POLYTECHNIC OF TURIN

MASTER's Degree in MECHATRONIC ENGINEERING

Politecnico di Torino

MASTER's Degree Thesis

CAMERA LIDAR FUSION FOR UNMANNED AERIAL VEHICLE DETECTION

Supervisors

Prof. MARCELLO CHIABERGE

Dr. SEDAT DOGRU

Prof. LINO MARQUES

Candidate

COSIMO MARIA LEUCI

ACADEMIC YEAR 2024-2025

Contents

List of Tables

List of Figures

Acronyms

EKF Extended Kalman Filter.

FCNs fully convolutional neural networks.

FoV Field of View.

HMM Hidden Markov Model.

HSV Hue-Saturation-Value.

KF Kalman Filter.

LiDAR Light Detection and Ranging.

PTU Pan-Tilt Unit.

RGB Red-Green-Blue.

UAV Unmanned Aerial Vehicle.

YOLO You Only Look Once.

Abstract

Cameras are very popular for drone detection due to their low cost and ease of use. The feature-rich images obtained from cameras enable the classification of drones with a relatively high success rate. However, cameras have limitations when it comes to accurate localization, providing only a rough estimate of the target Unmanned Aerial Vehicle (UAV)'s orientation.

To address this limitation, we propose a method that fuses camera and LiDAR measurements to enhance the accuracy of UAV localization. Lidar systems, with their precise distance measurements, complement the visual data from cameras by providing accurate spatial information. This fusion allows for more precise detection and tracking of UAVs, combining the strengths of both sensors.

In this study, an algorithm that integrates camera and LiDAR data is developed, leveraging the complementary nature of the two sensor types. The camera provides detailed visual information that helps in drone detection, while the LiDAR offers precise 3D spatial data that improves localization accuracy. Our method involves synchronizing the data from both sensors and applying advanced fusion techniques to achieve a more reliable and accurate detection system.

The experimental setup involved a drone, a camera and LiDAR sensors mounted on Pan-Tilt Units (PTUs). This approach was proven in an outdoor environment to validate its effectiveness and performance. The results of our experiments show that this fused approach significantly enhances the localization accuracy of UAVs compared to

using camera data alone. This improvement can have important implications for various applications, including security, surveillance, and airspace management, where accurate UAV detection and tracking are crucial.

By combining the high success ratio of camera-based classification with the precise localization capabilities of LiDAR, our fusion approach offers a robust solution for UAV detection. This method not only try to overcome the individual limitations of each sensor but also provides a comprehensive and efficient system for real-time UAV monitoring.

Chapter 1

Introduction

1.1 Context and Motivation

Unmanned Aerial Vehicles (UAVs), commonly known as drones, have revolutionized various sectors through their versatility, efficiency, and advanced technological capabilities. Initially developed for military purposes, UAVs are now widely used in numerous civilian applications, providing significant benefits in terms of efficiency, safety, and cost-effectiveness and usage range from agriculture and environmental control to security and logistics [6, 41]. According to Kapustina *et al.* [24] the global drone market is rapidly growing, with a projected tripling by 2025, and with USA, China, and France being the main producers. The cost of professional drones is increasing and the drone market is set to grow further, with greater integration between component manufacturers like Intel, Qualcomm, and DJI and service providers such as Amazon, Google, and Microsoft,

and future consolidation among a few large IT conglomerates like Google, Apple, and Microsoft. Trends indicate an increasing importance of software solutions and services offered by these companies.

Figure 1.1: Different use of drones [22] [31]

In the agricultural sector the UAVs are used for enhancing cultivation processes by performing monitoring and spraying missions. They optimize the efficiency of pesticides and fertilizers, detect possible pests and diseases in a timely manner, and facilitate the spraying procedure [35].

UAVs play a crucial role in disaster management by providing real-time data and

high-resolution imagery of affected areas. They are used for search and rescue operations, assessing damage, and delivering supplies to inaccessible regions [3].

Zhi *et al.* [49] analyze the possible security and privacy issues of UAVs. They rely heavily on sensors for localization and flight attitude determination, making them susceptible to attacks that can lead to incorrect decisions or crashes. Communication links between UAVs and Ground Control Stations (GCS) are at risk of interception and hijacking if not adequately secured. Furthermore, multi-UAV applications depend on stable ad-hoc networks, which have inherent security weaknesses. Additionally, UAVs pose privacy risks through aerial photography, as photos may inadvertently disclose sensitive information such as locations and timestamps when shared on social media. Addressing these issues requires ongoing research into sensor security, communication protection, and secure networking methods.

Although role of the drones in the world is becoming more and more important in helping people, unfortunately they are also used for targeting civilians, surveillance abuse and disturbing air-travel. Improper use of UAVs has triggered research on their detection, resulting in different detection systems using different sensors including but not limited to cameras and LiDAR.

LiDAR has been used in different scenarios, such as a ground-based UAV tracking utilizing a solid-state LiDAR [5] with adjustable frequency and varying FoV(Field of View), sparse detection with the aim of not only detection but also motion estimation and active tracking of the target [13], or LiDAR-generated image technologies for real-time tracking fusing the LiDAR-images and the point clouds from a single LiDAR sensor

for tracking UAVs without prior knowledge of their initial position [39].

There are also studies focused on air-borne UAV detection approaches, such as [12] which use an air-borne 2D only millimeter wave radar to detect and track drones, and [44] which exhibits minimal detection latency when combined with a multi-target tracker.

Detection and tracking are also performed using camera sensors, but object detection is still difficult in videos captured by a moving camera, since camera motion and object motion are mixed [20].

In order to detect and track using moving cameras, optical flow technique has been used, focusing on the different displacement of the target in the image sequence [21].

1.2 Objective

This work focuses on the fusion of two types of sensor for UAV detection: LiDAR and camera. The LiDAR technology allows to measure the distances of the targets in three-dimensional space. The camera allows for the continuous scanning of the environment in a relatively cheap manner, facilitating also the identification of the drones in the image. In this work the objective is to obtain a more robust method for the detection and tracking of drones.

LiDAR typically offers higher resolution and accuracy in detecting and mapping objects due to its shorter wavelengths and precise laser scanning capabilities, which allow creating detailed 3D maps of the surroundings.

The use of 3D lidars, like the Velodyne VLP-16, faces challenges at greater distances.

While accurate point clouds can be obtained for nearby drones, detection becomes sparse at longer ranges, often providing only a few data points when the drone is aligned with the scanning beams. Despite these limitations, lidar remains a valuable tool for detecting drones at significant distances and higher altitudes. A primary challenge is the dynamic background subtraction, as elements like water surfaces or moving trees may interfere with the process. This issue results in false clusters, but fixing the camera and using lidar motion has shown better results in handling these challenges.

1.3 Contributions

This thesis, conducted as part of the Erasmus program at Instituto de Sistemas e Robótica (ISR) in Coimbra, introduces a new research in the field of sensor fusion for aerial target detection. Although sensor fusion of LiDAR and cameras is common in the literature with examples on vehicle detection [47], we are not aware of any study fusing the two for UAV detection. The primary goal of this project was to fuse two specific sensors for detection and tracking of UAVs, a combination that has no precedent in existing literature. The detailed methodologies and findings of this research were published in the 7th Iberian Robotics Conference (ROBOT 2024) "Camera LiDAR fusion for Unmanned Aerial Vehicle detection" [26].

1.4 Document Structure

This dissertation is organized as follows:

- **Chapter 2** presents the state of art of the topic. It includes multiple drone detection methods and a section on tracking.
- **Chapter 3** describes in detail Kalman Filters and how clustering and background subtraction work in pointclouds, and then in images.
- **Chapter 4** explains with a clear block diagram the full pipeline of the project.
- **Chapter 5** illustrates the setup, the experiments and the results obtained.
- **Chapter 6** provides the conclusion.

Chapter 2

Related Work

2.1 Drone detection using LiDAR

Several approaches use a sensor-based LiDAR to detect drones. LiDAR has become a crucial instrument in robotics and autonomous system. Catalano *et al.* [5] proposed a method in which the LiDAR frame integration time is dynamically adjusted based on the UAV's distance and speed, obtaining a better accuracy and persistence. Hammer *et al.* [18] investigated the feasibility of drone detection and they conducted experiments using a platform equipped with two Velodyne VLP-16 and two Velodyne HDL-64 LiDARs. Dogru *et al.* [13] focused on a method to calculate the probability of detection under different scenarios with a LiDAR-turret system, introducing an approach aimed at achieving a more uniform distribution of laser pulses, thereby the probability of detecting target increases compared to conventional LiDAR-turret setups.

Vrba *et al.* [44] introduces a new method for detecting and localizing flying objects using 3D LiDAR on autonomous aerial vehicles. The approach employs a 3D occupancy voxel map and a multiple hypothesis tracker, offering superior localization accuracy, robustness, and range. It operates independently onboard the UAV, effectively suppressing false positives and performing well in various environments. Validated through experiments, it outperforms existing methods, though further research is needed for complete UAV detection solutions in aerial interception tasks.

2.2 Drone detection using camera

2.2.1 Non-flying Object Detection

Camera-based object detection has been researched extensively in literature. Bouwmans et al. [7] elaborated an important review analyzing the main challenges of detection using a moving camera. In the literature background subtraction is a common method used in the pipeline of object detection with a camera, however the process becomes more complicated when used for images taken by moving cameras.

A low frame rate often does not allow to distinguish clearly background and foreground changes. In the method without additional sensor proposed by Wu-Chih Hu et al. [20] the feature points in the frames are detected using a modified Harris corner detector [37]. The challenge lies in distinguishing between camera motion and object motion. The proposed method first identifies feature points in the video frames and classifies them as foreground (moving objects) or background features. Moving objects are detected

by integrating these foreground feature points with regions obtained through an image difference scheme.

Jiang et al. [23] proposed a Multi-Camera 3D-object detection in autonomous driving using an onboard camera (2D images) from the ego car's perspective. They argue that traditional Cartesian coordinates are inappropriate for the wedge-shaped perspective of onboard cameras and propose using a Polar coordinate system instead. Their model incorporates a cross-attention based decoder designed to manage irregular Polar grids and a multi-scale representation learning strategy to address object scale variations.

Viola [34] implemented a fast object detection framework to achieve robust real-time detection. The framework uses an integral image representation for rapid feature computation, AdaBoost-based feature selection for efficient classification, and a cascade of classifiers to focus processing on promising regions. This approach enables real-time face detection at 15 frames per second on standard hardware, with performance comparable to the best systems available.

Zhang et al. [48] presented a framework for object detection, tracking, and recognition in non-overlapping multi-camera surveillance networks. It introduces a new detection algorithm using mean shift (MS) segmentation enhanced by depth information from stereo vision, which improves object masks. The framework also includes a superpixel-based object recognition algorithm that utilizes both shape and color features, outperforming traditional methods.

The survey of Yazdi et al. [46] analyzes the latest methods of moving object detection in video sequences captured by a moving camera. It categorized the methods into four

main types: background modeling, trajectory classification, matrix decomposition, and object tracking. The study highlights that recent advancements include using deep neural networks, efficient keypoint detectors, and improved camera motion models.

2.2.2 Flying Object Detection

Birch and Woo [1] conducted an evaluation of the visual detection of drones at various wavelengths within the visible and infrared spectra, finding that visual band analysis is more efficient with a uniform background.

Rozantsev et al. [36] deal with the challenge of using a single camera to detect flying objects. The proposed method uses image changes caused by camera movement to distinguish flying objects from those that are part of the static background. This technique analyzes the variations in the scene and the apparent motion of objects to facilitate their identification and tracking.

A real-time system designed to detect and track multiple moving objects using a controlled pan-tilt camera platform [21] can be also applied. The system employs an input/output Hidden Markov Model (Hidden Markov Model (HMM)) [21] within a spherical camera coordinate system to effectively manage the relationship between targets and the camera. A hybrid detection algorithm that combines color and optical flow information is proposed for fast and accurate target detection.

The fusion between camera and lidar has been tested for the collision avoidance system [39], and by fusing camera outputs in the form of lidar detections, more accurate and robust detections for overlapping objects are achieved.

2.3 Object detection fusing camera and LiDAR

The fusion between camera and LiDAR has been tested for collision avoidance system [45],and by fusing camera outputs in the form of LiDAR detections, more accurate and robust detections for overlapping objects are achieved. In the work of Wei *et al.* [45] object detection from cameras involves both classification and localization, which is challenging due to unknown number, positions, and sizes of objects. Two main Deep Learning-based approaches were used: regional proposal, with separate networks; and a single network (You Only Look Once (YOLO)) to predict regions and classify them. This implies less accuracy for small object. In the work of Caltagirone *et al.* [4], a fusion fully convolutional neural network (FCN) was been developed to integrate camera images and LiDAR point clouds for carrying out road detection. In this case the fusion is implemented with a strategy called cross fusion, which allows the FCN to learn from the data where to integrate the LiDAR and camera information. Liu *et al.* [29], to overcome the inaccuracy of LiDAR–camera fusion object detection methods, proposed a new deep neural network (namely FuDNN), that improves the performance of LiDAR-camera fusion for autonomous driving.

The article by Li *et al.* [27] introduced an advanced approach to integrating data from LiDAR and camera to improve the detection of three-dimensional objects. The work presented a deep fusion framework combining detailed spatial information provided by LiDAR with rich textured details of camera images. This combination allowed more precise and robust object detection. The proposed framework is designed to extract and

integrate the multi-modal characteristics of the two types of data, being particularly effective in detecting objects of varying size and distances.

In this chapter we have examined several advanced approaches for merging data from cameras and LiDAR to improve the detection of three-dimensional objects. The work analysed includes collision avoidance systems, deep learning-based approaches such as regional proposal networks and YOLO, as well as fully convolutional neural networks (FCNs) for road detection.

However, it is important to note that none of these studies focus on the detection of flying objects. Although these methodologies demonstrate significant progress in the fusion of data for the detection of ground-based objects and with various sizes, they do not specifically address the challenges associated with the detection and tracking of flying objects,such as drones or unmanned aerial vehicles. This is an area potentially under explored and could benefit from further research and development.

2.4 Drone detection using other sensors

2.4.1 Radio-frequency (RF) Sensors

One of the techniques to detect drones is the use of other drones, focusing in particular on locating moving targets that emit radio signals. Dressel *et al.* [14] described how drones equipped with radio-frequency (RF) sensors detect radio emissions from target drones. The seeker drone carries two antennas and two commodity radios. These sensors collect data on radio waves, which are then processed to estimate the position of the target

drone in real time. Exploiting the radio-frequency (RF) communication of the drones with a base station or operator, Nguyen *et al.* [32] proposed a system using commercial RF receivers to monitor the air for signals emitted by drones. The approach is based on measuring the strength of the received signal (RSSI, Received Signal Strength Indicator) and analyzing signal displacement patterns to determine the presence and location of a drone. Also Shi *et al.* [38] used radio-frequency, but combined with radar sensors, optical and infrared cameras. Each technology has its own specific role: radars are used for long-range detection and drone tracking; optical and infrared cameras provide visual identification and classification capabilities; radio frequencies help to detect drone command and control signals.

2.4.2 Radar Sensors

Radar is another type of sensor for drone detection. A specific challenge of detection with radar is related to small drone size and the characteristics of their radar cross section (RCS), which can be very small and make detection difficult, especially at long distances [16]. Hoffman *et al.* [19] used a stationary multi-static radar consisting of three identical nodes operating in the S-band (2.4 GHz) for the detection of small UAVs. This approach uses time signatures and micro-Doppler of drones to improve detection; the technique applied allows acquiring data from multiple angles, reducing RCS. De Quevedo *et al.* [10] used an X-band (8.75 GHz) ubiquitous static radar for in-flight drone detection, with a range of up to 2 km and a resolution of 0.878 metres. X-band radars, due to their

higher frequency, offer better resolution than S-band radars, but their range can be limited by the greater attenuation of the long-range signal. Dogru *et al.* [11]- [12] proposed the use of active radar mounted on aerial platforms for the detection, to track and pursuit target drones. This technique has proved effective in outdoor environments, thanks to the ability of airborne radar to provide wider coverage and overcome the constraints imposed by ground obstacles, offering more flexibility and a wider range of action compared to fixed radars. The work [30] highlights the critical role of detect-and-avoid capabilities for Unmanned Aircraft Systems (UAS), particularly in applications like transportation and surveillance. It details the design, implementation, and experimental outcomes of a UAS-borne radar system aimed at detecting drones, emphasizing its use in enhancing navigation safety and airborne surveillance, particularly for security in restricted airspace. The system demonstrated exceptional performance, detecting drones like the DJI Phantom 4 and DJI Matrice 600 Pro at ranges up to 440 and 500 meters, respectively, surpassing previous systems by tenfold. The radar utilizes a frequency modulated continuous-wave (FMCW) architecture with fast-chirp modulation, offering the highest detection range reported for UAS-borne sensors. Radar is a sensor that uses radio waves to detect objects, providing long-range detection and robustness in various weather conditions, but with lower resolution. LiDAR uses laser pulses, offering higher resolution and accuracy.

2.4.3 Passive Radar

Another radar sensor is the passive radar [43]. Passive radars do not emit active signals but detect radio waves emitted from other sources, such as the RF signals of the drones themselves. This approach may be beneficial in terms of reduced environmental impact and increased discretion, but is limited by the need for appropriate signal emission sources and the complexity of separating signals of interest from background noise.

2.4.4 Optical and Infrared Sensors

Christnacher *et al.* [9] integrated a range-gated active imaging system with an array of acoustic antennas to capture the noise of the drones. This imaging system includes a SWIR camera mounted on a pan and tilt mechanism and uses an infrared laser to illuminate the target actively.

2.5 Object tracking

Tracking involves estimating the trajectory or state of a moving object over time. This process is essential in various applications and is closely related to the detection of objects. Tracking systems utilize sensor data to continuously predict and update the position of the object. In literature there are a lot of works on this topic. Du textitet al. [15] explored methods of tracking objects using correlation filters and described the principles of correlation filters, which compare a model of the object with images to determine its position and analyses also traditional algorithms such as mean-shift and camshift, and

the use of Fourier transformations to optimize the calculation.

Gunjal et al. [17] proposed a method for tracking moving objects using the Kalman Filter. In this case the Kalman Filter is employed to estimate the location of a target in a video stream by processing frames individually. The approach involves several steps: converting video into frames, detecting targets within each frame using color recognition, and tracking their movement by calculating the coordinates of targets across frames. The Kalman Filter aids in predicting the target's position and correcting estimates based on new measurements, effectively managing errors and uncertainties in the tracking process. This study highlights the Kalman Filter's effectiveness in improving tracking precision and reducing errors compared to simpler methods.

Cho *et al.* [8] implemented an innovative real-time object tracking system that leverages particle filters to handle complex tracking scenarios. The authors address the limitations of traditional tracking methods that rely on linear models and Gaussian noise assumptions, which often fail in dynamic and noisy environments. Instead, they propose using particle filters, which are well-suited for managing non-linear and non-Gaussian characteristics of real-world tracking problems. The proposed system involves representing the state of the tracked object using a set of particles, which are weighted according to their likelihood of matching the observed data. This approach allows for more accurate and robust tracking, particularly in situations where the object's motion is highly variable or the measurements are noisy.

Boers *et al.* [2] applies a particle filter to handle nonlinear, non-Gaussian noise, integrating data over time for improved target detection. The particle filter approximates

the state probability density function using a set of particles, which helps in tracking weak targets in noisy environments. Detection is achieved through a likelihood ratio test based on the particle filter's output.

The aim of the study is to validate the model considering a short time interval, so the relation between the duration of the flight of the drone at constant speed and the update of the filter has a crucial role. The survey of *et al.* [28] points out that the time during which a constant speed model remains valid is strongly related to the speed at which the filter is updated. It is essential to balance the realistic motion and an appropriate rate, according to the conditions of the system.

Chapter 3

Theoretical Background

3.1 Clustering and Filtering point clouds

Firstly, my research focused on the use of LiDAR technology. To achieve this, I employed the Point Cloud Library, which offers a comprehensive range of mechanisms for filtering, clustering, and locating surfaces and objects. Initially I focused on the filter part necessarily to remove the outliers and noise from the overall point cloud and set the background.

3.1.1 Point cloud filtering and subtraction

Filtering a fundamental step in point cloud processing. To remove outliers and noise, a background point cloud is created by accumulating data over time. In order to accurately set the background, it was necessary to account for the fact that the LiDAR was in motion.

This was achieved by accumulating for a certain period of time the points to create a distinctive point cloud, which was then saved as the background. I subtract each new clouds with the saved background and then I applied some filter of the point cloud library.

The `PassThrough` filter is applied to delimit the region of interest along the three axes, ensuring that only relevant points are maintained. Using a k-dimensional tree(`kdTree`) technique, a set of k-dimensional points is stored in a space-partitioning tree structure, enabling efficient range searches and nearest neighbor searches. Nearest neighbor searches are essential when working with point cloud data, allowing for the identification of correspondences between groups of points or feature descriptors and the definition of the local neighborhood around specific points.

3.1.2 Point cloud clustering

The clustering step is performed using the library `EuclideanClusterExtraction`. This algorithm segments individual object clusters from a point cloud by leveraging a kd-tree for efficient nearest neighbor searches. It begins by creating a `KdTtree` representation for the input point cloud dataset P. An empty list of clusters C is initialized, along with a queue Q to keep track of points to be checked.

For each point $\boldsymbol{p}_i$ in the point cloud P, the point is added to the queue Q. The algorithm then processes each point in Q by searching for its neighbors within a sphere of radius r (where $r < d_{th}$, a threshold distance). For every neighboring point that hasn't already been processed, the point is added to Q.

Once all points in Q have been processed, the queue Q is added to the list of clusters

C, and Q is reset to an empty list.

It is created an object `EuclideanClusterExtraction` with a point type PointXYZ, since our point cloud is of type PointXYZ. We have set the parameters and variables needed for cluster extraction. It is important to pay special attention to the choice of value of `setClusterTolerance`: a value too low could lead to split an actual object into multiple clusters, while a value too high could merge several distinct objects into one cluster. Therefore, it is recommended to experiment and test different values to identify the one that best fits your dataset.

The clusters found must have a number of points between the values set with `setMinClusterSize` and `setMaxClusterSize`. Once the clusters are extracted from the point cloud, the indexes of each cluster are saved in a variable. To separate each cluster within the vector of indices, you need to iterate through the variable, create a new point cloud for each element and write all current cluster points into the new point cloud.

This procedure allows for the effective segmentation of objects in the point cloud, allowing better analysis and manipulation of data for subsequent applications.

3.2 Object detection in images

In this section, theoretical concepts related to image color representation, morphological operations, and blob localization are discussed. These techniques are fundamental in image processing for object detection and segmentation.

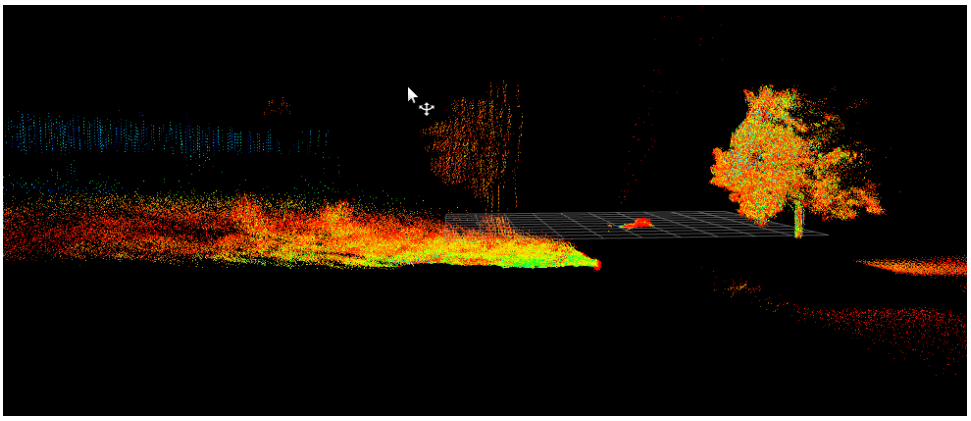


Figure 3.1: Point cloud in RVIz.

3.2.1 Image Color Representation

Color representation is a crucial aspect of image processing, as different color models provide distinct advantages depending on the application. The most commonly used models are RGB (Red, Green, Blue) and HSV (Hue, Saturation, Value). The image format is RGB(cube), color model based on the three primary additive colors:Red, Green and Blue. The shape of the representation is a cube(each axis corresponds to one of the three color).

To enable filtering better reflecting human perception HSV model(cylyndrical) is used. HSV represents colors with Hue, Saturation and Value and it's useful for filtering images by color ranges. This model is represented as a cylindrical coordinate system:

- Hue (H): Represents the type of color, ranging from 0 to 360 degrees.
- Saturation (S): Measures the intensity or purity of the color.
- Value (V): Represents the brightness of the color.

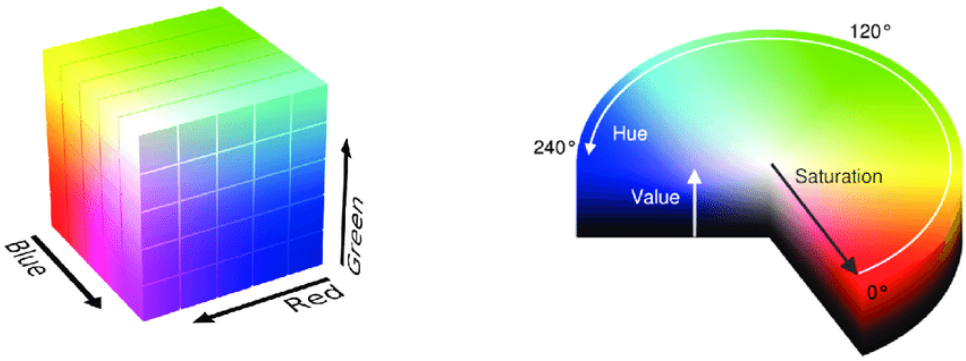


Figure 3.2: (a)RGB color cube (b) HSV color cylinder. Source : [25]

3.2.2 Morphological Operations

Morphological operations are a set of image processing techniques that analyze and modify the structure of objects within an image. These operations use a structuring element (or kernel) to probe and transform an image based on shape characteristics. The two fundamental morphological operations are erosion and dilation.

- Erosion: this operation shrinks objects in a binary image by removing pixels from the boundaries. A pixel remains in the output image only if all pixels under the kernel are set to 1 (white). Erosion is useful for removing small noise and separating connected objects.

- Dilation:this operation expands objects in a binary image by adding pixels to the object boundaries. A pixel in the output image remains white if at least one pixel under the kernel is white. Dilation is often used to fill gaps in an object.

3.2.3 Blob Localization

The technique used is blob detection to identify regions of interest in an image that differ in properties like color or brightness.

Finding a blob centroid is possible with an image moment function, which extracts the coordinates of a selected contour. The coordinates of the centroid (C_x, C_y) are calculated using the following formulas:

$$C_x = \frac{M_{10}}{M_{00}}, \quad C_y = \frac{M_{01}}{M_{00}}$$

Where:

- M_{00} represents the area of the contour (zeroth-order moment),
- M_{10} is the first spatial moment along the x axis,
- M_{01} is the first spatial moment along the y axis.

To select the centroids first contours are selected. Contours are a useful tool for shape analysis and object detection, because allow to select boundaries of an object in a image. There is a function to compute a contour area. Similarly to moments, the area is computed using the Green formula. The formula for calculating the area of a region D using Green's theorem is given by:

$$\text{Area}(D) = \oint_C x \, dy$$

where:

- C is the closed curve bounding the region D.
- x is the x-coordinate of a point on the curve.
- dy is the differential change in the y-coordinate along the curve.

To remove possible noise and obtain a unique centroid an average value it is found between the centroids and also a minimum and maximum area is selected.

3.3 Kalman Filter

In this work, both the Kalman Filter and the Extended Kalman Filter were used to complete the detection and tracking process. The Kalman Filter (Kalman Filter (KF)) is a recursive estimation algorithm used to predict the state of a dynamic system and update it based on new observations [40]. It is particularly effective in linear systems, where both the system model and the measurement noise can be described by linear equations and Gaussian distributions. A key advantage of the KF is its computational efficiency, since it is recursive and does not need to store the entire time series of data, but only the current state and error covariance. This makes it well suited for real-time applications.

The state of the system represents the position and velocity of the target in a three-dimensional space. Symbolically, the state vector is defined as:

$$(x, y, z, \dot{x}, \dot{y}, \dot{z}) \tag{3.1}$$

where:

- represent the target's position in 3D space,
- represent the corresponding velocities in each direction.

The function to predict the states $(x, y, z, \dot{x}, \dot{y}, \dot{z})$ follows the target motion according to the discrete time model given as

$$\hat{\vec{x}}_{t|t-1} = F_t \hat{\vec{x}}_{t-1|t-1} \tag{3.2}$$

where the transition matrix of the state F in discrete time is given by

$$F = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

where Δt is the time step. The covariance estimate at time to predict the target position is given by

$$\hat{\Sigma} t|t-1 \quad = \quad F t \hat{\Sigma} t-1|t-1 F t^{\mathrm{T}} + Q_{t-1} \tag{3.4}$$

where models process covariance and accounts for uncertainty not taken into account in

the system model.

When a new centroid

$$z = \begin{bmatrix} x_{measurement} \\ y_{measurement} \\ z_{measurement} \end{bmatrix} \tag{3.5}$$

is detected by the LiDAR, the measurement update is activated and the first step is corrected introducing an innovation term and a Kalman gain continuously evaluated by the filter at each step, as follows

$$e_t = z_t - h(\hat{\vec{x}}_{t|t-1}) \tag{3.6}$$

$$S_t = H_t \Sigma_{t|t-1} H_t^{\mathrm{T}} + R_t \tag{3.7}$$

$$K_t = \Sigma_{t|t-1} H_t^{\mathrm{T}} S_t^{-1} \tag{3.8}$$

The matrix S estimates the uncertainty in the measurements, the actual covariance Σ_t and the actual state are updated with the innovation term and the Kalman gain

$$\hat{\vec{x}}_{t|t} = \hat{\vec{x}}_{t|t-1} + K_t e_t \tag{3.9}$$

$$\Sigma_{t|t} = (I - K_t H_t)\Sigma_{t|t-1} \tag{3.10}$$

$h(\hat{\vec{x}}_t)$ is linearized around the current state estimate as H_t 3x6 matrix with entries 1

on the main diagonal for the LiDAR update because the measurements z are the three Cartesian coordinates.

$$H_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.11}$$

The Extended Kalman Filter (EKF) is a variant of the KF designed to handle nonlinear systems [40]. While the standard KF works well only with linear systems, the EKF uses a first-order linear approximation (via the Jacobian) of the motion and measurement equations around the current estimate. This allows the EKF to deal with the nonlinearity of the system by continuously adjusting the local linearization as new measurements are taken. For the camera update, the measurements are angles between x and z, and between y and z, denoted as Θ and Φ, respectively.

$$z_{extended} = \begin{bmatrix} \Theta \\ \\ \Phi \end{bmatrix} \tag{3.12}$$

For the Extended Kalman Filter instead the H_t is a 2x6 matrix. For the camera update the measurements are angles between x and y and between x and z. So the map measurement matrix entries are given by the partial derivative of the angles with respect to the states (Jacobian matrix).

$$H_t = \begin{bmatrix} \frac{\partial\theta}{\partial x} & \frac{\partial\theta}{\partial y} & \frac{\partial\theta}{\partial z} & \frac{\partial\theta}{\partial v_x} & \frac{\partial\theta}{\partial v_y} & \frac{\partial\theta}{\partial v_z} \\ \frac{\partial\phi}{\partial x} & \frac{\partial\phi}{\partial y} & \frac{\partial\phi}{\partial z} & \frac{\partial\phi}{\partial v_x} & \frac{\partial\phi}{\partial v_y} & \frac{\partial\phi}{\partial v_z} \end{bmatrix} \tag{3.13}$$

3.4 Camera model

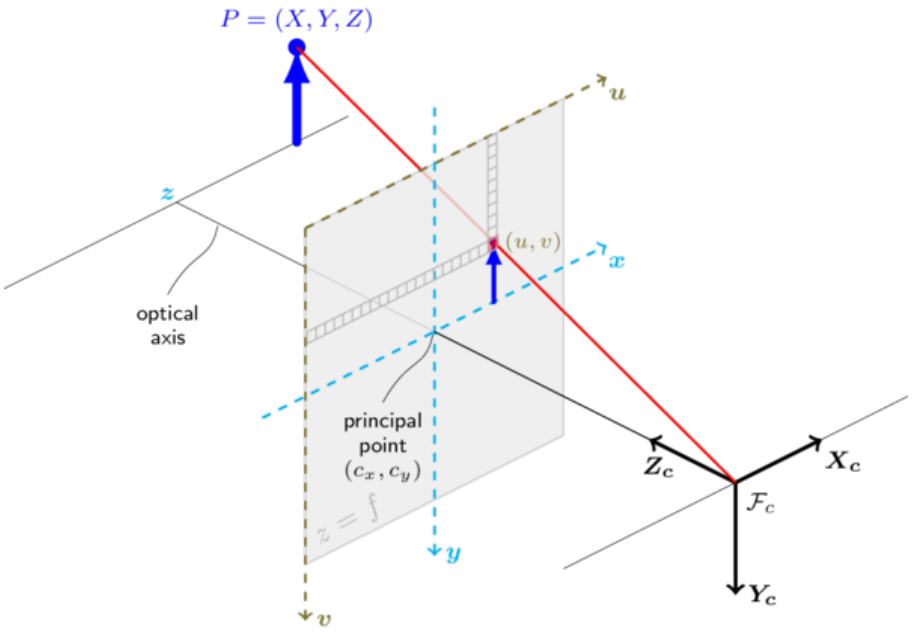


Figure 3.3: The pinhole camera model [42]

The pinhole camera model is used to estimate the angles of the drone with respect to the camera (Fig. 3.3).This model can only be used as a first approximation of the mapping between 3D scene to 2D image. The following equations are satisfied for the

image of a 3D point on the camera in the pinhole camera model [33]:

$$u = f_x * x' + c_x \tag{3.14}$$

$$v = f_y * y' + c_y \tag{3.15}$$

where x' and y' represent respectively the ratios $\frac{x}{z}$ and $\frac{y}{z}$. f_x, f_y, c_x, c_y are known as the intrinsic camera parameters, with the first two being the focal length and the later two being the principal point of the image, and they can be found following a calibration sequence. They are usually presented in the camera matrix A as

$$A = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.16}$$

Using the above identities the angle of the drone with respect to the camera can be found using

$$\theta = \tan^{-1}\left(\frac{x}{z}\right) \tag{3.17}$$

$$\Phi = \tan^{-1}\left(\frac{y}{z}\right) \tag{3.18}$$

Chapter 4

Proposed Method

In this chapter, the complete processing pipeline will be presented, detailing the algorithms and procedures used for object detection. The approach integrates LiDAR and camera measurements, leveraging their complementary strengths to improve accuracy. The detection process, sensor fusion techniques, and estimation methods will be described, providing a comprehensive overview of the system's workflow.

4.1 Drone detection pipeline

The flow-chart involves six different blocks that interact each other (Fig. 4.1). The principle blocks are responsible for detecting the drone in images obtained from the camera, for detecting the drones using the point cloud data obtained from a 3D LiDAR, and fusing them in an (EKF). The secondary blocks are responsible for controlling the pan-tilt units holding the LiDAR and the camera.

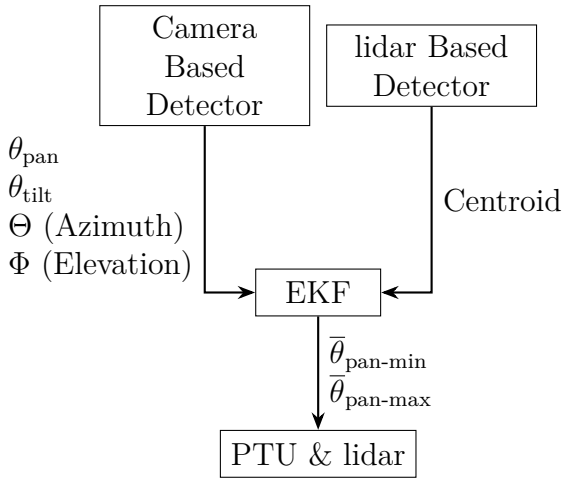


Figure 4.1: Workflow of the detection and tracking process

In the camera based detector the input is the rectified image, in the LiDAR based detector the input consists of point clouds. At the beginning both save the background to execute background subtraction and find the centroid coordinates of the object. The extended Kalman filter block receives and outputs all the necessary data to track the drone's path in 3D space.

4.2 LiDAR Based Drone Detection

On start, the LiDAR-based detection block performs a continuous sweep of the environment for 15 s to construct a point cloud corresponding to the background. A boolean variable was created, and a new empty cloud was initialized to store the background. This proved to be the most efficient solution found to save the background and begin the detection process. By accurately capturing the background, the system can better identify and track the target in the environment, ensuring that moving objects are distinguished

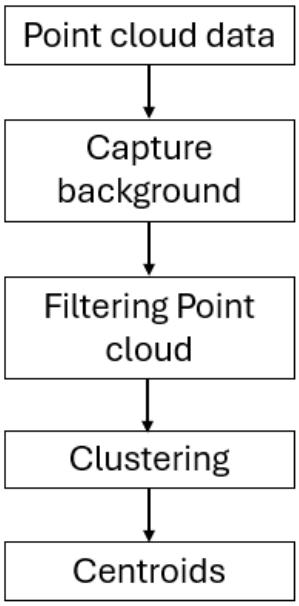


Figure 4.2: LiDAR data flow diagram

from static ones.

After completing the overall scanning of the environment the Pan-Tilt Unit (PTU) is rotated to an initial angle and left waiting for detections to be reported by the camera node. After receiving an alarm from the camera, the PTU is rotated to align the LiDAR with the reported angle of the target drone, and processing of point cloud data begins. The algorithm's logic is detailed in section 4.6. First an x,y,z filter which removes points by a simple thresholding is applied, filtering the point cloud roughly. The limits are measured in meters and are selected based on the surroundings. This procedure is applied to all the three axis. In the *compare-background* function, after applying the PassThrough filter, background subtraction is carried out, removing all points corresponding to pre-existing objects. Filtering the point cloud is necessary to eliminate pre-existing objects

that may have similar angular distances to the drone with respect to the LiDAR.

The background is constructed as a voxel grid of the collected point cloud data. The voxel grid both helps smooth the point cloud and improve the search speed when comparing the new point clouds to the background. The voxel grid is represented as a k-d tree (k-dimensional tree) data structure to organize points in k-dimensional space. This structure is ideal for range and nearest neighbor searches,focusing on three-dimensional point clouds. This method allows to find correspondences between groups of points or feature descriptors or to define the local neighborhood around a point or points, just setting the correct threshold.

The clustering of the point cloud after filtering is done using Euclidean distance, clustering the points using several criteria: maximum distance between two points in a cluster, minimum and maximum number of points in a cluster.

Then the points of the cloud around the angles reported by the camera are found, they are clustered and their centroids are reported. After the clusterization the coordinates of the centroids of the object are the outputs of the block and are received by the block of the Extended Kalman filter.

4.3 Camera Based Drone Detection

Camera based drone detection aims to detect the drone in a given image, as well as its 3D orientation with respect to the camera. The orientation contains the azimuth and elevation angles of the detected drone.

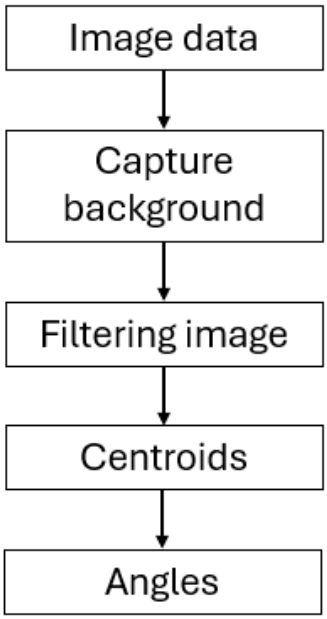


Figure 4.3: Camera data flow diagram

For this purpose, background subtraction is used and then after filtering the left cluster's pixel coordinates are used to find the orientation of the drone with respect to the camera. The image frames are received in RGB format and then they are transformed to HSV format. In this color space, which roughly separates color from light, thresholding is relatively simpler, allowing easy creation of binary masks.

Several tests were conducted before selecting the appropriate threshold for the filter, as much influenced by external lighting conditions. Multiple videos and images in different backgrounds were analyzed to fine-tune the parameters and obtained the best values. By analyzing many frames and setting 0.2 (scale from 0 to 1) as value for the third channel of the image obtained by subtraction(Fig.4.4(b)), the drone was perfectly detected. The

Figure 4.4: (a) Third channel (b) image subtraction

aim is minimize as much as possible the noise to identify only the target in the mask. The image obtained with the subtraction removes almost all noise in the third channel(value), leaving enough visible the drone(Fig.4.4(a)). The selection of a threshold value within the range of 0-1 (or 0-255 in MATLAB) for the value channel results in the generation of a binary mask. Subsequently the erosion operation removes sparse white pixels, thereby the calculation of the centroids is more accurate.

In this work, the camera was placed on top of a pan tilt unit (PTU) to allow easy positioning of the camera. Hence, the angles obtained above are also transformed to the base of the PTU to have consistent angles with respect to the LiDAR's PTU, as to be explained in section 4.5.

4.4 Camera and LiDAR Fusion

The fusion block receives the centroid of the detected object and its angles from the camera, and then sends these angles to control the PTU of the LiDAR. For fusion a

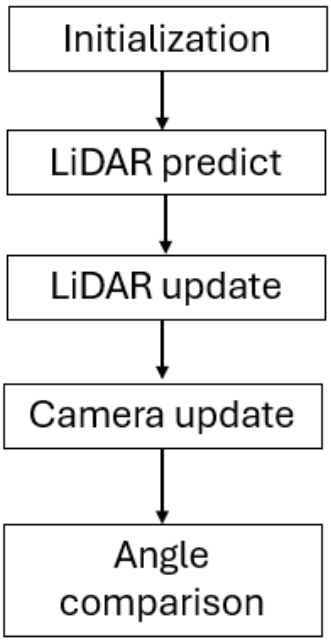


Figure 4.5: Fusion flow diagram

hybrid EKF was used.The hybrid EKF is an approach that combines data from different sensors (such as LIDAR and camera) to more accurately estimate the state of a dynamic system [40]. A constant speed model was used for the drone. A linear system was assumed for the LiDAR update, in which the measurements are the 3D Cartesian coordinates of the centroid. A non-linear system was assumed for the camera update, in which the measurements are the azimuth and elevation angles of the drone with respect to the world frame.

The fusion block also monitors the workflow of drone detection. It uses different states for the camera and the LiDAR. NO DETECTION, INITIALIZATION, LiDAR TRACKING and CAMERA TRACKING. The purpose of the work is to detect the

object with the camera and send the position of the object to the LiDAR's PTU, so the LiDAR can start to detect and to track the drone. Afterwards the LiDAR is expected to let the camera keep tracking until the target changes its maneuver, or the EKF has a large covariance in the state estimate. These states change depending on the inputs of the block. Firstly the state is initialized to NO DETECTION, that means the camera and LiDAR are saving the background and the LiDAR has not detect anything.

Then when an object is seen by the camera, it first triggers the LiDAR, getting the first 3D pose measurement and then initialize a new track in the EKF. At this point the current state becomes LiDAR TRACKING and the tracking with LiDAR starts. The LiDAR reports detections from the LiDAR for a few seconds (3 s in current tests), and then stops tracking moving to the default position. At the same time, the camera continues to track the drone in state CAMERA TRACKING.

When the LiDAR is in EVOLVING STATE, the EKF update of the camera begins receiving angle measurements in real time. The error between the azimuth angle predicted by the EKF in camera coordinates and the new azimuth angles of the target in the camera is continuously monitored, and when it exceeds 10 degrees, it is assumed that the drone has changed its track considerably, and hence the PTU of the LiDAR is commanded to the new angles, and the Kalman filter starts all over again. This process is also done when the entries of the covariance matrix of the state, Σ, exceed a preset threshold. Then the state is changed to CAMERA TRACKING, and tracking of the target with the camera continues, updating the EKF with the angle measurements from the camera. The error between the azimuth angle predicted by the EKF in camera coordinates and the new

azimuth angles of the target in the camera is continuously monitored.

4.5 LiDAR Tracking Angle Calculation

Since the camera cannot measure depth, only the direction along which the object was seen is monitored as the corresponding azimuth and elevation angles. However, the LiDAR sensor and the camera are not concentric. Hence, the LiDAR cannot directly align with the azimuth and elevation angles of the drone to measure the distance. This presents a challenge in determining the correct target angle to move the PTU of the LiDAR. To overcome this issue, two points were selected in the camera frame along the direction of the target as reported in the camera frame. Then, the angles of these

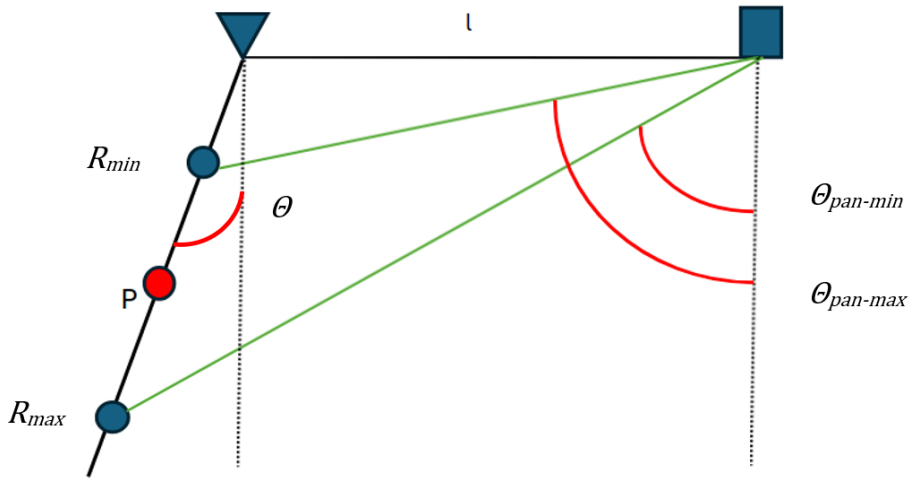


Figure 4.6: Diagram of the detection system. The triangle represents the camera, and the rectangle represents the LiDAR. The drone is observed along the black line as the red circle P.

two points with respect to the PTU of the LiDAR are calculated analytically, using the known geometry of the PTUs (Fig. 4.6). The two points are selected as the minimum

range (R_{min}) and maximum range (R_{max}) the drone is expected to be seen. In the configuration shown in Fig. 4.6, the two points R_{min} and R_{max} lying along the direction of the drone, shown by the point P, allow us to calculate the range of pan angles necessary to move the LiDAR-PTU to detect the drone. These angles are given by

$$\overline{\theta}_{\text{pan-min}} = -\arctan\left(\frac{l - R_{max}\sin(\theta)}{R_{max}\cos(\theta)}\right) \tag{4.1}$$

$$\overline{\theta}_{\text{pan-max}} = -\arctan\left(\frac{l - R_{min}\sin(\theta)}{R_{min}\cos(\theta)}\right) \tag{4.2}$$

where l is the distance between the pan frame of the PTU of the LiDAR and the world frame (camera PTU). The arctan is negative to comply with the command input required to move the PTU correctly.

4.6 Implementation

In this chapter, some implementation details of this thesis are presented. The focus is on the development and implementation of algorithms for detection and tracking of an UAV. These algorithms have been implemented within the Robot Operating System (ROS) framework, utilizing libraries such as the Point Cloud Library (PCL) for 3D data processing and OpenCV for computer vision tasks, all implemented in C++. This chapter will accurately describe the architecture of the system, which consists of multiple ROS nodes, which communicate through topics and services.

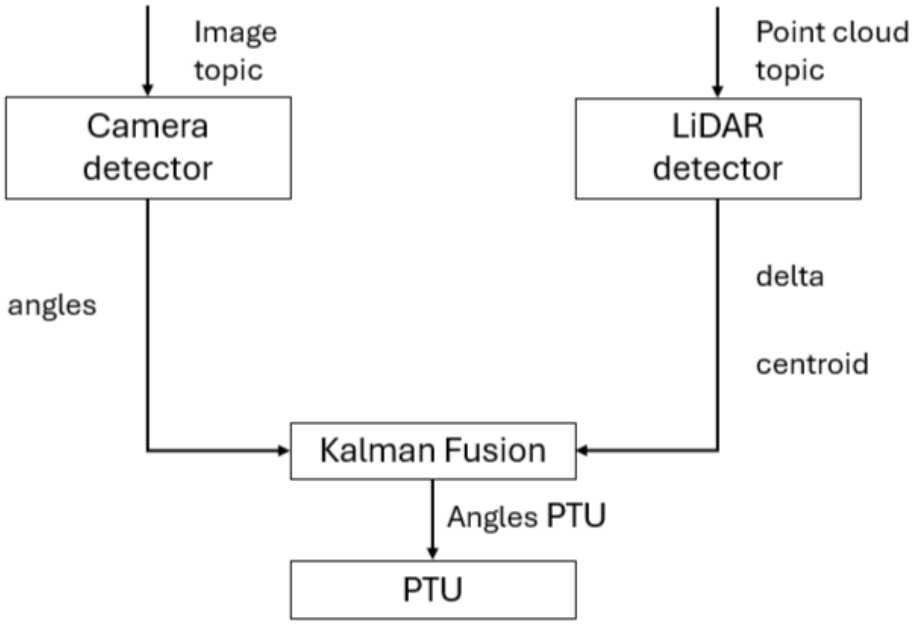


Figure 4.7: Node and topic diagram

4.6.1 Pan-Tilt Units node

The initial focus of the project is on the movement of the two PTUs (Pan-Tilt Units). It was necessary to implement two separate nodes to control the pan and tilt motions and enable the smooth and precise swinging movement in both directions.

The communication of both PTUs is handled through a serial port, while the movement parameters are set according to the user manual of the individual turret. The serial library is a software library used to manage serial communication, for interfacing between computers and USB connection. The serial library allows to establish connections between devices via serial ports, allowing to send and receive data in real time.

```
  void sendCommand(serial::Serial& serial, const std::string&
      command) {
    serial.write(command + "\r\n");
}
```

Listing 4.1: Function to communicate with serial port

The LiDAR PTU node receives a topic containing the values for the pan minimum and maximum angles (as calculated in section 4.5). Two callback functions are responsible for processing this message, setting these angular limits, one for each angle. In the following code a value is received and saved in the variable.

```
void ptuCallback_min(const std_msgs::Float64::ConstPtr& msg) {
   pan_min=msg->data;
}
```

Listing 4.2: Callback for PTU

The movement logic is managed by two statements that check that the turret has reached the two limit corners before repeating the movement. The chosen step determines the speed of movement. As can be seen in the code below, setting a minimum and a maximum value the turret knows which step is the following.

```
void movement_pan_tilt(int& start_value, int& min_value, int&
   max_value, int& step, bool& move_forward) {
        if (move_forward) {
          start_value += step;
        if (start_value >  max_value) {
            move_forward = false;
            start_value= max_value;
            }
    } else {
        start_value -= step;
        if (start_value < min_value) {
            move_forward = true;
            start_value = min_value;
        } } }
```

Listing 4.3: Function to move the PTU

All parameters are entered in degrees, after a conversion from pan and tilt units to the equivalent angles in degrees. This allows a more intuitive and precise control of the PTU movement, based on standard angular values. The camera PTU node is simpler because it does not receive a topic, but simply configures an angle to move the camera upwards along the vertical axis, setting a direct and predefined movement.

4.6.2 LiDAR detection node

Initially, a study was conducted on the operation of the LiDAR, and the necessary libraries were analyzed to achieve the objectives. Some tests were carried out in the testing arena to develop the detection system for a moving target. Firstly the target was a person then tests were performed with a smaller object. During the first field test with a drone, it was possible to select the appropriate parameters and understand what potential noises could affect the detection system. Later, the LiDAR was placed on top of a PTU-46-17.5 to increase the field of view.

The detection node of the LiDAR is composed of differents functions to manage the data incoming and process precisely to detect the target. A callback function gets called every time the message is received on the topic of the point cloud and it operates each

```
pcl::KdTreeFLANN<pcl::PointXYZI> kdtree;
kdtree.setInputCloud(background_downsampled);
float threshold = 0.9f;
```

Listing 4.4: KdTree threshold

After saving the background and filtering through a PassThrough filter, both background and the new point cloud in a function to process both point clouds a KdTree structure is created.

The reference cloud, `background_downsampled`, is the input on which the KdTree will base its spacial search. A distance threshold is then defined, set to 0.9 units, which represents the radius within which points close to each point of the cloud of `cloud_downsampled`

points will be searched. During the iteration on each point of the reduced cloud, two vectors are declared: indices, to store the indexes of the nearest points found, and distances, for their distances. The radiusSearch() function of KdTree is used to search for points in the background cloud that are within 0.9 radius. If no neighbor is found (i.e., the function returns 0), the current point is considered isolated from the background and is then added to the filtered points cloud, `cloud_filtered`. This function allows to select the points that don't belong to the background.

```
for (const auto& point : cloud_downsampled->points) {
    std::vector<int> indices;
    std::vector<float> distances;
    if (kdtree.radiusSearch(point, threshold, indices,
       distances) == 0) {
        cloud_filtered->points.push_back(point);
    }
```

Listing 4.5: KdTree extraction

Using the Point Cloud Library (PCL), the algorithm to extract clusters from a point cloud has been implemented.

Initially, a KD-tree structure is created, and later a vector is declared to store the indices of the points that belong to the found clusters. The object EuclideanClusterExtraction allows to identify the clusters based on euclidean distance between the points.

```
pcl::search::KdTree<pcl::PointXYZI>::Ptr tree(new pcl::search
   ::KdTree<pcl::PointXYZI>);
tree->setInputCloud(cloud_filtered);
std::vector<pcl::PointIndices> cluster_indices;
pcl::EuclideanClusterExtraction<pcl::PointXYZI> ec;
ec.setClusterTolerance(1);
ec.setMinClusterSize(2);
ec.setMaxClusterSize(50);
ec.setSearchMethod(tree);
ec.setInputCloud(cloud_filtered);
ec.extract(cluster_indices);
```

Listing 4.6: Clusterization

The parameters set are the tolerance(in meters) which represents the maximum distance between points in the same cluster(in meters),the minimum and maximum cluster size represent respectively the minimum and maximum number of points required to form a cluster.

Then, an iteration through the cluster indices is carried out to fill a 3D point cloud and construct each clusters. In the code below the function to fill the point cloud is presented.

```
for (size_t i = 0; i < cluster_indices.size(); ++i) {
    pcl::PointCloud<pcl::PointXYZI>::Ptr cloud_cluster(new pcl::
       PointCloud<pcl::PointXYZI>);
    for (const auto& idx : cluster_indices[i].indices)
    cloud_cluster->push_back((*cloud_filtered)[idx]);
    cloud_cluster->width = cloud_cluster->size();
    cloud_cluster->is_dense = true;
```

Listing 4.7: Function to fill the point cloud

Later, it is possible to find the centroid of each cluster point cloud using the compute3DCentroid function. A structure is created to store the timestamp, size, and coordinates of the centroid. Subsequently, a pose message is declared to extract the centroid coordinates.

```
    pcl::compute3DCentroid(*cloud_cluster, centroid);
    cluster_info.centroid = centroid.head<3>();
    cluster_info.timestamp = msg->header.stamp;
    cluster_info.num_points = cloud_cluster->size();
    geometry_msgs::Pose pose;
    pose.position.x = centroid[0];
    pose.position.y = centroid[1];
    pose.position.z = centroid[2];
    pose.orientation.w = 1.0;
```

Listing 4.8: Function to compute Centroids

4.6.3 Camera detection node

The real-time video of the camera is processed with OpenCV library. Firstly the images are received via a subscriber, that allows a node to receive messages from the topic of the rectified image. A callback function gets called every time the message is received on that topic and it operates each time a new frame arrives. Another function is placed in callback to process each image, subtract background, select a threshold and detect the drone.

Each image that arrives is transformed into the HSV format. The background is then saved for 5 seconds. Each new frame is compared with the saved background and can be acquired the difference image:

```
cv::Mat diff_image;
cv::subtract(hsv_background,hsv_image, diff_image);
filter_no_mask(diff_image,original,pan,tilt,threshold);
```

Listing 4.9: Image subtraction operation

The function *subtract* function in OpenCV is used to calculate the subtraction between two images or matrices, it subtracts the values of one matrix from the values of another, element by element. Once this image is obtained, a new function takes care of target detection.

```
cv::Mat filtered_image;
cv::Mat element = cv::getStructuringElement(cv::MORPH_RECT, cv::
    Size(5, 5));
cv::inRange(diff_image, cv::Scalar(0,0*255, threshold*255), cv::
    Scalar(180,255,255), filtered_image);
cv::erode(filtered_image, filtered_image, element);
computeCentroids(filtered_image,centroids,min_area,max_y,max_area
    );
cv::Point mean_centroid;
cv::Mat centroidsMat(centroids);
cv::Scalar meanCentroid = cv::mean(centroidsMat);
    mean_centroid=cv::Point(static_cast<int>(meanCentroid[0]),
    static_cast<int>(meanCentroid[1]));
cv::circle(original, mean_centroid, 10, cv::Scalar(0, 0, 255),
    -1);
```

Listing 4.10: Image thresholding operation and centroids

This section of code demonstrates how to use OpenCV for target detection after performing image subtraction. Firstly a new image and a rectangular element 5x5 pixels are declared. The aim is to obtain a binary mask(Fig. 5.2b) so as to highlight the target in the image. The `inRange` function in OpenCV is used to isolate pixels of a certain color range or intensity values and in this case the operation of thresholding is applied on the third channel in HSV format, leaving the range for Hue and Saturation unchanged, to cover the entire possible range. The `erode` function applies a morphological erosion operation,

which removes white areas in a binary image based on a structuring element. It is useful for removing pixels at the edges, noise (isolated white dots) and strengthening the forms in the binary image. The target centroids have been identified by `computeCentroids` function that receives binary image as input. In the following code lines, OpenCV is used to identify the contours in an image and calculate its areas.

```
void computeCentroids
{
    std::vector<std::vector<cv::Point>> contours;
    cv::findContours(filtered_image, contours, cv::RETR_EXTERNAL,
        cv::CHAIN_APPROX_SIMPLE);
    for (const auto& contour : contours) {
        double area = cv::contourArea(contour);

        if (area > min_area && area<max_area) {
            cv::Moments m = cv::moments(contour,true);
            cv::Point center(m.m10 / m.m00, m.m01 / m.m00);

           if(center.y<max_y && center.x<max_x){
             centroids.push_back(center);
             }
```

Listing 4.11: Detailed function to compute image Centroids

First, a *contours* vector is declared, which is a vector of point vectors. Each point represents a 2D point with coordinates u and v, while each vector represents a single contour. Next, the function *findContours* is called to detect contours in the image *filtered_image* (binary mask), where objects are represented in white on a black background. The vector *contours* is the output parameter that will contain the detected contours. With the *RETR_EXTERNAL* parameter, only the outer contours are retrieved, ignoring the inner ones. This means that if there is a shape contained within another, only the outer contour will be detected. After the contours are detected, the code scans each of them with a for loop. At each iteration, the area of each contour is calculated using the *contourArea* function, which returns the number of pixels within the contour. This value, stored in the area variable, represents the size of each shape in the binary image. The area calculation for each contour is useful for analyzing based on the dimensions of the detected shapes.

The area, the threshold and the coordinates of the camera are selected in real-time with getParams, to achieve the highest possible detection accuracy removing possible noises from the images.

Following the camera pinhole model in equations 3.15 and declaring the parameters of the camera, the normalized coordinate are calculated (x' and y'), in order to find the Azimuth and Elevation angles. After computing the angles, a marker object is used to display a 3D arrow pointing in the direction of the target, with the end point oriented using the sine of the two calculated angles.

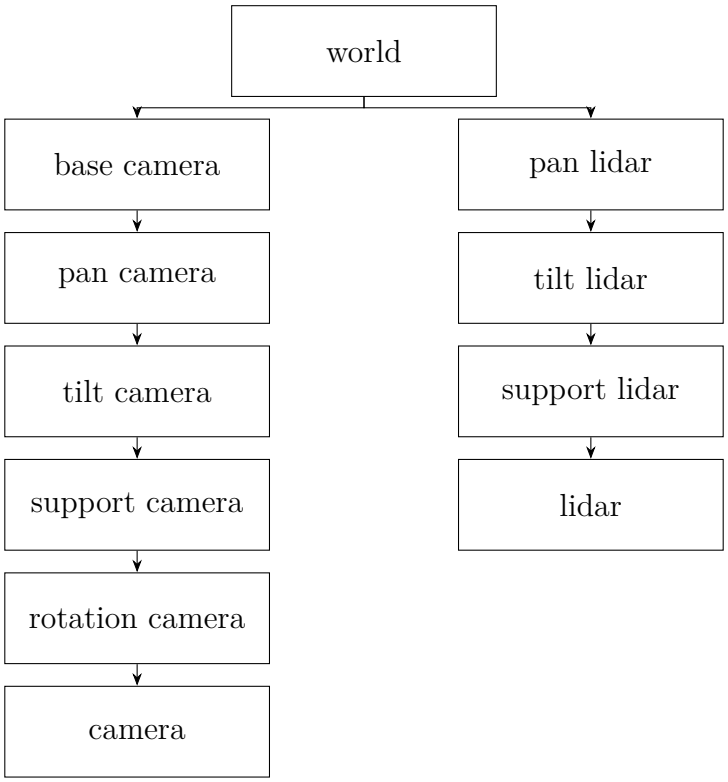


Figure 4.8: TF diagram

4.6.4 Transforms and Frames in ROS

Another crucial aspect of this study involves the setting of the appropriate frame for all utilized sensors and robots. The sensors are placed on two PTUs, and to ensure coherent output it is necessary to align the references frames for analyzing data from both the camera and the LiDAR. First, by taking into account the calculation of pan angle to move the LiDAR, the world frame is aligned with the base of the camera's PTU. Using a metal bar (as shown in Fig. 5.1), the two turrets were fixed, allowing to compute the distance between the frames of the basis.

The definition of the frames was made possible thanks to *tf broadcast*, which allows for the establishment and management of relationships between different coordinate frames in real time. This functionality enables to accurately track the position and orientation of each frame relative to others, facilitating integration and interpretation of data from the sensors (frames in RViz Fig.4.9).

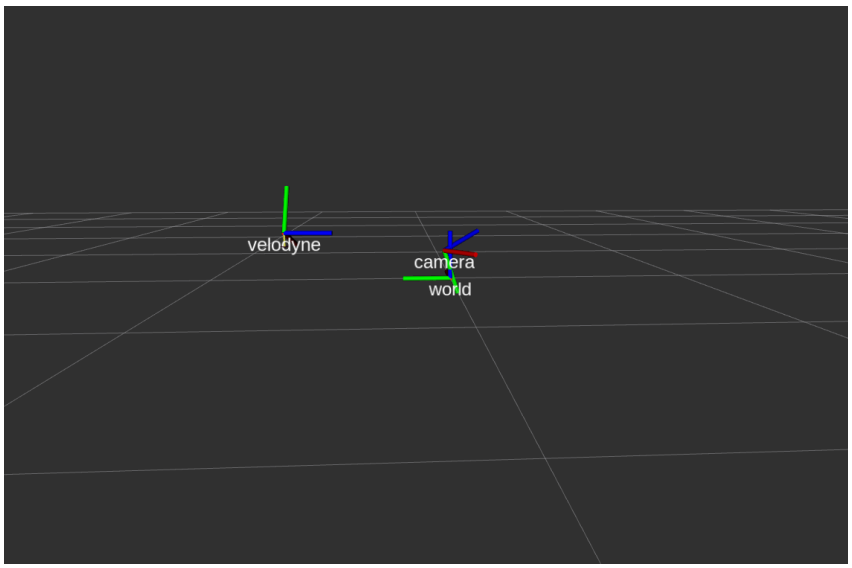


Figure 4.9: Frames in RViz

The diagram in Fig. 4.8 expresses the relationships that occur between the frames, illustrating the hierarchical structure of the frames of the LiDAR and the camera in relation to the central frame *world*.

The frames for the LiDAR and its turret include the pan and tilt frames, plus the support frame and the LiDAR frame. Referring to its manual, the distances between the pan frame and LiDAR frame were calculated. As shown in the following section of the code, an origin is first established, followed by the creation of a quaternion. Finally, the

transform is transmitted. Everything has been set up to conform to the LiDAR frame specifications as outlined in the manual.

```
tf::Transform transform1;
    transform1.setOrigin(tf::Vector3(0,0.69,0.045));
    tf::Quaternion q1;
    q1.setRPY(0,0,yaw);
    transform1.setRotation(q1);
    br1.sendTransform(tf::StampedTransform(transform1, ros::Time
       ::now(), "world", "pan_frame"));
```

Listing 4.12: Frames setting

To obtain the correct positioning of the LiDAR frame, after setting the pan and tilt frame, a rotation of 90° degrees is applied around the x axis, which is subsequently followed by a rotation of -19 degrees around the newly defined z-axis, based on the configuration of the support of the LiDAR (Fig.5.1).

The camera configuration is easier because the PTU already has its default frames in its configuration file (base,pan,tilt,support).

```
static tf2_ros::StaticTransformBroadcaster static_broadcaster
geometry_msgs::TransformStamped transformStamped1;
transformStamped1.header.stamp = ros::Time::now();
transformStamped1.header.frame_id = "pxxls/surface_link";
transformStamped1.child_frame_id = "first_rotation";
transformStamped1.transform.translation.x = -0.02;
transformStamped1.transform.translation.y = 0;
transformStamped1.transform.translation.z = 0;
tf2::Quaternion q1;
q1.setRPY(0, M_PI/2, 0);
transformStamped1.transform.rotation.x = q1.x();
transformStamped1.transform.rotation.y = q1.y();
transformStamped1.transform.rotation.z = q1.z();
transformStamped1.transform.rotation.w = q1.w();
static_broadcaster.sendTransform(transformStamped1);
```

Listing 4.13: Static transformer

This allows to use a static transformer broadcast to manage the transformations for the PTU and set also the parameter for the *support* and *rotation camera* frames as outlined in the lines of the code above.

Chapter 5

Experimental Work

5.1 Experimental setup

The experimental setup consists of a tracking system and a drone. The tracking system contained two Pan-Tilt units that were fixed at a distance of 0.69 meters on an aluminum bar as shown in Fig. 5.1(a). A Velodyne VLP16 and a 5MP camera were mounted on the pan-tilt units as shown in Fig. 5.1(b) and Fig. 5.1(c). The Velodyne VLP16 has 16 vertical beams, separated by 2°, with a horizontal resolution varying from 0.1° to 0.4°, and a corresponding frame rate varying between 5 Hz and 20 Hz, but set at 10 Hz. The pan-tilt unit where the LiDAR is mounted is a PTU-46-17.5. The PTU is able to control the position and speed of the end-effector (payload) in the vertical axis only, because the tilt axis was mechanically fixed. The camera had 5MP resolution and a 5 mm lens, giving a Field of View (FoV) reaching 75°.

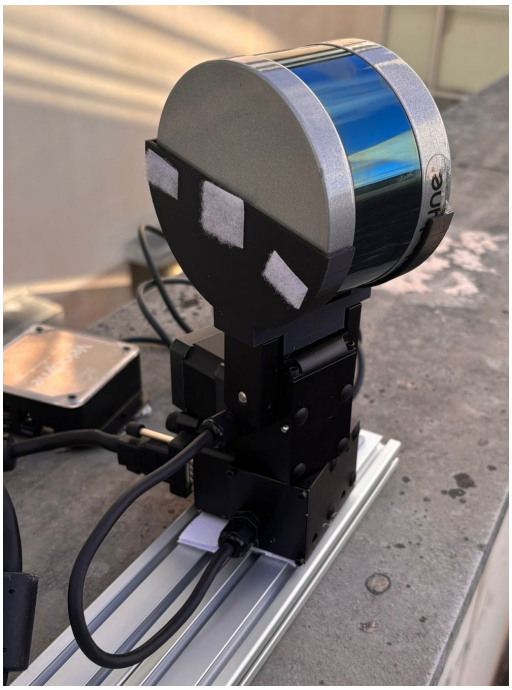


Figure 5.1: (a) Overall setup (b) Velodyne VLP16 on a PTU 46-17.5 (c) Camera on a PhantomX XL430 (d) Drone

It uses a USB interface to communicate and the input image is initially calibrated and continuously rectified. In this work a Sky Hero Spyder X4 drone was used as the object to detect. This drone has a carbon fiber frame of size of 0.85 m (Fig 5.1(d)). It is powered by 400 rpm/V motors and a 6S 16000 mAh LiPo battery. The code for the algorithms was programmed as C++ nodes for Robot Operating System (ROS).

5.2 Tests and Results

5.2.1 Field Tests

Figure 5.2: (a) Detected drone, marked with a red dot (b) The corresponding binary mask

The proposed approach was validated with multiple tests. Due to the large FoV of the camera, it was possible to cover a large area while the camera was fixed. This allowed for saving a single background image in the beginning of each test, which lasted several minutes. Background subtraction followed by thresholding allowed the detection of the drone in many of the tests (Fig. 5.2).

However, the method also started to report false detections when moving clouds were present in the background. Additionally, detection became unreliable under low light conditions when the drone had trees as the background.

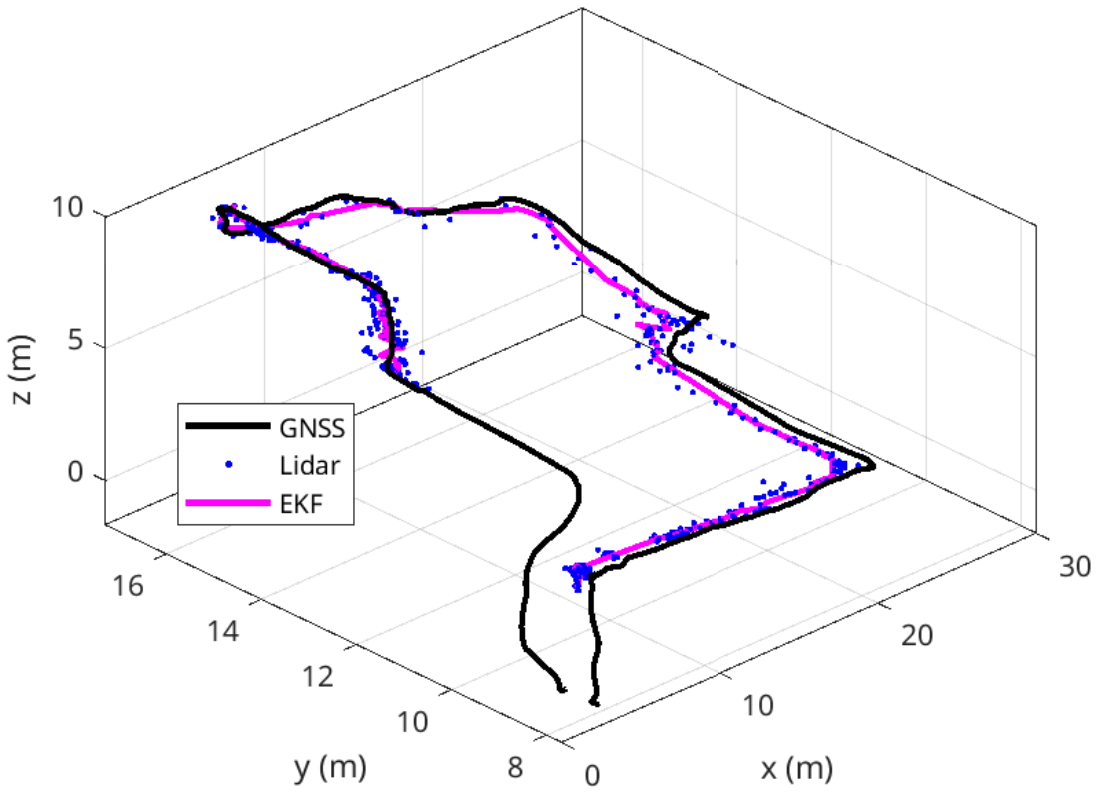


Figure 5.3: 3D view of the path of the drone

Fig. 5.3 shows the drone path (black) and the corresponding detections (blue) and EKF estimates (magenta) in one of the tests. The updated EKF coordinates correctly follow the path of the drone. At the beginning the drone goes up; the camera begins detecting when it reaches a height of 2 meters. Immediately after, the LiDAR starts to

look in that direction and starts an EKF to track the state of the drone.

The results in Table 5.1 show the measured euclidean distance, and the corresponding number of points, providing an analysis of the precision of the LiDAR measurements. It can be observed that as the distance increases, the number of points in the clusters tends to decrease. This phenomenon indicates a reduction in measurement density and suggests that increasing the distance from the origin makes it possible to detect the drone with only two points. In the best test conducted only 3 false positives were found out of 160 detected centroids, obtaining an accuracy rate of approximately 98.13 %. These results are significant to understand the accuracy and reliability of LiDAR measurements.

Table 5.1: Euclidean distance and cluster size

Euclidean distance	**Cluster size**
9.3565	7
9.48618	3
9.60682	2
11.3246	6
11.0818	6
11.709	2
12.0096	3
12.4192	2

The plot in Figure 5.4 represents the drone trajectory in camera coordinates (u, v with the origin on the top left), which shows highly accurate detection without the presence of outliers (after removing the noise caused by the tree on the right side of the image). The points that may appear to be outliers correspond to the drone's takeoff and landing.

The transitions of the states for the camera based detector and LiDAR based detector as well as the estimates of the EKF are shown in Fig. 5.6 for a subset of a test. The

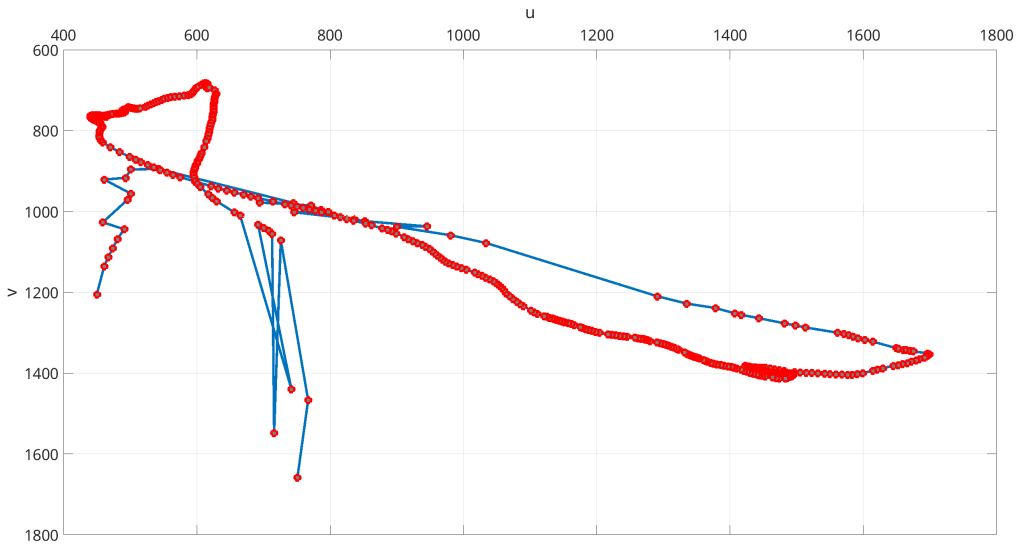
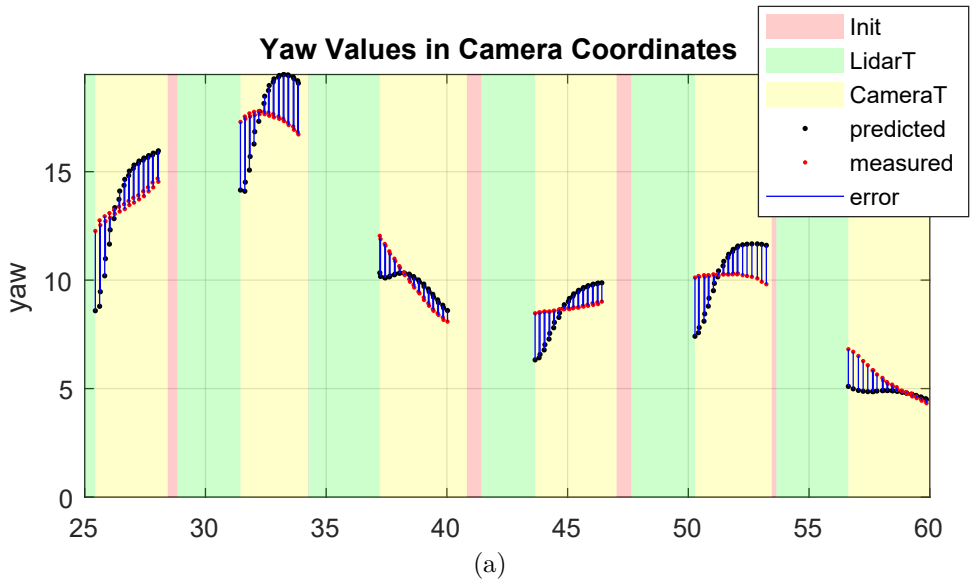


Figure 5.4: Path of the drone in camera coordinates

azimuth angles of the drone with respect to the camera in the same time interval are shown in Fig. 5.5. The *Init* (INITIALIZATION) phase, which is the time between seeing in the camera and starting detecting with the LiDAR can be seen to be brief, but not always instantaneous. This is a side effect of the sparsity of the beams as well as the time it takes for the LiDAR to align with the angle. LiDAR tracking is 3 s in this test, and the EKF can be seen to closely match the clusters (green zones). During camera tracking (yellow zone), the tracking performance can be seen to vary both in time and across the different axes. For example between 37 s and 41 s, camera based EKF follows the drone in the x-axis and y-axis closely, with errors less than 1 m. However, in the same time interval, the error along the z-axis quickly reaches 1.3 m. A consistent error can be observed along the z-axis when EKF relies only on camera. In Fig. 5.5, it can be seen

that there is almost always an error between the azimuth angle of the drone predicted by the EKF during camera only tracking, and the azimuth angle measurement of the camera. This error is mainly caused by the motion of the drone.



(a)

Figure 5.5: Angle of the target as predicted by the EKF and as seen by the camera, with respect to the camera frame.

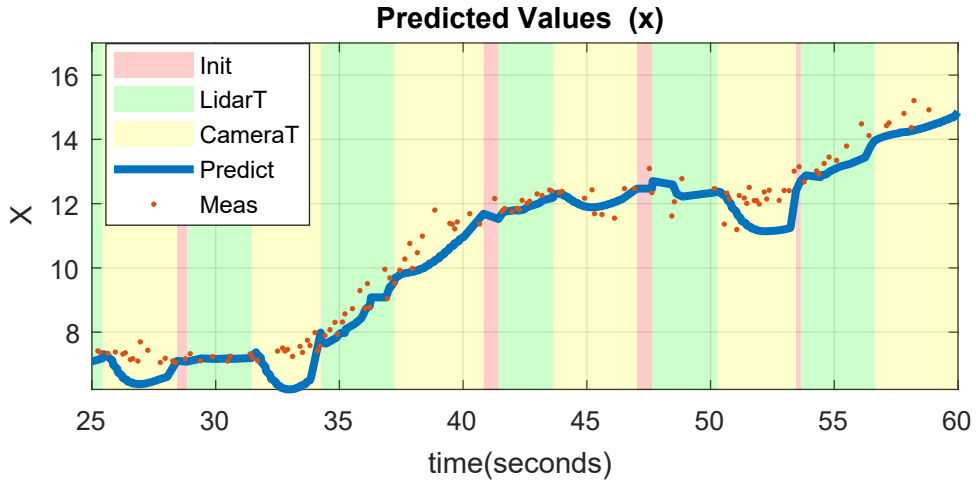
The change in the state of the LiDAR can be better understood by analyzing it in parallel with how one of the coordinates of the estimates of the drone changes over time. Fig. 5.6 presents the EKF predicted values and the corresponding states of the LiDAR. The motion estimation starts after the first detection of the LiDAR. However, motion estimation is executed only when new data is available.

The prediction starts after the first detection of the LiDAR. It is not executed constantly due to the fact that depends on the timestamp of each message entering in the Kalman block. The short duration of the evolving state can be analyzed focusing on the

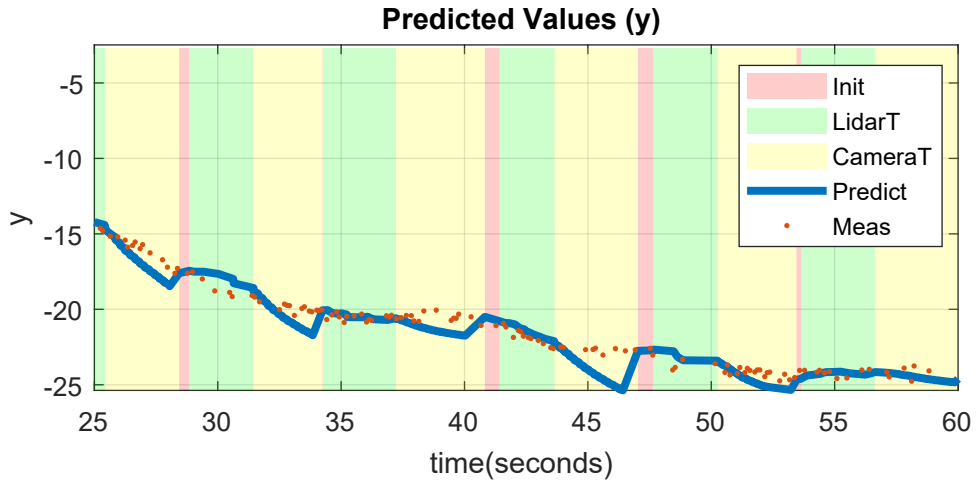
angle comparison between the yaw angle of the camera when it sees the drone and the angle predicted by the filter.

Fig. 5.5 focuses on few instants when the yaw angle of the drone (and hence the predicted path of the drone) starts to diverge from the angles predicted by the EKF. By considering a time interval of 2–3 seconds, the evolution of the error over time can be analyzed. For instance, between $t = 30\,\mathrm{s}$ and $t = 35\,\mathrm{s}$, the estimated azimuth and the measured azimuth move in opposite directions, eventually reaching an error. This marks the end of the *EVOLVING* state, as the predictor fails to estimate the future position, leading to a significant increase in error. Afterwards, the detection restarts for both sensors.

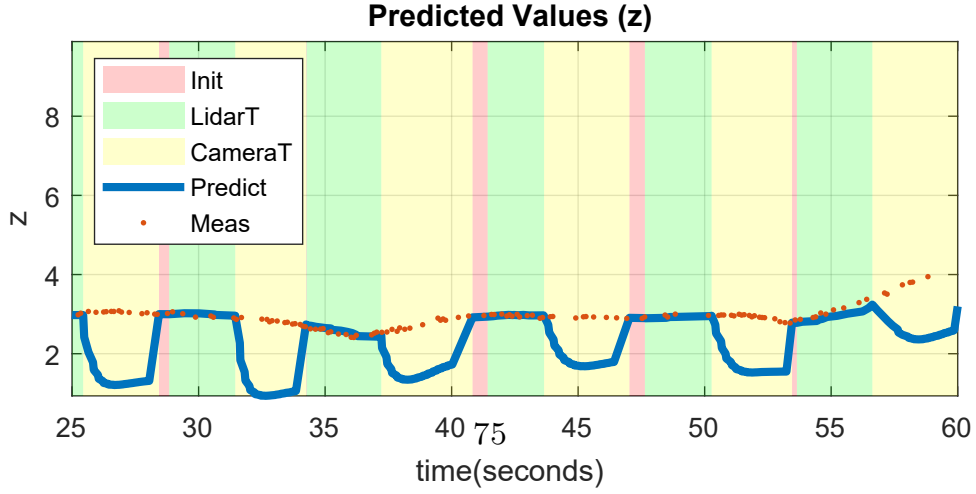
When the drone changes the trajectory, the predictor is unable to estimate the future position and the error significantly increases. The assumption that the axis of rotation of the camera is the same as the PTU influences this error. During the time between each evolving state, the camera and the LiDAR are detecting and tracking the target. The tracking time for the LiDAR is set to 3 s to there is a short delay(1-2 s) between two consecutive evolving states due to the seconds necessary to detect and enter in the following state.



(a)



(b)



(c)

Figure 5.6: Output of the EKF for x,y,z and the state of the system at the time

Chapter 6

Conclusion

In this work a system for object detection and tracking using data from LiDAR and camera sensors, fusing both via an Extended Kalman filter is presented. The fusion of the two sensors has enabled a more robust detection system for drones. Cameras are faster to detect changes in the sky compared to a 3D LiDAR, which has sparse beams. However, the camera lacks the capability to accurately localize a drone in the sky. In this work we fused the advantages of these two sensors. The experimental results have demonstrated that background subtraction for the camera works acceptably well in static conditions. Additionally, we showed a method to find the sweeping range for the LiDAR through the angles found by the camera and the geometric configuration. The method was effective in localizing targets with the LiDAR, helping achieve good localization. One objective can be to overcome the issues related to the dynamic background subtraction of the camera. Further research could investigate alternative solutions for cases where both sensors are

in motion, and an alternative approach could be applied for the detection and tracking of an object. As future work it can be planning to use a more robust drone detection algorithm based on YOLO instead of the current background subtraction method. This will allow validation of the approach in dynamic environments.

The Extended Kalman Filter has proven to be an efficient method for data fusion, predicting the future state of tracked objects and continuously correct estimates based on new observations. The concept of sending two angles to the PTU has enabled to keep the target in the LiDAR FoV almost continuously. Thereby the update and predictor are more precise, resulting in an enhanced tracking system.

Bibliography

[1] Gabriel Carlisle Birch and Bryana Lynn Woo. Counter unmanned aerial systems testing: Evaluation of VIS SWIR MWIR and LWIR passive imagers. 1 2017.

[2] Y. Boers and J.N. Driessen. Particle filter based detection for tracking. In *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, volume 6, pages 4393–4397 vol.6, 2001.

[3] Osama Bushnaq, Debashisha Mishra, Enrico Natalizio, and Ian Akyildiz. *Unmanned aerial vehicles (UAVs) for disaster management*, pages 159–188. 01 2022.

[4] Luca Caltagirone, Mauro Bellone, Lennart Svensson, and Mattias Wahde. Lidar–camera fusion for road detection using fully convolutional neural networks. *Robotics and Autonomous Systems*, 111:125–131, 2019.

[5] Iacopo Catalano, Ha Sier, Xianjia Yu, Tomi Westerlund, and Jorge Peña Queralta. Uav tracking with solid-state lidars: Dynamic multi-frequency scan integrationroza. In *2023 21st International Conference on Advanced Robotics (ICAR)*, pages 417–424, 2023.

[6] Vinay Chamola, Pavan Kotesh, Aayush Agarwal, Naren, Navneet Gupta, and Mohsen Guizani. A comprehensive review of unmanned aerial vehicle attacks and neutralization techniques. *Ad Hoc Networks*, 111:102324, 2021.

[7] Marie-Neige Chapel and Thierry Bouwmans. Moving objects detection with a moving camera: A comprehensive review, 2020.

[8] Jung Uk Cho, Seung Hun Jin, Xuan Dai Pham, Jae Wook Jeon, Jong Eun Byun, and Hoon Kang. A real-time object tracking system using a particle filter. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2822–2827, 2006.

[9] Frank Christnacher, Sébastien Hengy, Martin Laurenzis, Alexis Matwyschuk, Pierre Naz, Stéphane Schertzer, and Gwenael Schmitt. Optical and acoustical UAV detection. In *Electro-Optical Remote Sensing X*, volume 9988, pages 83–95. SPIE, 2016.

[10] Álvaro Duque De Quevedo, Fernando Ibañez Urzaiz, Javier Gismero Menoyo, and Alberto Asensio López. Drone detection with x-band ubiquitous radar. In *2018 19th International Radar Symposium (IRS)*, pages 1–10. IEEE, 2018.

[11] Sedat Dogru, Rui Baptista, and Lino Marques. Tracking drones with drones using millimeter wave radar. In *Iberian Robotics conference*, pages 392–402. Springer, 2019.

[12] Sedat Dogru and Lino Marques. Pursuing drones with drones using millimeter wave radar. *IEEE Robotics and Automation Letters*, 5(3):4156–4163, 2020.

[13] Sedat Dogru and Lino Marques. Drone detection using sparse lidar measurements. *IEEE Robotics and Automation Letters*, 7(2):3062–3069, 2022.

[14] Louis Dressel and Mykel J. Kochenderfer. Hunting drones with other drones: Tracking a moving radio target. In *2019 International Conference on Robotics and Automation (ICRA)*, page 1905–1912. IEEE Press, 2019.

[15] Shide Du and Shiping Wang. An overview of correlation-filter-based object tracking. *IEEE Transactions on Computational Social Systems*, 9(1):18–31, 2022.

[16] Jan Farlik, Miroslav Kratky, Josef Casar, and Vadim Stary. Radar cross section and detection of small unmanned aerial vehicles. In *2016 17th International Conference on Mechatronics - Mechatronika (ME)*, pages 1–7, 2016.

[17] Pramod R. Gunjal, Bhagyashri R. Gunjal, Haribhau A. Shinde, Swapnil M. Vanam, and Sachin S. Aher. Moving object tracking using Kalman Filter. In *2018 International Conference On Advances in Communication and Computing Technology (ICACCT)*, pages 544–547, 2018.

[18] Marcus Hammer, Marcus Hebel, Martin Laurenzis, and Michael Arens. Lidar-based detection and tracking of small UAVs. In Gerald S. Buller, Richard C. Hollins, Robert A. Lamb, and Markus Mueller, editors, *Emerging Imaging and Sensing Technologies for Security and Defence III; and Unmanned Sensors, Systems, and Countermeasures*, volume 10799, page 107990S. International Society for Optics and

Photonics, SPIE, 2018.

[19] Folker Hoffmann, Matthew Ritchie, Francesco Fioranelli, Alexander Charlish, and Hugh Griffiths. Micro-Doppler based detection and tracking of UAVs with multistatic radar. In *2016 IEEE Radar Conference (RadarConf)*, pages 1–6, 2016.

[20] Wu-Chih Hu, Chao-Ho Chen, Tsong-Yi Chen, Deng-Yuan Huang, and Zong-Che Wu. Moving object detection and tracking from video captured by moving camera. *Journal of Visual Communication and Image Representation*, 30:164–180, 2015.

[21] Cheng-Ming Huang, Yi-Ru Chen, and Li-Chen Fu. Real-time object detection and tracking on a moving camera platform. In *2009 ICCAS-SICE*, pages 717–722, 2009.

[22] Fortune Italia. Venezia sperimenta droni per consegna pacchi, 2021.

[23] Yanqin Jiang, Li Zhang, Zhenwei Miao, Xiatian Zhu, Jin Gao, Weiming Hu, and Yu-Gang Jiang. Polarformer: Multi-Camera 3D object detection with polar transformer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(1):1042–1050, Jun. 2023.

[24] Larisa Kapustina, Natalia Izakova, Elizaveta Makovkina, and Michail Khmelkov. The global drone market: main development trends. In *SHS Web of Conferences*, volume 129, page 11004. EDP Sciences, 2021.

[25] Istvan Lauko, Adam Honts, Jacob Beihoff, and Scott Rupprecht. Local color and morphological image feature based vegetation identification and its application to human environment street view vegetation mapping, or how green is our county? *Geo-spatial Information Science*, 23:1–15, 08 2020.

[26] Cosimo Leuci, Sedat Dogru, and Lino Marques. Camera lidar fusion for unmanned aerial vehicle detection. In *2024 7th Iberian Robotics Conference (ROBOT)*, pages 1–6. IEEE, 2024.

[27] Yingwei Li, Adams Wei Yu, Tianjian Meng, Ben Caine, Jiquan Ngiam, Daiyi Peng, Junyang Shen, Yifeng Lu, Denny Zhou, Quoc V Le, et al. Deepfusion: Lidar-camera deep fusion for multi-modal 3D object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17182–17191, 2022.

[28] Zhongmin Li and Haochen Wu. A survey of maneuvering target tracking using kalman filter. In *2015 4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering*, pages 542–545. Atlantis Press, 2015.

[29] Leyuan Liu, Jian He, Keyan Ren, Zhonghua Xiao, and Yibin Hou. A lidar–camera fusion 3D object detection algorithm. *Information*, 13(4), 2022.

[30] Christos Milias, Rasmus B. Andersen, Bilal Muhammad, Jes T. B. Kristensen, Pavlos I. Lazaridis, Zaharias D. Zaharis, Albena Mihovska, and Dan D. S. Hermansen. Uas-borne radar for autonomous navigation and surveillance applications. *IEEE Transactions on Intelligent Transportation Systems*, 24(7):7215–7229, 2023.

[31] Drone Blog News. Tecnologia avanzata contro gli incendi: arrivano gli sciami di droni, 2025.

[32] Phuc Nguyen, Mahesh Ravindranatha, Anh Nguyen, Richard Han, and Tam Vu. Investigating cost-effective rf-based detection of drones. In *Proceedings of the 2nd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, DroNet '16, page 17–22, New York, NY, USA, 2016. Association for Computing Machinery.

[33] OpenCV Documentation Team. Camera calibration and 3D reconstruction, 2024. Accessed: 2024-07-21.

[34] Viola Paul. Robust real-time object detection. In *Second International Workshop on Statistical and Computational Theories of Vision-Modeling, Learning, Computing, and Sampling, Vancouver, Canada, 2001*, 2001.

[35] Panagiotis Radoglou-Grammatikis, Panagiotis Sarigiannidis, Thomas Lagkas, and Ioannis Moscholios. A compilation of UAV applications for precision agriculture. *Computer Networks*, 172:107148, 2020.

[36] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Flying objects detection from a single moving camera. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4128–4136, 2015.

[37] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[38] Xiufang Shi, Chaoqun Yang, Weige Xie, Chao Liang, Zhiguo Shi, and Jiming Chen. Anti-drone system with multiple surveillance technologies: Architecture, implementation, and challenges. *IEEE Communications Magazine*, 56(4):68–74, 2018.

[39] Ha Sier, Xianjia Yu, Iacopo Catalano, Jorge Peña Queralta, Zhuo Zou, and Tomi Westerlund. Uav tracking with lidar as a camera sensor in gnss-denied environments.

In *2023 International Conference on Localization and GNSS (ICL-GNSS)*, pages 1–7, 2023.

[40] D. Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, 2006.

[41] Jasmina Pašagić Škrinjar, Pero Škorput, and Martina Furdić. Application of unmanned aerial vehicles in logistic processes. In Isak Karabegović, editor, *New Technologies, Development and Application*, pages 359–366, Cham, 2019. Springer International Publishing.

[42] OpenCV Team. *OpenCV: Camera Calibration and 3D Reconstruction*, 2025.

[43] Evgenii Vinogradov, Dmitry A Kovalev, and Sofie Pollin. Simulation and detection performance evaluation of a uav-mounted passive radar. In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1185–1191. IEEE, 2018.

[44] Matouš Vrba, Viktor Walter, Václav Pritzl, Michal Pliska, Tomáš Báča, Vojtěch Spurnỳ, Daniel Heřt, and Martin Saska. On onboard lidar-based flying object detection. *arXiv preprint arXiv:2303.05404*, 2023.

[45] Pan Wei, Lucas Cagle, Tasmia Reza, John Ball, and James Gafford. Lidar and camera detection fusion in a real-time industrial multi-sensor collision avoidance system. *Electronics*, 7(6), 2018.

[46] Mehran Yazdi and Thierry Bouwmans. New trends on moving object detection in video images captured by a moving camera: A survey. *Computer science review*, 28:157–177, 2018.

[47] Feihu Zhang, Daniel Clarke, and Alois Knoll. Vehicle detection based on lidar and camera fusion. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1620–1625, 2014.

[48] Shuai Zhang, Chong Wang, Shing-Chow Chan, Xiguang Wei, and Check-Hei Ho. New object detection, tracking, and recognition approaches for video surveillance over camera network. *IEEE Sensors Journal*, 15(5):2679–2691, 2015.

[49] Yueyan Zhi, Zhangjie Fu, Xingming Sun, and Jingnan Yu. Security and privacy issues of UAV: A survey. *Mobile Networks and Applications*, 25(1):95–101, 2020.