STEREO VISION-BASED SYSTEM FOR DETECTION, TRACK AND CAPTURE OF INTRUDER FLYING DRONES

by

Maria Nieves Brunet Avalos

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Electrical and Computer Engineering

School of Electrical and Computer Engineering

West Lafayette, Indiana

May 2020

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF COMMITTEE APPROVAL

Dr. Mo Rastgaar, Chair

Purdue Polytechnic Institute

Dr. Shreyas Sundaram, Chair

School of Electrical and Computer Engineering

Dr. Jianghai Hu, Member

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

Head of Electrical and Computer Engineering

Dedicated to my family and friends.

ACKNOWLEDGMENTS

I would like to put into words my sincere thanks to Professors Shreyas Sundaram and Mo Rastgaar for their constant support, guidance and encouragement in the course of this work, and for always making time to give me guidance and direction when I faced obstacles in my program and with my project. I greatly appreciate the opportunities mentioned by both to expand my engineering knowledge and experience. Participating in meetings with their research groups provided me with the preparation to excel in the presentation of this work.

I would also like to extend my gratitude to all the members of the Human Interactive Robotics Lab (HIRoLab), for their insights, patience, and humor. Special thanks to Dr. Guilherme Aramizo whose guidance was key for the development of this project.

I am grateful to the Fulbright Commission, the BECAL scholarship from Paraguay, and Purdue University for allowing me to pursue my master's degree surrounded by great people.

Lastly, I would like to express my thanks to my family and friends, who kept me strong along this journey by being immensely supportive and positive.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

2D Two Dimension

3D Three Dimension

AI Artificial Intelligence

CPU Central Processing Unit

CV Computer Vision

ESC Electronic Speed Controller

FCW Forward Collision Warning

FN False-Negative

FP False-Positive

FPS Frames Per Second

GPS Global Positioning System

GT Ground Truth

IMU Inertial Measurement Unit

KF Kalman Filter

LiDAR Light Detection And Ranging

LiPo Lithium Polymer

MAV Micro Aerial Vehicle

MOT Multi Object Tracking

MOTA Multi Object Tracking Accuracy

PWM Pulse Width Modulation

RF Radio Frequency

RMSE Root Mean Square Error

ROS Robot Operating System

TP True-Positive

UAV Unmanned Aerial Vehicle

ABSTRACT

In this thesis, the design and implementation of an autonomous system that will equip a multi-rotor unmanned aerial vehicle (UAV) for visual detection and tracking of other UAVs is presented. The results from detection and tracking are used for real-time motion planning.

The goal is to effectively detect unwanted UAVs, track them and finally capture them with a net. Having a net that traps the UAVs and enables dragging intruders to another location is of great importance, since these could be carrying dangerous loads.

The project consists of three main tasks: object detection using a stereo camera, video tracking using a Kalman filter based algorithm, and lastly executing an optimal flight plan to aim a net at the detected intruder UAV. The computer vision, motion tracking and planning algorithms are implemented as ROS nodes what makes them executable on a reduced size onboard computer that is installed on the aerial vehicle.

Previous work related to this project consists of either a UAV detection system with computationally heavy algorithms or a tracking algorithm that does not include information about the dynamics of the UAVs. For the capture methods, previous ideas do not consider autonomous decisions or an optimized method to guarantee capture. In this thesis, these three aspects are considered to develop a simple solution that can be mounted on any commercially available UAV.

1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) and multi-copter drones, keep increasing in quantity. As a result of reduced prices, ease of access, advancements in technology, and creation of developer communities, numerous applications for UAVs were enabled. That has also led to the misuse of this technology. For example, commercial UAVs can be easily misused by carrying harmful material to populated areas and important infrastructure [1], or to perform unwarranted reconnaissance. Therefore, there is a demand for systems that can help safely reduce the risk that UAV misuses may pose to the security of people and infrastructure. For these cases, the use of a UAV hunter to intercept, capture, and remove intruder UAVs to a safe zone for neutralization is proposed.

Previously a prototype hunter [2] that can capture other UAVs by trapping the intruder UAV in a tethered net and dragging it to a remote location was developed by the research group. For that system, the operator of the hunter UAV prototype used the video downlink feedback from an onboard camera to manually aim and trigger the net launching mechanism. In this thesis the development and evaluation of a visual-based UAV tracking algorithm and a UAV motion controller to automate the aiming of the hunter UAV is presented.

1.1 Motivation

Drones can be used in the wrong way, either for terrorism, incidents in mass events, interference with air traffic, spying, or disrespecting privacy. There have been several incidents reported to be caused by UAVs.

Looking at the future, the projected number of hobbyist UAVs in use by 2021 is around 3.5 million. E-commerce is a market looking to benefit by the drone industry's growth. As indicated by a review, 79% of U.S. web users are positive about selecting drones as a means for delivery [3]. Another examination that the United States Postal Service directed, found that 56% of users consider drone deliveries possibly being quicker, and at the same time 53% believe deliveries using drones will be more environmentally friendly. These are only a few reasons that assure that the UAV market, and therefore their availability and use, will only grow in number in the following years.

There is also a downside to the growing number of UAVs, and that is the threat to safety. In 2018, drones carrying explosives approached as Venezuelan President gave a speech and the attack injured seven soldiers [4]. On December 2018, some flights had to be delayed due to the presence of drones near London's Gatwick Airport [5]. Another concern is that drones can be used to spy on people. A simple investigation simulated ordinary activities both downstairs and upstairs in a typical house, and showed that it was easy for a drone to monitor a person on both floors while hovering out of sight [6].

To counter the misuse of drones, a solution is needed to help decrease the risks without compromising public safety.

1.2 Research objective

The objective of this work is to devise a reliable system that minimizes the danger of unwanted UAVs. The proposed system consists of one UAV capable of detecting the presence of other UAVs. A UAV is chosen as the detection platform since it has the capability of reacting with dynamics comparable to that of the desired target.

The final intention of this project is to mount a visual detection device on the platform hunter drone. From the visual feed the location of the invader will be estimated and then an autonomous planner will determine the best way to approach the intruder. Finally, a net launcher will be integrated as mentioned in the Introduction.

1.3 Assumptions

To limit the scope of the project, it is supposed that the hunt starts somewhere close to the desired target. This is a reasonable assumption, since the protected area can have a local radar that informs the hunter of the approximate location of an intruder. Alternatively, it can be assumed that a fleet of drones flying constantly around a protected environment would detect the presence or approach of an undesired UAV, that would be communicated to the hunter drone.

Another consideration is that the target UAV is of the size and shape of commercial, general-use UAVs available as of today. Therefore, the dynamics of the proposed hunter UAV will be carefully selected so that it can be as fast and maneuverable as the targets. This study does not include large and fast military missile type drones.

The hunter UAVs are thought to be hexacopters or octocopters, which provide a more stable platform with the ability to carry heavier payloads.

1.4 Notation and Terminology

Throughout the text, the designed UAV is referred to as the *hunter*, and the unwanted UAV is referred to as the *intruder*. Coordinate frames fixed to the world (inertial reference frame) and to the hunter UAV are used, labeled as *W* and *C*, respectively. Figure 1.1 shows the coordinate frames of the world, UAV, and the camera. The hunter drone will carry the stereo camera and net launcher system. The X-Y-Z world axes point to East-North-Up (ENU coordinate frame convention), Z and Y for the UAV axes point forward and down, respectively.

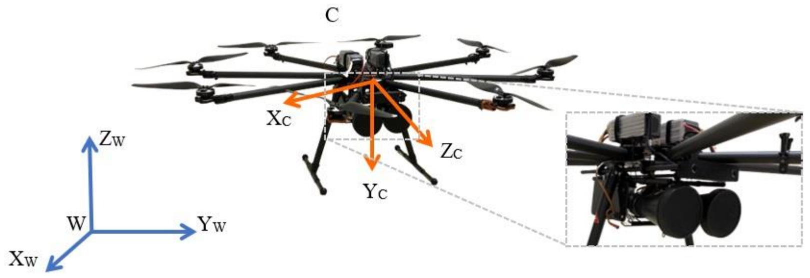


Figure 1.1 Coordinate frames of the world, UAV, and camera. The hunter drone will carry the stereo camera and net launcher system. Image of the Tarot T18 drone carrying the proposed net launcher and camera system. Tarot T18 can carry an 8 kg payload and it is proposed as the final hunter prototype.

Only for clarity of notation, it is considered that the hunter UAV and camera are fixed to each other, and that they share the same coordinate frame.

The *net launcher* is the system that consists of the net cannon and will be carried by the hunter drone. It has two nets containing bullets on the ends, to ensure a conic opening of the net used to capture the intruder drone.

1.5 Limitations

The limitations of the Hunter drone are the range of detection of around 10m, the size of the target UAVs, and the possibility of launching two nets. In future work these can be improved.

2. LITERATURE REVIEW

2.1 Object detection

Object detection is a pre-requisite for the tracking process, as it is necessary to find the location of the object of interest in every frame. There are various approaches for detection found in the literature.

Radar sensors have been a crucial part of safety systems because of their weather and lights independence. For this reason, radar is considered good for handling the detection and classification of a variety of targets in city environments. Despite this factors, radars can face challenges for the detection of UAVs, since these are low objects and fly with changing speed [1]. Low altitude and slow speeds may cause difficulties for separating the target from a large clutter response. Additionally, the drones need to be differentiated from biological and other environmental targets like birds and insects often present in the same monitored regions. Birds and drones may have comparable values and flying patterns for radar signals, which produces a hard challenge for classifiers to difference them.

A Radio Frequency (RF)-based approach to identify the presence of UAVs is proposed in [7]. In general, commercial UAVs communicate with the pilot via a controller that uses RF communication. By detecting these signals, they estimate the presence of UAVs. The interference for this method was reduced implementing a background filtering method.

Acoustic sensors can provide an inexpensive object detection by, for example, an array with dynamically placed microphones as presented in [8]. The method can identify and estimate the position of the broadband sources. The paper proposes a calibration strategy to resolve the best arrangement of the microphones to be placed. The experiments demonstrated favorable results.

Laser sensors have great potential as object detectors. In particular the method described in [9], uses an inertial measurement unit and laser scanners to form a map of the surroundings and perform obstacle avoidance. Laser sensors can generate point clouds and thus form a 3D map of the environment. Constantly updating such a grid is computationally expensive as every ray scanned by the laser has to be updated for all the cells it travels through.

LiDAR (Light Detection And Ranging) sensors emit laser rays of light and by analyzing the reflection from the surrounding objects obtain a description of the environment. By filtering details in the 3D scan of the scene, the probability of detecting objects arises. The work from [10] shows that UAVs can be detected by this method if the dynamics of the UAVs fall within the LiDAR sensor's capabilities in scanning performance, range, and resolution. The downsides of LiDAR are the high cost and the vulnerability to weather conditions.

2.1.1 Computer vision

Object Detection is a common Computer Vision problem that aims to identify and locate objects in a digital image or video sequence. There are a few attempts to detect and track moving objects using camera-based systems in UAVs. Background subtraction [11] is extensively used in video sequences when they have a static background. This idea focuses on separating the image into the foreground (moving objects) and background (static objects).

Temporal differencing presented in [12] differs from the traditional background image subtraction method by taking differences of consecutive frames, making it more suitable for scenarios where the camera is moving.

Optical Flow [13] is the measure of how far a pixel moves from one frame to another. Optical flow creates a vector field in the plane of the image, also known as a motion field. This represents the velocity and direction of the pixels in successive frames. Discontinuities in the optical flow help to split each frame into areas that represent different objects. This method is computationally expensive but has an advantage of detecting moving objects in video feeds having a non-static background.

Object detection can also be achieved by training a classifier. Deep learning classifiers can learn different object views and appearances. A trained classifier can recognize different objects and decide if they correspond to the target object or not. These trained classifiers show powerful detection performance even in challenging environments with lightning variations and background clutters. However, to extract shape and appearance features, they assume sufficiently large and clearly visible moving objects which are not sometimes the cases for UAV applications.

This project is based on stereo imaging and computer vision and further review of the method is presented in the next section.

Stereo Vision

Normally, visual object detection is handled using videos acquired from a single camera setting. Yet, replacing a single camera by a multi-camera system results in more information about the environment. The most common multi-view camera system consists of two cameras, also known as a stereo camera.

The stereo correspondence was a very active topic of research in the computer vision field in the 1970s. Stereo imaging consists of obtaining two images and from them, deducing a three-dimensional map of the scene. From personal experiences, it is known that the perception is different when comparing the left and right eye images. These two different images allow the reconstruction of the 3D understanding of the environment [14]. The sense of depth comes from the distance (*disparity*) for the same point between the left and right retinal images.

A stereo parallel camera is shown in Figure 2.1. A parallel stereo camera has two cameras to obtain the right and left images. The cameras are placed facing in the same direction, separated by a horizontal baseline distance. The correspondence between depth (Z) and disparities (d) is explained by equation 2.1, where f and b are the focal and baseline distance respectively (both parameters of the stereo camera). This method of determining disparity from depth is called triangulation.

$$d = f\frac{b}{Z} \qquad (2.1)$$

An object detector using a stereo vision camera is presented in [15]. This idea detects the location of a moving object and estimates the tracking error to servo-actuate a loop in the eye-hand system structure of a humanoid robot. An application of depth images used for a fall detector for elderly people is presented in [16]. In this case, the contour detection of the walking figure is achieved by subtracting the background frame and estimating the floor plane in the depth images.

The effect of motion, when a camera is in movement is analyzed in [17]. A vehicle tracking procedure based on stereo vision is presented there, where an autonomous robot is actuated and the camera is under the movements transmitted to the structure. This study shows the incidence of the camera movements in the calculation of the range and direction angle of the complete system.

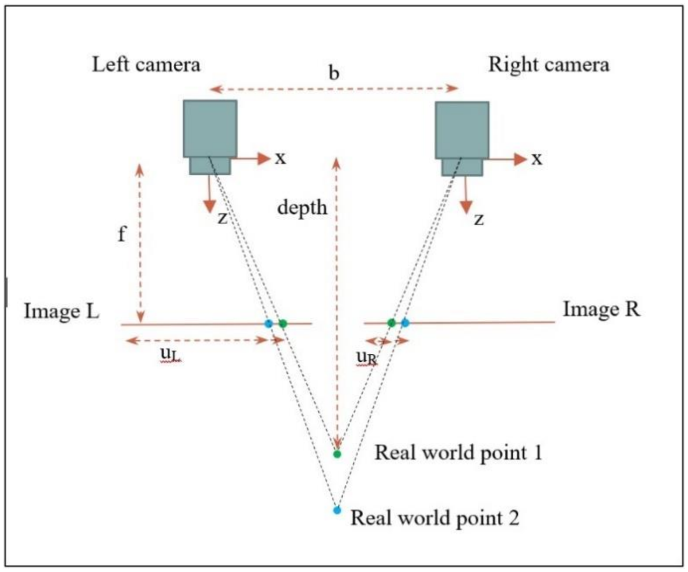


Figure 2.1. Parallel stereo camera.[18]

2.2 Video tracking

Many attempts are described in the literature to track the position of moving objects in a video feed. Multiple object tracking (MOT) can be divided in two parts, the first is object detection (previous section 2.1) and the second is motion prediction. The target detection is based on image processing and thus it tends to require a large part of the computational resources.

For multi-object tracking, the main challenge is the label association problem, for connecting object detections to their respective tracks. Most of the batch methods formulate MOT as an optimization problem. Other online methods try to achieve the data association either probabilistically or determinatively [19].

The work from [20] features a football player tracking algorithm where the implementation is based on a modified Hungarian algorithm for the data association and the prediction being obtained by the use of the Kalman filter. The most important part of the tracking algorithms is the ability of the system to manage an object's identity even in occlusion scenarios or at boundary conditions. An approach based on Markov Decision Processes is found in [19], where the MOT problem is described in terms of decisions from a Markov Decision Process.

The method of Multiple Hypotheses Tracking (MHT) is presented in [21]. For every new observation, MHT analyzes the possible associations by keeping the potential object-to-track assignments represented as target trees. When new observations arrive for each frame, the target trees keep spawning, and growing. The trees keep a set of hypotheses that propagate with the hope that future input data will help resolve any assignment ambiguities. The potential thus is that MHT can salvage wrong object-to-track associations.

For tracking in videos recorded by UAVs, an improved method for object tracking is presented in [22]. This work implements a Mean Shift algorithm based on particle filtering, to optimize the search of the origin of the Mean Shift algorithm. This idea can achieve the tracking of fast-moving objects. A study for tracking vehicles was presented in [23]. In this work, the state estimation for the agents is obtained using a Kalman filter method. The data association is done through the Hungarian method. For this study, an extension was needed to prevent assignments of objects too far apart from each other in space. This method is the most similar to the one being implemented in this work.

2.3 UAV capture methods

To neutralize unwanted UAVs, work is being done to develop a drone-killing microwave weapon to knock down drones with pulses of energy [24]. Also, radio control frequencies that disrupt commercial drones' communications, smart bullets, and even mobile high-energy laser weapons to damage or destroy UAVs are being thought of. The problem of these attempts is that they either alter the flight path or force-land the UAV to an unpredictable location. If the UAV is carrying harmful materials, dropping the UAV to the ground is not an effective way to protect people or infrastructure.

One valid approach was the use of a net hanging from a UAV as was shown by the Japanese police [25], where the hunter system captures the intruder UAV by flying over it and having the propellers caught in the mesh of the net. With that method, the possibility exists for the target drones falling from the sky and causing harm or damage.

2.4 Robotic operating system

The Robot Operating System (ROS) is an open-source framework for robotic systems development. ROS is considered a meta-operating system, because despite not being considered an operating system in the same way Windows or Linux are, it provides a system of nodes that allows inter-processes to occur within the selected intelligent platform. These allow the sharing of functional messages. The architecture of a ROS system consists of five components: a ROS Master, nodes, publishers, subscribers, and topics.

ROS allows the creation of robust software for robot platforms in general, multiple sensors and effectors. Nodes in ROS are pieces of software written in C++ or Python. It allows the functionality of a robot to be divided into subsets of tasks. In further chapters, it will be shown how different parts of the system are instantiated as nodes. ROS grants the communication between different nodes. In this way, the result obtained from a node can be an input to another.

The communication between nodes is through topics. Topics use the same type of message for the node publishing and the node subscribing [26]. ROS is suitable for sensor data, since topics are unidirectional and stay continuously connected to send or receive messages. Also, multiple subscribers can receive a message from a publisher and vice versa. Multiple publishers and subscribers’ connections are available as well (Figure 2.2).

Among the advantages of using ROS, the nodes can be written in either C++ or Python and work perfectly together. Also, nodes can run on different hardware and be connected through a common network. ROS is open source and extensive documentation can be found.

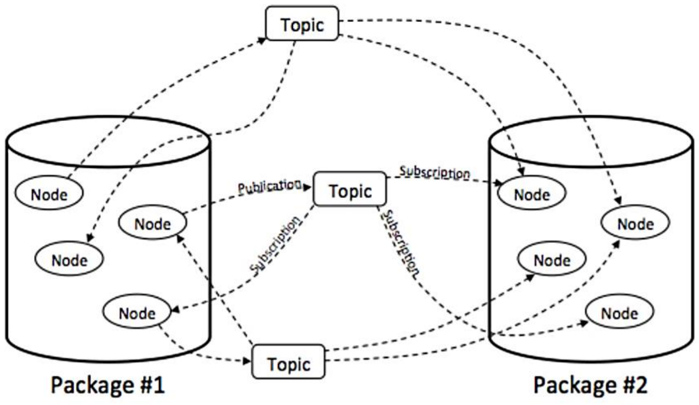


Figure 2.2. ROS communication between nodes.
Nodes can publish and subscribe to topics [27].

2.5 Summary

The detection method for the project will be based on a vision system, a stereo camera. A stereo camera gives the same information as a single camera plus the information of distance to objects in the scene. The tracking method will be based on the idea of a Kalman filter for prediction of the location of the object, and the Hungarian algorithm to associate the detected objects to their tracks. The capture system will be mounted on the hunter drone, carrying a net to have the invader drone tangle in it. The selected platform for development is ROS, providing the modularity for the design of the system. Figure 2.3 illustrates the pipeline for the project.

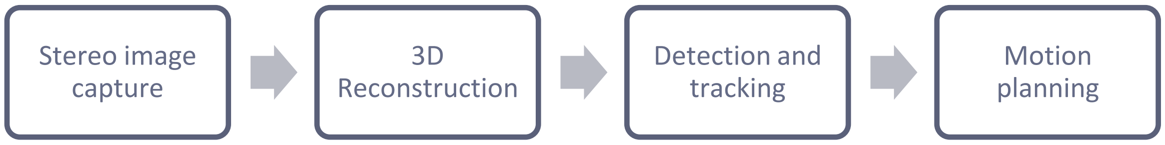


Figure 2.3. Simplified project pipeline.

3. PROPOSED METHODOLOGY

3.1 3D Reconstruction

The main purpose of the Vision System in this project is to localize an object, so that the hunter system can act upon this information. There are a few things to consider when choosing the detection method. The detection system is mounted on a UAV and sends the feed to an onboard computer. Thus, the weight, communication protocol, and specific parameters related to the detection must be considered.

The first limitation is weight, as the drone can only carry a certain amount of weight. More payload directly impacts the battery life. Thus, choosing a device that is as light as possible is important. Another requirement is the processing capabilities required for detection. The data coming from the device that does the detection will have to be processed by an onboard computer that has limited capabilities. Concerning the video quality, since the target is another flying UAV, the detection needs to be able to detect a fast-flying object in the video feed. A good option to account for these requirements is a camera.

A stereo camera can help to detect objects by using both the depth information (3D environment reconstruction) and the monocular images for a further match of image features (2D image classification). The depth information provided by a stereo camera makes tracking features over a long distance comparatively robust. This method is effective and practical, as NASA had a stereo camera on Curiosity, the Mars Rover.

The 3D image reconstruction pipeline is illustrated in Figure 3.1.

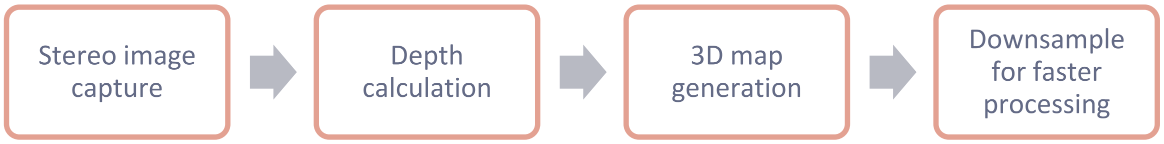


Figure 3.1. 3D Image reconstruction pipeline.

3.1.1 Stereo cameras

Two stereo cameras were selected at the beginning to be evaluated for the project, WithRobot oCamS-1CGN-U and Mynteye S1030. The characteristics of these are shown in Table 3.1.

Table 3.1. Stereo cameras considered for the project.

	WithRobot oCamS-1CGN-U	Mynteye S1030
Resolution	1280 x 720 @ 60 FPS	752 x 480 @ 60 FPS
Field of view	V: 50˚ H: 92.8˚	V: 76˚ H: 122˚
Weight	30 grams	85 grams
IR Light	No	Yes
Range	0.5 ~ 12 m	0.5 ~ 18 m
Baseline	120 mm	120 mm
IMU	Yes	Yes
Color images	Yes	No

Figure 3.2. WithRobot oCamS-1CGN-U. Image retrieved from http://withrobot.com/en/camera/ocams-1cgn-u/.

Calibration

To calibrate the stereo cameras, the *camera_calibration* library was used within ROS. This library takes a left and right image of a printed checkerboard of known dimensions to perform the parameter tuning. A precise calibration is key to obtaining the best results of depth in the video feed.

The stereo cameras have two sets of parameters that must be tuned. The *Intrinsic* parameters are internal and referent to the camera setup; these allow a correspondence between the camera coordinates and the pixel coordinates in the image (camera model parameters, focal length, lens distortion). The *Extrinsic* parameters are external to a camera; these are referent to the spatial disposition of the camera with reference to a fixed frame (both rotation and translation can be expressed as a homogeneous transformation matrix). The extrinsic parameters consider the location and orientation of the two cameras related to a common world frame (Figure 3.3).

In the case of Mynteye S1030, the calibration parameters can be uploaded and saved to the camera directory itself. For WithRobot oCamS-1CGN-U, the parameters are stored in the user's directory and loaded every time the camera is launched.

The selected camera for the rest of the work is WithRobot oCamS-1CGN-U (Figure 3.2, Figure 3.3). It has shown a better resolution for the same frame rate, lighter weight, and color images. Another interesting feature of this camera is the possibility to customize the field of view by changing the lenses.

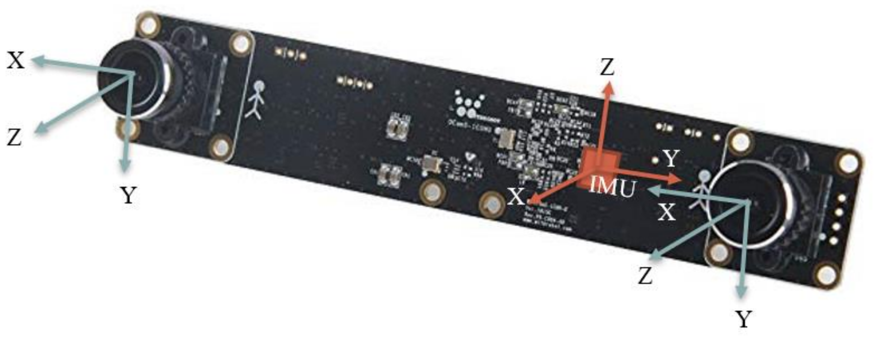


Figure 3.3. oCamS-1CGN-U, showing the coordinate systems of the IMU sensor and the image sensors. Image retrieved from https://us.amazon.com/Stereo-Shutter-Disparity-WITHROBOT-oCamS-1CGN-U/dp/B07R5NG6HK.

3.1.2 Preprocessing

The stereo camera provides the image feed for the left and right cameras. The following step uses the library *stereo_image_proc* to obtain the Disparity image and the Point Cloud. In the disparity image, every pixel contains information about the disparity (the difference in location for the same features) within two stereo images. In the case of a point cloud, which is an assembly of points in three-dimensional space, each point presents the spatial location from the camera to objects in the environment.

Using the disparity image

The first tests for object detection using a stereo camera were done on disparity images. For this approach the first step consists of the use of a filtering stage to smooth the images and remove noisy data.

After the filtering stage, a ground removal algorithm is needed to get rid of areas that can be close to the camera and not representative of a UAV. A RANSAC (Random Sample Consensus) algorithm was used to approximate the ground plane and remove it from the disparity image. After that step, the remaining image was segmented considering the size and location of the objects in the depth axis of to the camera. Morphological operations play an important role in segmenting the right blobs in a disparity image [28]. The results from this stage were promising, but there is more potential in using 3D information of the environment.

Using point cloud data

An array of points in 3D space is obtained from the left and right images. Using the camera with a VGA resolution (640x480 pixels), running at 30 FPS, the amount of points to process every second is of approximately nine million points. Thus, the processing cost for every frame will be expensive.

Downsampling the point cloud data (reducing the number of points), yields a less populated point cloud with similar information of the environment. This procedure reduces the computational cost. With this goal, a Voxel Grid filter is considered [29]. This filter generates a three-dimensional voxel grid (seen as boxes in 3D space) over the original point cloud data. For each voxel (each box in 3D), all the points in that region will be approximated with their centroid. Voxel Grid is an implementation of the method from the Point Cloud Library.

A filtering stage is launched after the data is downsampled. The point cloud is restricted to the working distance range, between 0.5 ~ 10 m in the camera z-axis.

3.2 Object detection

3.2.1 Cluster extraction

A Euclidean clustering algorithm identifies clusters of points after the point cloud is filtered. A cluster of points is at least within 0.5 m from the next one. Clusters with more than 200 points or less than 20 points are removed since they likely do not represent the desired UAV. The detections are selected as the centroid of each remaining cluster. Again, the Euclidean Cluster Extraction is an implementation of the method from the Point Cloud Library.

3.2.2 Metrics used for object detection

For performance interpretation of the detection method, the Precision-Recall curves were considered as in [30]. The precision and recall metrics evaluate the performance in detecting all UAV appearances and in ignoring environmental or sensor noise. The equations 3.1 and 3.2 present the calculation of these indices. The quantities considered are the amount of true-positive (TP), false-negative (FN), and false-positive (FP) detections.

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

3.3 Video tracking

The essence of multi-object tracking is matching objects in successive frames of a video feed. The objects of interest within the images would be tracked as the objects move around in the image in successive frames. The tracking starts by initializing the position of the object with the information coming from the objects' detection.

For this work, the detected UAVs need to be tracked. A prediction of their motion is useful to anticipate the trajectory of the invader. For this purpose, a Kalman filter approach is implemented as the tracking method.

3.3.1 Kalman Filter

A discrete Kalman filter algorithm evaluates the state variables for a system, based on inaccurate and uncertain measurements. For the studied case, the Kalman filter tracks the intruder UAV's position, velocity, and acceleration, given the noisy visual detections. The Kalman filter propagates the current track state into the future assuming a motion for the intruder UAV.

Note that the sensed information (detections) can support or disprove the predictive model over time, therefore, the algorithm keeps multiple UAV tracks at any time and continuously adds or eliminates tracks depending on the incoming detected objects. New tracks are created when incoming detections are too far from all existing tracks. Similarly, deficient tracks are removed when they do not correspond to incoming detections after several consecutive iterations.

This tracking framework has the advantages of: 1) smoothing the trajectory of the track for stable aiming, 2) interpolating temporarily-undetected UAVs, 3) estimating the velocity and accelerations of the UAV, and finally, 4) quantifying the statistical confidence of the UAVs' position, all using low computation resources [20].

The filter assumes a motion model of constant acceleration in 3D space; thus, the track state is $X = [x, \dot{x}, \ddot{x}]^T$ (position, velocity, and acceleration respectively). This is not entirely true, since the UAV track has a time-varying acceleration. This variation was incorporated as an additive random noise to the state update equation.

The covariance matrix of this noise in the discrete space is defined as in equation 3.3, where T is the interval between updates.

$$Q^x = q \begin{bmatrix} T^5/_{20} & T^4/_8 & T^3/_6 \\ T^4/_8 & T^3/_3 & T^2/_2 \\ T^3/_6 & T^2/_2 & T \end{bmatrix} \qquad (3.3)$$

The Kalman filter uses noisy detection measurements to correct the state estimates. It assumes a 3D reconstruction error with a noise covariance matrix of $R^x = r * I$, where r is a scaling factor. The state initialization error is modeled as a random variable with a covariance matrix in equation 3.4.

$$P^x{}_0 = p \begin{bmatrix} 1^2 & 0 & 0 \\ 0 & 30^2 & 0 \\ 0 & 0 & 5.5^2 \end{bmatrix} \qquad (3.4)$$

These numbers were selected since 1.0 m, 30 m/s, and 5.5 m/s^2 are reasonable maximum 3D reconstruction error, UAV speed, and UAV acceleration. Note that Q^x, R^x, and $P^x{}_0$ are respectively the model noise covariance, the measurement noise covariance, and the initial state error covariance matrices of a Kalman filter (Figure 3.4). The parameters q, r, and p are scaling factors, manually tuned based on the intruder UAV maneuverability and the stereo camera noise.

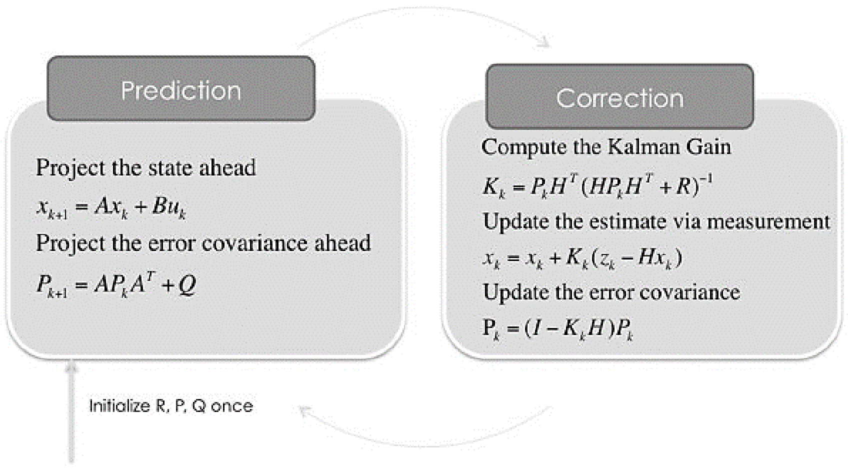

Prediction
Project the state ahead
$x_{k+1} = Ax_k + Bu_k$
Project the error covariance ahead
$P_{k+1} = AP_kA^T + Q$
Correction
Compute the Kalman Gain
$K_k = P_kH^T(HP_kH^T + R)^{-1}$
Update the estimate via measurement
$x_k = x_k + K_k(z_k - Hx_k)$
Update the error covariance
$P_k = (I - K_kH)P_k$
Initialize R, P, Q once

Figure 3.4. Discrete Kalman filter, the involved matrices are A: Dynamics, B: Control input, P:State error covariance, H: Measuring, I: Unit , Q: Process noise covariance, R: Measurement noise covariance.

3.3.2 Linear Sum Assignment

Often there might be multiple active tracks at a moment in time and multiple incoming detections from a single camera frame (Figure 3.5). Using the linear sum assignment, also known as the Hungarian algorithm, detections are assigned to tracks by minimizing the sum of distances between the detections and their assigned tracks [31].

New tracks are created to accommodate detections that are farther to any other track by a maximum position error, ε_{max}. The state of the tracks that are not assigned to any detections are still projected into the future via the update stage of the Kalman filter algorithm (this increases the state error covariance); however, since they do not incorporate information from the measurements (which decreases the state error covariance), their state error increases.

If the position error exceeds ε_{max}, the track is eliminated. Finally, the longest living track was selected for aiming after observing that the detections due to environmental or camera noise are transient, and their corresponding tracks are quickly eliminated.

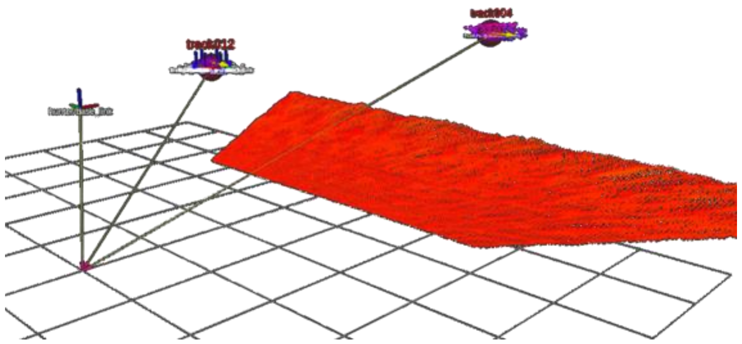


Figure 3.5. Multiple detections and tracks in the same frame. Image obtained from simulated experiments.

3.3.3 Performance

For evaluation of the tracking method, the MOTA (Multiple Object Tracking Accuracy) index as used by MOT16 (a standardized measure for object tracking), was used as a reference [32]. This framework is widely used for evaluation of the performance of Multi-Object Tracking in videos, since it combines multiple sources of errors. Besides the errors considered for Precision and Recall; they are true-positive (TP) that describes an actual target, false-positive (FP) in the case of an erroneous target, and false-negative (FN) when a target that is missed. MOTA also accounts for the incorrectly matched error or Identity Switch (IDSW). An identity switch occurs when a target has been associated with a different track number. MOTA is reported by the next equation (3.5), where the index *k* refers to the frame number and GT represents the quantity of ground truth targets, known by empirical evidence.

$$MOTA = 1 - \frac{\sum_k (FN_k + FP_k + IDSW_k)}{\sum_k GT_k} \quad (3.5)$$

In Figure 3.6, an ID switch occurs when the mapping switches from the previously assigned green track to the blue one. In frame 1 a false-negative is present. For frames 2 and 3, true-positive detections for the green target are tracked. For the frames 4,5 and 6, true-positive detections are tracked for the blue target. An identity switch occurs in the frame 4 where the blue target is assigned to the same track that was started for the green target.

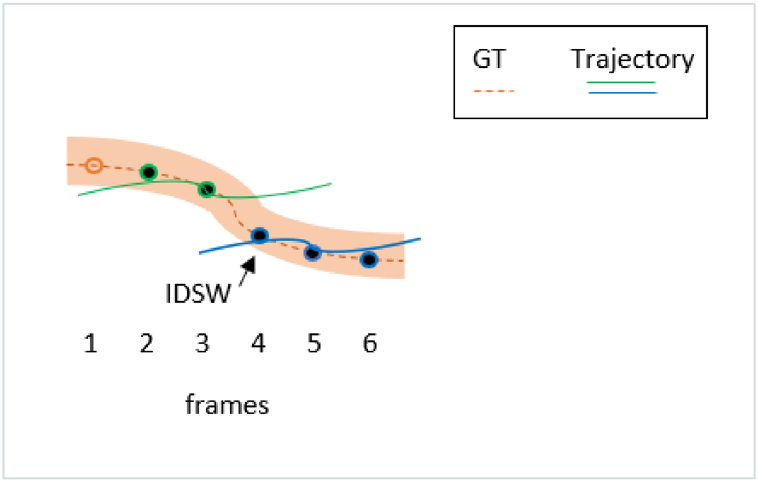


Figure 3.6. Figure adapted from [32]. A case illustrating an identity switch in the tracker-to-target assignments.

3.4 Motion control

For the control of the hunter UAV, the work of [33] is taken as reference. There, a geometric controller is studied and presented for the case of a quadcopter. The dynamics are studied considering them part of the special Euclidean group SE(3). This controller focuses on tracking a set of trajectories for the position and heading of the UAV.

For simplicity, the hunter will be controlled for position in x and z and orientation in the yaw axis (Figure 3.7). The motion controller will be implemented using the available implementation for the *lee_position_controller* in the RotorS package for ROS.

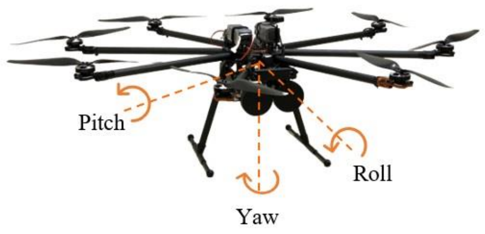


Figure 3.7. Rotations: Roll is the rotation around the front-to-back axis, Pitch is the turn about the side-to-side axis and Yaw is the direction of the heading of the UAV.

The theory for the hunter algorithm is adapted from [34]. The dynamical model can be represented by the two defined coordinate frames, W and C (Figure 1.1). Euler angles are used to define the hunter's attitude (roll, pitch, yaw) which are convenient to define the rotation matrix between the two coordinate frames W and C.

For the position control of the UAV, the dynamical model developed by [33] aims to obtain asymptotic behavior for the track of four variables, three are for the position of the UAV, and one for the orientation of one body-fixed axis. This applies to this study, by controlling the position of the hunter in the x-axis, z-axis, and the rotation in the yaw axis, as to always keep the invader UAV in its field of view.

After the extraction of the best track, the hunter would know the position of the invader drone $x_E = [x, y, z]_C^T$ in its reference frame, C. The best location of the intruder to be captured is defined as x_E^d. The error in the position controller of the hunter is given by equation 3.6.

$$e_X = x_E^d - x_E \tag{3.6}$$

3.5 Net launcher

The net launcher system will be the one in [2]. The system has at least one net launcher being disposed at the base of the Hunter UAV, operable to launch a net toward a target object. A controller responsive to a signal indicative of a detected target object, will trigger the net launch toward the target object (Figure 3.8).

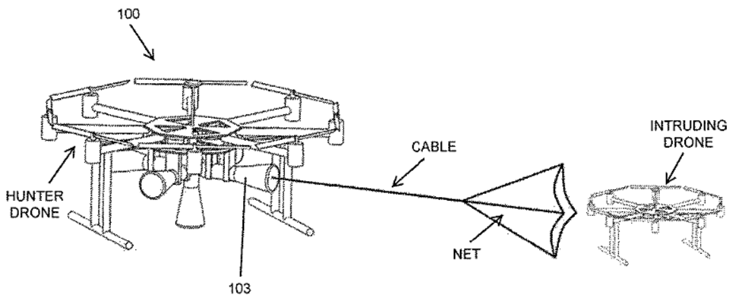
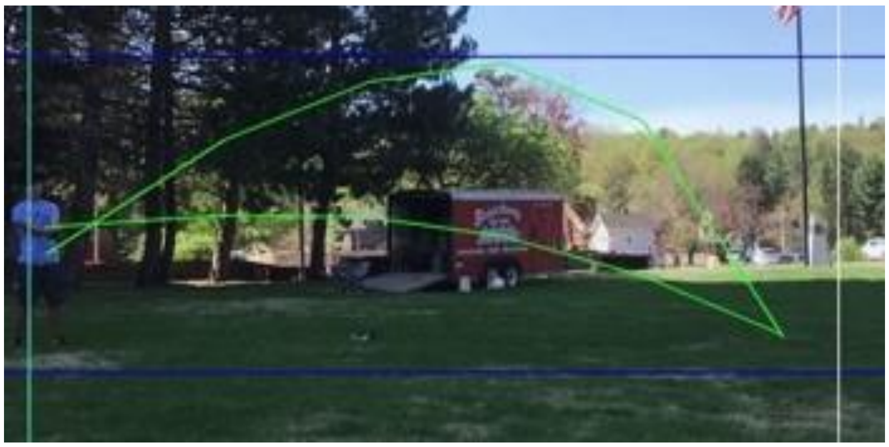


Figure 3.8. Net launcher system, as designed by [2].

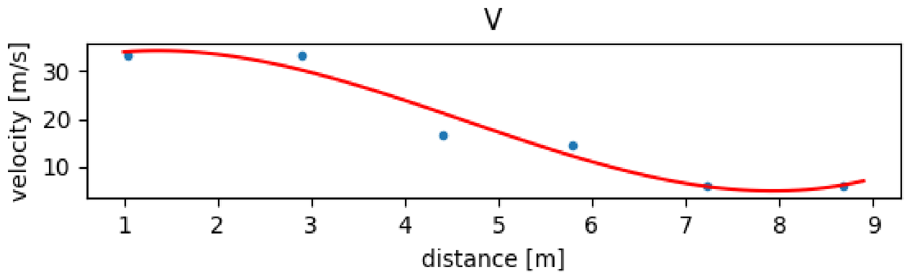
From preliminary experiments (Figure 3.9) it was found that the net launcher shoots the net with a parabolic trajectory, initially expanding in size, then retracting. The net reaches a maximum expanded area when its center is located approximately 2 meters in front and 1 meter below the net launcher (equation 3.7), in the *C* frame.

$$x_E^d = [0, 1, 2]^T \tag{3.7}$$

Thus, the hunter UAV should control its attitude to place the moving intruder UAV at the desired x_E^d location.



a)



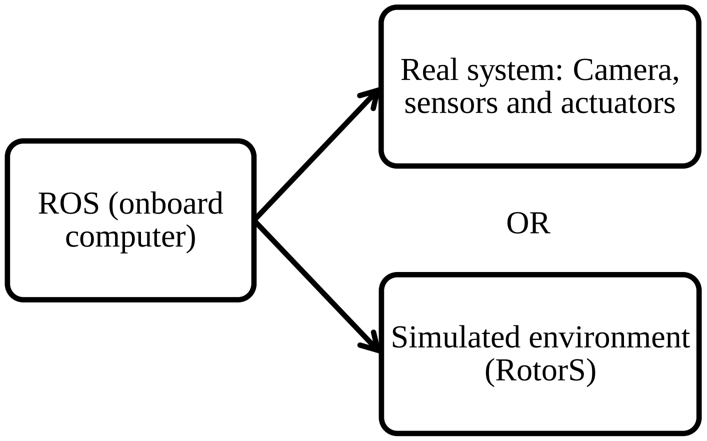
b)

Figure 3.9. Experiment to determine the trajectory and speed of the net when launched by the net launcher system.

4. IMPLEMENTATION

The system is designed using ROS nodes for different processes. Both in simulation and a physical scenario, the UAV can be controlled using ROS. It serves as an interface for the robot and the sensor and actuator systems. For the simulation environment, the algorithm will subscribe to the simulated sensor data.

ROS allows the functionality of a robot to be divided into subsets of tasks. In the further sections of this chapter, it will be presented how different parts of the system are instantiated as nodes. For example, the conversion from 2D images to point cloud data is done by a node, the detection of the intruder is done by a different node, and the control of the hunter UAV will be planned by another node.


Real system: Camera, sensors and actuators
ROS (onboard computer)
OR
Simulated environment (RotorS)

4.1 Simulation environment

To test the proposed methods, a simulated environment using the Gazebo simulator is launched. Gazebo allows the test of algorithms using simulated robots, and to train AI systems using realistic scenarios.

Along with Gazebo, the RotorS simulator [35] is used. RotorS is a MAV (Micro Aerial Vehicle) Gazebo simulator. It comes with a few multi-rotor UAV models and the possibility to include simulated sensors (such as an IMU, odometry, and vision) mounted on them. RotorS also contains a few examples of controllers.

A realistic scenario (also referred to as World) for the simulation is created. A few houses, an uneven ground plane, and roads are placed. The hunter and intruder UAV are launched. The hunter UAV is a hexacopter (Firefly model in RotorS), which is similar to the one used for tests later on in the real scenario (Figure 4.1). On the hunter, the simulated stereo camera is mounted, with the parameters matching the ones from the selected camera (see Table 3.1). The intruder drone/s can be any model of UAV. A small quadrotor is chosen to be the invader UAV to test the algorithms (Hummingbird model in RotorS).

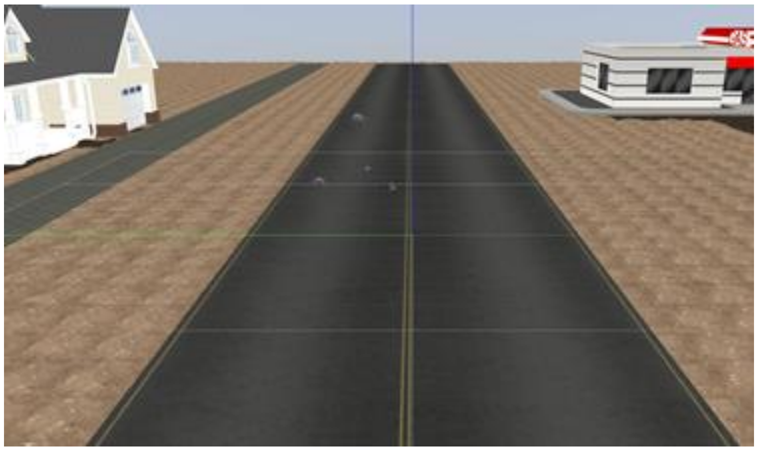


Figure 4.1 Simulated scenario for testing. Gazebo simulator.

Rviz is another useful tool for the development of the system. It is a 3D visualization tool for ROS applications. There the robot model can be seen, and sensor information analyzed. It also allows the replay of recorded data.

For the simulation stage, a laptop computer running ROS Kinetic Kame, supported on Ubuntu 16.04 Xenial was used. The computer has a processor Intel Core i7-9750, video card NVIDIA Quadro T1000 with 4GB, and 16Gb of RAM.

4.1.1 Evaluation of the UAV targeting in a simulated environment

As mentioned in section 3.1.2, the images coming from the simulated stereo camera were sampled at 30 FPS, with a resolution of 640x480 pixels. Figure 4.2 illustrates the image processing pipeline. The stereo images obtained from the simulated camera are converted to point cloud data (Figure 4.2 b). To reduce the computational time, the point cloud was downsampled into a voxel grid (Figure 4.2 c) and filtered to the working environment values (between 0.5m and 10m).

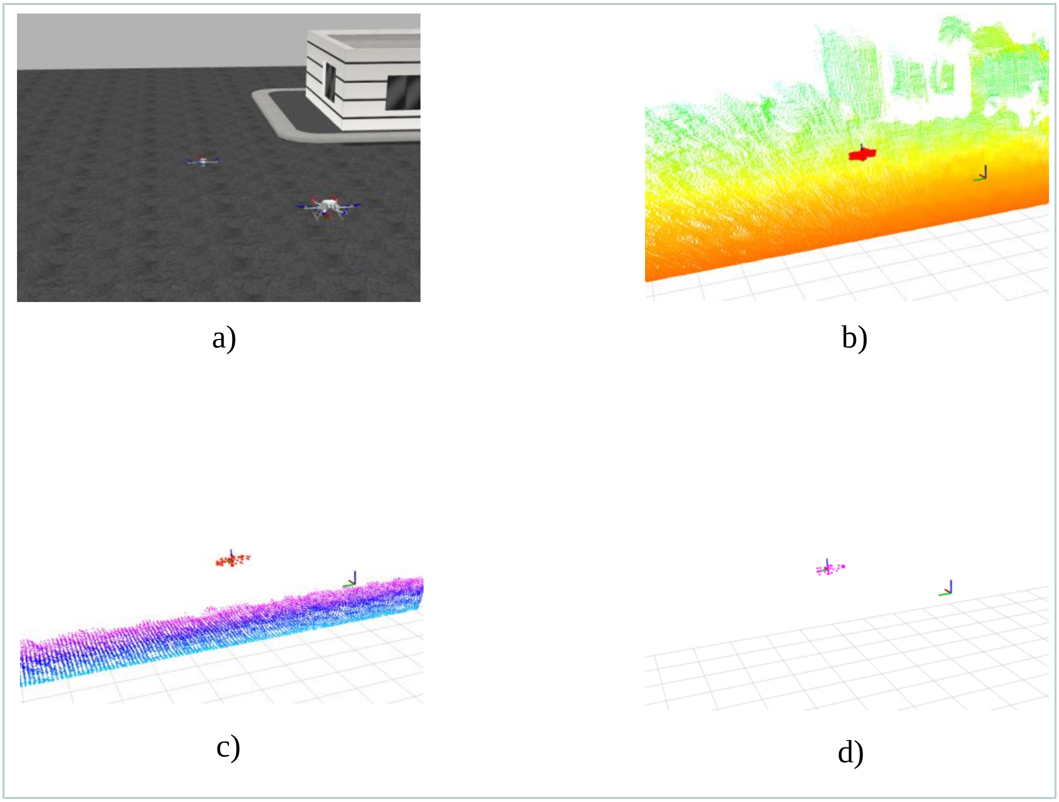


Figure 4.2. Image processing pipeline. Grid 1x1 m. (a) Simulated environment, a hunter UAV, and an intruder UAV. (b) Point cloud data obtained from the camera mounted on the hunter. (c) Filtered point cloud down-sampled into a voxel grid. (d) Cluster extraction.

A Euclidean clustering algorithm (Figure 4.2 d), identifies clusters of points after the point cloud is filtered. Clusters that are too large and too small possibly represent environment landscape (e.g., ground, walls) or stereo camera noise. Those clusters are removed. The detections are selected as the centroid of each remaining cluster.

Error Analysis of the Detection

In the RotorS simulator, using a simulated stereo camera attached to the hunter UAV, an intruder UAV is placed for testing. The simulated environment provides ground truth information about both UAVs' position, which is used to compare with the results from the visual detection to obtain the performance of the method.

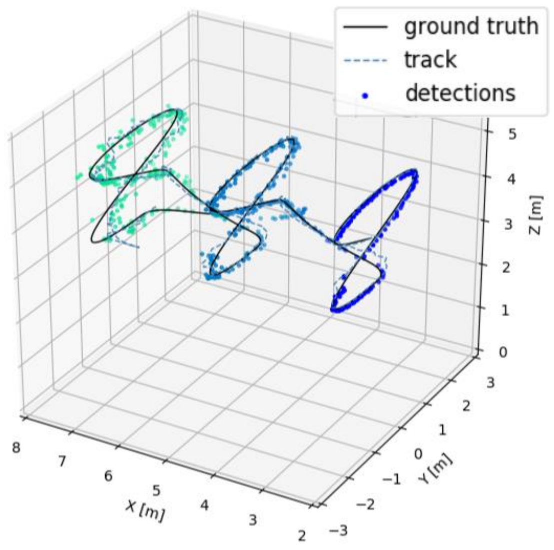


Figure 4.3. Trajectory of the intruder UAV for the simulated experiment. This trajectory is used to evaluate the detection and tracking accuracy.

The experiment was designed as follows: the stereo camera mounted on the hunter UAV sees an intruder UAV following a figure-8 trajectory in three different planes normal to the camera z-axis, as seen in Figure 4.3. The Root Mean Square Error (RMSE) analysis in detecting the intruder's centroid is shown in Table 4.1. This error analysis is comparing the ground truth against the detections (not to the tracking predictions).

Table 4.1. Spatial Root Mean Square Error (RMSE) in meters.

Distance [m]	RMSE			Precision	Recall
	X	Y	Z		
2.5	0.063	0.112	0.032	1.000	0.974
5.0	0.073	0.108	0.035	1.000	0.983
7.5	0.174	0.083	0.032	0.998	0.993

Execution time of detection

It is important to account for the delay in the detection of the algorithm. A detection lag of $\Delta t_{Det} = 0.15\ s$ was calculated as the maximum cross-correlation coefficient for the ground truth trajectory and detected trajectory (Figure 4.4). Similarly, the tracking lag for the tracking trajectory, was $\Delta t_{Track} = 0.15\ s$. This means that the computational time of the detection is the limiting part of the algorithm.

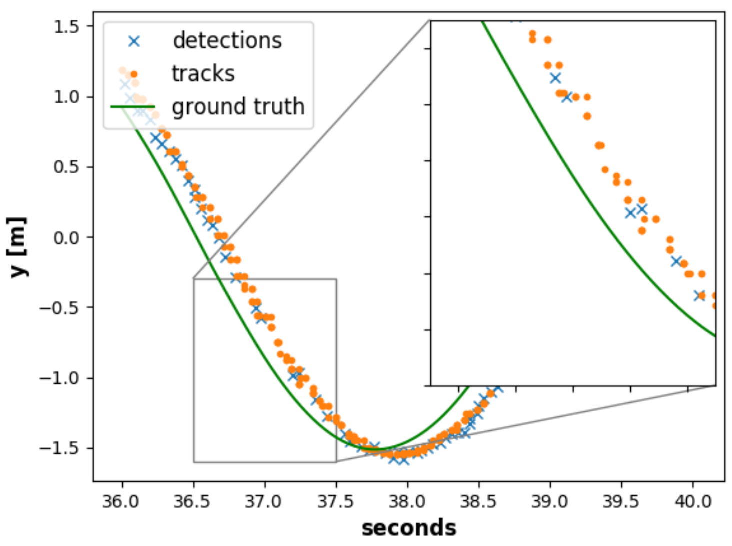


Figure 4.4. Detection lag for the simulated experiment. Comparison of part of the ground truth, detected and tracked trajectory of the intruder in the y axis.

Tracking Accuracy of the Intruder UAV

In Figure 4.5, the MOTA and RMSE were evaluated for several *r* parameters, ranging from 0.5 to 5.0. The *r* parameter scales the measurement noise covariance matrix *R* of the Kalman filter supporting the tracking algorithm. For all tested values of *r*, the MOTA index was above 0.999 and the RMSE had a minimum value of 0.09 m at *r* = 1.0. Note that for very small values of *r* the tracks rely more on the noisy measurement (detection) data, producing a noisy track. While for very large values of *r*, the tracking algorithm trusts the motion model too much, and does not update the acceleration state on time.

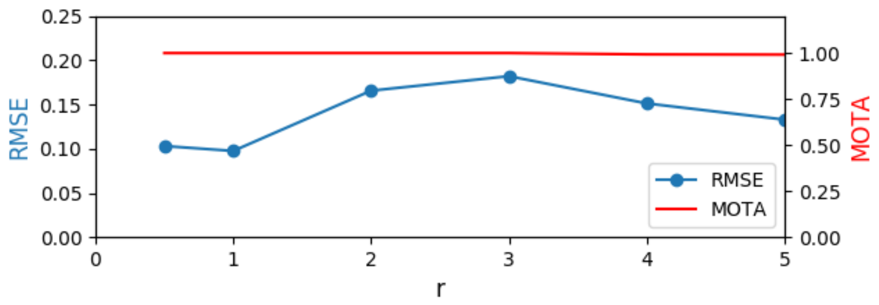


Figure 4.5. In red, MOTA index for the analysis of tracking accuracy. In blue, the RMSE of the best track when compared to the ground truth data, for different values of *r*, scaling factor in the Kalman filter R matrix.

An advantage of the Kalman filter based algorithm is shown above in Figure 4.4, where the tracking procedure correctly predicts the position of the intruder UAV even when there are temporarily missing detections (between 37.0 and 37.5 seconds).

Aiming Control

The motion control for the simulated hunter was achieved sending desired pose commands to a position controller. The sequence of pose commands set the trajectory of the hunter UAV that will reduce the position error in Equation (3.5).

4.1.2 Discussion on the simulated experiment

Using point cloud data from a stereo camera as the rubric for UAV recognition appears to be a feasible solution. A good recall index is needed to guarantee the detection of all invader UAVs. Table 4.1 shows a reliable recall index, and a precision that stays high as recall increases. For object detection, it is desirable to have good precision as recall increases.

The tracking algorithm based on a Kalman filter has the smoothing and predictive characteristics required for a steady aiming, as shown in Figure 4.4. The detection and tracking lags were similar, suggesting that the detection step has a longer execution time than the tracking step. This is expected, considering that the detection step performs operations in thousands of points contained in the point cloud. In contrast, the tracking step performs operations in a few 9x9 matrices (the Kalman filter state has nine variables).

A value of MOTA above 0.999 means the tracking algorithm always assigned the intruder UAV to the same track, even during missing detections. This is important for target aiming applications to avoid switching between targets. Additionally, the tracking error of 0.09 m RMSE is considerably smaller than the area of the net, which spans above 1 meter in diameter. Therefore, this tracking error is small enough for a successful intruder UAV capture.

The simple control of the simulated hunter UAV using its visual feed for position control performed as expected, keeping the intruder drone at the desired distance from the hunter and staying centered in the camera's field of view.

The next section will focus on adapting and testing the algorithm on a real-world scenario, performing the detection and tracking using a feed from the WithRobot stereo camera, and implementing a motion controller to send the position to a flight controller and thus aim at the invader UAV.

4.2 Experimental platform

In this section, the overview of the proposed hunter testing platform is introduced along with the hardware and software components. The proposed hunter drone can be analyzed by parts: aerial vehicle frame, flight control, onboard computer, and visual input device (Figure 4.6).

A hexacopter UAV is chosen as the test platform. The DJI Flamewheel F550 (Figure 4.6 a) is built with light and strong compounds to grant better crashworthiness. It comes with a high strength compound PCB frame board, easy to wire, and optimized with assemble space for autopilot systems. The specifications for the frame are found in Table 4.2. For this frame, six E300 ESCs (Electronic Speed Controller) of 15 A control the six E300 motors. The specifications about the propulsion system can be found in Table 4.3.

Table 4.2. DJI Flamewheel frame specifications.

DJI Flamewheel F550	Specifications
Frame Weight	478 g.
Diagonal Length	550 mm.
Takeoff Weight	1200g ~ 2400g.

Table 4.3. DJI propulsion system for the F550 frame.

DJI propulsion system E300	Specifications
Recommended Payload	300 g/axis
Maximum Thrust	600 g/axis
Recommended Battery	3S - 4S LiPo
KV	920 rpm/V

The flight controller used for the hunter drone is the Pixhawk PX4, manufactured by Radiolink (Figure 4.6 b). The PX4 is an open-hardware project by Pixhawk which features leading-edge processor and sensor technology for good performance, flexibility, and reliability for the control and development of autonomous vehicles. The details about this flight controller are in Table 4.4.

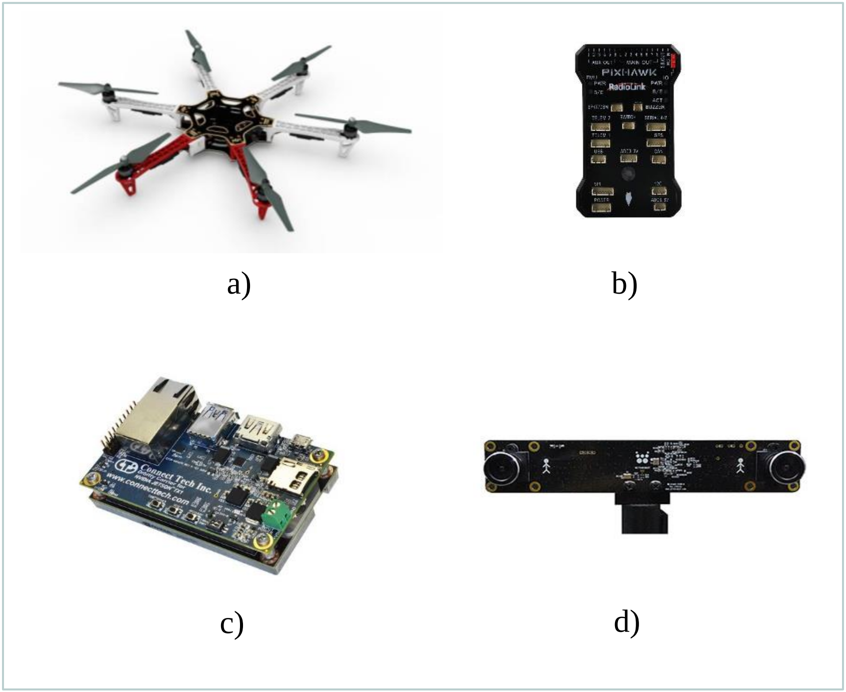


Figure 4.6. a) the DJI Flamewheel F550, selected frame for the experimental hunter platform, image from https://www.dji.com/flame-wheel-arf/feature; b) Radiolink PixHawk PX4, open source flight controller, image from https://spexdrone.com/products/pixhawk-classic-radiolink-pixhawk-px4-autopilot-pixhawk-px4; c) Selected onboard computer, Orbitty Carrier for NVIDIA Jetson TX1, image from http://connecttech.com/product/orbitty-carrier-for-nvidia-jetson-tx2-tx1/; d) Stereo camera, WithRobot oCamS-1CGN-U, image from http://withrobot.com/en/camera/ocams-1cgn-u/.

A remote controller, FrSky Taranis X9D is connected to the flight controller for manual operation of the hunter UAV and override of the autonomous mode if necessary. The Taranis X9D has 16 channels available, it was configured using QGroundControl to use the first 8 channels as follows: the first four channels control the roll, pitch, throttle and yaw; channel 5 selects the flight mode and channel 8 arms or disarms the hexacopter.

Table 4.4. Specifications of the fight controller.

Radiolink Pixhawk PX4	Specifications
Processor	32bit 32F427 ARM Cortex M4 Core 168 MHZ / 256 KB RAM / 2 MB Flash
Gyroscope	ST Micro L3GD20H 16 bit
Accelerometer/magnetometer	ST Micro LSM303D 14 bit
Accelerometer/gyroscope	MPU 6000 3-axis
Barometer	MEAS MS5611
GPS	M8N GPS Module

The flight controller needs feedback on the vehicle position to perform autonomous missions. The Pixhawk PX4 comes with a GPS module (Figure 4.7 a). For indoor tests where the GPS signal is not available, an option to obtain odometry estimation consists of an Optical Flow setup, which requires a downward-facing camera and a distance sensor (preferably a LiDAR). For the indoor tests, a PX4FLOW optical flow smart camera as presented in [36] is used in conjunction with a Garmin Lidar Lite V3 distance measurement sensor (Figure 4.7 b).

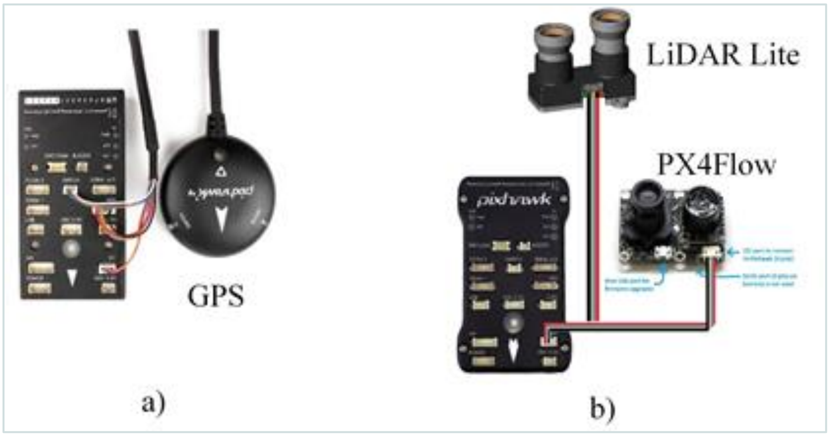


Figure 4.7. Pixhawk PX4 requires a position estimation system. a) GPS module, image from https://www.getfpv.com/holybro-px4-2-4-6-pixhawk-m8n-gps-pm-100mw-radio-telemetry-v3-915mhz.html, b) PX4Flow and LiDAR Lite, image from https://ardupilot.org/copter/docs/common-px4flow-overview.html. Both optical flow camera and distance sensor are connected to the I2C bus of the flight controller.

For the Onboard computer, the Orbitty Carrier for NVIDIA Jetson TX1 was chosen (Figure 4.6 c). This Orbitty Carrier board incorporates connectivity and multimedia interfaces, a few of them being USB, Ethernet, and HDMI for the computer module Jetson TX1. This module is ideal for robotics and unmanned applications, being low cost and granting the performance required while keeping a low power consumption. Jetson TX1 fits the requirements for visual computing applications.

Table 4.5. Specifications of the onboard computer.

NVIDIA Jetson TX1	Specifications
Size	87mm x 50mm
GPU	256-core NVIDIA Maxwell
CPU	Quad-Core ARM Cortex - A57 MP Core
Memory	4GB 64-bit
Storage	16GB
Power	Under 10W
PCIE	Gen 2
Wi-Fi	Yes

The input voltage required by the module is in the range 9V~14V, which makes it feasible to power it by a LiPo battery once mounted on the hunter drone.

Data flow diagram

The information flow is as seen in Figure 4.8. Connectivity-wise, the stereo camera is attached to the onboard computer via a USB port; the onboard computer communicates to the flight controller using serial communication; and the flight controller sends the commands to the motors of the hunter using PWM signals.

Software architecture

ROS nodes command the execution of the system. A diagram is shown below in Figure 4.9. A similar ROS-based system proposed for the simulated environment processes the images obtaining the point cloud data. This time, the images are processed by ROS nodes on the companion computer. The detections and tracks together with a position reference given by either visual odometry or a GPS module, feed an algorithm in ROS that will plan the motion of the hunter. This plan is then communicated to the flight controller through MAVROS.

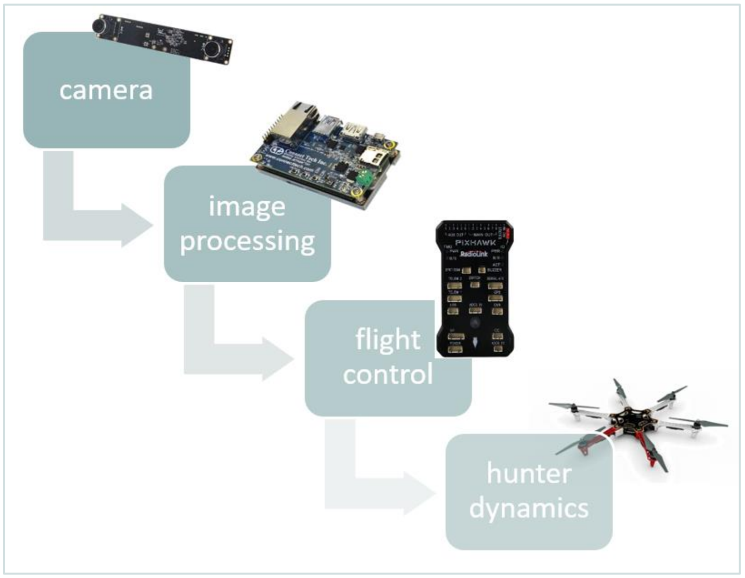


Figure 4.8. Information flow diagram for the experimental platform.

MAVROS is a package of communication tools used between systems running ROS and various autopilots with MAVLink protocol. A joystick or remote controller is also linked to the flight controller to provide a manual mode.

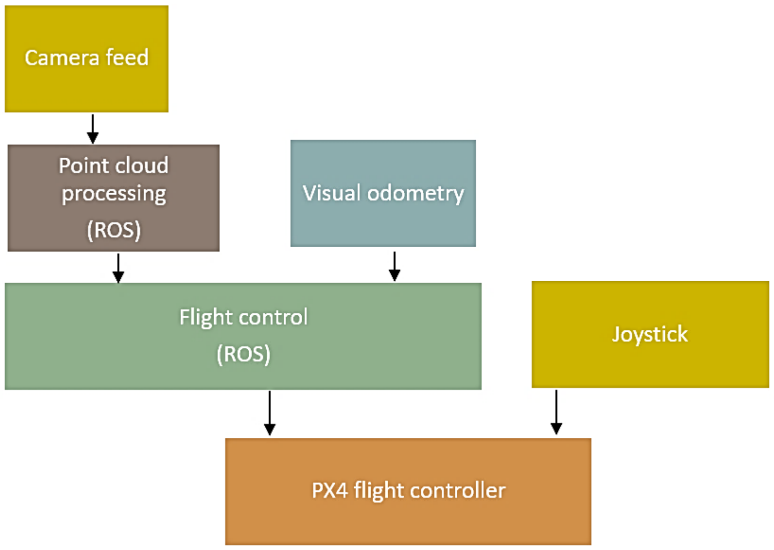


Figure 4.9. Software architecture for the detection, tracking and motion planning.

The flight controller Pixhawk PX4 is configured using QGroundControl. This is open source and provides the support for the PX4 autopilot stack. QGroundControl was used to configure and tune the parameters for the hunter hexacopter, and the desired flight modes and sensors.

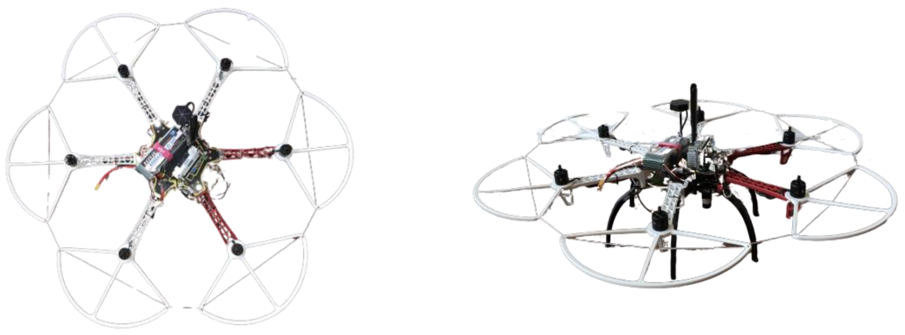


Figure 4.10. Experimental platform using the DJI F550 hexacopter frame and the components mentioned in Figure 4.6 and Figure 4.7.

Flowchart of the experimental platform

The final experimental platform is shown in Figure 4.10, and the principal components and the flow of information is represented in Figure 4.11.

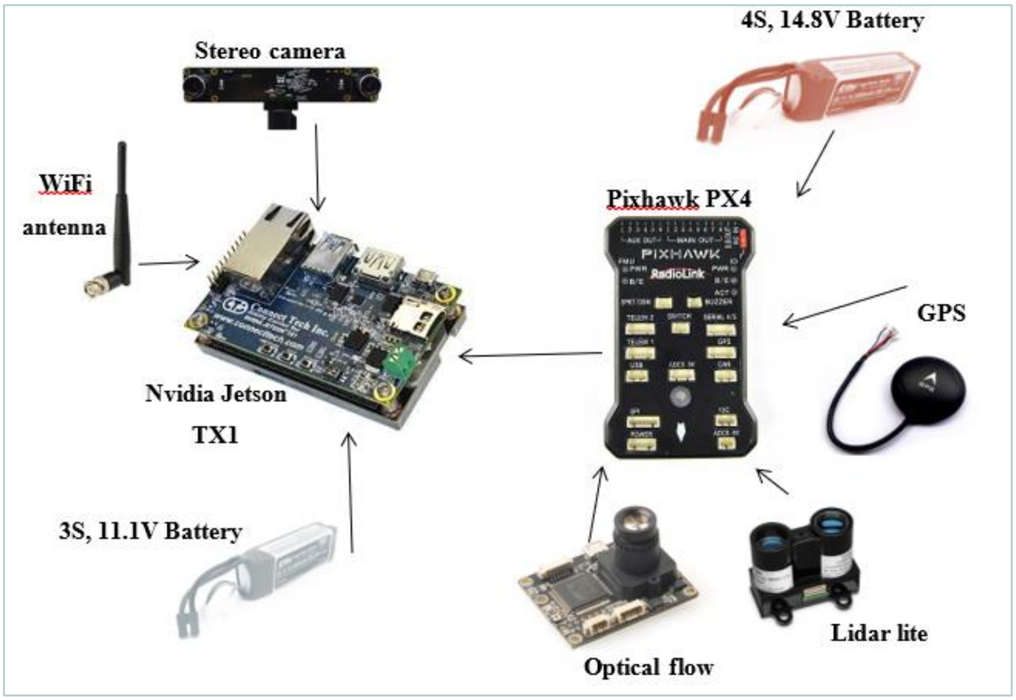


Figure 4.11. Components of the experimental platform and data flow indication. Images obtained from the previously mentioned sources in Figure 4.6 and Figure 4.7.

4.2.1 Evaluation of the UAV tracking in the real scenario

The detection and tracking algorithms were tested in an outdoor scenario. A static camera was set for the experiment, and an intruder UAV (manually controlled) was in the camera field of view.

When testing outdoors, the exposition parameter of the camera needed adjustment. A mechanism for automatic exposure adjustment will be useful for future tests. In Figure 4.12, the obtained image processing pipeline is shown. As a difference to the simulated environment, the Euclidean cluster extraction needed some parameters to be tuned. The cluster tolerance was modified to 0.5m (minimum distance between different clusters), and the smallest and largest cluster size admitted were set to 60 and 500 respectively. These parameters were tuned after the experiment, using the data stored in a bag file. A bag file subscribes to one or more ROS topics and stores the serialized message data in a file as it is received. The extraction of the clusters was successful in this scenario, as can be seen in Figure 4.12 d).

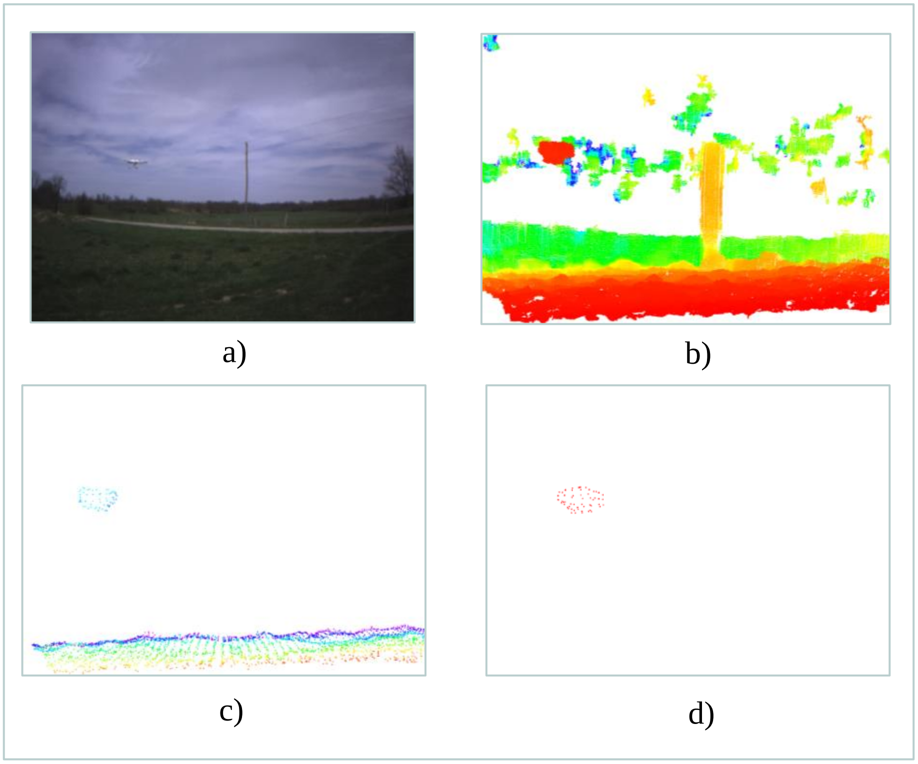


Figure 4.12. Image processing pipeline in the real scenario. (a) Outdoor testing environment with an intruder UAV, image feed from the oCamS-1CGN-U camera. (b) Point cloud data obtained. (c) Filtered point cloud down sampled into a voxel grid. (d) Euclidean cluster extraction.

The tracking algorithm, by keeping the best track and using the prediction of the speed of the target in real-time (Figure 4.13, Figure 4.14), recovered from cases when false positives were present in the images.

The onboard camera used for the experimental platform has limited processing capabilities. Using the library provided by the manufacturer of the stereo camera WithRobot oCamS-1CGN-U, the maximum speed of the image feed was of 10 Hz. After a modification of the library for this camera, with the focus of maintaining only the parameters and data that are used for this project, an input image feed of 20 Hz was obtained.

From the image feed coming at 20 Hz, the point cloud 3D reconstruction was obtained with a frequency of 15 Hz. The detection algorithm published a result with a frequency of 30 Hz, and the best track runs at 100 Hz. The system proves to be valid for use as a real-time method for the detection of unwanted UAVs.

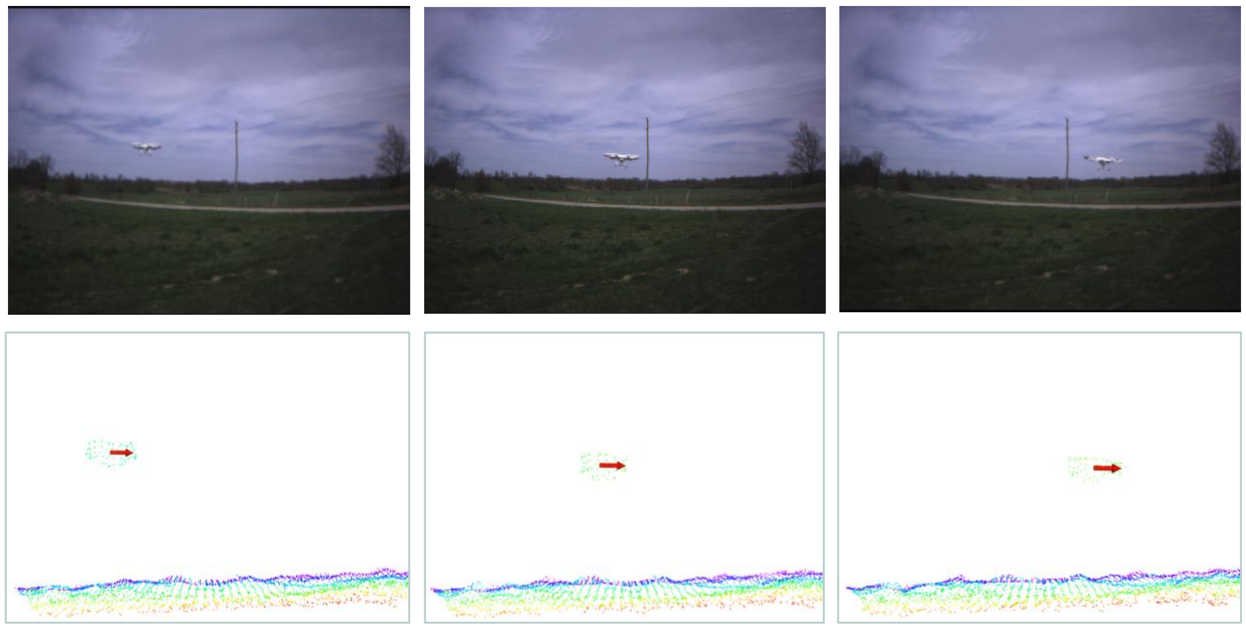


Figure 4.13. The red arrow represents the best track in every frame presented. The algorithm keeps track of the position of the intruder in successive frames. Images in a), b) and c) are taken after successive intervals of 1s.

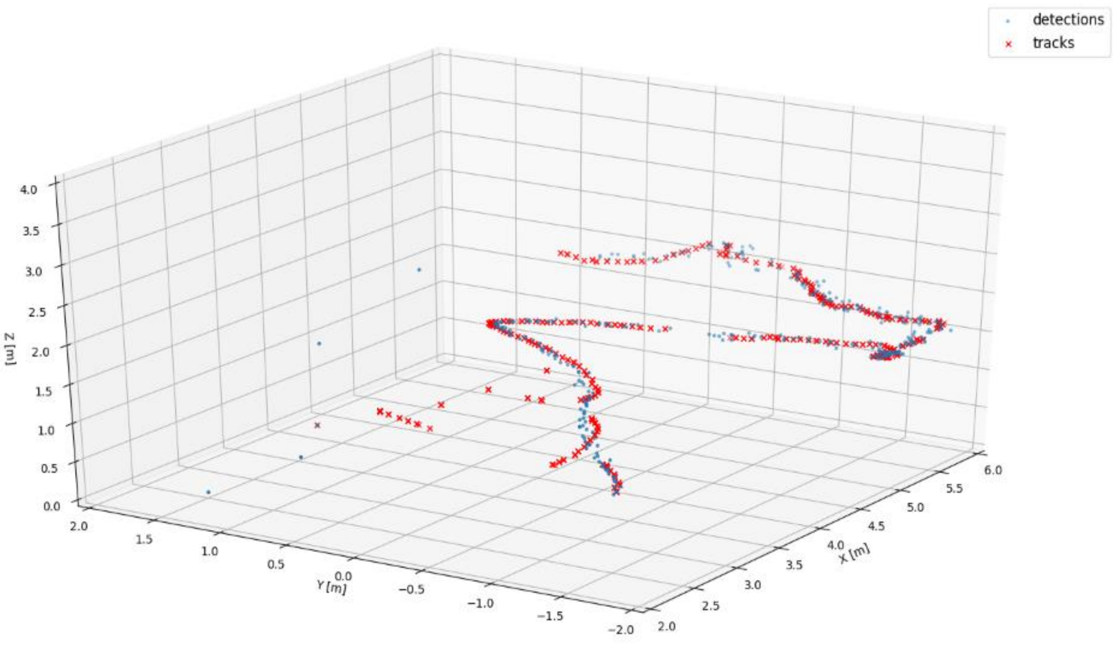


Figure 4.14. Detection and best track obtained in the outdoor environment.

For the outdoor experiment the metrics obtained were a value of 0.95 for the Recall index, a value of 0.92 for the Precision index, and a value of 0.94 for the MOTA index. The values stay relatively high and close to 1.0, the ideal case scenario. With respect to the simulated tests, these indexes have lower values due to the presence of false-positives which were mostly due to noise detected by the stereo camera.

Quantitative Evaluation

For validation of the computer vision algorithm, a motion capture system can precisely record the trajectory of an intruder UAV to then compare it to the estimates obtained by the onboard camera (Figure 4.15). The setup for this test uses an array of Qualisys motion capture cameras to determine the location of the stereo camera and the intruder drone. The onboard computer at the same time records the detections and tracks obtained for the movement of the intruder indoors.

This experiment needs to be done indoors, in a laboratory setting. From this test, several comments arise and need to be approached in the future. In an indoor environment the point cloud data reconstruction results in many clusters being recognized as positive targets. Due to objects, walls, ceiling, or the motion capture system, more clusters are detected and become part of the false-positive information (Figure 4.16). When the intruder UAV is close to the walls, floor, or ceiling, the Euclidean cluster extraction algorithm fails to detect the intruder. This happens due to the cluster tolerance required by the method. This situation makes clear that for indoor detection of UAVs, or environments with more features, a further cluster analysis is needed.

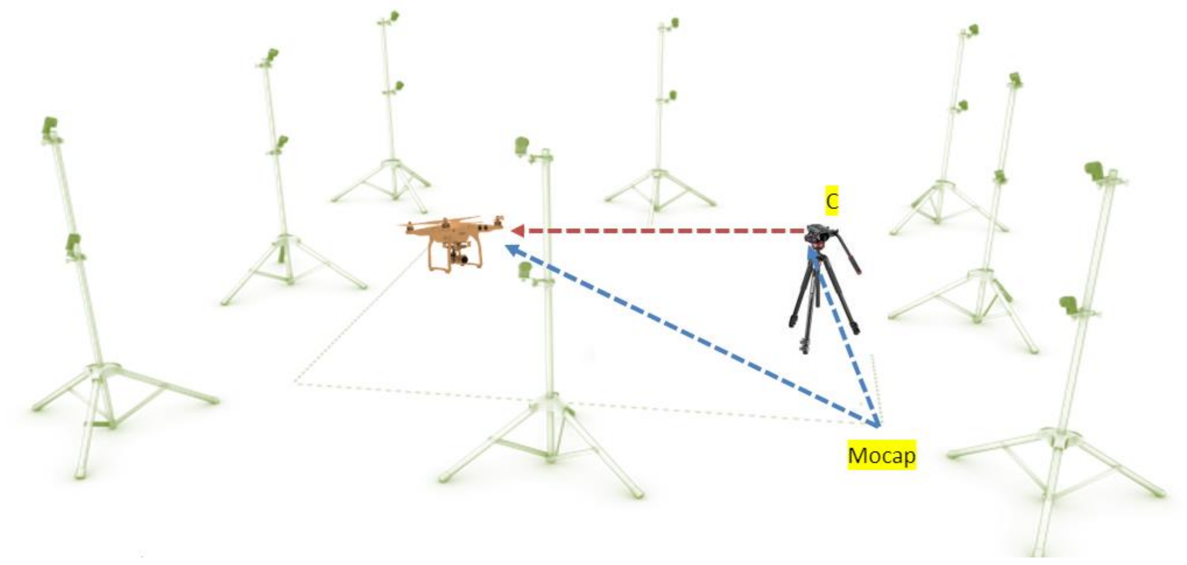
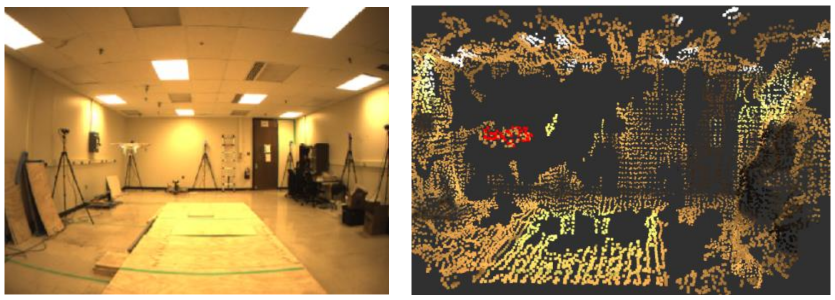


Figure 4.15. Motion capture system setup for the performance analysis.

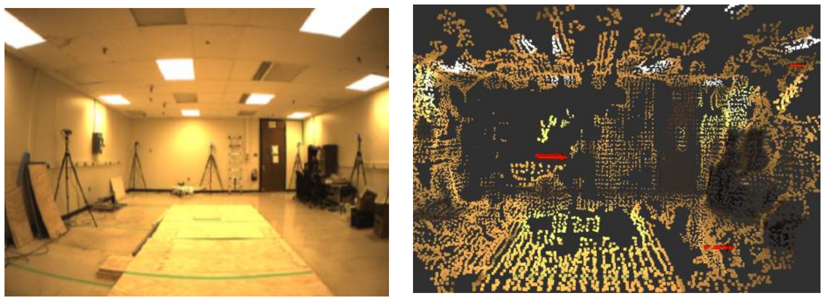
4.2.2 Discussion about the experiments

From the previous experiments, the outdoor detection and tracking performed as expected, in a similar way to the simulated tests. The hunter UAV successfully detected, tracked and acted upon the recognition of an intruder UAV.

For indoor environments, the detection and tracking relying only on stereo vision and cluster extraction is almost unfeasible. A few options to overcome this situation can be the use of an object classification method based on deep learning, or SLAM (Simultaneous Localization And Mapping) for the generation of maps of the indoor settings.



a)



b)

Figure 4.16. Indoor detection using point cloud data and the cluster extraction method. In a) only true-positive data is detected; in b) the detections include false-positive data.

From simple tests using Darknet, an open-source neural network framework, with support for CPU and GPU computation [37], and a set of training images for drones as in [38], promising results were obtained (Figure 4.17).

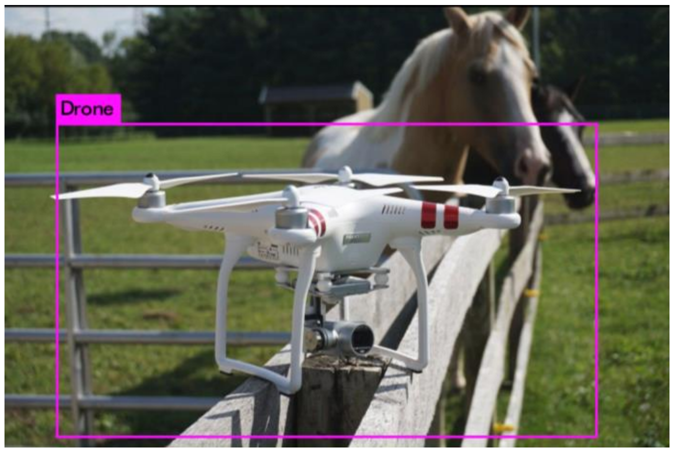


Figure 4.17. Object recognition using deep learning.

4.2.3 Motion control experiment

A simple test to check the integration of the system can be obtaining a yaw changing motion of the hunter drone, so that its heading tends to align with the position of the invader drone. The hunter UAV changes its yaw orientation, hovering at a static height, in an aim to keep the intruder drone centered in the camera's field of view.

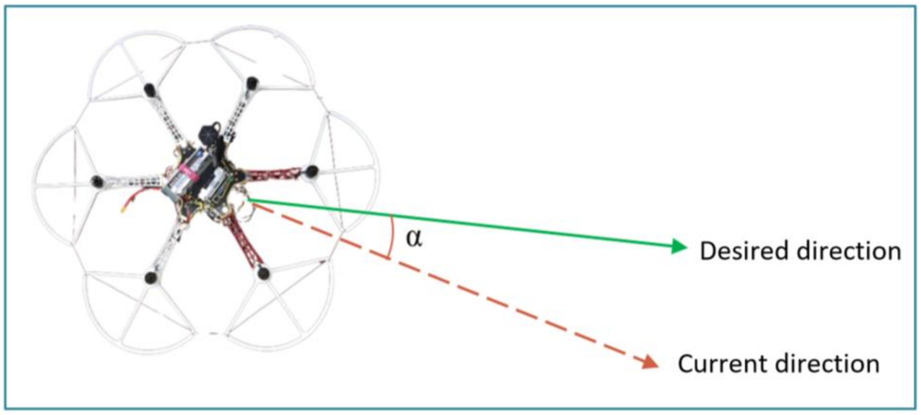


Figure 4.18. Simple yaw control algorithm for the experimental platform.

To achieve the motion in Figure 4.18, the onboard computer runs over ROS the detection and tracking algorithm, a motion control node, and a MAVROS node, which allows communication to the flight controller. A topic message controlling the desired attitude of the hunter UAV is published with a 10 Hz frequency. The flight controller interprets and actuates the motors to perform the desired motion.

5. SUMMARY AND FUTURE WORK

5.1 Summary

A drone hunting platform capable of detecting invader UAVs in an area using stereo vision was proposed in this work. The method includes an object detection step based on computer vision, using 3D point cloud information. The tracking step is based on a Kalman filter combined with the Linear Sum Assignment algorithm.

This method can be executed in real-time on an onboard computer and mounted on a commercial UAV platform. The fast execution time of the algorithm added to the predictive nature of the tracking algorithm, makes this a reliable system for the detection of intruder UAVs for use in aerial security.

The final stage of the system controls the hunter UAV with the motion dynamics necessary to capture the intruder.

5.2 Future Work

This project paves the way for further development. The tests using the stereo camera and point cloud data were successful in outdoor environments more than in indoor environments.

There is an opportunity for improvement related to processing time. Adding the functionality of the CUDA cores, already available in the selected hardware, will support parallel processing and speed up the processing considerably.

For enhance of the detection with this method in indoor environments, a layer of control of the positives targets can be added. The use of deep learning object recognition or SLAM algorithms can help detect false positives and give the precision for the detection needed for a surveillance system.

Related to the performance of the camera-based method, details that can be improved in future work include using a source of infrared light for low lit environments and the use of an automatic exposure corrector for different lighting conditions.

Concerning the motion dynamics required for the hunter drone, the addition of a gimbal for the camera can overcome this limitation. Another gimbal can be used to mount the net launcher cannon as well.

The adoption of a larger platform for the hunter, able to carry the weight of the net launcher and a captured drone is necessary for a closed-loop test of the complete system. For this test, the motion algorithm should compensate for the detection and tracking lag, the ballistic dynamics of the net, and anticipate to the predicted trajectory of the intruder.

REFERENCES

[1] M. Ritchie, F. Fioranelli, H. Griffith, and B. Torvik, “Micro-drone RCS analysis,” Oct. 2015.

[2] M. R. Aagaah, E. M. Ficanha, and N. Mahmoudian, “Drone having drone-catching feature,” US10005556B2, Jun. 26, 2018.

[3] “10 Drone Stats That Will Blow You Away.” https://www.fool.com/investing/2018/01/19/10-drone-stats-that-will-blow-you-away.aspx (accessed Mar. 25, 2020).

[4] “The Explosive-Carrying Drones in Venezuela Won’t Be the Last,” *Wired*.

[5] “25 Interesting Drone Facts and Statistics (2020) | By the Numbers.” https://expandedramblings.com/index.php/drone-statistics/ (accessed Mar. 25, 2020).

[6] “Peeping drones could be spying on you in your own home,” *TODAY.com*. https://www.today.com/money/peeping-drones-could-be-spying-you-your-own-home-t128590 (accessed Mar. 09, 2020).

[7] C. Schüpbach, C. Patry, F. Maasdorp, U. Böniger, and P. Wellig, “Micro-UAV detection using DAB-based passive radar,” in *2017 IEEE Radar Conference (RadarConf),* May 2017, pp. 1037–1040.

[8] E. E. Case, A. M. Zelnio, and B. D. Rigling, “Low-Cost Acoustic Array for Small UAV Detection and Tracking,” in *2008 IEEE National Aerospace and Electronics Conference*, Jul. 2008, pp. 110–113.

[9] B. Adler, J. Xiao, and J. Zhang, “Autonomous Exploration of Urban Environments using Unmanned Aerial Vehicles,” *J. Field Robot.*, vol. 31, no. 6, pp. 912–939, 2014.

[10] M. Hammer, M. Hebel, B. Borgmann, M. Laurenzis, and M. Arens, “Potential of lidar sensors for the detection of UAVs,” in *Laser Radar Technology and Applications XXIII,* May 2018, vol. 10636, p. 1063605.

[11] R. Medhi, “A Review on Object Detection and Tracking Methods,” Accessed: Jan. 30, 2020. [Online]. Available: https://www.academia.edu/33497279/A_Review_on_Object_Detection_and_Tracking_Methods.

[12] “[PDF] Moving Object tracking Based on Background Subtraction Combined Temporal Difference | Semantic Scholar.” https://www.semanticscholar.org/paper/Moving-Object-tracking-Based-on-Background-Combined-Chen-Wang/9a39050f72e91c7e3351d86c91cd0bcc14f2dad9 (accessed Mar. 25, 2020).

[13] S. Krukowski and A. Perkins, “Tracking of Small Unmanned Aerial Vehicles,” p. 6.

[14] R. Szeliski, “Computer Vision: Algorithms and Applications,” p. 979.

[15] D. M. Kim, H. J. Choi, J. S. Kim, W. K. Lee, D. H. Song, and S. Jung, “Tracking Control of a Moving Object for Robokers with Stereo Visual Feedback,” in *2007 IEEE International Conference on Integration Technology,* Mar. 2007, pp. 52–57.

[16] “3D depth image analysis for indoor fall detection of elderly people | Elsevier Enhanced Reader.” (accessed Jan. 15, 2020).

[17] N. D. Kehtarnavaz and W. Sohn, “Analysis of camera movements in stereo vision-based vehicle tracking,” in *Proceedings of 1st International Conference on Image Processing,* Nov. 1994, vol. 2, pp. 710–714 vol.2.

[18] “Stereo Vision Basics.” http://chriswalkertechblog.blogspot.com/2014/03/stereo-vision-basics.html (accessed Apr. 01, 2020).

[19] Y. Xiang, A. Alahi, and S. Savarese, “Learning to Track: Online Multi-object Tracking by Decision Making,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, Dec. 2015, pp. 4705–4713.

[20] B. Sahbani and W. Adiprawita, “Kalman filter and Iterative-Hungarian Algorithm implementation for low complexity point tracking as part of fast multiple object tracking system,” in *2016 6th International Conference on System Engineering and Technology (ICSET)*, Oct. 2016, pp. 109–115.

[21] Y. Ma, J. Anderson, S. Crouch, and J. Shan, “Moving Object Detection and Tracking with Doppler LiDAR,” *Remote Sens.*, vol. 11, no. 10, p. 1154, Jan. 2019.

[22] P. Fang, J. Lu, Y. Tian, and Z. Miao, “An Improved Object Tracking Method in UAV Videos,” *Procedia Eng.*, vol. 15, pp. 634–638, Jan. 2011.

[23] F. Luetteke, X. Zhang, and J. Franke, “Implementation of the Hungarian Method for object tracking on a camera monitored transportation system,” in *ROBOTIK 2012; 7th German Conference on Robotics*, May 2012, pp. 1–6.

[24] J. Pappalardo, “The Air Force Is Deploying Its First Drone-Killing Microwave Weapon,” *Popular Mechanics*, Sep. 24, 2019. https://www.popularmechanics.com/military/weapons/a29198555/phaser-weapon-air-force/ (accessed Mar. 09, 2020).

[25] “Watch Japan’s bizarre net-wielding police drone in action,” *Digital Trends*, Dec. 14, 2015. https://www.digitaltrends.com/cool-tech/watch-japans-bizarre-net-wielding-police-drone-in-action/ (accessed Jan. 29, 2020).

[26] “Books/ROS_Robot_Programming_English - ROS Wiki.” http://wiki.ros.org/Books/ROS_Robot_Programming_English (accessed Mar. 12, 2020).

[27] X. Yun, “Utilizing robot operating system (ROS) in robot vision and control,” 2015.

[28] “Morphological Transformations — OpenCV-Python Tutorials 1 documentation.” https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html (accessed Jan. 30, 2020).

[29] R. B. Rusu, “Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments,” *KI - Künstl. Intell.*, vol. 24, no. 4, pp. 345–348, Nov. 2010.

[30] J. Davis and M. Goadrich, “The relationship between Precision-Recall and ROC curves,” in *Proceedings of the 23rd international conference on Machine learning*, Pittsburgh, Pennsylvania, USA, Jun. 2006, pp. 233–240.

[31] H. W. Kuhn, “The Hungarian method for the assignment problem,” *Nav. Res. Logist. Q.*, vol. 2, no. 1–2, pp. 83–97, Mar. 1955.

[32] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, “MOT16: A Benchmark for Multi-Object Tracking,” *ArXiv160300831 Cs*, May 2016, Accessed: Feb. 17, 2020. [Online]. Available: http://arxiv.org/abs/1603.00831.

[33] T. Lee, M. Leok, and N. H. McClamroch, “Geometric tracking control of a quadrotor UAV on SE(3),” in *49th IEEE Conference on Decision and Control (CDC)*, Dec. 2010, pp. 5420–5425.

[34] T. Elmokadem, “Distributed Coverage Control of Quadrotor Multi-UAV Systems for Precision Agriculture$**$This work was supported by the Australian Research Council. Also, this work received funding from the Australian Government, via grant AUSMURIB000001 associated with ONR MURI grant N00014-19-1-2571.,” *IFAC-Pap.*, vol. 52, no. 30, pp. 251–256, 2019.

[35] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, “RotorS—A Modular Gazebo MAV Simulator Framework,” in *Robot Operating System (ROS): The Complete Reference (Volume 1)*, A. Koubaa, Ed. Cham: Springer International Publishing, 2016, pp. 595–625.
[36] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, “An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications,” in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 1736–1741.
[37] J. Redmon, *pjreddie/darknet*. 2020.
[38] Chuan-en Lin, *chuanenlin/drone-net*. 2020.

VITA

Maria Nieves Brunet Avalos received her Engineering Degree in Electronics from Universidad Nacional de Rosario in 2015. She is currently a Master of Science student in the School of Electrical and Computer Engineering at Purdue University. She was awarded the Fulbright Scholarship to pursue her graduate studies. Her interests include automatic control systems, computer vision, embedded systems and robotics.