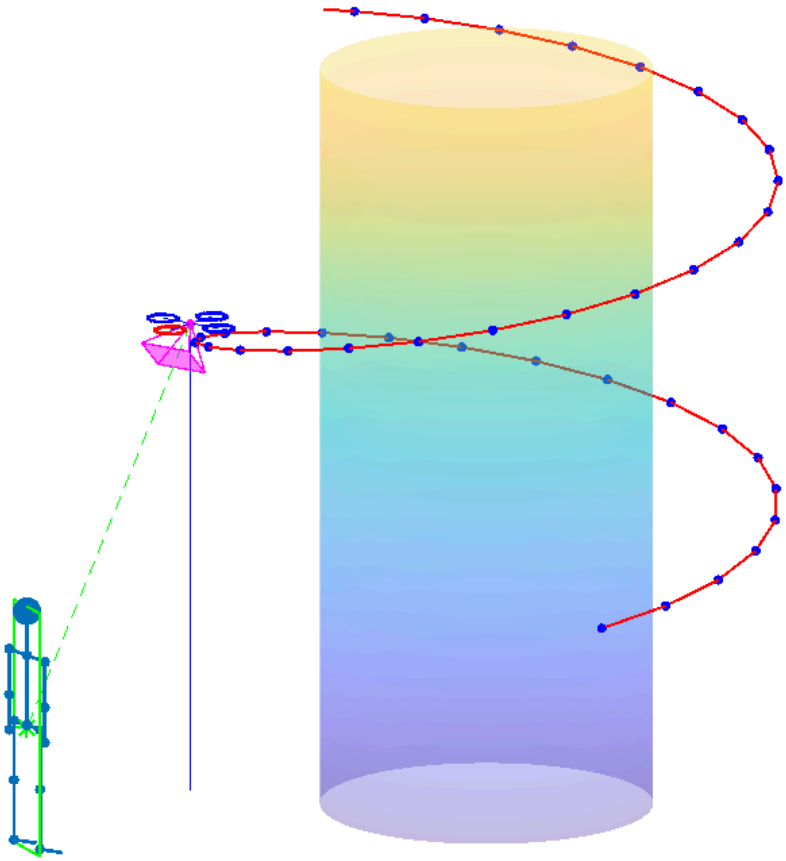
FS 2024-2025, RO-II, DISAL-SP194

Start: 09/09/2024
End: 10/01/2025

VLOS Inspection for MAVs

Alexandre Hébert (368443)



Professor: Alcherio Martinoli
Assistant: Kagan Erünsal

Contents

Figures

Tables

1. Introduction

Visual Line-Of-Sight (VLOS) flight of a Micro Aerial Vehicle (MAV) requires a remote pilot/observer to clearly see the vehicle and the surrounding airspace at all times while it is airborne. In Switzerland, for the "Open" category, VLOS should always be maintained with the vehicle [1]. Although it is possible to apply for a Beyond VLOS (BVLOS) certification in the "Specific" category for fully autonomous flight, this procedure is rather complex and takes a long time.

This project aims to develop a control and estimation strategy to perform autonomous inspections of MAVs while considering VLOS constraints. In particular, the MAV should detect the observer by using its sensors, estimate its position while performing inspection tasks, and adjust its trajectories to help the remote pilot maintain VLOS conditions. It is planned to leverage optimization-based methodologies [2], [3] for motion planning such as Model Predictive Control (MPC). The algorithms will be developed and tested in a high-fidelity robotic simulator (Webots) using the Robot Operating System (ROS) in environments where the complexity of inspection regions changes. Various metrics will be defined to evaluate the performance of inspection and quality of VLOS flights.

2. Literature review

2.1. Perception-Aware MPC

Our inspection tasks consist of two main objectives:

- Inspection of an asset
- Keep VLOS condition with an operator

Multi-objective MPC problem formulation have been explored in the scientific community:

- [4] developed a perception model to keep a feature of interest in the middle of the camera's FOV while minimizing its velocity in the image plane. Doing so, they can achieve smooth tracking with minimal motion blur. A combined MPC model is defined with the dynamic and perception objectives weighted with time dependent cost matrices Q. Usual input constraints are defined.
- [5] identified two issues with standard approaches: control is decoupled from vision and collision avoidance is decoupled from inspection task. This results in poor planning and efficiency with conflicting goals. The author developed a MPC approach that couples perception-awareness with a probabilistic obstacle avoidance constraint, which uses a weight to define the priority of the two objectives and prevent infeasibility issues.

2.2. Other optimization approaches

Many tried using other optimization approaches such as QP with cost minimization or A*. However, these methods have one main drawback in common: they usually do not consider constraints in the problem formulation, which can lead to practical infeasibility (actuator saturation, lost of VLOS condition, etc.).

- [6] presents the *MOAR* planner, a Risk-Aware path planning algorithm derived from another know technique called Multi-Objective Path Planning (MOPP). They added risk-awareness to perform real-time trajectory adjustment based on potential collision risks. However, the authors do not use any quadratic programming nor MPC formulation. They instead utilize graph search algorithms such as A* to find optimal path by dividing space into a finite grid. Thus, this approach lacks certainty for maintaining VLOS conditions.

- [7] presents the *SVPTO* framwork to overcome two issues of traditional target tracking and other *PAMPC* approaches. The first issue is the usage of only the distance between target and nearest obstacle to predict VLOS conditions, which does not work well within complex environments. The second issue is with the orientation of the drone to maintain target focus. This could potentially lead to inaccuracies in obstacles detection. With their approach, authors use a QP solver with well define objective costs to mitigate the abovementioned issues. They add occlusions prediction cost and environment complexity estimations to define the optimal distance between target and drone to limit lost of VLOS. However, since no MPC is used, such VLOS conditions are never guaranteed!

2.3. Human-in-the-loop

Human-in-the-loop refers to an operator being part of the drone operations, usually when dealing with semi-autonomous operations.

In [8], a novel approach for controlling the drone using motion and augmented reality is proposed. The camera feed from the drone is coupled with the VR headset vision to allow see-through walls capabilities (see Figure 1) enabling easier and smoother operations of the drone. Pilot can specify with hand gestures the point of interest to be inspected by the drone.

Another approach to semi-autonomous inspections is to provide a handheld device (such as a laser pointer [9]) used to define the target for inspection. The drone can therefore autonomously perform the rest of the task by positioning the camera and lighting (see Figure 1). Operator can, at anytime, abort the process (which is an important aspect in VLOS operations).

Finally, others find novel ways to utilize simple drone vision (monocular cameras) to perform human pose tracking. [10] propose a framework to automatically take specific selfies by template matching using only GPS and visual tracking (see Figure 2). While still in a development phase, this shows how interaction human-to-drone interactions can have many different meanings.

For my project, human detection and tracking are two essentials components to include in the optimization framework to maintain VLOS conditions.

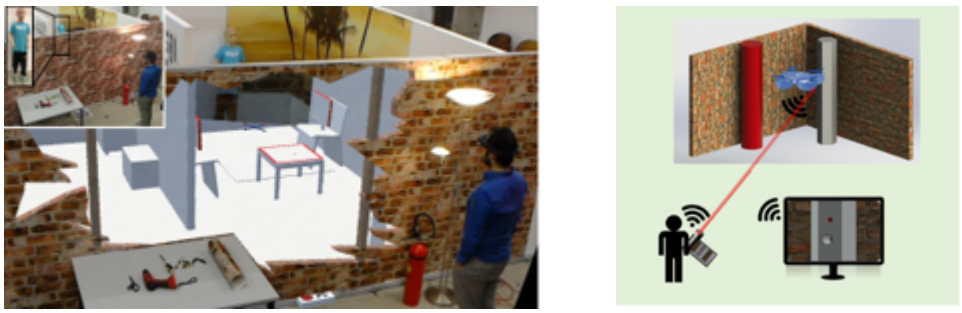


Figure 1: (left) exocentric vision for “see-through” walls. (right) laser-pointer guided inspection

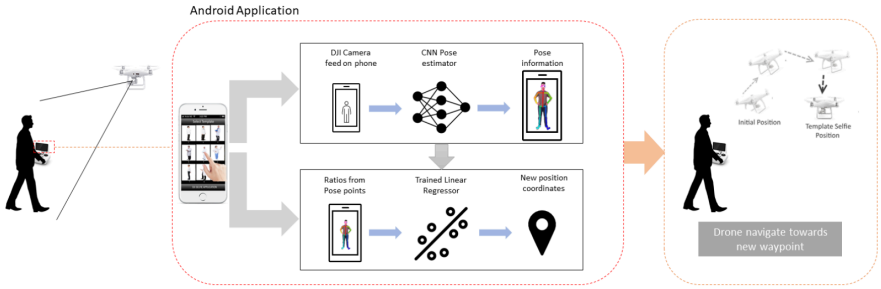


Figure 2: Selfie drone framework pipeline

2.4. 3D path planning

The drone needs to follow an offline generated path in order to perform a complete inspection. [11] propose a good framework (implemented as a ROS package) to generate an optimized inspection path offline by considering camera field of view (FOV). It uses a two steps optimization approach by first solving for the minimal number of viewpoints (depending on the 3D facets of the asset) and then tries finding the shortest connecting path by solving a Traveling Salesman Problem (TSP). This is interesting as it gives an already optimized path offline (see Figure 3). Then, our MPC formulation only needs to take care of the VLOS part of the problem!

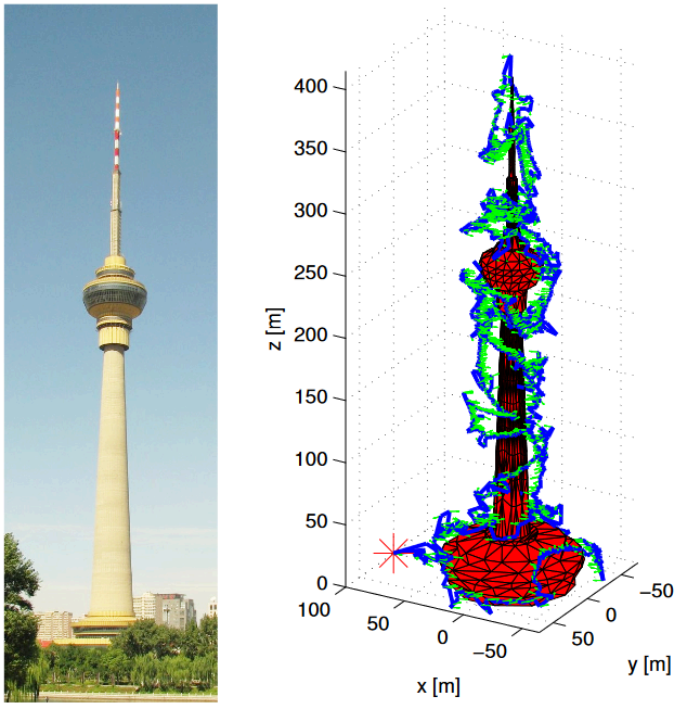


Figure 3: Offline computed inspection path with the iterative viewpoint sampling method

2.5. Drone regulations

In Switzerland, *FOCA* is the entity responsible for defining and applying regulations with respect to aerial vehicles. From [1], three types of Visual Line-of-Sight conditions are defined:

- **VLOS**:

  > A mode of drone operation during which the remote pilot is able to maintain uninterrupted visual contact with the drone without the assistance of an observer

- **EVLOS**:

  > [...] the UAS operation is being conducted via visual airspace surveillance through one or more human observers, possibly aided by technology means. The Pilot In Command (PIC) has a direct control of the UAS at all time.

- **BVLOS**:

  > A mode of drone operation that is not conducted with visual line of sight (VLOS).

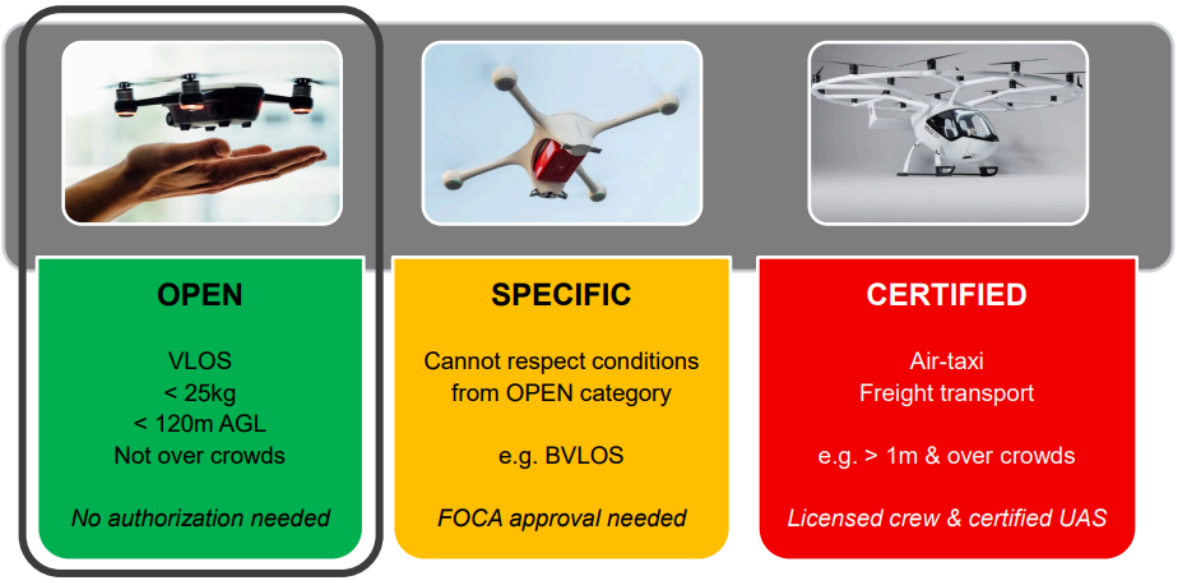


Figure 4: The 3 categories defined by *FOCA*

Taking into consideration inspection drones having a payload < 25 kg, it falls into two possible categories according to Figure 4:

- **OPEN**: If the drone is flown VLOS and stay within 120m above ground
- **SPECIFIC**: If the drone should be flown BVLOS

Since the **specific** category requires special permits and approvals by the *FOCA*, it is advantageous to fall into the **open** category. Thus, it is interesting to investigate an **optimization framework capable of assuring VLOS flying conditions at all time during inspection operations**.

Example of max distance to drone calculation for the one used in simulation:

$$\text{ALOS} = 327 * CD + 20m$$

Our drone CD is $2 \times$ arm length:

$$\text{CD} = 2 * 0.21 = 0.42$$

Thus, this gives the following maximum distance:

$$\text{ALOS} = 327 * 0.42 + 20m = \boxed{157m}$$

3. Problem definition

Design an **optimization** framework to ensure **VLOS** conditions with an **operator** while performing a simple asset **inspection**.

3.1. Pipeline

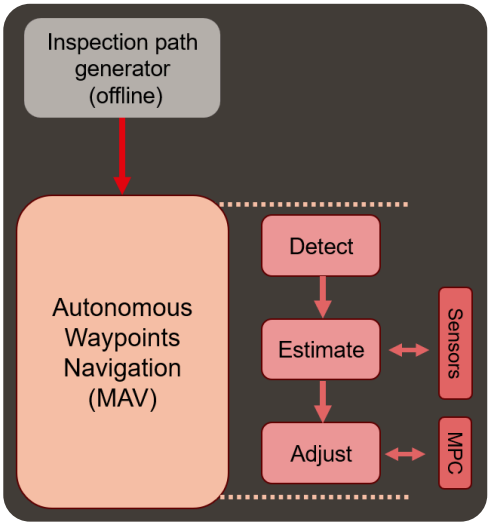


Figure 5: Proposed pipeline for VLOS inspections

The main focus of this project is the path adjustment module (*Adjust*) used to control path progression using MPC to ensure VLOS at all time.

In order to rapidly design the MPC formulation, a **MATLAB** simulation environment is used with simpler dynamics and sensors implementation. If time allows, **Webots** simulator will be used to assess performance under more complex and realistic environmental conditions.

3.2. Assumptions

The following assumptions are made about the simulation environment:

1. Operator is detected when drone starts inspection (initial condition)
2. No noise (process and sensing)
3. No disturbances
4. Obstacles are static
5. Obstacles positions are known
6. Constant velocity model for the operator (also known)
7. Absolute positioning of the drone is available (for path following)
8. Only simple convex shapes for inspection (path following is defined for circular inspections)

Some assumptions will be relaxed (1,2,5,7) for the **Webots** environment to test the robustness of the MPC formulation.

3.3. MPC formulation

The following formulation is intended for the basic case (*scenario 1*). Added terms will be specified under appropriate section where they will be used (*scenarios 2 & 3*).

$$\min_{\boldsymbol{u}_{0:N-1}} \quad J = \sum_{i=0}^{N-1} \Big(\alpha_{\text{pos}} \left\| \boldsymbol{p}_i - \boldsymbol{p}_{\text{ref},i} \right\|_2^2 + \alpha_{\text{vel}} \left\| \boldsymbol{v}_i \right\|^2 + \alpha_{\text{ang}} \left\| \boldsymbol{t}_{\boldsymbol{d},i} \right\|_2^2 + \alpha_F F_{\text{min},i}^2 + \alpha_{\text{op}} \left(\theta_{\text{op},i}^c \right)^2 + \alpha_\psi \left(\psi_{\text{inspect},i}^c \right)^2 + \alpha_s \left(s_{d,i} - s_{d,i-1} \right)_i^2 + \left\| \boldsymbol{\xi}_i \right\|_P^2 \Big) + \beta_{\text{op}} \left(\theta_{\text{op},N}^c \right)^2 + \beta_\psi \left(\psi_{\text{inspect},N}^c \right)^2 + \beta_{\text{vel}} \left\| \boldsymbol{v}_N \right\|_2^2$$

s.t.

$$\boldsymbol{x}^+ = f(\boldsymbol{x}) \tag{1}$$

$$0 \leq \boldsymbol{\xi}_i \tag{2}$$

$$0 \leq \left\| \boldsymbol{v}_i \right\|^2 \leq v_{\max}^2 \tag{3.1}$$

$$\theta_{\min} \leq t_{\theta,i} \leq \theta_{\max} \tag{3.2}$$

$$\varphi_{\min} \leq t_{\varphi,i} \leq \varphi_{\max} \tag{3.3}$$

$$\theta_{d,\min} \leq t_{d,\theta,i} \leq \theta_{d,\max} \tag{4.1}$$

$$\varphi_{d,\min} \leq t_{d,\varphi,i} \leq \varphi_{d,\max} \tag{4.2}$$

$$T_{\min} \leq T_i \leq T_{\max} \tag{4.3}$$

$$d_{\min}^2 - \xi_i \leq \left\| \boldsymbol{r}_{\text{op},i} \right\|^2 \leq d_{\max}^2 + \xi_i \tag{5.1}$$

$$\alpha_{\min}^2 \leq \left(\theta_{\text{op},i}^c \right)^2 \leq \alpha_{\max}^2 + \xi_i \tag{5.2}$$

$$0 \leq s_{d,i} \leq 1 \tag{6.1}$$

$$0 < s_{d,i} - s_{d,i-1} + \xi_i \tag{6.2}$$

Legend: drone state | drone inputs | VLOS | path progression

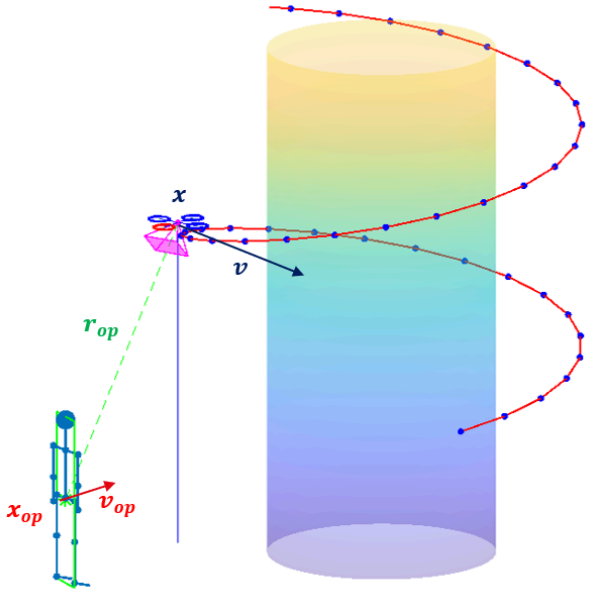


Figure 6: Environment and variables definition for the proposed MPC formulation

3.3.1. Variables definition

$$\boldsymbol{x} : \text{state vector} \rightarrow [\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{t}, \boldsymbol{\omega}]^T$$

$$\boldsymbol{u} : \text{control input vector} \rightarrow [T_d, \boldsymbol{t_d}]^T$$

$$Q : \text{Objective weights matrix} \rightarrow \begin{pmatrix} \alpha_{\text{pos}} & 0 & ... & 0 \\ 0 & \alpha_{\text{vel}} & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \alpha_{\text{angular}} \end{pmatrix}$$

$$Q_N : \text{Terminal weights matrix} \rightarrow \begin{pmatrix} \beta_{\text{op}} & 0 & 0 & 0 \\ 0 & \beta_{\text{psi}} & 0 & 0 \\ 0 & 0 & \beta_{\text{vel}} & 0 \\ 0 & 0 & 0 & \beta_{\text{acc}} \end{pmatrix}$$

$$P : \text{Slack weights matrix} \rightarrow \begin{pmatrix} \gamma_1 & 0 & ... & 0 \\ 0 & \gamma_2 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \gamma_n \end{pmatrix}$$

$$\boldsymbol{p} : \text{Drone's position (w.r.t inertial frame)}$$

$$\boldsymbol{p}_{\text{ref}} : \text{Desired position (w.r.t inertial frame)}$$

$$\boldsymbol{v} : \text{Drone's velocit (w.r.t inertial frame)}$$

$$\boldsymbol{t} : \text{Drone's Euler angles} \rightarrow [\varphi, \theta, \psi]^T$$

$$\boldsymbol{t}_d : \text{Desired Euler angles}$$

$$F_{\text{min}} : \text{Minimum thrust required} \rightarrow (T_d - mg)$$

$$\psi^c_{\text{inspect}} : \text{Inspection cam yaw angle (in camera frame)}$$

$$\theta^c_{\text{op}} : \text{Operator cam yaw angle (in camera frame)}$$

$$r_{\mathsf{op}} : \text{Operator relative distance}$$

$$s_d : \text{Path progression}$$

$$\boldsymbol{\xi} : \text{Slack variables for constraints smoothness}$$

3.3.2. Drone dynamics model

The dynamical model used for the drone is the same described in [2].

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{7.1}$$

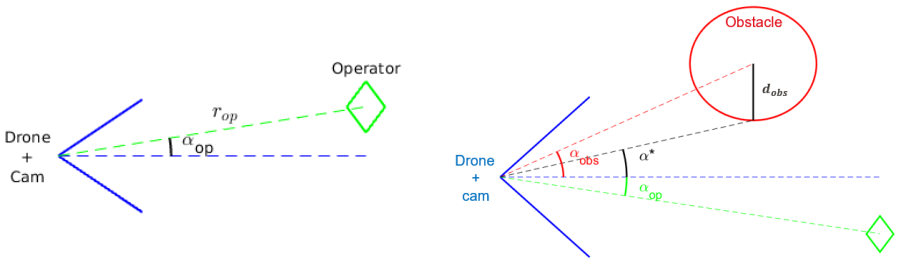
$$\dot{\boldsymbol{v}} = \frac{1}{m_b} \left(\boldsymbol{R}_b \begin{bmatrix} 0 \\ 0 \\ T_d \end{bmatrix} \right) + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{7.2}$$

$$\dot{\boldsymbol{t}} = \frac{1}{\boldsymbol{\tau}} (\boldsymbol{K} \boldsymbol{t_d} - \boldsymbol{t}) \tag{7.3}$$

3.3.3. Operator occlusion model

To maintain VLOS conditions with the operator, a different camera is used to keep track of the on-ground operator. Thus, the goal is to keep the operator angle (θ_{op}) within the limits of the camera frame (FOV). Also, an estimated relative distance (r_{op}) is used to respect maximum allowed distance between drone and operator at all time. The operator angle is included in the cost function to try as much as possible to keep it centered in camera frame to avoid possible tracking issues. The model is as follow:

$$\text{Operator} := \begin{cases} r^c_{\text{op}} = R^b_c R^w_b \boldsymbol{r}_{\text{op}} \\ \alpha_{\text{op}} = \tan^{-1}\left(\frac{r^c_{\text{op},y}}{r^c_{\text{op},x}}\right) \end{cases} \tag{8}$$



a. Operator model b. Obstacle model

Figure 7: Obstacle occlusion model definition for VLOS conditions

3.3.4. Obstacles occlusion model

To account for possible occlusions caused by obstacles during the inspection process, the following model is used to maximize the angle between any obstacles and the operator in the current camera frame (see Figure 7b).

$$\text{Obstacle} := \begin{cases} r^c_{\text{obs}} = R^b_c R^w_b \boldsymbol{r}_{\text{obs}} \\ \alpha_{\text{obs}} = \tan^{-1}\left(\frac{r^c_{\text{obs},y}}{r^c_{\text{obs},x}}\right) \end{cases} \tag{9}$$

$$\Rightarrow \alpha^*_{\text{obs}} = \alpha_{\text{obs}} - \tan^{-1}\left(\frac{d_{\text{obs}}}{\|\boldsymbol{r}^c_{\text{obs}}\|}\right)$$

4. Results

To test the performances of the above mention MPC formulation, a simple *MATLAB* environment is used (see #). A single inspection asset centered in the world frame is present. The inspection surface is modeled as a cylindrical shape. Two cameras are present on the drone: one used for the inspection task and the other used to detect and track the operator for maintaining VLOS at all time. Three different scenarios test different aspects of the formulation. Sections below (Section 4.1 - Section 4.3) describe the scenarios as well as the obtained results. Scenarios 1 and 2 use the environment **A** and scenario 3 uses environment **B**.

N.B. A short half-turn inspection path is defined to limit simulation time.

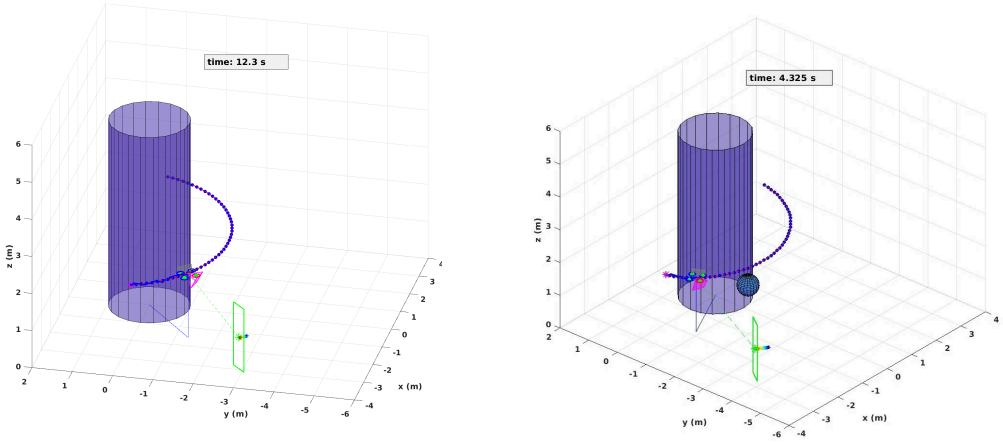
Recordings of the different simulations (scenarios) are available in the following *Youtube* playlist:

https://www.youtube.com/playlist?list=PL8cx2cevn_sbN0pgqydm0C-0Qm3tfScyd

Table 1 presents some fixed parameters that are used within the 3 defined scenarios.

Parameter	Value
Operator cam FOV	H: 70^o, V: 90^o
Operator speed	0.5 m/s
t_θ, t_φ	$\left[\frac{-3\pi}{16}, \frac{3\pi}{16}\right]$
$t_{d,\theta}, t_{d,\varphi}$	$\left[\frac{-\pi}{8}, \frac{\pi}{8}\right]$
T	$[0, 21]$
θ_{op}	$[-30, 30]^o$
r_{op}	$[0, 3.5]\ m$

Table 1: Parameters used throughout all scenarios



Environment **A** Environment **B**

Figure 8: Environment used in the different the scenarios

4.1. Scenario 1

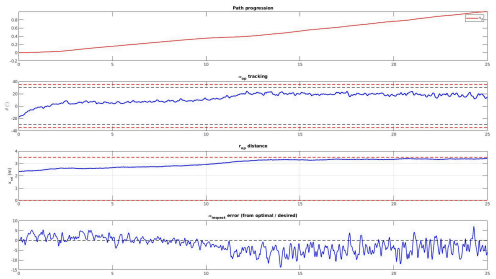
The simplest possible scenario. Drone starts on the path and the operator is initially detected and within the VLOS limits of the drone. The operator follows a circular trajectory at constant speed while the drone performs its inspection. All the variables parameters are defined in Table 2.

Parameter	Value
Weights (Q)	$\alpha_{\text{pos}} = 5, \alpha_{\text{vel}} = 0.01$ $\alpha_{\text{ang}} = 0.5, \alpha_F = 0.02$ $\alpha_{\text{op}} = 1.0, \alpha_{\text{inspect}} = 2.0$ $\alpha_{\text{path}} = 5, \alpha_{\text{eps}} = [100, 500, 5]$
Terminal weights (Q_N)	$\beta_{\text{vel}} = 0.01$ $\beta_{\text{op}} = 1.0, \beta_{\text{inspect}} = 2.0$

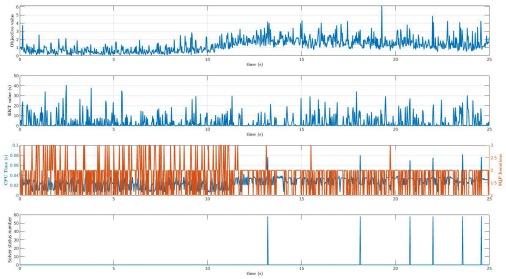
Table 2: Weights used for scenario 1

Design choices:

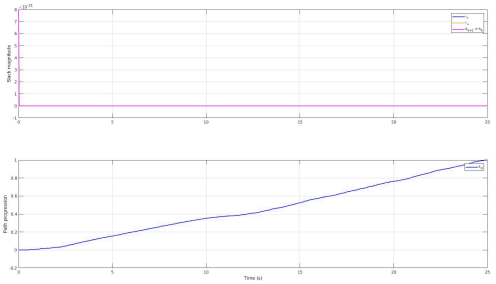
The path following is done by forcing the path progression variable s_d to increase each timestep (see Equation 6.2). A slack term (ξ) is present to limit instabilities and to allow the drone to stop on the path or even go a little backward if really needed (for instance to satisfy operator constraints). The VLOS constraints are defined as an operator angle (α_{op}) in the camera frame to make sure it is always trackable and the operator relative distance (r_{op}) to satisfy regulations. These also include slack, but the weight associated to it is much higher than the one of the path to make sure it is **most** of the time satisfied.



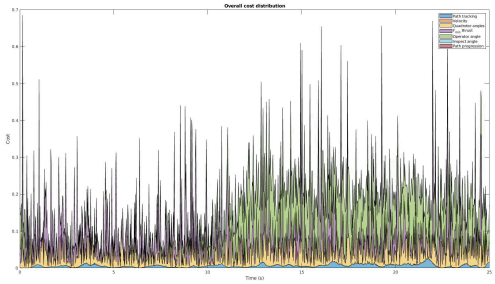
a. VLOS metrics



b. Solver status



c. Slack variables



d. Cost function distribution

Figure 9: Performance metrics for the scenario 1

Observations:

As one can see in Figure 9a, the path progression is quite smooth and linear. However, the inspection angle varies a lot, with spikes up to +/- 5^o. This could be improved by adding a cost component to limit the change in ψ, such as $\alpha_{\psi,\text{diff}} \left\| \psi_i - \psi_{i,\text{prev}} \right\|$. As for the VLOS conditions with the operator, all of the constraints are satisfied for the entire simulation, which is expected with the MPC formulation.

Regarding the solver status (Figure 9b), the KKT value, which represents the convergence (or the quality) of the found solution contains high spikes. Also, the solver reached its maximum allowed iterations 6 times during the simulation (*error flag* 58), which explains these high KKT peaks. However, the problem is highly complex and nonlinear (especially for the path following error) which does not help the solver to converge quickly and efficiently.

As for the objective cost distribution (Figure 9d), the minimum thrust required (F_{min} and the operator angle (α_{op})) are the highest. This means the solver struggles more to make these values smaller and accepts the cost related to these variables.

Finally, it is interesting to notice that **all the slack variables were kept as 0**.

4.2. Scenario 2

In this scenario, the goal is to **ensure** VLOS conditions are respected at all time by testing the defined constraints. So, the operator will stop its motion at 5 seconds after the start of the inspection. It will wait another 5 seconds before moving again, **but in the opposite direction** of the inspection asset. The drone should, at some point, leave the path to follow the operator to **respect the maximum allowed operator distance** w.r.t the drone. At this point, inspection is considered **aborted**.

MPC adjustments:

A path activation variable is added (δ_{path}) to change the composition of the objective function and to disable specific constraints to allow the drone to no longer follow the path.
Modified objective:

$$
\begin{aligned}
J = \sum_{i=0}^{N-1} \Big(& \boldsymbol{\delta_{\text{path}}} + \alpha_{\text{pos}} \left\| \boldsymbol{p}_i - \boldsymbol{p}_{\text{ref},i} \right\|_2^2 + \alpha_{\text{vel}} \|\boldsymbol{v}_i\|^2 + \alpha_{\text{ang}} \left\| \boldsymbol{t_d}_{,i} \right\|_2^2 + \alpha_F F_{\text{min},i}^2 \\
& + \alpha_{\text{op}} \left(\theta_{\text{op},i}^c\right)^2 + \boldsymbol{(1 - \delta_{\text{path}})} \alpha_r \left\| r_{\text{op},i} - r_{\text{des}} \right\|_2^2 + \boldsymbol{\delta_{\text{path}}} \alpha_\psi \left(\psi_{\text{inspect},i}^c\right)^2 \\
& + \boldsymbol{\delta_{\text{path}}} \alpha_s \left(s_{d,i} - s_{d,i-1}\right)_i^2 + \|\boldsymbol{\xi}_i\|_P^2 \Big) + \beta_{\text{op}} \left(\theta_{\text{op},N}^c\right)^2 \\
& + \boldsymbol{(1 - \delta_{\text{path}})} \beta_r \left\| r_{\text{op},N} - r_{\text{des}} \right\|_2^2 + \boldsymbol{\delta_{\text{path}}} \beta_\psi \left(\psi_{\text{inspect},N}^c\right)^2 + \beta_{\text{vel}} \|\boldsymbol{v}_N\|_2^2
\end{aligned}
$$

Also, some constraints have been adjusted (Equation 6.1 and Equation 6.2):

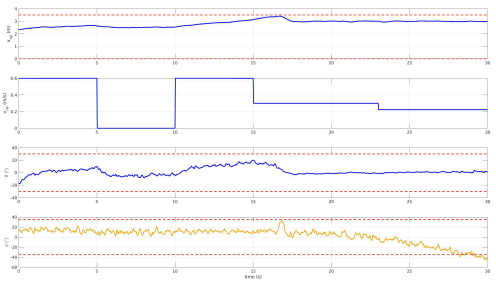
$$
0 \leq \boldsymbol{\delta_{\text{path}}} s_{d,i} \leq 1 \tag{10.1}
$$

$$
0 < \boldsymbol{\delta_{\text{path}}} \left(s_{d,i} - s_{d,i-1}\right) + \xi_i \tag{10.2}
$$

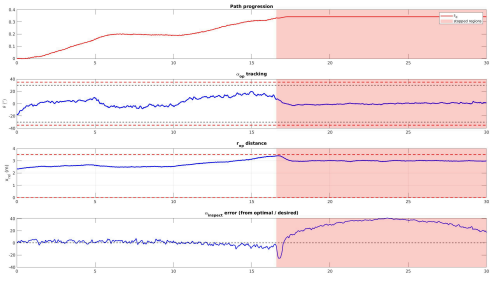
The operator distance (r_{op}) is added to the objective to give the drone a target when it is no longer required to follow the path. Otherwise, some instabilities occur due to the under-constrained problem.
The path active (δ_{path}) is determined offline by detecting when the operator is close to violate VLOS conditions (angle and/or distance). δ_{path} acts as a latching boolean state which cause the inspection to be aborted and the drone to follow the operator for the rest of the simulation.

Observations:

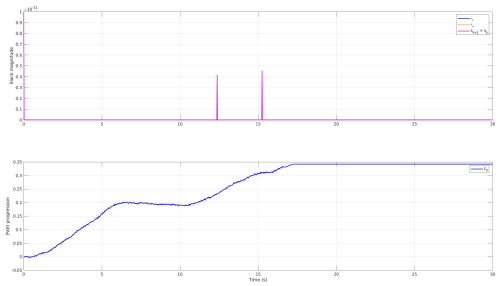
Regarding the operator and VLOS conditions, one can see from Figure 10ab that the path progression (s_d) stalls as soon as the operator stop moving (between 5 and 10 seconds). After 15 seconds, when the operator starts walking away from the inspection asset, VLOS conditions are maintained by not following the path anymore, which is shown by the red region on the plots. This happens as soon as the operator reaches maximum allowed relative distance with the drone. Then, the controller switches to fixed distance reference to follow the operator.
Another interesting part to look at is the slack variable associated with the path progression (magenta trace in Figure 10c). It spikes twice, where the operator limits are close to be violated. This means the drone tried to stop or even backtrack on the path.
Finally, the KKT values are still high, which is expected because the problem still remains highly complex and non linear, with the addition of bilinear terms.



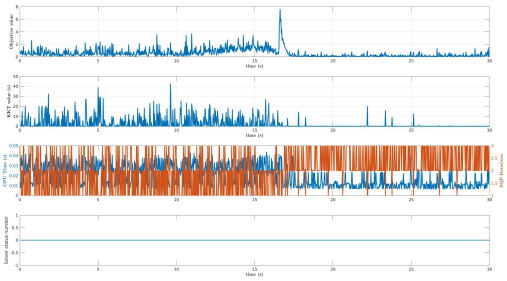
a. Operator states



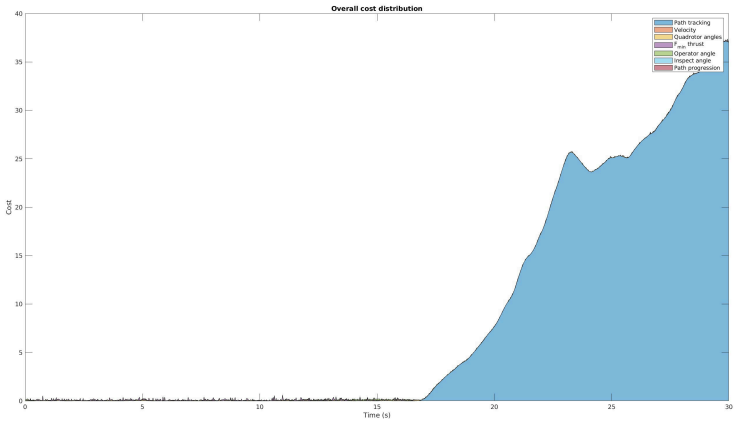
b. VLOS metrics



c. Slack variables



d. Solver Info



e. Cost distribution

Figure 10: Performance metrics for the scenario 2

4.3. Scenario 3

In this last scenario, the goal is to add an obstacle to the environment that will cause an occlusion if not taken into account by the MPC controller. The obstacle model defined above (see Equation 9) is added to the formulation of the problem. Since *Acado* needs a predefined amount of variables and online data, a placeholder for two detected obstacles at the same time were added. It is only to verify that the constraints and objective cost behave as expected. Small modifications are needed to account for many more obstacles concurrently.

MPC adjustments:

Starting again from the MPC baseline (scenario 1), the position of the detected obstacles by the camera is given as absolute, so that the MPC solver can predict **online** the change in relative position since the drone is moving within the prediction horizon N. A boolean variable (δ_{obs}) is also provided to indicate if the obstacle in the placeholder is to be considered or not.

Added constraint:

$$\boldsymbol{\delta_{\text{obs}}}\left(\theta^c_{\text{obs,i}} - \theta^c_{\text{op,i}}\right) + \xi_i \geq \theta^{\min}_{\text{obs}} \tag{11}$$

where

$$\theta^c_{\text{obs}} = \tan^{-1}\left(\frac{y^c_{\text{obs}}}{x^c_{\text{obs}}}\right)$$

$$\boldsymbol{x}^c_{\text{obs}} = R^T_{\text{cam}} R^T_{\text{drone}} (\boldsymbol{x}_{\text{obs}} - \boldsymbol{x}_{\text{drone}})$$

Modified objective:

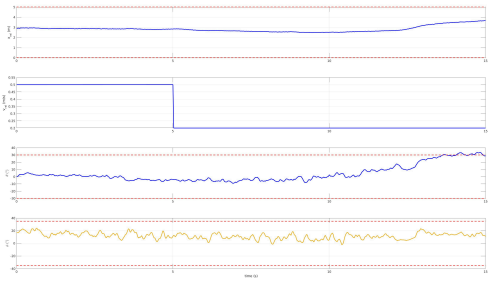
These 2 terms were added:

$$\alpha_r \left\| \boldsymbol{r}_{\text{op},i} \right\|_2^2 \quad \& \quad \beta_r \left\| \boldsymbol{r}_{\text{op},N} \right\|_2^2$$

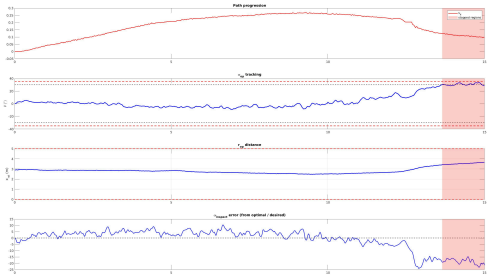
To help the drone limit the possible occlusion by keeping a minimal distance to the operator. By sizing accordingly the weights, this can improve the stability of the controller.

Parameter	Value
Weights (Q)	$\alpha_{\text{pos}} = 5$, $\alpha_{\text{vel}} = 0.01$ $\alpha_{\text{ang}} = 0.5$, $\alpha_F = 0.02$ $\alpha_{\text{op}} = 1.0$, $\alpha_{\text{inspect}} = 2.0$ $\alpha_{\text{path}} = 5$, $\alpha_r = 0.0005$ $\alpha_{\text{eps}} = [50, 500, 10, 1000, 1000]$
Terminal weights (Q_N)	$\beta_{\text{vel}} = 0.01$ $\beta_{\text{op}} = 1.0, \beta_{\text{inspect}} = 2.0$ $\beta_r = 0.001$
$\theta^{\min}_{\text{obs}}$	5^o

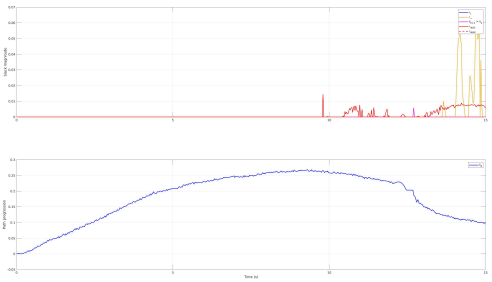
Table 3: Weights used for scenario 3



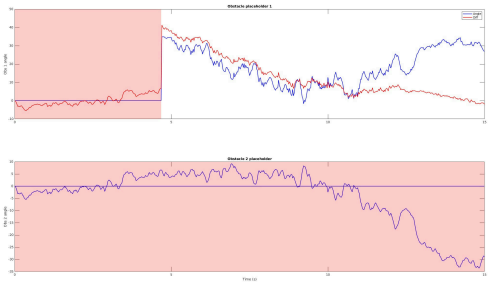
a. Operator states



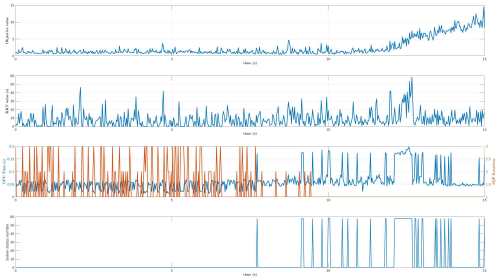
b. VLOS metrics



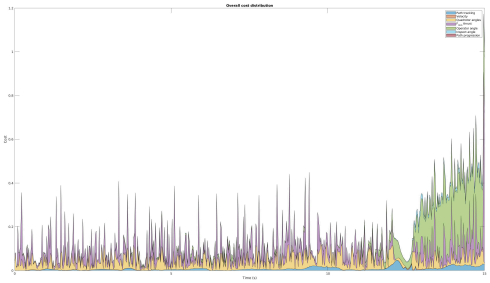
c. Slack variables



d. Obstacles detection



d. Solver info



e. Cost distribution

Figure 11: Performance metrics for the scenario 3

Observations:

The most interesting plot to look at for this particular scenario is Figure 11d, where one can see, by the red region, that only one obstacle was detected during the simulation, and this happened right before 5 seconds, when the sphere entered the camera FoV. The 'diff' angle represents the difference between the obstacle angle and the operator angle, all in the camera frame. To avoid occlusions, this angle should stay above zero. This is almost always the case, except towards the end of the simulation, where the MPC controller struggled a bit more.
As for the solver output, the KKT value remains within the same range as scenario 1 and 2, even dough we added more complexity to the formulation. The objective cost distribution is also similar as previous scenarios, where this time the operator angle is taking over the cost towards the end. That is expected since the operator was far away from the drone while it was trying to avoid occlusions with the sphere.

5. Conclusion

In summary, an MPC controller was designed to maintain VLOS conditions with an on-ground operator **at all time**. This includes conditions such as maintaining the operator in the camera FoV, maintaining a proper distance to the operator and avoiding possible occlusions during inspection task. In order to test the proposed problem formulation, three different scenarios where implemented in *MATLAB*, adapted from an existing simulator [2]. Also, to be able to focus on the design of the problem, sensing noise, disturbances and state estimation where assumed "perfect". Many techniques are well known to address these in practice, so this will be left for future work.

Although obtained results are satisfying, they are not perfect. Mainly, the proposed MPC formulation is highly complex and nonlinear, making the solver struggle to find a quality solution (indicated by the high KKT value). Also, only simple cases were tested, it might fail or become unstable when adding more environmental complexity.

Finally, the MPC framework was supposed to be also trested in *Webots*. However, we decided to instead have different scenarios with good MPC formulation in *MATLAB* before moving to a more realistic simulator. Thus, this will be for a future work.

6. Future work

6.1. Regarding the MPC formulation

→ **Reactivation of the path** : In scenario 2, when the operator goes to far from the inspection asset, the drone as to disable the path progression and change its objective to follow the operator instead. However, this decision aborts the inspection. It could be desirable to activate again the inspection if the operator comes back to a point where the drone can resume.

→ **Solver iterations and KKT value**: As noted in all the scenarios, the solver often hits the maximum allowed *QP* iterations (flag 58) and outputs a solution with a high KKT value. This could be linked to the design of the formulation. One might want to try convexifying more of the constraints and/or apply some linearization to some of the functions used, by making reasonable assumptions. This could greatly help the solver to converge quicker, give more stable solutions and limit the risk of finding sub-optimal solutions due to nonlinearity.

→ **Add the beta angle:** As for now, the simulation was designed to not violate the β_{FOV} constraint for keeping the operator in the camera frame. This is a bit more tricky since the drone cannot easily

tilt forward to adjust this angle since it will starts accelerating at the same time. Thus, it might need to change its distance with the operator and/or its reference height.

6.2. Regarding the simulation environment

The next step is to move to a more realistic simulator (such as *Webots*) to test the MPC formulation under uncertainties and disturbances coming from the drone sensors state estimation and the tracking framework for the operator. Figure 12 shows a possible pipeline to be used within the *Webots* environment to further test the MPC formulation while utilizing a pose detection framework for operator tracking.

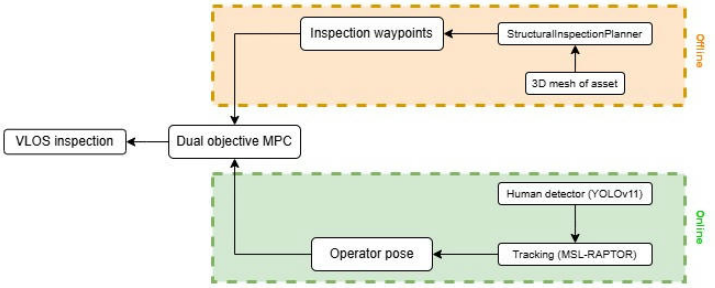


Figure 12: Proposed pipeline for the *Webots* simulation

Using the following components:

- Offline Structural Inspection Planner: *ASL-ETHZ* lab [11]
- Human detector: YOLOv11 model from *Ultralytics* (https://github.com/ultralytics/ultralytics)
- Tracking framework: *MSL-RAPTOR* (https://github.com/StanfordMSL/MSL-RAPTOR)
- Operator pose estimation pipeline inspired by previous semester project done at *DISAL*

References

[1] F. Gm, "UAS Authorization and Oversight."

[2] I. Erunsal, J. Zheng, R. Ventura, and A. Martinoli, "Linear and Nonlinear Model Predictive Control Strategies for Trajectory Tracking Micro Aerial Vehicles: A Comparative Study," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan: IEEE, Oct. 2022, pp. 12106–12113. doi: 10.1109/IROS47612.2022.9981880.

[3] M. E. ELzaiady and A. Elnagar, "Next-best-view planning for environment exploration and 3D model construction," in *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, Dubai, United Arab Emirates: IEEE, Dec. 2017, pp. 745–750. doi: 10.1109/ICTUS.2017.8286106.

[4] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-Aware Model Predictive Control for Quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid: IEEE, Oct. 2018, pp. 1–8. doi: 10.1109/IROS.2018.8593739.

[5] J. Xing, G. Cioffi, J. Hidalgo-Carrió, and D. Scaramuzza, "Autonomous Power Line Inspection with Drones via Perception-Aware MPC," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Detroit, MI, USA: IEEE, Oct. 2023, pp. 1086–1093. doi: 10.1109/IROS55552.2023.10341871.

[6] L. Petit and A. L. Desbiens, "MOAR Planner: Multi-Objective and Adaptive Risk-Aware Path Planning for Infrastructure Inspection with a UAV," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan: IEEE, May 2024, pp. 8422–8428. doi: 10.1109/ICRA57147.2024.10610907.

[7] H. Wang, X. Zhang, Y. Liu, X. Zhang, and Y. Zhuang, "SVPTO: Safe Visibility-Guided Perception-Aware Trajectory Optimization for Aerial Tracking," *IEEE Trans. Ind. Electron.*, vol. 71, no. 3, pp. 2716–2725, Mar. 2024, doi: 10.1109/TIE.2023.3270541.

[8] O. Erat, W. A. Isop, D. Kalkofen, and D. Schmalstieg, "Drone-Augmented Human Vision: Exocentric Control for Drones Exploring Hidden Areas," *IEEE Trans. Visual. Comput. Graphics*, vol. 24, no. 4, pp. 1437–1446, Apr. 2018, doi: 10.1109/TVCG.2018.2794058.

[9] M. Shaqura, K. Alzuhair, F. Abdellatif, and J. S. Shamma, "Human Supervised Multirotor UAV System Design for Inspection Applications," in *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Philadelphia, PA: IEEE, Aug. 2018, pp. 1–6. doi: 10.1109/SSRR.2018.8468648.

[10] D. Ahmed, W. Shahid Qureshi, S. Arsalan Aijaz, B. Muhammad Imran, S. Manshoor Ali Naqvi, and C.-Y. Lin, "Towards Selfie Drone: Spatial Localization and Navigation of drone Using Human Pose Estimation," in *2021 International Conference on Robotics and Automation in Industry (ICRAI)*, Rawalpindi, Pakistan: IEEE, Oct. 2021, pp. 1–7. doi: 10.1109/ICRAI54018.2021.9651330.

[11] A. Bircher *et al.*, "Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA: IEEE, May 2015, pp. 6423–6430. doi: 10.1109/ICRA.2015.7140101.