On the Lateral Non-Stabilizability of PID Controllers for Vision-Based Line-Following Multicopters through Roll Angle Setpoints

P. Van Holm, A. B. Mallikarjuna, C. Simon Soria and L. R. Lustosa [*]
ISAE-SUPAERO, Université de Toulouse, 31400 Toulouse, France

ABSTRACT

This paper investigates challenges in designing a single-input, single-output stabilizing controller for the lateral dynamics of vision-based autonomous line-following multicopters using roll angle setpoints. The system input is at the multicopter roll angle setpoint level. At the same time, its output is the outcome of a computer vision system that yields the distance of the line marker reference to the optical center of the image frame in pixels. After linearization, we show that the resultant transfer function is nonminimum phase. Furthermore, we show that pure Proportional-Integral-Derivative (PID) controllers render the closed-loop plant unstable for all gains values. To solve this issue, we propose an additional feedforward term, prove its effectiveness in theory, and validate it through multicopter simulations including multicopter dynamics with its internal cascaded stabilizing angle and angular rate loops. Furthermore, we present a computer vision algorithm for line tracking using contour analysis and Hue-Saturation-Value color segmentation. Lastly, we validate our controller in fully integrated system through flight tests in a realistic simulation of an indoors environment.

1 INTRODUCTION

This work studies the autonomous trajectory tracking problem of a multicopter drone using downwards-facing camera measurements of a visual line reference on the ground. Typical applications and scenarios include power transmission lines and pipeline following and inspection. While the commonplace strategy for solving this problem involves a guidance law that sets yaw angle setpoints to the flight management unit (FMU) — responsible for implementing low-level flight stabilization through cascaded angle and angular velocity proportional-integral-derivative (PID) controllers —, we examine in this work an alternative solution involving roll angle setpoints instead. However, since rolling towards a line forcibly casts the line even away in the image plane, this control strategy yields a nonminimal phase system. We prove this fact through linearization and study of the resulting transfer function. Furthermore, we show that no PID controller gains would stabilize the closed-loop system. This fact might explain why this strategy is not found in the literature.

1.1 Motivation

Our interest in line-following multicopters rose originally from IMAV 2024's indoor competition description [1], and is driven by the applications that can be derived from it. As explained in [2, 3], the wide application spectrum covers areas such as monitoring and surveillance, infrastructure inspection, precision agriculture, or real-time monitoring of road traffic. Inspection of power lines and pipelines [4, 5, 2] together with crop row detection [6] are among the most discussed topics regarding autonomous UAV's that require line-following capabilities.

Additionally, the interest of rejecting GNSS positioning and navigation follows from the unfeasibility of flying GNSS-reliant drones in deprived areas such as indoors or large cities [7]. These systems rely on external data, making them susceptible to hijacking or spoofing [8, 9, 3] and are expensive in terms of weight and power budget [7]. GNSS-reliant UAV's do not perform well at low altitudes either, and as [10] indicated in 2015, "Small autonomous drones flying at low altitude will need more complex levels of control autonomy and additional sensors to detect distances from the surrounding environment and perform safe and stable trajectories. Vision is a promising sensor modality for small drones".

1.2 Related work

Most previous studies deliberately avoid combining vision-based localization techniques and roll control. In [11, 4, 5, 6, 12, 13], the multicopter's orientation is adjusted using only yaw-angle inputs. In [14], the guidance law similarly sets the yaw-angle setpoints through a spatially discretized control action conditioned on the line's location inside the image. Another strategy employed by [15] is the use of an inward-tilted rotor configuration that allows for lateral movement with reduced induced roll or pitch angles. This strategy enabled the authors to minimize the distance to the line without significant roll, thereby avoiding the non-

[*]Email addresses: pieter.van-holm@student.isae-supaero.fr, apoorva.busunur-mallikarjuna@student.isae-supaero.fr, carmelo.simon-soria@student.isae-supaero.fr, leandro.ribeiro-lustosa@isae-supaero.fr

minimal phase system considered in this paper. To completely avoid the issue, the authors in [16, 17, 18] use an inertially stabilized gimbal camera, which keeps the camera's orientation aligned with the ground and independent of the roll, pitch, and yaw motion of the multicopter.

In contrast to these methods, the approach proposed by the authors in [19] for their autonomously landing multicopter aligns conceptually with the idea presented in this paper. In their work, the distance to the landing target is estimated using a downward-facing camera, which is then employed to control the quadcopter's XY-position via PID roll and pitch controllers. Similar to our approach, they account for the additional distance introduced by the quadcopter's tilt. However, in their method, this control strategy is integrated into an outer control loop, while an inner control loop utilizes gyros and optical flow to stabilize the quadcopter and maintain proximity to the target. Furthermore, a GNSS-dependent position controller is used beforehand to bring the quadcopter close to the target. The camera-based distance measurements are thus not used as the sole inputs for completely autonomous navigation, thereby ignoring the challenge this introduces. In contrast, our method relies exclusively on visual input for controlling the multicopter's positioning, offering a more direct approach to autonomous vision-based navigation and control.

In general, prioritizing roll-angle setpoints is beneficial, as rolling dynamics are typically faster and less prone to saturation. Notably, although this work focuses on the multicopter case, the problem is analogous to fixed-wing architectures, which similarly rely on roll maneuvers to execute turns.

1.3 *Paper structure*

This paper is structured as follows: Section 2 states the problem and the desired control structure architecture. Section 3 contains the main contributions of this paper and shows that although no PID controller gains stabilize the proposed system, a feedforward term exists that renders the unstable PID controllers closed-loop stable for a nonempty set of gains. Sections 4 and 5 illustrate a practical hardware/software implementation for the required computer vision algorithms and multicopter architecture for validating the theory. Section 6 explains the practical implementation and shows the results of flight simulations using the proposed roll setpoint controller with feedforward compensation. Section 7 closes the paper with the conclusion and discusses future work.

2 Problem statement

2.1 *Dynamics model*

The main focus of this work will be the control of the multicopter's side motion through roll. A simplified 2D model is considered, consisting of a fixed reference frame (Y_L, Z_L) aligned with the line, and a moving reference frame (y_c, z_c) aligned with the camera on the bottom of the multicopter. Figure 1 displays the system and the difference vector $\boldsymbol{p_{l/d}} = (0, D, -h)^T$ between the two reference frames. The difference vector considered in the fixed reference frame is noted $\boldsymbol{p^L_{l/d}}$ and in the camera's reference frame $\boldsymbol{p^c_{l/d}}$.

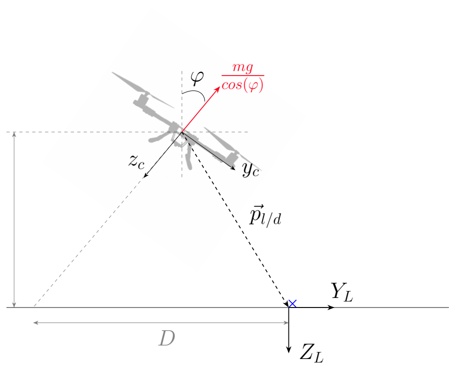

Figure 1: **2-Dimensional configuration:** Two-dimensional representation of the multicopter and the line.

For the camera, a simple pinhole model is considered. The distance d seen by the camera in pixels, as shown in Figure 2, corresponds to the real distance D according to the relation

$$\frac{d}{f} = \frac{D}{h} \Leftrightarrow d = f \cdot \frac{D}{h}. \tag{1}$$

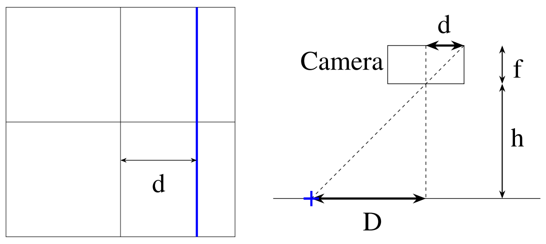

Figure 2: **Pinhole Camera model:** The distance to the line seen on the screen compared to the real the real distance to the line.

2.2 *State space model derivation*

A two-dimensional state vector $\boldsymbol{x} = (p_y \; v_y)^T$ composed of the position and velocity of the camera along the y-axis is chosen. Let the output of the system y be the measured distance d on the camera frame in pixels and the control input u the multicopter's roll angle φ. Assuming the multicopter will only undergo small angle variations, the derivative of the state vector is obtained:

$$\dot{\vec{x}} = \frac{d}{dt}\begin{pmatrix} p_y \\ v_y \end{pmatrix} = \begin{pmatrix} v_y \\ g \cdot \tan(\varphi) \end{pmatrix} \approx \begin{pmatrix} v_y \\ g \cdot \varphi \end{pmatrix}. \tag{2}$$

To express the measured distance d in the camera's frame, the vector $\boldsymbol{p^L_{l/d}}$ must be rotated around the x-axis according to the rotation matrix $\boldsymbol{R_L^c}$:

$$\vec{p}_{l/d}^{c} \equiv \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} = R_L^c \cdot \vec{p}_{l/d}^{L} = \begin{pmatrix} 0 \\ h\sin(\varphi) - D\cos(\varphi) \\ h\cos(\varphi) + D\sin(\varphi) \end{pmatrix}.$$

Knowing $\boldsymbol{p_{l/d}^c}$, Equation (1) is expressed in the camera's reference frame:

$$d = f \cdot \frac{d_y}{d_z} = f \cdot \frac{h\sin(\varphi) - D\cos(\varphi)}{h\cos(\varphi) + D\sin(\varphi)}. \tag{3}$$

Calculating the partial derivatives and evaluating at the point $(0, 0)$ yields

$$\begin{cases} \dot{\vec{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{bmatrix} 0 \\ g \end{bmatrix} u \\ \\ y = \begin{bmatrix} -\frac{f}{h} & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{bmatrix} f \end{bmatrix} u \end{cases}. \tag{4}$$

The controllability and observability matrices

$$\mathcal{O} = \begin{bmatrix} 0 & g \\ g & 0 \end{bmatrix} \; ; \; \mathcal{K} = \begin{bmatrix} -f & 0 \\ 0 & -f \end{bmatrix} \tag{5}$$

are full rank, showing that the system is both controllable and observable. The resulting transfer function is

$$\begin{aligned} G(s) &= C \cdot (s \cdot I - A)^{-1} \cdot B + D \\ &= f \cdot \left(\frac{s^2 - \frac{g}{h}}{s^2} \right). \end{aligned} \tag{6}$$

Equation (6) shows the presence of a zero in the right half of the s-plane, indicating a non-minimum phase system. To reduce the distance to the line d, the multicopter must roll in the direction of the line, creating the acceleration $a_y = g \cdot \tan(\varphi)$. This maneuver temporarily increases the error d, exemplifying the well-known characteristic behavior of non-minimum phase systems. Unlike many typical and counter-intuitive examples used to illustrate non-minimum phase behavior, the dynamics of this system is easily understood and visualized, making it particularly valuable for educational purposes.

3 CONTROL OF THE SYSTEM

This section outlines the controller design aimed at achieving the desired system behavior. Initially, the system's instability under a PID controller is demonstrated. Subsequently, a feed-forward mechanism is introduced to stabilize the system. The section concludes with simulation results, showcasing the control performance.

3.1 Instability under PID control

Using a PID controller of the form

$$C(s) = K_p + K_I \frac{1}{s} + K_d s$$

in a feedback control loop yields the following closed-loop transfer function:

$$\begin{aligned} H_{cl}(s) &= \frac{C(s)G(s)}{1 + C(s)G(s)} \\ &= \frac{s^4 + \left(\frac{K_p}{K_d}\right)s^3 + \left(\frac{K_I}{K_d} - \frac{g}{h}\right)s^2 - \left(\frac{K_p}{K_d}\frac{g}{h}\right)s - \left(\frac{K_I}{K_d}\frac{g}{h}\right)}{s^4 + \left(\frac{fK_p+1}{fK_d}\right)s^3 + \left(\frac{K_I}{K_d} - \frac{g}{h}\right)s^2 - \left(\frac{K_p}{K_d}\frac{g}{h}\right)s - \left(\frac{K_I}{K_d}\frac{g}{h}\right)}. \end{aligned} \tag{7}$$

A necessary condition for stability is that the coefficients of the polynomial in the denominator must be non-negative. This leads to the constraints

$$\begin{cases} \frac{K_p + 1/f}{K_d} > 0 \\ \frac{K_I}{K_d} > \frac{g}{h} \\ \frac{K_p}{K_d} < 0 \\ \frac{K_I}{K_d} < 0 \end{cases}. \tag{8}$$

Considering the case where $K_d > 0$, the combination of the second and last constraint of Equation (8) lead to an empty set for the possible values K_I can take. Considering the second case where $K_d < 0$, the directions of the inequalities are flipped and a similar contradiction is obtained. As a result, a PID controller is not able to stabilize the system in a feedback control loop.

3.2 Feed-forward compensation combined with a PD controller.

Due to the inability of the PID controller to stabilize the system, an alternative control strategy is investigated. In particular, a feed-forward compensation combined with a PD controller is introduced.

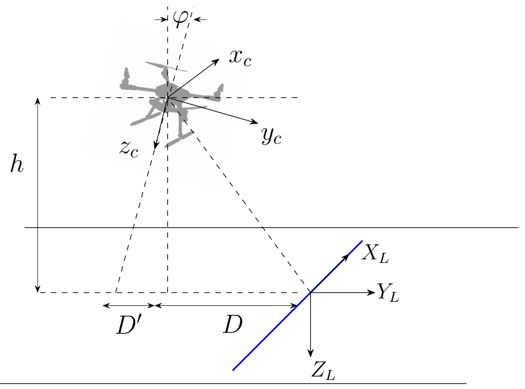

Figure 3: **3-Dimensional configuration:** Visualization of the additional error D' created by the rolling of the multicopter.

A measurement h_2 of the multicopter's height h_1 is used and the additional error D' created by the roll angle, as illustrated in Figure 3, can be fed forward. The additional error is approximated as $D' = h\tan(\varphi) \approx h\varphi$, making again the assumption of small angles. By subtracting D' to the output of the system, the remaining error will be the real distance D between the multicopter and the line. The resulting control scheme is shown in Figure 4.

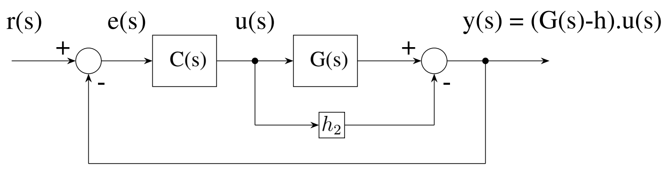

Figure 4: **Control scheme:** Addition of the feed-forward term to compensate the error.

By considering a PD controller

$$C(s) = K_p + K_d s \ , \tag{9}$$

the closed-loop transfer function becomes

$$\begin{aligned} H_{cl}(s) &= \frac{C(s)(G(s)-h_2)}{1+C(s)(G(s)-h_2)} \\ &= \frac{s^3 + \left(\frac{K_p}{K_d}\right)s^2 - \left(\frac{fg}{h_1(f-h_2)}\right)s - \left(\frac{K_p}{K_d}\frac{fg}{h_1(f-h_2)}\right)}{s^3 + \left(\frac{K_p}{K_d} + \frac{1}{K_d(f-h_2)}\right)s^2 - \left(\frac{fg}{h_1(f-h_2)}\right)s - \left(\frac{K_p}{K_d}\frac{fg}{h_1(f-h_2)}\right)} \ , \end{aligned} \tag{10}$$

where a distinction is made between the actual height of the multicopter h_1 present in the system Equation (6) and the measured height h_2 to emphasise that these are different values.

The Routh-Hurwitz criterion for a third order transfer function $d(s) = s^3 + a_2 s^2 + a_1 s + a_0$ ensures that the closed-loop system is stable if and only if the following two conditions are met: $a_2, a_1, a_0 > 0$ and $a_2 \cdot a_1 > a_0$. The transfer function (10) is rewritten to

$$H_{cl} = \frac{s^3 + \lambda s^2 - \gamma s - \lambda\gamma}{s^3 + (\lambda+\alpha)s^2 - \gamma s - \lambda\gamma} \tag{11}$$

by defining

$$\lambda = \frac{K_p}{K_d} \ ; \ \gamma = \frac{fg}{h_1(f-h_2)} \ ; \ \alpha = \frac{1}{K_d(f-h_2)} . \tag{12}$$

Consequently, the first Routh-Hurwitz condition yields the constraints:

$$\begin{cases} \lambda + \alpha > 0 \\ \gamma < 0 \\ \lambda\gamma < 0 \end{cases} \Leftrightarrow \begin{cases} \lambda > -\alpha \\ \gamma < 0 \\ \lambda > 0 \end{cases} . \tag{13}$$

The second constraint is met when $h_2 >> f$ is considered. The hovering height of the multicopter must be greater than the camera's focal length, which is the case for real systems. The second Routh-Hurwitz condition yields:

$$(\lambda+\alpha)(-\gamma) > (-\lambda\gamma) \Leftrightarrow \alpha > 0 \ , \tag{14}$$

which enforces that K_d must be negative. Consequently, the third constraint of Equation (13) dictates that K_p must also be negative. The first constraint of Equation (13) yields

$$K_p < \frac{1}{h_2 - f} \ , \tag{15}$$

which is always met when $h_2 >> f$ and $K_d < 0$. As a result, the addition of the feed-forward term allows the stabilization of the system using the PD controller in Equation (9).

The robustness of to feedback loop to variations of the measurement h_2 must be ensured. Considering small variations δh_1 such that $h_2 = h_1 + \delta h_1$, Equation (12) changes to

$$\lambda = \frac{K_p}{K_d} \ ; \ \gamma = \frac{fg}{h_1(f-h_1-\delta h_1)} \ ; \ \alpha = \frac{1}{K_d(f-h_1-\delta h_1)} . \tag{16}$$

The previously obtained conditions (13) will be violated when $\delta h_1 < 0$ and $|\delta h_1| > |h_1|$, which goes against the assumption that δh_1 is a small variation.

3.3 Simulation

A Matlab Simulink model incorporating the nonlinear dynamics of a quadrotor and the Pixhawk attitude controller is implemented as a first proof of concept for the proposed controller. To model realistic operational constraints, a roll angle limit of 0.26 radians ($\approx 15°$) has been imposed on the controller output. This constraint ensures that the roll angle remains within realistic bounds during simulation.

The system's performance in tracking a line with abrupt changes is evaluated. The results from the simulations, illustrated in Figure 5, indicate that roll adjustments allow the multicopter to follow the line and react to the abrupt changes.

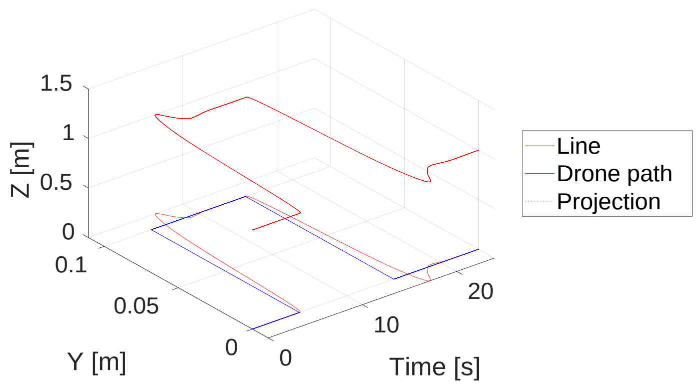

Figure 5: **Matlab simulation results:** Tracking of an abrupt changing line with gains $K_p = -130$ and $K_d = -100$. The control is only performed in the y-direction and the fictional z component is only for visualization purposes.

4 Line detection algorithm

This section details the vision-based line detection algorithm essential for the proposed control strategy, which relies

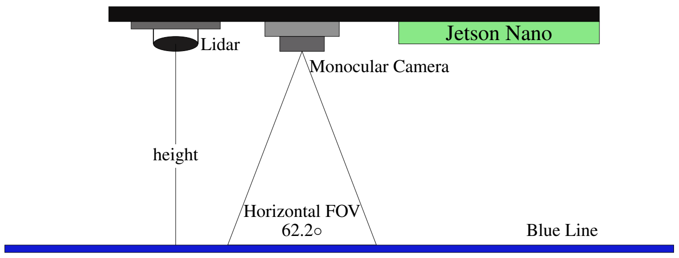

Figure 6: **Camera setup:** Positioning of the monocular camera and Lidar on the frame for effective data collection.

Table 1: Sony IMX219 camera parameters.

sensor width w_s	**sensor height** h_s	**camera focal length** f
$3.68\ mm$	$2.76\ mm$	$3.04\ mm$

on continuous distance measurements from the camera to the line.

4.1 Camera Setup

To achieve a wide field of vision for detection, a camera sensor with a fixed focal length and wide lens angle is required. For example, a downwards pointing monocular Raspberry Pi Camera Module v2 with a Sony IMX219 CMOS sensor is considered in this study. The dimension of the camera are given in Table 1.

In a static image capturing mode, the camera has an active pixel area of $3280 \times 2464\ p$, which can be used to determine the camera's horizontal and vertical field of view:

$$\begin{aligned} HFOV &= 2 \cdot \arctan\left(\frac{w_s}{2f}\right) = 62.37^\circ \\ VFOV &= 2 \cdot \arctan\left(\frac{h_s}{2f}\right) = 48.83^\circ \ . \end{aligned} \tag{17}$$

4.2 Contour Analysis and HSV Color Segmentation

The line-detection algorithm aims to extract the 2D coordinates of the middle of the line by identifying it's contour within the image frame. These coordinates enable the calculation of the distance d to the center of the the image.

The image obtained from the monocular camera is represented in the RGB color model, where each color is defined by its red, green, and blue spectral components. Luminosity significantly influences the accuracy and reliability of object detection within this model. To isolate therefore a specific color and improve the robustness of the detection algorithm, the image is converted to the HSV (Hue, Saturation, Value) color space. A mask is applied to this HSV image and filters out all colors except those within the specified color range. Additionally, morphological operations like erosion and dilation are applied on the mask to improve the image quality through noise reduction. These computer-vision algorithms increase the visual distinctness of the line, making it better discernible against the background.

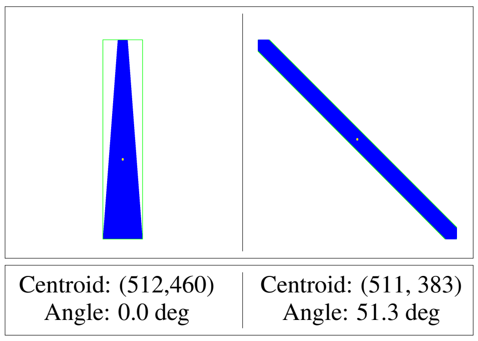

Figure 7: **Line detection:** Output of the line-detection algorithm for a straight line and a line with an angle w.r.t. the camera.

OpenCV functions are used to identify and filter contours from the processed mask, applying a specified minimum area threshold to exclude noise and irrelevant contours. Following this, the contour with the largest area is determined and approximated by a minimum area bounding rectangle. Parameters of this rectangle, like it's corner coordinates, allow to compute the centroid and the angle of the line. Figure 7 demonstrates a simulated scenario with the multicopter positioned at two distinct locations relative to the detected line.

5 HARDWARE AND SOFTWARE ARCHITECTURE

A big gap still exists between the proposed control strategy and a practical implementation. This section proposes a hardware and software architecture that allows to implement a autonomous line-following multicopter.

5.1 Hardware

The proposed multicopter hardware features a Pixhawk 6X Autopilot Flight Controller integrated into a multicopter frame similar to the Holybro X500 V2 quadcopter frame, which is the one used in our lab. A platform is added to the bottom of the multicopter's frame to allow placement of the camera and other sensors.

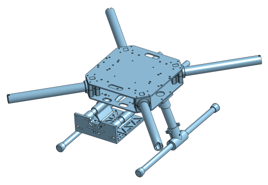

Figure 8: **Frame:** Holybro X500 V2 quadcopter

The Pixhawk autopilot system manages primary decision-making tasks, control, and guidance, while a companion computer (ex. a Jetson Nano) provides computer-vision capabilities such as object detection and line tracking. Figure 9 shows how the Pixhawk controller is linked to a telemetry module for data transmission to a ground station. An RC receiver

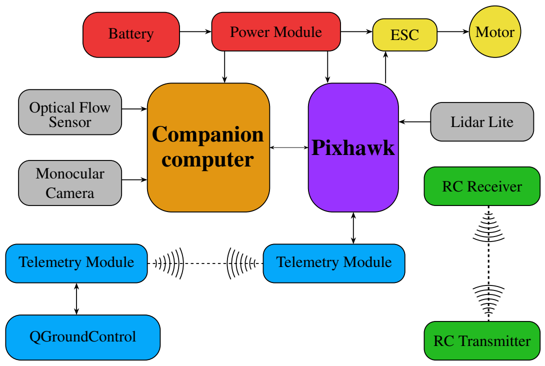

Figure 9: **Hardware:** Key components needed of the avionics architecture.

must be considered to allow switching to manual control for safety. Finally, a high accuracy range sensor like a Lidar-Lite can be used to achieve millimeter-level precise height measurements.

5.2 *Software Network*

The PX4 software contains a flight stack consisting of cascaded controllers with an estimation library that uses an Extended Kalman Filter for state estimation. Flight commands and sensor information are received by the flight controller from input interfaces such as the Radio control receiver, telemetry link, and the companion computer. Communication robustness between various components in the modular system is ensured by the MAVLink messaging protocol. The companion computer transmits the position and angle information to the Pixhawk flight stack through these MAVLink messages, as visualized in Figure 10.

The Pixhawk flight control architecture for the quadcopter consists of a cascade of P and PID controllers. The angle control and altitude control functionalities are managed by two separate Proportional(P) controllers at different locations in the pipeline, each operating at distinct frequencies of 250Hz and 50Hz respectively. Based on the received information, the Flight Control algorithm computes the attitude setpoint quaternion q_{sp} and sends it to the Pixhawk cascade controllers through the uORB subscribe/publish messaging protocol. By directly transmitting an attitude setpoint, the control system bypasses the initial stages of the Pixhawk cascade controllers, specifically the position (P) controller and the velocity (PID) controller.

6 PRACTICAL IMPLEMENTATION

In this last chapter, the proposed hardware and software is implemented in a high-fidelity simulator with realistic physics in order to close the gap between the simulation results and a real-world application. Results are obtained to demonstrate that the proposed vision-based control method performs successfully when integrated into a complete system.

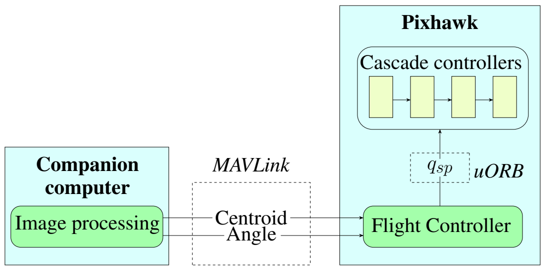

Figure 10: **Software:** Communication pipeline between the companion computer and the flight controller.

6.1 *Simulation pipeline*

The simulation made use of the Software-In-The-Loop (SITL) framework provided by PixHawk, which integrates the PixHawk autopilot flight stack with the Gazebo-Classic simulator. A simulated indoor environment featuring blue 19mm lines on the ground was recreated within Gazebo-classic, as shown in Figure 11. The 3DR Iris quadcopter model, already implemented in the PX4 Firmware, was used as the base multicopter model. A downward facing camera with the same specifications as the previously mentioned RaspberryPi Camera Module V2 and a range sensor were incorporated to this quadcopter model.

The video stream captured by the camera within the simulation is streamed via *gstreamer* to UDP port 5600. In parallel, a Python function established a connection to this port to retrieve the video, acting therefore as the companion computer. The OpenCV image processing library was used to compute the centroid and angle of the recorded line in each frame. The x and y coordinates of the centroid were then used to calculate the distance between the quadcopter and the line. This calculated data was subsequently transmitted back to the Pixhawk autopilot via a custom MAVLink message using pyMAVLink. Upon receipt, the Pixhawk autopilot translated the MAVLink message into a custom uORB topic for internal utilization. This streamlined communication pipeline enabled seamless communication between the Gazebo simulator, the image processing algorithm and the PX4 autopilot.

6.2 *Control algorithms*

The control algorithms were implemented in a custom Flight Task, which has been added to the Altitude Flight mode. Since the quadcopter is controlled through attitude setpoints, four separate control algorithms were used to control the thrust, pitch, yaw and roll inputs respectively.

The thrust controller was particularly crucial, as it was noticed that its performance influenced the roll controller's performance. The camera's field of view and image processing capabilities necessitate the line features to be stable and sufficiently large for accurate detection. Furthermore, without height compensation, roll inputs can cause the quadcopter to descend unexpectedly. A PID-controller, operating at a frequency of 100 Hz, was chosen and manually fine-tuned to achieve stable hovering at a desired height setpoint. Although good performance was achieved, its control performance was limited and was not without imperfections.

The roll controller, operating at 10Hz, was implemented as prescribed in Section 3. The measured angle of the line could be directly fed-forward to the yaw input, while the pitch input was simply held constant.

The resulting pitch, yaw and roll Euler angles were converted to a quaternion and together with the thrust setpoint were packed into the attitude setpoint. Finally, disabling PX4 position controller and transitioning to Altitude mode, enabling the custom Flight Task, allowed for the autonomous system to execute the developed control strategy.

6.3 Experiments and results

The performed simulation mimics the simulation of Section 3.3, where a straight line with an abrupt 1m change must be followed. A constant pitch value of $\theta = 0.008\ rad$ and a roll saturation of $\varphi = 0.3\ rad$ were used. These small angles were chosen to prevent the multicopter from exhibiting aggressive dynamics.

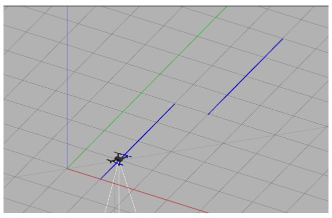

Figure 11: **Simulator:** View of the line configuration in the Gazebo simulator.

The obtained results from the simulation are illustrated in Figure 12. The multicopter is able maintain accurate tracking of the line and quickly correct the sudden 1-meter shift. Compared to the theoretical simulation in Section 3.3, where the coupling of roll control with other complex dynamics of the multicopter was not addressed, there is noticeably more oscillatory behavior observed. Overall, the drone is able to successfully track the proposed configuration, validating the proposed roll-control method. At the end of the simulation, when the multicopter is commanded to land and the controller is deactivated, the trajectory deviates from the blue markers.

By achieving consistent tracking in a realistic simulation of the fully integrated system, the control algorithm's effectiveness in autonomously guiding the multicopter using only visual input and range sensing is substantiated. Naturally, this assumes the availability of suitable line markers for the image processing algorithm.

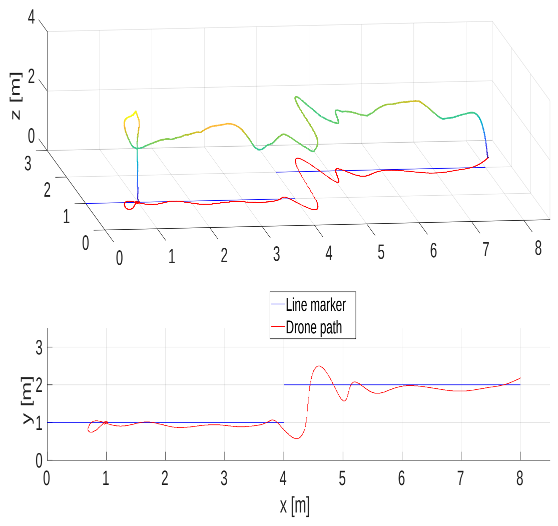

Figure 12: **Simulation results:** The colored path illustrates the three-dimensional trajectory of the multicopter as it follows the blue line markers on the ground, with the color gradient representing the drone's altitude. The red curves denote the projection of this trajectory onto the XY-plane.

7 CONCLUSION

This paper addressed the challenge of designing visual line-following autonomous guidance laws using roll setpoints instead of the commonplace yaw angle alternative. We show that such a system could not be closed-loop stable with classical PID controllers. However, we propose a feedforward term that renders the system closed-loop stable for a nonempty set of PID controller gains. In particular, we show and discuss a PD controller design. We first evaluate the efficiency of the proposed control strategy using a theoretical simulation in Matlab that includes the nonlinear dynamics of the multicopter and incorporates the Flight Management Unit (Pixhawk) controllers' architecture. Finally, the theoretical analyses are validated by practically implementing a quadcopter demonstrator in a high-fidelity simulation. The results show that the proposed control method is successful for line-following missions.

Having detailed the complete system design—from hardware and software to practical implementation—our future work will focus on transitioning this design to the x500 quadcopter present in our lab. The control method will be incorporated as the baseline autonomous controller with a state machine to perform other, more complex autonomous tasks. We aim to assess how the control method performs under real-world conditions, addressing any discrepancies between simulation and practice.

REFERENCES

[1] `https://2024.imavs.org/`.

[2] Syed Agha Hassnain Mohsan, Muhammad Asghar Khan, Fazal Noor, Insaf Ullah, and Mohammed H Alsharif. Towards the unmanned aerial vehicles (uavs): A comprehensive review. *Drones*, 6(6), 2022.

[3] Hazim Shakhatreh, Ahmad H. Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *IEEE Access*, 7:48572–48634, 2019.

[4] Yago MR da Silva, Fabio AA Andrade, Lucas Sousa, Gabriel GR de Castro, João T Dias, Guido Berger, José Lima, and Milena F Pinto. Computer vision based path following for autonomous unmanned aerial systems in unburied pipeline onshore inspection. *Drones*, 6(12):410, 2022.

[5] Alexander Cerón, Iván Mondragón, and Flavio Prieto. Onboard visual-based navigation system for power line following with uav. *International Journal of Advanced Robotic Systems*, 15(2):1729881418763452, 2018.

[6] Maik Basso and Edison Pignaton de Freitas. A uav guidance system using crop row detection and line follower algorithms. *Journal of Intelligent & Robotic Systems*, 97(3):605–621, 2020.

[7] Amer Al-Radaideh and Liang Sun. Self-localization of tethered drones without a cable force sensor in gps-denied environments. *Drones*, 5(4):135, 2021.

[8] Hamid Didari Khamseh Motlagh, Faraz Lotfi, Hamid D. Taghirad, and Saeed Bakhshi Germi. Position estimation for drones based on visual slam and imu in gps-denied environment. In *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*, 2019.

[9] AbdelRahman Eldosouky, Aidin Ferdowsi, and Walid Saad. Drones in distress: A game-theoretic countermeasure for protecting uavs against gps spoofing. *IEEE Internet of Things Journal*, 7(4):2840–2854, 2020.

[10] Dario Floreano and Robert J Wood. Science, technology and the future of small autonomous drones. *nature*, 521(7553):460–466, 2015.

[11] Tien-Loc Le and Nguyen Huu Hung. Trajectory tracking control of a line-following quadcopter using multilayer type-2 fuzzy petri nets controller. *Neural computing & applications*, 36:13617–13627, 2024.

[12] Shicheng Wu, Rui Li, Yingjing Shi, and Qisheng Liu. Vision-based target detection and tracking system for a quadcopter. *IEEE Access*, 9:62043–62054, 2021.

[13] Gomez A. Sotomayor J. and Castillo A. Visual control of an autonomous aerial vehicle for crop inspection. *Revista Politecnica*, 33(1), 2014.

[14] David García-Olvera, Armando Hernández-Godínez, Benjamín Nicolás-Trinidad, and Carlos Cuvas-Castillo. Line follower with a quadcopter. *PÄDI boletín científico de ciencias básicas e ingenierías del ICBI*, 7(14):30–36, 2020.

[15] Alexandre S. Brandão, Felipe N. Martins, and Higor B. Soneguetti. A vision-based line following strategy for an autonomous uav. 02:314–319, 2015.

[16] Alejandro Rodriguez-Ramos, Adrian Alvarez-Fernandez, Hriday Bavle, Pascual Campoy, and Jonathan P How. Vision-based multirotor following using synthetic learning techniques. *Sensors*, 19(21):4794, 2019.

[17] Torsten Merz, Simone Duranti, and Gianpaolo Conte. Autonomous landing of an unmanned helicopter based on vision and inertial sensing. In *Experimental Robotics IX: The 9th International Symposium on Experimental Robotics*, 2006.

[18] A. Masselli K. E. Wenzel and A. Zell. Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle. *Journal of Intelligent Robotic Systems*, 61(1-4):221–238, 2010.

[19] T. K. Venugopalan, Tawfiq Taher, and George Barbastathis. Autonomous landing of an unmanned aerial vehicle on an autonomous marine vehicle. In *2012 Oceans*, 2012.