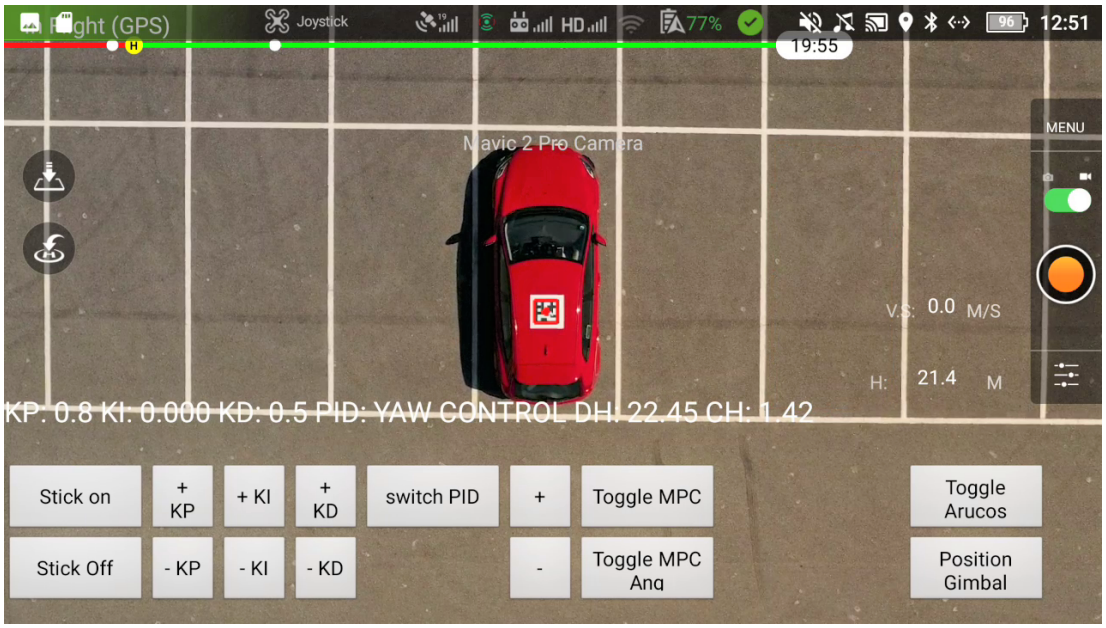
CHALMERS
UNIVERSITY OF TECHNOLOGY




# A Drone-based Approach to Monocular-Camera Distance Estimation using Computer Vision

## A Tool for Surround View System Validation

Master's thesis in Systems, Control and Mechatronics

Joakim Cewers Bredberg
Jonatan Flink Ornung

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

# A Drone-based Approach to Monocular-Camera Distance Estimation using Computer Vision

Aerial Tracking for Autonomous Parking: Generating Ground-level Distance Data for Surround View System Validation

Joakim Cewers Bredberg
Jonatan Flink Ornung


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
*Division of Systems and Control*
Chalmers University of Technology
Gothenburg, Sweden 2025

A Drone-based Approach to Monocular-camera Distance Estimation using Computer Vision
Aerial Tracking for Autonomous Parking: Generating Ground-level Distance Data for Surround View System Validation
Joakim Cewers Bredberg
Jonatan Flink Ornung



Supervisors: Jonas Fredriksson, Chalmers
Altug Altetmek, Aptiv
Examiner: Jonas Fredriksson, Department of Systems Control and Mechatronics

Master's Thesis 2025
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Drone camera preview in a custom controller app while autonomously tracking a target car during parking maneuvers.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

A Drone-based Approach to Monocular-Camera Distance Estimation using Computer Vision
Aerial Tracking for Autonomous Parking: Generating Ground-level Distance Data for Surround View System Validation
Joakim Cewers Bredberg
Jonatan Flink Ornung
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology

# Abstract

Autonomous parking is a field within ADAS/AD that requires exceptionally high spatial precision, given the close proximity to vehicles, people, and other obstacles. Surround-view camera systems are often used to both map the environment and estimate the vehicle's position by creating a stitched birds-eye view image of the vehicle's surroundings. To aid the development and evaluation of these systems, high-accuracy distance measurements between the car and its environment are needed. This thesis presents a drone-based system that utilizes a monocular camera to estimate ground-level distances in parking environments. Using car-mounted AR fiducial markers as reference objects, the system achieves centimeter-level accuracy at distances up to 10 meters relative to the car. In addition to the distance estimation pipeline, a vision-based drone controller was implemented to track a target car and maintain optimal positioning for video data collection during dynamic parking scenarios. Real-world testing demonstrates that the proposed method yields consistent results, highlighting its potential as a low-cost and flexible tool for generating ground truth distances. To our knowledge, this is the first application of monocular drone-based distance estimation for the development and validation of ADAS/AD systems.

Keywords: Monocular Camera Distance Estimation, ADAS, Autonomous Parking, Surround View 360-camera, Vision-based Drone Control, PID

# Acknowledgements

We would like to thank our supervisor and examiner Jonas Fredriksson for his support and pedagogical guidance that substantially improved the thesis project. We genuinely appreciate your mentorship and the rewarding discussions we've had.

Furthermore, we express our gratitude to our supervisor Altug Altetmek and the people at Aptiv who went out of their way to facilitate this thesis project as well as creating a welcoming working environment.

This thesis concludes our years at Chalmers, filled with challenges, learning, and a lot of laughter. We would like to thank everyone who has been part of this journey for putting up with us during challenging times and with our bad jokes, but more importantly, for creating good times and lifelong memories.

To our families – Thank you for your endless support throughout our journey at Chalmers. We could not have done it without you.

Joakim Cewers Bredberg and Jonatan Flink Ornung, Gothenburg, June 2025

# List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

| | |
|---|---|
| ADAS/AD | Advanced Driver-Assistance Systems / Autonomous Driving |
| AR | Augmented Reality |
| ARE | Absolute Relative Error |
| BEV | Bird's-Eye View |
| CLAHE | Contrast Limited Adaptive Histogram Equalization |
| FOV | Field of View |
| LiDAR | Light Detection And Ranging |
| MAE | Mean Absolute Error |
| MPC | Model Predictive Control |
| PID | Proportional–integral–derivative |
| RMS | Root Mean Square |
| SDK | Software Development Kit |
| SIFT | Scale Invariant Feature Transform |

# Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

## Sets

| | |
|---|---|
| $x$ | State vector |
| $u$ | Input vector |
| $y$ | Output vector |

## Parameters

| | |
|---|---|
| $c_{overhang}$ | Car rear overhang |
| $c_{length}$ | Car length |
| $c_x, c_y$ | Principal point coordinates (optical center) |
| $D$ = Distortion vector | |
| $f_w, f_h$ | Frame width and frame height |
| $f_x, f_y$ | Focal lengths in pixels along the x and y axes |
| $K$ = Camera matrix | |
| $K_p$ | Proportional gain |
| $K_i$ | Integral gain |
| $K_d$ | Derivative gain |
| $k_1, k_2, k_3$ | Radial distortion coefficients |
| $l_{bb}$ | Bounding box length |
| $P_{desired}$ | Desired brightness |
| $p_1, p_2$ | Tangential distortion coefficients |

| | |
|---|---|
| $w_{bb}$ | Bounding box width |

# Variables

| | |
|---|---|
| $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ | State space matrices |
| $\vec{C} = \begin{bmatrix} c_x \\ c_y \end{bmatrix}$ | Car's position vector |
| $C, G$ | Control and system transfer functions |
| $d_{px}$ | Distance in pixels |
| $d_{cm}$ | Distance in centimeters |
| $e$ | Control error |
| $\vec{e_p}$ | Positional error vector |
| $e_\psi$ | Rotational error |
| $e_h$ | Altitude error |
| $h$ | Drone height |
| $M_P, PO$ | Peak overshoot, percent overshoot |
| $m_{px}$ | Marker pixel circumference |
| $\vec{O} = \begin{bmatrix} o_x \\ o_y \end{bmatrix}$ | Origin's position vector |
| $pixelPerCm$ | Number of pixels per centimeter for each frame |
| $t$ | Time |
| $t_r$ | Rise time |
| $t_s$ | Settling time |
| $\vec{P} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$ | Drone's position vector |
| $u_c$ | Control input |
| $y_{ref}$ | Reference output |
| $y$ | System output |
| $y_m$ | Measured output |
| $\psi$ | Drone yaw angle |

# Contents

# List of Figures

# List of Tables

# 1
# Introduction

With the rise of autonomous driving and advanced driver-assistance systems (ADAS), the demand for accurate data to support vehicle perception has grown significantly. Parking is a specific area within ADAS that requires exceptionally high spatial precision due to the close proximity to surrounding vehicles and obstacles. In modern vehicles, surround-view camera systems are commonly used for enhanced vehicle perception during parking. However, achieving the required level of precision is challenging in practice, as sensor noise and environmental variability introduce significant uncertainty. This is mainly due to the difficulty of extracting reliable measurements from the cameras, which are hindered by their low perspective, high lens distortion, and imperfect stitching, all of which impair accurate environmental mapping.

To improve surround view systems, accurate ground truth distance data consisting of the vehicle's position relative to the parking area needs to be collected and evaluated, which in turn will enable improvements to the surround view system's distance measurements. A drone-mounted downward-facing camera offers a promising solution, providing a stable bird's-eye view (BEV) image, similar to a car's surround-view camera BEV image. However, the drone has a more comprehensive and reliable high-resolution image, better suited for extracting relative distances to parking lines and neighboring cars. This higher vantage point enables better feature tracking, better visibility of parking spaces, and a more accurate understanding of the environment, improving data-driven decision-making for parking functionalities.

## 1.1 Purpose and Objective

This thesis aims to develop a drone-based system that gathers video data of parking scenarios and accurately extracts real-world distances from this data. The objective is to implement a drone controller that tracks and follows a target car autonomously while recording video. Using computer vision, the video data will be post-processed to find distances between different features in the recorded frames. In current surround view systems, the relative position between the car and the parking spot corners is the key feature for accurately positioning the car in the world frame. Hence, in the post-processing, we measure the distances only between these features. We define the accuracy of the distance measuring algorithms as the primary performance metric for our system. To enable desired accuracy, it is crucial to have both accurate distance estimation algorithms, a responsive controller, and a stable

camera view.

The objective is summarized into three research questions:

- To what accuracy can real-world distances between features in the frame be extracted from the video data collected from a single drone-mounted BEV camera?
- What are the benefits and limitations of using a drone's bird's eye view with computer vision to determine physical distances for evaluation of ADAS parking systems?
- What is the qualitative and quantitative performance of a decoupled PID controller used for drone tracking of a moving car?

## 1.2 Contributions

In this thesis, we present a novel drone-based system for estimating ground plane distances, tailored for parking-ADAS applications. The distance estimations achieve centimeter accuracy for distances up to 10 meters and are sufficiently accurate to serve as ground truth distances during the development of surround view systems used in modern cars.

## 1.3 Delimitations

In this project, we will limit the scope to address distance estimations to parking spots with specific specifications, focusing on rectangular parking spaces with four solid, clearly visible boundary lines that are not obstructed or occluded by other vehicles or debris. The system will handle all visible parking spots of this type simultaneously. These can be located in large outdoor parking lots as well as custom-configured spaces outlined by movable lane markers. The project will not handle residential area parking, street parking, or other types of parking areas.

The tracking controller for the drone should be adapted for real-time tracking of the target vehicle during standard parking velocities and maneuvers. No attempt will be made to control the car or establish any telemetry connection to the vehicle.

All video data is recorded while the drone is in flight and subsequently processed offline on separate computing systems, independent of the drone's remote controller hardware. Hence, the processing will not be handled in real time.

Throughout the entire project, we will exclusively utilize the DJI Mavic 2 Pro drone, and our solution will be specifically tailored to this drone using the DJI Mobile SDK.

## 1.4 Related Work

The dual objective of the thesis, to both implement a drone controller and develop monocular camera distance estimation algorithms, necessitates a review of literature across multiple research domains. This subsection examines relevant work in vision-based drone control, distance estimation using monocular cameras, and data collection for ADAS development.

Vision-based sensing and control for drones is a rapidly evolving field. Current research is often focused on high-velocity navigation and obstacle avoidance using Machine Learning [3], which builds on previous research using classical model-based methods, often model predictive control (MPC) [4][5][6]. However, our thesis focuses on slow-speed applications where accuracy and stability are of high importance. Several papers have investigated methodologies for improving these specific aspects. In [7], the author focuses on maximizing state estimation accuracy by using a Kalman filter whose output is fed to an MPC. Accurate state estimation is crucial for data collection in monitoring, inspection, and patrolling applications. Jeon and Jin Kim [8] present another method for similar drone control applications, focusing on autonomously following an object as smoothly as possible for videography by integrating a target motion prediction module. A more general method for drone control implemented by Reizenstein [9], is a simplified system model, that maps throttle and angular control inputs to positional and vertical control through a black box model, developed by Kugelberg [10]. The drone's latitudinal position $x$, longitudinal position $y$, and height $z$, are controlled independently using PID controllers. In our implementation, we employ a similar control methodology, utilizing independent PID controllers for position, yaw, and height.

There are several papers on distance estimation from monocular cameras, both for the general camera problem [25] and with the use of drones [26], in this case to determine distances from a person to an object. These papers are focused on distance estimation *relative to the camera* and utilize deep learning-based monocular depth estimation models. Their approaches to depth perception could theoretically be applicable to our work, as our system similarly requires accurate height estimation between the ground and the drone, to enable accurate distance measuring between features on the ground plane. However, the resulting accuracy of the distance estimations is far from sufficient for ground truth applications, where [25] had a mean relative error of 11% and [26] had up to 30 cm error to a target person and an error to an obstacle of up to 60 cm. Other papers focus on determining distances between features in the image. In a paper on single-view metrology [27], Criminisi et al. use the geometry of the vanishing point and a reference length to estimate 3D affine measurements in a single picture. However, this approach typically depends on images of a 3D environment, containing identifiable parallel lines to determine the vanishing point, and reference lengths in each desired direction of measurement.

A byproduct of our system is the collection of large amounts of aerial video data of parking spaces, which we subsequently label with distances to facilitate the develop-

ment of ADAS. There are several large datasets with images for ADAS development, such as Zenseact Open Dataset [11], Waymo Open Dataset [12], Cityscapes Dataset [13], and nuScenes Dataset [14]. However, these are primarily road-focused and do not contain specific parking images. Wang et. al have collected parking area data specifically for ADAS parking functions and automatic parking [15], but it relies solely on car-mounted sensors, such as forward-facing cameras, radar, and surround-view camera systems. There are studies exploring the use of drones for monitoring and managing parking spaces in urban areas, [16] and [17], but none focus on collecting parking images and data with drones for ADAS development. We believe that this approach is underutilized and represents a gap in the research that we intend to address.

## 1.5 System Description

The system consists of two main parts: the drone, equipped with a vision-based controller, which tracks the car while video recording, and the algorithms that process the recorded videos to measure distances. The controller utilizes a roof-mounted ArUco marker [19] as the setpoint to facilitate tracking, as illustrated in Figure 1.1. From the recorded video, the car is detected, as well as all the parking spots. Pixel distances are then used to approximate physical distances between the car and parking spots, illustrated in Figure 1.2.

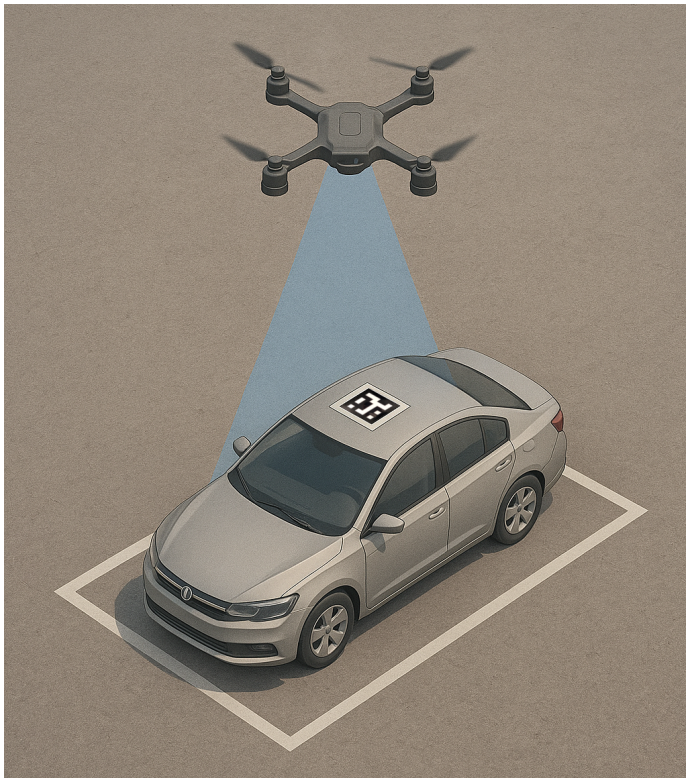


**Figure 1.1:** Illustration of a drone tracking a roof-mounted ArUco marker. (Generated by OpenAI's ChatGPT 4o, private communication, 12 May 2025)

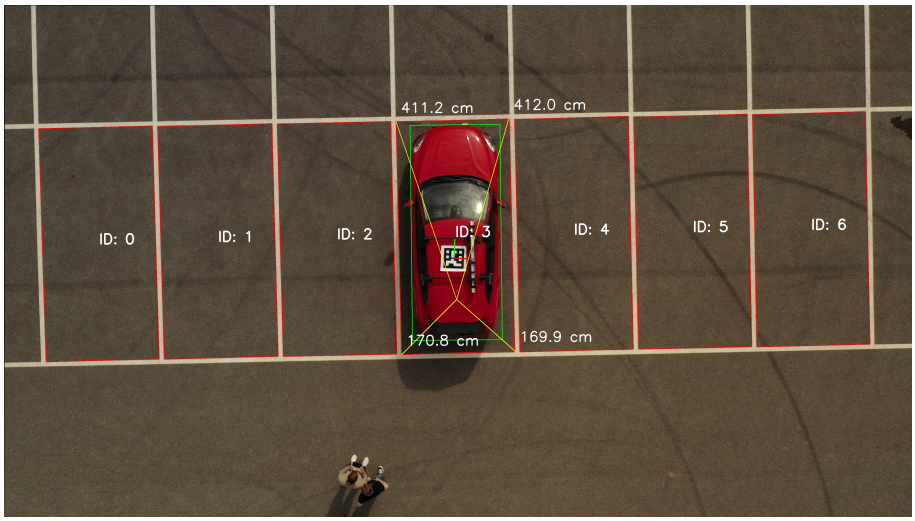



**Figure 1.2:** Image showing distance measurements from the car's origin to one of the detected parking spot's corners

## 1.6 System Hardware

The drone used for this project is, as mentioned earlier, a DJI Mavic 2 Pro equipped with a Hasselblad L1D-20c camera. This camera features a 1-inch sensor, capturing still images and full field of view (FOV) video at 5.5K (5472×3648) [1]. However, all recordings in this project have used the "HQ Mode". When recording videos in HQ Mode, the resolution is limited to 4K. The video mode crops the full sensor readout, utilizing only the center of the full sensor area. This crop effectively narrows the field of view from 77 degrees to 55 degrees, reducing lens distortion and improving image quality.

## 1.7 System Software

The drone control software runs on the DJI Smart Controller, which is the drone's primary remote controller. The controller runs Android, which enables us to customize an Android app to control the drone. The app utilizes DJI's mobile software development kit (SDK) V4, as this is the latest version compatible with the Mavic 2 Pro drone. The SDK gives access to many of the drone's systems. Most important for this thesis are the virtual stick functions, gimbal control, and access to the camera feed.

The virtual stick functions enable the drone to be controlled directly from the app, mimicking the controller sticks, which are high-level controls that build upon the drone's stabilizing mechanisms.

The gimbal control allows us to get the desired BEV camera angle. At the start of the application, the camera is positioned in a downward-facing position, with the image plane parallel to the ground. During operation, the drone's gimbal stabilizers

are active, resulting in stable and smooth video, regardless of the drone's movement.

The camera feed access enables us to track the car-mounted marker. For the marker tracking on the Android platform, we utilized the pre-implemented ArUco detection algorithms available in the Java OpenCV library.

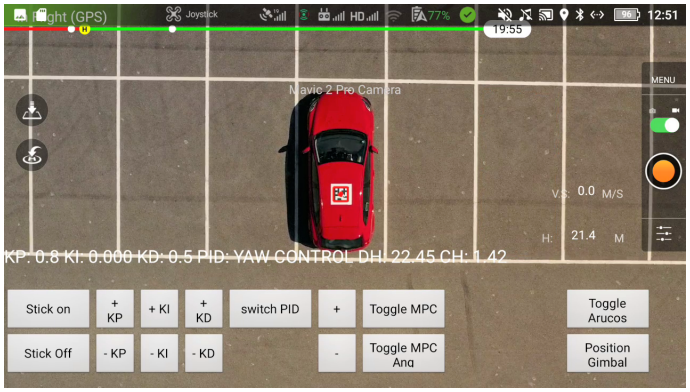



**Figure 1.3:** An image of the Android app in use.

# 2
# Methods

This chapter aims to describe the methods used to construct our system. The system consists of two main components: the controller and the computer vision distance measuring algorithm.

## 2.1 Drone Control

### 2.1.1 System Modeling

A quadrotor is an inherently nonlinear, multivariable, and strongly coupled system [18]. In most cases, this will result in a complex model for the system dynamics. However, for the drone used in this project, the system dynamics does not have to be modeled, since the drone already has low-level stabilizing controllers implemented. The heading, speed, rotation, and height can be controlled directly by sending virtual RC stick commands to the drone through the custom controller app. Similarly to [9], the model can be simplified to treat the virtual stick-to-thrust and stabilizing controllers as a black box.

To simplify calculations in the distance measuring step, it is desirable to maintain a constant height when filming. Under the assumption of constant height, the drone control is effectively reduced to a 2D plane control problem. Therefore, a modified kinematic unicycle model is used, given by the x and y positions, $p_x$ and $p_y$, and the heading angle $\psi$. A quadrotor does not need to head in the same direction it is facing, and therefore, lateral slip is allowed by switching the heading angle to simply the yaw angle. The model is also extended with height $h$ to make sure it remains constant. Hence, the state vector is given by:

$$x = [p_x, p_y, \psi, h]^T \tag{2.1}$$

and by controlling the states directly with the corresponding linear and angular velocities, the control vector is given by:

$$u = [v_{p_x}, v_{p_y}, \omega, v_h]^T \tag{2.2}$$

As the model is linear, the standard state-space equation formulation can be used:

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$

$$y = \mathbf{C}x + \mathbf{D}u$$

where the $A$ matrix is zero, since the system has no inherent dynamics or cross-state dependencies. Since each state is controlled directly and independently, the $B$ matrix becomes the identity matrix. Similarly, as each state variable is directly observable, the $C$ matrix is also the identity matrix. The $D$ matrix is zero since there is no direct feedthrough from inputs to outputs. This leads to the following system equations:

$$\dot{x} = \begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{\psi} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} v_{p_x} \\ v_{p_y} \\ \omega \\ v_h \end{bmatrix} = u \tag{2.3}$$

$$y = \begin{bmatrix} p_x \\ p_y \\ \psi \\ h \end{bmatrix} = x \tag{2.4}$$

The control system uses the drone's camera coordinate system as the primary reference frame. Consequently, the drone's position $\vec{P}$ is the middle point of the camera frame

$$p_x = \frac{f_w}{2} \tag{2.5}$$

$$p_y = \frac{f_h}{2} \tag{2.6}$$

where $p_x$ and $p_y$ are the drone's $x$ and $y$ positions, $f_w$ and $f_h$ are the frame width and frame height. From this perspective, the car's position and orientation are tracked relative to the drone. While the angle and position error are directly used in the control law, the relationship between these coordinate systems can be formally expressed using a homogeneous transformation matrix:

$$R = \begin{bmatrix} \cos\psi & -\sin\psi & c_x \\ \sin\psi & \cos\psi & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

where $\psi$ is the orientation difference between the drone's camera frame and the car's coordinate system, $c_x$ is the x-coordinate of the car's position in the drone's camera frame, and $c_y$ is the y-coordinate of the car's position in the drone's camera frame.

### 2.1.2 Control System

The objective of the control system is to follow the target car as closely as possible. To track the car, ArUco markers, [19], were used to identify it. These markers are designed to be easily detected in video and identified with individual IDs. The

marker is printed on a magnetic car sign, which is put on top of the car at a known position. The marker serves as the set-point target for the control system. From accessing the video feed of the drone, the position of the ArUco marker mounted on the car can be detected, given by the vector $\vec{C}$. From this, a pixel error $\vec{e_p}$ is calculated from

$$\vec{e_p} = (p_x - c_x, p_y - c_y), \tag{2.8}$$

where $\vec{P} = [p_x, p_y]^T$ is the drone's pixel positions and $\vec{C} = [c_x, c_y]^T$ is the car's pixel positions.
From the relative orientation of the ArUco marker in the frame, the rotation error is determined as

$$e_\psi = \tanh \psi \tag{2.9}$$

The altitude error is determined from the desired height $h_d$ and the estimated height $h$, obtained from the *height function*, found using the method in Appendix A.1. The error is determined as

$$e_h = h_d - h \tag{2.10}$$

Based on these four errors, decoupled PID controllers are employed to control each axis, allowing for separate tuning. As previously mentioned, high-level virtual stick RC commands are sent through the custom controller app, enabling us to control the states directly.

While each controlled degree of freedom requires its own PID controller, the underlying mathematical formulation remains consistent across all four implementations. To avoid redundancy, the general control law is presented as:

$$u_c = K_p e + K_i \int e dt + K_d \frac{de}{dt} \tag{2.11}$$

where $e$ is the control error for each variable, and $K_p$, $K_i$ and $K_d$ are the proportional, integral, and derivative gains, respectively. The control system is visualized in Figure 2.1.

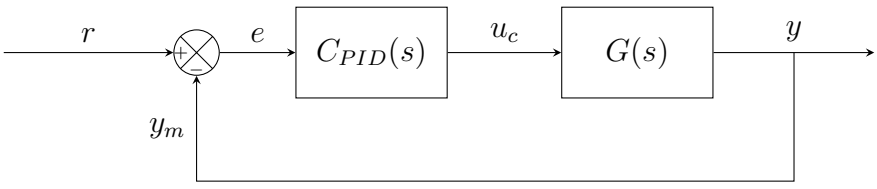



**Figure 2.1:** Block diagram of the PID controller $C_{PID}(s)$ and system *G(s)*.

## 2.2 Vision Algorithms

This section describes all the algorithms needed to process the video footage once collected. It follows the structure of Figure 2.2. It begins with image pre-processing, where the frame is filtered to leave only the parking lines. The output from the pre-processing is called the threshold frame, on which the parking spot detections are

done. This step contains two parts: the detection of parking spots and the tracking of said parking spots. The final step is the distance estimation, where the distances from the car to parking spots are calculated.

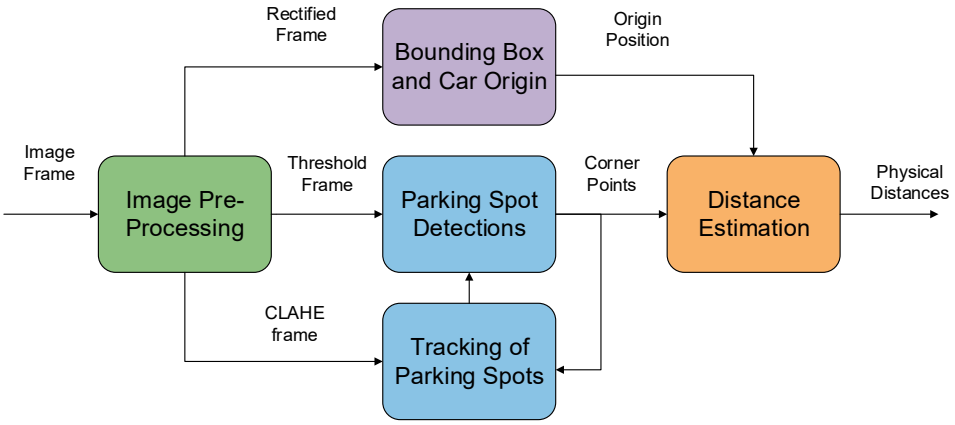



**Figure 2.2:** Overview diagram of how the vision algorithms work together.

### 2.2.1 Image Pre-processing

The image pre-processing consists of five main steps, as illustrated in Figure 2.3. These five steps are *frame undistortion*, *grayscale conversion*, *brightness equalization*, *CLAHE* and *Thresholding*.

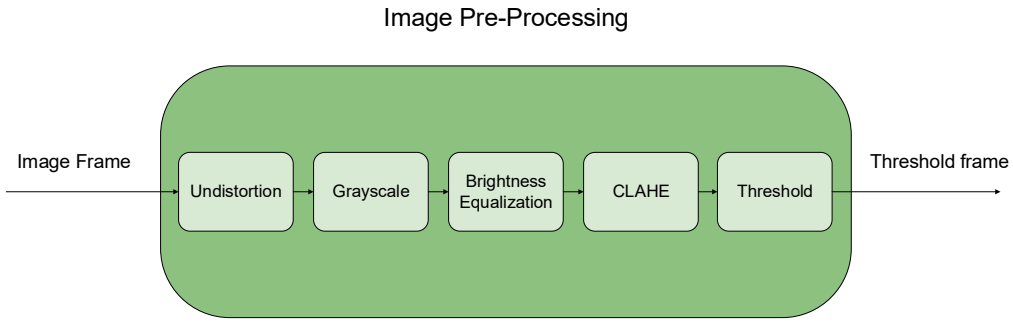



**Figure 2.3:** Overview diagram showing the Pre-Processing steps.

#### 2.2.1.1 Calibration and Frame Undistortion

Camera images always contain lens distortions that create non-uniform scaling across the frame, which in turn affects measurement accuracy. The most common types of distortion are barrel distortion (present in our case) and pincushion distortion, illustrated in Figure 2.4.

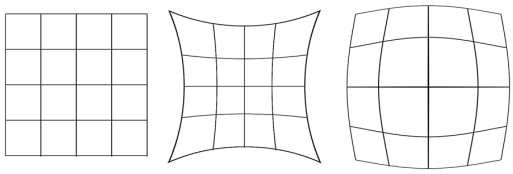


**Figure 2.4:** No distortion, pincushion distortion, and barrel distortion. [24]

In the *frame undistortion* step, the camera's intrinsic matrix and distortion coefficients are used to remove lens distortion, thereby establishing a consistent transformation that enables accurate mapping of real-world 3D points to their corresponding 2D image coordinates. The camera matrix and distortion coefficients were determined using Zhang's method [23], as described in the OpenCV calibration guide [22].

The camera matrix $K$ is typically represented as:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

where $f_x, f_y$ are the focal lengths in pixels along the $x$ and $y$ axes, $c_x, c_y$ are the coordinates of the principal point (optical center).
The distortion vector $D$ containing radial and tangential distortion coefficients is typically represented as:

$$D = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{2.13}$$

where $k_1, k_2, k_3$ are the radial distortion coefficients, $p_1, p_2$ are the tangential distortion coefficients.

#### 2.2.1.2 Grayscale Conversion and Brightness Equalization

When collecting real-world parking data, the lighting conditions will not always be the same. Therefore, the brightness of the frame is adjusted to make it standardized for every lighting scenario. This is achieved by converting the image to grayscale, calculating the mean pixel intensity value, determining the difference from the desired average, and adjusting each pixel.

$$P_{avg} = 1/N \sum_{n=1}^{N} (P_n) \tag{2.14}$$

$$x = P_{desired} - P_{avg} \tag{2.15}$$

$$NewP_n = Max(0, Min(255, P_n + x)) \tag{2.16}$$

Pixel intensity values range from 0 (black) to 255 (white), with various shades of gray in between. The adjustment process adds the same offset value, $x$, to all pixels, thereby shifting the overall brightness toward the desired mean. However, when pixels are adjusted, some may exceed the valid range. The Max and Min functions in the equation ensure that adjusted values remain within the valid range by clamping out-of-range values to their respective limits.

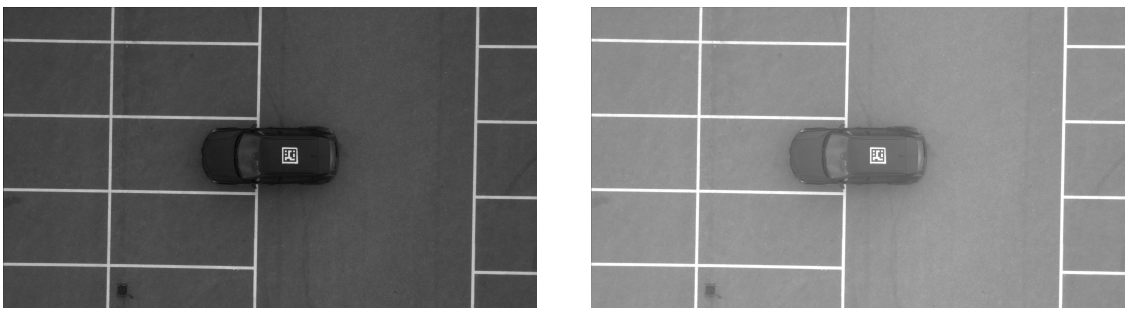


**Figure 2.5:** Gray scale image to the left, image after brightness adjustment on the right.

#### 2.2.1.3 CLAHE and Threshold

The next step is to apply Contrast Limited Adaptive Histogram Equalization (CLAHE). It works by splitting the image into tiles and equalizing them separately to achieve a more even lighting distribution. This enhances the bright spots and makes the dark spots darker, as shown in Figure 2.6.
After the CLAHE, a Gaussian blur is applied to the image. This is done to smooth out the image, reducing very bright pixels in the frame. Finally, a binary threshold is applied, where all pixels above the threshold value are masked as white and those below are masked as black.

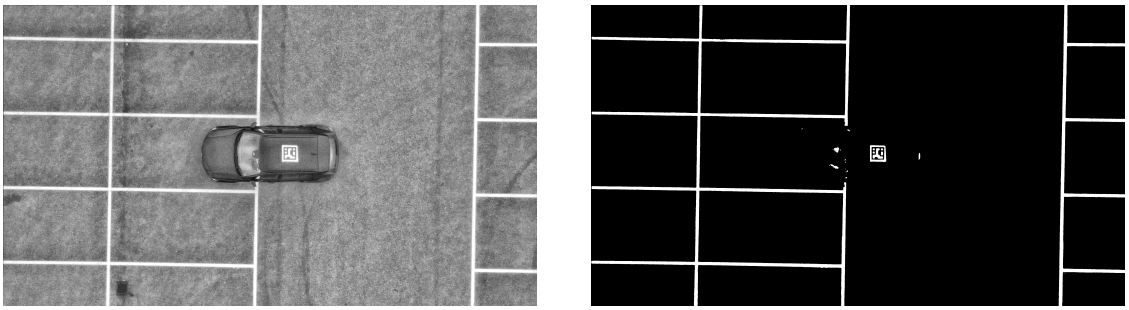


**Figure 2.6:** CLAHE image to the left, Threshold image to the right.

### 2.2.2 Parking Spot Detection

Once the Pre-Processing is done, the next step is to detect parking spots. An overview of the detection and tracking algorithms' structure can be seen in Figure 2.7. The parking spot detection algorithm works by identifying the contours of the parking spot within the threshold frame.

#### 2.2.2.1 Long Lines

Depending on the quality of the parking spot, there may be gaps in the parking lines in the threshold frame. To combat this, lines are detected in the threshold frame using OpenCV's Fast Line Detector. The lines will be detected on each side of the gap, and by joining the lines together, a longer line will be created that bridges the gap. The longer line is created by checking if any of the detected lines are co-linear, and if these lines are within a set distance of each other. If they are, join them to make a single, longer line. This effectively bridges any gaps in a parking spot.

#### 2.2.2.2 Contour Detection

In the threshold frame with added lines, contours are detected. To do this, OpenCV's *findContours* function is used. The function finds all contours in the threshold frame, essentially highlighting all parking lines.

Then, all the complete contours are looped through, and the shape of each contour is approximated. If the shape is a rectangle, its dimensions are checked to

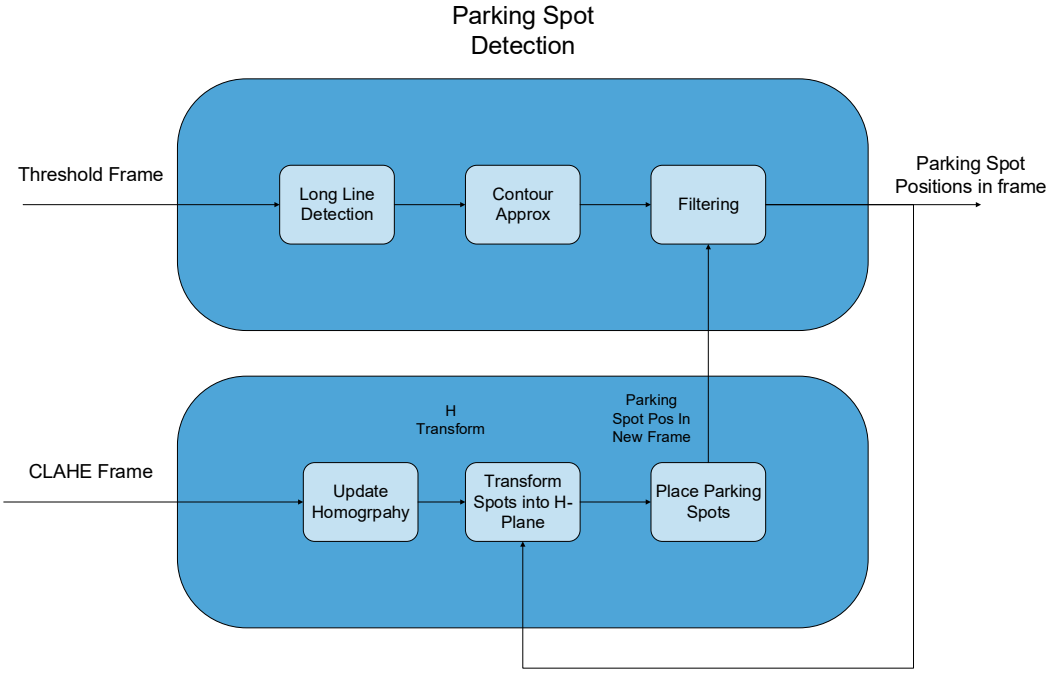



**Figure 2.7:** Overview diagram of the parking spot detection.

determine if it is a possible candidate for a parking spot. The standard size for a parking spot in Sweden is approximately 2.5 m x 5 m; this information is used to further filter the rectangular contours. If the ratio between the sides is approximately 2:1 and the total calculated area matches the standard area of 12.5 $m^2$, a parking spot has been identified. The corner points of the parking spot are stored in a parking spot object to track each spot between frames.

#### 2.2.2.3 Parking Spot Tracking

Two methods have been developed for tracking parking spots between frames: a simple distance check method and a more complex homography-based method. The

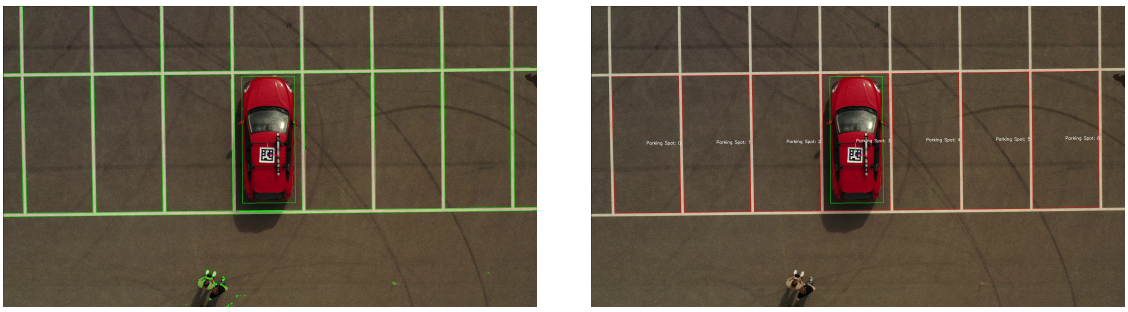


**Figure 2.8:** Frame with contour detection and a frame with detected parking spots from the contour frame

distance check method compares the detected spots with those in the previous frame. The center points of all detected parking spots are calculated. If the center is within a certain margin of a spot in the previous frame, it's assumed that the same spot is found. With this functionality, unique IDs are assigned to all spots. This approach performs reliably provided that parking spots are fully visible. However, if a spot becomes obscured or out of frame, the method will not be able to detect the parking spot. This is not ideal, as the parking spot will no longer be detected if the car drives over a part of it.

For scenarios like these, the homography-based method is employed. As the drone is filming from above, it can be assumed that the background (or ground in this case) will remain unchanged while the drone is flying. This assumption reduces the ground to a 2D plane, and by tracking the background, it should be possible to determine the positions of the parking spots even when the detection is lost. To track the ground, Scale Invariant Feature Transform (SIFT), [30], can be used. SIFT finds a list of points in the frame that are suitable for tracking. These tracking points are then compared to the new frame and matched up using a BF-Matcher [32]. From the matched points, a homography matrix is created, which transforms the camera frame coordinates into the homography plane. As the drone moves, the homography matrix is updated to keep the relationship between the H-plane and the camera frame. This lets us "remember" the positions of parking spots even when they are not detected, or outside of the frame. Consequently, this allows us to progressively map the parking lot area.

### 2.2.3 Bounding Box and Car Origin

Determining a well-fitted bounding box around the car and establishing its origin point is a critical step, as this origin serves as the reference for all subsequent distance measurements. Additionally, measuring distances from the vehicle's edges enables a more accurate assessment of parking alignment and nearby obstacles. The bounding box is manually defined to precisely enclose the car and is placed with fixed offsets relative to the roof-mounted ArUco marker. The rotation of the bounding box is continuously updated to follow the rotation of the ArUco marker, and thus the car's orientation. The bounding box is also used to mask the car out of frame, so it does not interfere with the SIFT tracker and the parking spot detection algorithms.

The origin of the car is defined as the midpoint of the rear axle, projected onto the ground, visualized in Figure 2.9.

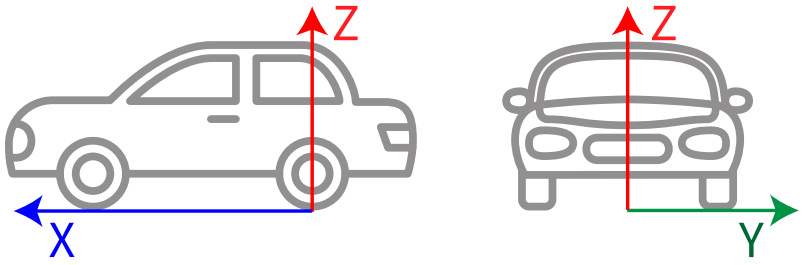



**Figure 2.9:** Vehicle coordinate system. [33]. Reprinted with permission.

This point cannot be seen from the drone, but it is found in the frame by comparing the bounding box dimensions with the real-world dimensions of the car. The relevant dimensions for the longitudinal rear axle point are car length and rear overhang. The lateral rear axle center is always positioned at the vehicle's lateral midpoint, making the car's width irrelevant. Notably, the origin's $x$ and $y$ positions can be determined using the following equations.

$$O_x = \frac{w_{bb}}{2} \tag{2.17}$$

$$O_y = l_{bb} \cdot \frac{c_{overhang}}{c_{lenght}} \tag{2.18}$$

where $O_x$ and $O_y$ are origin's pixel positions, $w_{bb}$ and $l_{bb}$ are the bounding box width and length in pixels, and $c_{overhang}$ and $c_{lenght}$ are the car's overhang and length dimensions.

### 2.2.4 Distance Estimation

Figure 2.11 provides an overview of the distance estimation step's structure.

In the distance estimation step, the drone's height is estimated from the ArUco pixel size in the frame. A pixel-to-centimeter constant is calculated using the estimated drone height, ArUco pixel size, and the known physical size of the ArUco marker. Finally, the system estimates the distances between the car's origin and parking spot corners by converting pixel measurements using the pixel-to-cm function. The estimated distances $\mathrm{d}_1 - \mathrm{d}_4$ for the closest parking spot are visualized in Figure 2.10. While Figure 2.10 only shows the closest spot, the distances are estimated for all parking spots in the frame.

#### 2.2.4.1 Height Measurement and Pixel-to-centimeter Conversion

During testing, the drone's onboard altitude estimation was found to be inaccurate, with deviations of up to ±3 meters when hovering at an indicated altitude of 22 meters. This deviation stems from the main altitude sensors, the internal barometer and GPS, which are inherently susceptible to environmental factors such as wind and temperature.

To obtain more reliable altitude estimates, a function is derived that numerically

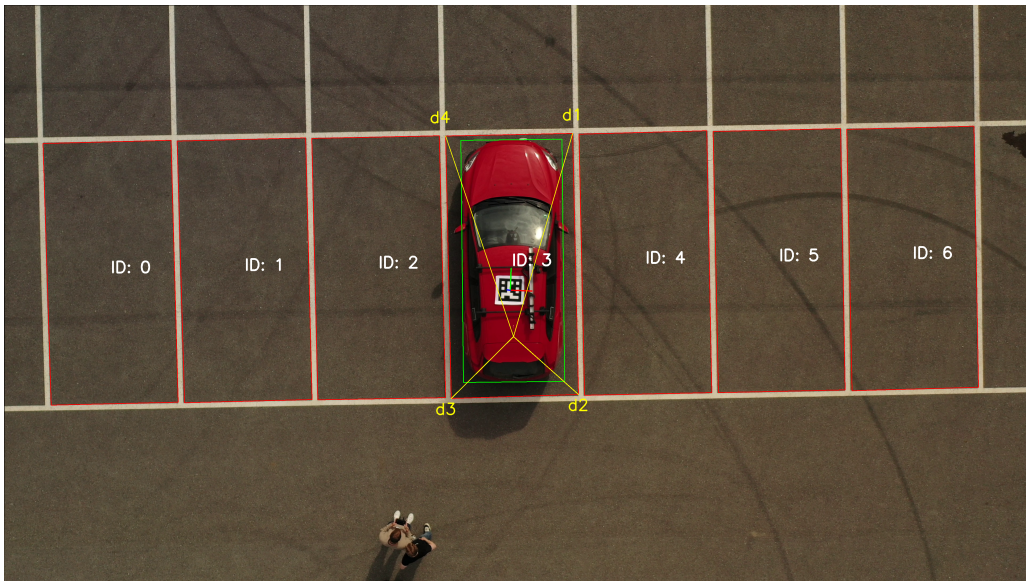



**Figure 2.10:** Visualization of measured distances for the closest parking spot

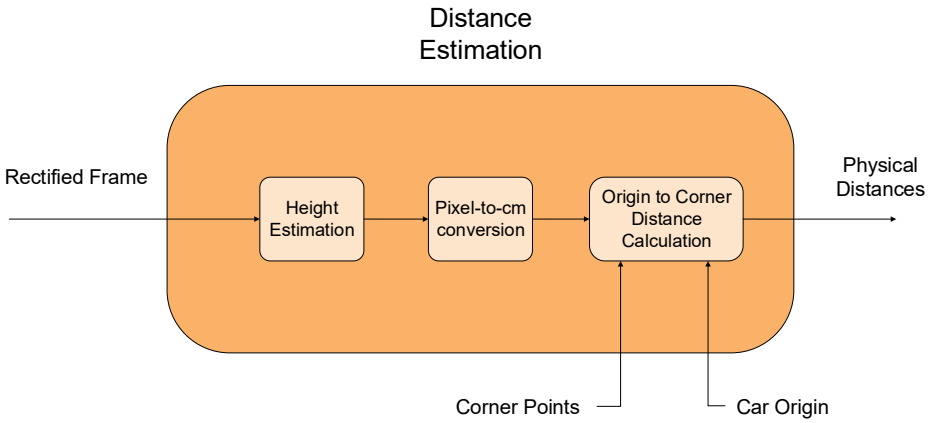



**Figure 2.11:** Overview diagram of the Distance Estimation section.

estimates the drone's height based on the observed pixel circumference of the ArUco marker in the frame. By comparing the marker's known real-world circumference with its observed pixel circumference, a height-dependent conversion factor can be obtained. The test procedure to find the function can be found in Appendix A.1.

The function estimates the distance to the roof-mounted ArUco marker. Since the car's height is known, the ArUco marker's pixel circumference on the ground plane can be estimated by inverting the function. Subsequently, the pixel-to-centimeter conversion can be estimated using the physical perimeter of the ArUco markers. The procedure is shown in Algorithm 1.

**Algorithm 1** Calculation of pixels per cm

**Input:** arucoCornerPoints, carHeight, arucoPerimiterCm
**Output:** pixelPerCm

1: $arucoPerimiterPx \leftarrow$ **cv2.arcLength**$(arucoCornerPoints)$
2: $heightFromGround \leftarrow$ **getHeightFromPerimiter**$(arucoPerimiterPx) + carHeight$
3: $perimiterGround \leftarrow$ **getPerimiterFromHeight**$(heightFromGround)$
4: $pixelPerCm \leftarrow \frac{perimiterGround}{arucoPerimiterCm}$
5: **return** $pixelPerCm$

#### 2.2.4.2 Distance Estimation from Origin to Parking Spot Corners

Using the car's origin as the reference, the pixel distances to the parking spot corners are calculated using the Euclidean norm.

$$d_{px}(\vec{O}, \vec{P}_{corner}) = \sqrt{(O_x - P_{corner_x})^2 + (O_y - P_{corner_y})^2} \tag{2.19}$$

Using the $pixelPerCm$ conversion factor, the estimated physical distance can be computed as

$$d_{cm} = \frac{d_{px}}{pixelPerCm} \tag{2.20}$$

## 2.3 Verification

This section describes the verification test procedures for both parts of the system.

### 2.3.1 Distance Algorithms Verification

During the development phase, several continuous tests were conducted to verify the system's different components. While several different test setups were explored, the final iteration was deemed the most accurate method for static measurements. The test was conducted as follows:

- Place the car in a desired position, in or outside a chosen parking spot.
- From the car's origin on the body, i.e., the center point of the rear axle, use a string suspended plumb bob to find the exact position of the car's origin on the ground. Mark this point.
- Gather drone video data of the car in its current position, together with the parking spot.
- Move the car such that it no longer covers the marked origin or the parking spot.
- Measure and note the exact distances from the marked origin to the inner corners of the parking spot. This is now the ground truth distance.

- Place the ArUco marker on the ground, and center the marker over the marked origin.
- Gather video data of the parking spot, now with the ArUco marker on the ground, with its center acting as the origin.
- After collecting the videos, run the vision algorithms to find the origin of the car as well as the distance from the origin to the corners. Also, measure the distances using the video without the car, from the marked origin on the ground.
- Compare with your ground truth data. Evaluate the accuracy of the distance measuring algorithms, as well as whether the distances are the same for the estimated origin in the first video and the marked origin on the ground in the second.

### 2.3.2 Control System Verification

The control system was evaluated both qualitatively and quantitatively. The qualitative part included verifying that the drone controller was capable of tracking and following the car during the desired maneuvers and speeds. This included both straight driving and up to full-lock turns, with speeds slightly exceeding normal parking speed, as well as accelerations and decelerations during these maneuvers. The following tests were conducted:

1. Set the horizontal error to the maximum by translating the Aruco marker to the horizontal edge of the frame, with no error for the other variables.
2. Set the vertical error to the maximum by translating the Aruco marker to the vertical edge of the frame, with no error for the other variables.
3. Set the rotational error to the maximum by rotating the marker 180 degrees. Place the marker in the middle of the frame to set the positional error to zero.
4. Perform a joint test, with the horizontal and rotational error set to the maximum. Set the vertical error to a moderate value to avoid the marker rotating out of frame.

Quantitatively, the following metrics were observed during step response tests:

- Rise time $t_r$
- Peak overshoot $M_p$ and percent overshoot $PO$
- Settling time $t_s$

The metrics were observed for each axis during both separate and joint step response tests.

# 3
# Results

## 3.1 Distance Measurements

For all the plots referring to measurements, the ordering of parking spots will follow Figure 3.1. The parking spot IDs start on the left of the frame with ID 0 and increase up to ID 6 on the right of the frame. The origin of the car from which the measurements are taken is in the middle of the frame, which means that parking spot 3 is the closest.

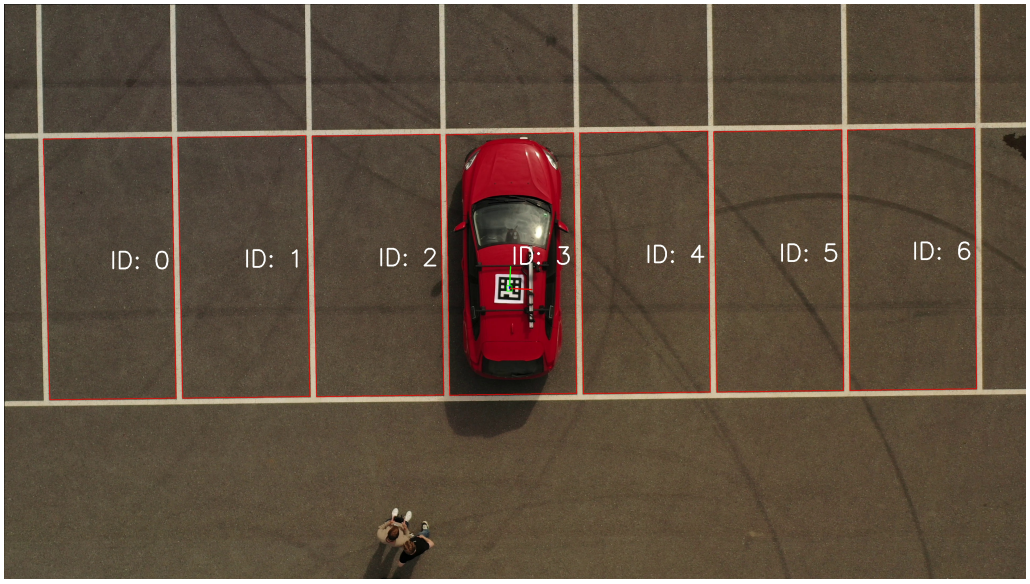



**Figure 3.1:** Figure showing the IDs of parking spots used in the measurement results.

### 3.1.1 Numerical Height Function

The numerical function describing the ArUco marker's circumference $m_{px}$ depending on the height $h$ was determined using the method in Appendix A.1. This resulted in the following function

$$m_{px} = \frac{6978.9}{h^{1.012}} \tag{3.1}$$

which nearly perfectly describes a camera-dependent constant divided by the height, confirming the expected theoretical relationship where the apparent size of the

marker in the frame is inversely proportional to distance. The model achieved an R-squared value of 0.992, indicating that our measurements align very well with the model found, as shown in Figure 3.2.

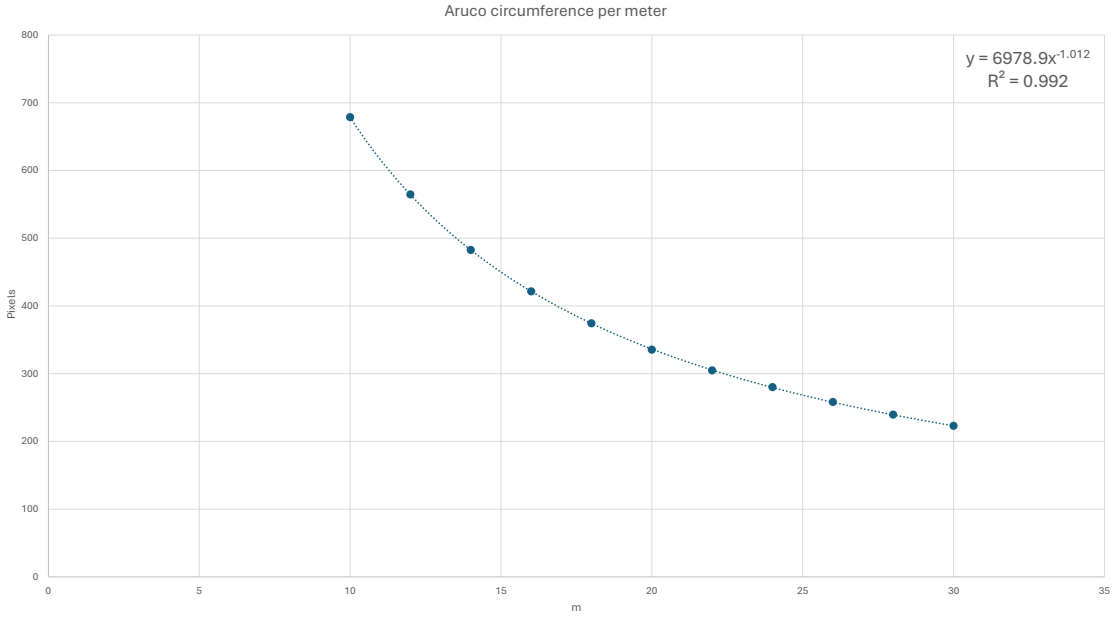



**Figure 3.2:** The numerically derived height function, based on the ArUco pixel size in frame

## 3.2 Ground Plane Distance Estimation

In this section, the accuracy of the distance estimation algorithms are presented. The results shown are from the algorithms when applied *without* a target car, with the ArUco placed on the ground plane. Hence, any uncertainty in the car's measured physical height and the origin determination algorithm are excluded. This is followed by the accuracy results of the complete system, with the ArUco mounted on the target car, and with the car's determined origin as the reference point.

In Figure 3.3, the signed measurement error to the corners of parking spots is shown. The different colors indicate which parking spot each corner belongs to. In the figure, clusters of points at similar distances can be observed, as there were parking spots on both sides of the target car, whose measurements almost coincided. A linear trend is observed, with parking spots further away having a larger measurement error. The maximum observed error at one standard deviation is 27.13 cm, which is observed for the corners furthest away in parking spots 0 and 6. The signed errors reveal a systematic error where distances are overestimated in direct proportion to the measured distance. A linear compensation function was derived from these measurements to address the systematic error. After applying this compensation, Figure 3.4 demonstrates significantly improved accuracy in distance estimation, with

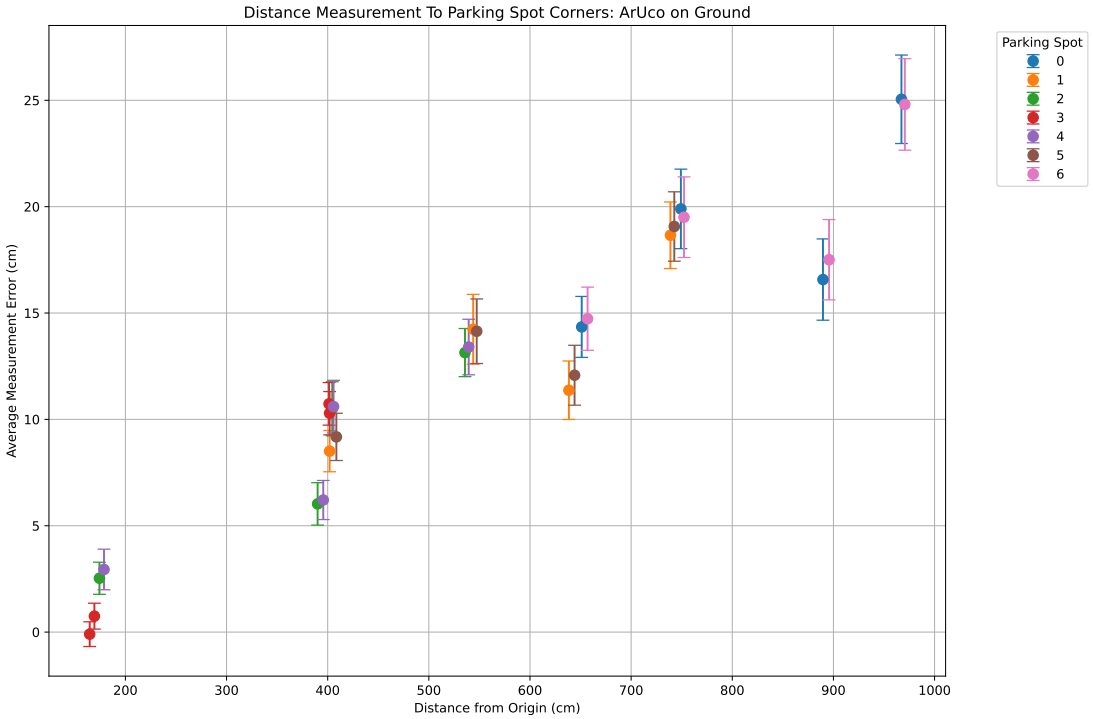



**Figure 3.3:** Plot showing the signed mean measuring error with one standard deviation error bars, between parking spots corners and an ArUco marker positioned on the ground plane.

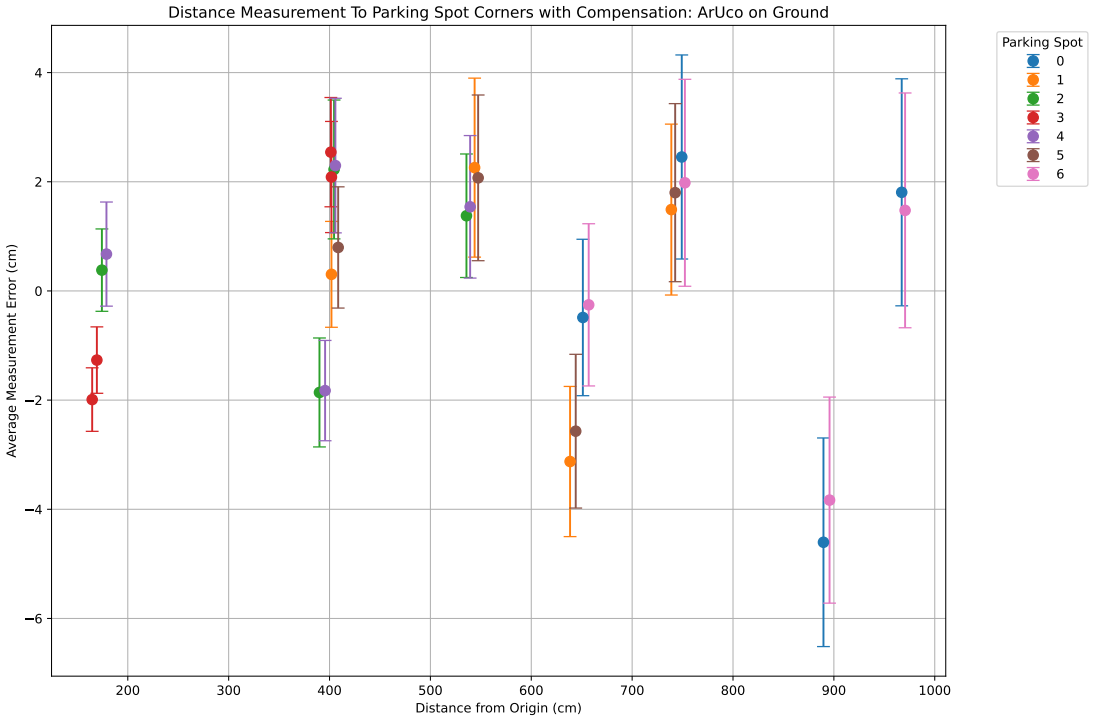



**Figure 3.4:** Plot showing difference to ground truth from measurements from an ArUco marker on the ground plane with systematic error compensation.

the maximum observed error at one standard deviation reduced to 6.46 cm across all measurements and to 2.56 cm for distances less than 2 meters.

### 3.2.1 Distance Estimation Using Origin

In Figure 3.5, signed measurement errors are shown, from video taken with the marker on the roof of the target car. In the graph, a trend is observed where both measurement errors and standard deviations increase proportionally with the distance from the origin. The maximum observed error at one standard deviation is 7.95 cm across all measurements and 4.64 cm for distances less than 2 meters.
It can be noticed that the points of similar distances deviate more than what the individual standard deviations describe. To combat this, similar distances have been binned into an average point. In Figure 3.6, points within a 20 cm span are binned together into the same point. The standard deviation is then calculated for the entire bin, rather than for each individual point. Not surprisingly, the error and standard deviation increase with distance from the origin, as seen earlier. This trend is quite strong, with an $R^2$ value of 0.816.

In Figure 3.6, the absolute errors for different distance ranges are shown. A relative error of 1.00% is observed at distances 0-200 cm and a relative error of 0.54% is observed for distances in the range 200-1000 cm.

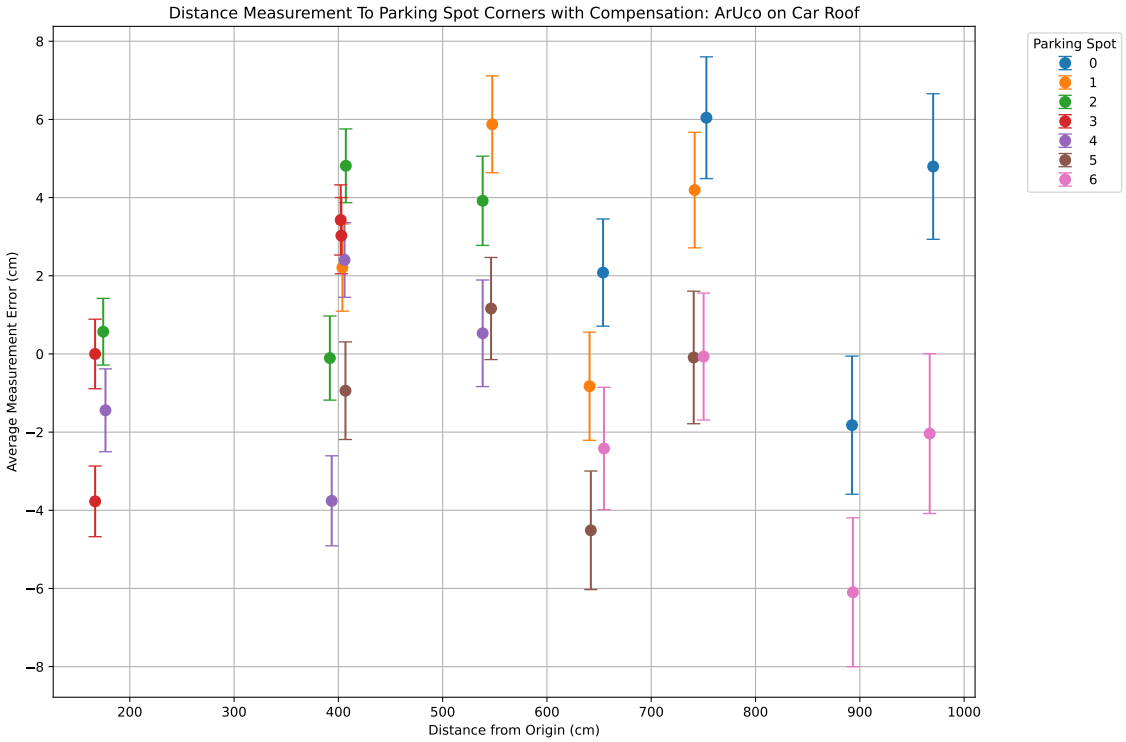



**Figure 3.5:** Plot showing difference to ground truth for measurements to parking spot corners from an ArUco marker on the roof of a car with systematic error compensation.

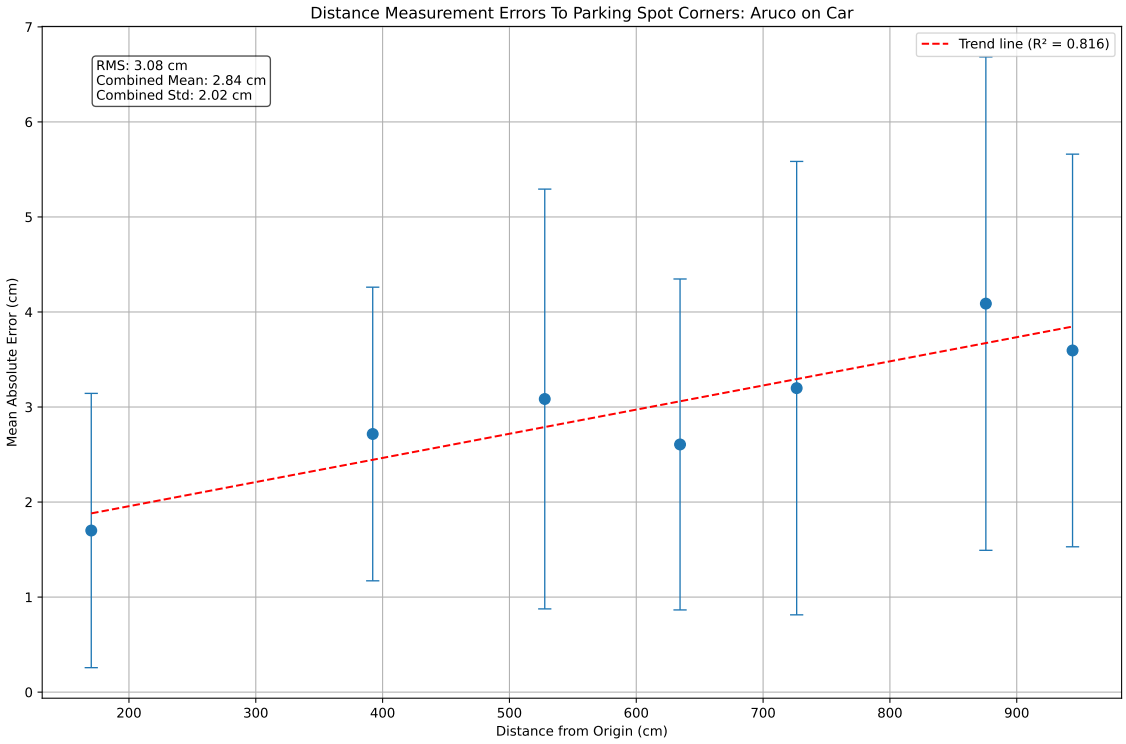



**Figure 3.6:** Plot showing absolute error for measurements from an ArUco marker on a car to parking spots, with a systematic compensation.

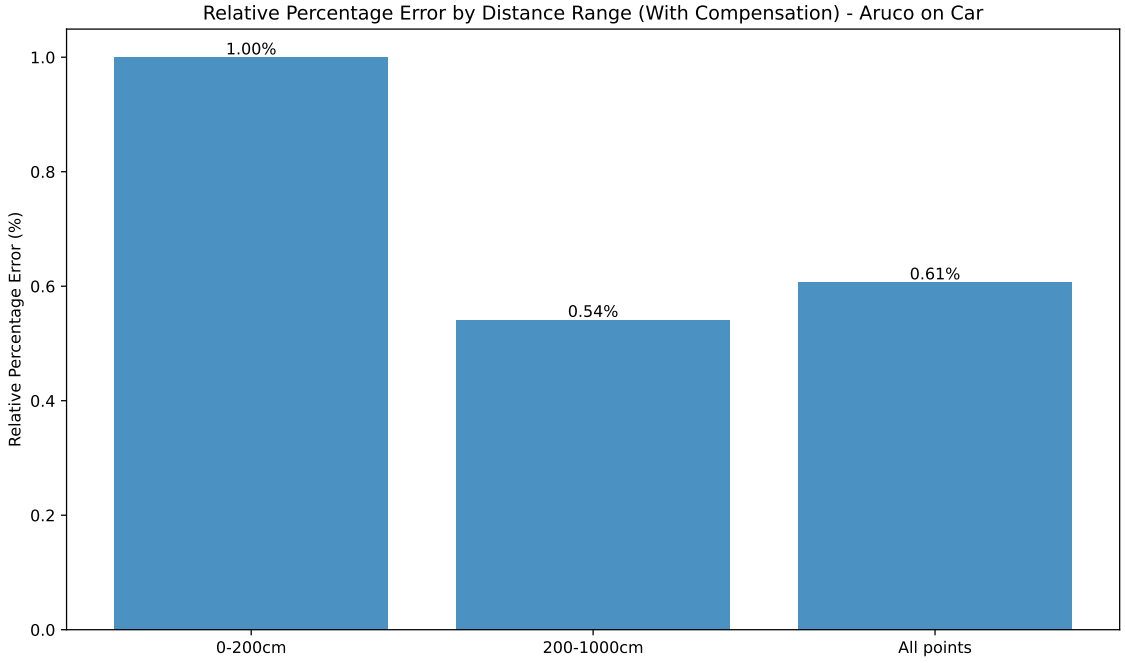



**Figure 3.7:** Plot comparing absolute relative error for distances at different distance ranges.

The system's accuracy is further evaluated and presented in Table 3.1, where the accuracies for both short and long distances are also separated.

**Table 3.1:** Distance Estimation Results – The table presents accuracy assessment results using mean absolute error (MAE), MAE plus one standard deviation ($\sigma$), root mean square (RMS) error, and absolute relative error (ARE) across different distances. Lower values indicate better performance for all metrics.

| **Measured Distance** | **MAE** (cm) | **MAE** + **1**$\sigma$ (cm) | **RMS** (cm) | **ARE** (%) |
|---|---|---|---|---|
| 0 – 2 m | 1.7 | 3.14 | 2.23 | 1.00 |
| 2 – 10 m | 3.03 | 5.07 | 3.65 | 0.54 |
| 0 – 10 m | 2.84 | 4.86 | 3.48 | 0.61 |

## 3.3 Drone Controller Performance

### 3.3.1 Qualitative Performance

The decoupled PID control strategy, combined with DJI's internal stabilizing controllers, successfully follows the target car while recording stable video. The control strategy demonstrated robust tracking of the car up to 18 km/h longitudinally and up to 14 km/h during full-lock steering maneuvers, which exceeds typical requirements for parking maneuvers. The gimbal-mounted camera reduces oscillations and yields constant, high-quality video data; however, the gimbal also compromises the yaw controller's performance, as observed in the following section.

### 3.3.2 Quantitative Performance

In this section, the quantitative performances of the PID controllers are presented. The rise time, settling time, and overshoot during step responses can be observed for the $p_x$-controller, $p_y$-controller, and $\psi$-controller, as well as a joint step response. For each step response, the error graphs for each controlled variable are presented. The height controller is evaluated solely by analyzing the response during the other step response tests, since its target is to maintain a constant altitude, which is only relevant when done in conjunction with other control actions. The height determination of the drone is not as accurate as in the post-processing due to down-sampling of the image frame. The controller also utilizes a dead band to reduce oscillations. Hence, the initial values are not always the target altitude, and the final values do not always coincide fully.

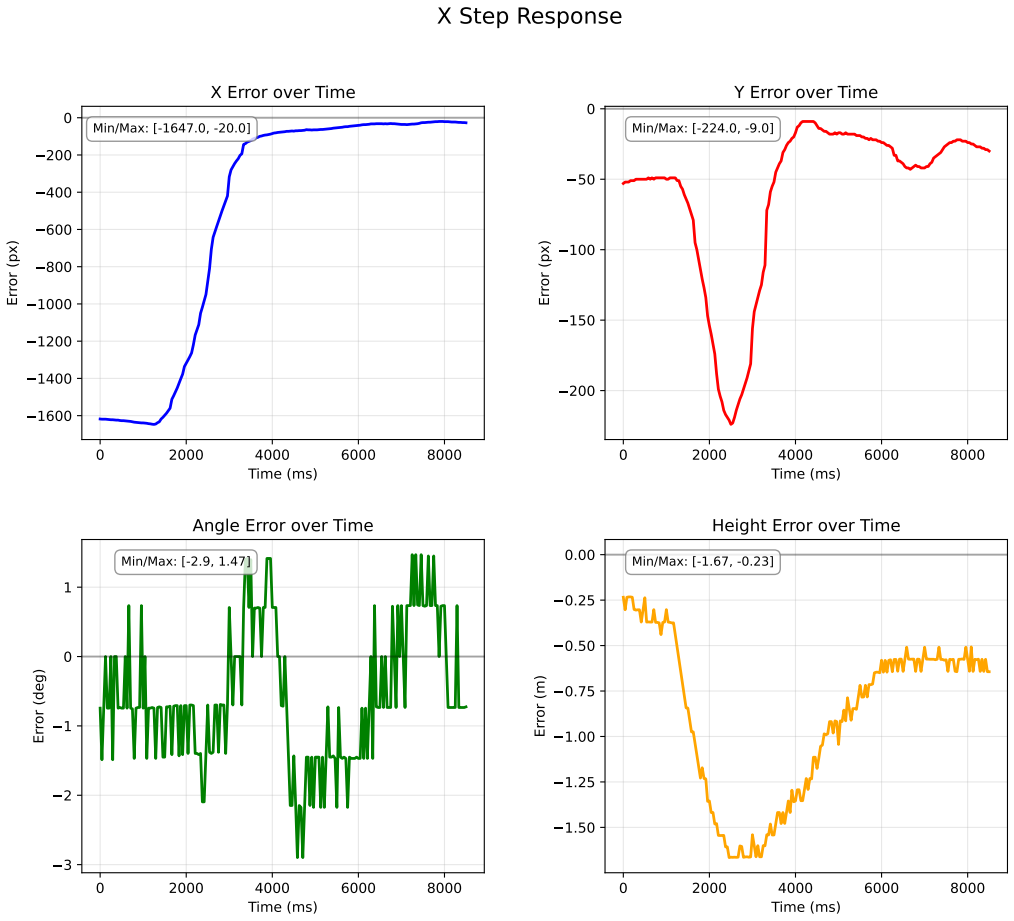



**Figure 3.8:** $p_x, p_y, \psi$, and height error graphs during the X step response test

**Table 3.2:** Control System Results - Step response characteristics for each controlled variable, with rise time $t_r$, settling time $t_s$, peak overshoot $M_p$, and percent overshoot PO

| **Controlled Variable** | $t_r$ **(s)** | $t_s$ **(s)** | $M_p$ (Pixels, Degrees) | PO (%) |
|---|---|---|---|---|
| **Separated step response** | | | | |
| $p_x$ | 1.58 | 2.83 | 0 | 0 |
| $p_y$ | 0.90 | 1.31 | 38 | 4.36 |
| $\psi$ | 1.79 | 4.79 | 7.64 | 4.29 |
| **Joint step response** | | | | |
| $p_x$ | 0.86 | 2.58 | 119 | 7.60 |
| $p_y$ | 0.82 | 4.98 | 196 | 20.21 |
| $\psi$ | 1.94 | 5.65 | 2.07 | 1.15 |

In all of the separate step response tests, cross-coupling effects between the controlled variables were observed. For instance, this can be observed during the x-step response seen in Figure 3.8. A step input in the x-axis introduced a deviation in the y-axis error. The same behavior was observed during the y-axis step response in

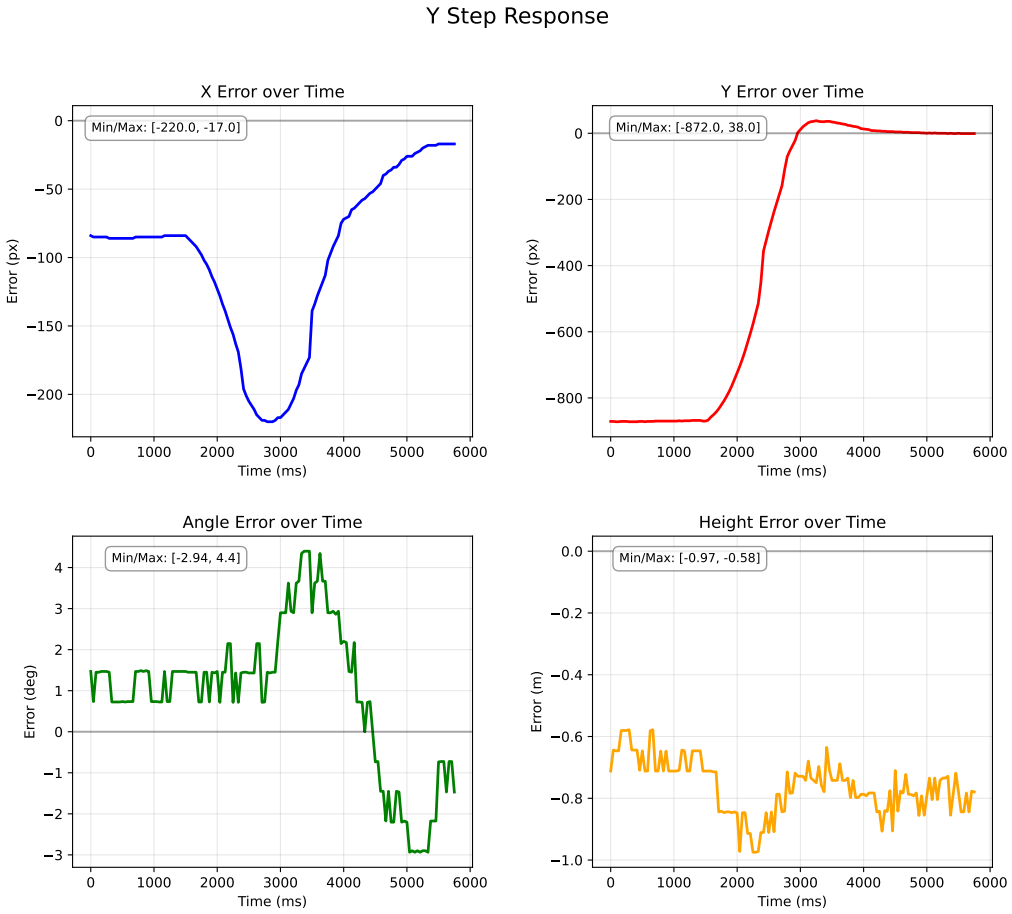



**Figure 3.9:** $p_x, p_y, \psi$, and height error graphs during the Y step response test

Figure 3.9. In this test, the angle error also exhibits a noticeable deviation. The altitude error also deviates in both magnitude and direction for both step response tests.

In Figure 3.8, the step response for the $p_x$ controller exhibits the characteristics of an over-damped system, with no overshoot and a settling time of 2.83 seconds. The step response for the $p_y$-controller, seen in Figure 3.9, shows both a shorter rise time and settling time compared to the $p_y$-controller. However, it does have a slight overshoot of 38 pixels. The rise time is 43% shorter, which is attributed to the fact that the initial pixel error is 38% smaller. This is due to the frame size, which is shorter in the Y-direction. However, the settling time is significantly shorter. As the $p_x$ and $p_y$ controllers are tuned identically, the discrepancy between them is most likely due to cross-coupling effects between the controlled variables.

The $\psi$-controller shows a rise time of 1.79 seconds and a settling time of 4.79 seconds. It also shows a small undershoot before a slight overshoot of 7.64 degrees. This is mainly because of the gimbal smoothing controllers, that does not allow rapid movements of the camera frame, even if the drone rotates.

During the joint test, described in Section 2.3.2, a significant decrease in rise time

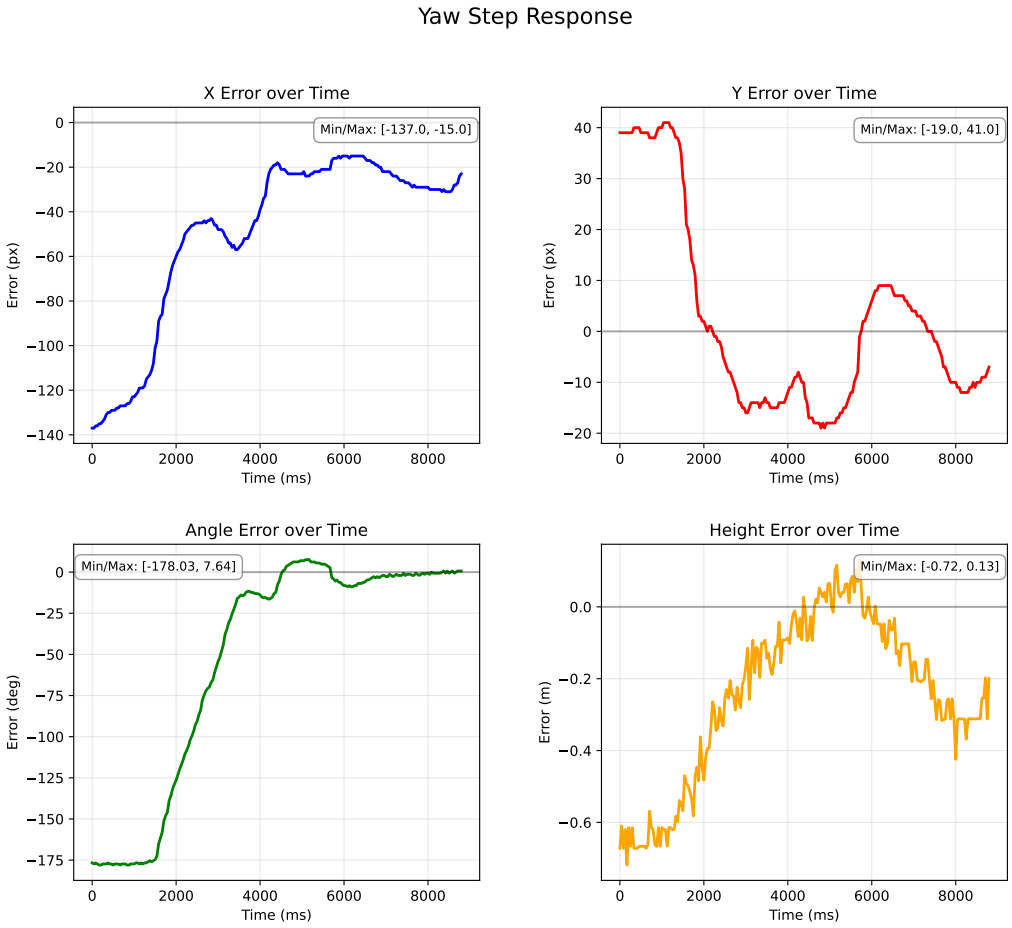



**Figure 3.10:** $p_x, p_y, \psi$, and height error graphs during the yaw step response test

for the $p_x$-controller is observed, along with a notable increase in the $p_y$-controller's settling time. The $p_y$ rise time is similar to the separate test. However, the $p_y$ rise time is calculated from after the y-error dip. The settling time is still calculated from the time of the step input.

The results can be partly attributed to the test's configuration. As can be seen in the y-error graph in Figure 3.11, the initial y-error is small but quickly rises. This is due to the rotation of the drone, which shifts the positional error between x and y, proportional to the drone's rotation. Hence, at a 90-degree rotation, the remaining error in x has shifted to y, and vice versa, as shown by the dip in the y-error graph. Subsequently, the position controller's performance cannot be separated from each other in this test and has to be observed jointly.

For both the $p_x$ and $p_y$ controllers, a larger overshoot is observed compared to the separate step response test, with 119 pixels for $p_x$ and 196 pixels for $p_y$. The percent overshoot of the $p_y$-controller reaches 20.21% while the $p_x$-controller reaches 7.60% only. However, the frame size and consequently the maximum possible error in the y-direction are smaller, which is a contributing factor to this discrepancy.

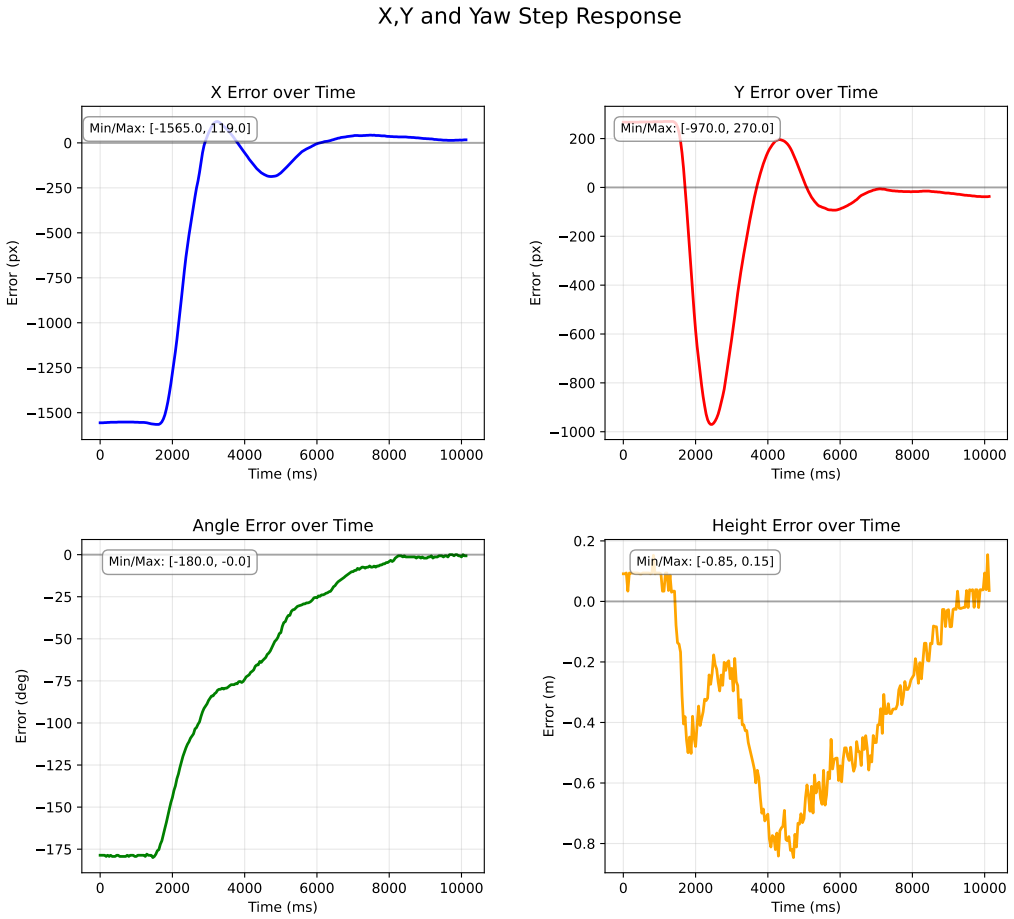



**Figure 3.11:** $p_x, p_y, \psi$, and height error graphs during the joint step response test

After observing the results and the joint test itself, it is evident that the positional controllers' rise time benefited from the contribution of the rotational controller. When the marker was rotated, it was also translated towards the middle of the frame. However, this resulted in a higher overshoot, which increased the settling time for the remaining positional error.

As previously stated, the height controller's altitude determination is not as accurate as in the post-processing algorithms. During the step response tests, it is clear that the controller either loses or gains altitude in response to other large control inputs, with a peak difference of 1.43 m. The controller consistently returns to within 0.8 meters of the desired altitude of 22 meters, corresponding to an error of 3.6%. This level of accuracy maintains a similar height that keeps a similar field of view, enabling the detection of parking spaces located over 10 meters away laterally.

# 4

# Discussion

This chapter analyzes the findings presented in the Results section. We first examine the distance estimation performance, followed by an evaluation of the control system's efficacy. The chapter concludes with an assessment of the system's limitations and recommendations for future research directions.

## 4.1 Distance Estimation Discussion

The system has achieved a MAE + $1\sigma$ of 3.14 cm for distances up to 2 meters, and up to 5.07 cm for distances between 2 and 10 meters. These error margins are comparable to those of established automotive ground-truth distance estimation systems, meeting the precision requirements necessary for verifying surround-view camera systems. However, all results are generated during stationary tests. Hence, the performance of the distance estimation algorithm has not been evaluated in scenarios where the car is moving. We have collected data for various scenarios using the procedure outlined in A.2, but due to time constraints, we have not had sufficient time to process and analyze the data. During these scenarios, we predict that the distance estimation errors will increase slightly. As the car moves, its relative position to the drone shifts, causing the estimated origin to drift slightly from the true position. Furthermore, during hard control inputs, the drone's altitude does not remain constant, as observed in the control system results. This could introduce an error, as the current bounding box algorithms assume that the drone is at a constant height, causing the car's bounding box to shift slightly in size and position relative to the car, thereby altering the origin.

The preliminary results for the distance estimation showed a systematic error that overcompensated distances. We suspect that the error may arise from imperfect camera calibration, resulting in a small amount of barrel distortion in the frame. This would, in turn, result in the image being enlarged near the center of the frame and compressed more tightly near the edges. This would result in a higher amount of pixels per ground unit distance along the edges. As the pixel-per-centimeter function assumes a fixed number of pixels per unit distance, this would lead to an overestimation of the distances.

### 4.1.1 Comparison with Industry Requirements and Other Automotive Validation Systems

In a comprehensive overview of the current literature and methods for "Surround-View Fisheye Camera Perception for Automated Driving" [31], the authors state that "surround-view perception poses additional challenges due to high precision object detection requirements of 10 cm and partial visibility of objects". Our system demonstrates a total accuracy of 4.86 cm (MAE + $1\sigma$) and 3.14 cm for distances of up to two meters. For parking applications, accurately estimating short distances is more important than estimating long distances. Hence, for short distances, our system is more than three times better than the general requirements for surround-view systems.

Our system was also compared to the widely used OxTS RT-Range distance verification system for automotive applications. The RT-Range system is specified to have a distance-to-lane or distance-to-point RMS accuracy of 2 or 3 centimeters, depending on the configuration. Comparing this to our system's RMS of 3.48 cm for up to ten meters, and 2.23 cm for up to two meters, our system is nearly on par with current state-of-the-art automotive verification systems. Therefore, the centimeter-level accuracy of the developed system was deemed sufficient for verifying surround-view camera systems.

### 4.1.2 System Benefits

The most common method for collecting ground truth data to verify automotive camera systems is to use a roof-mounted LiDAR, which generates high-definition 3D maps of the environment with high-precision distance estimations. However, the LiDAR can not detect the environment in close proximity to the car, since the car's roof obstructs its view. Our system solves this problem since the drone hovers more than 20 meters above the car and is able to see all of the vehicle's surroundings.

As stated earlier, RT-Range is also a widely used system for verifying automotive sensor distance estimations. Our system, however, has several benefits compared to the RT-Range system. The most significant benefit is the versatility. Our system is cheap, portable, and quick to deploy. In comparison, RT-Range is costly and is typically confined to automotive test tracks, as it requires a fixed base station. Additionally, it needs to be installed and calibrated in each car every time it is used, making the process time-consuming and even more costly.

## 4.2 Control System Discussion

The qualitative results for the drone controller exceeded the typical velocity requirements for parking maneuvers, demonstrating robust tracking of the target car while consistently capturing high-quality video data. The drone's stabilizing controllers, combined with the gimbal smoothing, resulted in increased freedom when tuning the PID parameters, as these systems enhance frame and drone stability. In some

instances, we even allow for some (decreasing) oscillations of the drone, which resulted in quicker positional performance, while the camera frame remained stable. However, the gimbal smoothing introduces a delay in the system, especially during rotations, where the drone rotates but the camera does not rotate with the same velocity. This results in the yaw controller response time being increased, which is why we cannot track full lock turns at the same velocity as longitudinally, even though the drone itself is capable of rotating with sufficient velocity. While this limitation could be addressed by increasing gimbal responsiveness or locking the camera orientation to the drone, we prioritized video quality over maximum tracking performance, as the achieved performance exceeded the requirements for parking applications.

## 4.3 Limitations and Future Work

### 4.3.1 Bounding Box and Car Origin Definement

Manually defining the bounding box, as described in Method 2.2.3, can yield highly accurate estimates of the car's origin position relative to its true origin. However, this restricts the speed at which data can be generated, since each generated video relies on manual labor. To remedy this, the bounding box (or origin) would need to be automatically identified and fitted. It was briefly explored to use the YOLO11 OBB (Oriented Bounding Boxes) object detection model to both detect and rotate the bounding box, instead of relying on the ArUco marker. However, the model's detections were not sufficiently accurate for our purposes. One could possibly train the model to achieve more accurate detections, but we believe that classical computer vision techniques would perform as well and run more efficiently. Whether machine learning (ML) is used or not, developing automatic methods for the car's bounding box may be worth exploring in future research.

### 4.3.2 Use of Markers

A main component in our system is the ArUco marker. It is used as the set point for the control system and as the reference size and length for the height and distance estimation functions. However, the detection of these markers is not as robust as we desired, leading to false detections and loss of detections in specific frames. This induced errors in the controller as we lost the reference signal, leading to a higher response time and a longer rise time for the controller. Future research should evaluate different types of markers, such as AprilTags, or consider using other features or methods to find a setpoint. However, since the marker is used as the reference length for both the height estimation and the pixel-to-centimeter algorithms, the system would require an additional reference length. Possibly, the car's length could be used for this, but this would demand even greater accuracy from the fitted bounding box.

The marker improved both height determination and pixel-to-cm conversion precision. The numerically derived function relating height to marker pixel circumference effectively bypasses errors in the camera and calibration models, resulting

in enhanced height estimation accuracy. We compared our function to the ArUco library functions for determining the camera height. Previous studies have shown that these built-in functions are estimated to have an average translational pose error of 2-3% for viewing distances up to ~7 m, [20] and [21]. Camera calibration heavily affects this accuracy, as errors in intrinsic parameters propagate to pose estimation errors. The calibration of the drone's camera resulted in a reprojection error of 0.58 pixels. However, our comparison did not produce translational pose errors as large as those reported in the cited literature. Notably, the built-in pose estimation functions align very well with the numerical function, with an average difference in estimated height of 0.37 percent. The remarkably small discrepancy observed can likely be attributed to the consistent central positioning of the marker in the camera frame during the comparison. This placement minimizes the influence of lens distortion, which is typically more pronounced at the edges of the frame. Consequently, any inaccuracies in our distortion model parameters would have a negligible impact on height estimation, which may explain the improved performance compared to previous studies where marker positioning varied across the field of view.

### 4.3.3 Ground Plane Slope

Our solution assumes that the ground plane is flat, which could become a limiting factor when applied to specific real-world scenarios. When measuring distances on non-planar surfaces, the algorithm introduces systematic errors that correlate with the surface's slope. The measured distances will appear shorter than the actual distances when measuring uphill and longer than the actual distances when measuring downhill. The magnitude of this error increases proportionally with the surface angle relative to the horizontal plane.

Although the system may be unsuitable for surfaces with steep slopes, we expect this limitation to be of minor importance in general real-world scenarios, as most parking spot surfaces are constructed to be flat. More specifically, both the American "The Americans with Disabilities Act" (ADA) [28] and the Swedish Boverkets byggregler (BBR) [29] states that for accessible parking spaces, which is the general recommendation for public parking spaces, the slope should not exceed 2.08% or 2%, respectively. With simple trigonometry, we can calculate these slopes to yield an error of ~2 mm per meter, which has a minor effect on the system's accuracy, especially for short distances.

# 5
# Conclusion

The system developed in this thesis has proved to be a versatile and accurate system for estimating real-world distances on the ground plane. The system exhibits centimeter-level accuracy for distances up to 10 meters. Hence, the system achieves the desired accuracy, making it suitable for use as a ground truth system in the development of surround-view systems for parking-related ADAS/AD.

After analyzing the control system's performance, it can be concluded that the decoupled PID control strategy worked as intended; however, there are cross-coupling effects between the variables that affect both their separate and total system performance. This, together with the gimbal smoothing, resulted in longer rise and settling times of the controllers but yielded the desired frame stability to enable accurate distance estimations.

# Bibliography


[1] DJI. "Mavic 2 Specs" dji.com. Accessed: May 8, 2025 [Online]. Available: https://www.dji.com/se/mavic-2/info

[2] Oxford Technical Solutions. "RT-Range." OxTS.com. Accessed: April 8, 2025 [Online]. Available: https://www.oxts.com/products/rt-range/

[3] Y. Zhang, Y. Hu, Y. Song, D. Zou, W. Lin, "Back to Newton's Laws: Learning Vision-based Agile Flight via Differentiable Physics", Shanghai Jiao Tong Univ., Shanghai, China, Jul. 2024, doi: 10.48550/arXiv.2407.10648

[4] P. Roque, E. Bin, P. Miraldo and D. V. Dimarogonas, "Fast Model Predictive Image-Based Visual Servoing for Quadrotors," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 7566-7572, doi: 10.1109/IROS45743.2020.9340759

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

[6] E. Bin. "MPC-based Visual Servo Control for UAVs", School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, 2020. [Online]. Available: https://kth.diva-portal.org/smash/get/diva2:1479331/FULLTEXT01.pdf

[7] J. Yingjie. "Online control of quadrotor and camera with MPC for robust vision-based flight", Dep. of Inform.Technol. Electrical Eng, ETH Zürich, Zürich, 2022. doi: 10.3929/ethz-b-000542229

[8] B. F. Jeon, H. Jin Kim, "Integrated Motion Planner for Real-time Aerial Videography with a Drone in a Dense Environment", Nov. 2019, doi: 10.48550/arXiv.1911.09280

[9] A. Reizenstein, "Position and Trajectory Control of a Quadcopter Using PID and LQ Controllers", Department of Electrical Engineering, Division of Automatic Control, Linköping University, Linköping, 2017. [Online]. Available: https://liu.diva-portal.org/smash/get/diva2:1129641/FULLTEXT01.pdf

[10] I. Kugelberg, "Black-Box Modeling and Attitude Control of a Quadcopter", Department of Electrical Engineering, Division of Automatic Control, Linköping University, Linköping, 2016. [Online]. Available: https://liu.diva-portal.org/smash/get/diva2:908582/FULLTEXT02.pdf

[11] Zenseact, "Zenseact Open Dataset (ZOD)", zod.zenseact.com, Accessed: Feb 17, 2025 [Online]. Available: https://zod.zenseact.com/

[12] Waymo, "Waymo Open Dataset", waymo.com, Accessed: Feb May 22, 2025 [Online]. Available: https://waymo.com/open/

[13] Citiscapes, "The Cityscapes Dataset", cityscapes-dataset.com, Accessed: Feb May 22, 2025 [Online]. Available: https://www.cityscapes-dataset.com/

[14] Motional, "nuScenes", nuscenes.org, Accessed: Feb May 22, 2025 [Online]. Available: https://www.nuscenes.org/

[15] L. Wang, B. Yu, C. Chen, “Parking Area Data Collection and Scenario Extraction for the Purpose of Automatic Parking ADAS Function”, 2020 IOP Conf. Ser.: Mater. Sci. Eng. vol. 780, pp. 032026, 2020. doi:10.1088/1757-899X/780/3/032026

[16] D. Zhao, M. Cao, L. Ding, Q. Han, Y. Xing and H. Ma, "DroneSense: Leveraging Drones for Sustainable Urban-scale Sensing of Open Parking Spaces," IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, London, United Kingdom, 2022, pp. 1769-1778, doi: 10.1109/INFOCOM48880.2022.9796795.

[17] S. Kim, Y. Tak, E. Barmpounakis and N. Geroliminis, "Monitoring Outdoor Parking in Urban Areas With Unmanned Aerial Vehicles," in IEEE Transactions on Intelligent Transportation Systems, vol. 25, no. 10, pp. 13393-13406, Oct. 2024, doi: 10.1109/TITS.2024.3397588

[18] O. E. Mahmoud, M. R. Roman and J. F. Nasry, "Linear and Nonlinear stabilizing control of Quadrotor UAV," 2014 International Conference on Engineering and Technology (ICET), Cairo, Egypt, 2014, pp. 1-8, doi: 10.1109/ICEngTechnol.2014.7016810

[19] OpenCv (Original Authors, S. Garrido, A. Panov), "Detection of ArUco Markers", docs.opencv.org, Accessed: Feb May 22, 2025 [Online]. Available: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

[20] B. Benligiray, C. Topal and Cuneyt Akinlar, "STag: A stable fiducial marker system", Image and Vision Computing, Volume 89, 2019, pp. 158-169, https://doi.org/10.1016/j.imavis.2019.06.007.

[21] J. Isaksson, L. Magnusson, "Camera pose estimation with moving Aruco-board", B.S. thesis, Jönköping University School of Engineering, Jönköping University, Jönköping, 2020, [Online]. Availible: https://www.diva-portal.org/smash/get/diva2:1505194/FULLTEXT01.pdf

[22] OpenCV, "Camera Calibration" docs.opencv.org. Accessed: May 8, 2025 [Online]. Available: `https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html`

[23] Z. Zhang, "A Flexible New Technique for Camera Calibration," Microsoft Research, Redmond, WA, USA, Tech. Rep. MSR-TR-98-71, Dec. 1998.

[24] Jason, "Distorton barrel and pincushion", Public Domain (CC0), Accessed: May 15, 2025, https://commons.wikimedia.org/w/index.php?curid=3935782

[25] M. Vajgl, P. Hurtik, T. Nejezchleba, "Dist-YOLO: Fast Object Detection with Distance Estimation" *Applied Sciences* vol. 12, no. 3, pp. 1354. Jan. 2022. doi: 10.3390/app12031354

[26] R. Suman, M. Bhavani, K. Madhav, G. Prabhav, S. Yogesh, "Distance Estimation to Support Assistive Drones for the Visually Impaired using Robust Calibration", Mar. 2025, doi: 10.48550/arXiv.2504.01988

[27] A. Criminisi, I. Reid, and A. Zisserman, “Single view metrology,” *International Journal of Computer Vision*, vol. 40, pp. 123–148, Nov. 2000. doi: 10.1023/A:1026598000963

[28] U.S. Department of Justice Civil Rights Division, "Accessible Parking Spaces", ada.gov, Accessed: May 22, 2025. [Online]. Available: https://www.ada.gov/topics/parking/

[29] Boverket, "Vad säger lagen om parkering". boverket.se, Accessed: May 22, 2025. [Online]. Available: https://www.boverket.se/sv/PBL-kunskapsbanken/teman/parkering_hallbarhet/lag/

[30] Davig G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints",University of British Columbia, Vancover B.C, Canada, 2004, Avalible: `https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf`

[31] V. R. Kumar, C. Eising, C. Witt and S. K. Yogamani, "Surround-View Fisheye Camera Perception for Automated Driving: Overview, Survey & Challenges," in IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 4, pp. 3638-3659, April 2023, doi: 10.1109/TITS.2023.3235057

[32] OpenCV Team, "OpenCV: Brute-Force Matcher Documentation," 2023. Available: `https://docs.opencv.org/4.x/d3/da1/classcv_1_1BFMatcher.html#details`. Accessed: May 22, 2025.

[33] ASAM OpenSCENARIO, "Coordinate systems", publications.pages.asam.net. Used under ASAM's unrestricted distribution license for OpenSCENARIO DSL. Accessed: June 18, 2025. [Online]. Available: `https://publications.pages.asam.net/standards/ASAM_OpenSCENARIO/ASAM_OpenSCENARIO_DSL/latest/index.html`

# A

# Appendix 1

## A.1 Aruco Pixel Circumference Test

To find the ArUco markers pixel circumference in the frame at different distances, the following test procedure was conducted:

- Place the ArUco marker on a wall or standing flat board, and the drone on a movable surface, facing the middle of the ArUco marker.
- Record short videos at 2-meter intervals between distances of 10 and 30 meters. Use a long measuring tape to measure the exact distance between the ArUco marker and the and drone's camera. Make sure to point the camera center towards the middle of the ArUco marker for each measurement.
- Postprocess the videos using OpenCv. It is important to set the Aruco detector's cornerRefinementMethod parameter to CORNER_REFINE_CONTOUR. We suggest using the arcLength() function to find the circumference from the ArUco corners. Save this value for each frame, and use the avrage as the datapoint for your function at the given distance.

## A.2 RT-Range Verification

To enable verification of both the systems static and dynamic performance, we collected ground truth distance data using the OxTS RT-Range system [2] available at Asta Zero. RT-Range is a differential GPS-based system designed for ADAS testing. It measures vehicle-to-vehicle and vehicle-to-line distances with an accuracy of up to 0.03 m. To facilitate testing, we used a set of movable road markings to create the parking spot. We collected the RT-Range distance estimations while operating the drone system during the following scenarios:

- Standstill outside of the parking spot, with different distances and positions.
- Standstill inside of the parking spot.
- Straight parking maneuver.
- 90° parking maneuver.

The RT-Range distance data from these scenarios could be used to assess the accuracy of the distance measuring algorithms. The primary benefit of RT-Range is the ability to assess the accuracy during a dynamic parking scenario.

**DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY**
**CHALMERS UNIVERSITY OF TECHNOLOGY**
Gothenburg, Sweden
www.chalmers.se


CHALMERS
UNIVERSITY OF TECHNOLOGY