \bar{\bar{N}}\bar{u}(t - \bar{d})$$

Where

$$\bar{\bar{M}} = \begin{bmatrix} & c\psi & -s\psi & 0 & 0 \\ & s\psi & c\psi & 0 & 0 \\ & 0 & 0 & 1 & 0 \\ & 0 & 0 & 0 & 1 \\ \bar{\bar{0}}_{8\times4} & -\tau_f^{-1} & 0 & 0 & 0 \\ & 0 & -\tau_s^{-1} & 0 & 0 \\ & 0 & 0 & -\tau_u^{-1} & 0 \\ & 0 & 0 & 0 & -\tau_h^{-1} \end{bmatrix} \quad \bar{\bar{N}} = \begin{bmatrix} & \bar{\bar{0}}_{4\times4} & & \\ K_f/\tau_f & 0 & 0 & 0 \\ 0 & K_s/\tau_s & 0 & 0 \\ 0 & 0 & K_u/\tau_u & 0 \\ 0 & 0 & 0 & K_h/\tau_h \end{bmatrix}$$

Note that ψ is considered constant to keep $\bar{\bar{M}}$ and $\bar{\bar{N}}$ time invariant. Thus, the model can be discretized as shown in equation 3 and the delay approximated by $\bar{n}_d = \bar{d}/\Delta t$.

$$\bar{x}_{i+1} = \bar{\bar{A}}\bar{x}_i + \bar{\bar{B}}\bar{u}_{i-\bar{n}_d} \tag{3}$$

Where

$$\bar{\bar{A}} = e^{M\Delta t} \qquad \bar{\bar{B}} = (e^{M\Delta t} - I)\bar{\bar{M}}^{-1}\bar{\bar{N}} \tag{4}$$

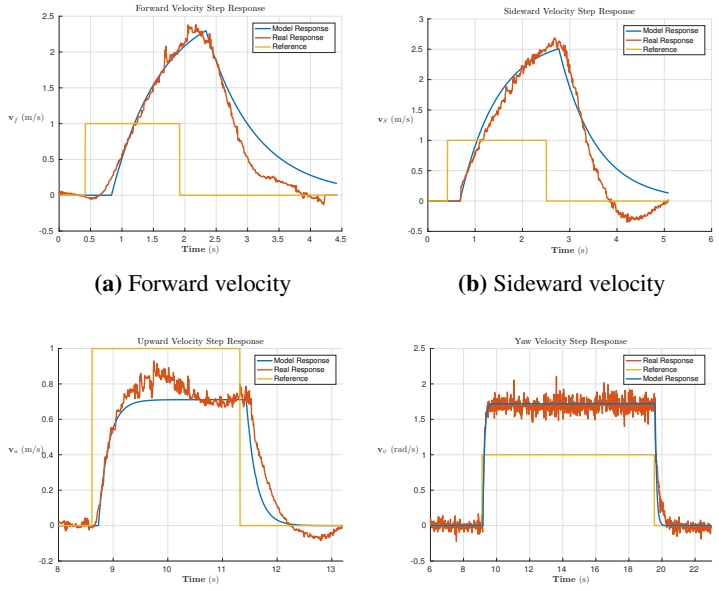
A classical step response tangent method (equation 5) is used to identify the first order with delay model of the AR.Drone (see table 1). The experimental testbed described in chapter 2 is used for the identification, where the Motion Capture System is used to get the UAV pose and the commands where sent manually from the laptop through a WiFi router.

$$K = \frac{v(\infty)}{u(\infty)} \qquad \tau = \frac{3}{2}(t_{63} - t_{28}) \qquad d = t_{63} - \tau \tag{5}$$

	K	τ (s)	d (s)
v_f/u_f	2.7000	0.7889	0.4164
v_s/u_s	2.7000	0.7889	0.2600
v_u/u_u	0.7110	0.1815	0.1148
v_h/u_h	1.7200	0.0912	0.0483

Table 1: Parameters of the FOD model for AR.Drone 2.0

As shown in Figure 4, the FOD model presents a consistent prediction of the UAV behaviour. Even though the upward velocity response presents a second order response, the FOD model is kept to reduce the computational load and keep highly stable solutions.



(a) Forward velocity **(b)** Sideward velocity

(c) Upward velocity **(d)** Heading velocity

Fig. 4: UAV step response (real vs modelled)

4 MPC Controller Design

Making use of the linear state space model developed in chapter 3 an Optimal Control Problem (OCP) is defined in the set of equations 6. The equation 6a defines the objective function, where $\bar{x}_i^*$ and $\bar{u}_i^*$ are the desired values for the state and input vectors

respectively at step i. The weighting matrices P, Q and R penalize the difference between the desired and the real values of the states and control inputs. The equation 6c adds the discrete state space model as a constraint of the problem, setting the limits of the state and inputs variables in equation 6d. Finally, the equation 6e introduces an obstacle modeled as an ellipsoid situated in (x_o, y_o, z_o) with its respective radius (r_x, r_y, r_z).

$$\min_{X,U} J(X,U) = \frac{1}{2}\sum_{i=0}^{N-1}\left(\|\bar{x}_i - \bar{x}_i^*\|_Q^2 + \|\bar{u}_i - \bar{u}_i^*\|_R^2\right) + \|\bar{x}_N - \bar{x}_N^*\|_P^2 \tag{6a}$$

subject to: (6b)

$$\text{Model:} \quad \bar{x}_{i+1} = \bar{\bar{A}}\bar{x}_i + \bar{\bar{B}}\bar{u}_{i-\bar{n}_d} \tag{6c}$$

$$\text{Limits:} \quad \bar{x}_{min} \leq \bar{x}_i \leq \bar{x}_{max} \qquad \bar{u}_{min} \leq \bar{u}_i \leq \bar{u}_{max} \tag{6d}$$

$$\text{Obstacle:} \quad 1 \leq \frac{(x_i - x_o)^2}{r_x^2} + \frac{(y_i - y_o)^2}{r_y^2} + \frac{(z_i - z_o)^2}{r_z^2} \tag{6e}$$

Thus, given the initial values of the OCP variables, it generates the future controls $U = [\bar{u}_0 \ldots \bar{u}_{N-1}]$ that makes the UAV follow the desired state $(\bar{x}_i^*)$ through a collision-free trajectory $X = [\bar{x}_0 \ldots \bar{x}_N]$ that minimizes the objective function $J(X,U)$ over a prediction horizon $i = [0, \ldots, N]$. Note that the time horizon depends on the sample time of the model with $t_H = [0, \ldots, N] \cdot \Delta t$

ACADO Code Generation Tool for C++ is used to create a standalone solver for the previous OCP with Sequential Quadratic Programming (SQP), i.e. an algorithm to transform a Non Linear Program (NLP) into multiple Quadratic Programs (quadratic cost function and linear constraints), which are simpler to solve. Each Quadratic Program (QP) is solved using a Gauss-Newton with active-set method, which evaluates the constraints that really affects (active constraints) to the problem. See [17] for a detailed description of the ACADO SQP algorithm.

In order to be generalistic, a decentralized architecture has been developed for the MPC Controller operation as shown in Figure 5. Thus, only new interfaces needs to be programmed in case of hardware changes and most of the parameters can be modified online.

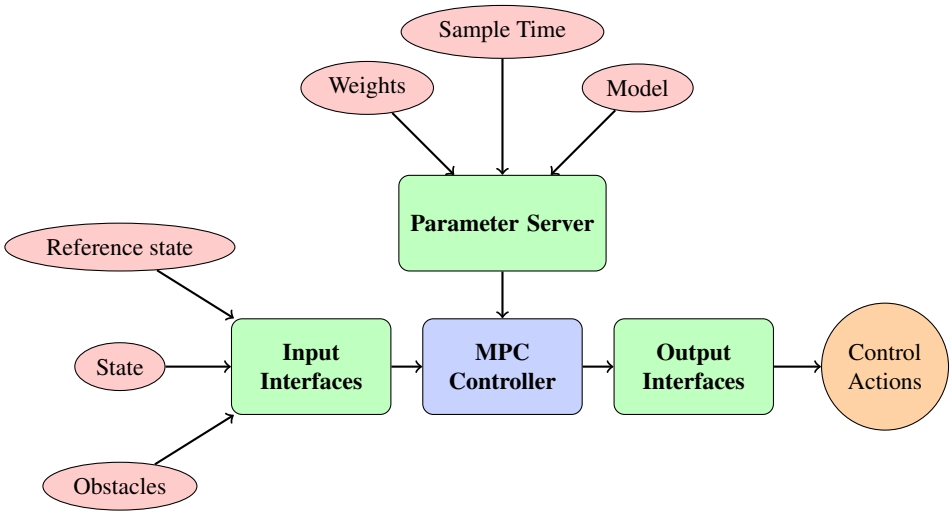


Fig. 5: MPC Architecture

5 Experiments and Results

The configuration of the MPC controller is shown in table 2. A reduced number of control intervals are used to provide a less optimal but fast solution that generates less aggressive trajectories. The MPC sample time is much greater than the MPC control delay, which frees the CPU to sleep to save energy or attend other processes.

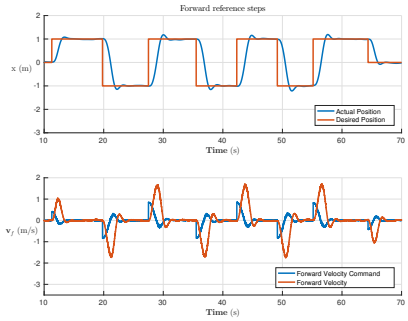
Prediction horizon	4 s
Control intervals	4
Simulation intervals	32
Integrator type	Runge-Kutta 4
Maximum velocity	1 m/s
MPC sample time	10 ms
MPC control delay	0.46 ms

Table 2: MPC controller configuration

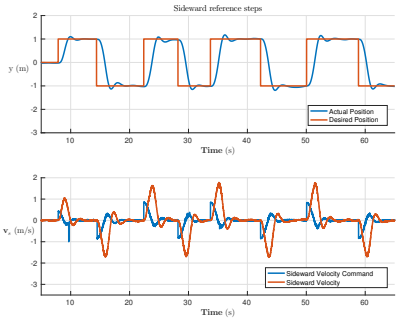
The weighting matrices $P = Q = I\left[200\ 200\ 100\ 300\ 100\ 100\ 100\ 100\right]^T$ and $R = I\left[600\ 600\ 50\ 200\right]^T$ have been manually tuned to priorize energy saving movements over precise positioning.

Using the experimental testbed described in chapter 2, the step response of the controller is tested. Initially, the UAV stands in $(x, y, z, \psi) = (0, 0, 1, 0)$ (S.I.). As shown in Figure 6, the reference pose is changed alternatively for each input, obtaining a stable

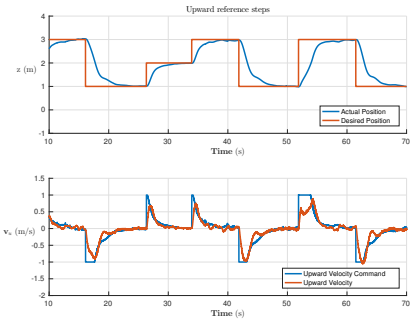
and smooth response of the MPC Controller. Note the unstabilities presented by the AR.Drone height controller that makes our controller less precise in z axis. In addition, the agressivity can be adapted online by changing the weighting matrices P, Q and R.



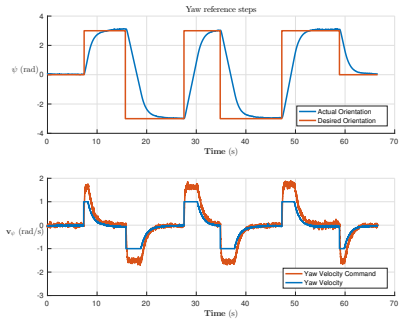
(a) Forward step response



(b) Sideward step response



(c) Upward step response



(d) Heading step response

Fig. 6: Reference steps response of the UAV driven by the MPC Controller

To test the surveillance and obstacle avoidance capabilities of the MPC Controller, we define a squared waypoint path to be followed at 0.5 m/s with two ellipsoidal obstacles with the parameters shown in table 3. In Figure 8 the trajectory of the UAV and the modelled obstacles are represented precisely using the raw data from the experiment during 3 rounds. For a better visualization of the experiment, the video recorded during the experiment is processed to obtain the UAV trajectory represented by its centroid in the image plane, as shown in Figure 7.

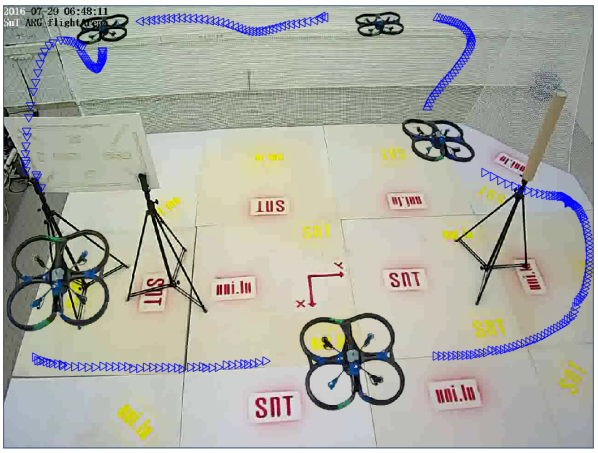


Fig. 7: UAV trajectory when performing the path tracking with obstacles experiment

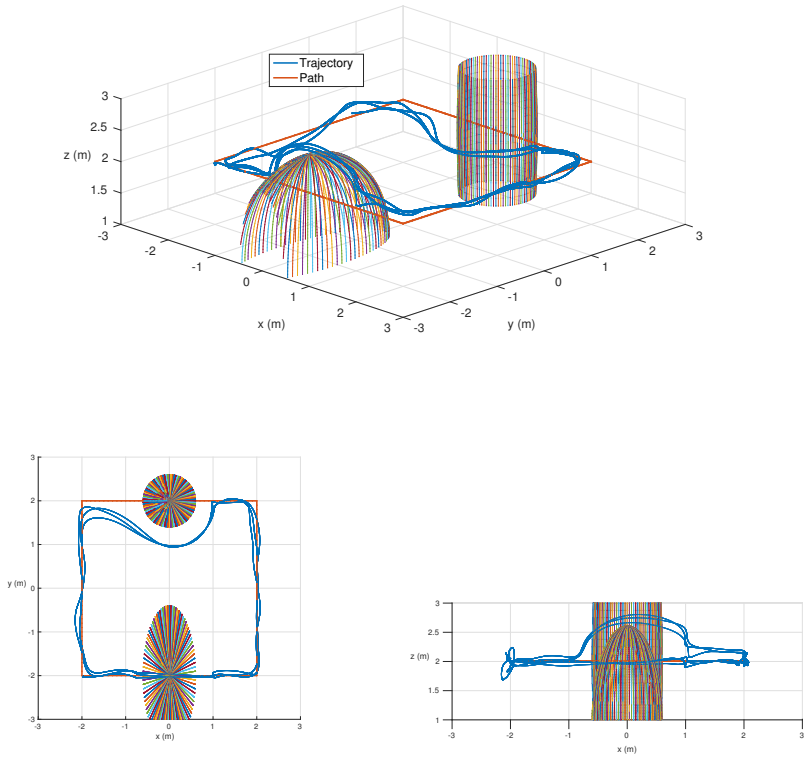


Fig. 8: Trajectory of the UAV while performing a real path tracking with obstacles experiment. Isometric, top and front view respectively.

	Obstacle 1	Obstacle 2
(x,y,z)	$(0,-2,1)$	$(0,2,2)$
(r_x,r_y,r_z)	$(1,2,2)$	$(1,1,10^4)$

Table 3: Position and shape of the ellipsoidal model of the obstacles in meters

6 Conclusion and Future Work

The position controller developed in this work is a general purpose Model Predictive Control approach for UAVs. Is suitable to be used with different platforms (parametric model) in a wide variety of missions e.g, surveillance, obstacle avoidance, autonomous landing (not tested), etc. The control can be implemented in on-board and tele-operated systems and can be tuned for an aggressive or soft response. It is also lightweight enough for small UAV applications, having 0.46 *ms* of average control delay with the experimental testbed described in chapter 2.

Different developement lines can be traced from this work. One could improve its autonomy using Simultaneous Localization and Mapping for unkown environments or implementing autonomous landing. Due to its parametric and modular structure, it is also possible to include agressivity control, deep learning, obstacle dynamics and any other processes running alongside the MPC controller.

Acknowledgements

This work was supported by the "Fonds National de la Recherche" (FNR), Luxembourg, under the project C15/15/10484117 (BEST-RPAS).

References

1. Valavanis, K.P., Vachtsevanos, G.J.: Handbook of unmanned aerial vehicles, Section XX: UAV Applications. Springer Publishing Company, Incorporated (2014)
2. Goerzen, C., Kong, Z., Mettler, B.: A survey of motion planning algorithms from the perspective of autonomous uav guidance. Journal of Intelligent and Robotic Systems **57**(1-4), 65–100 (2010)
3. Abdolhosseini, M., Zhang, Y., Rabbath, C.A.: An efficient model predictive control scheme for an unmanned quadrotor helicopter. Journal of intelligent & robotic systems pp. 1–12 (2013)
4. Alexis, K., Papachristos, C., Siegwart, R., Tzes, A.: Robust model predictive flight control of unmanned rotorcrafts. Journal of Intelligent & Robotic Systems **81**(3-4), 443–469 (2016)
5. Andersson, O., Wzorek, M., Rudol, P., Doherty, P.: Model-predictive control with stochastic collision avoidance using bayesian policy optimization. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on, pp. 4597–4604. IEEE (2016)

6. Darivianakis, G., Alexis, K., Burri, M., Siegwart, R.: Hybrid predictive control for aerial robotic physical interaction towards inspection operations. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 53–58. IEEE (2014)
7. Desaraju, V.R., Michael, N.: Fast nonlinear model predictive control via partial enumeration. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on, pp. 1243–1248. IEEE (2016)
8. Garimella, G., Kobilarov, M.: Towards model-predictive control for aerial pick-and-place. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 4692–4697. IEEE (2015)
9. Vlantis, P., Marantos, P., Bechlioulis, C.P., Kyriakopoulos, K.J.: Quadrotor landing on an inclined platform of a moving ground vehicle. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 2202–2207. IEEE (2015)
10. Zhang, T., Kahn, G., Levine, S., Abbeel, P.: Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on, pp. 528–535. IEEE (2016)
11. Dentler, J., Kannan, S., Mendez, M.A.O., Voos, H.: A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors. In: Control Applications (CCA), 2016 IEEE Conference on, pp. 519–525. IEEE (2016)
12. Bouffard, P., Aswani, A., Tomlin, C.: Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 279–284. IEEE (2012)
13. Houska, B., Ferreau, H., Vukov, M., Quirynen, R.: ACADO Toolkit User's Manual. http://acado.github.io (2009–2013)
14. Garage, W.: ROS. The robot operating system. URL `www.ros.org`
15. Olivares-Mendez, M.A., Kannan, S., Voos, H.: Setting up a testbed for uav vision based control using v-rep & ros: A case study on aerial visual inspection. In: Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, pp. 447–458. IEEE (2014)
16. Rohmer, E., Singh, S.P.N., Freese, M.: V-rep: a versatile and scalable robot simulation framework. In: Proc. of The International Conference on Intelligent Robots and Systems (IROS) (2013)
17. Quirynen, R., Vukov, M., Zanon, M., Diehl, M.: Autogenerating Microsecond Solvers for Nonlinear MPC: a Tutorial Using ACADO Integrators. Optimal Control Applications and Methods (2014)