Project Report

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Autonomous Landing of a Quadrotor on a Moving Vehicle

Research Internship Report

Winter 2022
Département de génie électrique
École Polytechnique de Montréal

May 6, 2022

Author: Adam Ghribi
Supervisor: Prof. David Saussié

Contents

List of Figures

Nomenclature

$\mathbf{\Phi} = [\phi\ \theta\ \psi]^\top$ The three Euler angles : roll, pitch, yaw.

$\boldsymbol{p^i} = [x\ y\ z]^\top$ The position vector.

$\boldsymbol{\omega}^b_{b/i} = [p\ q\ r]^\top$ The angular velocity of frame {b} with respect to frame {i} expressed in {b}.

$\mathbf{g}^i = [0\ 0\ -g]^\top$ The gravitation vector expressed in {i}.

$\mathbf{R}^i_b$ The rotation from frame {i} to frame {b}.

$\mathbf{I}_b$= diag(I_{xx},I_{yy},I_{zz}) The drone's inertia matrix

Q State weighting matrix

R Control weighting matrix

$\mathbf{u} = [T\ L\ M\ N]^\top$ The input vector

ENU The inertial **E**ast-**N**orth-**U**p frame.

FLU The Body **F**ront-**L**eft-**U**p frame.

1 Introduction

The autonomous landing of Unmanned aerial vehicles (UAVs) on static and dynamic platforms represents a requirement for several modern applications. A high autonomy level is crucial for a variety of missions, e.g., docking into a recharging station, landing on delivery trucks, and rescue operations [1] [6]. This complex task requires precise ground vehicle localization and the ability to generate a feasible landing trajectory. The subject has been an active research area for several years. [6] represents a promising solution, that has demonstrated deployment in real-world conditions. The team was able to land successfully on a car moving at 15 km/h during the Mohamed Bin Zayed International Robotics Challenge. In this work, we focus on developing a performant landing algorithm for a quadrotor based on the approach in [6].

In the work that follows, we first define the problem. We describe then the plant and provide a linearized model. In section 4, we design a controller to stabilize the drone using a Linear Quadratic Regulator (LQR). In section 5, we describe our model predictive control (MPC)-based approach to generate the landing trajectories. Finally, the simulated results are presented along with an analysis of the system's performance.

2 Problem Definiton

The task consists of the autonomous landing of a multirotor micro aerial vehicle (MAV)on a moving ground vehicle (GV). Our focus in this work is to develop an algorithm enabling a feasible and smooth landing. The drone has first to reach the waiting point and hovers at the height of 5 m above the road. As soon as the car passes, the MAV starts the landing manoeuvre.

3 Dynamic Model of The Quadrotor

In this section, we present the dynamic model of the drone, which will be linearized and then used for the design of the control laws. All the equations are based on the work of [2] and [4]. Four assumptions are needed to simplify the modelling of the drone:

1. The Earth is considered locally flat and non-rotating.
2. The mass m of the MAV is constant.
3. The drone is symmetric.
4. The drone is a rigid body.

The first hypothesis is valid since the curvature of the Earth and its speed are negligible compared to the distances travelled in the simulations. This assumption implies that the gravity vector points towards the ground. The second and the third hypotheses permit the simplification of the dynamic equations of the MAV so that $\dot{m} = 0$ and the inertia matrix is diagonal $\mathbf{I}_b$= diag(I_{xx},I_{yy},I_{zz}). The last Hypothesis makes it possible to neglect the deformation and the flexible modes of the drone.

3.1 Frames of Reference and Orientation

To model the dynamics of the drone, we define the following two frames.

The inertial frame is a non-accelerating frame of reference in which Newton's laws of motion apply. Its origin is a fixed arbitrary point on the surface of the Earth. The $\boldsymbol{x}_{\mathrm{i}}$ axis points towards the East whereas the $\boldsymbol{y}_{\mathrm{i}}$ axis points towards the North. The $\boldsymbol{z}_{\mathrm{i}}$ axis is defined to complete the right-handed coordinate system so that it points Up. The frame is denoted {i} and follows the convention **ENU** (**E**ast-**N**orth-**U**p).

The Body frame is fixed to drone and is denoted {b}. Its origin is located at the quadrotor's center of mass. The $\boldsymbol{z}_{\mathrm{b}}$ axis goes from the bottom to the top of the body so that it points up . The $\boldsymbol{x}_{\mathrm{b}}$ axis points from the front to the back (front). The $\boldsymbol{y}_{\mathrm{b}}$ axis is defined to complete the right-handed coordinate system (left). The body frame follows the convention **FLU**(**F**ront-**L**eft-**U**p).

As described in [4], we define the position vector as $\boldsymbol{p^i} = [x\ y\ z]^\top$. The orientation of frame {b} relative to the inertial frame is given by the three Euler angles $\boldsymbol{\Phi} = [\phi\ \theta\ \psi]^\top$. The rotation from frame {i} to frame {b} is then defined as follow

$$\mathbf{R}_b^i = \begin{bmatrix} \mathrm{c}_\theta \mathrm{c}_\psi & \mathrm{c}_\theta \mathrm{s}_\psi & -\mathrm{s}_\theta \\ \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{c}_\psi - \mathrm{c}_\phi \mathrm{s}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{s}_\psi + \mathrm{c}_\phi \mathrm{c}_\psi & \mathrm{s}_\phi \mathrm{c}_\theta \\ \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{c}_\psi + \mathrm{s}_\phi \mathrm{s}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{s}_\psi - \mathrm{s}_\phi \mathrm{c}_\psi & \mathrm{c}_\phi \mathrm{c}_\theta \end{bmatrix} \tag{1}$$

where $c_x = \cos x$ and $s_x = \sin x$. Note that $\mathbf{R}_b^i$ is an element of the orthogonal group SO(3), so that, $\mathbf{R}_b^i = \mathbf{R}_b^{i\,-1} = \mathbf{R}_b^{i\,\top}$.

3.2 The Dynamic Equations of the Quadrotor

The angular velocity of frame {b} with respect to frame {i} expressed in {b} is defined as $\boldsymbol{\omega}_{b/i}^b = [p\ q\ r]^\top$. [4] gives the kinematics of the drone as follow

$$\dot{\boldsymbol{\Phi}} = \begin{bmatrix} 1 & \mathrm{s}_\phi \mathrm{t}_\theta & \mathrm{c}_\phi \mathrm{t}_\theta \\ 0 & \mathrm{c}_\phi & -\mathrm{s}_\phi \\ 0 & \mathrm{s}_\phi / \mathrm{c}_\theta & \mathrm{c}_\phi / \mathrm{c}_\theta \end{bmatrix} \boldsymbol{\omega}_{b/i}^b \ \text{, with } \ \mathrm{t}_x = \tan x. \tag{2}$$

By applying the Newton's second law we obtain

$$m\ddot{\mathbf{p}}^i = m\mathbf{g}^i + \mathbf{R}_i^b \mathbf{F}^b \tag{3}$$

$$\mathbf{I}_b \dot{\boldsymbol{\omega}}_{b/i}^b = -\boldsymbol{\omega}_{b/i}^b \times \mathbf{I}_b \boldsymbol{\omega}_{b/i}^b + \mathbf{M}^b \tag{4}$$

Eq.3 describes the translation dynamics of the UAV, while Eq.4 describes the rotation dynamics. The gravitation vector expressed in {i} is $\mathbf{g}^i = [0\ 0\ -g]^\top$ and the forces and moments vectors created by the rotors and expressed in {b} are respectively $\mathbf{F}^b = [0\ 0\ T]^\top$ and $\mathbf{M}^b = [L\ M\ N]^\top$, where T represents the total thrust $T = T_1 + T_2 + T_3 + T_4$, L is roll moment, M is the pitch moment, and N is the yaw moment. Each motor creates a thrust $T_i = k_t \omega^2$ pointing always to the

positive $\mathbf{z}_b$ axis, and where k_t is the thrust coefficient and ω is the rotor angular rate. Note that the moment vector $\mathbf{M}$ is the sum the thrust $\mathbf{M}_t$ and the induced $\mathbf{M}_i$ torques created by all rotors.In this project, in order to simplify the work, we use $\mathbf{u} = [T\ L\ M\ N]^\top$ as input vector. In reality, to control the motors, we need to transform $\mathbf{u}$ to $\mathbf{v} = [\omega_1\ \omega_2\ \omega_3\ \omega_4]^\top$. The relation is given by [2] as follow

$$\mathbf{u} = \begin{bmatrix} k_t & k_t & k_t & k_t \\ 0 & 0 & dk_t & dk_t \\ -dk_t & dk_t & 0 & 0 \\ k_d & k_d & -k_d & -k_d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{5}$$

where d is the distance between each rotor and the centre of {b}. All forces, moments, and frames described above are represented in figure 1.

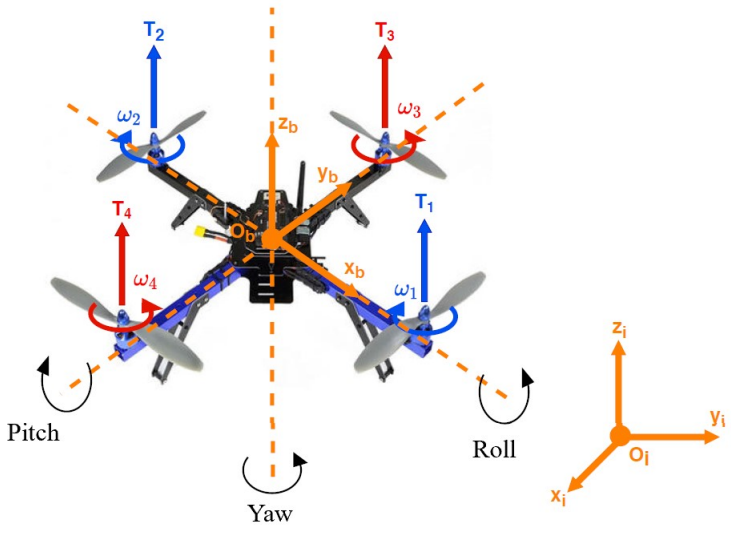

Figure 1: Forces, moments and frames. Modified from [2]

The Linearized Model is given by [4] as follow

$$\begin{array}{rclcrclcrcl} \Delta\ddot{x} & = & g\Delta\theta & & I_{xx}\Delta\dot{p} & = & \Delta L & & \Delta\dot{\phi} & = & \Delta p \\ \Delta\ddot{y} & = & -g\Delta\phi & & I_{yy}\Delta\dot{q} & = & \Delta M & & \Delta\dot{\theta} & = & \Delta q \\ m\Delta\ddot{z} & = & \Delta T & & I_{zz}\Delta\dot{r} & = & \Delta N & & \Delta\dot{\psi} & = & \Delta r \end{array} \tag{6}$$

So that the linear state-space representation is

$$\begin{cases} \boldsymbol{\Delta}\dot{\mathbf{x}} = \mathbf{A}\boldsymbol{\Delta}\mathbf{x} + \mathbf{B}\boldsymbol{\Delta}\mathbf{u} \\ \boldsymbol{\Delta}\mathbf{y} = \mathbf{C}\boldsymbol{\Delta}\mathbf{x} \end{cases} \tag{7}$$

where $\boldsymbol{\Delta}\mathbf{x} = [\boldsymbol{\Delta p}^\top\ \boldsymbol{\Delta\Phi}^\top\ \boldsymbol{\Delta\dot{\Phi}}^\top\ \boldsymbol{\Delta\dot{p}}^\top]^\top$, $\boldsymbol{\Delta}\mathbf{y} = [\Delta x\ \Delta y\ \Delta z\ \Delta\psi]^\top$, and $\mathbf{u} = [\Delta T\ \Delta L\ \Delta M\ \Delta N]^\top$.

4 Controller Design

In this section, we describe our approach to stabilising the quadrotor. The latter has to follow smooth trajectories. Aggressive manoeuvres are not required. We suggest using the linear system 7 for the design of an LQR controller. The method ensures a stable closed-loop system while providing optimal control gains suitable for the problem requirements. We synthesize optimal state feedback with an integral action so that the control law is given as

$$\mathbf{\Delta u} = -\mathbf{K}\mathbf{\Delta x} + \mathbf{K}_i\mathbf{\Delta x}_i \tag{8}$$

where $\mathbf{\Delta x}_i = [x\ y\ z\ \psi]^\top$. We obtain the following augmented linear system

$$\mathbf{\Delta \dot{x}}_a = \begin{bmatrix} \mathbf{A} & \mathbf{0}_{12\times4} \\ -\mathbf{C} & \mathbf{0}_{4\times4} \end{bmatrix} \mathbf{\Delta x}_a + \begin{bmatrix} \mathbf{A} \\ -\mathbf{D} \end{bmatrix} \mathbf{\Delta u} + \begin{bmatrix} \mathbf{0}_{12\times4} \\ \mathbf{I}_{4\times4} \end{bmatrix} \mathbf{r} \tag{9}$$

where the augmented state $\mathbf{\Delta x}_a = [\mathbf{\Delta x}^\top\ \mathbf{\Delta x}_i^\top]^\top$ and the command input $\mathbf{r} = [x_{des}\ \ y_{des}\ z_{des}\ \psi_{des}]^\top$. The optimal gains $\mathbf{K}$ and $\mathbf{K_i}$ are obtained by minimizing the quadratic cost function

$$\mathcal{J} = \frac{1}{2}\int_0^\infty \mathbf{\Delta x}_a^\top \mathbf{Q}\mathbf{\Delta x}_a + \mathbf{\Delta u}^\top \mathbf{R}\mathbf{\Delta u}\mathrm{d}t \tag{10}$$

where $\mathbf{Q} \geq 0$ and $\mathbf{R} > 0$ are both diagonal weighting matrices. The values used in this work are

$$\begin{aligned} \mathbf{Q} &= \mathrm{diag}(1, 1, 1, 0.1, 0.1, 0.1, 0.0001, 0.0001, 0.0001, 0.1, 0.1, 0.1, 1500, 1500, 1500, 1500). \\ \mathbf{R} &= \mathrm{diag}(0.1, 0.1, 0.1, 0.1). \end{aligned} \tag{11}$$

We analyse the step response of the following closed-loop system

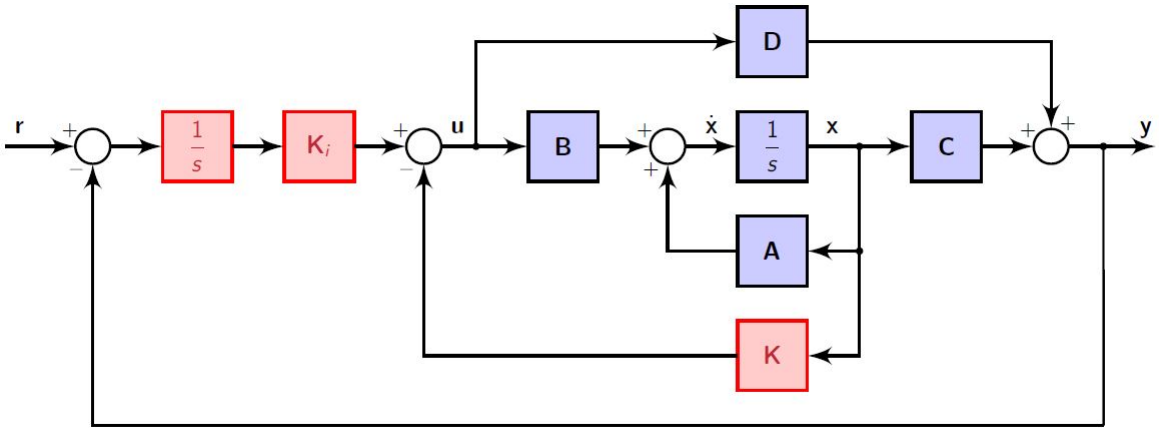

Figure 2: Closed-loop system. Modified from [5]

The Results are represented in figure 3. We note a response time $T_r(2\%)$ of 1.41 s, 1.33 s, 1.53 s, and 0.62 s, as well as relatively high overshoots values ranging from 8% to 11%.

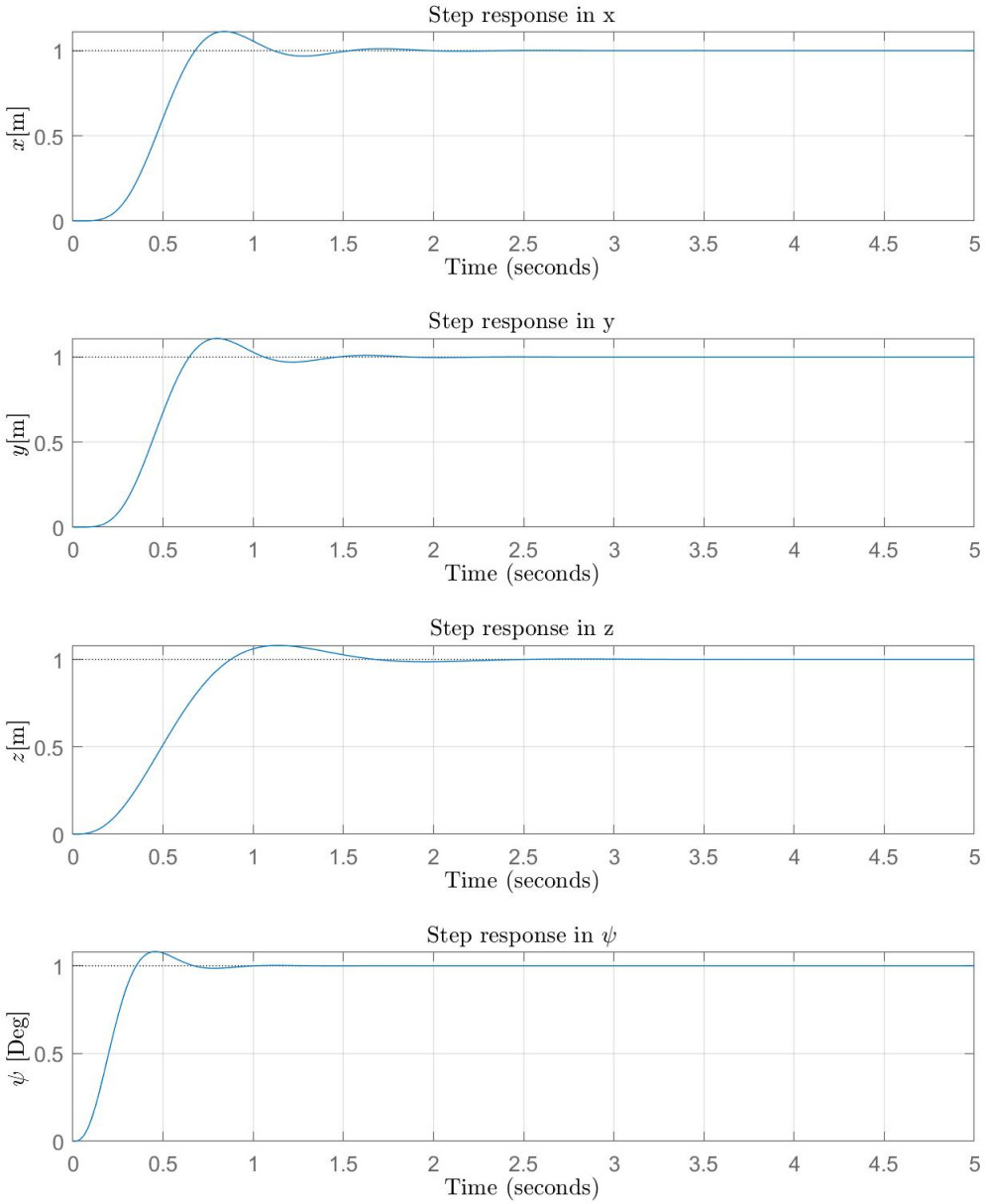

Figure 3: Step response of the closed-loop system

5 Trajectory Planner

In this section, we describe our guidance algorithm. The solution is based on the work of [6]. The idea is to design a controller using a Model Predictive Control (MPC) approach. It controls a simple model of the drone translation dynamics in real-time simulation. We use the output of the closed-loop system as reference $\mathbf{r}$ for the main system presented in figure 2.

Model Predictive Control
MPC uses a linear system to estimate the controller state and predicts future plant outputs, which are used to solve quadratic programming (QP) optimization problem [3]. For a given future prediction horizon interval τ_p, the algorithm finds the optimal control action $\mathbf{u}$ that minimizes a cost function. The solution is only available during the control horizon $\tau_m \leq \tau_p$. The control $\mathbf{u}$ is updated every τ_m.

Linear Plant
The model is a discrete Linear Time-Invariant (LTI) system that describes the transition dynamics of the quadrotor. The state vector is given as $\mathbf{x}_k = [x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y}, z, \dot{z}, \ddot{z}]_k^\top$, where

$$\mathbf{A}_p = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } \mathbf{B}_p = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \Delta t & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Delta t \end{bmatrix} \tag{12}$$

where the same time step $\Delta t = 0.01$ s is used for plant and MPC.

In this work, the optimal control action $\mathbf{u}$ minimizes the following cost function given by [6]

$$\mathcal{V} = \frac{1}{2} \sum_{i=1}^{m-1} \mathbf{e}_i^\top \mathbf{Q} \mathbf{e}_i + \mathbf{u}_i^\top \mathbf{P} \mathbf{u}_i \quad , \text{ with } z_i \geq 0. \tag{13}$$

where $\mathbf{e}_i = \mathbf{x}_i - \tilde{\mathbf{x}}_i$ is the control error. $\tilde{\mathbf{x}}_i$ represents the state vector of the target. It is used as a setpoint for the MPC. $\mathbf{Q}$ and $\mathbf{P}$ are both diagonal weighting matrices defined as follow

$$\begin{aligned} \mathbf{Q} &= \text{diag}(10, 10, 10, 2, 2, 2, 0.1, 0.1, 0.1). \\ \mathbf{P} &= \text{diag}(0.05, 0.05, 0.05). \end{aligned} \tag{14}$$

We set the prediction horizon interval $\tau_p = 1$ s and the control horizon interval $\tau_m = 0.2$ s.

6 Simulation

In this section, we describe all setups required to recreate the simulation results. We present the quadrotor model, the controller, the guidance system, and the model from the ground vehicle. All simulations are performed using Simulink.

6.1 Setup

To obtain the state vector $\mathbf{x} = [\boldsymbol{p}^\top \ \boldsymbol{\Phi}^\top \ \dot{\boldsymbol{\Phi}}^\top \ \dot{\boldsymbol{p}}^\top]^\top$ of the drone, we have to integrate the differential equations 2, 3, and 4. The drone parameters are given by [2] and presented in table 1.

Table 1: Physical parameters of the drone

Parameters	Values	Units
m	1.507	kg
I_{xx}	0.035	kg.m^2
I_{yy}	0.046	kg.m^2
I_{zz}	0.0977	kg.m^2
d	0.215	m

The closed-loop system is presented in figure 4. It shows the Simulink block of the stabilized drone model.

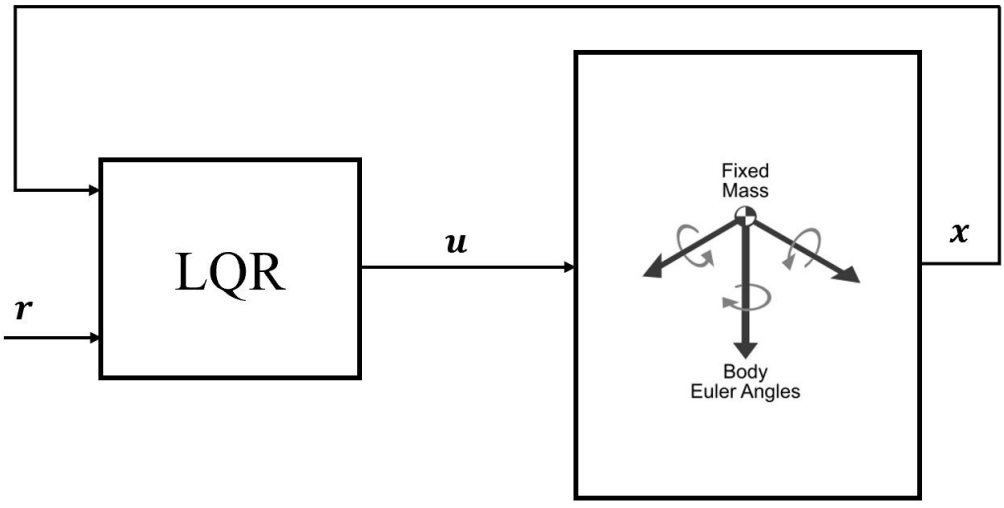

Figure 4: Stable drone model

We use the Model Predictive Control Toolbox from Mathworks [3]. It provides functions for designing and simulating controllers using linear and nonlinear MPC. The toolbox lets specifying models, horizons, constraints, and weights. It helps evaluate the controller performance by running closed-loop simulations. The guidance system described in section 5 can be illustrated as follow

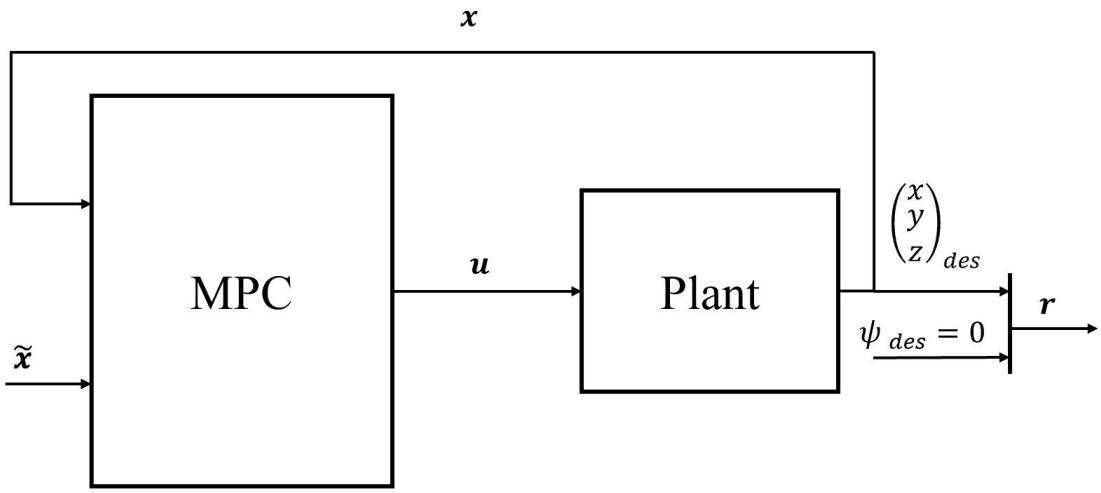

Figure 5: The trajectory planner

As mentioned in section 6.1, $\tilde{\mathbf{x}}$ represents the estimated target state vector. We first model the GV using the MATLAB Driving Scenario Designer application. We create the two following paths

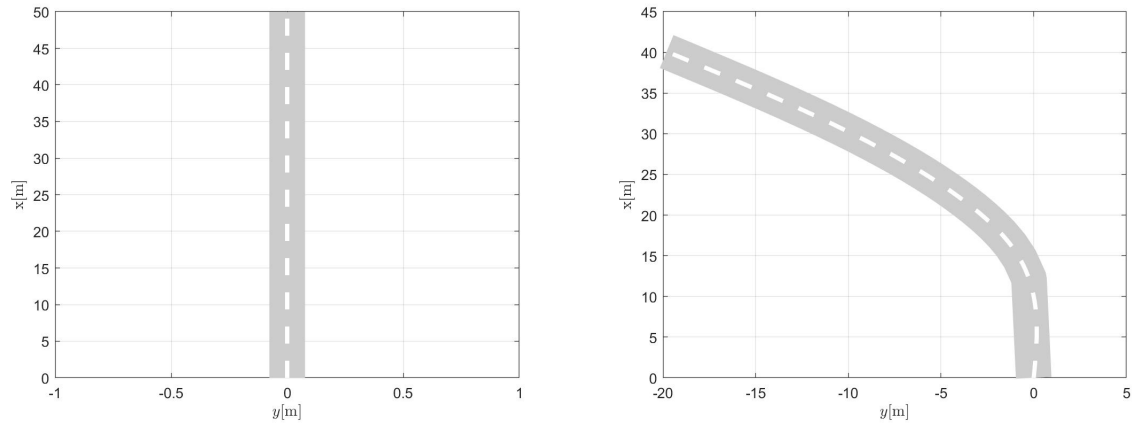

Figure 6: GV paths, Road1: straight and Road2: curved

The scenario reader in Simulink generates the true vehicle state vector. In [6] the helipad of the GV is equipped with a visual marker. A landing pattern detector provides a measurement of the target position. The car positions are then predicted using the Unscented Kalman Filter (UKF). In this work, we simulate the measurement by adding White Gaussian Noise (WGN) to the position coordinates generated by the scenario reader. We estimate $\tilde{\mathbf{x}}$ with a Linear Kalman Filter (LKF). The linear discrete model used is given as follows

$$\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{A}_d \boldsymbol{x}_k + \boldsymbol{B}_d \boldsymbol{u}_k \\ \boldsymbol{y}_{k+1} = \boldsymbol{C}_d \boldsymbol{x}_{k+1} + \boldsymbol{D}_d \boldsymbol{u}_k \end{cases} \tag{15}$$

$\mathbf{B}_d$ and $\mathbf{D}_d$ are both zero matrices. $\mathbf{A}_d$ and $\mathbf{C}_d$ are defined as

$$\mathbf{A}_d = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } \mathbf{C}_d = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

6.2 Analysis of the Results

In this section, we present and analyse our results. We test the above-described method for different driving scenarios. We define a successful landing as when the drone reaches a distance of 0.7m from the centre of mass of the GV.

Scenario 1
The GV is on Road 1 (straight). It starts with a velocity $v = 0m/s$. At $x = 50$m the target has to reach $v = 5m/s$.

Scenario 2
The GV is on Road 1 (straight). It starts with a velocity $v = 0m/s$. At $x = 50$m the target has to reach $v = 7m/s$.

The Quadrotor should have a fast system response, so it can follow the landing trajectory and catch up with the GV. The LQR controller described in section 3 provides a short response time. The MAV was however unable to land on the GV. we adjusted then the inputs of the MPC controller. We multiply the estimated car velocities with the factor of 1.2. We obtain the results represented in figure 7 and figure 8. During the first scenario, the MAV achieves a successful landing after 11 s at $x = 4.3$m. It was however unable to catch up with the car during the second scenario.

To examine the effect of the road's curvature, we perform two others simulations on Road 2 (curved).

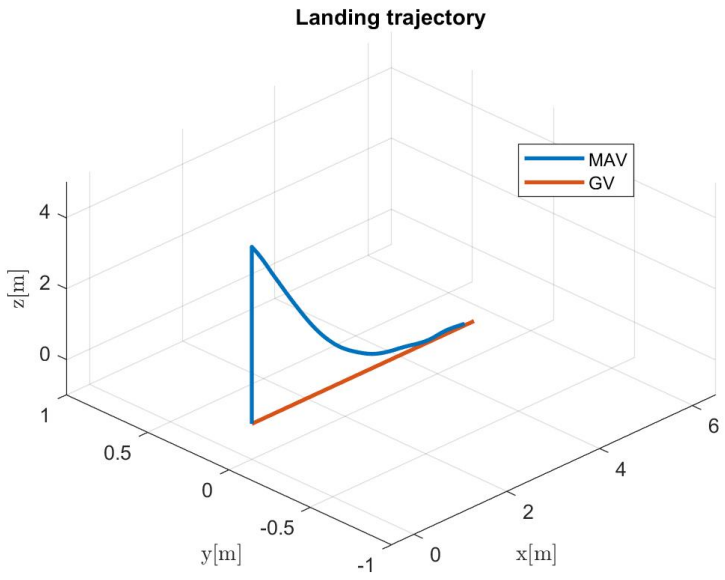

Figure 7: Scenario 1

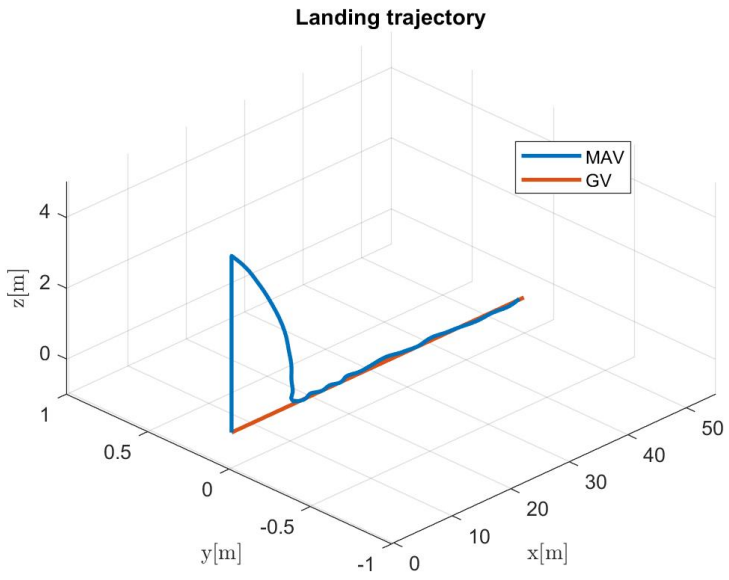

Figure 8: Scenario 2

Scenario 3
The GV is on Road 2 (curved). It starts with a velocity $v = 0m/s$. At $x = 10$m the target has to reach $v = 5m/s$ and then maintain a constant speed until reaching the end of the road.

Scenario 4
The GV is on Road 2 (curved). It starts with a velocity $v = 0m/s$. At about $x = 10$m the target has to reach $v = 5m/s$ and then accelerating until reaching $v = 7m/s$ at $x = 40$m.

Figure 9 shows a successful landing during scenario 3. The drone catches up with the GV after 13.5 s in (x, y) =(30,-9).

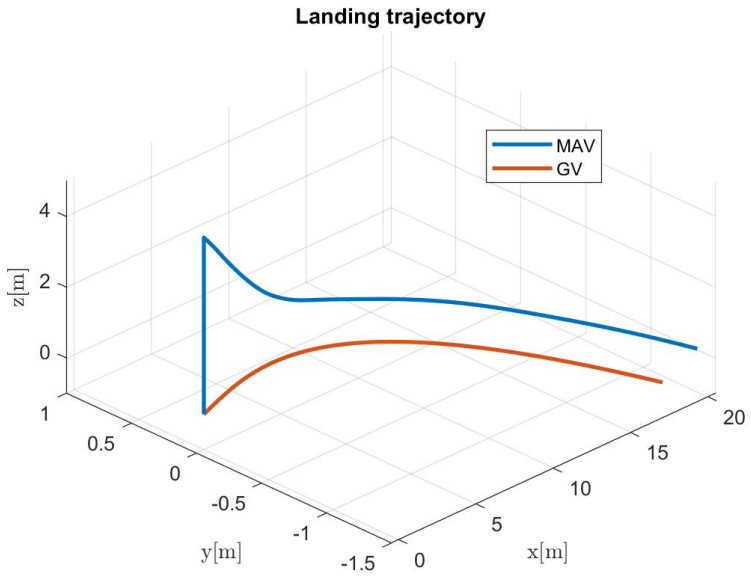

Figure 9: Scenario 3

During scenario 4, the MAV was unable to land on the target before reaching the end of the road. The experiment is presented in figure 10. We note that while applying a GV's speed $v \geq 10$ m/s on both roads 1 and 2, the system was unstable. To explain that, remember that the guidance system generates the landing trajectory in real-time simulation. For a higher car's speed, the waypoints are generated at a higher rate. The controller indeed has a short time response, but the relatively high overshoots cause large deviations from the desired path. Imagine that the drone has to reach a waypoint 1. And just before reaching the goal, other waypoints close to the first one are generated. So that the MAV needs to catch up on all the points while maintaining a smooth trajectory.

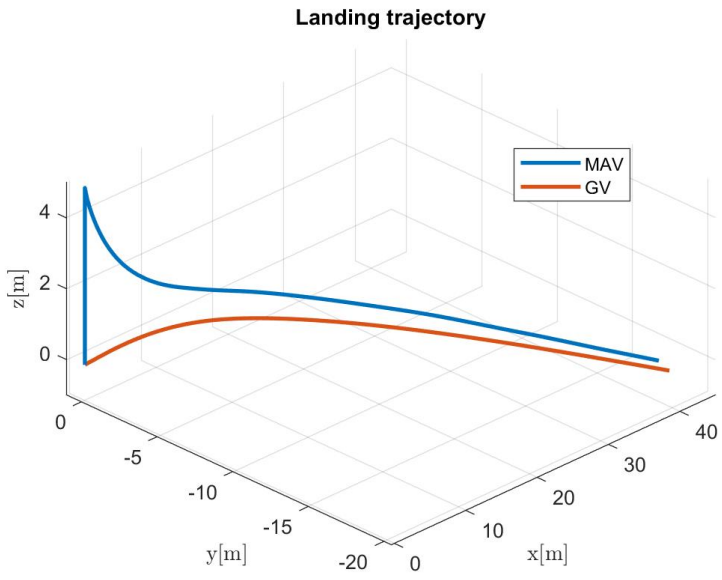

Figure 10: Scenario 4

In summary, we can affirm that the guidance system can generate acceptable and feasible landing trajectories during the given scenarios. It depends however on the ground vehicle's speed and the controller design. According to the results presented above, we recommend improving the controller's performance to reach a maximal overshoot of about 2% while maintaining a short time response of about 1.5 s. In [6] a nonlinear controller is used so that the system is capable of large deviations from the hover positions.

7 Conclusion

In this project, we have presented a method to perform an automatic landing of the MAV on a moving car. We have first created a nonlinear model of a quadrotor. We have then stabilized the drone with an LQR controller. Our guidance system is based on the approach described in [6]. It consists of an MPC tracker that controls a virtual model from the MAV's translation dynamics. We use the controlled state vector as a reference. We have demonstrated that the drone equipped with this trajectory planner can catch up with the car and performs a successful landing. A valuable lesson learned from this work is that these systems are complex and their performance is driven by a large number of variables such as the controller limits, the accuracy of the GV's pose estimation, the driving scenario, and the physical parameters of the drone, etc. For future work, we recommend improving the controller, modelling the motors, and implementing a performant navigation system. We recommend also designing a Proportional-Navigation (PN)-based guidance system described in [1], and comparing it to the system in this work. The simulation is

Bibliography

[1] Borowczyk A. et al. "Autonomous landing of a quadcopter on a high-speed ground vehicle". In: *Journal of Guidance, Control, and Dynamics* 40.9 (2017), pp. 2378–2385.

[2] Mathieu Ulysse Ashby. "Pilotage autonome agressif de drone dans un environnement de course". MA thesis. École Polytechnique de Montréal, 2020.

[3] MathWorks. *MPC*. [Online; accessed 05-05-2022]. 2022. URL: https://de.mathworks.com/help/mpc/gs/introduction.html.

[4] Duc-Tien Nguyen, David Saussie, and Lahcen Saydy. "Design and Experimental Validation of Robust Self-Scheduled Fault-Tolerant Control Laws for a Multicopter UAV". In: *IEEE/ASME Transactions on Mechatronics* 26.5 (2021), pp. 2548–2557. DOI: 10.1109/TMECH.2020.3042333.

[5] David Saussié. *AER8410 Lecture Notes*. École Polytechnique de Montréal. 2021.

[6] Baca Tomáš et al. "Autonomous landing on a moving vehicle with an unmanned aerial vehicle". In: *Journal of Field Robotics* (2019).