POLITECNICO DI TORINO

Master Degree course in Data Science and Engineering

Master Degree Thesis

Relocalization of Autonomous Agents Using Monocular Depth Estimation on PTZ Cameras

Supervisors
Prof. Giorgio Guglieri
Ph.D. Alessandro Minervini

Candidate
Davide Fassio

Academic Year 2024-2025

"A monad is a monoid
in the category of endofunctors"

Acknowledgements

I would like to express my deepest gratitude to all the people who supported and accompanied me throughout the journey of this thesis, which is part of my academic path and, in turn, a chapter of my broader life adventure.

First and foremost, I would like to thank Professor Giorgio Guglieri for supervising this thesis and for the life lessons he has shared during our interactions. I would also like to sincerely thank my co-supervisor, PhD Alessandro Minervini, whose mentorship has been invaluable, not only in the context of this thesis, but also as my team leader.

This work would not have been possible without the incredible experience of collaborating with DRAFT PoliTo, a team that has significantly shaped my learning and growth. I am grateful to all my teammates, and in particular to the members of my Tribe, Deep Learning & Computer Vision, for their constant support, brainstorming sessions, and shared passion.

Beyond technical contributions, I want to express my gratitude to the people who have always been by my side on a more personal level.

To my family, thank you for standing by me unconditionally and for supporting my path and my choices. A special thank you goes to my sister, who paved the way before me and has always been a point of reference.

To my friends and colleagues, thank you for sharing both the joyful and the challenging moments of this journey, for the laughs, the long talks, and all the small memorable moments.

To my girlfriend, thank you for always encouraging me, for pushing me to give my best, and for being my unwavering source of motivation and balance.

And finally, a heartfelt thank you goes to my faithful MacBook, which has been with me from Analysis I all the way to this thesis, still holding on despite its outdated architecture and no longer receiving software updates.

To all of you, thank you.

Abstract

This thesis presents a systematic approach for calibrating a Pan-Tilt-Zoom network camera and integrating it with state-of-the-art object detection and depth estimation algorithms for distance measurement of autonomous agents.

A calibration methodology was adopted to compute the intrinsic parameters of the camera across various zoom levels, addressing the non-linear variations induced by dynamic zoom adjustments. This calibration process is fundamental for establishing an accurate mapping between the three-dimensional world and the two-dimensional image plane of the camera.

Subsequently, the study focuses on estimating the distance of autonomous agents by leveraging a dual-framework approach. Object detection is performed using YOLOv8, chosen for its balance between computational efficiency and detection accuracy in real-time applications. For depth computation, a monocular image-based technique is employed using the Apple Depth Pro model, which incorporates a Vision Transformer architecture to capture high-level contextual features and infer depth from a single frame. The integrated system exploits the complementary strengths of YOLOv8 and ViT-based depth estimation. The object detection component provides precise localization, while the depth estimation framework supplies spatial measurements, resulting in an accurate distance assessment to the target autonomous agent.

The results indicate that the proposed methodology achieves robust performance in distance estimation, with notable improvements in detection accuracy and depth precision compared to traditional methods.

In conclusion, the primary goal of this work and its main results is the relocalization of autonomous agents thanks to an external PTZ camera. This approach originated from the challenges posed by the Leonardo Drone Contest. Moreover, it holds significant promise for real-world applications, including warehouse logistics, precision agriculture, and smart city infrastructure.

Contents

Chapter 1

Introduction

1.1 Problem Statement

Autonomous robots, often referred to as agents, rely heavily on accurate positioning information to navigate, plan tasks, and coordinate with other agents in shared environments. While numerous state-of-the-art localization techniques exist, such as GPS-based navigation, LiDAR-based SLAM, and multicamera vision systems, each solution presents distinct challenges related to cost, infrastructure demands, and integration complexities.

In certain contexts, such as the Leonardo Drone Contest, an external stationary camera may be available, necessitating novel approaches for robust localization. Recent computer vision advancements, particularly in object detection (e.g., YOLO) and depth estimation (e.g., Apple Depth Pro), allow for the extraction of three-dimensional coordinates from a monocular camera feed. Incorporating a Pan-Tilt-Zoom (PTZ) mechanism further enhances coverage through its rotational capabilities and addresses issues with small or distant targets using its optical zoom.

Despite these technological advances, significant gaps remain in understanding how best to integrate PTZ control, object detection, and monocular depth estimation systems to accurately determine the 3D positions of autonomous agents. Challenges include selecting adaptive camera control policies for targets at varying ranges, minimizing the impact of motion blur at higher zoom levels, and managing diverse lighting or environmental conditions. Furthermore, it is unclear how sensor noise and the precision of modern monocular depth estimation methods impact the overall accuracy and robustness of the robot relocalization process.

Therefore, this thesis aims to design and evaluate a systematic workflow that:

1. Calibrates both intrinsic and extrinsic camera parameters to compensate for manufacturing inaccuracies.

2. Detects autonomous agents within a single camera feed via object detection.

3. Estimates reliable 3D positions for these agents by combining monocular depth estimation techniques with knowledge of the camera's pan, tilt, and zoom.

By tackling this problem, the work evaluates the feasibility of using a single PTZ camera as a versatile and cost-efficient solution for robot relocalization in scenarios where deploying multiple cameras or specialized sensors is impractical or economically unfeasible.

1.2 Multi-robot Collaboration

Multi-robot collaboration refers to the coordinated operation of multiple autonomous agents, each possessing distinct capabilities and resources, working together to achieve a shared objective. This approach contrasts with single-robot deployments by leveraging complementary strengths, distributing workload, and enhancing the overall system's flexibility and robustness. Recent advances in robotics have demonstrated the potential of heterogeneous teams in complex and dynamic environments, including disaster response, environmental monitoring, and city-scale surveillance tasks.

The overall coordination involves:

- Mission Planning and Task Allocation: Determining which robot is best suited for a particular sub-task based on real-time sensor data and global mission objectives (e.g., coverage, inspection, or tracking).
- Information Sharing: Exchanging sensor data (video, GPS location, etc.) among all team members to reduce the uncertainty of target localization and optimize the decision-making process.
- Coordinated Navigation: Adjusting paths and trajectories to avoid collisions and minimize operational time in search-and-detect missions.

A critical aspect of such coordinated deployments is designing robust communication protocols and real-time decision-making algorithms that handle high-bandwidth sensor streams, such as camera feeds and LiDAR point clouds. Furthermore, safety mechanisms and fail-safe routines must be carefully integrated to manage partial failures or drops in communication, which are more likely when multiple heterogeneous platforms operate concurrently.

In the context of our work, we focus on a collaborative system composed of three types of robots: an unmanned aerial vehicle, a ground rover, and a PTZ camera unit. Each agent in the team offers unique advantages:

- Drone (UAV): Provides a broad overhead view and rapid response to high-level tasks such as mapping, target detection, and relaying positional information. Its aerial perspective and agility are particularly useful for overcoming visual occlusions and navigate complex environments.
- Rover (UGV): Operates on street level, carrying out close range inspection and capture detailed images of potential targets and navigate urban terrains.
- PTZ Camera: Offers a stationary vantage point with the ability to quickly reorient and zoom in on areas of interest. The PTZ camera can complement both the

aerial and ground views, providing persistent surveillance or targeted monitoring in critical zones.

Collaboration among these heterogeneous robots is particularly beneficial in city-like environments and by leveraging distinct sensing modalities and vantage points, the multi-robot team can efficiently locate and identify targets such as ArUco markers or other objects of interest.

1.3 Leonardo Drone Contest

The *Leonardo Drone Contest* is an Italian university competition organized by Leonardo S.p.A. The contest serves as a platform for fostering technological innovation in autonomous systems, combining research, experimentation, and competition to push the boundaries of aerial and ground robotics. This initiative aligns with Leonardo's strategic goal of enhancing the development of collaborative and intelligent robotic systems for real-world applications.

I had the opportunity to participate multiple times in this competition as a member of the DRAFT PoliTo student team. My contributions to the team, particularly in the context of this competition, serve as the primary foundation of this thesis.

The main objective of the competition is to develop and test heterogeneous autonomous robotic systems, including Unmanned Aerial Vehicles (also referred as drone) and Unmanned Ground Vehicles (also referred as rover), capable of working together aided by an external PTZ camera. The competition emphasizes:

- Autonomous navigation and path planning in unknown GNSS-denied environments.
- AI-driven localization, mapping, and situational awareness.
- Real-time decision-making and coordination between drone and rover.
- Data processing and communication with a Ground Control Station (GCS).

Each participating team must integrate and develop robotic platforms that will complete a series of predefined tasks within a partially unknown environment. The competition area has dimensions of $20 \times 10 \times 3$ meters and is enclosed with a net to ensure safety. A schema of the positions of known obstacles is shown in figure 1.1. The main tasks are:

- Target Identification: Detecting ArUco markers and specific objects from the COCO dataset.
- Follow-Me: The drone must autonomously track and follow the rover within a defined corridor.
- Emergency Response: Performing predefined emergency actions, such as emergency landing or return-to-base procedures.

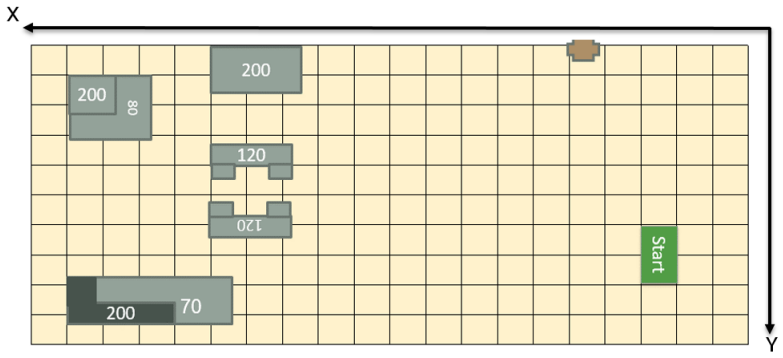

Figure 1.1. Competition area with the position of known obstacles. Movable obstacles are placed in the right half of the map.

1.4 Applications & Related Work

Relocalization of autonomous agents using monocular depth estimation on PTZ cameras is an emerging approach that combines cost-effective sensing with active vision capabilities. This approach finds broad applications across various domains.

In warehouse logistics mobile robots and automated guided vehicles can be tracked and relocalized by overhead PTZ cameras as they navigate storage aisles, enabling efficient inventory management and collision avoidance. Monocular depth cues help estimate the distance of a robot from racks or obstacles, which is critical in tight indoor spaces.

Similarly, in precision agriculture, autonomous tractors and drones equipped with monocular cameras can estimate crop row distances and terrain undulations to localize themselves in fields. A PTZ camera mounted on farm machinery can actively survey large farmland plots, zooming in to assess plant health or guide robots for targeted spraying. Monocular depth estimation provides a low-cost alternative to LiDAR in these outdoor environments, assisting in mapping crop heights and detecting obstacles.

Smart city infrastructure is another key domain: PTZ surveillance cameras installed at traffic intersections or in street lamps can monitor autonomous delivery robots and connected vehicles, using object detection and depth estimation to determine their positions on roadways or sidewalks. This can also aid the current development of autonomous driving and advanced driver assistance systems (ADAS), and in the future enable smarter traffic management. In self-driving cars, a forward-facing monocular camera — in combination with the detection of car's taillight — can estimate distances to vehicles [1] and pedestrians, providing essential input for collision avoidance and navigation.

Developing robust relocalization on PTZ monocular cameras requires integrating advances from several research areas.

A foundational component is monocular depth estimation. Since inferring depth from a single image is an inherently ill-posed problem, early methods relied on machine learning-based approaches to predict plausible depth maps. A pioneering work by Eigen et al. [4] introduced a multi-scale convolutional network to regress depth from a single image, demonstrating that coarse scene layout and fine details could be learned together.

This opened the door to purely vision-based distance sensing. Subsequent approaches significantly improved accuracy and generalization. Many leveraged deep convolutional neural networks trained on large RGB-D datasets or stereo images; others moved towards self-supervised learning to reduce the need for ground truth depth. For instance, Zhou et al. [23] showed that depth and camera motion can be learned jointly from monocular video sequences by enforcing geometric consistency between consecutive frames. Similarly, Godard et al. [5] developed a framework to learn depth from stereo image pairs without explicit depth labels, using consistency between the left-right views. These self-supervised techniques achieve remarkable results, making monocular depth estimation viable in domains where collecting LiDAR data is impractical. More recently, Transformers have been employed to capture global context, with Ranftl et al. [10,11] demonstrating that vision transformer models can boost depth estimation performance across diverse scenes. By training on mixed datasets spanning indoor and outdoor environments, such models achieve robust zero-shot generalization, a crucial attribute for relocalization in new or changing environments. The culmination of these advances is that from a single RGB frame one can obtain a dense depth map that provides the *metric* distance of each pixel from the camera. This breakthrough was first achieved by Piccinelli et al. [9] and later refined by Bochkovskii et al. at Apple [3].

Another critical component is object detection and recognition, which allows identifying and tracking the autonomous agent, or other relevant objects, in the camera's field of view. Modern deep learning-based object detectors have achieved high accuracy and real-time performance, making them well-suited for guiding PTZ cameras. For example, the Faster R-CNN detector by Ren et al. [13] employs a region proposal network to efficiently locate objects, achieving near real-time speeds without sacrificing detection precision. Redmon et al. [12]'s YOLO (You Only Look Once) series further optimized detection to run at high frame rates by using a single neural network pass to predict bounding boxes and class probabilities in an image. Beginning with YOLOv5 [16], the series has been developed by Ultralytics, introducing further enhancements in performance and usability. Such detectors can reliably recognize robots, drones, or vehicles in complex scenes, even from a single viewpoint. The synergy of detection and depth estimation thus yields not only the direction of the agent in the image but also an estimate of how far it is, essentially localizing the agent in 3D space relative to the camera.

The use of PTZ cameras brings unique advantages and challenges: the zoom mechanism actively alters the internal geometry of the camera, meaning that a calibration performed at one zoom level may become invalid when the lens is adjusted. In particular, as the zoom changes, the focal length varies, the effective optical center may shift, and the characteristics of lens distortion change. These changes are critical because they directly affect how a three-dimensional scene is projected onto the two-dimensional image plane, and thus any error in these parameters can lead to inaccuracies in distance estimation and overall relocalization performance [20, 21]. Traditional calibration techniques fail to capture nonlinear dynamic changes, necessitating either separate calibrations for each zoom setting or the development of a continuous model that predicts internal parameters as functions of the zoom level [2, 6]. Early approaches to intrinsic calibration relied on controlled laboratory conditions with known patterns. Willson's

work provided one of the seminal methods in this area by imaging a calibration grid at discrete zoom settings to build a lookup table of focal lengths, principal points, and distortion coefficients [20, 21]. Recent methods favor self-calibration using scene features and geometric constraints. Agapito et al. [2] leveraged the Image of the Absolute Conic (IAC) to estimate intrinsic parameters from multiple images. Initially suited for minimal lens distortion, later refinements incorporated optimization techniques like linear matrix inequalities to better manage distortion variability [6, 15]. With the rise of deep learning, Zhang et al. [22] proposed a new model, known as DeepPTZ, capable of predicting focal lengths and distortion coefficients directly from image pairs, even as the camera zooms. These methods leverage large datasets and the power of convolutional neural networks to learn the nonlinear relationships between zoom level and intrinsic parameters without requiring manual intervention.

Combining these components, researchers have developed integrated systems demonstrating effective monocular relocalization. One approach is to incorporate known visual markers on the autonomous agents and use specialized detectors to improve range estimation. Oh et al. [7] describe a system that attaches circular fiducial markers on a robot; a PTZ camera guided by a CNN-based ellipse detector can recognize the marker at a distance and compute its 3D position using the marker's apparent size and the calibrated zoom parameters. Their results show infrastructure-free localization in indoor environments with centimeter-level accuracy, without relying on external sensors. Conversely, Unlu et al. [19] demonstrate a PTZ camera system that can autonomously track a flying UAV by continuously detecting it with a neural network and controlling the camera motors to center the target. While their primary goal was persistent tracking, the system implicitly performs relative relocalization whenever the drone re-enters the camera's view after a maneuver, owing to the wide-area coverage of the PTZ and the learned model's ability to re-detect the UAV on appearance. During such tracking, the distance to the drone can be inferred either through the drone's known dimensions or via the depth estimation networks discussed earlier.

Overall, the literature shows that by fusing advances in single-image depth perception, object detection, and camera control, one can achieve reliable relocalization of autonomous agents across a range of practical scenarios.

1.5 Thesis Structure

This thesis is organized into five main chapters, each addressing a critical aspect of the research. The structure is designed to provide a clear and logical progression from the background and technical context to the methodological approach, results, and conclusions.

Below is an overview of each chapter:

1. **Introduction:** This chapter introduces the research topic, outlining the problem, its significance, and the motivation behind the study. It outlines the research questions and the significance of the work within the broader academic and practical context.

2. **Hardware and Software Context:** This chapter details the technical environment in which the research was conducted. It describes the hardware components utilized, including specifications and performance considerations, as well as the software stack employed. The programming languages, APIs, and frameworks used in the development and implementation of the research are also discussed, providing necessary background for understanding the methodology and experimental setup.

3. **Methodology:** The methodology chapter explains the approach taken to address the research problem. It describes the experimental design, data collection methods, and analytical techniques used. Justification for the chosen methodologies is provided, along with details of the integration of hardware and software components and the underlying assumptions made.

4. **Results:** The findings of the research are presented and analyzed in this chapter. The results are interpreted in relation to the research questions and objectives, highlighting key insights.

5. **Conclusion and Future Work:** The final chapter summarizes the main findings and discusses their implications. It reflects on the contributions of the research and its practical applications. Finally, it suggests directions for future research, highlighting potential improvements and alternative approaches.

Chapter 2

Hardware and Software Context

This chapter provides an overview of the fundamental principles, components, and technologies underlying PTZ cameras, setting the stage for the innovative work discussed in subsequent chapters. Firstly we will talk about the specific camera at hand, then we will explain all the software used to develop and deploy the algorithms employed, lastly we will explore all the functionality of this camera from the most basic ones to the more advanced.

2.1 AXIS M5525-E PTZ Network Camera

PTZ cameras use pan, tilt and zoom to provide both wide-area coverage and great detail with a single camera. Great image quality and the ability to zoom in make it possible to verify and detected security events. Axis cameras are equipped with a variety of intelligent features and can move between pre-set positions and zoom in automatically in response to detected events.

The M55 line is specifically designed to be affordable and to be deployed both in indoor and outdoor environments.

Figure 2.1. AXIS M5525-E PTZ Network Camera mounted on its support.

2.1.1 Technical Specifications

This camera supports a maximum video resolution of 1920x1080 pixels, which ensures full high-definition (HD) video quality. The camera operates at a maximum frame rate of 25 frames per second (fps) in regions with a power supply frequency of 50 Hz, such as Europe.

The camera features a complete pan range of 360 degrees, enabling it to rotate fully and cover an area without any blind spots. The tilt range is from 0 to -90 degrees, allowing the camera to cover areas directly below it.

The camera is equipped with a 10× optical zoom. Additionally, the camera offers a 12× digital zoom, which, while increasing the total magnification to 120×, may result in some loss of image quality. Moreover, the camera includes auto-focus and auto-iris functionalities, ensuring that images remain sharp and correctly exposed even when lighting conditions change or when zooming in and out.

The camera utilizes Power over Ethernet (PoE) Class 3, which simplifies the installation process by allowing both power and data to be transmitted over a single Ethernet cable.

2.1.2 On-Board Capabilities and API

The on-board capabilities of Axis cameras are accessible through the proprietary VAPIX® APIs, which are RESTful APIs designed to enable easy integration and management of the camera. REST (Representational State Transfer) APIs rely on stateless communication between the client and server.

The API is divided in sections. The most used ones are explained in details below.

Network Video

The API provides endpoints to access live video streams directly from the camera. Users can request streams in various formats, including MJPEG and H.264.

The VAPIX API offers precise control over the camera's pan, tilt and zoom capability, allowing to programmatically adjust the camera's orientation and field of view.

It is also possible to set the focus and the brightness level, or let the camera find the best combination automatically.

The camera has the ability to go to preset position and follow preset tours.

The API allows for querying the camera to obtain detailed information about its status, supported features and current configuration settings.

Applications

Axis cameras have various preinstalled onboard applications that enhance the camera's functionality. These applications are accessed through the Common Gateway Interface (CGI). The preloaded applications are focused on crucial aspect of security, like:

- Fence, loitering, and motion guard: enables the camera to identify and alert on specific behaviors, such as crossing a virtual boundary (fence), lingering in an area (loitering), or detecting movement (motion).

- Auto-tracking: enables the camera to automatically track moving objects within its field of view. The camera can rotate and zoom to better follow the target.
- People counter: allows the camera to count the number of people entering or exiting a defined area. It is useful to understand visitor trends, occupancy levels and the flow of people for operational planning.
- Queue monitor: allows the camera to count the number of people in a defined area.
- Object detection and analytics: facilitates advanced object recognition and classification, allowing the camera to distinguish between different types of objects (e.g. people, vehicles, etc...) and trigger appropriate actions based on predefined rules.
- License plate verifier: enables the camera to recognize and verify vehicle license plates, making it ideal for access control systems in parking lots, gated communities, or other restricted areas.

Audio Systems

Axis products support two-way audio with full duplex modality, that is the Axis product can both transmit and receive audio at the same time, using built-in or external microphone and speaker. This API provides also functionality to react to audio events with an alert or an automatic response.

Device Configuration

The management and configuration of the Axis PTZ camera are handled through the Device Configuration APIs (DCA). These APIs provide comprehensive control over the camera's settings, including device administration and initial configuration.

2.2 Development Environment

The development environment for this project was chosen to ensure robustness, compatibility, and ease of deployment for a robotic software system.

Our software must run on three different systems: the UAV, the UGV and the ground station. To ensure consistency and portability all the applications run inside Docker containers. A Docker container contains the application and all the dependencies needed to run it, like operating system, libraries, and tools.

The operating system used inside the Docker containers is Ubuntu 20.04 LTS (Focal Fossa). Ubuntu is a Linux distribution popular for development in the robotics community due to its large repository of software packages, and support for ROS (Robot Operating System). A long-term support release was selected because it ensures that the system will receive updates and security patches for an extended period.

ROS2 Foxy Fitzroy is the version of ROS used in this project. ROS2 is a middleware suite designed for developing robotic applications, providing services such as hardware abstraction, device control, and communication between processes. The Foxy release

is an LTS version, ensuring that it is stable and well-supported, which is essential for long-term projects.

ROS2 officially supports both Python and C++, but for this project, we chose to use C++ over Python due to the significant overhead introduced by the Python interpreter in embedded systems. Specifically, we opted for C++14, as it offers modern language features while maintaining compatibility with all relevant compilers. However, for machine learning applications on the ground station, where resources are less constrained, we used Python because of its ease of use and the wide variety of available packages.

Useful libraries that are used throughout the codebase are:

- OpenCV (Open Computer Vision Library): a popular library used for computer vision and image processing tasks. It provides a wide range of highly optimized functionality for image manipulation, including reading and writing images and videos, image filtering, ArUco marker detection, and more.
- Boost: a collection of portable, header-only, C++ libraries that provide a variety of functionalities to enhance programming. In particular we used Asio, for networking tasks, and Process, for managing system processes.
- PyTorch: a deep learning framework developed by Facebook's AI Research lab. It is widely used to train, evaluate, and deploy neural network models with GPU acceleration.

2.3 Base Functionality

In this section we will discuss all the basic functionality needed to operate a PTZ camera. We will make ample use of Linux commands and VAPIX APIs. We will start by checking that the camera is connected and reachable, then we will move on to move it by sending commands and finally getting the current status and the image.

2.3.1 Connectivity Testing

To verify the connectivity between the host and the Axis camera we use a process commonly known as *pinging*, offered by the Internet Control Message Protocol (ICMP). The ICMP ping request allows us to determine if a specific IP address is reachable and measure the round-trip time of packets.

To perform a ping operation programmatically we need to send ICMP packets directly. Since ICMP is encapsulated within the IP protocol and user-mode sockets only support TCP/UDP, establishing a raw socket is needed for direct packet manipulation, at the cost of requiring root privileges.

By using the built-in ping command we are able to access the information required as regular user, without managing the sockets ourselves.

From C++ we can call the command with `std::system("ping -c 1 -W 5 <IP>")`. In this case we send 1 packet (`-c 1`) and wait a maximum of 5 seconds for a response (`-W 5`) from the server (`<IP>`). If a response is received and the return value is zero, we can conclude that the server is reachable.

2.3.2 Sending Commands

The API endpoint for this functionality is:

`http://<IP>/axis-cgi/com/ptz.cgi?<argument>=<value>`

To move the camera there are three modalities:

- Absolute position: set the absolute coordinates of the camera in its coordinate system. We can use the arguments `pan`, `tilt`, `zoom`, with the values in degrees.
- Absolute velocity: set the velocity of pan, tilt and zoom and follow the trajectory till the limit is reached. We can set the velocity with the argument `speed`, then we start the movement with `continuouspantiltmove` and `continuouszoommove`.
- Relative position: set the coordinates relative to the current position. We have two options:
  - move by a certain amount of degrees, using `rpan`, `rtilt` or `rzoom` to increase/decrease the zoom level.
  - move to a certain pixel in the image, using `center` or `areazoom` if we want also to zoom in or out.

In our work we only used the ability to set the absolute position and to look at a specified pixel in the captured image.

When sending commands we must always keep in mind that, in VAPIX, the pan is positive going right and the tilt is negative looking down.

2.3.3 Receiving Status

To get the current status of the camera we must use the same endpoint as in 2.3.2, with the argument `query`.

We can query various aspects by using different values:

- `position`: current values of pan, tilt, zoom, focus, brightness, auto-focus and auto-iris.
- `speed`: current value of pan-tilt speed
- `limits`: range of the different parameters
- `attributes`: miscellaneous camera attributes

The most used ones are `position` and `speed`, whereas `limits` and `attributes` are usually only used while configuring or debugging.

2.3.4 Receiving Images

Getting the images captured by the camera is a fundamental step.

The VAPIX APIs provide two interfaces: over HTTP or over RTSP (Real Time Streaming Protocol). For ease of implementation, we utilized the HTTP interface.

We can choose between continuous video streaming or capturing single images on demand. Each approach offers distinct advantages.

Video

To get the Motion JPEG video we use the endpoint:

```
http://<IP>/axis-cgi/mjpg/video.cgi?<argument>=<value>
```

This will get us a continuous flow of JPEG images.

By employing the `VideoCapture` class from OpenCV and constructing it with the URL, the class handles the API request internally, allowing us to interface with it as though it were a conventional video stream.

However, a limitation of this approach is that the camera may buffer some frames to maintain a smooth video feed. As a result, the images received may be slightly delayed, which could be problematic for applications requiring real-time processing.

Image

To get a single JPEG image, a snapshot, we use the endpoint:

```
http://<IP>/axis-cgi/jpg/image.cgi?<argument>=<value>
```

Before using the image, it is necessary to decode the raw data received. This can be achieved by employing the OpenCV function `imdecode`.

In our task, we only need to capture a single frame every two seconds. To achieve this efficiently, we utilize this endpoint to minimize network load and reduce latency.

2.4 Advanced Functionality

In this section, we will explore advanced functionalities essential for effectively utilizing a PTZ camera. Our approach will leverage algorithms implemented in OpenCV and employ various coordinate system transformations. The process begins with the calibration of the camera, followed by the detection and pose estimation of an ArUco marker. Finally, we will explain the algorithm used for orienting the camera to focus on a specified point in 3D space.

2.4.1 Camera Calibration

Camera calibration is a fundamental process in computer vision and photogrammetry that aims to determine the parameters of a camera to accurately relate the 3D world to a 2D image. This process is essential for applications that require high precision, such as robotics. Camera calibration involves estimating both intrinsic and extrinsic parameters.

Intrinsic Parameters

Intrinsic parameters are the internal characteristics of the camera that affect how it captures an image.

These parameters are specific to the camera's optics and sensor.

To model these parameters, we use the pinhole camera model (Fig. 2.2), a theoretically sound and mathematically simplified approach based on projective geometry, where each point in a three-dimensional scene is projected onto the image plane along a straight line passing through a single aperture, known as the pinhole.

This assumption is justified under the following conditions:

- Linear Projection: The model assumes that light travels in straight lines, which holds true in the absence of significant diffraction or scattering effects.
- Small Aperture Approximation: While real cameras have lenses with finite apertures, the pinhole model approximates the behavior of a camera with a very small aperture, minimizing the effects of lens-induced aberrations. This simplification is often sufficient for many computer vision applications where the primary concern is the geometric relationship between the scene and the image.

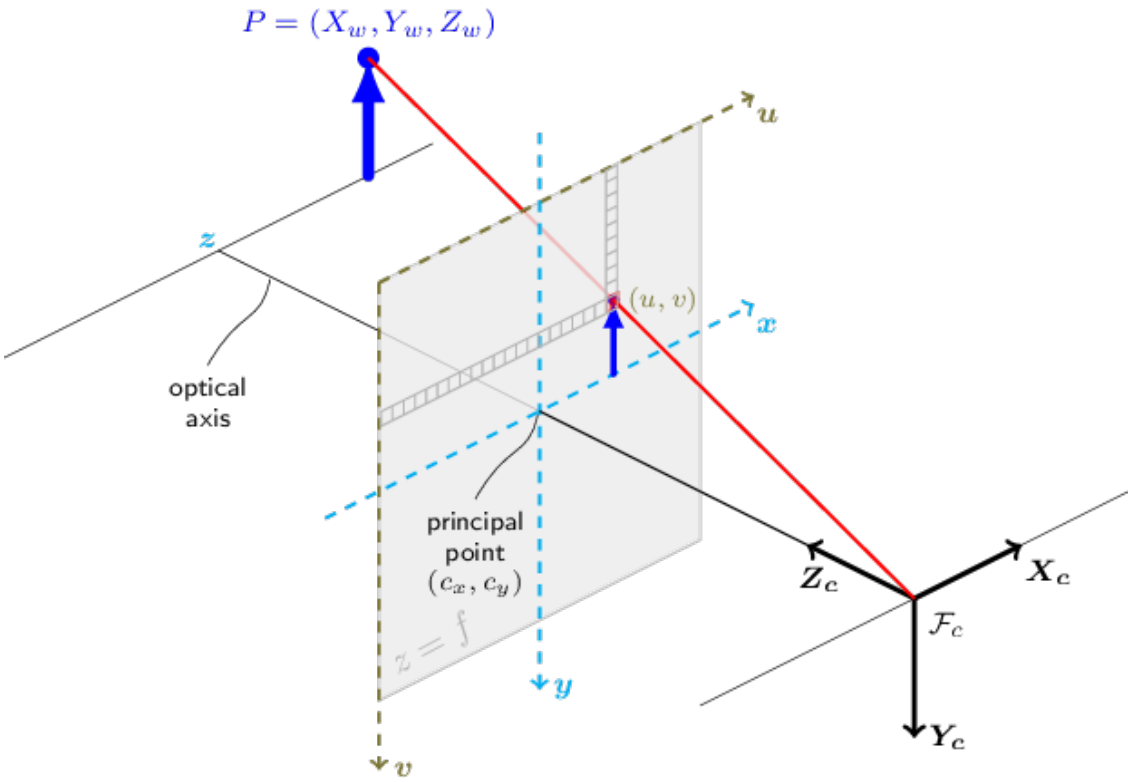

Figure 2.2. Pinhole camera model. It is shown how the focal length and principal point are constructed. Adapted from [8].

The main intrinsic parameters include:

- Focal length (f): is a measure of how strongly the camera lens converges light. It affects the field of view and the scale of the image. The focal length is typically represented in pixels in the camera matrix and can be different for the x and y axes (f_x, f_y).
- Principal point (c_x, c_y): is the point where the optical axis intersects the image plane. Ideally, this is at the center of the image, but due to manufacturing imperfections, it is often offset slightly.
- Distortion coefficients: estimate the distortion introduced by real lenses, particularly wide-angle lenses. The most common types are radial distortion and tangential distortion. Radial distortion causes straight lines to appear curved, while tangential distortion occurs when the lens and image plane are not perfectly parallel.

We follow the convention used by OpenCV to represent the intrinsic parameters with a camera matrix, which combines the focal lengths and principal point, and a vector of 5 distortion coefficients, 3 for radial distortion (k_1, k_2, k_3) and 2 for tangential distortion (p_1, p_2):

$$\textit{camera matrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

$$\textit{distortion coefficients} = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{2.2}$$

Extrinsic Parameters

Extrinsic parameters define the camera's position and orientation in the 3D world, relative to the global coordinate system.

The extrinsic parameters consist of:

- Rotation matrix (R): describes the orientation of the camera in 3D space relative to a world coordinate system. It is a 3×3 matrix that encodes the rotation needed to align the camera's coordinate system with the world coordinate system.
- Translation vector (t): represents the camera's position in the world coordinate system. It is a 3×1 vector that specifies the displacement from the world origin to the camera origin.

Together, the rotation matrix and translation vector can be combined into a single 3×4 matrix known as the extrinsic matrix:

$$\textit{extrinsic matrix} = [R \mid t] \tag{2.3}$$

This matrix transforms coordinates from the world coordinate system to the camera coordinate system.

Intrinsic Parameters Estimation

As recommended by OpenCV we use a chessboard pattern, because it is easy to recognize the locations where two black squares touch each other.

The important input data needed for calibration of the camera is the set of 3D real world points and the corresponding 2D coordinates of these points in the image.

The 2D points can be found in the image using the function `findChessboardCorners` (Fig. 2.3). It is also recommended to refine their position with `cornerSubPix`, to increase the accuracy. We need a representative set of images to get a good calibration, usually at least 20.

Figure 2.3. On the left the chessboard pattern used for calibration. On the right the location of 2D points.

The 3D location of the points can be derived with some assumptions:

1. the chess board is kept stationary, so the location of the points is always the same
2. the chess board is kept on the XY plane, so $z = 0$
3. the x and y axis are aligned with the rows and columns of the chessboard
4. the origin is congruent with the top-left point
5. the square size is 1 unit wide

Given this, we can say that the points are at:

$$\begin{matrix} (0,0,0) & (0,1,0) & (0,2,0) & \cdots \\ (1,0,0) & \ddots & & \\ \vdots & & & \end{matrix}$$

Once we have all necessary data we can use the OpenCV function `calibrateCamera` to obtain the camera matrix and distortion coefficients. Internally this function calls `solvePnP` that minimizes the difference between the projected 3D points, using the estimated parameters, and the actual detected 2D image points, implementing the Levenberg-Marquardt algorithm.

Extrinsic Parameters Estimation

To estimate the translation vector we measure, with a laser distance meter, the distance of the camera from the origin of the global coordinate system along each axis.

To estimate the initial orientation of the camera, we only need to calculate the yaw (pan) angle, because we assume that both the pitch (tilt) and roll angles are zero, as the camera is mounted parallel to the ground.

To compute the initial pan we:

1. set an ArUco marker (see 2.4.2) along the x-axis
2. align the center of the marker with the center of the image
3. query the status to obtain the current pan

2.4.2 Estimate Marker Pose

An important goal in the Leonardo Drone Contest and a crucial task in many robotic applications is to detect and estimate the pose of fiducial markers. If they are attached to an important object we can compute its position in space, otherwise if they are stationary and their position is known we can recompute the camera position and orientation.

To make it possible we must first examine the characteristics of the ArUco markers being utilized. Following this, we will explain how to detect and estimate the pose of these markers.

The whole process is illustrated in the figure 2.4.

ArUco

ArUco (Augmented Reality library from the University of Cordoba) markers are a type of fiducial marker used extensively in computer vision and robotics for pose estimation and augmented reality applications. These markers are designed to be easily recognizable by image processing algorithms, facilitating robust and accurate tracking in a variety of environments.

ArUco markers are binary square markers with a distinct pattern of black and white squares arranged in a grid. Each marker is uniquely identified by its pattern, which encodes a specific identifier (id) used to distinguish it from other markers.

The border of the marker is always black, which helps in detecting the marker's boundaries by classical computer vision algorithms.

ArUco markers are susceptible to partial occlusion as they lack error correction codes, a limitation addressed with the development of AprilTags.

Detection

To detect an ArUco marker we use the OpenCV function `aruco::detectMarkers` that takes in input the image and the dictionary that generated the binary patterns.

Internally this function performs:

1. Candidates detection:
   (a) Apply adaptive threshold over all the image
   (b) Extract contours that approximate to a square shape
   (c) Remove the contours too small, too big, too close to each other, etc...
2. Analysis of the inner codification, one contour at a time:
   (a) Apply perspective transformation to obtain the marker in its canonical form
   (b) Threshold using Otsu to separate white and black pixels
   (c) Divide the image into cells according to the marker size and the border size
   (d) Count the number of black or white pixels in each cell to determine if it is a white or a black bit
   (e) Analyze the bit pattern to determine if the marker belongs to the specific dictionary

The function returns, for each valid marker, the position of its four corners in the image and its id.

Pose estimation

To determine the position and orientation of an ArUco marker, we use the OpenCV function `aruco::estimatePoseSingleMarkers`. This function requires as input the corner positions of the marker, the side length of the marker, and the intrinsic parameters of the camera. It returns the marker's distance and rotation relative to the camera.

Given the camera's extrinsic parameters, and through the application of coordinate system transformations, it is possible to compute the marker's global position.

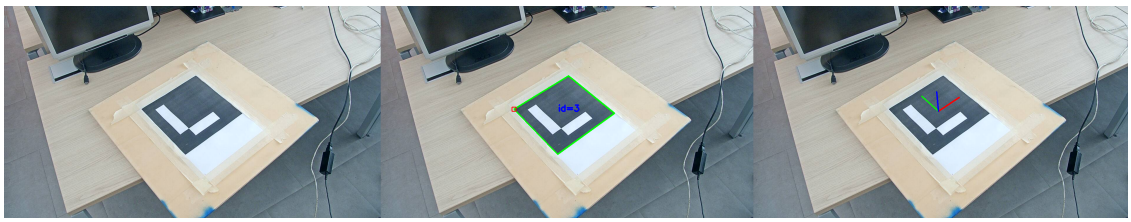

Figure 2.4. The left image shows an example of an ArUco marker. The middle image illustrates the detection process, with the corners highlighted by green lines outlining the sides, the first corner marked in red, and the marker ID displayed in blue. The right image depicts the coordinate system relative to the marker's center.

2.4.3 Look at Specified 3D Coordinates

An important quality-of-life feature and a crucial component of downstream applications is the ability to rotate the camera only by specifying a single 3D point of interest.

Given that the camera possesses two degrees of freedom (pan and tilt), the angles that satisfy the desired viewpoint are uniquely determined.

First, we calculate the vector originating from the axis position and extending towards the designated point:

$$\Delta = \text{point} - \text{axis\_position} \tag{2.4}$$

Now that we know the lengths of the catheti, we can calculate the angles using the arctangent function:

$$\text{pan} = \text{atan2}\left(\Delta_y, \Delta_x\right) \tag{2.5}$$

$$\text{tilt} = \text{atan2}\left(\Delta_z, \sqrt{\Delta_x^2 + \Delta_y^2}\right) \tag{2.6}$$

Finally, to send the command, we must correct for the initial pan, computed with all the extrinsic parameters:

$$\text{axis\_pan} = \text{axis\_init\_pan} - \text{pan} \tag{2.7}$$

Chapter 3

Methodology

This chapter presents the systematic approach developed for the relocalization of autonomous agents using a PTZ camera, integrating state-of-the-art object detection and monocular depth estimation techniques. The methodology is structured into two separate sections: the calibration of the PTZ camera and the agent relocalization through advanced computer vision algorithms.

3.1 Camera Calibration with Zoom Lenses

As outlined in Section 2.1.1, the AXIS M5525-E camera features a 10× optical zoom, internally encoded across 9,999 discrete levels ranging from 1 to 9,999. To simplify the calibration process, we have selected 10 primary levels: the base level (1×) and successive increments from 2× to 10×. This approach is a trade-off between complexity and usability.

During deployment, we can opt for one of the primary levels to ensure maximum accuracy or approximate by utilizing the calibration of the nearest available level.

3.1.1 Experimental Setup

The mathematical and programming details of the calibration process are described in Section 2.4.1.

To generate the dataset required for calibration, we collected a comprehensive set of images. A well-structured calibration dataset should contain at least 20 images per zoom level, ensuring diversity with respect to position within the frame, distance, rotation, and skew.

For each zoom level, we systematically captured 30 images with the following characteristics:

- 9 different positions within the frame for 3 distinct distances, for a total of 27 images.
- 3 additional images with the calibration pattern skewed at a close distance.

In total, we acquired 300 images of the calibration pattern. Figure 3.1 presents a selection of examples.

This process was made significantly faster by leveraging the pan-tilt capabilities of the camera to adjust its viewpoint, rather than manually repositioning the calibration pattern.

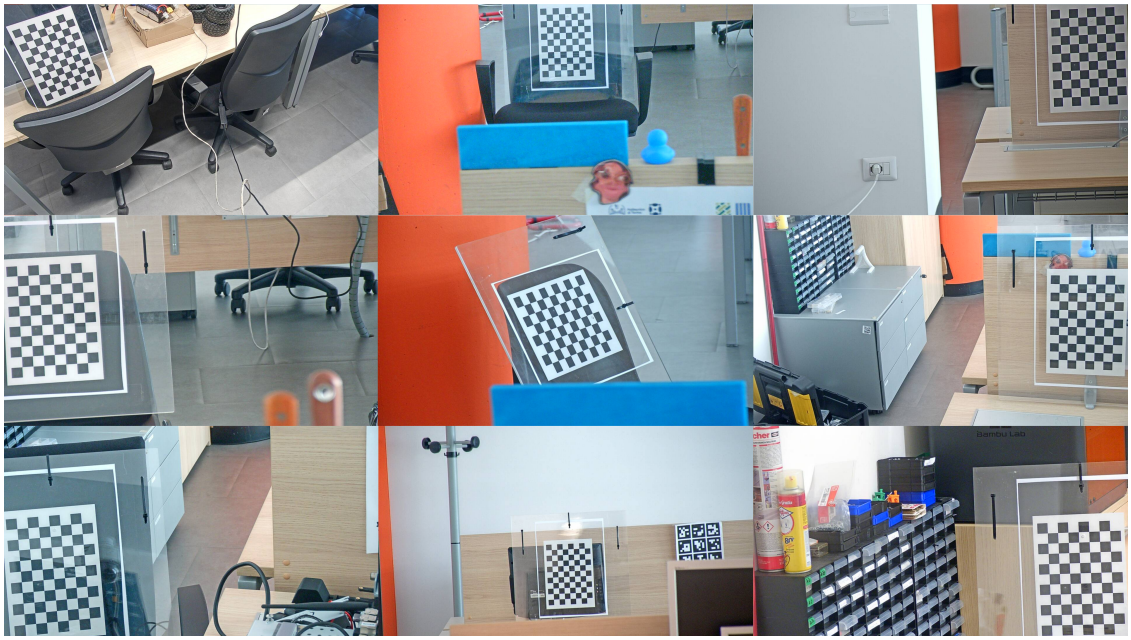

Figure 3.1. Samples from the calibration dataset showcasing variations in pattern position, distance, and skew across different zoom levels.

3.1.2 Evaluation Setup

To assess the accuracy of the calibration, we collected a test dataset consisting of 10 images, one per zoom level, each containing an ArUco marker at a fixed distance of 7.9 meters. Representative examples are presented in the figure 3.2.

The quality of the calibration was evaluated in the context of a downstream task: estimating the distance to the marker, as detailed in Section 2.4.2.

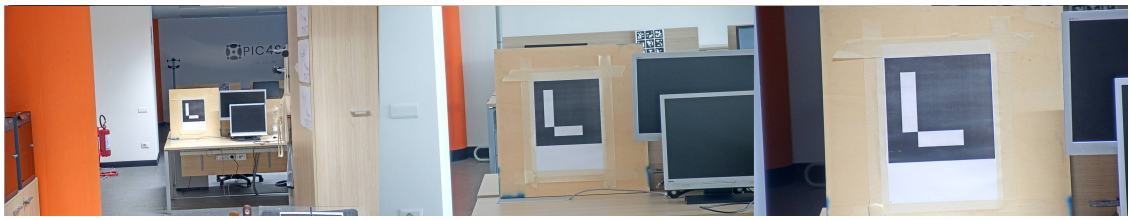

Figure 3.2. Samples from the evaluation dataset illustrating different zoom levels: 2× (left), 6× (center), and 10× (right).

3.2 Agent Relocalization

The agent relocation pipeline is implemented in Python, with each component analyzed independently to better assess its performance.

3.2.1 Object Detection Using YOLO

YOLO (You Only Look Once) is an object detection model that unites detection and classification in a single pass through a CNN, enabling fast real-time detection.

We used YOLOv8 [17], one of the most recent advancements in the YOLO family. The architecture is based on a CSPDarkNet53 backbone, a PAN (Path Aggregation Network) neck, and a detection head that predicts the bounding box coordinates, the class scores, and the confidence score. In this work, we opted for YOLOv8-small, a lightweight variant designed for resource-constrained devices, with 11.2 million parameters.

Fine-tuning is a transfer learning technique that allows a pre-trained model to adapt to a specific dataset. Instead of training from scratch, we initialize the model with weights pre-trained on a large dataset, in this case COCO, and subsequently refine these weights using a smaller, task-specific dataset

In practice for YOLO we have to:

- Replace the classification head to align with the number of target classes.
- Freeze the initial layers to preserve low-level feature representations.
- Train the later layers on the specialized dataset to enhance task-specific performance.

We collected a custom dataset of 279 manually annotated images for the detection of our agents. Each image is labeled with bounding boxes and corresponding class labels in the YOLO format.

To enhance generalization and robustness, we applied a set of data augmentation techniques to the training subset of the dataset. These transformations help the model learn invariant features and prevent overfitting to specific patterns in the training data. In practice, we used:

- Shear: both horizontal and vertical within a range of $\pm 5^{\circ}$.
- Saturation: within a range of $\pm 20\%$.
- Brightness: within a range of $\pm 15\%$.
- Exposure: within a range of $\pm 8\%$.
- Gaussian Blur: with standard deviation of 1 pixel.

3.2.2 Distance Estimation via Bounding Box Dimensions

Estimating the distance to an object from its image, when only its bounding box is available, is an underdetermined problem. Nevertheless, by leveraging the known intrinsic camera parameters and the real dimensions of the target, we can derive a reasonable estimate. The pose of the object is estimated using the OpenCV function `aruco::estimatePoseSingleMarkers`. This function requires as parameters:

1. `corners`: the coordinates of the four marker corners in clockwise order,
2. `markerLength`: the real-world length of one side of the marker,
3. `cameraMatrix`: the intrinsic camera matrix containing focal length and optical center,
4. `distCoeffs`: the radial and tangential distortion coefficients.

It returns the rotation vector `rvec` and the translation vector `tvec` that describe the 3D orientation and position of the target relative to the camera.

In the absence of full 3D information, we propose using the dimensions of the bounding box as a proxy for the objects physical measures. In particular, an approach rooted in integral geometry suggests that the *mean width* of a body can serve as a robust characteristic length. According to the definition introduced in convex geometry, the mean width of a convex body can be obtained via the Crofton formula as the average of its directional widths over all orientations of space [14]. For a rectangular prism with edge lengths a, b, and c, the width in any direction $u \in S^2$ is $a|u_x| + b|u_y| + c|u_z|$, and by symmetry, integrating over the unit sphere yields:

$$MW = \frac{a+b+c}{2} \tag{3.1}$$

In the image, we denote the measured width and height of the bounding box by W and H, respectively, and define their arithmetic mean as:

$$AM = \frac{W+H}{2} \tag{3.2}$$

Recall the following properties for any random variables X and Y:

- **Linearity of Expectation:** $\mathbb{E}[aX + bY] = a\mathbb{E}[X] + b\mathbb{E}[Y]$
- **Scaling Property of Variance:** $\mathrm{Var}(aX) = a^2 \mathrm{Var}(X)$
- **Variance of a Sum:** $\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) + 2\ \mathrm{Cov}(X, Y)$

Assuming that the object is subjected to a random 3D rotation, we obtain:

$$\mathbb{E}[W] = \mathbb{E}[H] = MW \tag{3.3}$$

We can now demonstrate that AM is an unbiased estimator of the mean width of the target:

$$\begin{aligned}\mathbb{E}[\text{AM}] &= \mathbb{E}[\frac{W+H}{2}] = \frac{1}{2}(\mathbb{E}[W] + \mathbb{E}[H]) \\ &= \frac{1}{2}(2\mathbb{E}[W]) \quad \text{(by symmetry, } \mathbb{E}[W] = \mathbb{E}[H]) \\ &= \mathbb{E}[W] = MW \end{aligned} \tag{3.4}$$

We compute the variance of AM as follows:

$$\begin{aligned}\text{Var}(AM) &= \text{Var}\Big(\frac{W+H}{2}\Big) = \frac{1}{4}\,\text{Var}(W+H) \\ &= \frac{1}{4}\Big(\text{Var}(W) + \text{Var}(H) + 2\ \text{Cov}(W,H)\Big) \\ &= \frac{1}{4}\Big(2\ \text{Var}(W) + 2\ \text{Cov}(W,H)\Big) \quad \text{(by symmetry, } \text{Var}(W) = \text{Var}(H)) \\ &= \frac{1}{2}\Big(\text{Var}(W) + \text{Cov}(W,H)\Big) \end{aligned} \tag{3.5}$$

By iterating through all possible side ratios, we empirically prove that:

$$\text{Cov}(W,H) \leq \text{Var}(W) \tag{3.6}$$

Which implies:

$$\text{Var}(AM) \leq \text{Var}(W) \tag{3.7}$$

Therefore, using the arithmetic mean reduces the variance relative to using a single dimension, leading to a more precise estimation.

In conclusion, we construct `corners` as a square with side length AM and set the `markerLength` parameters to MW. The magnitude of `tvec` provides an unbiased estimate of the distance of the target.

3.2.3 Depth Estimation using Vision Transformers

Monocular depth estimation seeks to infer scene depth from a single image, a fundamentally ill-posed problem that relies on learned scene cues and context.

Vision Transformers (ViTs) have emerged as a powerful alternative to CNNs for dense prediction tasks. The self-attention mechanism at the core of ViTs allows each image patch to dynamically attend to every other patch. This global attention is particularly advantageous for depth estimation as it enables the model to capture long-range dependencies within the image, whereas CNNs are inherently limited by their localized receptive fields. The self-attention mechanism computes pairwise interactions between all image patches, effectively learning affinities that correlate with the underlying 3D structure. This enables the network to group pixels belonging to the same distant surface or to detect sharp depth discontinuities along object boundaries. Moreover, the model can infer metric scale information by recognizing familiar objects, like cars or people, and their

relative sizes, even in the absence of explicit camera intrinsics. The attention layers implicitly learn a form of camera calibration by observing structural patterns in the image, enabling the model to estimate the focal scale from context.

Apples Depth Pro is designed to produce high-resolution depth maps with fine details in a single forward pass, making it suitable for real-time applications. As illustrated in Figure 3.3, the network consists of two complementary encoder branches followed by a specialized decoder. The first branch is a multi-scale, patch-based ViT encoder that processes overlapping patches of 384×384 pixels extracted from a high-resolution input of 1536×1536 pixels. The same encoder weights are shared across multiple scales, which encourages the learning of scale-invariant features. The patch-wise features are then reassembled into feature maps at each scale. Complementing the patch-based encoder, a second, image-level encoder processes a downsampled version of the entire image to capture global contextual features. This branch effectively anchors the depth estimation by providing a global representation of the scene, which is later fused with the local features in the decoder.

The decoder, similar to the one proposed in Dense Prediction Transformers (DPT) by Ranftl et al. [10], performs similar feature fusion. It upsamples and merges the reassembled feature maps from each scale with the global features, progressively constructing a high-resolution depth map that is both locally detailed and globally coherent. In addition, a focal length estimation head predicts the horizontal field-of-view from the image, thereby enabling the conversion of the inverse depth map into a metric depth map.

This architecture was chosen to meet demanding requirements: outputting a 2.25-megapixel depth map (1536x1536) with metric scale, in 0.3 seconds on a standard GPU [3].

We used this model for zero-shot metric monocular depth estimation while ignoring the focal length produced.

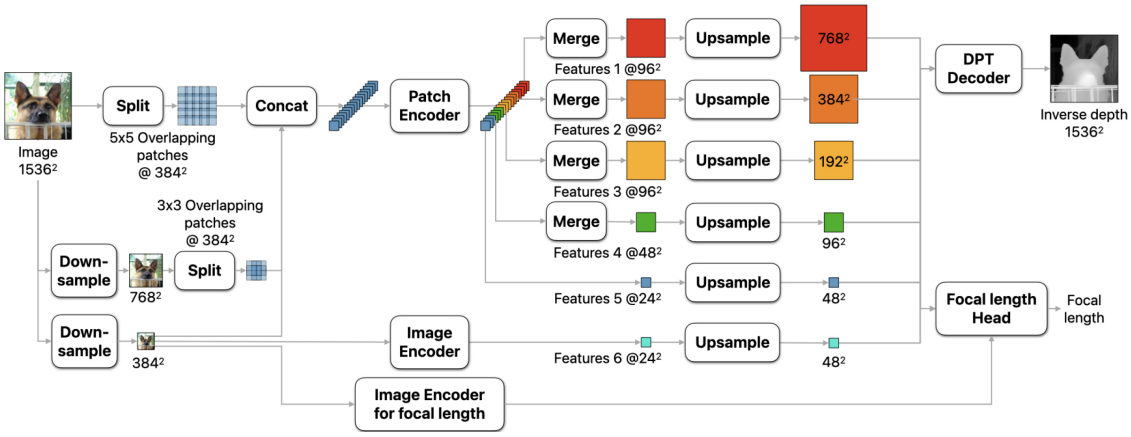

Figure 3.3. Overview of the network architecture. Adapted from [3].

3.2.4 Evaluation Setup

To assess the agent relocalization pipeline, we evaluated each component independently.

For the evaluation of object detection using YOLO, the original dataset was partitioned into three subsets: training (75%), validation (15%), and testing (10%). Figure 3.4 provides representative examples of these splits.

For depth estimation evaluation, we collected a dataset consisting of 32 images captured at varying zoom levels, in which we recorded the 3D positions of the agents relative to the camera. In 6 of these images, both agents are present, allowing for an assessment of relative accuracy. Figure 3.5 presents a selection of examples.

Our depth estimation methods operate in a zero-shot setting; therefore, this dataset is used exclusively for testing. Before applying any distance estimation techniques, we must extract the bounding boxes: 3.2.2 requires them to determine side lengths, while 3.2.3 uses them for positioning. To maximize the number of correctly labeled samples for evaluation, we applied our pretrained YOLO model and manually annotated any missing ones.

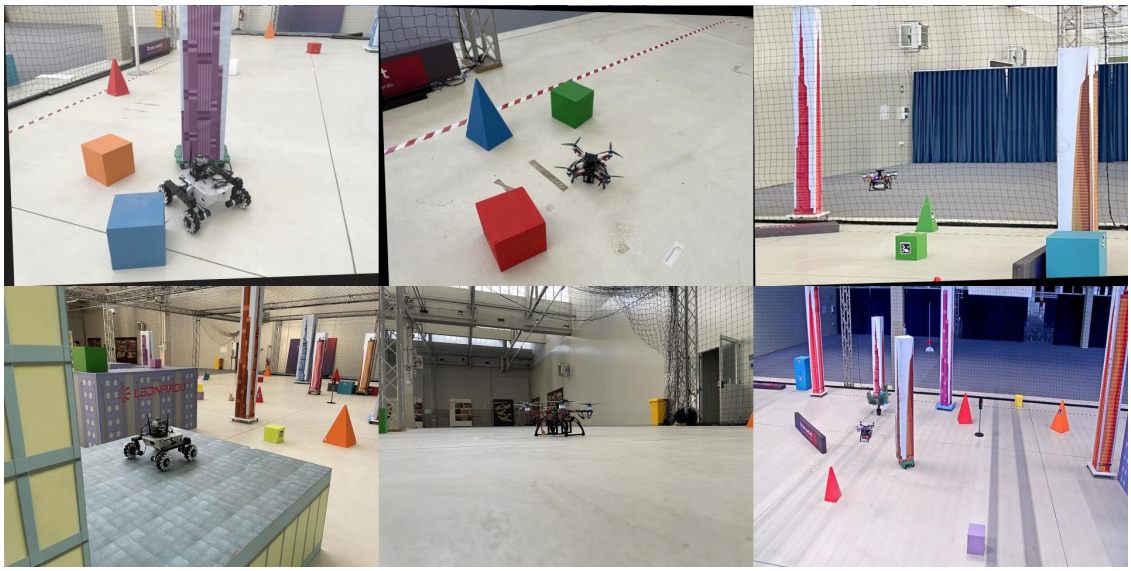

Figure 3.4. Samples from the object detection dataset. The top row illustrates images from the training set with data augmentation, while the bottom row displays test set images without augmentation.

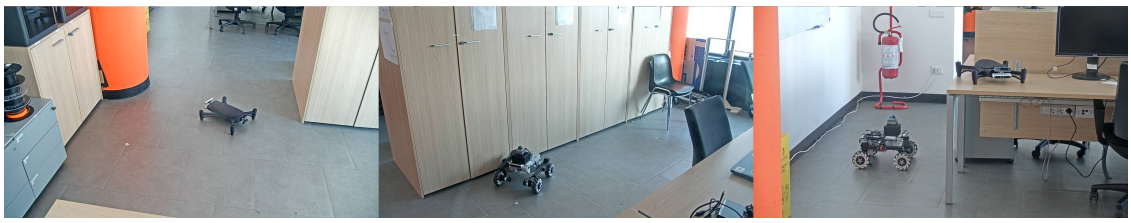

Figure 3.5. Samples from the test dataset used for depth estimation.

Chapter 4

Results

This chapter presents the experimental outcomes of the proposed approach for relocalizing autonomous agents using a PTZ camera and monocular depth estimation. The results are divided into two main parts: the camera calibration process under varying zoom levels and the relocalization pipeline, which includes object detection and distance estimation.

4.1 Camera Calibration with Zoom Lenses

Intrinsic parameters were estimated using the standard checkerboard calibration method, implemented via OpenCV's calibration tools. The calibration was performed across discrete zoom levels ranging from 1× to 10× to account for the varying internal geometry of the PTZ camera under different focal settings.

As shown in Figure 4.1, the focal length, computed as the average of the focal lengths along the x and y axes, exhibits a linear trend with increasing zoom levels. The observed slope aligns with the focal length at the base zoom level (1×), indicating a predictable scaling behavior. In contrast, the optical center shows a more irregular variation across zoom levels. However, since its contribution to the downstream task of distance estimation is relatively minor, this variability does not pose a significant concern.

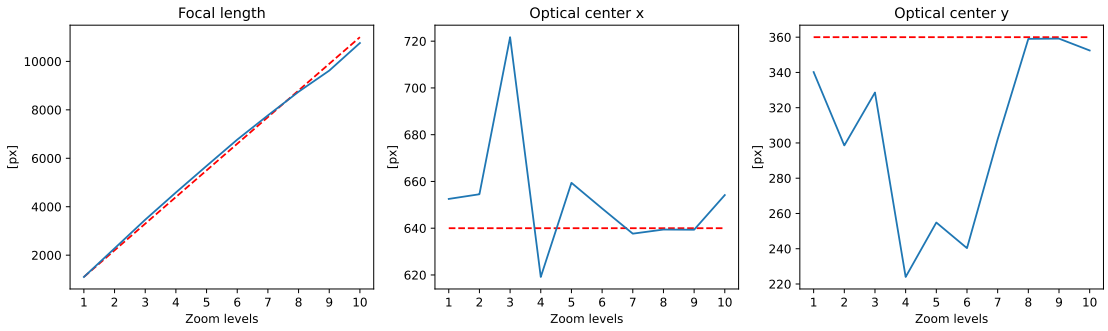

Figure 4.1. Evolution of intrinsic camera parameters across zoom levels. The red line indicates the ideal reference value.

To validate the calibration, the distance to a fixed ArUco marker placed at a known position of 7.9 meters was estimated using the computed intrinsic parameters. Table 4.1 summarizes the estimated distances at each zoom level, and the trend is visualized in Figure 4.2.

Zoom Level	Estimated Distance [m]
1×	7.93
2×	8.11
3×	8.14
4×	8.07
5×	7.97
6×	7.90
7×	7.68
8×	7.71
9×	7.62
10×	7.76

Table 4.1. Estimated distances to the ArUco marker at varying zoom levels.

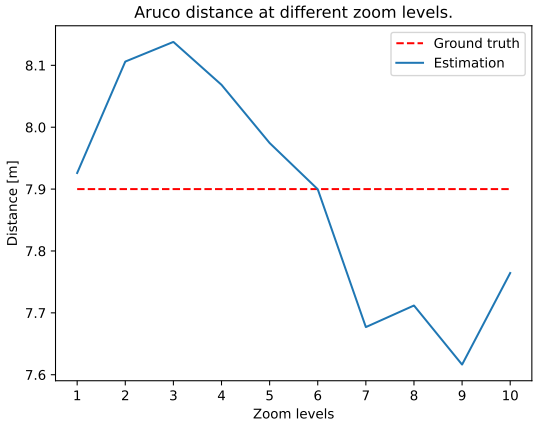

Figure 4.2. Estimated distances to the ArUco marker at varying zoom levels.

The maximum relative error in the distance estimation task is observed at zoom level 9×, where it reaches 3.5%, remaining within acceptable limits for the application.

An interesting observation is the correlation between focal length deviations and distance estimation errors. At lower zoom levels, the focal length tends to overshoot, resulting in an overestimation of distance. Conversely, at higher zoom levels, the focal length increase slows down, leading to a slight underestimation.

4.2 Agent Relocalization

4.2.1 Object Detection

A YOLOv8-small was fine-tuned on a custom dataset to detect the target autonomous agent within the camera's field of view.

The model was trained for 100 epochs using a batch size of 32. Optimization was performed with the Adam optimizer provided by the Ultralytics framework. During training, the performance of the model was monitored on a validation set using the following metrics:

- Precision: The ratio of true positive detections to the total predicted positives.
- Recall: The ratio of true positive detections to the total actual positives.
- mAP@0.5: The mean average precision computed at an Intersection over Union (IoU) threshold of 0.5.
- mAP@0.5:0.95: The mean average precision averaged over IoU thresholds ranging from 0.5 to 0.95.

While these metrics are presented as percentages in our reports, they are computed and store internally on a scale from 0 to 1.

Throughout the training process, the Ultralytics library automatically saves both the final model checkpoint and the checkpoint corresponding to the best performance. The selection of the best model is based on a fitness score, defined as:

$$\text{Fitness} = 0.1 \times \text{mAP@0.5} + 0.9 \times \text{mAP@0.5:0.95} \tag{4.1}$$

In our experiment the optimal model was identified at epoch 83.

The best-performing model was subsequently evaluated on an independent test set. Table 4.2 summarizes the key performance metrics.

Metric	Value
Precision	91%
Recall	93%
mAP@0.5	93%
mAP@0.5:0.95	82%

Table 4.2. Performance metrics on the test set, expressed as percentages.

These results demonstrate the robustness of the object detection module, with high precision and recall indicating reliable detection performance.

Figure 4.3 displays the precision-recall curve and the F1-confidence curve, each serving a distinct role in evaluating the models performance.

The precision-recall curve illustrates the trade-off between precision and recall as the confidence threshold varies. A high precision indicates that when the model predicts a detection, it is likely to be correct; a high recall ensures that most of the true instances are

detected. The curve helps to visualize how these metrics change with different threshold settings, for example, lower thresholds often yield higher recall but lower precision due to an increased number of false positives, while higher thresholds typically improve precision at the expense of recall. This detailed view is especially valuable in imbalanced scenarios, where the number of positive samples is much lower than negatives, by focusing on the models ability to correctly identify the target class.

In contrast, the F1-confidence curve plots the F1 score, defined as the harmonic mean of precision and recall, against varying confidence thresholds. The peak of the F1-confidence curve signifies the threshold at which the model achieves the best compromise between precision and recall, thereby indicating the optimal operational point for the detection task.

In addition to these curves, the confusion matrix shown in Figure 4.4 provides a complementary perspective on the model's performance. This matrix breaks down the distribution of true positives, false positives, and false negatives for each class, offering an intuitive visualization of classification accuracy.

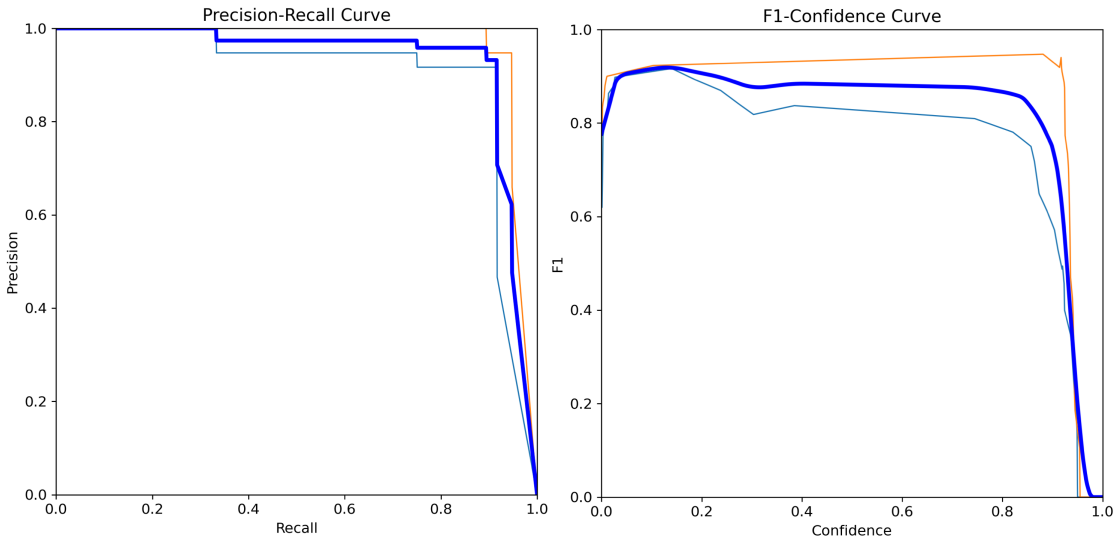

Figure 4.3. Precision-Recall and F1-Confidence curves. The light blue curve represents the drone, the orange curve the rover, with the blue highlight indicating the average performance.

Accurate bounding box detection is essential for the application of our distance estimation techniques. To ensure that the object detection model generalizes effectively to the specific dataset used for depth estimation, we also evaluated its classification performance on it. The resulting confusion matrix, presented in Figure 4.5, offers a detailed breakdown of the models predictions on the depth dataset.

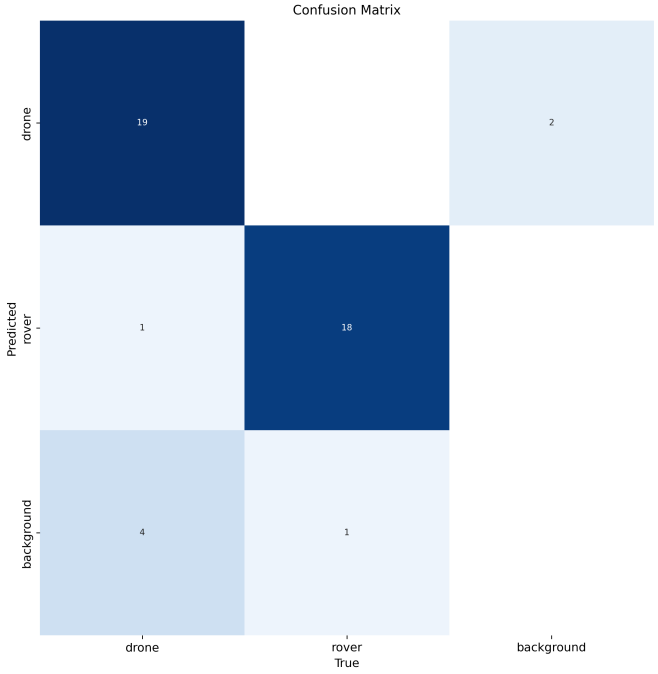

Figure 4.4. Confusion matrix on the object detection test set.

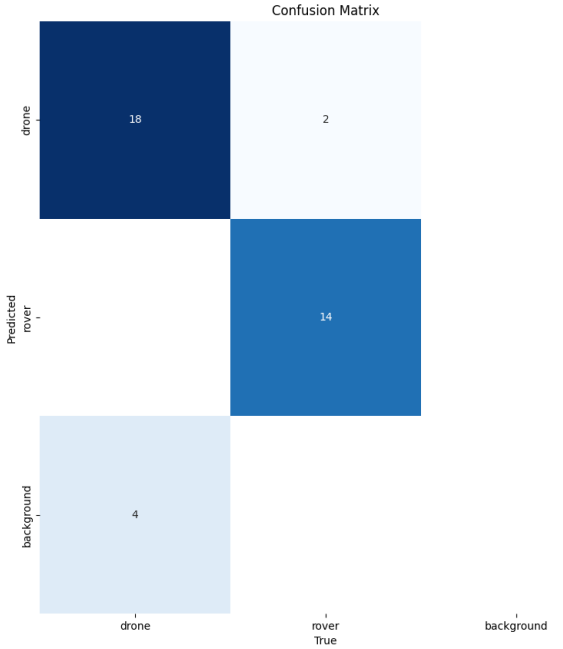

Figure 4.5. Confusion matrix on the distance estimation dataset.

4.2.2 Distance Estimation

This section evaluates the performance of two distance estimation approaches: the Bounding Box Dimensions method and the Apple Depth Pro method. We analyze their accuracy using two error metrics: the total relative error and the rover-corrected drone distance (RCDD) relative error.

The total relative error quantifies the percentage deviation between the estimated and true distances across all object classes. The results are summarized in Table 4.3, which reports the mean error and the corresponding 95% confidence intervals over 38 detections.

Method	**Mean**	**95% CI**
Bounding Box Dimensions	-4.54%	± 6.90%
Apple Depth Pro	9.93%	± 4.27%

Table 4.3. Mean total relative error and 95% confidence intervals computed over 38 detections.

These results illustrates a classic example of the bias-variance tradeoff. The Bounding Box Dimensions method, grounded in integral geometry, is theoretically unbiased, which is confirmed by its confidence interval encompassing zero. However, this method exhibits higher variance, leading to less stable estimations. Conversely, the Apple Depth Pro method introduces a non-zero bias but compensates with a significantly lower variance. In many engineering scenarios, a small bias is often an acceptable compromise if it results in increased reliability and smoother performance.

The RCDD error metric accounts for the known absolute distance of the rover to refine the drone's estimated position. The corrected drone distance is computed as:

$$d'_{drone} = d_{drone} + e_{rover}, \tag{4.2}$$

where e_{rover} is the absolute error of the rover in that frame.
This correction relies on the following assumptions:

1. Both the rover and the drone are detected within the same frame;
2. The true distance of the rover to the camera is known;
3. Relative depth estimation is more accurate than absolute depth estimation.

All assumptions are satisfied in this study:

1. The RCDD correction is applied only to frames where both the rover and drone are detected, only 6 out of 32 total samples;
2. The rover's ground-truth distance is available for evaluation, simulating real-world deployment where onboard LiDAR enables highly accurate localization through LIO algorithms;
3. Monocular depth estimation models inherently struggle with absolute scale but capture relative spatial relationships with higher accuracy.

Method	Mean	95% CI
Bounding Box Dimensions	21.49%	± 13.63%
Apple Depth Pro	-6.70%	± 12.63%

Table 4.4. Mean RCDD relative error and 95% confidence intervals over 6 samples. The Bounding Box Dimensions method shows degraded performance in relative depth refinement.

Table 4.4 presents the RCDD relative errors and confidence intervals.

As shown, the Bounding Box Dimensions method performs poorly in this setting. This is expected, as it fails to capture relative spatial consistency, thus violating the third assumption, due to its reliance on pure geometry without contextual cues.

In contrast, the Apple Depth Pro method reduces both bias and variability in this context. Its mean error is closer to zero, and its lower standard deviation reflects a better ability to model fine-grained spatial relationships. Comparing standard deviation values between the total error (13%) and the RCDD setting (12%) confirms this improved relative consistency.

These findings reinforce that, for applications requiring stable and locally accurate estimations, a method with low variance, even if slightly biased, is often preferable to one that is theoretically unbiased but highly variable.

From a computational perspective, the Apple Depth Pro model processed a single frame in approximately 0.35 seconds on an NVIDIA 2080 Super GPU. These results are in line with Apple's claims of obtaining a high-resolution depth map in around 0.3 seconds on consumer hardware. Throughput can be increased by processing multiple frames in batches, but this comes at the cost of additional latency, as the system must wait to accumulate a full batch before inference. This introduces a trade-off between per-frame latency and overall system efficiency.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis presented a comprehensive framework for the relocalization of autonomous agents through the integration of monocular depth estimation and PTZ camera calibration. Motivated by the challenges encountered during the Leonardo Drone Contest, the work focused on addressing critical issues in both hardware calibration and algorithmic depth estimation to facilitate robust multi-agent operations in dynamic environments.

A two-way approach was adopted: first, a rigorous camera calibration procedure was implemented to account for non-linear variations in intrinsic parameters across a wide range of zoom levels; second, a dual-framework method for agent relocalization was developed by combining the strengths of YOLOv8 for object detection with a Vision Transformer-based depth estimation technique, in this work Apple Depth Pro. The calibration experiments, conducted using standard checkerboard methods and validated via ArUco markers, demonstrated a predictable linear evolution of the focal length and highlighted the acceptable variation in the optical center. The estimated distances maintained low relative errors across various zoom settings, confirming the reliability of the calibration procedure in mapping the three-dimensional world to the two-dimensional image plane.

In the relocalization phase, the object detection module achieved high precision and recall, as evidenced by the performance metrics, while the depth estimation strategies were evaluated using two error measures: the total relative error and the rover-corrected drone distance (RCDD) relative error. The experiments underscored a classic bias-variance tradeoff. The Bounding Box Dimensions method, although theoretically unbiased, suffered from higher variance, leading to inconsistent distance estimations. In contrast, the Apple Depth Pro method, despite introducing a small bias, yielded lower variance and improved local accuracy. These findings indicate that in practical applications, where stability and real-time performance are critical, a method with low variance, even if slightly biased, is preferable.

Overall, the integrated system demonstrated robust performance for the relocalization task, achieving accurate distance estimation under various operating conditions. This work not only addresses the technical challenges associated with PTZ camera calibration

and monocular depth estimation but also paves the way for deploying such systems in real-world scenarios, including warehouse logistics, precision agriculture, and smart city monitoring.

Additionally, a dedicated framework was developed to easily operate the Axis M5525-E PTZ Network Camera, integrating its onboard capabilities with the proposed algorithms. This framework streamlined the process of sending commands, receiving status updates, and managing video streams, thereby facilitating seamless control and real-time data acquisition from the camera.

5.2 Future Work

Despite the promising results, several avenues for further research and development remain open. Future work could address the following aspects:

Asynchronous Operation The current implementation of the system operates synchronously, which can limit throughput and increase latency during high traffic or when handling multiple concurrent requests. Transitioning to an asynchronous architecture would enable parallel processing, reduce response times, and improve scalability. This modification is expected to better accommodate high-frequency requests, manage heavy traffic loads, and achieve lower latency, thereby boosting overall system performance in demanding operational scenarios, albeit at the cost of a more complex architecture that requires sophisticated concurrency management, robust error handling, and intricate debugging due to non-deterministic execution patterns.

Dynamic Calibration Techniques The calibration procedure presented in this thesis assumes a controlled environment and fixed parameters for discrete zoom levels. Future research could investigate advanced calibration techniques that accurately map all intermediate zoom levels, potentially through neural network-based interpolation or other novel approaches. Such methods would enable continuous calibration over the full zoom range, adapting intrinsic parameters in real time to account for environmental changes and wear-induced variations in camera hardware. This dynamic calibration would significantly enhance robustness and accuracy, particularly in long-duration deployments where hardware characteristics may shift over time.

Advanced Machine Learning Techniques The success of the YOLOv8 and Vision Transformer models in this thesis suggests that advanced machine learning techniques hold significant promise for autonomous agent localization. Future studies could investigate the benefits of emerging architectures, such as transformer-based object detectors [18] and self-supervised learning methods, to further enhance both detection accuracy and depth estimation reliability. Additionally, the exploration of domain adaptation techniques could enable the system to generalize better across different environments and object classes.

Scalability and Multi-Agent Coordination Another promising direction is the extension of the presented framework to multi-agent systems. While this thesis primarily focused on relocalizing a single autonomous agent, real-world applications often require coordinated localization among multiple agents. Future research could focus on developing algorithms that manage the interactions between several agents, optimizing the use of shared camera resources, and ensuring robust communication between agents to maintain overall system coherence.

Application-Specific Customizations Tailoring the system to specific application domains presents an opportunity for further enhancement. For instance, in warehouse logistics, the integration of real-time inventory management and obstacle avoidance systems would be highly beneficial. Similarly, in precision agriculture, adapting the system for variable terrain and environmental conditions would extend its utility. These customizations would involve not only algorithmic improvements but also adjustments in hardware setup and sensor deployment.

5.3 Summary of Contributions

In summary, this thesis makes the following key contributions:

- Development of a robust calibration methodology for PTZ cameras, capable of handling dynamic zoom variations and ensuring accurate mapping between 3D space and 2D image projections at all zoom levels.
- Integration of state-of-the-art object detection, YOLOv8, and metric monocular depth estimation, Apple Depth Pro, methods to accurately relocalize autonomous agents.
- Creation of a dedicated framework to operate the Axis M5525-E PTZ Network Camera, enabling seamless control and real-time data acquisition.
- Detailed experimental evaluation and error analysis, which reveal a nuanced tradeoff between bias and variance in different distance estimation approaches.

The insights gained from this research not only demonstrate the feasibility of using PTZ cameras for autonomous agent relocalization but also lay the groundwork for future innovations in the field. Continued exploration and refinement of the techniques discussed herein will undoubtedly contribute to the advancement of autonomous systems in complex, real-world environments.

Bibliography

[1] Mehrnaz Farokhnejad Afshar, Zahra Shirmohammadi, Seyyed Amir Ali Ghafourian Ghahramani, Azadeh Noorparvar, and Ali Mohammad Afshin Hemmatyar. An efficient approach to monocular depth estimation for autonomous vehicle perception systems. *Sustainability*, 15(11):8897, 2023.

[2] L. Agapito, A. Bartoli, R. Evans, and S. Knowles. Self-calibration of rotating and zooming cameras. *International Journal of Computer Vision*, 45(2):107–127, 2001.

[3] Aleksei Bochkovskii, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R. Richter, and Vladlen Koltun. Depth pro: Sharp monocular metric depth in less than a second, 2024.

[4] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[5] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3828–3838, 2019.

[6] E. Hayman and D. W. Murray. The effects of translational misalignment when self-calibrating rotating and zooming cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):1015–1020, 2003.

[7] Xueyan Oh, Ryan Lim, and U.-Xuan Tan. Marker-based localization system using an active PTZ camera and CNN-based ellipse detection. *IEEE/ASME Transactions on Mechatronics*, 28:1984–1992, 2023.

[8] OpenCV. Camera calibration and 3d reconstruction, 2018.

[9] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation, 2024.

[10] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12179–12188, 2021.

[11] René Ranftl, Konstantinos Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1623–1637, 2022.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[14] Rolf Schneider. *Convex Bodies: The Brunn–Minkowski Theory*. Cambridge University Press, 2nd edition, 2014.

[15] K.-T. Song and J.-C. Tai. Dynamic calibration of pan–tilt–zoom cameras for traffic monitoring. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1091–1103, 2006.

[16] Ultralytics. Ultralytics yolov5, 2020.

[17] Ultralytics. Explore ultralytics yolov8, 2023.

[18] Ultralytics. Yolo12: Attention-centric object detection, 2025.

[19] Halil Utku Unlu, Phillip S. Niehaus, Daniel Chirita, Nikolaos Evangeliou, and Anthony Tzes. Deep learning-based visual tracking of UAVs using a PTZ camera system. In *Proceedings of the 45th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pages 638–644, 2019.

[20] R. Willson and S. Shafer. What is the center of the image? In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 670–671, 1993.

[21] R. G. Willson. *Modeling and Calibration of Automated Zoom Lenses*. PhD thesis, Carnegie Mellon University, 1994.

[22] C. Zhang, Y. Wang, X. Li, and H. Zhang. Deepptz: Deep self-calibration for ptz cameras. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 1040–1049, 2020.

[23] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from monocular video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6612–6621, 2017.