FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Design of a loading/unloading System for a small payload cargo drone

Paulo Ricardo Santos e Silva

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Vítor Hugo Pinto

October 9, 2024

© Paulo Ricardo Santos e Silva, 2024

Resumo

Nos últimos anos, tem-se registado um aumento substancial no desenvolvimento de sistemas robóticos aéreos, particularmente nas áreas da entrega de encomendas e transporte de cargas. Este progresso tem atraído uma atenção significativa da comunidade de investigação e de várias indústrias. Os Veículos Aéreos não Tripulados (UAV) concebidos para efeitos de entrega recolhem os artigos e armazenam-nos de forma segura em compartimentos predefinidos. Estes robots aéreos navegam então através de áreas específicas para garantir a entrega eficiente e precisa dos artigos nos seus destinos.

Esta dissertação investiga a utilização de UAVs para o manuseamento eficiente e seguro de cargas durante a carga e a descarga, bem como a localização de UAVs durante a aterragem. O projeto visa enfrentar o desafio do transporte de pequenas cargas em ambientes industriais estruturados. Ao desenvolver uma solução para este problema, pretende-se aumentar significativamente a eficiência e a versatilidade das entregas aéreas de pequena escala.

Os objetivos desta dissertação incluem a conceção do sistema de entrega, o desenvolvimento de um protótipo de carga de pequena dimensão e a criação de uma arquitetura flexível para a localização do drone durante as fases de aterragem. Estes objetivos são apoiados por processos de teste e validação para garantir a eficácia e a fiabilidade dos sistemas propostos.

Estruturado com uma introdução abrangente, o documento desdobra-se em capítulos que oferecem *backgrounds*, uma revisão da literatura, descrições detalhadas das fases de desenvolvimento do sistema e um resumo conclusivo que encapsula as principais conclusões, implicações da investigação e perspectivas de trabalho futuro.

Ao combinar conhecimentos teóricos com implementação prática, esta dissertação tem como objetivo estabelecer uma referência no setor da indústria logística, fornecendo uma ferramenta valiosa para a comunidade científica e abrindo caminho para investigação e desenvolvimento inovadores no domínio dos UAV.

Abstract

Over the past few years, there has been a substantial increase in the development of airborne robotic systems, particularly in the areas of package delivery and transportation of payloads. This progress has garnered significant attention from the research community and various industries. Unmanned Aerial Vehicles (UAVs) designed for delivery purposes collect items and securely store them in predefined compartments. These aerial robots then navigate through specified areas to ensure the efficient and precise delivery of items to their destinations.

This dissertation delves into the use of UAVs for efficient and secure handling of payloads during loading and unloading, as well as localising UAVs during landing. The project aims to tackle the challenge of transporting small payloads in structured industrial environments. By developing a solution to this issue, it aims to significantly enhance the efficiency and versatility of small-scale aerial deliveries.

The objectives of this dissertation include the design of the delivery system, the development of a small payload cargo prototype, and the creation of a flexible architecture for drone localisation during landing phases. These objectives are supported by testing and validation processes to ensure the efficacy and reliability of the proposed systems.

Structured with a comprehensive introduction, the document unfolds into chapters that offer backgrounds, a literature review, detailed descriptions of the system's development phases, and a conclusive summary that encapsulates key findings, research implications, and future work perspectives.

By combining theoretical insights with practical implementation, this dissertation aims to establish a benchmark in the logistics industry, providing a valuable tool for the scientific community and paving the way for innovative research and development in the field of UAVs.

Sustainable Development Goals of the United Nations

Delivery drones are a technological innovation that could have a significant impact on various aspects of sustainable development. This chapter examines how the use of delivery drones aligns with the United Nations' Sustainable Development Goals (SDGs), identifies the specific targets within these goals that the project addresses, evaluates the project's contributions to improving quality of life and environmental protection, and outlines performance indicators to measure its impact.

SDG	**Target**	**Contribution**	**Performance indicators and metrics**
3	8	Delivery drones can quickly transport medical supplies to underserved or isolated communities, improving access to healthcare and emergency medical services.	Number of medical deliveries made to underserved areas, reduction in delivery times for critical supplies and improvements in health outcomes
9	4	By integrating drones into the logistics chain, companies can reduce their carbon footprint and optimize delivery routes, enhancing resource efficiency.	Delivery speed, cost savings and reliability rates
11	6	Drones, being electrically powered, offer a lower-emission alternative to traditional delivery vehicles, thereby reducing urban air pollution and congestion.	Reduction in traffic congestion, improvement in air quality indices and carbon footprint reduction compared to traditional delivery vehicles
13	2	The use of drones can be part of broader national strategies to reduce emissions in the transportation sector, supporting climate action initiatives.	Measurement of carbon footprint reduction, integration of drones in national climate strategies and contribution to emission reduction targets

"Doubt is the origin of wisdom."

René Descartes

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation . . . 1
- 1.3 Goals . . . 2
- 1.4 Document Structure . . . 2

2 Background and Fundamental Aspects **3**
- 2.1 Drones . . . 3
  - 2.1.1 General Architecture . . . 3
  - 2.1.2 Use Cases . . . 5
  - 2.1.3 Advantages of Drone Delivery Services . . . 5
  - 2.1.4 Key challenges and solutions in autonomous delivery . . . 6
- 2.2 Camera Calibration . . . 7
  - 2.2.1 Camera Models . . . 7
  - 2.2.2 Linear Camera Model . . . 8
  - 2.2.3 Internal and External Parameters . . . 8
  - 2.2.4 Non-Linear Camera Model . . . 9
  - 2.2.5 Types of Distortion . . . 10
  - 2.2.6 Calibration Methods . . . 11
  - 2.2.7 Calibration Parameters . . . 11

3 Literature Review **12**
- 3.1 Loading/Unloading Drone Systems . . . 12
- 3.2 Comparison between the Loading/Unloading Drone Systems . . . 19
- 3.3 Indoor Localisation Technologies . . . 20
  - 3.3.1 Methodologies . . . 20
  - 3.3.2 Indoor Positioning Systems . . . 21
- 3.4 Comparison between Indoor Positioning Systems . . . 23
- 3.5 Localisation during Landing . . . 24
- 3.6 Landing Control . . . 26
  - 3.6.1 Detection of landing marker and estimation of position . . . 26
  - 3.6.2 Proportional Integral landing controller . . . 27
  - 3.6.3 Low-level position control . . . 27

4 Prototype Design **28**
- 4.1 Hardware Prototype . . . 28
  - 4.1.1 Raspberry Pi Pico W . . . 29
  - 4.1.2 PCA9685 . . . 29

4.1.3 Servo Motor . . . 30
4.1.4 Buck Converter . . . 31
4.1.5 Bearing . . . 31
4.1.6 3D Parts Design . . . 32
4.2 Communication . . . 34
4.3 Software Development . . . 35
4.3.1 External Libraries . . . 36
4.3.2 Code Structure . . . 36
4.3.3 Implementation . . . 37
4.3.4 State Machine . . . 38
4.3.5 Web server . . . 38
4.4 Results . . . 40

5 Localisation Algorithm during Landing **41**
5.1 Camera Calibration . . . 41
5.1.1 Software . . . 42
5.1.2 Save Snapshots script . . . 42
5.1.3 Camera Calibration script . . . 43
5.2 Pose estimation . . . 43
5.2.1 Algorithm . . . 44
5.2.2 Functionality . . . 45
5.3 Results . . . 46

6 Conclusion and Future Work **52**

References **54**

List of Figures

2.1 Architecture of a drone . . . 3
2.2 Pinhole Camera Model . . . 8
2.3 Transformation from 3D world coordinates to 2D computer image coordinates using the linear camera model . . . 8
2.4 Transformation from 3D world coordinates to 2D computer image coordinates using the non-linear camera model . . . 9
2.5 Radial distortion . . . 10
2.6 Tangential distortion . . . 10
2.7 Calibration parameters along the coordinate transformation direction . . . 11

3.1 UAV with payload release mechanism . . . 13
3.2 Quadcopter with payload basket . . . 14
3.3 3D Mechanism . . . 15
3.4 Two adaptable, lightweight grasping mechanisms . . . 16
(a) Net-based gripper design attached on hexacopter . . . 16
(b) Slider-based gripper configured as a shape-shifting UAV . . . 16
3.5 Origami-inspired cargo drone . . . 17
3.6 Quadcopter with the parcel placing above the rotor plane . . . 18
3.7 The prototype of the reconfigurable, tethered aerial grasping system is depicted attached to a co-axial octacopter . . . 18
3.8 Example of an E-ArUco . . . 25
3.9 Pipeline for UAV Landing Control . . . 26

4.1 Electrical Circuit Diagram for Loading/Unloading Operations. . . . 28
4.2 Raspberry Pi Pico W. . . . 29
4.3 PCA9685. . . . 30
4.4 MG90S Servo Motor. . . . 30
4.5 PWM Signal . . . 31
4.6 XL4005 Buck Converter. . . . 31
4.7 618/6 2Z Bearing. . . . 32
4.8 Design of the loading/unloading system. . . . 32
4.9 Box. . . . 33
(a) Design . . . 33
(b) Section Analysis . . . 33
4.10 Assembly of all electronic components. . . . 35
4.11 State Machine. . . . 38
4.12 Web page used for testing. . . . 39

5.1 Discarded image during Camera Calibration. . . . 43

5.2 Original ArUco with ID = 100. 44
5.3 Camera pose relative to the marker frame. 45
5.4 Errors as a function of height. 49
(a) Absolute Error on the x-axis 49
(b) Relative Error on the x-axis 49
(c) Absolute Error on the y-axis 49
(d) Relative Error on the y-axis 49
(e) Absolute Error on the z-axis 49
(f) Relative Error on the z-axis 49
(g) Absolute Error of the distance to the z-axis 49
(h) Relative Error of the distance to the z-axis 49
5.5 Connection Pinout between Pixhawk's telemetry port and Raspberry Pi. 50

List of Tables

3.1 Comparison between the loading/unloading systems. 19
3.2 Comparison between the indoor position systems 23

5.1 Positions obtained from the OptiTrack system in centimeters. 47
5.2 Positions obtained from the developed algorithm in centimeters. 47
5.3 Mean of the z-values obtained from the OptiTrack system at the same height in centimeters. 48
5.4 Mean of the absolute errors of the developed algorithm at the same height in centimeters. 48
5.5 Mean of the relative errors of the developed algorithm at the same height in percentage. 48

Abbreviations and Symbols

3D	Three Dimensional
CAD	Computer-Aided Design
GPS	Global Positioning System
IPS	Indoor Positioning Systems
PID	Proportional Integral Derivative
PI	Proportion Integral
PLA	Polylactic Acid
PWM	Pulse Width Modulation
R-CNN	Region-Based Convolutional Neural Network
UAV	Unmanned Aerial Vehicle
YOLOv3	You Only Look Once, Version 3

Chapter 1

Introduction

This first chapter provides a detailed perspective on the Dissertation by offering insights into its context, reasons for undertaking it, objectives, and document organisation. It aims to foster a comprehensive and clear understanding of the dissertation's ambitions, laying the foundation for the entire document.

1.1 Context

Modern industrial environments face increasing complexity and demands, making traditional material handling methods often inadequate. To address scalability, responsiveness, and cost issues, there's a growing trend towards automation technologies like Unmanned Aerial Vehicles (UAVs). These UAVs are expected to boost productivity, cut costs, and enhance adaptability to changing production needs.

This dissertation focuses on designing a UAV-based delivery system for transporting small payloads in industrial settings. The project involves creating a small payload cargo prototype to explore practical drone applications in material handling.

The interdisciplinary approach combines Electrical and Computer Engineering, utilising Autonomous Systems, Robotics, Control Systems, and Computer Vision. It integrates Mechanical Engineering aspects into computer-aided design, playing a crucial role in developing the drone's physical prototype. This integration aims to develop a comprehensive solution for the logistics industry's evolving demands.

1.2 Motivation

The driving force behind this research project is to develop a versatile product that serves the broader scientific community, transcending the constraints of an enterprise-specific use case. The system to be designed must be lightweight and implemented on physical hardware, following industrial requirements, which implies its unavailability on the commercial market. Furthermore,

it should adopt a general methodology that facilitates the accessible design of modular systems, enabling the migration of the overall solution to different manufacturing contexts.

This approach seeks to significantly advance knowledge in the field, promoting innovation and collaboration within the scientific community. By creating a solution that can be easily adapted and integrated across different sectors, we aim to foster a collaborative environment where researchers and industry professionals can work together to push the boundaries of what is possible. Proof that this work is important from an industrial point-of-view, is that this dissertation was developed under a Research and Development project "Drone system for quality control".

Ultimately, this project aims to provide a valuable tool for the scientific community and to stimulate further research and development, leading to new innovations and improved manufacturing processes.

1.3 Goals

The specific objectives of this dissertation are:

- Design of a delivery system for a drone and its loading/unloading mechanical techniques;
- Design of a small payload cargo prototype;
- Development of a flexible architecture to acquire relevant data for drone location while landing;
- Test and validation of the proposed systems.

1.4 Document Structure

Apart from the Introduction, this document comprises five additional chapters. Chapter 2 delves into the necessary background information. Chapter 3 provides an overview of the current state of the field, along with discussions on relevant works. Chapter 4 describes the design and implementation of the loading/unloading system. Chapter 5 focuses on the localisation algorithm during landing. The last chapter, 6, presents the conclusions derived from the work conducted during the Dissertation, summarising the key findings, discussing the implications of the research, and suggesting potential areas for future work.

Chapter 2

Background and Fundamental Aspects

In this chapter, it will be exposed some fundamental knowledge that is relevant to the understanding of the engineering area where the dissertation is contextualised.

2.1 Drones

This section, based on [1] and [2], delves into the intricate architecture of drones, exploring their key components and functionalities. It examines the myriad applications of drones, shedding light on their versatility and impact across various domains. Furthermore, it elucidates the advantages drones offer, while acknowledging the challenges they confront and presenting potential solutions to overcome them. Through this exploration, the section provides a comprehensive overview of drones' capabilities, benefits, and the complexities involved in their implementation.

2.1.1 General Architecture

The key components that collectively facilitate flight and perform various functions on drones are examined. Figure 2.1 depicts a general architecture of a drone.

Figure 2.1: Architecture of a drone [2].

These primary components include the frame, propulsion system (comprising motors and propellers), electronic speed controller (ESC), flight controller, radio receiver/transmitter and power source. Each element is discussed to provide a comprehensive understanding of their individual and collective roles.

Frame

The frame functions as the drone's skeleton, offering a structure that holds all other components intact. It ensures stability, maintains the overall shape, and contributes to the durability and performance of the drone. Additionally, the frame must be designed to balance weight and strength, achieving an optimal combination of lightness and resilience.

Propulsion System

The propulsion system in drones comprises motors and propellers, which work in tandem to enable flight and control of the UAV.

The primary purpose of motors, typically electric, is to rotate the propellers mounted atop them, driving the propellers to generate thrust. This thrust enables the drone to lift off, hover, and maneuver. Motor selection is closely linked to the weight of the drone and impacts decisions regarding other elements like batteries or ESCs.

Propellers play a vital role in enabling the drone to achieve flight and perform various tasks. Selection of propellers should be based on factors such as the length of the drone's arms and the speed of its motors. It is recommended to use smaller propellers for motors with higher spin speeds.

Electronic Speed Controller

An ESC regulates the power delivered to each motor based on commands from the flight controller, ensuring precise control over the drone's movement. Each ESC is tasked with controlling a single motor, so the number of ESCs required depends also on the drone's configurations. They receive signals from the flight controller and adjust motor speed and direction accordingly. ESCs connect to the battery either directly or via a power distribution board and often include a Battery Eliminator Circuit to power other components. Selecting ESCs with a high refresh rate is essential for achieving precise motor control and smooth operation.

Flight Controller

The flight controller acts as the "brain" of the drone, incorporating sensors such as gyroscopes, accelerometers, and magnetometers to ensure stability and control. It processes inputs from the pilot or autonomous system, along with data from these sensors, to ascertain and adjust the spin speed of each motor. This capability enables precise navigation and maneuvering, crucial for the drone's optimal performance.

Radio Receiver/Transmitter

The radio receiver and transmitter facilitate communication between the drone and the remote controller, enabling the pilot to transmit commands and receive telemetry data. It is essential to consider the number of channels, with a minimum of four required for basic multi-rotor control. Each channel governs specific functions, typically including throttle, pitch, yaw, and roll, ensuring precise control over the drone's movements.

Power Source

The drone's power source, commonly a rechargeable battery, plays a crucial role in supplying the necessary electrical energy to operate various components such as motors, onboard electronics, and other systems. While certain drones may utilise alternative energy sources like fuel cells, the majority rely on lightweight and compact Lithium Polymer (LiPo) batteries. Selecting the appropriate battery depends on factors such as the type of drone motors, the desired flight duration, and the required discharge rate to ensure adequate speed and performance.

2.1.2 Use Cases

The use of autonomous or remotely-controlled flying vehicles holds significant potential across various applications, promising substantial benefits across multiple contexts:

- **Environmental Conservation:** monitoring wildlife and forest surveillance;
- **Emergency Preparedness:** disaster assessment and search and rescue for survivors;
- **Crime Fighting:** fugitive tracking and border supervision;
- **Culture and Leisure:** aid in historical monuments preservation and enrich the exploration and tourism experience;
- **Delivery Services:** efficient deliveries and access to remote areas.

2.1.3 Advantages of Drone Delivery Services

The utilisation of drones for deliveries offers numerous advantages. Firstly, they facilitate faster deliveries by circumventing ground based-delays and navigating directly to their destination, thereby ensuring enhanced customer satisfaction through swift delivery. Secondly, drones provide access to remote and challenging terrains, such as mountain regions or isolated villages with poor road connections, expanding the reach of delivery services to previously underserved areas. Additionally, the adoption of drones reduces the reliance on road-based deliveries, mitigating traffic congestion and reducing the wear and tear on road infrastructure.

Moreover, autonomous drones diminish the need for human labor, resulting in lower operational costs for delivery companies. This transition to automation also enhances safety and security, as drones can be equipped with advanced navigation and tracking systems. Furthermore, by

minimizing ground-based transportation, drones contribute to environment sustainability by emitting fewer pollutants and reducing carbon emissions, thus promoting cleaner air and a healthier environment.

Overall, the integration of drone technology into delivery services offers a promising solution for improving efficiency, accessibility, cost-effectiveness, and environmental impact.

2.1.4 Key challenges and solutions in autonomous delivery

Autonomous flight for deliveries involves a range of challenges that must be addressed to ensure the safety, reliability and efficiency of the system. Below are the key issues to consider, along with potential solutions to mitigate these risks:

Safety

- **Issues:**
  - Collision with people, objects, or other drones;
  - Malfunctioning of hardware or software;
  - Environmental hazards.
- **Actions:**
  - Obstacle Detection and Avoidance;
  - Redundancy Systems;
  - Geofencing;
  - Weather Adaptability.

Correct delivery

- **Issues:**
  - Incorrect delivery address;
  - Dropping package at wrong location;
  - Theft or tampering with the package.
- **Actions:**
  - Precision Navigation;
  - Package Verification;
  - Secure Package Handling.

Ability to correctly return to the base

- **Issues:**
  - Loss of signal or communication with the base;

  - Depletion of battery or fuel;
  - Navigational errors.

- **Actions:**

  - Reliable Communication Systems;
  - Energy Managment;
  - Robust Navigation.

Drone preservation

- **Issues:**

  - Wear and tear of drone components;
  - Exposure to harsh environmental conditions;
  - Crashes or hard landings.

- **Actions:**

  - Durable Construction;
  - Regular Maintenance and Monitoring;
  - Safe Landing and Takeoff.

By addressing these aspects comprehensively through a combination of advanced hardware and sophisticated software solutions, the risks associated with autonomous drone deliveries can be significantly mitigated, ensuring a safer, more reliable, and efficient delivery system.

2.2 Camera Calibration

The core of machine vision technology lies in deriving the geometric information of 3D objects from the images captured by cameras, which are the primary tools used for image acquisition [3].

Camera calibration aims to employ the coordinates of feature points (X, Y, Z) from an object in 3-D space and its image coordinates (x, y) in 2-D image space. This is to compute the camera's internal and external parameters, thus creating a quantitative relationship between the actual scene and the image captured. The intrinsic parameters represent the optical center and focal length of the camera, while the extrinsic parameters encompass the camera's positional data within the global coordinate system [3].

2.2.1 Camera Models

The camera model describes how coordinates in the world coordinate system correlate with coordinates in the image coordinate system. Essentially, it defines how a point on an object projects onto an image point through the camera model [3].

2.2.2 Linear Camera Model

The linear model, also known as the pinhole model, describes a basic camera without a lens and with a single small aperture. Light rays enter through this opening, projecting an inverted image on the camera's opposite side [4].

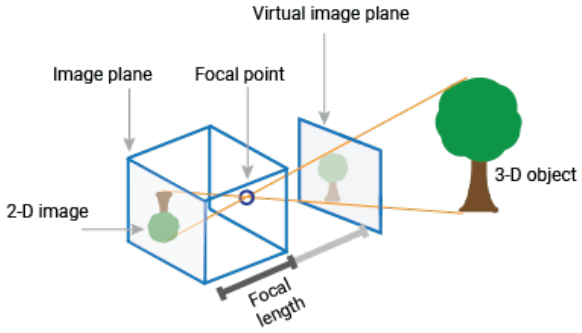


Figure 2.2: Pinhole Camera Model [4].

A comprehensive image processing operation entails three transformations across four non-coincident coordinate systems under the linear camera model, as illustrated in figure 2.3 [3].

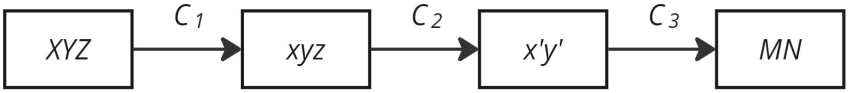


Figure 2.3: Transformation from 3D world coordinates to 2D computer image coordinates using the linear camera model [3].

1. Rigid transformation C_1 from the 3D world coordinate system XYZ to the 3D camera coordinate system xyz.
2. Projective transformation C_2 from the 3D camera coordinate system xyz to the 2D image plane coordinate system $x'y'$.
3. Conversion C_3 from the 2D image plane coordinate system $x'y'$ to the 2D computer image coordinate system MN, where M represents the total number of pixel rows and N represents the total number of pixel columns in the computer's memory (also known as computer coordinates).

2.2.3 Internal and External Parameters

The parameters required for calibrating a camera can be categorised into external parameters, which are outside the camera, and internal parameters, which are located inside the camera.

Understanding the difference between external and internal parameters is crucial when capturing multiple images with a camera from various positions and angles. While the camera's external parameters may vary with each image due to its changing position and orientation, the internal parameters remain constant. Therefore, after adjusting the camera's position, only the external parameters need recalibration, not the internal ones [3].

External parameters

The extrinsic parameters indicate the camera's position within the 3D world coordinate system, which is the initial phase of the process depicted in figure 2.3. They consist of a rotation and a translation, comprising six external parameters: the Euler angles (roll, pitch and yaw) and the translations along the three axes. The camera's coordinate system origin is at its optical center, with the x and y axes delineating the image plane [4].

Internal parameters

The intrinsic parameters map camera coordinates to the image plane, representing last two transformations depicted in figure 2.3. They consist of the focal length, the optical center, and the skew coefficient [4].

2.2.4 Non-Linear Camera Model

The pinhole camera model does not include lens distortion due to the fact that an ideal pinhole camera lacks a lens. In real-world scenarios, cameras typically employ lenses for capturing images. However, various elements like the manufacturing and installation of the lens can affect the camera's ability to project images. Consequently, the projection dynamics of a camera cannot be precisely encapsulated by a linear projection model, indicating the linear model is insufficient for accurately depicting the geometrical imaging relationships of the camera [3].

In practice, the functioning of an actual optical system does not perfectly align with the theoretical pinhole imaging concept, due to the presence of lens distortion. The error in optical distortion, which is the difference between the actual image point created by an object in space on the camera's imaging plane and the perfect image, becomes more pronounced near the edges of the lens. This results in alterations of the detected coordinates, compromising the precision of the world coordinates acquired. Hence, for camera calibration, it's essential to employ a non-linear camera model that accounts for such distortions as shown in figure 2.4 [3].

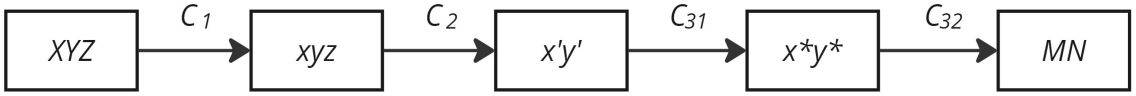


Figure 2.4: Transformation from 3D world coordinates to 2D computer image coordinates using the non-linear camera model [3].

Considering the transformation C_{31} from the undistorted 2D image plane coordinate system $x'y'$ to the actual 2D image plane coordinate system x^*y^*, the process of converting 3D world coordinate system to 2D computer image plane coordinate system follows the non-linear camera model, as shown in figure 2.4. The original conversion C_3 has been split into two distinct conversions, labeled C_{31} and C_{32}. The conversion from $x'y'$ to x^*y^* demonstrates the non-linearity [3].

2.2.5 Types of Distortion

Due to the impact of several distorting factors, when a point from a three-dimensional space object is projected onto a two-dimensional image plane, the coordinates of the image point that are actually obtained will differ from the coordinates of the ideal, undistorted image point. There are primarily two types of distortion: radial distortion and tangential distortion [3].

Radial Distortion

Radial distortion primarily results from irregularities in lens shape, specifically errors in its surface curvature. This distortion typically manifests symmetrically around the camera lens's central optical axis and becomes more pronounced the further out from this axis along the lens's radius. This causes straight lines in the real world to seem curved in images. Positive distortion is referred to as pincushion distortion, while negative distortion is known as barrel distortion, as illustrated in figure 2.5 [3].

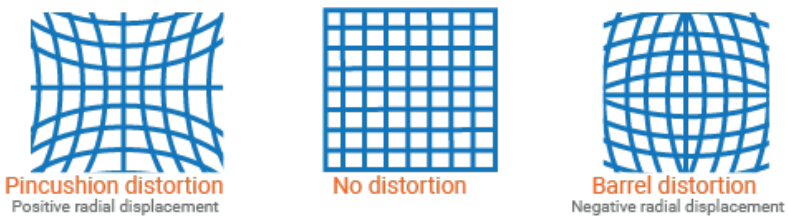


Figure 2.5: Radial Distortion [4].

Tangential Distortion

Tangential distortion is mainly caused by the non-collinear optical centers of the lens group, meaning the angle between the lens and image sensor. This results in the actual image point moving tangentially on the image plane, causing the image to appear tilted and stretched. This type of distortion has a specific orientation in space, with a maximum axis of distortion in one direction and a minimum axis in the perpendicular direction, as illustrated in figure 2.6 [3].

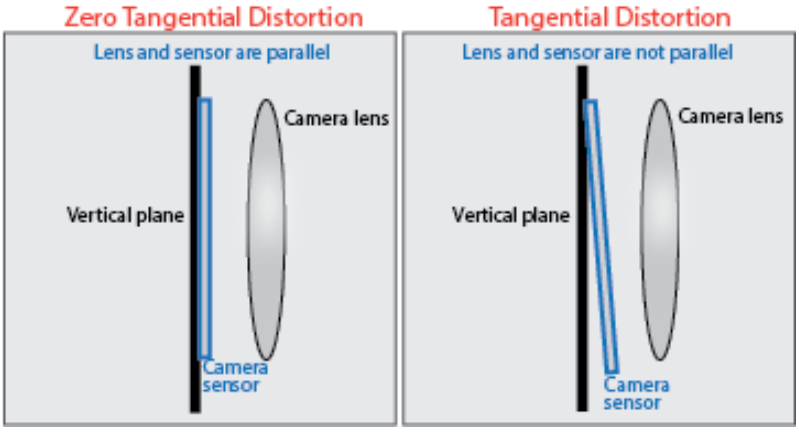


Figure 2.6: Tangential Distortion [4].

2.2.6 Calibration Methods

Conventional camera calibration requires a calibration object with known dimensions and shape, including the position and distribution of calibration markers. By matching these markers on the calibration object with their counterparts in captured images, both the camera's internal and external parameters can be determined [3].

2.2.7 Calibration Parameters

The calibration process can be performed following the conversion pathway from 3D world coordinates to computer image coordinates. A common methodology for this is illustrated in figure 2.7. The process of converting the world coordinate system to the computer image coordinate system comprises four phases, each requiring calibration of specific parameters [3].

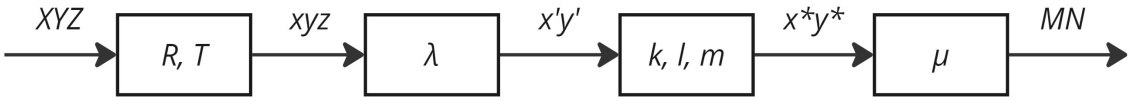


Figure 2.7: Calibration parameters along the coordinate transformation direction [3].

Phase 1 involves calibrating the rotation matrix R and the translation vector T. In phase 2, the focal length of the lens, denoted as λ, requires calibration. The calibration process in phase 3 addresses the lens radial distortion coefficient k, eccentric distortion coefficient l, and thin lens distortion coefficient m. Finally, phase 4 focuses on calibrating the uncertainty image scale factor, μ [3].

Chapter 3

Literature Review

This chapter is dedicated to a comprehensive exploration of the references encountered during research, aiming to provide a contextual understanding of the current state-of-the-art of loading/unloading drone systems, indoor localisation technologies and methods for controlling the landing of UAVs. The various authors proposals are discussed, highlighting both the advantages and disadvantages of each. Additionally, the works are compared and contrasted with one another. In the end, it is contextualized the current work within the broader state-of-the-art landscape, highlighting its unique contribution and potential impact on the field.

3.1 Loading/Unloading Drone Systems

Over the past few decades, there has been a notable surge in the development of aerial robots for applications such as monitoring, grasping, package delivery, and the transportation of arbitrary payloads, gaining momentum in both research and industry. Delivery UAVs operate by gathering items and storing them in a specified compartment. Subsequently, they navigate through a designated area and deposit the items at the intended destination. Key considerations in designing the UAV mechanism encompass the dimensions and weight of packages, security measures, and the effectiveness of the loading and unloading processes. Eight different loading/unloading systems will be introduced.

In [5], the design of the system features a unique hex-rotor configuration where propellers overlap. To optimize dimensions without compromising efficiency, half of the motors are inverted, resulting in a slight overlap between each propeller. Additionally, the system is outfitted with an innovative lightweight servo-activated two-ring release mechanism, enabling it to effortlessly carry and release payloads weighing up to two kilograms with minimal physical force required.

The payload compartment is situated beneath the lower plate of the airframe to facilitate vertical dropping of payloads from the aircraft. The servo mechanisms responsible for payload release are strategically mounted on top of the center frame [5].

A novel payload release mechanism has been developed, drawing inspiration from parachute equipment known as the three-ring release. This mechanism is designed to progressively reduce

the force required to secure payloads, utilizing only two rings due to the significantly lighter mass of the payload compared to a typical parachute load. The setup involves a large ring attached to the payloads, with a smaller ring threaded through a patch of nylon connected to the UAV's anchor point. The smaller ring is then threaded through the large ring, allowing the payload ring to rest on the nylon. A piece of string, also attached to the anchor, is threaded through the smaller ring and a hole in the anchor, creating a loop where the release pin is inserted. The force needed to pull the release pin is greatly diminished, enabling a 6-gram servo to effortlessly perform the task. This mechanism offers simplicity in design, affordability, and lightweight advantages [5].

Numerous tests and analyses have confirmed that the system consistently meets the release requirements with success. In summary, the system is designed to operate autonomously, efficiently executing a sequence of tasks that include navigating waypoints, delivering payloads, and then returning to its base [5].

Figure 3.1 shows the system presented.

Figure 3.1: UAV with payload release mechanism [5].

In [6], the authors aimed to address the query of the optimal method for releasing a package upon the drone's arrival at the designated drop-off point. The proposed solution outlines the use of a basket designed to securely retain the package until the drone executes a swift aerial maneuver, commonly referred to as a roll. During this maneuver, the centrifugal force is leveraged to expel the payload from the basket. Additionally, the system incorporates the activation of a parachute to ensure the safe delivery of the payload.

The objective was realized through the design and construction of a quadcopter equipped with 6 degrees of freedom, featuring an attached package carrier that facilitates successful parachute assisted delivery. The project culminated in the development of a fully operational quadcopter with the ability to hover, ascend, descend, pitch, yaw, and roll using a controller transmitter. Subsequently, a specifically designed basket was created through 3D printing and securely attached to the top of the drone, allowing the payload to remain safely housed inside. Upon executing a flip or roll maneuver, the centrifugal force is harnessed to expel the payload away from the aircraft [6].

In order to enable the drone to achieve rapid rotation around its axes for swift payload ejection, certain modifications to the drone's programming were essential. Through adjustments made in the flight controller software, the maximum angular velocity was heightened from the default 800 degrees per second to an almost 1200 degree rate. This augmentation in the rate of rotation provides the pilot with enhanced precision for making minor adjustments, while also allowing the capability to swiftly accelerate the angular velocity when necessary [6].

The payload envelope of the basket is designed to accommodate a single-serve snack, and its specific 10º taper is engineered for effective and efficient payload delivery during the drone's roll maneuver. Following the completion of the roll maneuver, the design allows for the seamless dispensing of the payload, triggering the activation of the parachute for a safe descent [6].

At the end, a cost-effective and reliable solution was developed, modeled, investigated, and successfully tested, achieving a 100% success rate. This innovation offers aerial robotics experts an alternative to the conventional approach of employing extra motors to facilitate payload delivery. As a result, it reduces costs, simplifies the process, and mitigates the risk of failures associated with drone deliveries [6].

Figure 3.2 shows the system presented.

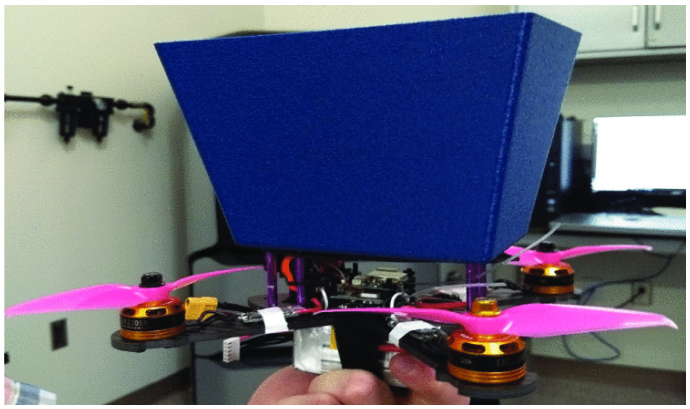


Figure 3.2: Quadcopter with payload basket [6].

In [7], a drop-box concept was envisioned and created using Computer-Aided Design (CAD), more concretly SOLIDWORKS, with a focus on designing a storage container that securely stores parcels internally, rather than exposing them. The dimensions of the mechanism must be compact enough to fit underneath a UAV.

The mechanism, being composed of individual parts, will initially undergo assembly within SOLIDWORKS using the "mate" feature to join various components. Subsequently, it's crucial to pinpoint any interference points to ensure proper attachment of all parts, preventing fitting issues during the Three Dimensional (3D) printing phase. Once interference checks are complete and no issues are identified, the design can proceed to the analysis stage using SOLIDWORKS simulation feature. Throughout this process, an "advisor" option is accessible for each step, guiding users through the analysis procedure [7].

To initiate the process, the first step involves assigning a material to the model to enhance the accuracy of the analysis. Subsequently, fixture points and connections are designated. Moving forward, an external force is applied, specifically within the storage box, simulating the action of placing a parcel. This step helps in simulating and analyzing the behavior of the mechanism under realistic conditions. The analysis reveals that the design can effectively handle a load of 2 kg, providing valuable information about its load-bearing capacity [7].

In order to accommodate a heavier load, adjustments would be required, such as increasing the thickness of the walls. However, this modification would result in a bulkier and heavier storage body. The consequence of these changes would be an increased cost due to the additional material required. Moreover, the added weight could potentially exceed the payload limits of different UAVs, necessitating careful consideration of the impact on the UAV's load-bearing capacity. Balancing the need for a higher load capacity with the constraints imposed by weight and cost is essential in refining the design [7].

The mechanism is equipped with three servo motors, with two positioned on the sides to manage the tilting movement. A third servo motor, located on the top of the main body, is responsible for securing the door to prevent unintentional opening during airborne operation. To activate these servo motors, an ultrasonic sensor and a force pressure sensor are incorporated into the design. The ultrasonic sensor, situated at the bottom, detects proximity to the ground, while the force pressure sensor inside the storage holder identifies the presence of a parcel, facilitating precise and automated control of the parcel to drop-off without requiring the recipient to retrieve it manually [7].

Figure 3.3 shows the system presented.

Figure 3.3: 3D Mechanism[7].

In [8], two adaptable, lightweight, and durable grasping mechanisms are suggested for integration into aerial robotic vehicles to enhance their grasping capabilities.

The first solution involves an extremely lightweight, net-based design employing twisted string actuation to bring together eight 3D-printed cylinders connected to a net, which conforms around

the object. The second solution presents a reconfigurable, slider-based grasping mechanism resembling a supersized parallel jaw gripper. It utilizes 3D-printed pads that slide on two carbon fiber shafts with linear bearings. Notably, the second design can be configured as a shape-shifting, grasping-capable device, effectively serving as a flying gripper [8].

The efficacy of these proposed grasping mechanisms has undergone experimental validation through a series of experiments that focus on grasping everyday objects. The assessment includes evaluating their force exertion capabilities and durability, affirming their practicality and performance [8].

Both mechanisms are depicted in figure 3.4.

(a) Net-based gripper design attached on hexacopter

(b) Slider-based gripper configured as a shape-shifting UAV

Figure 3.4: Two adaptable, lightweight grasping mechanisms [8].

In [9], the concept involves incorporating a quadcopter into a collapsible protective enclosure. This enclosure serves as a comprehensive protective framework, effectively isolating the propellers from the surroundings to ensure the safety of individuals. Both the drone and the protective cage can be conveniently folded with a single motion, greatly minimizing their size for convenient storage and transportation. The protective cage encompasses the entire multicopter, including the payload.

The cage is constructed using carbon-fiber tubes interconnected by pliable joints, crafted through 3D printing with a flexible material. Its design incorporates four key features that render it well-suited for load transportation. In the first place, the folding pattern is designed to impart rigidity to the cage in its deployed state, thereby guaranteeing stability throughout the flight. As a second consideration, the modular structure offers control over the spatial density of the cage. As a third feature, when in the folded configuration, the cage provides ample free space to encapsulate the components of the multicopter, ensuring their protection during transportation. The final feature involves positioning the cargo above the multicopter to prevent interference with the airflow generated by the propeller. This arrangement aims to avoid reduced efficiency caused by obstructions [9].

The cargo can be connected manually to the upper joint of the cage using two alternative interfaces. The first interface consists of a lightweight net with a round shape and ropes. The net's edges are tied with ropes to the central top joint, and the object is placed in the net's center.

This method is highly adaptable, allowing for the secure attachment of objects with various shapes easily. However, when parcels lack a shape conducive to lying close to the central top joint, they may experience swinging oscillations. To address this issue, the second type of interface involves using rigid boxes made of card or thin plastic. Nevertheless, boxes are heavier than nets, and using multiple boxes is necessary to transport objects of different shapes and sizes. These two interfaces thus serve as complementary solutions, depending on the nature of the transported cargo [9].

Figure 3.5 shows the system presented.

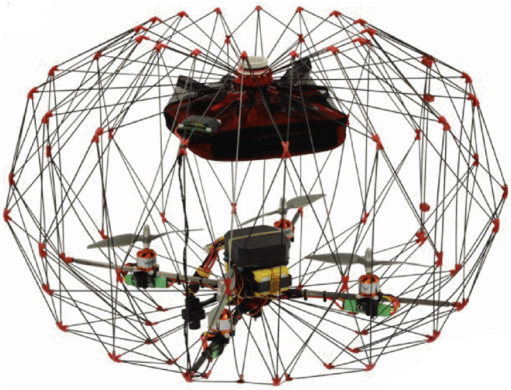


Figure 3.5: Origami-inspired cargo drone [9].

In [10], an approach has been suggested to facilitate the delivery of oversized parcels wherein the package is placed above and in close proximity to the propellers. This method also incorporates a compact rotor design to reduce storage costs and enhance portability.

This strategy takes advantage of distinct airflow dynamics, effectively eliminating slipstream drag. However, there are two primary challenges associated with placing a parcel above a quadcopter. The first pertains to determining the optimal center of gravity to ensure the drone's stability. To minimize the control effort required for the drone across a broad range of mass distributions, the center of gravity should be positioned as close as feasible to the propeller plane. The second challenge involves assessing the aerodynamic impact on the drone's lift when a payload is positioned above its propellers [10].

This method facilitates the transportation of parcels of varying sizes without constraints, unlike the scenario where a parcel is positioned beneath the frame of the quadcopter. Placing the parcel above the propellers results in a more compact footprint for the quadcopter, thereby reducing the airframe's size, weight, and drag. This, in turn, improves storage efficiency and enhances the drone's capabilities for transportation. Furthermore, a smaller drone is more practical for making deliveries in environments with obstacles or confined spaces [10].

Experimental findings indicate that positioning a parcel above the UAV at a distance equal to the radius of the operating propeller results in a 23% reduction in the drone's lifting capability. This stands in contrast to a 96% reduction observed when the parcel is located below the UAV. Moreover, increasing the distance between the quadcopter and the parcel placed above it has the potential to further mitigate the reduction in lift [10].

Figure 3.6 shows the system presented.

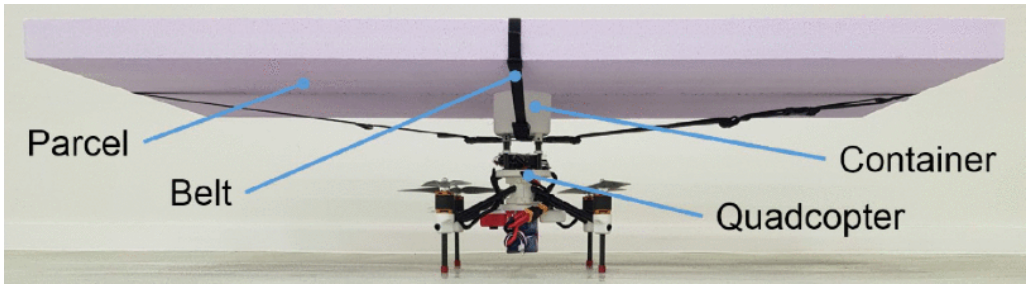


Figure 3.6: Quadcopter with the parcel placing above the rotor plane [10].

In [11], a versatile tethered aerial gripping system is introduced to enable a broader spectrum of package handling and transportation tasks through the utilization of autonomous aerial robots.

The system comprises a flexible, telescopic, rectangular frame designed to adapt to the geometry of the parcel, lifting it, and a net system employed to secure the parcel from the bottom, facilitating the implementation of caging grasps. This amalgamation delivers reliable capabilities for aerial grasping and transportation in the context of package delivery UAVs [11].

The grasping and transportation process within the proposed concept system can be delineated into three stages: i) the adaptable, telescopic frame conforms to the parcel geometry, ensuring its securement, ii) the package is elevated or tilted by the frame's lifting mechanism, exposing its lower part, and iii) the net is closed, effectively caging and securing the package for transportation [11].

A series of airborne gripping and transportation trials have empirically affirmed the system's efficacy, validating the feasibility and practicality of the proposed concept. The outcomes demonstrate that the prototype adeptly secures and transports a package box. Furthermore, the entire system can be tethered to various types of aerial robotic vehicles [11].

Figure 3.7 shows the system presented.

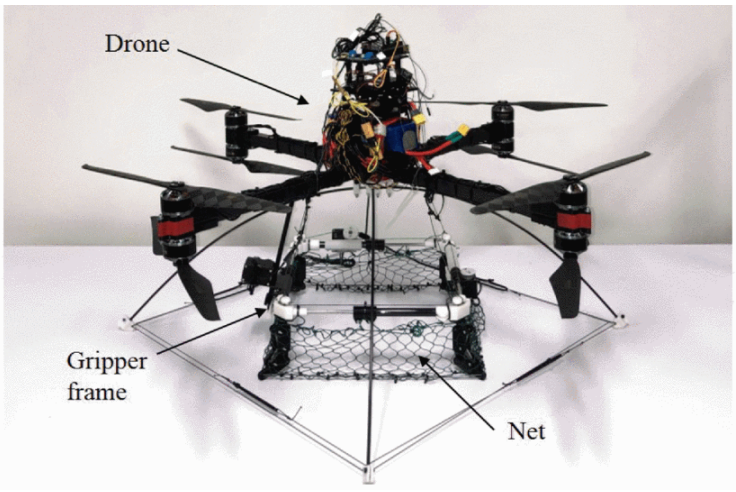


Figure 3.7: The prototype of the reconfigurable, tethered aerial grasping system is depicted attached to a co-axial octacopter [11].

3.2 Comparison between the Loading/Unloading Drone Systems

In this section, a comparative analysis of the eight distinct loading/unloading systems outlined in section 3.1 is conducted. The evaluation encompasses a systematic examination of the follow parameters:

- **Figure:** represents the image of each drone system;
- **Automated Payload Pickup:** UAV with a system capable of autonomously grasp and lift objects, without manual intervention;
- **Automated Payload Release:** UAV with a system capable of autonomously deploy objects, without manual intervention;
- **UAV needs to land:** UAV must execute a controlled landing in order to release its payload safely.

Table 3.1: Comparison between the loading/unloading systems.

Figure	Automated Payload Pickup	Automated Payload Release	UAV needs to land
3.1	No	Yes	No
3.2	No	No	No
3.3	No	Yes	No
3.4a	Yes	Yes	Yes
3.4b	Yes	Yes	Yes
3.5	No	No	Yes
3.6	No	No	Yes
3.7	Yes	Yes	Yes

All these systems are capable of transporting the payload to the designated location once it has been securely picked up.

In table 3.1, it can be observed that only three systems, specifically, the two systems depicted in figure 3.4 and the one illustrated in figure 3.7, have the capability to automatically perform both the pickup and release of a payload simultaneously. These characteristics are crucial for overcoming the problem proposed in this dissertation. Additionally, it would be advantageous if one system could execute the delivery without requiring the UAV to land, facilitating a seamless drop-and-go process and enhancing delivery efficiency.

If the selection of a drone lacking automatic pickup or release capabilities is contemplated, it becomes imperative to devise an automated mechanism for the placement or removal of payloads within the drone's payload transport system. This necessitates a thoughtful consideration of engineering solutions to facilitate seamless and efficient operations in the absence of inherent autonomous handling functionalities. The development of an automated system for the manipulation of payloads would contribute to enhancing the overall functionality and versatility of this type of drones.

3.3 Indoor Localisation Technologies

In [12], a comprehensive overview of indoor positioning systems (IPS) is provided, encompassing various technologies and methodologies. The current approaches are organised into a structured framework to facilitate the review and discussion of different methods. Additionally, a comparison of indoor positioning techniques is presented.

3.3.1 Methodologies

An IPS determines the location of a target object by analyzing observation data gathered from a network of sensing devices or sensors [13]. Generally, location estimation involves an algorithm with three stages. The first stage is evidence collection, where devices measure the characteristics of a signal. The second stage is range estimation, where devices use the obtained measurements to estimate the distance to or from the object to be located. The third stage combines these range estimates to determine the object's position [12]. In this subsection, the most common methodologies for estimating position in indoor environments will be presented.

Multilateration

Multilateration primarily employs geometry to combine range estimates from various reference devices [12]. These range estimates can be derived from different measurements:

- **Time of Arrival (ToA):** also known as Time of Flight (ToF), is the time it takes for a signal to travel from the transmitter to the receiver. By using ToA as evidence, the receiver can estimate the distance by utilising the speed of light [12].
- **Time Difference of Arrival (TDoA):** similar to ToA, uses the travel time of a signal from the transmitter to the receiver to estimate distances. However, when the exact emitting time is unknown, TDoA measures the difference in travel times between multiple receivers to estimate the distance to each receiver [12].
- **Angle of Arrival (AoA):** measures the angle at which a signal is received by a reference device [12].
- **Received Signal Strenght (RSS):** refers to the intensity of a signal at the point of reception. By measuring the RSS at the receiver, the distance can be estimated using a signal propagation model [12].

Proximity Techniques

Proximity technique determines the location of a target object based on its proximity to known objects. Proximity can be assessed through direct contact or by measuring nearness within a defined threshold distance [14].

Fingerprinting

Fingerprinting is a technique used for estimating approximate positions. The algorithm is composed of two phases: the creation of a location fingerprint database offline and online positioning. During the database establishment phase, specific points within a building are selected for sampling. A radio map of observed signals and their strengths is recorded to form position fingerprints. During online positioning, fingerprints are gathered around the target location. By employing a matching strategy, these online fingerprints are compared with those in the offline database. The position that yields the closest match is selected as the estimated location [15].

3.3.2 Indoor Positioning Systems

Indoor positioning systems utilise sensors and communication technologies to determine the location of objects within indoor settings. Indoor environments are typically filled with obstacles that can block signals between emitters and receivers. Various materials, shapes, and sizes impact signal propagation more significantly than in outdoor settings. IPS face a unique technical challenge due to the wide range of sensor technologies available, each with its own advantages and disadvantages. In this subsection, the various technologies utilised for IPS will be reviewed, based on [12].

Infrared technology

Infrared (IR) technology operates by utilising electromagnetic radiation that have wavelengths extending beyond what is visible to the human eye. It typically involves a simple setup that includes an infrared light-emitting diode, responsible for emitting signals in the form of bursts of light that are invisible to us, alongside a photodiode receiver designed to detect and collect these light pulses. Once captured, these pulses are processed to extract the encoded information [12].

Visible Light Communication

Visible Light Communication (VLC) is a data transmission technology that utilises visible light. The basis of VLC technology lies in the fact that each fixed light source flickers in a unique encoding pattern. This pattern is then picked up by a sensor that a user may have, which analyses the light's modulation and matches it with predefined encoding schemes. By identifying the predominant encoding, the technology can then link the sensor's position to the area near the specific light source [12].

Ultrasound

Location-based systems employing ultrasonics function by using sound frequencies above the audible threshold (over 20 KHz). They determine a user's location by calculating the duration it takes for an ultrasonic signal to move from a transmitter to a receiver [12].

Wi-Fi

A Wireless Local Area Network (WLAN), often called "Wi-Fi," uses electromagnetic waves to send and receive data, offering wireless network access within a specific area. To identify a user's location through WLAN technology, three predominant methods are utilised: using the propagation model from a known antenna to estimate distance to a recognised base, employing the relative signal strength from several known Wi-Fi access points to determine location via a multilateration technique, and using fingerprinting [12].

Bluetooth

Bluetooth is a wireless technology that communicates through radio frequency signals containing digitally encoded information. Following the extensive uptake of Bluetooth Low Energy (BLE), it has been contemplated for IPS owing to its accessibility, affordability, and minimal power usage, enabling stationary transmitters to operate on batteries for numerous months or even years [12].

ZigBee

ZigBee was designed specifically to meet the requirements for a low-cost, low-data-rate wireless networking solution that operates on extremely low power consumption. A ZigBee-based indoor positioning system utilises a collection of sensors and algorithms for wireless sensor networks to estimate location [12].

Radio Frequency Identification

Radio Frequency Identification (RFID) technology employs radio waves to trigger a specific circuit to emit a response that includes a distinctive identifier. This circuit can be affixed to people, animals, or items, offering a way to locate them [12].

Ultrawideband

Ultrawideband (UWB) operates by transmitting electromagnetic waves composed of extremely brief pulses across a wide bandwidth. This technology is well-suited for implementing IPS, thanks to UWB's capabilities for accurately measuring the ToF, its resistance to multipath interference, and its low power consumption, which allows for longer usage [12].

Geomagnetic

Contemporary systems frequently utilise the strength and orientation of Earth's inherent magnetic field for localization purposes. A magnetic field-based IPS employs a magnetometer to detect variations in the magnetic field, aiding in pinpointing the location of an individual or an object [12].

Inertial Technology

Inertial technology, or dead reckoning, calculates a future position from a starting point by considering speed and direction. Modern inertial systems utilise digital accelerometers and gyroscopes, often integrating data from additional sensors to enhance accuracy and performance [12].

Computer Vision

IPS leveraging computer vision utilise the data captured by cameras, alongside image processing methods, to recognise and track objects. There are two main setups for utilising computer vision in IPS: the first involves attaching the camera to the user or the item that needs tracking, allowing it to record visuals or videos from the user's perspective. In the second setup, one or more cameras are installed within the environment where the subjects are going to be observed [12].

3.4 Comparison between Indoor Positioning Systems

A comparative analysis of the various indoor positioning systems discussed in 3.3.2 is performed. The assessment includes a review of the following parameters:

- **Technology:** the type of positioning technology used in the IPS;
- **Precision:** the discrepancy between the approximated location and its true position;
- **Cost:** the expense involved in setting up the system and running it on a daily basis.

Table 3.2: Comparison between the indoor position systems [12].

Technology	Precision	Cost
Infrared	57 cm - 2.3 m	High
VLC	10 cm	High
Ultrasonic	1 cm - 2 m	High
Wi-Fi	1.5 m	Low
Bluetooth	30 cm - meters	Low
ZigBee	25 cm	High
RFID	1 - 5 m	High
UWB	15 cm	High
Geomagnetic	2 m	Low
Inertial	2 m	Low
Computer vision	1 cm - 1 m	Low

The choice of an indoor positioning system should be guided by the specific requirements of precision and cost constraints. A balanced consideration of these factors will ensure the selection of an optimal system tailored to the intended application.

In the comparison provided in 3.2, it is clear that the technologies with high precision are VLC, ultrasonic, ZigBee, and computer vision. High precision is crucial to overcome the problem in this dissertation. Among these four, only computer vision is cost-effective, making it the best choice for the localisation system that will be used to assist the drone in landing.

3.5 Localisation during Landing

In recent years, the integration of visual markers for UAV navigation has significantly advanced the precision and reliability of autonomous landing systems. Among these, ArUco markers have emerged as a particularly effective solution due to their robustness and ease of detection. This section explores various methodologies and innovations in the use of ArUco markers to enhance UAV landing accuracy.

The solution presented in [16] involves utilising mobile signs, evenly spaced along the runway, each bearing a unique ArUco marker. Details regarding the markers' IDs and their specific locations are incorporated into the UAV's navigational system. The application of a marker pose estimation algorithm enables the accurate determination of the UAV's altitude during landing. Additionally, by integrating data from inertial sensors, highly precise information regarding the UAV's approach path can be achieved.

In the paper referenced as [17], the process of landing UAVs on a designated marker involves configuring the aircraft's position in relation to this marker to enhance landing precision. This process includes the implementation of a vision-based algorithm for detecting the marker in images, determining the UAV camera's position and orientation with respect to the marker.

Initially, the algorithm establishes a link between the UAV and its camera to facilitate the streaming of video data. Following this, it attempts to detect the marker. If the marker remains undetected, the UAV shifts to a different location to continue its search. Once the marker is successfully identified, the algorithm calculates its coordinates and the UAV's altitude concerning the marker. If the altitude is below a certain threshold, an automatic landing sequence is activated, ending with the engines being powered down. If not, the UAV descends to the pre-set altitude, adjusting its position so that the ArUco-marker appears centered in the camera's field of view [17].

The experimentation revealed a landing deviation of 1.975 cm in simulations with positional adjustment and 2.12 cm in real-life scenarios. The detection of markers in real conditions was notably impeded by overly bright lighting conditions [17].

In [18], a novel kind of fiducial marker, known as embedded ArUco (e-ArUco), has been designed specifically for robust marker detection across various distances. These markers can be identified throughout the entire landing process, from both distant and close range. E-ArUco markers utilise the foundational approach of original ArUco markers, only necessitating the original ArUco detection algorithms for their operation.

The innovative e-ArUco marker design introduces an additional, centralized marker within the traditional ArUco marker, with both the central (inner) and surrounding (outer) markers having

unique IDs. An example is shown in figure 3.8. This enhanced design primarily aims at ensuring continuous and reliable detection during the entire process of drone landing [18].

The larger, outer marker facilitates rudimentary positioning of the drone, while the smaller, inner marker becomes particularly useful during the final phase of landing. At this stage, due to close proximity, the outer marker may not be fully visible in the camera's view, with parts of it extending beyond the video frame's edges. The inclusion of the inner marker guards against such instances, improving landing precision as the e-ArUco marker remains identifiable throughout the descent [18].

It is important to choose an outer ArUco marker that features a central black block, as this space is where the inner marker is positioned. Thus, the selection of the outer marker is constrained to those designs with a center black block, and it is advised to opt for an inner marker that primarily comprises black pixels to ensure higher visibility [18].

The effectiveness of the e-ArUco marker is demonstrated by its detection capability ranging from a distance of 0.2 meters up to 30 meters. Virtual trials have shown an average landing accuracy of 2.03 cm, with a standard deviation of 1.53 cm [18].

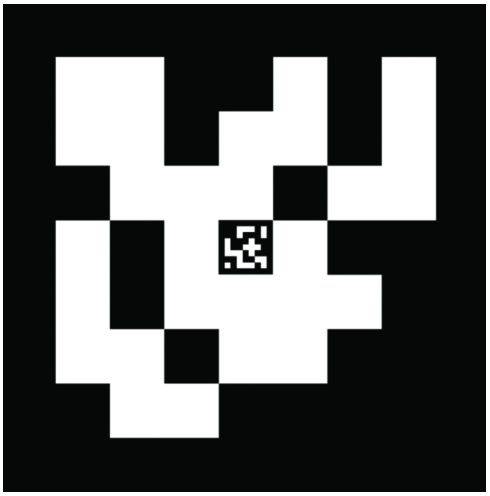


Figure 3.8: Example of an E-ArUco [18].

The utilisation of ArUco markers in UAV landing systems presents a significant advancement in achieving precise and reliable landings. The method of using mobile ArUco markers along the runway, as outlined in [16], provides effective altitude estimation and approach path correction by integrating marker pose estimation with inertial sensor data. Alternatively, the vision-based detection of ArUco markers detailed in [17] focuses on enhancing landing precision through a dynamic search and detection algorithm, though it faces challenges in overly bright conditions. The introduction of e-ArUco markers in [18] offers robust detection capabilities across a wide range of distances, maintaining accuracy even when parts of the marker extend beyond the camera's view. Comparatively, while traditional ArUco markers provide reliable performance, the e-ArUco markers significantly improve detection reliability and landing precision, showcasing their potential for superior UAV landing accuracy.

3.6 Landing Control

In [19], a method that employs a vision pipeline centered around a deep convolutional network for real-time detection of the landing mark and estimation of the system state is presented, which was only achieved by the latest advancements in object detection based on deep neural networks within the realm of Machine Learning. The control of the landing sequence is governed by a linear predictive algorithm, that estimates a series of upcoming waypoints for the UAV, effectively managing both the position and velocity of the vehicle. These trajectory points are inputted into both a low-level position controller and a velocity controller, both fine-tuned for the efficiency of the landing procedure, ensuring swift execution under a constant downward velocity. The entire system can be implemented within the onboard flight control computer of the UAV.

For the method to work, it is assumed that the UAV is within the visual range of the designated landing site, to ensure a high-confidence detection event. The comprehensive layout of the landing control is illustrated in Figure 3.9 [19].

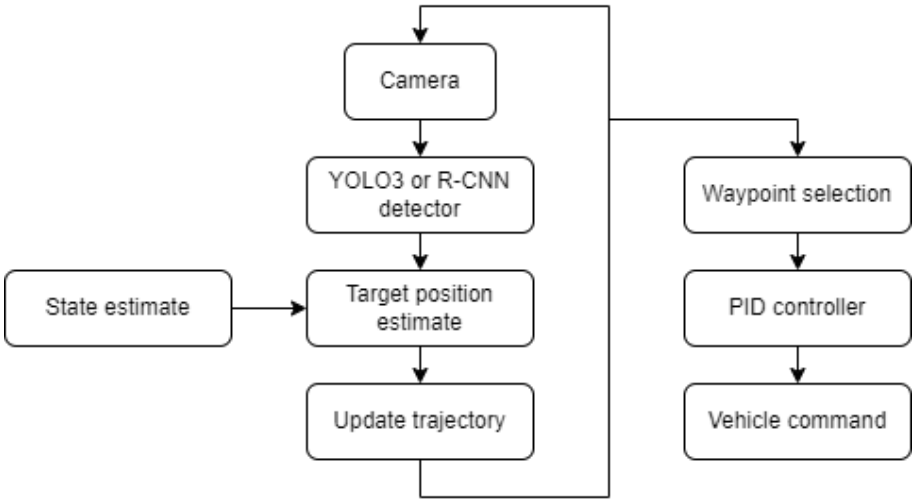


Figure 3.9: Pipeline for UAV Landing Control [19].

This method achieves an accuracy of less than 0.3 meters as the platform moves at speeds of up to 80% of the maximum UAV velocity [19].

In the next 3 subsections, the constituent parts of this process will be presented in greater detail.

3.6.1 Detection of landing marker and estimation of position

In the pursuit of landing mark detection, two types of networks were explored in [19]: You Only Look Once, Version 3 (YOLOv3) and Region-Based Convolutional Neural Network (R-CNN). In both the YOLOv3 and R-CNN architectures, the detector comprises a feature extraction network, complemented by additional layers dedicated to object detections. The feature extraction network fundamentally embodies a deep convolutional network.

The standard feature extractor layer is composed of a Two Dimensional (2D) convolution layer responsible for extracting low-level features. In the R-CNN network, the first convolution layer employs 64 filters, each with a 5x5x3 kernel. Subsequent to the convolutional layers, there are two fully connected layers and a softmax layer for classification and prediction bounding box generation. The training and evaluation algorithm makes use of the Tensorflow object detection API. The training dataset comprises a total of 70 field images captured by the UAV and 30 images generated through UAV simulation. The feature extraction network is pretrained, with only the last few layers fine-tuned during the retraining process [19].

Retraining of both YOLOv3 and R-CNN has been accomplished with notable success, attaining an accuracy exceeding 99% [19].

3.6.2 Proportional Integral landing controller

The trajectory controller follows a well-known Proportion Integral (PI) design, where the input (setpoint) of the controller consists of a discrete-time series of waypoints $s(t_i)$, delineating the desired path for the UAV. The state of the landing mark is characterised by vector $s_l = \{x, y, z, v_x, v_y, v_z\}$. This is then succeeded by the PI block [19].

For precise adjustment of the PI controller, the two UAV control modes are characterised by employing two distinct plant models. Considering a delay of τ resulting from the vision pipeline process, the inherent velocity response is approximated by an exponential decay process $f(t)$. The output $y(t)$ is essentially the integration of the actual velocity with an initial position $y(0)$. On the other hand, the vehicle's position control response is represented by a delayed exponential-approaching process [19].

3.6.3 Low-level position control

The inherent flight dynamics are regarded as a black-box plant process. In order to uphold a stable flight attitude, a low-level PID controller employs a closed-loop mechanism to continuously rectify the error function. It calculates an output control signal, to regulate the engine thrust. The error function represents the disparity between the desired attitude setpoint $r(t)$ and the estimated system state $y(t)$, typically encompassing vehicle velocity, altitude, thrust, roll, pitch, yaw, and their respective rates of change. The control signal comprises three components: proportional, integral, and derivative correction terms. Each term is scaled by a tunable gain factor to fine-tune the behavior of the control output [19].

Chapter 4

Prototype Design

In drone technology, managing payloads for various applications efficiently is crucial. The loading/unloading system is central in orchestrating the safe and precise transfer of payloads.

This chapter explores the complexities of the system, focusing on the design principles, hardware, and software intricacies. It aims to offer a thorough understanding of the system's architecture and how the components interconnect. The chapter concludes by presenting test results, assessing the system's performance, and suggesting areas for further enhancement.

4.1 Hardware Prototype

Figure 4.1 illustrates the electrical circuit responsible for the loading and unloading operations of the system.

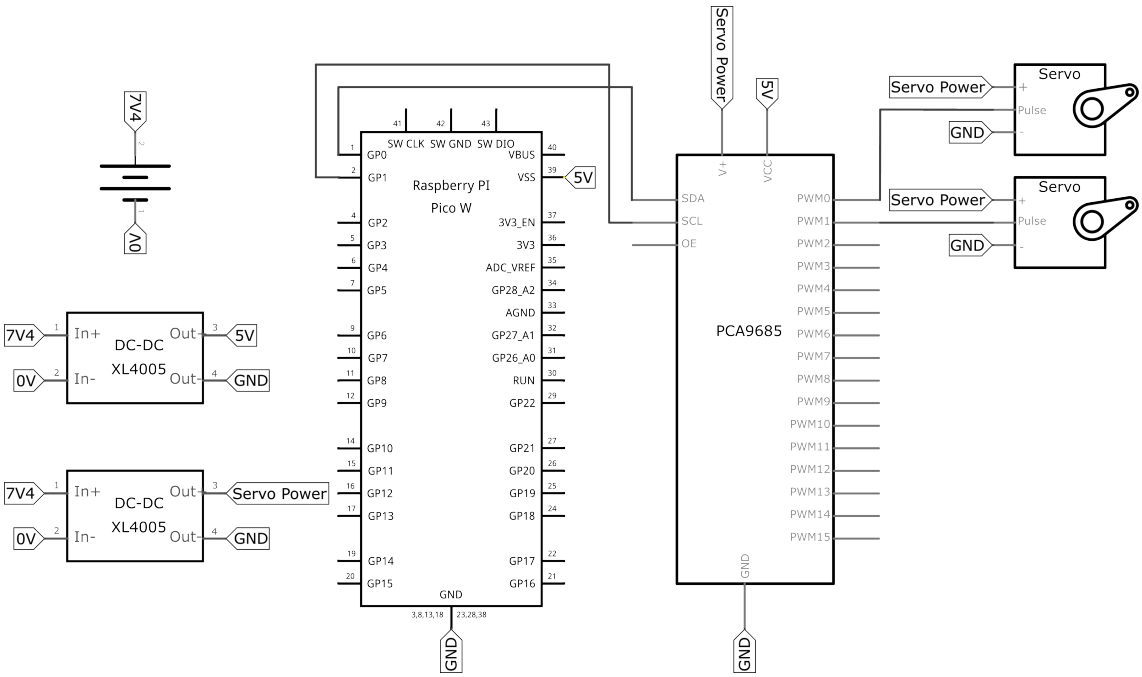


Figure 4.1: Electrical Circuit Diagram for Loading/Unloading Operations.

The components used in the construction of this circuit and the prototype are listed below:

- Raspberry Pi Pico W;
- PCA9685;
- 2 MG90S Servo Motors;
- 2 XL4005 Buck Converters;
- LipoBattery 7.4 V / 900 mAh;
- Breadboard;
- XT60 Connector;
- Jumper Wires;
- 2 618/6 2Z Bearings;
- Polylactic acid (PLA).

4.1.1 Raspberry Pi Pico W

The Raspberry Pi Pico W, an evolution of the original Pico model, integrates wireless and Bluetooth connectivity, enriching its capabilities for Internet of Things (IoT) and remote projects. Released as an extension of the Raspberry Pi Pico family, it inherits the powerful RP2040 microcontroller chip. Similar to its predecessor, it offers 30 multi-function General Purpose Input/Output (GPIO) pins, supporting diverse interfaces such as digital I/O, PWM, Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI), and Universal Asynchronous Receiver/Transmitter (UART). With its low power consumption and flexible power options, including battery operation, the Pico W remains suitable for a broad spectrum of embedded projects. Programming can be done in various languages, ensuring compatibility with different developer preferences [20]. With its compact size, low power consumption, and affordable price point, the Pico W serves as an accessible and cost-effective controller for the proposed system. Figure 4.2 shows a Raspberry Pi Pico W.

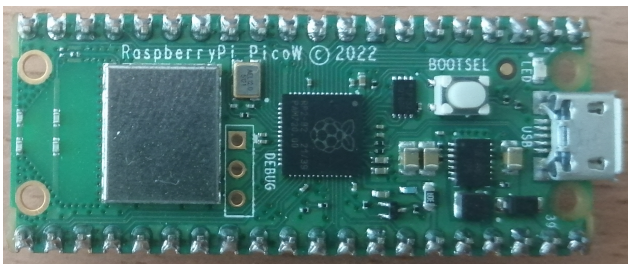


Figure 4.2: Raspberry Pi Pico W.

4.1.2 PCA9685

The PCA9685 is a 16-channel PWM controller with 12-bit resolution, facilitating precise control over connected devices. Its adjustable PWM frequencies, ranging from 24 Hz to 1526 Hz,

accommodate diverse application requirements. Utilising the I2C interface for communication, it seamlessly integrates into existing systems. Equipped open-drain outputs, it simplifies connectivity to external components such as Light Emitting Diodes (LEDs) and motors. With its low power consumption, it is well-suited for battery-powered and energy-efficient applications. The PCA9685 excels in various domains including LED lighting, motor control, display backlighting, and audio/visual effects systems. Its versatility extends to IoT devices, enabling control of actuators and servos [21]. Thanks to its sophisticated PWM control capabilities, the PCA9685 stands as a reliable choice for this system. Figure 4.3 portrays a PCA9685.

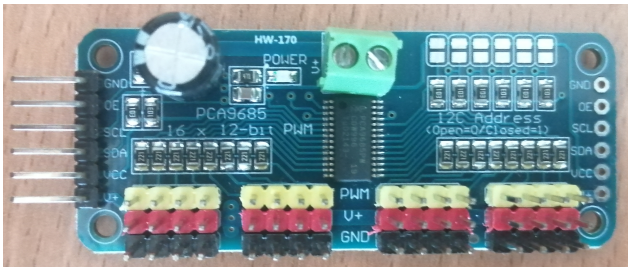


Figure 4.3: PCA9685.

4.1.3 Servo Motor

A servo motor serves as a precise actuator, capable of controlling angular or linear position, velocity, and acceleration with a high degree of accuracy. It is any motor-driven system with a feedback element built in and it comprises essential components including a DC motor, a gearbox, a controller circuit and a feedback device such as a potentiometer.

The MG90S servo motor was selected for its compact dimensions, lightweight construction, and robust output power. Notably, its incorporation of metal gears enhances strength and durability, bolstering its reliability in various applications. Figure 4.4 illustrates a MG90S [22].

Figure 4.4: MG90S Servo Motor.

Operationally, servo motors like the MG90S are controlled by Pulse Width Modulation (PWM) signals, typically delivered at a frequency of 50 Hz, equivalent to a period of 20 milliseconds. The width of the PWM signal dictates the position reference, typically spanning between one and two milliseconds, resulting in a duty cycle of approximately 5% to 10%, thereby enabling control

over a range of 0 to 180 degrees. The duration of the ON pulse defines the position of the servo, thereby determining the angle it assumes. Figure 4.5 depicts a typical PWM signal employed for controlling an MG90S servo.

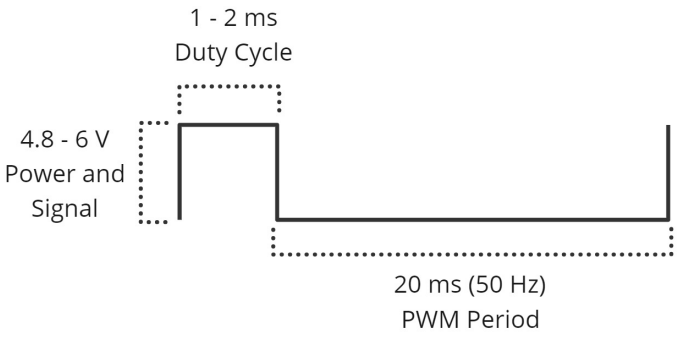


Figure 4.5: PWM Signal [22].

4.1.4 Buck Converter

A buck converter, also known as a step-down converter, is a type of DC-to-DC converter engineered to decrease voltage while concurrently increasing current from its input (supply) to its output (load). Its typical application involves reducing the voltage output from a power source, such as a battery, to a level suitable for powering electronic equipment connected to it.

The XL4005 buck converter, chosen for its precision multi-turn potentiometer for output voltage adjustment, operates within an input voltage range of 4 V to 38 V. It is imperative that the input voltage always exceeds the output voltage, which ranges from 1.25 V to 36 V. Additionally, this converter boasts a maximum current rating of 5 A and achieves an efficiency rating of up to 96% [23].

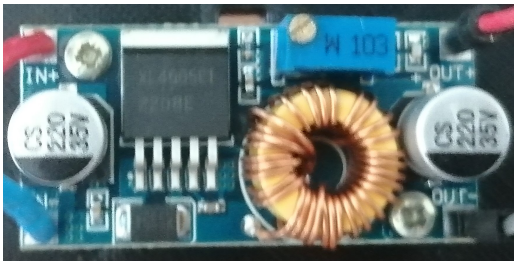


Figure 4.6: XL4005 Buck Converter.

4.1.5 Bearing

A bearing facilitates controlled motion by confining relative movement solely to the intended direction while concurrently mitigating the effects of friction between interacting components. This component was essential due to the material utilised in the construction of the system.

A 618/6 2Z bearing features an internal diameter of 6 mm, an external diameter of 13 mm, a thickness of 5 mm and a weight of 4 grams [24].

Figure 4.7: 618/6 2Z Bearing.

4.1.6 3D Parts Design

To create a system capable of loading and unloading payloads, it was crucial to design a structure responsible for holding the necessary equipment to perform these functions. Initially, mastering CAD design was essential, and Fusion 360 was chosen as the software due to its user-friendly interface and capabilities. Once the design was completed, the next step involved preparing it for printing using a *Creality Ender-3 v2* printer. This required converting the design (in STL format) into a .gcode file, which is achieved through a slicing program. Ultimaker Cura was selected as the slicing software, as it efficiently translates CAD drawings into instructions that a 3D printer can interpret.

With all the preparations in place, it was time to begin designing the structure that would be attached to the drone's legs. The final design of this structure is shown in figure 4.8, and it was manufactured using Polylactic acid (PLA) through 3D printing.

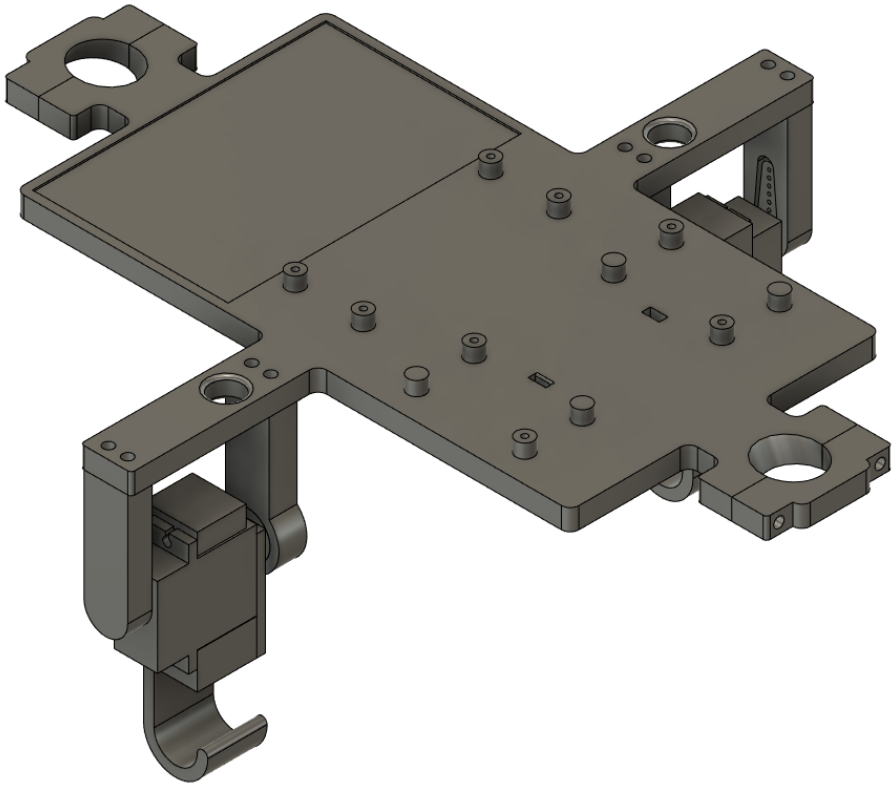
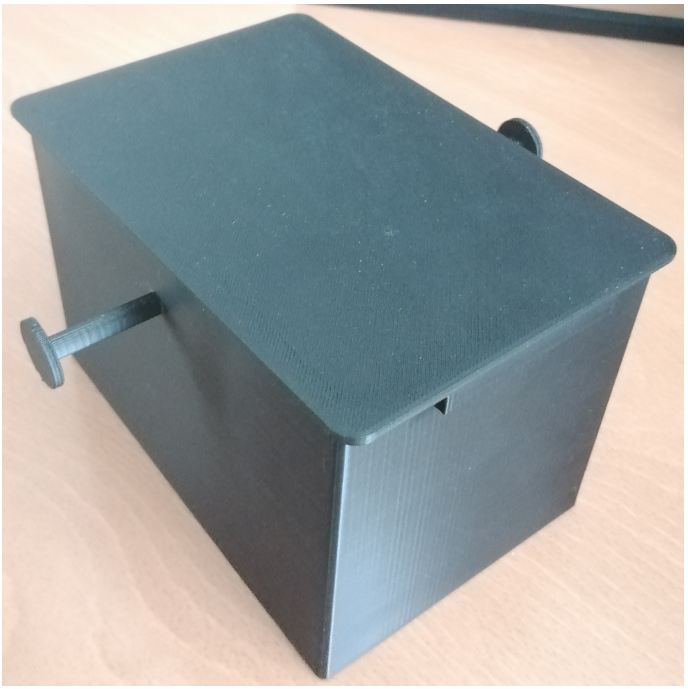
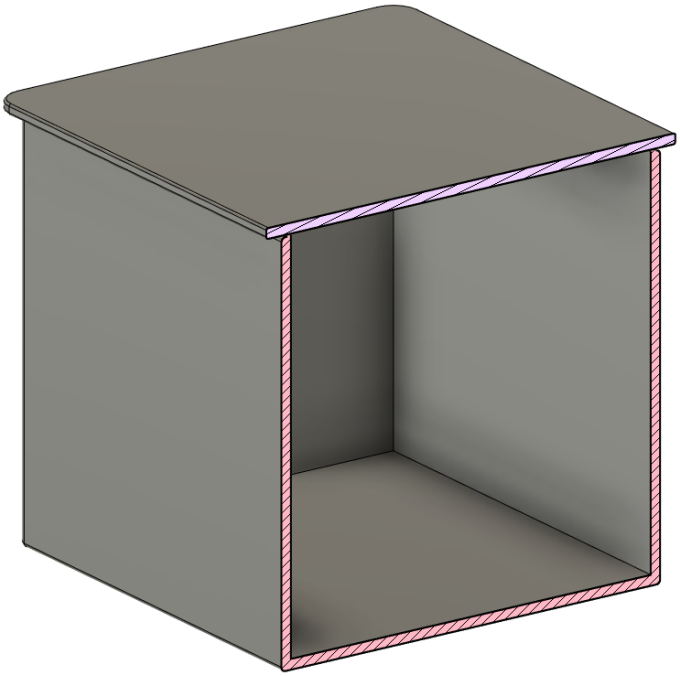


Figure 4.8: Design of the loading/unloading system.

It was also imperative to design a box for the payloads, as depicted in figure 4.9. The primary objective of this box is to render the system modular, enabling it to accommodate various types of loads while adhering to specific conditions.



(a) Design



(b) Section Analysis

Figure 4.9: Box.

After finalising the design of the entire system, it was determined that the payloads can weigh up to 1.52 kg, given the drone's capability to carry weights up to 2 kg, while considering the total weight of the entire system as 480 g. Additionally, the payloads can have dimensions of up to 145x95x97.5 mm, corresponding to the interior dimensions of the box.

4.2 Communication

This section provides a comprehensive documentation of all the connections between the components involved in the system, along with the precautions that were carefully considered.

In this system, two boards were utilised: the Raspberry Pi Pico W, which hosts the controlling code responsible for the movement involved in the load and unload process, and the PCA9685 board, which communicates via the I2C protocol to control the PWM signals sent to each servo motors.

While the use of the PCA9685 was not strictly necessary, its inclusion was justified by its ability to offload PWM processing from the Raspberry Pi Pico W. Additionally, the PCA9685 offers hardware-based PWM generation, which yields more precise and stable PWM signals compared to software-based PWM generation performed directly by the Raspberry Pi Pico's GPIO pins. This ensures enhanced control and reliability in the system's operation.

Pins 1 and 2 of the Raspberry Pi Pico W were connected to the SDA and SCL pins of the PCA9685, respectively, to adhere to the specifications of the I2C protocol.

It is imperative to ensure that both boards were powered by a voltage of 5 V, and the servo motors are supplied with a voltage ranging between 4.8 V and 6 V. Adhering to these voltage specifications is essential for the proper functioning and longevity of the components involved in the system.

Power is sourced from a LiPo battery with a voltage rating of 7.4 V and a capacity of 900 mAh. Given this voltage, it becomes necessary to reduce it to power the various components effectively. As a solution, two Buck converters were employed to step down the voltage from the input to the output.

The decision to employ two Buck converters stems from the potential for significant current draw by the servo motors, particularly when under load or during rapid movements. This high current draw can lead to voltage drops or fluctuations in the power supply, jeopardizing the stability of the control boards. One Buck converter is dedicated to powering the boards, while the other exclusively supplies power to the servo motors. Moreover, the utilisation of two distinct power sources enables independent voltage adjustment for the servo motors, ensuring that the remaining components operate within recommended values. This approach enhances flexibility and allows for precise control over the voltage supplied to the servo motors, optimizing their performance while maintaining the stability of the overall system.

In the subsequent setup, the VSYS (pin 39) of the Raspberry Pi Pico W and the VCC of the PCA9685 were connected to the positive output of the Buck converter, regulated to provide a 5 V supply. Simultaneously, the ground (GND) terminal of both boards were linked to the

negative output of the same Buck converter. Furthermore, the positive output of the secondary Buck converter was connected to the V+ terminal of the PCA9685, serving as the dedicated power supply for the servo motors. Additionally, the negative output of the secondary Buck converter was strategically linked to the ground of the remaining components, thereby establishing a common ground across the entire system.

At this stage, with the circuit appropriately powered and the boards successfully connected, the final step involves connecting the servo motors to complete the setup.

To connect the servo motors, each servo is linked to one of the PWM outputs of the PCA9685. It is crucial to ensure correct orientation during connection, aligning the servo's PWM with the board's PWM, and similarly, connecting V+ and GND appropriately.

After completing the assembly of all system components, the result depicted in figure 4.10 was obtained.

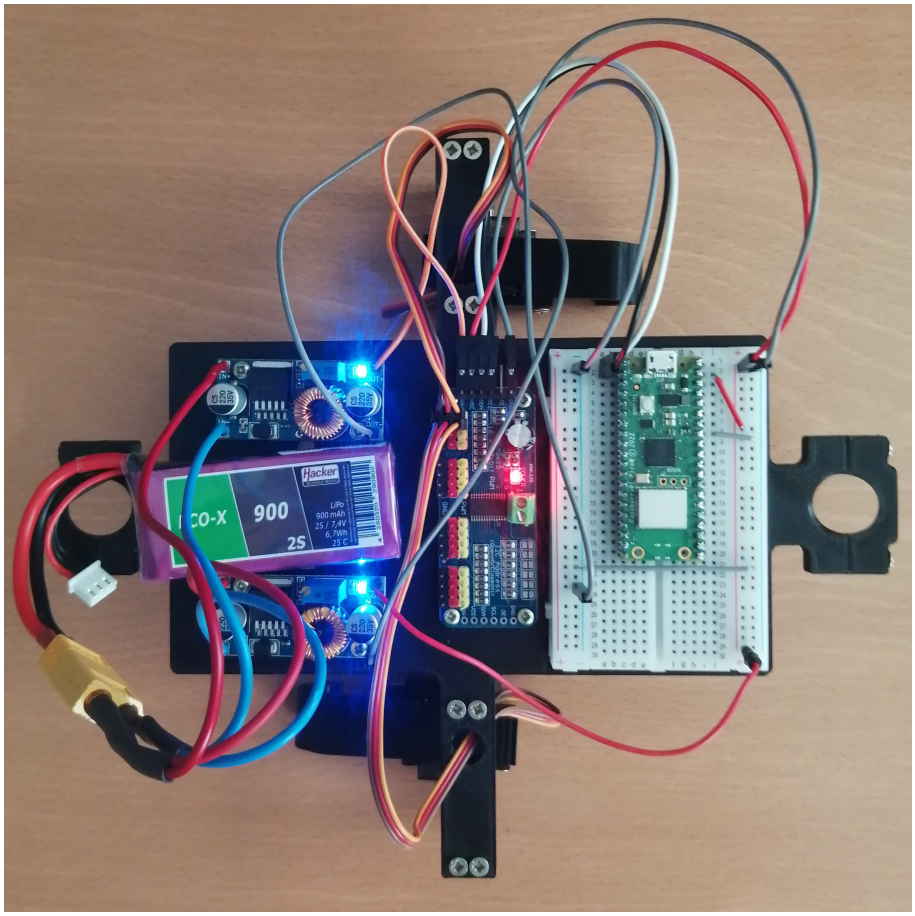


Figure 4.10: Assembly of all electronic components.

4.3 Software Development

This section provides a comprehensive documentation of all the logic and processing of the loading/unloading system.

4.3.1 External Libraries

At the onset, the project incorporates essential libraries required for its execution.

```
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include "WiFi.h"
```

Listing 4.1: Libraries utilised

The first library provides basic functionalities, sets up the programming environment and facilitates debugging of the implemented code. The second library facilitates I2C communication between the microcontroller and the Adafruit PWM Servo Driver. The third library controls the PWM signals to the servos, allowing precise positioning. The remaining library enables WiFi capabilities, sets up a WiFi access point, and handles HTTP requests to control the system through a web interface.

4.3.2 Code Structure

To adhere to best programming practices, two structs were created: one for controlling the angle of each servo motor and another for the state machine.

```
typedef struct
{
  int num;
  int pwm_90, pwm_180;
  int current_pwm;
} servo;
```

Listing 4.2: Servo Motor struct

The 'servo' struct comprises four variables. The 'num' variable denotes the pin number to which the servo is connected on the PCA9685 board, while 'pwm_90' and 'pwm_180' indicate the PWM values corresponding to angles of 90° and 180°, respectively. These angles are initially calibrated through trial and error in order to enable the use of the 'current_pwm' variable, which enables positioning the servo at a desired angle via PWM manipulation, utilising the function provided below.

```
void pwm_calculation(servo& servo, int angle)
{
  servo.current_pwm = servo.pwm_90 + (angle-90) * (servo.pwm_180-servo.pwm_90)/90;
}
```

Listing 4.3: Function that calculates PWM depending on a given angle

```
typedef struct
{
  int state, new_state;
  unsigned long tes, tis;
} fsm;
```

Listing 4.4: State Machine struct

The 'fsm' struct consists of four variables. The 'state' and 'new_state' variables denote the current and next states of the state machine, respectively. The 'tes' variable represents the time of entry into the state, while the 'tis' variable denotes the time spent in the state. The ensuing code snippet elucidates the process by which state transitions occur.

```
void set_state(fsm& fsm, int new_state)
{
  if (fsm.state != new_state)
  {
    fsm.state = new_state;
    fsm.tes = millis();
    fsm.tis = 0;
  }
}
```

Listing 4.5: Function that updates the states

4.3.3 Implementation

After incorporating the aforementioned libraries into the program and creating the structs for the servo motors and state machine, the initial step involved the creation of two 'servo' structs, each corresponding to a servo motor, alongside a 'fsm' struct for managing the loading and unloading operations of the system. Furthermore, it was initialised two variables, 'load' and 'unload', to determine the sequence for loading and unloading the payload for the drone.

Then, within the 'void setup()' function, all structures were initialised:

- Upon connecting the servos to the PCA9685 board and calibrating them for the angles of 90° and 180°, each servo was initialised with its corresponding values.
- The SDA and SCL pins for the servo motors were explicitly defined.
- The PCA9685 was initialised.
- The initial state of the state machine was specified.

Subsequently, within the 'void loop()' function, which operates indefinitely until the Raspberry Pi Pico is powered off, with a period of 10 milliseconds, the state machine was implemented using the following logic:

- The duration of time spent by the machine in a specific state is updated.
- The input variable (which determines the payload loading and unloading sequence) is read.
- The program verifies whether a state transition occurred.
- If a transition occurred, the state of the machine is updated accordingly.
- Actions are determined based on to the current state of the state machine.
- The outputs (servos) are activated.

4.3.4 State Machine

The state machine employed in the loading/unloading system is shown in figure 4.11. The variables 'load' and 'unload' assume the pivotal role in orchestrating transitions within the system for the loading and unloading processes, respectively.

When the system's state changes from unload to load or vice versa, the angle of each servo motor adjusts at a rate of 1 degree per 10 milliseconds. This adjustment continues until the servo motors reach the specific angles required to either load (90 degrees) or unload (180 degrees) the payload. The 'servoAngle' variable represents the angles at which the servos are positioned and the 'angle' variable is used to set a specific angle for the servos when transitioning between loading and unloading positions.

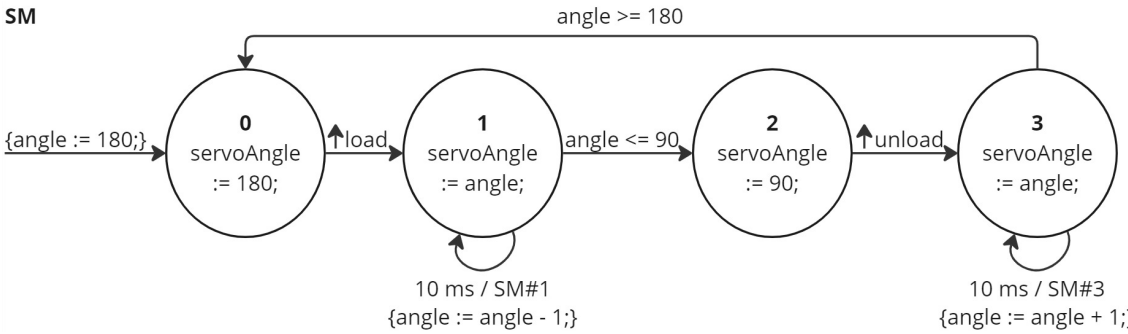


Figure 4.11: State Machine.

4.3.5 Web server

In order to test the system's functionality independently, a custom module was developed to handle WiFi setup and communication for the Raspberry Pi Pico W, simulating the input variable. Leveraging the wireless's microcontroller capability, a simple web server was established to control the system.

The microcontroller creates a WiFi access point, hosts a web page, and toggles the system's state based on HTTP requests.

The first step involved creating a 'void' function called 'setupWifi' to set up a WiFi access point (AP) on the microcontroller, requiring a Service Set Identifier (SSID) and a password. The AP was configured with specific IP address, gateway, and subnet mask settings. Subsequently,

a web server was initiated on port 80 (HTTP), allowing the microcontroller to handle incoming HTTP requests. This setup function was initialised within the 'void setup()' function.

Next, a 'String' method called 'updateWebPage' was developed to handle incoming HTTP requests from clients, specifically web browsers. This method listens for incoming connections, reads the HTTP request, and processes it to determine the requested action (loading or unloading). Based on the request, it updates the 'currentAction' variable of the system, enabling the user to view the current state on the web browser. It also constructs an HTML response to send back to the client, updating the input variable of the main program. The HTML response contains buttons that allow the user to switch between loading and unloading actions, reflecting the intended state changes for the servos controlled by microcontroller. This method is invoked within the 'loop()' function and the web page is depicted in 4.12.

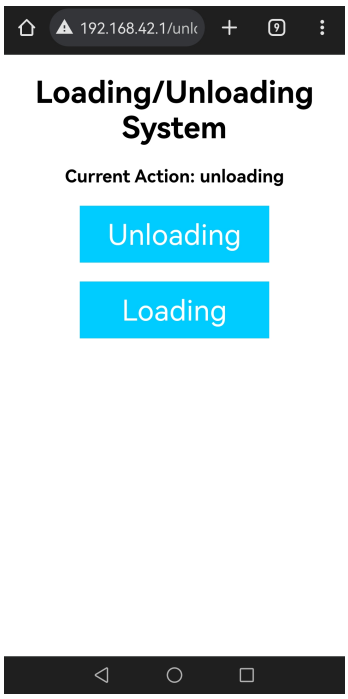


Figure 4.12: Web page used for testing.

The 'Loading' and 'Unloading' buttons simulate the signals received from the drone indicating that it has landed. Each one identifies if the drone has to pick up or release the payload, respectively.

The state machine in the main program monitors the current state and transitions based on the 'currentAction' received from 'updateWebPage', setting specific servo positions via PWM calculations.

This module allows for the management of the interaction between a web interface, accessed through a browser, and the physical control of servos based on user actions. It provides a user-friendly way to toggle the system's state and view its current status through a simple web page.

4.4 Results

Once the system was fully attached to the drone's legs and configured, testing commenced. Last-minute inspections were conducted to ensure that all cables were securely fastened, every screw was correctly tightened, the battery was fully charged, and the servos were accurately calibrated.

The testing was conducted independently of the drone's flight systems, solely using the web interface described in 4.3.5. Access to this web page was achieved via a phone connected to the wireless network provided by the Raspberry Pi Pico W.

The testing procedure began at the unloading position, starting logically without any payload. After establishing the connection and positioning the drone at the correct pickup location, the 'Loading' button on the web page was pressed to load the payload. The drone was then manually lifted to simulate flight, ensuring the payload remained securely attached. Upon "landing," the 'Unloading' button was pressed, and the payload was released.

The initial test was unsuccessful due to uncontrolled servo velocity, resulting in the improper attachment of the payload. Identifying this issue led to regulating the servo speed to rotate at 1 degree per 10 milliseconds. This adjustment enabled the system to successfully pick up and release the payload.

A remaining issue was the payload's rotation relative to the drone's orientation during pickup, potentially compromising flight stability.

Future work could involve adding two servo motors perpendicular to the existing ones and incorporating two additional magpies to the box, improving payload stability. To further optimize performance, lightweight and compact materials such as carbon fiber should be considered in the system's design. Additionally, further testing in varied conditions could provide insights into other potential improvements and robustness of the system.

Chapter 5

Localisation Algorithm during Landing

Autonomously landing a drone with precision and control is a complex challenge that relies on accurate localisation and pose estimation. This chapter focuses on the key components of a localisation algorithm designed to support precise landing operations. The algorithm's core is based on computer vision techniques, particularly camera calibration and pose estimation using fiducial markers. The chapter also discusses the results of testing the localisation algorithm and identifies potential areas for future improvements.

5.1 Camera Calibration

In section 2.2, it was noted that calibrating a camera is necessary due to the unique characteristics of each individual camera. Even cameras of the same model have slight variations in focus and lens alignment, making calibration essential for accurate results.

The calibration process was based on the OpenCV documentation, as referenced in [25].

OpenCV employs a pinhole model for cameras. As explained previously, the pinhole model mathematically represents the projection of a 3D object into a 2D space. When a 3D object is projected onto a 2D plane, it loses its depth perception due to the reduction of one dimension.

Certain pinhole cameras can cause substantial distortion in images. There are primarily two types of distortion involved: radial distortion and tangential distortion. The distortion coefficients are described using 5 parameters:

$$Distortion_{Coefficients} = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} [25]$$

Here, k_1, k_2, and k_3 represent the radial distortion parameters, while p_1 and p_2 represent the tangential distortion parameters.

Moreover, the intrinsic parameters of the camera are required, which are summarised in the camera matrix. The camera matrix is specific to each camera, meaning it can be applied to other images captured with the same camera once it's determined. This matrix is represented by a 3x3 matrix configuration [25].

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \text{[25]}$$

In this matrix, f_x and f_y denote the focal lengths, while c_x and c_y represent the optical center of the sensor.

With the five parameters of lens distortion and the four parameters of the intrinsic camera matrix, a total of nine parameters need to be estimated. The process of estimating these nine parameters is known as Camera Calibration.

To find these parameters, capturing several images of a clearly defined pattern, such as a chessboard, is necessary. By identifying specific points whose relative positions are already known (for instance, the corners of the squares on the chessboard), we can use their real-world coordinates and coordinates in the images to calculate the distortion coefficients. A minimum of 10 test patterns should be used to achieve more accurate results [25].

5.1.1 Software

Open Source Computer Vision Library (OpenCV) is an open-source software library designed for computer vision and machine learning applications. It includes over 2,500 optimized algorithms for image processing, video analysis, feature detection, object detection, machine learning, and 3D reconstruction. Key image processing features include resizing, blurring, edge detection, and color space conversions. Video analysis capabilities encompass frame capture, motion detection, and object tracking. OpenCV supports feature detection algorithms, and it enables object detection using models and deep learning frameworks. It also provides machine learning tools for training and prediction with custom datasets and facilitates 3D model construction from 2D images. Widely used in face recognition, augmented reality, robotics, medical imaging, and autonomous driving, OpenCV supports multiple platforms and languages. Its extensive functionality and ease of use make it a popular choice in academia and industry [26].

5.1.2 Save Snapshots script

This Python script uses OpenCV to capture and save snapshots from a video feed. It initialises the camera and sets custom frame dimensions if provided. The main function parses command-line arguments for width and height, setting defaults if none are given. It captures frames in a loop and displays them in a window. Users can press the spacebar to save the current frame as an image or 'q' to exit. The script ensures that the specified save directory exists, creates it if necessary, and sequentially names the snapshots. The program concludes by releasing the camera and closing all OpenCV windows. This setup is useful for tasks requiring periodic image capture from a camera. In this case, it was used to capture the chess board placed at different locations and orientations.

5.1.3 Camera Calibration script

This Python script performs camera calibration using images of a chessboard pattern. The script is based on [25] and utilises images captured by the previous script. It reads images from the specified folder and detects chessboard corners in each image. If corners are found, their positions are refined. A new feature has been added to allow the ability to check each picture and decide whether to keep or discard it, which enhances the accuracy of the calibration. Users can accept or skip images by pressing a key. Once at least 10 good images have been collected, the script calculates the camera's intrinsic parameters and distortion coefficients using these corner points. The calibration results are saved to text files and used later when needed for camera work. Additionally, the script computes the mean reprojection error to assess the accuracy of the calibration. Customization of the chessboard's dimensions and image settings can be done using command-line arguments.

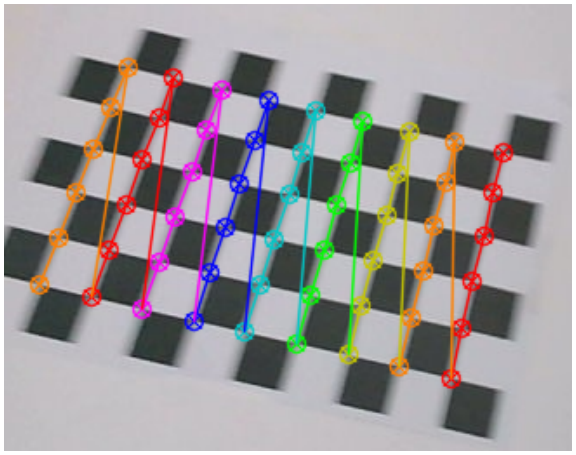


Figure 5.1: Discarded image during Camera Calibration.

5.2 Pose estimation

Pose estimation is crucial in various computer vision tasks, including robot navigation and augmented reality. This procedure involves identifying matches between points in the physical environment and their 2D projections in images. This step often presents challenges, leading to the frequent use of synthetic or fiducial markers to simplify the process [27].

A widely adopted method employs binary square fiducial markers. The primary advantage of such markers is that a single marker, through its four corners, provides sufficient correspondences to determine the camera's position. Moreover, the internal binary coding enhances their robustness by facilitating the application of error detection and correction strategies [27].

An ArUco marker is characterised by its distinctive square shape, featuring a broad black border surrounding a central binary matrix that defines its unique identifier (ID). The black border facilitates rapid recognition in images, while the binary encoding allows for both identification

and the implementation of error detection and correction methods. The dimensions of the marker dictate the size of the inner binary matrix [27].

The ArUco library allows for the determination of the camera position and rotation from a single marker, regardless of its orientation in the environment.

5.2.1 Algorithm

This Python script is designed for real-time ArUco marker detection using a webcam, to calculate and display the camera's position and orientation relative to a specific marker. It starts by importing necessary libraries and defining functions to validate rotation matrices and convert them to Euler angles.

The script specifies the ID and physical size of the marker and defines its 3D corner coordinates. The ArUco marker was created using the tool referenced in [28]. The chosen ArUco marker has an ID of 100 and belongs to the original ArUco dictionary.

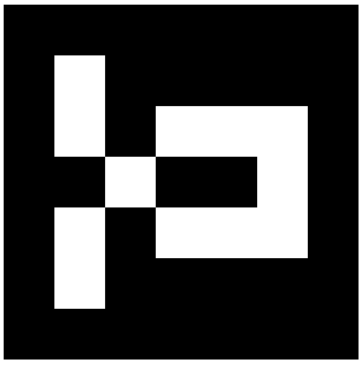


Figure 5.2: Original ArUco with ID = 100 [28].

The script also loads camera calibration data, including the intrinsic matrix and distortion coefficients, from text files. Additionally, it sets up a rotation matrix for axis flipping.

Using predefined parameters and a specific dictionary, the script initialises the ArUco marker detector. It then opens the webcam, sets the frame resolution according to the calibration, and enters the main loop.

Within this loop, the script captures video frames, converts them to grayscale, and detects ArUco markers. When the specified marker ID is detected, the script calculates the camera's position and orientation relative to the marker and vice-versa.

The loop continues until the 'q' key is pressed, at which point the webcam is released, and all OpenCV windows are closed. This script provides real-time feedback on the camera's position and orientation relative to the detected marker, making it useful for applications requiring precise localisation and orientation tracking.

5.2.1.1 Rotation matrix

The initial rotation matrix in the script accounts for the differences between the camera's reference frame and the ArUco marker's reference frame.

The camera's reference frame is centered at the camera's optical center, with the z-axis pointing outwards, the x-axis going to the right, and the y-axis going downwards. On the other hand, the ArUco reference frame is centered in the middle of the ArUco marker, with the z-axis pointing outwards, the x-axis going to the right, and the y-axis going upwards. When the camera and the ArUco marker are facing each other, their reference frames are inverted relative to one another. Specifically, the y and z axes of the camera and marker are opposite in direction.

To accurately determine the rotations and align the reference frames, a rotation matrix around the x-axis is necessary. This matrix accounts for the inversion of the y and z axes. The required rotation matrix R_x to flip the reference frames around the x-axis is:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

This transformation ensures that the coordinate systems of the camera and the ArUco marker are consistent, allowing for accurate pose estimation and reliable interpretation of the camera's position and orientation in the marker's view.

5.2.2 Functionality

To evaluate the cohesion of the algorithm, the initial step involved drawing the detected marker and subsequently drawing the marker's axes. Following this, text was overlaid on the image to provide information about the camera's position relative to the marker frame and the orientation in terms of Euler angles, as depicted in figure 5.3. This information is crucial as it enables the estimation of the drone's pose based on the marker.

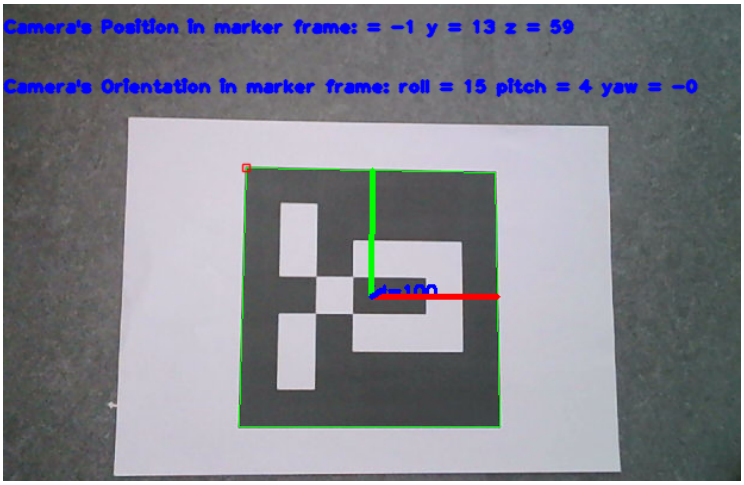


Figure 5.3: Camera pose relative to the marker frame.

5.3 Results

In order to test the functionality of the scripts, a camera was connected to the computer. The initial objective was to verify if the camera's position and orientation in relation to the marker frame were accurate while the camera was in motion with the marker stationary. The test confirmed that the camera moved correctly along the three axes (x, y, z) and rotated appropriately around them (roll, pitch, yaw). This indicated that the transformation from the camera to the marker frame was correct.

Subsequently, the OptiTrack system was employed to evaluate the precision of the camera's positional measurements.

OptiTrack is a widely-used motion capture technology known for its high precision in tracking and recording real-time movement. It operates through a network of infrared cameras and reflective markers, which are attached to objects or subjects of interest. These cameras detect the marker positions, and the system's software processes the data to create accurate 3D representations of motion. By triangulating the markers from different angles, OptiTrack delivers detailed movement data, often achieving sub-millimeter accuracy with errors as low as 0.2 to 0.5 mm. Its exceptional precision and low-latency performance make it valuable for various applications, particularly those requiring high accuracy [29].

The first step in using the OptiTrack system was calibrating its cameras via the Motive software. Initially, all physical elements that could interfere with the cameras were removed from the environment. Once the setup was prepared, the software masked out any remaining light sources. The calibration process was then initiated by selecting the "start wanding" option in Motive. The software automatically began recording samples as soon as it detected the Calibration Wand, which contains three reflective markers.

The main goal during the wanding process was to maximize coverage within both the 2D camera views and the desired physical capture volume. Once the entire capture volume and the 2D camera views were adequately covered, the calibration calculation could proceed. This process continued until the calibration engine converged to a solution. Upon completion of the calibration, the ground plane was aligned with the ArUco marker, with its origin set to the midpoint of the marker, using the OptiTrack Calibration Square. This specific point is also utilised by the algorithm to determine the pose of the ArUco marker.

Since it is not feasible to place a reflective marker directly at the optical center of the camera, a reflective marker was instead attached to the back of the camera to obtain measurements from the OptiTrack system. However, because this position does not represent the true optical center, an additional reflective marker was temporarily attached to the optical center of the ArUco marker. This step was necessary to calculate the transformation between the marker positioned behind the camera and the camera's optical center. This approach ensured that the measurements obtained from the camera and the OptiTrack system were consistent.

Once the OptiTrack system was properly configured and the camera was mounted on a tripod,

30 position measurements were recorded using both systems. After each set of three measurements, the camera was moved further away from the ArUco marker. During each set, the camera tripod was repositioned horizontally at the same height to obtain x and y coordinate measurements as a function of the same z-coordinate. The positional measurements obtained from both the OptiTrack system and the developed algorithm are presented in tables 5.1 and 5.2.

Table 5.1: Positions obtained from the OptiTrack system in centimeters.

x	y	z
0.5	0.9	31.7
-11.1	4.9	31.8
2.1	-2.3	31.7
-2.6	1.1	37.1
-10.5	5.6	37.1
4.9	-1.0	36.9
-0.1	1.6	42.3
-8.4	7.3	42.3
7.3	6.7	42.2
-2.6	-4.5	51.7
8.2	-7.9	51.5
-8.3	10.2	51.6
-10.0	4.9	59.6
-16.7	-8.1	59.7
12.6	13.4	59.3
-4.6	2.7	74.7
18.0	-14.4	74.7
-31.1	16.4	74.6
-9.5	2.9	82.2
16.6	19.2	81.8
-30.8	-13.1	82.4
-0.8	-2.7	91.3
15.9	2.0	91.2
-9.1	0.7	91.4
-5.5	-0.8	100.3
14.0	-9.5	100.2
-15.9	-12.1	100.5
-6.1	-4.6	106.9
14.5	-12.9	106.9
-11.9	4.5	106.9

Table 5.2: Positions obtained from the developed algorithm in centimeters.

x	y	z
4	4	34
-9	8	35
5	1	34
0	5	40
-8	10	40
8	3	40
3	6	45
-6	12	46
10	11	45
1	1	56
12	-2	56
-5	16	55
-7	12	65
-13	-2	66
16	20	64
-1	8	81
22	-6	82
-27	26	82
-3	12	90
23	28	88
-23	-6	92
-3	7	100
15	12	100
-11	11	100
-7	11	110
12	2	111
-17	-1	110
-8	8	117
13	0	119
-14	17	117

This approach is beneficial for testing a localisation system because it allows for controlled and consistent measurements across varying distances, providing a thorough evaluation of the system's accuracy in detecting positional changes. By maintaining the z-coordinate constant and varying only the x and y coordinates, the system's ability to accurately measure lateral movement is tested. Additionally, moving the camera progressively further away evaluates the system's performance at different depths, helping to identify potential inaccuracies over increasing distances. This method ensures comprehensive testing of both lateral and depth localization capabilities.

The two metrics used to assess the accuracy of the algorithm were the absolute and relative errors. Absolute errors represent the actual differences between the measured values (obtained from the developed algorithm) and the true values (obtained from the OptiTrack system), while relative errors are the absolute errors expressed as a percentage of the true value. Additionally, the error in the distance from the z-axis to the measured position, d, was computed using the obtained measurements, providing an estimate of how far it might deviate laterally from its intended landing point.

The mean of the z-values from the three measurements taken at each height from the ArUco marker, obtained from the OptiTrack system, was calculated since all measurements of the same set were based on the same height. The same approach was applied to the absolute and relative errors to express them as a function of the mean z-values.

Table 5.3: Mean of the z-values obtained from the OptiTrack system at the same height in centimeters.

z
31.7
37.0
42.3
51.6
59.6
74.7
82.2
91.3
100.3
106.9

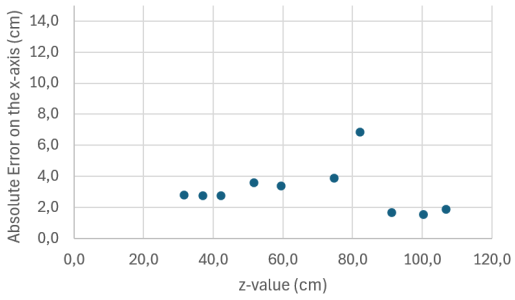
Table 5.4: Mean of the absolute errors of the developed algorithm at the same height in centimeters.

Absolute Error			
x	y	z	d
2.8	3.2	2.6	4.3
2.8	4.1	3.0	4.9
2.7	4.5	3.4	5.3
3.6	5.7	4.1	6.8
3.4	6.6	5.4	7.4
3.9	7.8	7.0	8.7
6.9	8.3	7.8	10.8
1.7	10.0	8.7	10.1
1.5	11.5	10.0	11.6
1.9	12.7	10.8	12.8

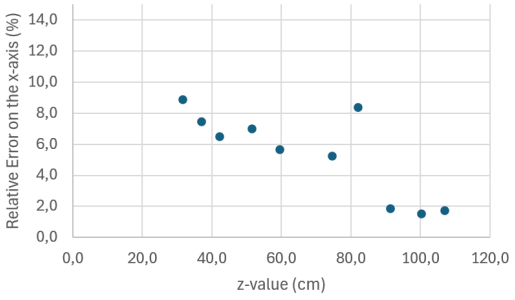
Table 5.5: Mean of the relative errors of the developed algorithm at the same height in percentage.

Relative Error			
x	y	z	d
8.9	10.0	8.3	13.4
7.4	11.0	8.1	13.3
6.5	10.6	8.0	12.4
7.0	11.1	7.9	13.1
5.7	11.1	9.1	12.4
5.2	10.4	9.4	11.7
8.4	10.1	9.6	13.2
1.8	10.9	9.5	11.1
1.5	11.4	10.0	11.5
1.7	11.9	10.1	12.0

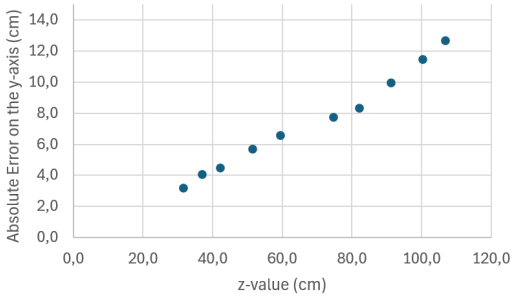
With these values, it was possible to create graphics that allow for a better understanding of the results.



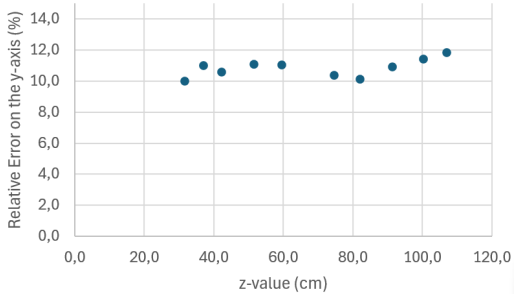
(a) Absolute Error on the x-axis



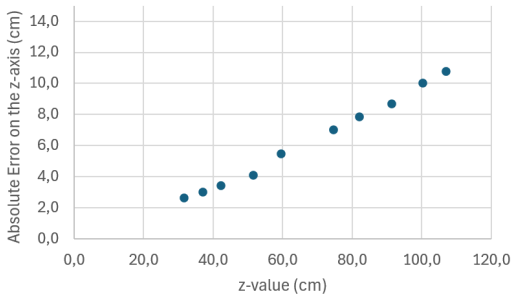
(b) Relative Error on the x-axis



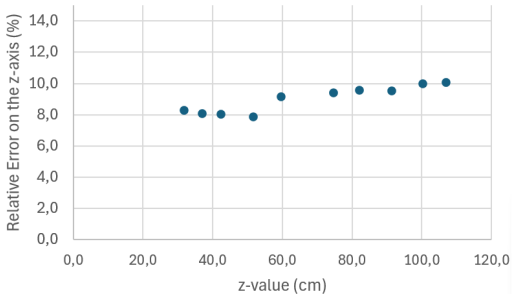
(c) Absolute Error on the y-axis



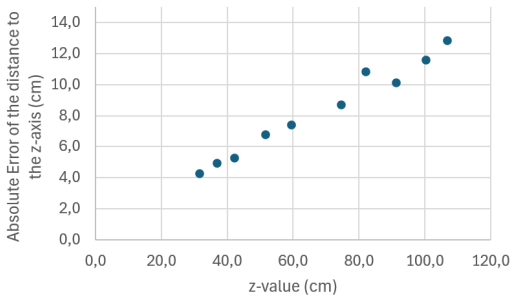
(d) Relative Error on the y-axis



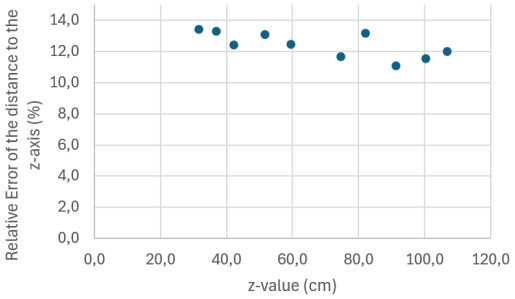
(e) Absolute Error on the z-axis



(f) Relative Error on the z-axis



(g) Absolute Error of the distance to the z-axis



(h) Relative Error of the distance to the z-axis

Figure 5.4: Errors as a function of height.

In the analysis of the graphics, the results indicate that absolute errors increase linearly with distance. This linear growth suggests that as the distance to the marker increases, measurement inaccuracies become more pronounced due to factors such as camera resolution and focus.

In contrast, relative errors remain nearly constant, indicating that the ratio of error to actual

distance stays stable. This consistency suggests that the localisation algorithm performs reliably across varying distances.

Overall, the linear increase in absolute errors, coupled with constant relative errors, highlights systematic measurement inaccuracies with distance while demonstrating the robustness of the employed localisation algorithm.

However, the errors on the x-axis were unusual, as they were anticipated to behave similarly to those on the other axes. Despite this anomaly, the x-axis demonstrated the best accuracy across all axes from the ArUco marker.

Regarding the values, at distances close to the marker, the absolute errors were 2.8 cm on the x-axis, 3.2 cm on the y-axis, and 2.6 cm on the z-axis, with all relative errors being less than 10%. Beyond one meter from the marker, the accuracy varied, with the absolute error on the y-axis increasing to 12.7 cm and the z-axis increasing to 10.8 cm. In contrast, the error on the x-axis decreased to 1.9 cm.

However, this work fell short of expectations. Further testing is required to identify errors in degrees around the three axes. This can be accomplished by using precise measurement tools such as gyroscopes for angular measurement.

As future work, a program should be developed to connect with the drone, call the function to detect the marker while it is landing, and estimate its pose. This will involve:

- Utilising a Raspberry Pi as the controller.
- Developing a connection script in Python to interface with the drone's Pixhawk via the MAVLink protocol [30].
- Ensuring real-time communication and data processing capabilities.

To establish a connection with the drone, the Raspberry Pi could be connected to the drone's Pixhawk via the telemetry port, as shown in figure 5.5.

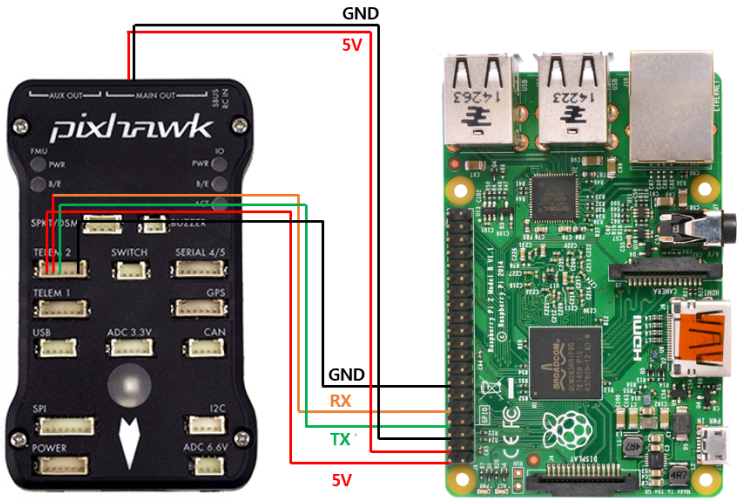


Figure 5.5: Connection Pinout between Pixhawk's telemetry port and Raspberry Pi [31].

Subsequently, the algorithm described in section Pose estimation should be converted into a library. This involves creating a class with arguments for the ID of the marker to be found and its size, resulting in the determination of the pose of the marker in the camera frame.

Since the software for controlling the drone's trajectory operates with geographic coordinates (latitude and longitude), the following steps should be taken:

1. Use the developed library to detect the marker and determine its pose within the camera's field of view.
2. Convert this position from the camera's perspective to the drone's body perspective.
3. Transform the marker's position from the drone's body frame to the north-east frame, centered on the drone's center of gravity.
4. Calculate the marker's geographic coordinates by adding its north-east position to the UAV's current location.

Finally, a camera needs to be connected to the Raspberry Pi so that the controller can run the script to determine the landing location for the drone. The integration should be tested through a series of controlled landing experiments to validate the accuracy and reliability of the system.

In conclusion, while the initial tests confirmed the correct transformation between the camera and marker frames, further detailed measurements and testing are essential. The proposed future work aims to integrate these findings into a practical system for drone landing, leveraging precise tools and robust programming to ensure accuracy and functionality.

Chapter 6

Conclusion and Future Work

This dissertation has explored the development of a loading/unloading system for drones, along with a localisation method for precise drone landings. The development and integration of these systems aim to address the growing demand for efficient and precise payload management in drone technology, which is crucial for a wide range of real-world applications.

The development and testing of the drone's loading/unloading system have demonstrated the feasibility and practicality of using a Raspberry Pi Pico W, PCA9685, and MG90S servo motors for precise payload management. This project not only showcased the integration of hardware components but also emphasised the importance of software control through a state machine and web-based interface.

Despite the successful implementation, the project faced challenges, particularly concerning the control of servo speeds and the stability of the payload during drone movement. Initial tests revealed the critical nature of precise servo control, which was addressed by adjusting the speed of the servos. However, issues such as payload rotation relative to the drone's orientation indicate areas needing further refinement. The findings underscore the necessity for additional stabilisation mechanisms to ensure consistent and reliable performance, especially under dynamic conditions.

The dissertation also presented a localisation algorithm for autonomous drone landing, focusing primarily on computer vision techniques involving camera calibration and pose estimation using fiducial markers. The initial emphasis was on understanding the intricacies of camera calibration, a necessary step to correct for lens distortion and ensure accurate projection of 3D objects into 2D images. This foundational work laid the groundwork for precise localisation, essential for subsequent pose estimation tasks. The utilisation of OpenCV's robust capabilities in both calibration and marker detection was pivotal, underscoring the importance of software in achieving high accuracy in vision-based systems.

The subsequent focus on pose estimation, particularly using ArUco markers, highlighted the practical aspects of identifying and tracking markers to determine the drone's position and orientation. The implementation involved a detailed algorithm capable of real-time marker detection and pose computation, providing essential data for the navigation system. This phase was validated through tests that demonstrated the system's ability to correctly interpret and respond to changes

in the drone's environment, crucial for safe and precise landings. The alignment of the camera and marker reference frames, facilitated by the calculated rotation matrices, ensured consistent and reliable measurements, reinforcing the robustness of the approach.

Future work should focus on improving the stability and versatility of the loading/unloading system. This includes exploring the use of lighter materials like carbon fiber for structural components and integrating additional servo motors for improved payload alignment. Expanding the system's capabilities to handle various payload types and conditions will be crucial for broader adoption in commercial and industrial applications.

The dissertation also identified necessary advancements to enhance the system's precision and functionality. Future work includes rigorous testing with accurate measurement tools to quantify angular errors, integration with a drone's control system using a Raspberry Pi and MAVLink protocol, and converting the pose estimation algorithm into a versatile library capable of estimate the pose of the drone. The proposed developments aim to translate this work into a fully operational drone landing solution, capable of real-time adjustments and precise landings.

References

[1] Ajay Vishwath N.C., Arvind R Yadav, Deep Mehta, Jatin Belani, and Ravi Raj Chauhan. A guide to novice for proper selection of the components of drone for specific applications. *Materials Today: Proceedings*, 65:3617–3622, June 2022.

[2] Valentina Gatteschi, Fabrizio Lamberti, Gianluca Paravati, Andrea Sanna, Claudio Demartini, Alberto Lisanti, and Giorgio Venezia. New Frontiers of Delivery Services Using Drones: A Prototype System Exploiting a Quadcopter for Autonomous Drug Shipments. In *2015 IEEE 39th Annual International Computers, Software and Applications Conference*, volume 2, pages 920–927, 2015.

[3] Yu-Jin Zhang. *Camera Calibration*, pages 37–65. Springer Nature Singapore, Singapore, 2023.

[4] MathWorks. What Is Camera Calibration? `https://www.mathworks.com/help/vision/ug/camera-calibration.html`. Accessed on 1st of July of 2024.

[5] Atar Fuady Babgei and Heri Suryoatmojo. Building an Unmanned Aerial Vehicle for Humanitarian Aid Delivery. *Indonesian Journal of Electronics and Instrumentation Systems (IJEIS)*, 10(1):53–64, 2020.

[6] Christopher Burke, Hung Nguyen, Mary Magilligan, and Rafiqul Noorani. Study of A Drone's Payload Delivery Capabilities Utilizing Rotational Movement. In *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, pages 672–675, 2019.

[7] Kwok Quan Eu and Swee King Phang. Automated Parcel Loading-Unloading Mechanism Design for Delivery UAV. *Journal of Physics: Conference Series*, 2523(1):1–12, July 2023.

[8] Lydia Hingston, Jonathan Mace, Joao Buzzatto, and Minas Liarokapis. Reconfigurable, Adaptive, Lightweight Grasping Mechanisms for Aerial Robotic Platforms. In *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 169–175, 2020.

[9] P.M. Kornatowski, S. Mintchev, and D. Floreano. An origami-inspired cargo drone. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6855–6862, 2017.

[10] Przemyslaw M. Kornatowski, Mir Feroskhan, William J. Stewart, and Dario Floreano. Downside Up:Rethinking Parcel Position for Aerial Delivery. *IEEE Robotics and Automation Letters*, 5(3):4297–4304, 2020.

[11] Shaoqian Lin, Joao Buzzatto, Junbang Liang, and Minas Liarokapis. An Adaptive, Reconfigurable, Tethered Aerial Grasping System for Reliable Caging and Transportation of Packages. In *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 7–13, 2022.

[12] Ramon F. Brena, Juan Pablo García-Vázquez, Carlos E. Galván Tejada, David Muñoz-Rodriguez, Cesar Vargas-Rosales, James Fangmeyer Jr, and Alberto J. Palma. Evolution of Indoor Positioning Technologies: A Survey. *Journal of Sensors*, 2017:1–21, March 2017.

[13] Hakan Koyuncu and Shuang-Hua H Yang. A Survey of Indoor Positioning and Object Locating Systems. *IJCSNS International Journal of Computer Science and Network Security*, 10(5):121–128, May 2010.

[14] Dragan Stojanovic, Billur Barshan, A. Papadopoulos, Nico Van de Weghe, and Christophe Claramunt. *Positioning Methods and Technologies in Mobile and Pervasive Computing*, pages 5713–5722. IGI Global, January 2015.

[15] Pei Jiang, Yunzhou Zhang, Wenyan Fu, Huiyu Liu, and Xiaolin Su. Indoor Mobile Localization Based on Wi-Fi Fingerprint's Important Access Point. *International Journal of Distributed Sensor Networks*, 2015:1–8, April 2015.

[16] Adam Marut, Konrad Wojtowicz, and Krzysztof Falkowski. ArUco markers pose estimation in UAV landing aid system. In *2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pages 261–266, 2019.

[17] Igor Lebedev, Aleksei Erashov, and Aleksandra Shabanova. Accurate Autonomous UAV Landing Using Vision-Based Detection of ArUco-Marker. In Andrey Ronzhin, Gerhard Rigoll, and Roman Meshcheryakov, editors, *Interactive Collaborative Robotics*, volume 12336, pages 179–188, Cham, 2020. Springer International Publishing.

[18] Artur Khazetdinov, Aufar Zakiev, Tatyana Tsoy, Mikhail Svinin, and Evgeni Magid. Embedded ArUco: a novel approach for high precision UAV landing. In *2021 International Siberian Conference on Control and Communications (SIBCON)*, pages 1–6, 2021.

[19] Ju Wang, Devon McKiver, Sagar Pandit, Ahmed F. Abdelzaher, Joel Washington, and Weibang Chen. Precision UAV Landing control based on Visual Detection. In *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 205–208, 2020.

[20] Raspberry Pi Pico W Datasheet. https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf. Accessed on 25th of April of 2024.

[21] Bill Earl. Adafruit PCA9685 16-Channel Servo Driver. https://learn.adafruit.com/16-channel-pwm-servo-driver?view=all. Accessed on 25th of April of 2024.

[22] MG90S Datasheet (PDF). https://pdf1.alldatasheet.com/datasheet-pdf/view/1132104/ETC2/MG90S.html. Accessed on 26th of April of 2024.

[23] Step Down Regulável 1.25 32Vdc 25W - XL4005. https://www.botnroll.com/pt/step-down/3741-step-down-regul-vel-1-25-32vdc-25w-xl4005.html. Accessed on 26th of April of 2024.

[24] ROL. 618/6 2Z=686 2Z-INOX ISA. https://www.rolisa.pt/p3-p-2730-rol-618-6-2z686-2z-inox-isa-pt. Accessed on 26th of April of 2024.

[25] OpenCV. Camera Calibration. https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html. Accessed on 11th of July of 2024.

[26] OpenCV. https://opencv.org/. Accessed on 12th of July of 2024.

[27] OpenCV. Detection of ArUco Markers. https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html. Accessed on 12th of July of 2024.

[28] Soumik Dhar. ArUco-Gen. https://aruco-gen.netlify.app/. Accessed on 31st of May of 2024.

[29] Inc. NaturalPoint. OptiTrack - Motion Capture Systems. https://optitrack.com/. Accessed on 5th of September of 2024.

[30] MAVLink. https://mavlink.io/en/. Accessed on 22nd of July of 2024.

[31] Jaeyoung Lim. Trajectory following with MAVROS OFFBOARD on Raspberry Pi. https://404warehouse.net/2016/08/10/trajectory-following-with-mavros-on-raspberry-pi/. Accessed on 19th of July of 2024.