



AMERICAN UNIVERSITY OF BEIRUT

DETECTION AND LOCALIZATION OF  
PROCESSIONARY MOTH NESTS IN PINE  
TREES USING MULTI-STREAM  
CONVOLUTIONAL NEURAL NETWORKS

by

NAEL AKRAM JABER

A thesis  
submitted in partial fulfillment of the requirements  
for the degree of Master of Engineering  
to the Department of Mechanical Engineering  
of the Maroun Semaan Faculty of Engineering and Architecture  
at the American University of Beirut

Beirut, Lebanon  
January 2021

AMERICAN UNIVERSITY OF BEIRUT

DETECTION AND LOCALIZATION OF  
PROCESSIONARY MOTH NESTS IN PINE  
TREES USING MULTI-STREAM  
CONVOLUTIONAL NEURAL NETWORKS

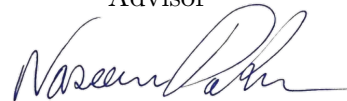
by  
NAEL AKRAM JABER

Approved by:

---

Dr. Naseem Daher, Assistant Professor  
Electrical and Computer Engineering

Advisor



---

Dr. Daniel Asmar, Associate Professor  
Mechanical Engineering

Co-Advisor



---

Dr. Elie Shammas, Associate Professor  
Mechanical Engineering

Member of Committee



Date of thesis defense: January 25, 2021

# AMERICAN UNIVERSITY OF BEIRUT

## THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: JABER NAEL AKRAM  
Last First Middle

Master's Thesis       Master's Project       Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One \_\_\_ year from the date of submission of my thesis, dissertation or project.**  
**Two \_\_\_ years from the date of submission of my thesis , dissertation or project.**  
**Three \_\_\_ years from the date of submission of my thesis , dissertation or project.**

  
Signature

February 8, 2021

Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

# Acknowledgements

I would first like to express my deep and sincere appreciation to my thesis advisors, Dr. Naseem Daher and Dr. Daniel Asmar, for giving me the opportunity to do research, believing in me, and providing invaluable guidance throughout this research. Their vision, sincerity and motivation have deeply inspired me. They have taught me the methodology to carry out research, challenge myself and push my limits. It was a great privilege and honor to work and study under their guidance.

I would like to acknowledge the advice given by Dr. Elie Shammas; it was an honor having you on my thesis committee.

I wish to extend my special thanks to all my colleagues at AUB, especially at the Vision and Robotics Lab for helping me reach my goal and for always being there.

I want to express my gratitude to my family members and friends for all that I have accomplished.

# An Abstract of the Thesis of

Nael Akram Jaber for Master of Engineering  
Major: Mechanical Engineering

Title: Detection and Localization of Processionary Moth Nests in Pine Trees  
Using Multi-Stream Convolutional Neural Networks

The Pine Processionary Moth (PPM) is considered the main defoliator of pine trees and a menacing threat to various other perennial species including oak and cedar. Given their negative secondary effects, spraying of pesticides has been banned as a means for the eradication of PPM; instead, an individualized approach is adopted, in which each nest is localized and destroyed. Detection of PPM nests using optical sensing is challenging because of the changing outdoor lighting conditions and the camouflaged appearance of moths in the underlying foliage. In this thesis, a promising solution was proposed for nest detection by fusing sensory data from a standard RGB camera on one hand and a thermal camera on the other. The proposed detection system is built on a two-channeled deep convolutional neural network (CNN), one for each spectrum of the collected sensor data. Experiments performed in a pine forest report successful detection rates with an average accuracy of 97%. Geo-localization is performed to report back the position of the detected nests, within the scanned forest map, by means of an estimation scheme that was designed for this purpose. The accuracy of the proposed geo-localization scheme demonstrated an average localization accuracy of a few centimeters. In summary, this thesis provides a novel scheme to detect and localize PPM nests by creating a localized, tree-level scanning system that can be deployed in urban areas.

# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Literature Review</b>	<b>6</b>
2.1 Object Detection . . . . .	6
2.2 PPM Nest Surveying . . . . .	8
2.3 Deep Learning for crop yield estimation, crop disease assessment, and pest detection. . . . .	10
2.4 Multi-stream Neural Networks . . . . .	14
2.5 Data Synchronization . . . . .	15
2.5.1 Software-Based Methods . . . . .	16
2.5.2 Hardware-Based Methods . . . . .	17
<b>3 Proposed System</b>	<b>19</b>
3.1 PPM Nest Detection . . . . .	19
3.1.1 Deep Learning Frameworks . . . . .	19
3.1.2 Multi-Channelled CNN . . . . .	20
3.1.3 Tuning Feature Extraction Network . . . . .	23
3.2 Tracking . . . . .	24
3.2.1 Method 1 . . . . .	24
3.2.2 Method 2 (Kalman Filter) . . . . .	25
3.3 PPM Nest Localization . . . . .	26
<b>4 Experiments</b>	<b>30</b>
4.1 Datasets . . . . .	31
4.1.1 Dataset-A . . . . .	31
4.1.2 Dataset-B . . . . .	36
4.1.3 Dataset-C . . . . .	36
4.1.4 Data Collection . . . . .	38
4.2 Training and Testing . . . . .	39

4.3	Experimental Setup . . . . .	43
4.4	Validation Experiments . . . . .	48
4.4.1	PPM Nest Detection . . . . .	48
4.4.2	PPM Nest Tracking . . . . .	66
4.4.3	PPM Nest Geo-Localization . . . . .	69
<b>5</b>	<b>Discussion</b>	<b>80</b>
<b>6</b>	<b>Conclusion</b>	<b>86</b>
<b>A</b>	<b>Abbreviations</b>	<b>88</b>



# List of Figures

1.1	(a): PPM feeding on the pine needles [2], (b): Silken PPM nest. . . . .	2
1.2	Schematic illustrating the proposed system and experimental . . . . .	4
1.3	The workflow of the PPM nests detection, tracking, and . . . . .	5
2.1	Pine processionary moth defoliation in healthy (green dots) . . . . .	9
2.2	Tree species identification and pine processionary moth defoliation in pines in the Codo forest site . . . . .	11
2.3	Network architecture of the early-fusion technique [34]. . . . .	15
2.4	Network architecture of the late-fusion technique [34]. . . . .	15
3.1	Architecture of the proposed Multi-Stream Convolutional Neural Network. . . . .	22
3.2	Architecture of the VGG16 feature extraction network [51]. . . . .	23
3.3	GPS coordinate calculation scheme. . . . .	27
3.4	Schematic of the localisation system used to calculate the bearing angle. . . . .	28
3.5	Workflow of the proposed system showing sample outputs at the level of the detection, tracking, and the localisation systems. . . . .	29
4.1	The workflow of the experimental work done in this thesis starting with the data collection process, performing training and testing on different deep learning models, testing the detector on actual trees, reaching the validation experiments which tests the proposed detection, localization and tracking in a combined manner. . . . .	31
4.2	Hand-crafted PPM nest made up from cotton. . . . .	32
4.3	3D model of the hand-crafted PPM nest. . . . .	34
4.4	Sample of the synthetic images. . . . .	34
4.5	Sample of the artificial images. . . . .	35
4.6	Samples of Dataset-A: real (a), synthetic (b), and artificial images (c). . . . .	36
4.7	Samples of thermal images of PPM nests. . . . .	37
4.8	Sample of a paired image of the same PPM nest: RGB on the left, thermal on the right. . . . .	37
4.9	A Sample of detection results from Experiments 1 and 2 . . . . .	40

4.10	Samples of the detection results in Experiment-7 . . . . .	43
4.11	Schematic of the setup used for testing the detector and synchronizing the data during the experiments . . . . .	45
4.12	Bebop drone with FLIR <sup>®</sup> C2 camera mounted on board. . . . .	46
4.13	RTK-GPS methodology schematic [57]. . . . .	46
4.14	Captured RGB, thermal, and overlaid frames showing synchronicity. . . . .	47
4.15	Octocopter used with the FLIR <sup>®</sup> C2 camera, the RGB . . . . .	48
4.16	Three samples of the detection results from each nest of the tested nests. Each row corresponds to the same nest. . . . .	49
4.17	Samples of the results of Experiment-1 showing TN,TP, and FN cases. . . . .	50
4.18	Samples of the results of Experiment-2 showing TN,TP, and FN cases. . . . .	50
4.19	Samples of the results of Experiment-3 showing TN,TP, FN, and FP cases. . . . .	51
4.20	Samples of the results of Experiment-4 showing TN,TP, and FN cases. . . . .	52
4.21	Sample of the results showing the performance of the detector trained on the initial dataset vs the new detector trained on the updated one in identifying the three nests of Experiment-2. . . . .	54
4.22	Output visualizations of three convolution layers of the ResNet101 feature extraction network. . . . .	55
4.23	Output visualization of four ReLu-activation layers while testing a thermal image (ResNet). . . . .	56
4.24	Output visualization of three ReLu-activation layers while testing an RGB image (ResNet). . . . .	57
4.25	Output visualizations of six convolution layers of the VGG16 feature extraction network for an RGB image. . . . .	57
4.26	Three samples of the detections from Experimet 1. . . . .	59
4.27	Three samples of the detections of each nest from Experimet 2. . . . .	60
4.28	Three samples of the detections of each nest from Experimet 3. . . . .	60
4.29	Three samples of the detections of each nest from Experimet 4. . . . .	61
4.30	ROC Curve of Experiment-2 . . . . .	63
4.31	ROC Curve of Experiment-4 . . . . .	64
4.32	Concept of the Intersection over Union (IOU). . . . .	66
4.33	Tracking using the Kalman Filter method - Experiment-1 . . . . .	67
4.34	Tracking using the Kalman Filter method - Experiment-2. . . . .	68
4.35	Tracking using the Kalman Filter method - Experiment-3. . . . .	69
4.36	Tracking using the Kalman Filter method - Experiment-4. . . . .	69
4.37	Comparing the performance of methods 1 and 2 on frames from Experiment-4. . . . .	70

4.38	Schematic showing the experimental setup. The RTK-Base-GPS module is configured in the field receiving the satellite signals and sending corrections to the RTK-Rover-GPS module on board of the drone. The user drives the drone manually scanning the tree for PPM nests. . . . .	71
4.39	Three instances from Experiment-1 showing the estimated . . . . .	72
4.40	Localization result of Experiment-2 showing the drone's . . . . .	73
4.41	Localization result of Experiment-3 showing the drone's . . . . .	74
4.42	Two instances showing the estimated position of each nest . . . . .	75
4.43	Result of Experiment-2 and 3 showing the final output as a 2D map. . . . .	76
4.44	Mosaic Maps created from the frames of Experiment-1 using . . . . .	77
4.45	Orthomosaic map (center) created from Experiment-1 frames . . . . .	78
4.46	Position of the three nests detected from Experiment-2 marked on our created orthomosaic image. . . . .	79

# List of Tables

4.1	Composition of the Image Datasets . . . . .	37
4.2	Detection results of the different experiments and algorithms. . .	41
4.3	Comparison between Faster R-CNN and YOLO in terms of accuracy and computational speed . . . . .	42
4.4	Performance comparison of different detection algorithms upon training and testing using an 80-20% split. . . . .	44
4.5	Detection results of the different experiments conducted on actual trees. . . . .	52
4.6	Detection results of the different experiments and algorithms. . .	58
4.7	Detection results of the four conducted experiments using ResNet 101. . . . .	59
4.8	Detector's performance when tested on the original dataset (Dataset-C) at several thresholds for ROC metric. . . . .	62
4.9	Detector's performance when tested on Experiment-2 frames at several thresholds for ROC metric. . . . .	62
4.10	Detector's performance when tested on Experiment-4 frames at several thresholds for ROC metric. . . . .	63
4.11	Mean Average Precision (mAP) of the two feature extraction algorithms tested on various image datasets (visualization threshold set to 0.5). . . . .	65
4.12	Tracking Results of the four conducted experiments using the Kalman Filter. . . . .	67
4.13	Estimation of the nests' positions in the four conducted experiments. . . . .	74

# Chapter 1

## Introduction

The Pine Processionary Moth (PPM), also known as *Thaumetopoea pityocampa*, is considered one of the most threatening pests to pine trees. Furthermore, although pine forests are the most prone to its attack, other tree species such as cedars and larch are also threatened by them. The life cycle of a PPM is one-year, characterized by the adults emerging during the summer (from June to September), feeding on pine sprouts during the larval phase in the fall and winter, and finally in the pupation phase (March to June), the larvae enter the ground and disperse [1].

PPMs are life-threatening to trees since they feed on the pine sprouts in their larval phase (Figure 1.1(a)), which results in their defoliation. Furthermore, to stay warm during the winter season, PPMs build themselves silken nests (also known as tents) on the branch tips to entrap heat, as shown in Figure 1.1(b).

Caterpillars that turn into PPMs are nocturnal, that is, they come out of their nests at night, crawl onto the pine tree branches and feed on their needles before coming back to rest during the day [3]. Besides reducing tree growth and increasing tree mortality rates [4, 5, 6], PPMs feeding on conifer needles also have indirect negative consequences, such as they render trees unable to resist fires or droughts [7], and make them more susceptible to other hazardous pests, such as



Figure 1.1: (a): PPM feeding on the pine needles [2], (b): Silken PPM nest.

pine beetles.

PPMs require high ambient temperatures to survive and infest. Due to global warming, PPM are no longer restricted to their native areas in Southern Europe, North Africa, and Southern Mediterranean, and are spreading northwards. In fact, PPM spread is becoming a worldwide concern, with colonies starting to appear in North Paris, Brittany, and Strasbourg, with new studies suggesting their expansion in the north-western direction in the near future [8].

PPMs are also considered a threat to humans and to warm-blooded animals because of a toxin (Thaumetopoein) that they release from their natural urticating hairs. The toxin causes severe skin irritation and allergic reactions, and in some cases causes oedema or dermatitis in children [9]. Other side effects include skin rashes and respiratory problems in humans [10], and the toxin has even been reported to kill domestic animals when ingested [11].

To spot PPM colonies in forests, entomologists traditionally manually scan the trees from the ground, and report what can be seen from a limited perspective. Unfortunately, such methods are extremely limited and yield poor estimates inside dense forests, where areas are difficult to access, trees are high, and nests are often not visible from the ground. Traditional PPM treatment and removal

includes spraying of pesticides, but the sprayed chemicals pose a risk to the environment and are banned in many countries, especially when the infested trees are located in rural areas [12]. Alternatively, a new eco-friendly solution was proposed in which a combination of entomopathogenic fungi and essential oils were used to treat the PPMs as an alternative to chemical pesticides [13]. This solution proved to be most effective when the essential oils are injected into the nests of third instar larvae achieving a significant mortality rate of 87.4%, but it was not effective on the fourth instar larvae with a reduced mortality rate of less than 5%.

As an alternative to mass spraying pesticides in an indiscriminate manner, some farmers have resorted to identifying the location of each nest inside of a forest and manually destroying them. While this seems like a daunting task for humans to do, the task lends itself well for automation in which an aerial drone can autonomously detect and destroy all of the nests in a given forest. However, to be able to do so, the drone must first detect and localize the nests within a forest area.

While the long term goal of this work is to locate the PPM nests and exterminate the moths, in this thesis we restrict our intervention to the ability to automatically detect and localize moth nests in each tree. The detection of the caterpillars themselves is very technically challenging, hence in lieu of doing so people have attempted to detect their nests, which they inhabit for a considerable part of their life cycle. PPM nests are relatively large and salient, but they vary widely in their geometry, size, and construction style.

This thesis proposes an automated system for the detection and localization of PPM nests inside of a pine forest, as represented in Figure 1.2. The proposed detection system employs a deep convolutional neural network (CNN), which

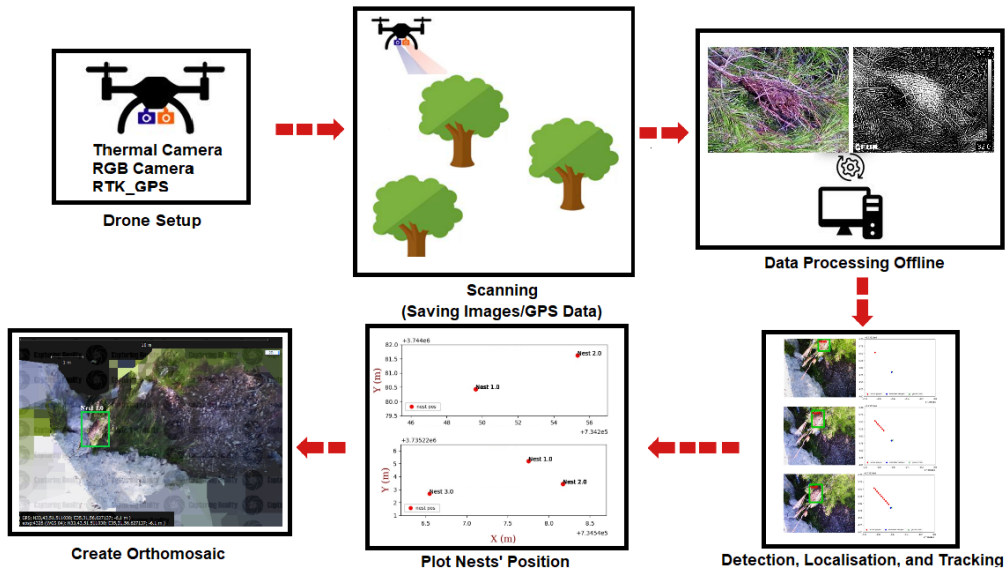


Figure 1.2: Schematic illustrating the proposed system and experimental procedure. The drone scanning the trees for PPM nests carries an RGB camera, thermal camera, and a Real-Time Kinematic GPS (RTK-GPS) on board. The captured data are stored and processed later on in an offline manner, fed into the system to perform detection, localization, and tracking before reporting back the position of the detected PPM nests on a 2D-plot, and on an orthomosaic image as a better visualization.

takes input sensory data from two cameras: the first being an RGB camera, and the second a thermal camera. Once detected, each nest is geo-localized on a two dimensional (2D)-geo-reference map. The workflow is shown later in Figure 1.3. Experiments in a real pine forest demonstrated the success of the proposed system with an average nest detection accuracy of 97% and an average localization accuracy of few centimeters ( $< 10cm$ ).





Figure 1.3: The workflow of the PPM nests detection, tracking, and geo-localization.

# Chapter 2

## Literature Review

### 2.1 Object Detection

Detection can be defined as the process of classification and localization combined, where classification entails determining whether an object is present in an image or not, without specifying its position in the image, whereas localization is finding the position of an object within the given image. In this context, detecting PPM nests entails discerning the presence of nest(s) as well as specifying their locations within a given image.

Extensive research has been conducted in the field of detecting objects in two-dimensional (2D) images, where in the past decade deep learning methods have significantly surpassed traditional hand-crafted methods [14].

In hand-crafted features-based algorithms, the type of features to look for are specified beforehand. For example, [15] used a Gaussian model classifier for face detection applications. [16] proposed detecting faces in images using Haar

wavelets features. Similarly, [17] used the same wavelet feature in proposing a general detection framework, and tested it on face recognition and on human body recognition. Such hand-crafted methods proved to be effective in some applications but had lower accuracy than the more recent CNN-based detection algorithms [18].

On the other hand, deep learning algorithms use a large amount of labelled images (dataset) to learn what features are most suitable for the problem at hand. Using CNNs, significant improvements have been reached in the most important Computer Vision problems such as segmentation and object detection. A study by [19], which compared different deep learning methods—including CNNs, Deep Boltzmann Machines (DBMs), Support Vectors Machines (SVMs), Gaussian and Markov Processes, and other traditional Computer Vision methods—showed the typical uses of each method, and where each method excels given the appropriate type of data. It proved that Deep Neural Networks (DNNs) are best at dealing with new representations of data, which gives it an advantage over other learning methods. Moreover, it was shown that classifiers such as SVMs and K-Nearest Neighbors (KNNs) are generic and not robust to the diversity of data [14].

Solutions based on Deep Neural Networks have widely spread in the past decade due to their high accuracy in solving real world problems. For example, [20] used a two eight-layered CNN in a coarse-to-fine manner for the purpose of detecting inshore ships and they achieved an accuracy of 95.3%. [21] proposed using a light-weighted deep CNN instead of heavy one for the purpose of detecting tiny objects (objects in small-scaled images), since heavy CNNs involve a large number of parameters accompanied by large-scale datasets, and training them with small datasets causes over-fitting. They trained and tested their network on the CIFAR-10 (which includes 10 classes, with 6000 images per class) and CIFAR-

100 (which includes 100 classes containing 600 images each) datasets achieving 94.34% and 73.65% accuracy, respectively. [22] used deep CNNs on RGB-D images containing small household objects (such as fruit, vegetables, boxes, and water bottle) for the purpose of their recognition. Their approach achieved a 91.84% accuracy.

The disadvantage of using deep learning algorithms is their need for large datasets for training the detector and their large computational costs, which makes it difficult to use deep learning on applications with limited datasets. To mitigate this issue, networks in transfer learning are previously trained on very large datasets before they are subsequently trained on new sets of data. Transfer learning helps in reducing execution time, minimizing computational expense, and increasing accuracy levels. For example, in their work towards object and action detection, [23] used layers previously trained on the ImageNet dataset while computing mid-level image representations of those in the dataset of PASCAL VOC; their work showed improved results compared to the ones found in the literature.

## 2.2 PPM Nest Surveying

Research studies investigating PPM's infestation in forests using aerial images are on the rise. Cardil et al [7] tackled a similar problem, with the aim of measuring the defoliation in a pine field caused by PPMs, as shown in Figure 2.1. Using unmanned aerial systems (UASs), they collected RGB images from very high altitudes to cover a large area of the monitored field in order to generate orthomosaic maps from the captured images. Manual labelling of the images was performed, followed by training a supervised maximum likelihood classifier, which yielded an

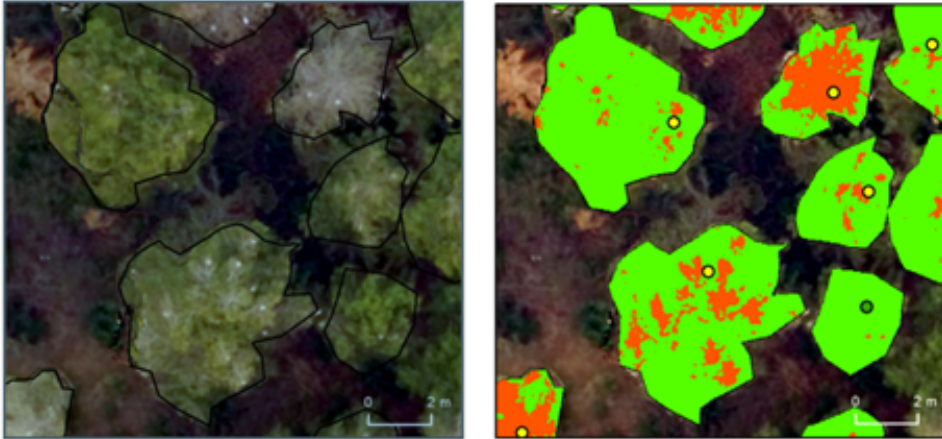


Figure 2.1: Pine processionary moth defoliation in healthy (green dots) and infested trees (yellow dots) in Torregassa forest. Manual crown segmentation is performed on the left side, and the automatic classifier measured defoliation as infested (red) or non-infested green using RGB-UAS images on the right side [7].

accuracy of 79% upon testing. Since their work covers defoliation estimation on 2D orthomosaic images, they are not able to estimate or measure the tree volume and they cannot access lower parts of the trees, which may contain a considerable number of nests, thus resulting in an inaccurate tree-level defoliation estimator. Furthermore, they state that their proposed system faces restrictions represented by having a limited time window (around noontime) for image acquisition for the sake of minimizing the amount of shadowing in the images, and by considering the terrain aspect due to the shadows and brightness which may affect the final classification outputs depending on the sun illumination. Such a classification system can only be used to report back the infected areas of a field and can not be used later for a localized tree-level solution. Hence, the proposed approach in this work requires a detection system that works on a tree-level, where it should be able to locate the nest's position within a tree with high accuracy.

In a more advanced work in 2019, [24] estimated the defoliation rate in a pine-oak mixed field caused by the PPM. After acquiring aerial multi-spectral

images, ortho-mosaics were also created and used to estimate vegetation indices, and to quantify the level of defoliation in the tested forests by reporting back infested areas, and classifying each tree as non-defoliated, partially defoliated, or completely defoliated as shown in Figure 2.2. The obtained results yielded an accuracy of 81.8%. Although they use multi-spectral imaging, their field assessment is different from the method proposed in this work. In this updated research, they still aim to report back infected areas and classify tree defoliation levels, which may suffer from low accuracy while scanning from high altitudes. Furthermore, they still miss a considerable number of nests using this classification method in the inaccessible lower branches of the trees. Next, we provide a survey of related works and categorize the literature per research area.

[11] identify the location of PPM nests in aerial images by first segmenting each captured image into three categories: foliage, PPM nest, and floor. Their results were relatively accurate for floor and foliage with rates of 95% and 99%, respectively; however, they obtained sub-optimal classification results of PPM nests at approximately 72%. In this thesis, I elect to perform detection, and not segmentation, for nest identification. Furthermore, I exploit the nest’s silky characteristic of being a heat absorber, thus rely on multi-spectral images that contain stark and complementary features to be learned by the network.

## **2.3 Deep Learning for crop yield estimation, crop disease assessment, and pest detection.**

Deep learning is being widely applied in agriculture upon different approaches. For instance, [25] proposed an automatic system that classifies plant types based on the shape of their leaves. They used artificial neural networks to perform

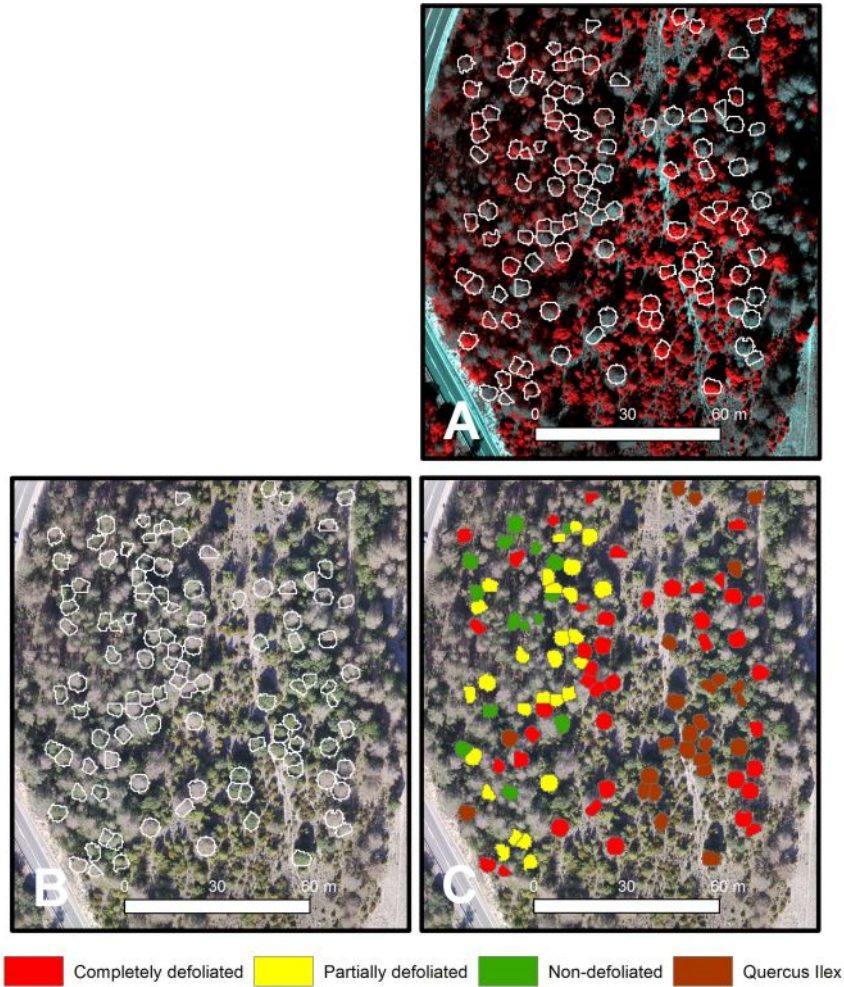


Figure 2.2: Tree species identification and pine processionary moth defoliation in pines in the Codo forest site. (A) Individual automatic tree delineation on the colour-infrared composite orthomosaic using green, red and nir reflectance bands. (B) Individual automatic tree delineation on the RGB orthomosaic. (C) Automatic tree classification in the field as holm oak or non-defoliated, partially defoliated and completely defoliated pine through multispectral highresolution imagery [24].

classification after training their algorithm on 817 leaf samples, yielding an accuracy of 96% when tested on two datasets (Flavia and ICL). [26] proposed a similar approach to perform classification on 22 different weed and crop species using convolutional neural networks. They trained their algorithm on six different datasets (total of 10,413 images), which consisted of different varying conditions such as lighting, resolution, and soil types achieving 86.2% classification accuracy. In our proposed solution, we perform detection on the input images, in addition to classification which allows recognizing the nest and reporting back its position in the image frame. Furthermore, the proposed system utilizes features from the thermal spectrum to boost its detection accuracy, as opposed to being trained on RGB images alone. Deep learning has been also employed to detect plants and estimate crop quality and yield. For instance, deep learning was used in [27] to detect and count green mangoes by training a YOLO V2 model on a dataset, collected aerially using a UAV, achieving a mean average precision (mAP) of 86.4%. In [28], deep learning was used to detect passion fruits and classify their maturity level using RGB-D images, combining color and depth information to boost the detection performance. A Faster R-CNN model was trained and tested on achieving a 92.71% detection accuracy. In similar work, [29] also used RGB-D images in order to detect apple fruits in dense foliage fruit-walling trees. Although they used Faster RCNN for the deep learning model, the approach was different from [28] in the sense that they used depth information to exclude background objects from the RGB images, creating two different datasets (original and the one including only foreground objects) to train and test on. The network which was trained and tested on Foreground RGB dataset, yielded the best performance with 89.3% mAP. In [30], a deep learning algorithm was trained on a dataset, collected aerially from multiple altitudes (40, 50, and 60 meters), to detect and



count banana plants in a field. The dataset consisted of RGB images only and the algorithm yielded relatively high accuracy rates of 96.4%, 85.1%, and 75.8% for the aforementioned altitudes, respectively.

Deep learning has been also applied for crop disease assessment as in [31] who trained and tested the performance of several deep learning architectures in recognising tomato plant diseases and pests. In [32], the authors trained a detector using convolutional neural networks to identify apple leaf diseases in real-time. The apple leaf disease dataset (ALDD) was used to train and test the algorithm on laboratory images and more complex images under real field conditions, which consisted of 26,377 images of diseased apple leaves. The system was able to identify five different apple leaf diseases with a mean detection accuracy of 78.8%. We here note that detection of large banana plants is relatively easier than detecting PPM nests, which is due to 1) the lack of access to a large dataset that accommodates for the vast variability in the shape, color texture, lighting conditions, and backgrounds of PPM nest; and 2) the inability to rely on depth values as a beneficial feature for learning given the nests' varying positions and distances from the camera. Therefore, a more radical solution is established in this work by using multi-spectral imaging to avoid solely relying on the visible spectrum.

An innovative solution to boost the detection performance of immature citrus fruits was established in [33]. Similar to the problem faced in PPM nest detection, immature citrus fruits are very challenging to detect since their color is very similar to the leaves, which makes RGB cameras inefficient. For this, they mounted a thermal camera to a ground vehicle with a water spray system, which applies mist to citrus trees. The water mist induces a temperature difference between the fruit and leaf surface, which makes the immature citrus fruit

more visible in the thermal spectrum. The authors collected their own dataset and used it to train and test the performance of two deep learning detection models, Faster R-CNN and SSD, which yielded 87.2% and 85.5% mAP values, respectively. In their work, they discounted the visible spectrum since it did not always provide beneficial learning features for their application, thus they solely relied on thermal images. Whereas the solution that we are herewith proposing to detect PPM nests involves training a dual-channeled CNN on data from both visible and thermal spectra with the aim of achieving optimal features learning.

## 2.4 Multi-stream Neural Networks

The solution that we propose for detecting PPM nests involves the fusion of sensory data from an RGB camera (visible spectrum) and a thermal camera (infrared spectrum), where the two sources are fed into a multi-channeled deep network. Similar approaches have been proposed before for other applications. For instance, [34] proposed two multi-channel CNNs for pedestrian detection. In the first solution, RGB and thermal images are concatenated to form a four-channeled input (RGBT) image fed into a single-stream network, called the early-fusion method, shown in Figure 2.3. The second architecture inputs each image to its own stream and processes each image in a sub-network; feature maps extracted from each stream are set to be concatenated, and then enter a fully connected layer, as shown in Figure 2.4. Results demonstrated the superiority of late-fusion versus early-fusion methods, showing that the best learning of features occurs in the dual-streamed network. In [35], the aforementioned fusion methods were tested on a fruit detection application, with the aim of performing yield estimation and automated harvesting. Training and testing was performed on a

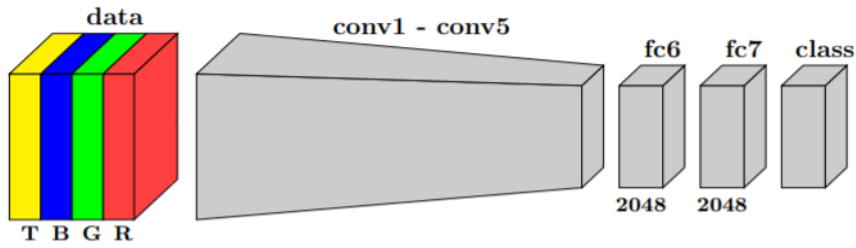


Figure 2.3: Network architecture of the early-fusion technique [34].

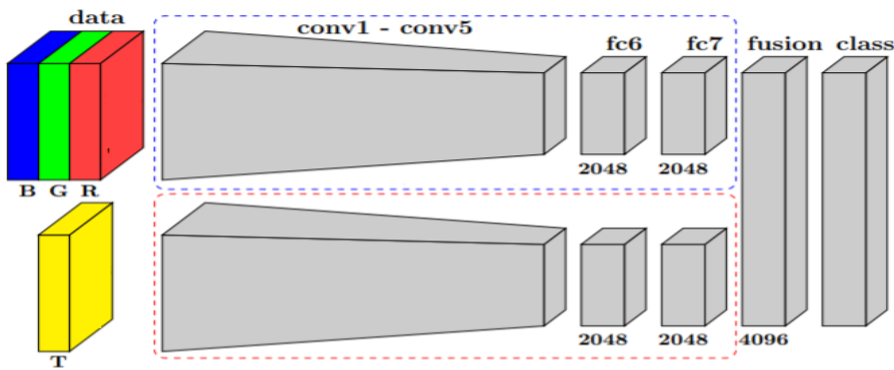


Figure 2.4: Network architecture of the late-fusion technique [34].

dataset containing both RGB and near-infrared images of seven different fruits. The obtained results demonstrated boosted accuracy, a noticeable reduction in training time, and the late-fusion technique outperforming other methods again.

## 2.5 Data Synchronization

To insure that measurements acquired from multiple sensors correspond to the same object and time instance, care must be taken to synchronize the data acquisition from the various sources. Three common synchronization methods are found in the literature to achieve this task.

### 2.5.1 Software-Based Methods

This approach for camera synchronization is based on various "post-processing" algorithms which are applied on unsynchronized sequences. Knowing the time-shifts between a significant number of view pairs, the entire network can be synchronized. For example, some papers [36] rely on the cameras' built-in microphone to capture sound and use the audio signal to align video frames. [37] considered post-processing based on the timing of a flickering light source recorded with a shutter speed of 1000 Hz. Also, [38, 39] use flashes to synchronize individual camera timelines. Such a method for example is not always applicable, such as in our case, because light sources may be either obstructing the field-of-view, or overexpose the entire frame.

In [40], Sinha et al proposed an automatic approach to synchronize a network of uncalibrated and unsynchronized video cameras by computing the epipolar geometry from dynamic silhouettes and finding the temporal offset between them. There are other methods similar to the aforementioned ones, but all are related to the same family of solutions which is based on including strong markers or features to the pair of images in order to match and synchronize.

In [41], a more sophisticated approach to software-based camera synchronization is introduced. This method uses the server-client architecture such as the previous method with a simple error-checking technique. The problem of the discrepancy in synchronization is solved by calculating the time for sending data over the network by sending some test data and calculates the network latency. Then this latency is added to the reference time and sent to all the computers which they accept this value as their time value thereby synchronizing their clocks with the true reference time. This technique improves the camera synchronization, but assumes a constant latency which is added to the reference time.

These post-processing methods are quite effective in automatically recovering the frame temporal offset between image sequences and thus enabling the "post synchronization" of the cameras. However, these algorithms cannot be applied real-time and assume a constant temporal offset. Furthermore, such approaches are sensitive to occlusion, since they rely on the tracking of image features; which can be hidden by another imaged surface at arbitrary time frames.

## 2.5.2 Hardware-Based Methods

Hardware-based solutions are based on having an external device (Multiplexer, Videos encoders, etc.) which is directly connected to the cameras to trigger the recordings on all connected devices at the same time. Several approaches for synchronization at the hardware level are proposed in [42] using either specialized cameras or external dedicated electric signal.

All these hardware-based solutions properly and precisely synchronize the cameras but they are also potentially costly, technically complex and not very flexible (not applicable for many applications).

Another synchronization methods relying on hardware is known as the Special-Purpose hardware methods, and it's the most common one to use. It usually consists of a microcomputer control unit which is dedicated to propagating external synchronization signals for triggering the cameras and achieving the cameras synchronization. In [43], the proposed system consists of the camera-computers and the triggering-computer. This triggering computer can launch simultaneously all the cameras. Then, the cameras will immediately start capturing an image. Since there isn't any proposal to handle the situation in case of failure of the synchronization, the method is dependent on the quality of the Ethernet connection, the operating system's latency of response to the received triggering signal,

as well as the camera drivers and hardware. Thus it is of course inconsistent and inaccurate.

In [44], which is one of the very recent papers (2018) used such a method to capture images from different cameras to construct a panoramic image. Their approach was to directly connect the recording machines to the cameras and send trigger signals from the machines. The challenge was the synchronization accuracy of the signals sent to the cameras. Their external trigger box that sends a signal to all the cameras to capture an image was an Arduino device which uses its local clock to send a broadcast shutter trigger signal every  $1/\text{fps}$  seconds.

The most suitable technique to adopt was the Special-Purpose Hardware Method where a Raspberry Pi was used as an external trigger for the cameras to capture frames synchronously. The idea is in avoiding recording a whole video sequence at different frame rates, and then syncing them in post processing using the software-based techniques mentioned before. The aim was to capture two frames from both cameras at the same time for 'x' times per second. In this way, we get the same number of frames from both cameras which should be of the same exact scene.

# Chapter 3

## Proposed System

### 3.1 PPM Nest Detection

#### 3.1.1 Deep Learning Frameworks

The image datasets are used for training and testing on several deep learning frameworks to arrive at an optimal solution. Training of the detector is first performed using the open source framework built by TensorFlow [45], which is the TensorFlow Object Detection API. This API provides the ability of constructing, training, and deploying object detection models. Transfer learning is applied using pre-trained neural networks on the created dataset of the PPM nests.

TensorFlow provides several detection models that are pre-trained on the COCO dataset [46], the Kitti dataset [47], and several others. The pre-trained models are advantageous when initializing the training on personalized datasets. Each pre-trained classifier has its own neural network architecture, thus the most suitable model is selected based on the application's needs. Faster RCNN was chosen to be the most suitable architecture for our application given that it achieves the highest accuracy in comparison with other models, at the cost of

being computationally expensive. Since the detection process in this application is performed offline, in a post-processing manner and not in real-time, Faster R-CNN is considered the best option.

On the other hand, you only look once (YOLO) [48] is a real-time object detection system that uses a different architecture from that of Faster R-CNN. YOLO does not utilize a Region Proposed Network (RPN), rather it employs 24 convolution layers and two fully connected (FC) layers. So the input is the image itself passing through the convolutions reaching the fully connected layers with a number of bounding boxes that may contain an object. Classification occurs at this point displaying the ones above a specific threshold. YOLO is considered much faster than Faster R-CNN with very good detection accuracy relative to its speed. However, for applications that are not being performed in real-time (offline processing), speed and computational power are not considered an issue. So applications aiming for highest detection accuracy would adopt the Faster R-CNN instead.

An ‘Inference Graph’ that contains the object detection classifier is exported in TensorFlow, and a weights file is exported into YOLO. The detector is tested using a test subject such as an image, video, or camera feed of a PPM nest, where the exported ‘Inference Graph’/ ‘Weights File’ is fed as an input and the imported test subject with a bounding box around the detected PPM nest, if any, as the output.

### **3.1.2 Multi-Channelled CNN**

The proposed network is a multi-stream Faster R-CNN architecture [49]. For the detection of PPM nests, high accuracy is desired to detect (and later eradicate) the vast majority of present nests, even if computational power and time



are required. As such, we choose Faster R-CNN [49] as the PPM nest detector network. Faster R-CNN is an updated version of Region-based Convolutional Neural Network (R-CNN), which underwent several iterations until evolving into the Faster-version. In R-CNN, the input image is scanned for possible objects using selective search to generate region proposals given that manual labeling is performed beforehand. Features are then extracted from the proposed regions by running a CNN on each region, followed by Support Vector Machine (SVM) classifier for region classification. R-CNN is computationally expensive since a CNN is performed on each image based on its specific feature map. Hence, Fast R-CNN and Faster R-CNN were subsequently proposed to increase the computational efficiency of the algorithm. A Faster R-CNN reduces the dimension of the feature map by having a sliding window move across it, outputting a pre-defined number of regions (bounding boxes) that may contain an object. The output is represented by these regions along with a weight and coordinates that are assigned to each. This process, known as Region Proposed Network (RPN), is more computationally efficient since it guides the detector where to look for objects instead of checking the entire initial feature map.

This architecture is adopted and applied on a dual-stream network consisting of two branches, one for the RGB image and the other for the corresponding thermal image, as shown in Figure 3.1. The architecture is initialized with pre-trained RGB and thermal weights. The RGB image and its corresponding thermal image are passed through Resnet Blocks which together constitute a 3x3 convolutional layer design consisting of a convolution layer followed by a batch normalization layer (BN) and a ReLU activation function. Each branch of the dual stream framework outputs two feature maps of 1024 dimension each. These two feature

maps are stacked and passed through a 1x1 convolution to learn combining these features appropriately for the given task. The output of the 1x1 convolution is directly passed into the rest of the Faster-RCNN network, which is the RPN. Regions produced by the RPN are then cropped out of the feature map and passed into a classification layer, which learns to classify the objects in each ROI. The RPN loss is adapted from the Faster R-CNN and is denoted by:

$$L(p_i, t_i) = \frac{1}{N} \sum_{i=1} L(p_i, p_i^*) + \lambda \frac{1}{N} \sum_{i=1} p_i^* R(t_i, t_i^*), \quad (3.1)$$

where  $i$  is the index of an anchor,  $p_i$  is the predicted probability of anchor  $i$  being an object,  $p_i^*$  is the ground truth,  $t_i$  represents the coordinates of the predicted bounding box,  $t_i^*$  represents the ground truth bounding box coordinates,  $L$  is the logarithmic loss,  $R$  is the robust loss function (smooth  $L1$ ) as defined in [49], and  $\lambda$  is a hyper-parameter. The multi-task classification and regression loss at the end of the network are adopted from Fast R-CNN in [49].

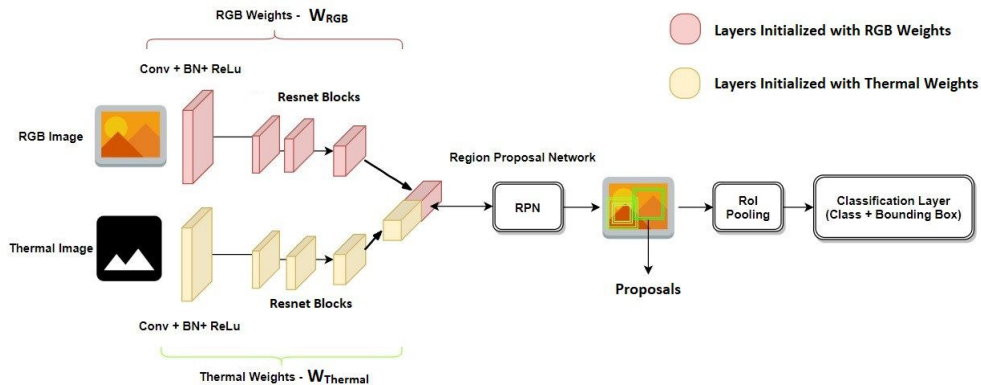


Figure 3.1: Architecture of the proposed Multi-Stream Convolutional Neural Network.

### 3.1.3 Tuning Feature Extraction Network

To enhance the performance of the deep network, we first investigate potential improvements at the feature extraction level; for this purpose, we test the efficiency of another feature extraction network: VGG16. The architecture of the ResNet blocks used earlier [50] is represented by 3x3 convolutional layer design, followed by a batch normalization (BN) layer and a rectified linear activation function (ReLU). VGG16 is represented by a stack of convolutional layers followed by three fully connected (FC) layers that output a feature map [51], shown in Figure 3.2.

A contribution lies in testing these two feature extraction networks in this multi-channel manner, with this performance comparison not being previously performed in such architecture.

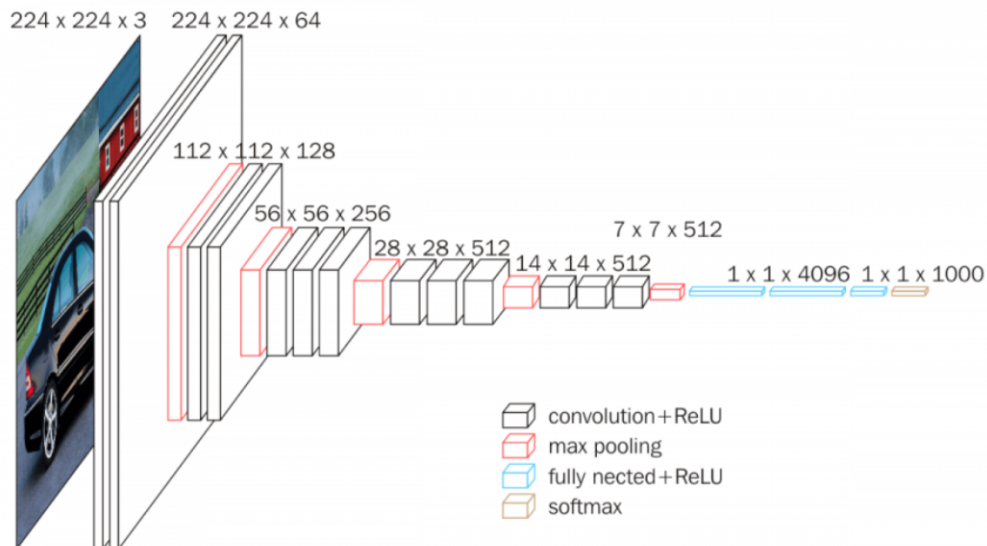


Figure 3.2: Architecture of the VGG16 feature extraction network [51].

## 3.2 Tracking

To avoid replication of detected nests since the same nest can be detected in several subsequent frames, we account for data correspondence by tracking the position of each nest once it first appears in a frame. This object tracking or data association could be performed in several ways, for this two methods were applied.

### 3.2.1 Method 1

The first technique was to combine three data association methods, and use them to track the detections. These three methods are represented by the IOU metric, center tracking, and GPS thresholding. The IOU metric is basically calculating the IOU between the two consecutive detections, and if these two detections overlay (match) then it means that same object was detected. Center tracking method is represented by calculating how much the center of the bounding box shifted from one frame to another. The GPS thresholding is a similar method to check the GPS coordinates of two consecutive frames, so as long as the detections refer to the same GPS position upon a minimal tolerance then it means that the same object is detected.

A python code was prepared and tested on several experiments, with a very good performance where the tracking was performed properly, but the main problem of this method is that it relies always on two consecutive frames. So for example, in case a false positive was recorded in two consecutive frames, this false positive object will be given an ID. Moreover, it relies on three different thresholding values, which makes it sensitive to any unpredicted variation, thus the tracking easily fails.

### 3.2.2 Method 2 (Kalman Filter)

The second method is commonly used for object tracking applications which is the Kalman Filter. Kalman Filters are very popular for tracking and predicting objects in current and future positions. Bewley et al. [52] implemented this method for object tracking where they used YOLO object detector using PyTorch [53] framework to feed the Kalman Filter with detections. The work of this paper was applied to our work feeding the Kalman Filter the detections of our system.

As mentioned earlier, a Kalman Filter estimates the position of the detected object in the coming up frames. So, when the system started to detect an object and since the first frame, the Kalman filter (KF) starts estimating the position of the detected object in the next frame. This estimated bounding box by the KF, which means the estimated position at time  $t + 1$  or in the next frame, is matched with the actual detection at time  $t + 1$  to check the IOU between them. If the IOU value is higher than a predefined threshold (0.5), the prediction is deemed accurate and tracking is resumed; if not, as is the case for false positives, tracking is assumed to have failed and the filter eliminates this falsely tracked object from its input.

The inputs of the Kalman Filter are two lists of bounding box coordinates: prediction and tracking. The IOU is calculated for each detection with the predicted trackers, giving the same tracker identification (ID) to the matched detection. Thus, the KF provides the position of the bounding box at time  $t + 1$  based on its previous position at time  $t$ . The state of each tracker is represented by  $x = [c_x, c_y, w, h, c'_x, c'_y, w']^T$ , where  $c_x$  and  $c_y$  represent the center coordinates of the bounding box,  $w$  and  $h$  represent the width and height of the bounding box, and  $c'_x, c'_y$ , and  $w'$  represent the changes (velocities) of each of these parameters. When a detection is associated to a correct prediction, the state vector is updated

by solving for the velocity components. The approximation of a target's position from one frame to another is based on a constant velocity model.

### 3.3 PPM Nest Localization

In the work of this thesis, the goal is to detect and localize the PPM nests, and report back its position. In case the drone does not pass perfectly on top of a nest, the nest would be detected at the borders of the image, which would give a slightly inaccurate GPS value. Hence, a GPS estimation method is implemented to report back the actual position of the detected nests.

The idea applied is already used on large scales, where as shown in Figure 3.3, given initial GPS coordinates  $Lt_1$  and  $Ln_1$  (latitude and longitude) of the aerial drone and the distance  $d$  from the desired waypoint, the bearing angle ( $brng$ ) between this desired point with respect to the North is calculated using (3.2):

$$Lt_2 = \sin^{-1}(\sin(Lt_1) \cdot \cos(d) + \cos(Lt_1) \cdot \sin(d) \cdot \cos(brng)), \quad (3.2)$$

$$Ln_2 = Ln_1 + \tanh^{-1}\left(\frac{\sin(brng) \cdot \sin(d) \cdot \cos(Lt_1)}{\cos(d) - \sin(Lt_1) \cdot \sin(Lt_2)}\right).$$

As mentioned earlier, each matched pair of captured images has a corresponding GPS position, which represents the center of the frame at  $(Lt_1, Ln_1)$ . We note that the distance between the center of the frame and the nest cannot be determined using a monocular camera since it lacks depth perception. For this reason, we make a practical simplification that considers the standard size of PPM nests (15cm) and assume a constant scanning altitude of 2m above the trees. Hence, using pixel counting and the aforementioned assumptions, distance

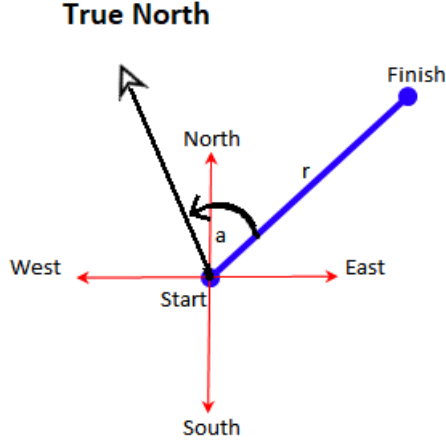


Figure 3.3: GPS coordinate calculation scheme.

d is estimated.

In case the drone is controlled manually, access to the bearing angle is lost, thus we propose a solution based on equation (3.2), where given the GPS position of two points represented by  $(Lt_1, Ln_1, Lt_2, \text{ and } Ln_2)$ , the bearing angle "*brng*" between them, with respect to the North, can be calculated as follows:

$$\begin{aligned}
 \Delta L &= Ln_2 - Ln_1, \\
 X &= \cos(Lt_2) \times \sin(\Delta L), \\
 Y &= \cos(Lt_1) \times \sin(Lt_2) - (\sin(Lt_1) \times \cos(Lt_2)) \times \sin(\Delta L), \\
 brng &= \tan^{-1}\left(\frac{X}{Y}\right).
 \end{aligned} \tag{3.3}$$

Since detection and localization are performed in a post-processing fashion, the GPS coordinates of all frames are recorded. Using Equation (3.3) and as illustrated in Figure 3.4, the bearing angle is calculated between each two consecutive GPS recordings.

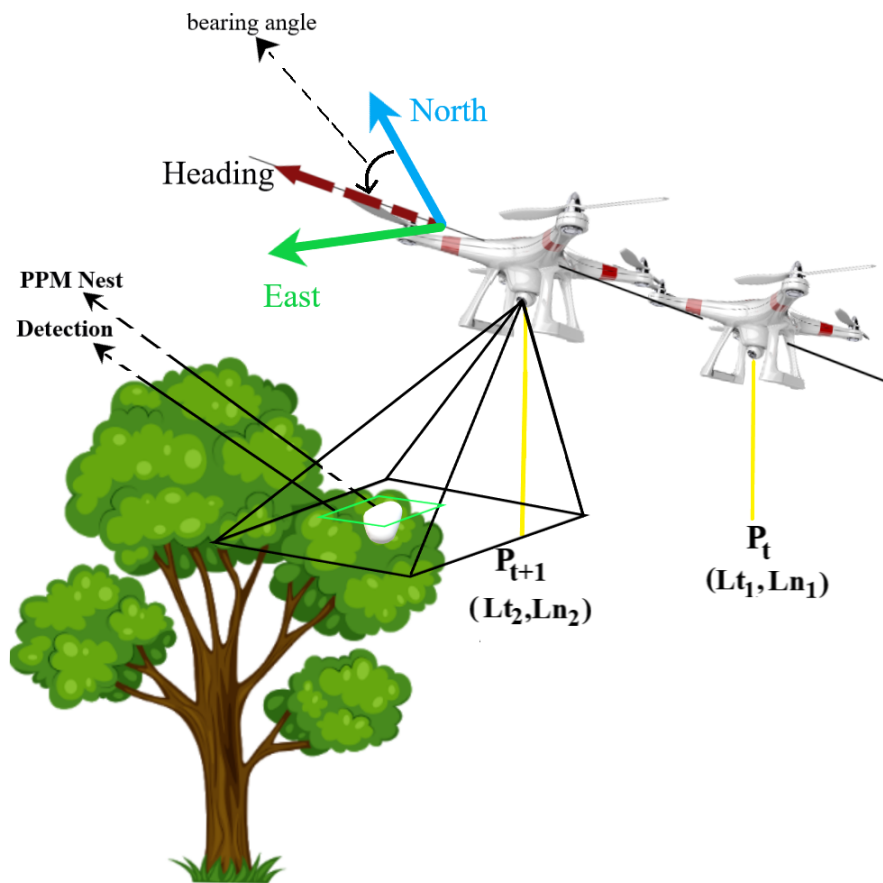


Figure 3.4: Schematic of the localisation system used to calculate the bearing angle.

Figure 3.5 is a scheme showing the workflow of the proposed system, and a sample output at the level of each sub-system.



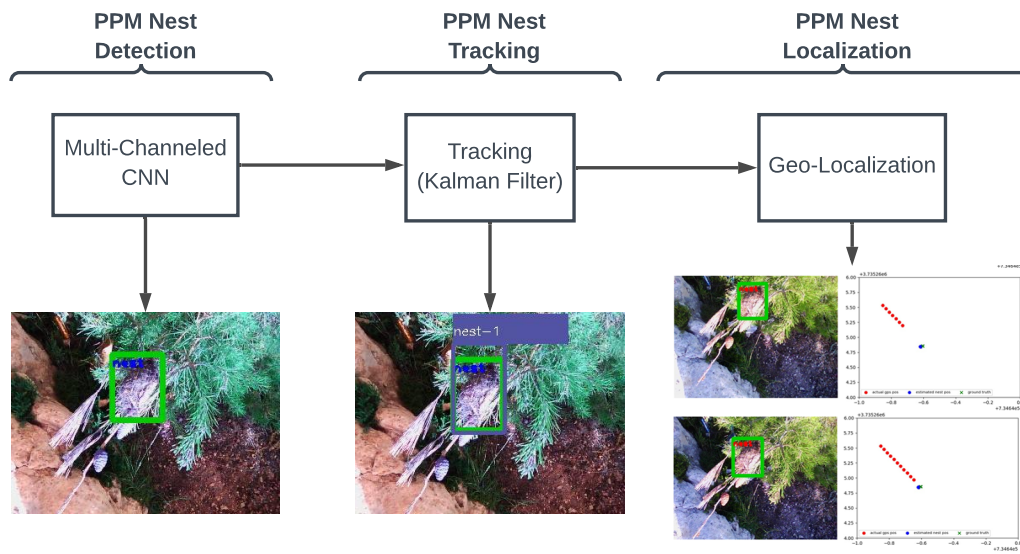


Figure 3.5: Workflow of the proposed system showing sample outputs at the level of the detection, tracking, and the localisation systems.

# Chapter 4

## Experiments

In this section, results of the training and testing of several deep learning models on our collected datasets are provided as a primary work, which motivates the use of the proposed multi-channelled CNN. The results of conducted validation experiments in a pine field are then provided to demonstrate the detection, tracking, and localization performance of our proposed system. Figure 4.1 provides a visual illustration of the experimental procedures and a clear breakdown of the workflow.

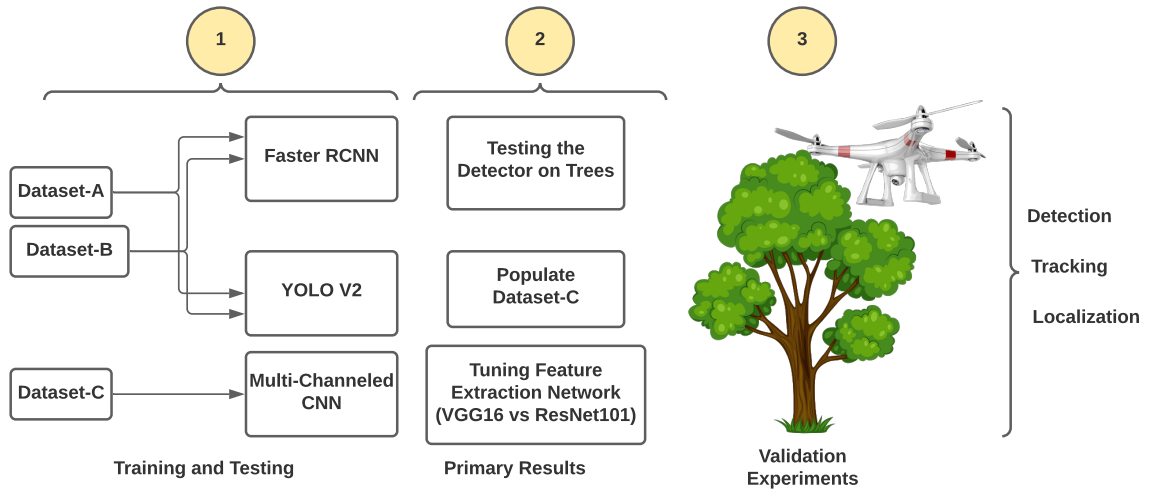


Figure 4.1: The workflow of the experimental work done in this thesis starting with the data collection process, performing training and testing on different deep learning models, testing the detector on actual trees, reaching the validation experiments which tests the proposed detection, localization and tracking in a combined manner.

## 4.1 Datasets

Three datasets are gathered in this work for detecting PPM nests: Dataset-A that includes real, synthetic, and artificial RGB images; Dataset-B that includes (unpaired) thermal images of PPM nests; and Dataset-C that includes paired thermal and RGB images that are simultaneously taken of the same PPM nests.

### 4.1.1 Dataset-A

With the aim of designing a standard single-channel object detector, three types of RGB images are collected: real, synthetic, and artificial. The real images of PPM nests are taken by a hand-held digital camera and an aerial drone's camera as well as some images of PPM nests found on the internet. Synthetic images are virtually synthesised via rendering software tools to replicate PPM nests on tree

branches, whereas artificial images are real images of artificially created nests, using materials that mimic those of PPM nests, placed on real pine trees.

### **Synthetic Images**

Synthetic images are images of moth nests attached to pine trees that are created purely using software. For the creation of the synthetic trees, 5 tree version were created using the open-source Plant Factory 2016 [54] software and transferred to the VUE Infinite 2016 R5 PLE [55] software. On the other hand, for creating synthetic PPM nests, three 3D structures of the nests were constructed using cotton. Figure 4.2 shows one of the hand-crafted 3D nest structures.

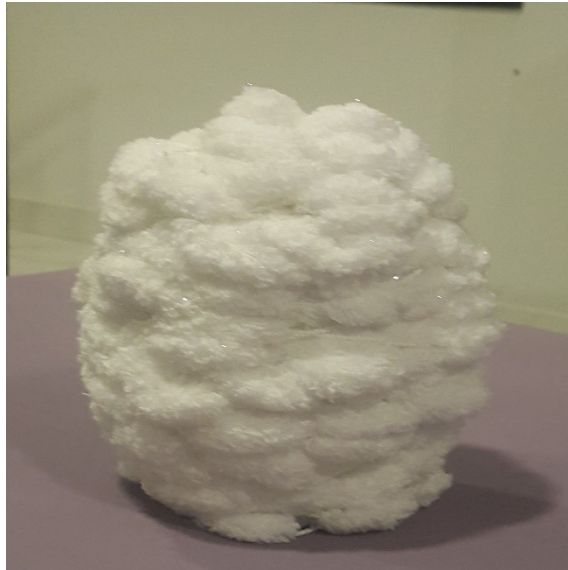


Figure 4.2: Hand-crafted PPM nest made up from cotton.

Afterwards, 3D models of the cotton-based nest structures were developed by applying structure from motion as explained in the following section, and was transferred to the VUE Infinite Software [55].

## Structure from Motion

In order to have a realistic PPM nest model to be used for image synthesis, structure from motion (SFM) was applied. SFM is based on using a series of 2D images of a specific object to reconstruct the 3D model of it, having these 2D images taken from all angles to cover the entire structure. Applying this technique, can provide a high-resolution digital copy of the 3D model, reconstructing the object with all its little details from color texture to curved edges, without the need of expensive equipment. To apply this technique, a hand-crafted nest was prepared using cotton, and well-prepared to look realistic. Afterwards, images of this handcrafted nest were taken from different angles covering the whole object. To apply structure from motion, several specialized software packages are available, which are able to automatically identify matching features in these images. Estimates of camera positions and orientations are produced by tracking the features from one image to another. After processing of images and deriving camera positions and orientations, dense cloud is created calculating depth points, which afterwards allows the meshing of these points, creating the 3D model. Finally, texture can be added to the meshed 3D model completing the rest of the missing features such as shades and color. The used software package for creating the 3D model was Agisoft PhotoScan [56], and the 3D model was exported to VUE software for scene synthesis. Figure 4.3 shows the 3D model developed on Agisoft PhotoScan for one of the hand-crafted nests.

After developing models for the pine trees and moth nests and transferring them to the VUE Software, various combinations of PPM nests, trees, lighting conditions, and background scenes were created and rendered on two computers, the first having an Intel Core i7-7700HQ processor and 16GB of DDR4 RAM, whereas the second having Intel Core i5-7200U processor and 8GB RAM. Ren-

dering of an image having 1060x707 pixels and around 40,000 polygon meshes took an average of 4 minutes on the first computer, and an average of 9 minutes on the second one. 233 synthetic images in total were created. Figure 4.4 shows a sample of the synthetic images created.

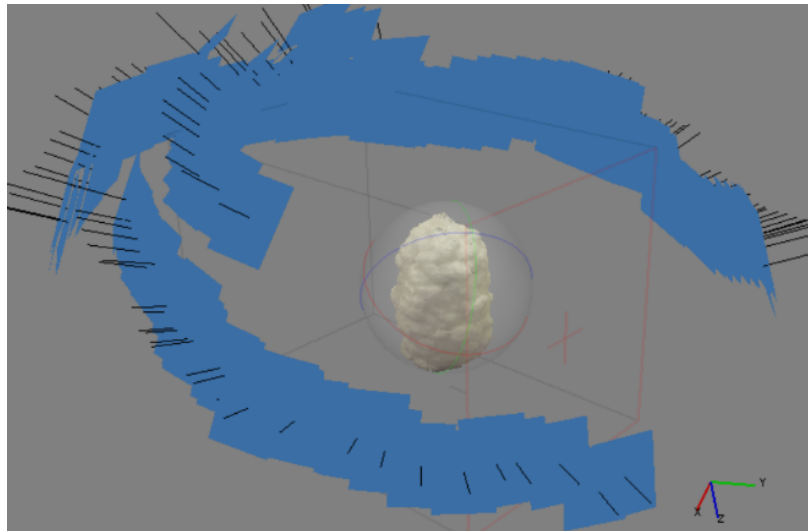


Figure 4.3: 3D model of the hand-crafted PPM nest.



Figure 4.4: Sample of the synthetic images.

## Artificial Images

Artificial images are images of cotton structures formed to have the shape of a nest, attached to real pine trees. A 16 Megapixels camera was used in taking these images under different lighting conditions. A total of 363 artificial images were created. Figure 4.5 shows a sample of the artificial images.

The addition of synthesized 3D images, fabricated artificial images, alongside the real ones is employed to bootstrap the limited dataset of available real images. Dataset-A contains 1386 images including 232 synthesized images, 362 artificial images, and 792 real images. Figure 4.6 shows a sample image from each image type. Training and testing on Dataset-A produced good results, as shown later, however it is deemed inadequate for the application at hand where a significant number of PPM nests went undetected in certain cases, with several false positive detections obtained in other cases.



Figure 4.5: Sample of the artificial images.

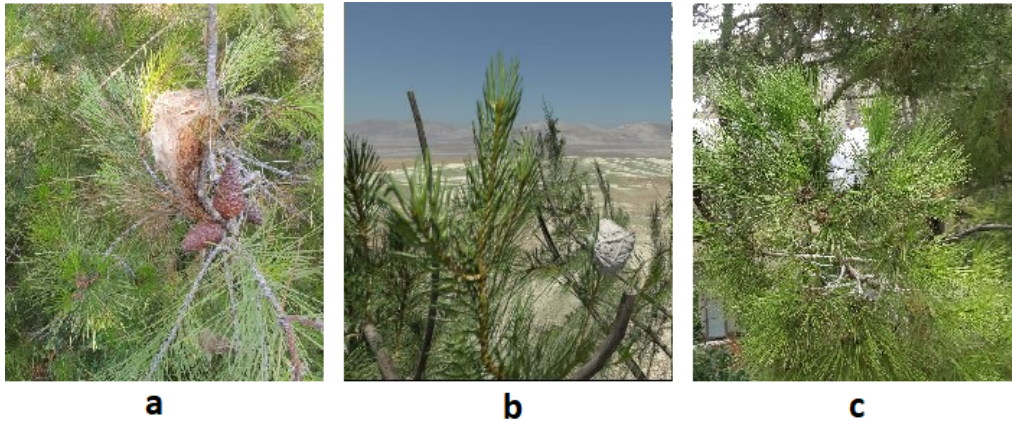


Figure 4.6: Samples of Dataset-A: real (a), synthetic (b), and artificial images (c).

### 4.1.2 Dataset-B

This dataset is composed of thermal images only. The FLIR<sup>®</sup> TG165 camera is used for acquiring thermal images of PPM nests. It is noted here that While collecting RGB images, the task is relatively simpler since images of nests located at high altitudes (at the top of the trees) can be captured using a drone’s camera. For this dataset, using a hand-held device, fewer images were collected reaching only 514 images. Although the dataset is not very large, training and testing on it provide a proof-of-concept of how features extracted from thermal images of the nests can improve detection accuracy. Samples of the thermal images collected are shown in Figure 4.7.

### 4.1.3 Dataset-C

With the aim of building a dataset composed of paired images, *i.e.* thermal and RGB images of the same PPM nest, the FLIR<sup>®</sup> C2 camera is utilized since it can take both images simultaneously. This dataset is used to train and test the proposed multi-stream CNN, and it is composed of 542 images (271 images of



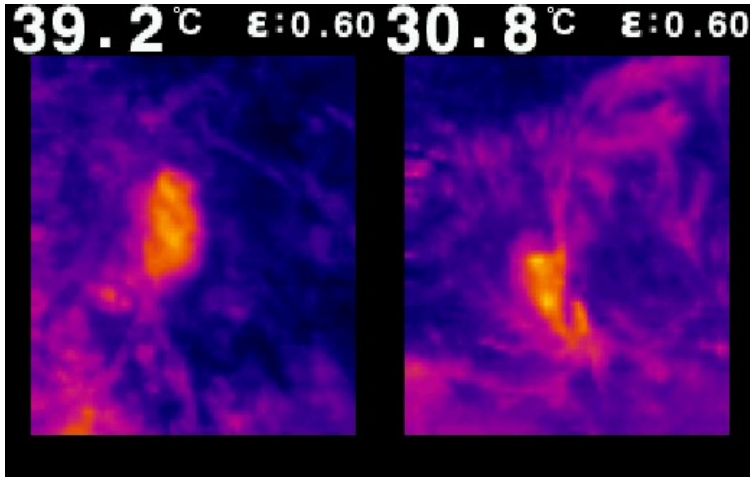


Figure 4.7: Samples of thermal images of PPM nests.

each image type). Sample of the paired images is shown in Figure 4.8.

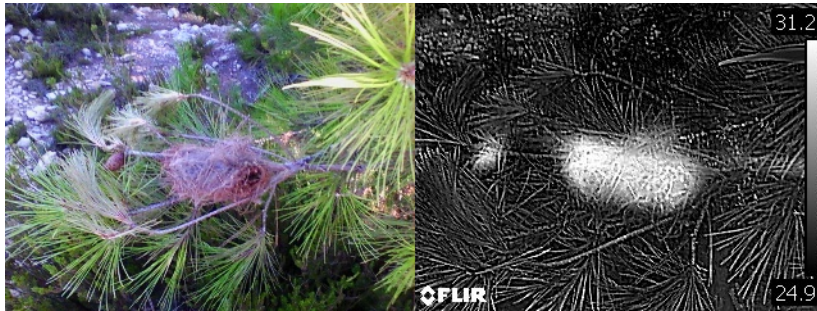


Figure 4.8: Sample of a paired image of the same PPM nest: RGB on the left, thermal on the right.

Dataset	Real	Synthetic	Artificial	Thermal	Total
A	792	232	362	0	1386
B	0	0	0	514	514
C	271	0	0	271	542

Table 4.1: Composition of the Image Datasets

#### 4.1.4 Data Collection

The data collection was performed over *Pinus Brutia* trees in several regions of the Mount Lebanon Governorate, southeast of Beirut. A considerable number of images were captured from the pine forests of "Kfarchima" town, which is located in the Baabda District of Mount Lebanon Governorate, and it is around 300 meters above sea level. Images were also acquired from "Dawhet Aramoun", which is a village in the Aley District of Mount Lebanon, Lebanon, lying to the east of Khalde and 22 kilometres away from Beirut, at an elevation level of around 450 meters. Some images and experiments were conducted in the "Abey" village as well, which is around 800 meters above the sea level, located in the Aley District of Mount Lebanon. Another set of images were acquired from the pine trees of "Ain Wazein" which is a village located in the Chouf District, in the Mount Lebanon Governorate. This village is around 1100 meter above the sea level.

These areas were visited to collect several datasets of images of PPM nests, between the end of February and June. The RGB images were collected at different times of the day, to cover the various backgrounds, brightness, color textures and shadings. For the thermal images, different times of the day were also tested, for example in the early morning, at noon, and even and near sunset. The silky material of the nest, in all of the aforementioned timings, has always proved to be capturing heat more than the surroundings, and having a different temperature than the trees' branches and leaves. It was intentional to target locations at multiple altitudes in order to study the effect of this variation in elevation on the PPM nests' characteristics in terms of size, color, and temperature.

## 4.2 Training and Testing

Several experiments were conducted to evaluate the performance of the various algorithms and networks, the obtained results are summarized in Table 4.2. Each algorithm was trained on both datasets A and B (RGB and thermal images). The datasets were split into 80% for training and 20% for testing, which is a widely used split for testing a detector’s accuracy, especially in cases where limited datasets are available.

In the first experiment, Dataset-A is used for training and testing the Faster R-CNN algorithm using the TensorFlow framework. As shown in Table 4.2, the system detected 215 nests out of the 285 nests (True Positives or TP) present in all images. The algorithm failed to detect 70 nests (false negatives or FN) since the algorithm missed detecting a nest even though it is indeed present within an image. On the other hand, the algorithm yielded 55 instances of false positives (FP) when it falsely detected a nest when it was not present within an image. Zero true negatives (TN) were obtained as the algorithm accurately detects the absence of a nest when it is not present in an image. To clearly illustrate this, Fig. 4.9 shows a sample of TP, FP, and FN cases. The overall accuracy of the detector in Experiment-1 is around 63% based on the following equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (4.1)$$

In Experiment-2, Dataset-B that only contains thermal images of PPM nests is used for training and testing using TensorFlow. The system detected 103 (TP) nests out of the 124 nests present in all of the thermal images. It failed to detect the presence of four nests (FN), and resulted in 21 instances of false positives and zero true negatives. The resultant accuracy of this detector is near 80%, which is

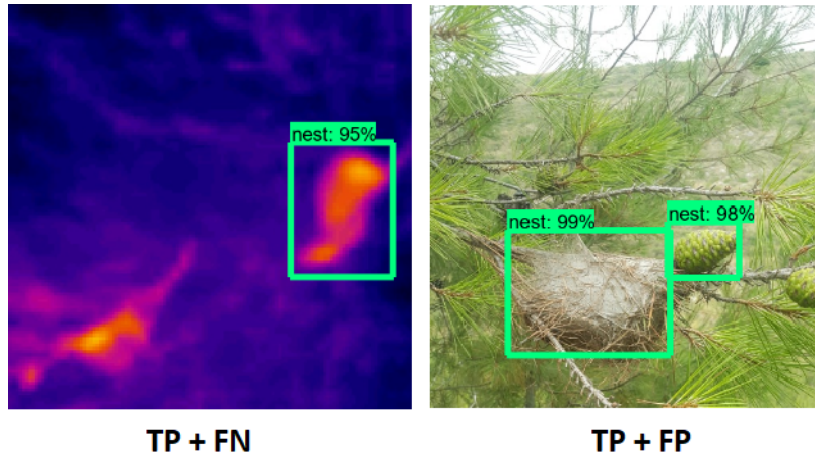


Figure 4.9: A Sample of detection results from Experiments 1 and 2. Thermal image on the left: the nest in the bottom left corner is not detected (FN) whereas the nest in the middle right is detected (TP), which is indicated by the green bounding box placed around the nest. RGB image on the right: the nest in the middle left corner is detected (TP) whereas the pine cone in the middle right is mistaken for a nest (FP).

significantly higher than the accuracy in Experiment-1 where only RGB images were used. This result illustrates the richness of features that are detected in thermal images as compared to RGB images, which motivates the idea of using both image types to boost the detector accuracy.

Experiment-3 is basically the same as Experiment-1 with the only difference being that training and testing on Dataset-A are performed via the YOLO algorithm instead of Faster R-CNN. The YOLO detector’s accuracy reached 51% where it successfully detected 176 (TP) nests out of the 285, thus missing 90 of them (FN). In Experiment-4, the YOLO detection algorithm is trained and tested on Dataset-B (thermal images only) and achieved an accuracy of 70%, where it successfully detected 92 nests out of 124 and missed the remaining 32 nests. Comparing the Faster RCNN architecture with YOLO while tested on the same dataset shows that Faster RCNN outperforms YOLO in terms of accuracy. On the other hand, YOLO had a faster computational speed when tested on

a video stream, which makes it a viable option for real-time applications, even though it provides faster detection rate at the cost of decreased accuracy. Table 4.3 presents this comparison between the two algorithms (YOLO vs. Faster R-CNN).

Table 4.2: Detection results of the different experiments and algorithms.

Experiment	No. of Nests	No. of Detections	TP	FP	TN	FN
1	285	272	215	55	0	70
2	124	107	103	4	0	21
3	285	254	176	78	0	90
4	124	100	92	8	0	32
5	78	58	22	36	0	56
6	78	91	58	33	0	20
7	78	72	72	0	0	6

In an effort to improve the accuracy of the detector, the network is first trained on both image types: RGB and thermal. In experiments 5-6, RGB and thermal images of different PPM nests, simultaneously taken by the same camera (FLIR<sup>®</sup> C2), are used to separately train each branch of the proposed dual-stream network. The purpose of conducting these two experiments is to determine the resulting calculated weights,  $W_{RGB}$  and  $W_{Thermal}$ , for initializing the two branches of the proposed multi-stream network in Experiment-7. The dataset includes 542 total images of PPM nests, where 156 images were used for testing purposes (78 images from each modality). The RGB branch is trained on the RGB images of Dataset-C and the thermal branch is trained on the thermal images of Dataset-C. As expected, training and testing each stream alone results in very poor accuracy, as shown in Table 4.4, where Experiment-5 (RGB branch) yielded 19% accuracy while the thermal detector in Experiment-6 reached 52%.

Having determined the initialization weights of each branch, Dataset-C that contains paired RGB and thermal images of the same PPM nests is used in

Table 4.3: Comparison between Faster R-CNN and YOLO in terms of accuracy and computational speed on NVIDIA GeForce GTX 1060 GPU.

Experiment	Network	Accuracy	Frames per Second (FPS)
1	Faster R-CNN	63%	$\approx 10$
2	Faster R-CNN	80%	$\approx 10$
3	YOLO	51%	$\approx 60$
4	YOLO	70%	$\approx 60$

Experiment-7 to train the multi-stream CNN. Training required approximately one hour on the used GPU (NVIDIA GeForce GTX 1060), and an accuracy of 92% was achieved during the testing phase. 72 nests out of 78 were successfully detected (TP), 6 nests were missed (FN), and no false positive instances. The obtained results demonstrate the accuracy of the proposed network architecture and the effectiveness of the combined learned features. Figure 4.10 shows samples of the detection results of the RGB branch alone (a), the thermal branch alone (b), and the multi-stream network with both image spectra. In general, the behavior of the detection algorithm is fairly represented by the image samples shown in Fig. 4.10. The RGB network branch, initialized with the RGB weights obtained in Experiment-5, is tested on the RGB version of the nest image, which does not successfully detect the nest in the two images shown in Fig. 4.10(a). In Fig. 4.10(b), the thermal branch, initialized with the thermal weights obtained in Experiment-6, is tested on the thermal version of the nests in (a). In both images (top and bottom) of column (b), the nest is successfully detected but with a false positive (FP) detection. In Fig. 4.10(c), the multi-stream CNN accurately detects the nest without any FP cases, which demonstrates the accuracy and reliability of the proposed architecture that benefits from enhanced learning due to a richer features extraction from both image spectra.

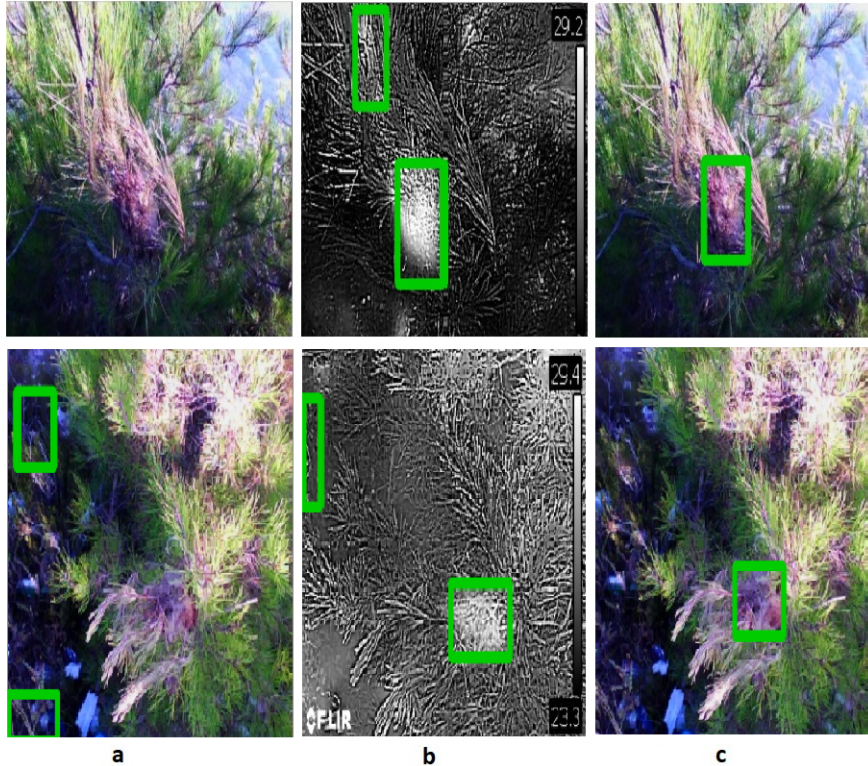


Figure 4.10: Samples of the detection results in Experiment-7. Each row (top and bottom) has three image types of the same nest: RGB on the left, thermal in the middle, and overlaid RGB and thermal on the right. The columns show sample detection results: **(a)** shows the detection results of training the RGB branch alone (single channel) and tested on the RGB image version of a nest, **(b)** shows the detection results of training the thermal branch alone (single channel) tested on a paired thermal image of the same nest, and **(c)** shows the detection results of the multi-stream network tested on the overlaid images of the nests in **(a)** and **(b)**.

### 4.3 Experimental Setup

A main challenge, which was encountered while testing this detector was lacking the ability to access the FLIR<sup>®</sup> C2 camera for triggering the image capturing process, thus lacking the ability to acquire both image types (RGB and IR) in a synchronized manner from its built-in dual cameras, that is, to capture the identical scene of the PPM nest from both sensors. Therefore, a separate RGB

Table 4.4: Performance comparison of different detection algorithms upon training and testing using an 80-20% split.

Experiment	Network	Algorithm	Dataset	Accuracy
1	Single-Stream	Faster R-CNN	A (RGB)	63%
2	Single-Stream	Faster R-CNN	B (Thermal)	80%
3	Single-Stream	YOLO	A (RGB)	51%
4	Single-Stream	YOLO	B (Thermal)	70%
5	Single-Stream <sup>a</sup>	Faster R-CNN	C (RGB)	19%
6	Single-Stream <sup>b</sup>	Faster R-CNN	C (Thermal)	52%
7	Dual-Stream	Faster R-CNN	C (Both)	92%

<sup>a</sup> Represents the RGB stream of the proposed network, trained and tested on the RGB images of Dataset-C alone.

<sup>b</sup> Represents the thermal stream of the proposed network, trained and tested on the thermal images of Dataset-C alone.

camera is mounted on-board the drone.

Furthermore, and in order to localize the nest, a GPS module is needed. So the system to be used for validation experiments should include an RGB camera and a thermal camera to feed the multi-channelled network mentioned in section 3.1.2, and a GPS module to localize the detected nest.

Accessing these three different data streams, two synchronization issues had to be solved. The first one was represented by having two different cameras (RGB and Thermal), each having a different sight angle. The RGB camera is the Raspberry Pi Camera Module V2 which has a 48.8 degree field of view (FOV), and the thermal camera is the FLIR<sup>®</sup> C2 which has a 45 degree FOV. The other issue was having two video streams recorded at different frame rates, and another type of data which is the GPS data acquired separately as well.

For solving the first issue, it required a geometric transformation to warp the RGB image (wider sight angle) and bring it into the coordinate system of the thermal image, thus achieving full synchronicity. Pairs of control points are manually identified between the two images, which are then used to infer an affine



geometric transformation bringing the RGB image into the coordinate system of the thermal version.

The software-based trigger solution adopted from literature was applied by writing a Python code which runs three sub-processes in parallel. These sub-processes or Python codes are represented by one triggering the RGB camera for a shot, the second is triggering the thermal camera (FLIR<sup>®</sup> C2) for a shot, and the third capturing a reading from the GPS. Figure 4.11 illustrates the setup and the equipment used in conducting these experiments. The drone used was the Parrot Bebop 2. Figure 4.12 shows the Bebop 2 assembled with FLIR<sup>®</sup> C2 camera.

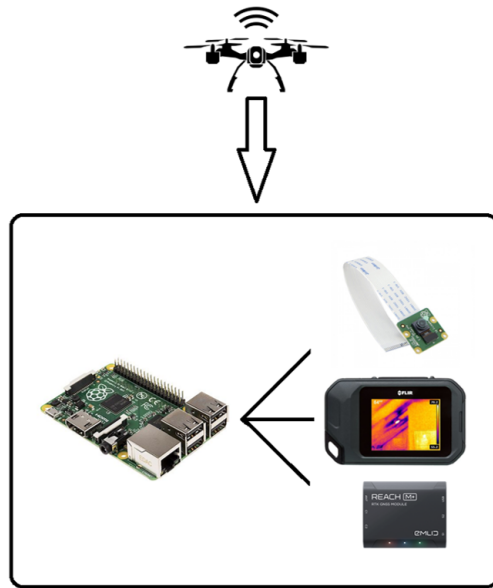


Figure 4.11: Schematic of the setup used for testing the detector and synchronizing the data during the experiments. A Python script is executed on a Raspberry Pi to synchronize the data streams of the three sensors: RGB camera, thermal camera, and GPS.

The GPS used for the experiments was an advanced RTK-GPS which is repre-

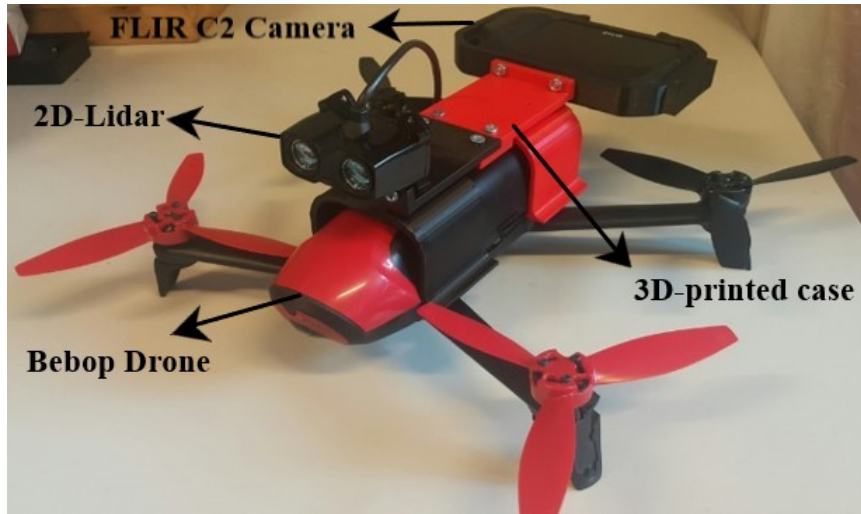


Figure 4.12: Bebop drone with FLIR<sup>®</sup> C2 camera mounted on board.

sented by two GPS modules working together where one is the base (stationary) and the other is the rover (on the drone) as illustrated in Figure 4.13. After the base reaches a minimum error, its reading is set as the base coordinates, and the rover starts getting input correction from the base. When fixating its signal, it reaches an accuracy of few centimeters.

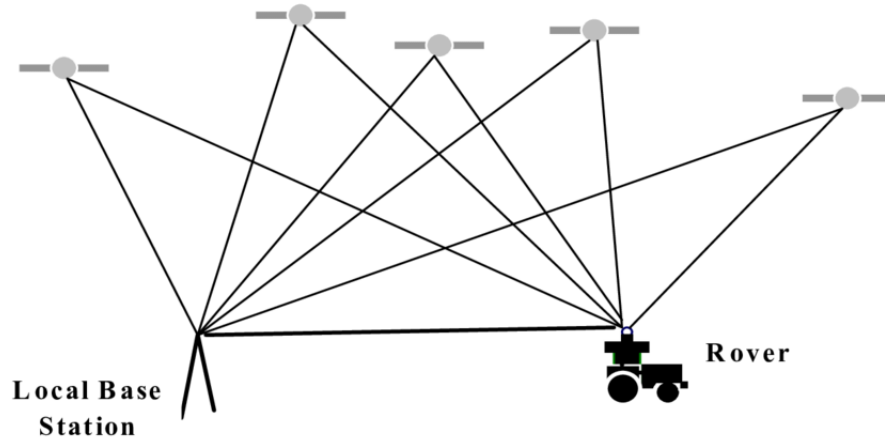


Figure 4.13: RTK-GPS methodology schematic [57].

The setup shown Figure 4.12 in was utilized in an experiment represented by hovering over two 'X' markers to prove synchronization of frames. Figure 4.14

shows samples of synced frames. The two frame sequences were converted into videos and when played overlaid synchronization seemed to be successful.

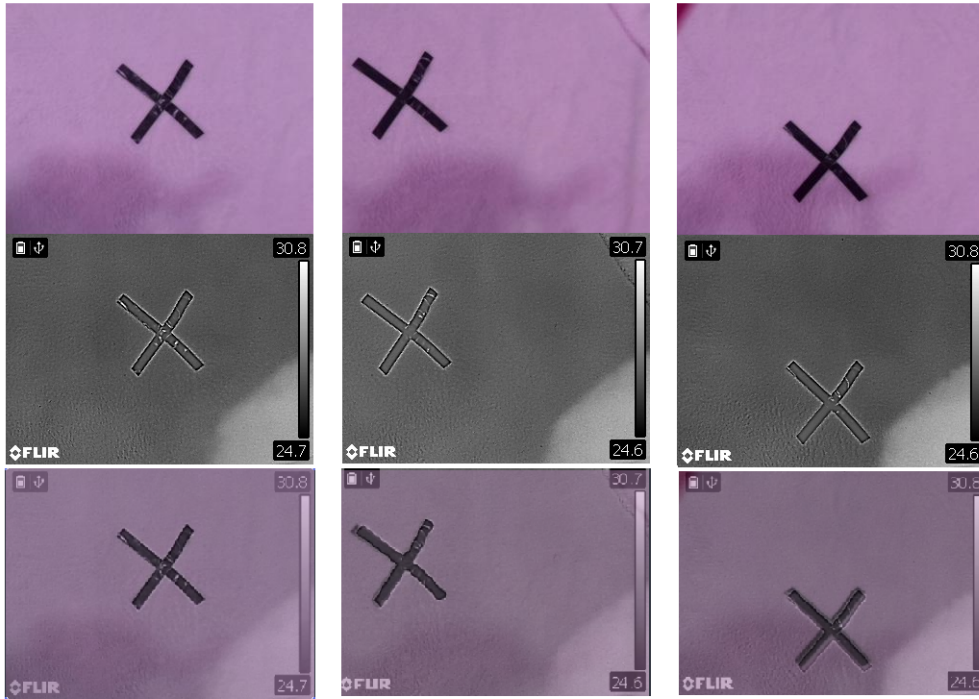


Figure 4.14: Captured RGB, thermal, and overlaid frames showing synchronicity.

Being able to synchronize the three data channels properly, the system was tested on actual trees as part of the validation experiments. These experiments were conducted using the setup shown in Figure 4.11, utilizing an octocopter instead of the Bebop 2 which has a payload of around one kilogram. A new 3D printed part was prepared for this drone, for the equipment to fit in place using SolidWorks. Figure 4.15 shows the octocopter assembled with the equipment installed onboard.

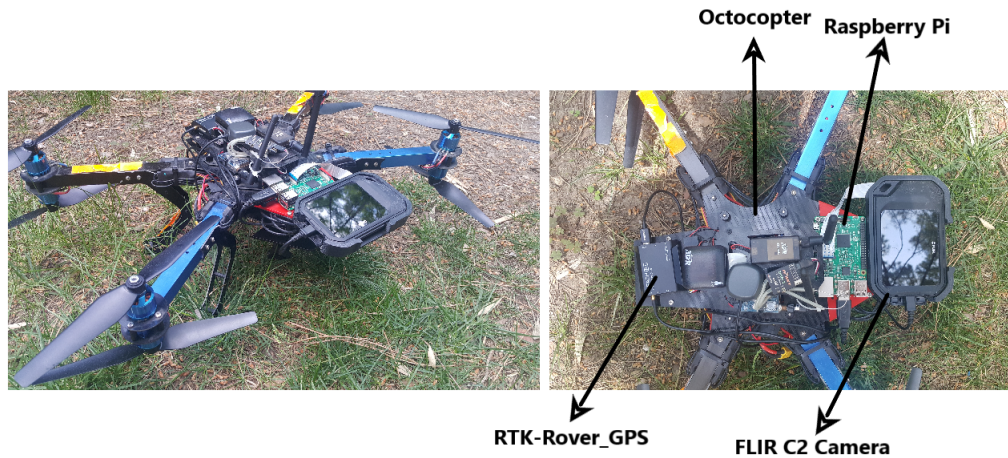


Figure 4.15: Octocopter used with the FLIR<sup>®</sup> C2 camera, the RGB camera, the Raspberry Pi micro-controller, and the RTK-rover-GPS module on board.

## 4.4 Validation Experiments

### 4.4.1 PPM Nest Detection

#### Primary Results

In the first set of the experiments, the detector was tested only on frame sequences which included nests. The main concern at that point was to check if the system was able to identify new nests on new trees, other than the ones that it has been trained on. This set of experiments was conducted on three trees; the first one containing two nests and the other trees containing one nest each. Detection was successful with the system able to identify all the four nests, each in several frames, as shown in Figure 4.16.

After achieving these results, the second stage of the experimentation started where the detector is tested on complete flights. Such experiments are needed to check if any false positive cases may be recorded, especially in frames which do not include a nest such as at the takeoff or while transitioning from one tree to

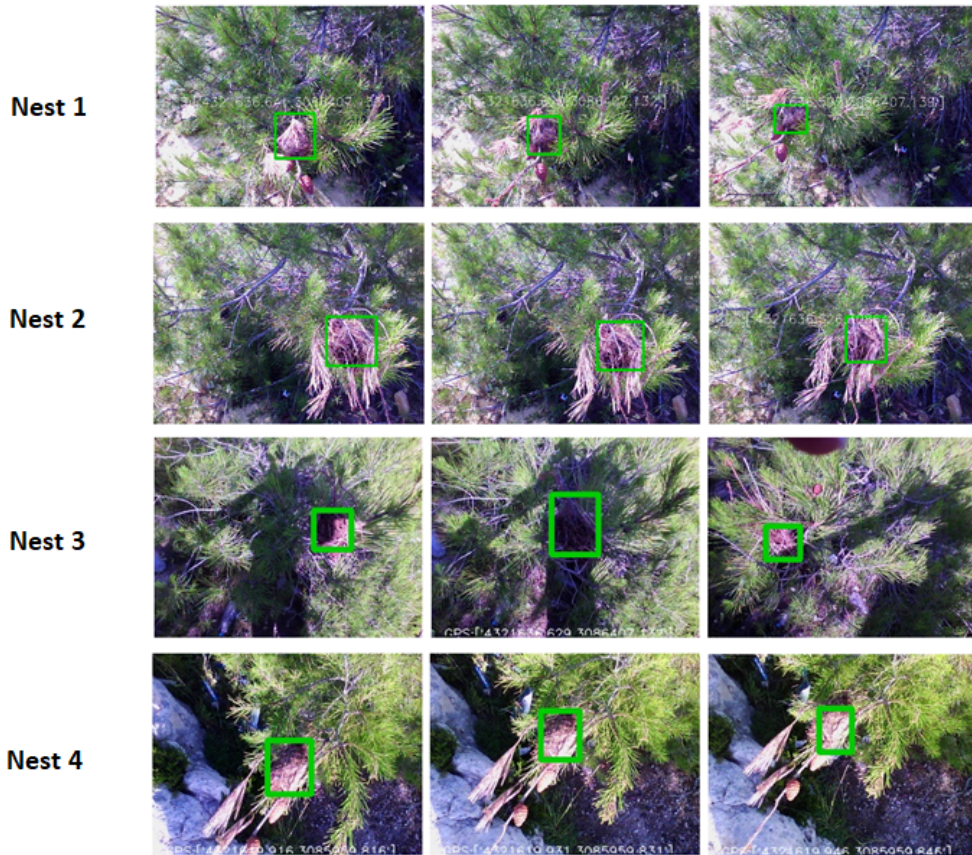


Figure 4.16: Three samples of the detection results from each nest of the tested nests. Each row corresponds to the same nest.

another.

The first experiment was conducted on a tree which contains one nest. Zero false positive (FP) cases were recorded in all the tested frames, however the detector failed to detect the nest in 7 frames (FN), successfully detected the nest in 5 frames, and successfully detected the absence of a nest when it is not present in 79 frames (TN). Table 4.5 shows the results along with the yielded accuracy and Figure 4.17 shows samples of the reported results.

The second experiment was conducted on three trees, each containing only one nest. In this experiment as well, all the flights' frames were tested. The first and



Figure 4.17: Samples of the results of Experiment-1 showing TN,TP, and FN cases.

the second nest were detected successfully even if not in all their corresponding frames, but the final nest was not identified in any of the frames including it. Zero false positive (FP) cases were recorded in all the tested frames, however the detector failed to detect the nest in 36 frames (FN), successfully detected the nest in 21 frames, and successfully detected the absence of a nest when it is not present in 152 frames (TN). Table 4.5 shows the results as well along with some samples of the results in Figure 4.18.



Figure 4.18: Samples of the results of Experiment-2 showing TN,TP, and FN cases.

The third experiment was conducted on one tree containing two nests. The first nest was detected successfully even if not in all its corresponding frames, but the second nest was not identified in any of the frames including it. In this

experiment, the first false positive case was recorded where a pine cone was detected as a nest in a very high confidence rate. In total, and after analyzing all the tested frames, 34 false positive (FP) cases were recorded in all the tested frames, however the detector failed to detect the nest in 73 frames (FN), successfully detected the nest in 16 frames, and successfully detected the absence of a nest when it is not present in 105 frames (TN). Figure 4.19 illustrates more the reported cases.

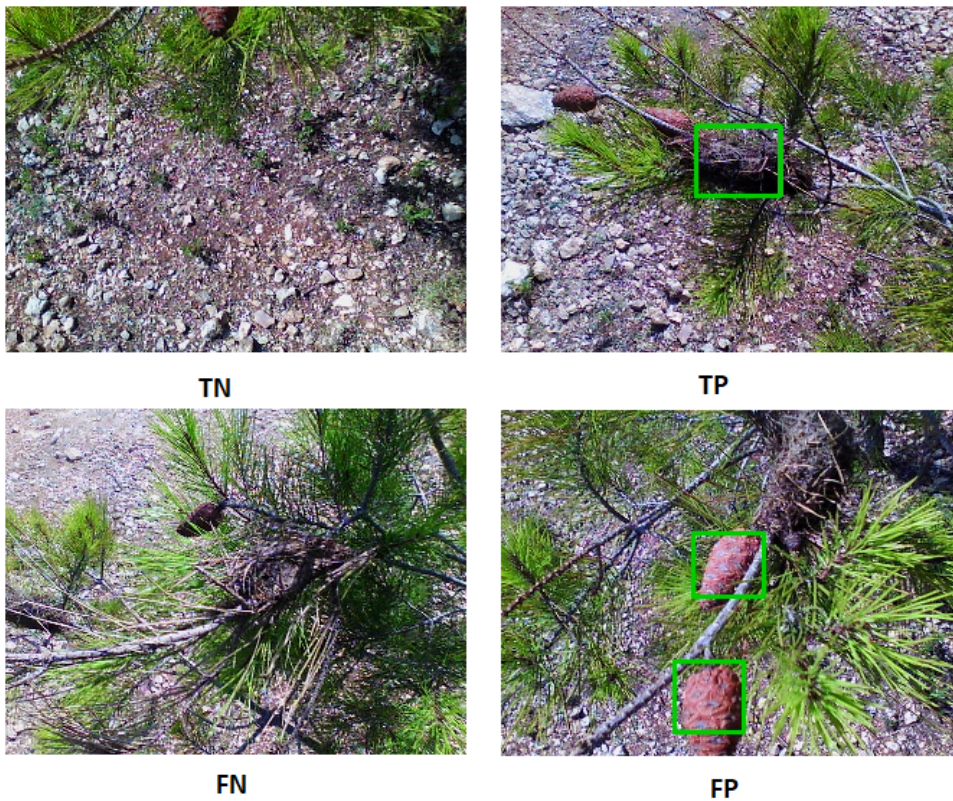


Figure 4.19: Samples of the results of Experiment-3 showing TN, TP, FN, and FP cases.

The fourth experiment was conducted on two trees, each containing two nests. The first nest was detected successfully even if not in all its corresponding frames, but the second nest was not identified in any of the frames including it. As shown

in Table 4.5, zero false positive (FP) cases were recorded in all the tested frames, however the detector failed to detect the nest in 20 frames (FN), successfully detected the nest in 3 frames, and successfully detected the absence of a nest when it is not present in 128 frames (TN).Figure 4.20 shows some samples of the results.



Figure 4.20: Samples of the results of Experiment-4 showing TN,TP, and FN cases.

Experiment	No. of Frames	TP	FP	TN	FN	Accuracy
1	91	5	0	79	7	92.3%
2	209	21	0	152	36	82.7%
3	151	3	0	128	20	86.7%
4	203	16	34	105	73	53.0%
5	209	48	0	152	9	95.7%

Table 4.5: Detection results of the different experiments conducted on actual trees.

After analyzing the results of the conducted validation experiments and taking into consideration that our detector was only trained on the initial dataset mentioned in section 4.1.3 ('Dataset C'- around 500 images), the detector did a good job identifying the new nests. However, the false positive case mentioned in Experiment-3 and missing some nests in all their corresponding frames was something that should be taken into consideration.

In the aim of testing the general hypothesis which says that the limited dataset trained on is the cause of such recorded cases, an update to the dataset was



applied. The frames captured during the aforementioned experiments which included nests were labelled and added to the dataset. Only the frames of Experiment-2 were not included. The new dataset (Dataset-C) contains now 1308 images (654 from each modality). The detector was trained again on this new dataset, and it was tested on the frames of Experiment-2 which are left out of the training process.

So as mentioned before, the fifth experiment is basically Experiment-2 but tested on the newly trained detector. The new detector did not miss the last nest on the third tree visited, and identified the nests in a higher number of frames than the old detector. The results in Table 4.5 illustrated more these findings. Zero false positive (FP) cases were recorded in all the tested frames, however the detector failed to detect the nest in 36 frames (FN), successfully detected the nest in 21 frames, and successfully detected the absence of a nest when it is not present in 152 frames (TN). Figure 4.21 shows the compared results between the old and new detector at several instances.

The numbers show that the performance was boosted where the number of FN cases dropped and switched into being TP cases, which means being able to capture the nest now in a higher number of frames than before.

## **Feature Visualization**

Although training the detector on the updated dataset showed improved results, but still the approach was considered a black-box testing method. In order to better understand and interpret the results, some visualizations were performed to show the extracted features.

As shown in Figure 4.22, the corresponding visualizations of the nest features

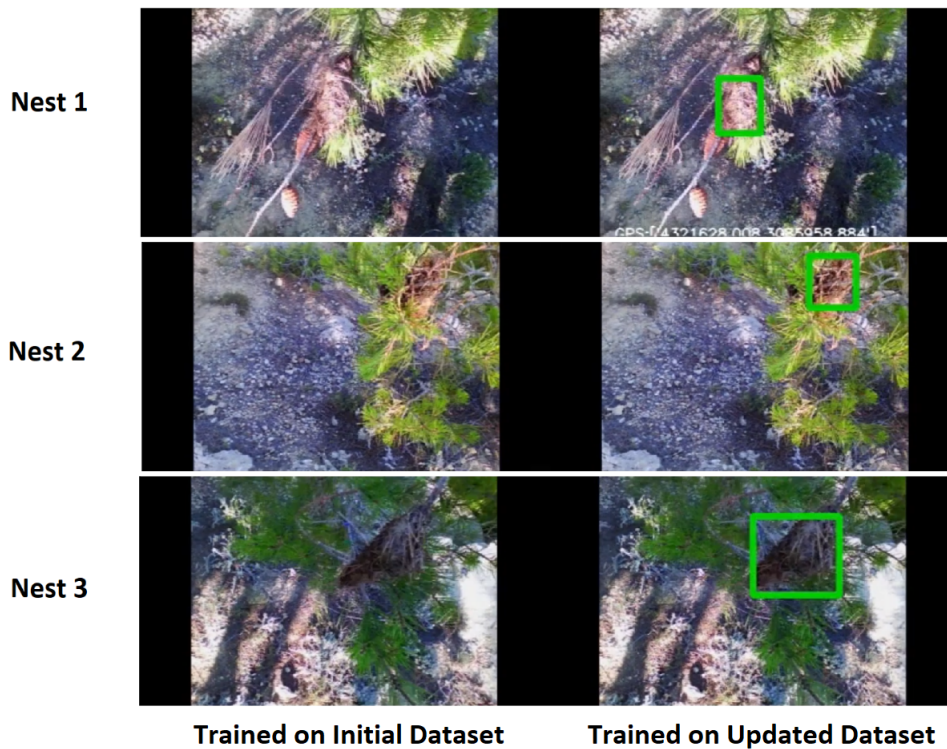


Figure 4.21: Sample of the results showing the performance of the detector trained on the initial dataset vs the new detector trained on the updated one in identifying the three nests of Experiment-2.

extracted from the RGB image are not very obvious, where features seem to be extracted not only for the nest compared by the features extracted from the thermal image. Although the visualizations are not perfect, but it definitely shows that the feature extraction process is better in the thermal image than the RGB image, which is reflected and illustrated in the results. I have to note that these visualizations refer to the “ResNet101” feature extractor implemented in the multi-Channeled CNN.

Figures 4.24 and 4.23 better interpret the aforementioned hypothesis by showing the features being fired for in the activation layers. It’s obvious that for the thermal image (Figure 4.23), features extracted in the convolution layers were

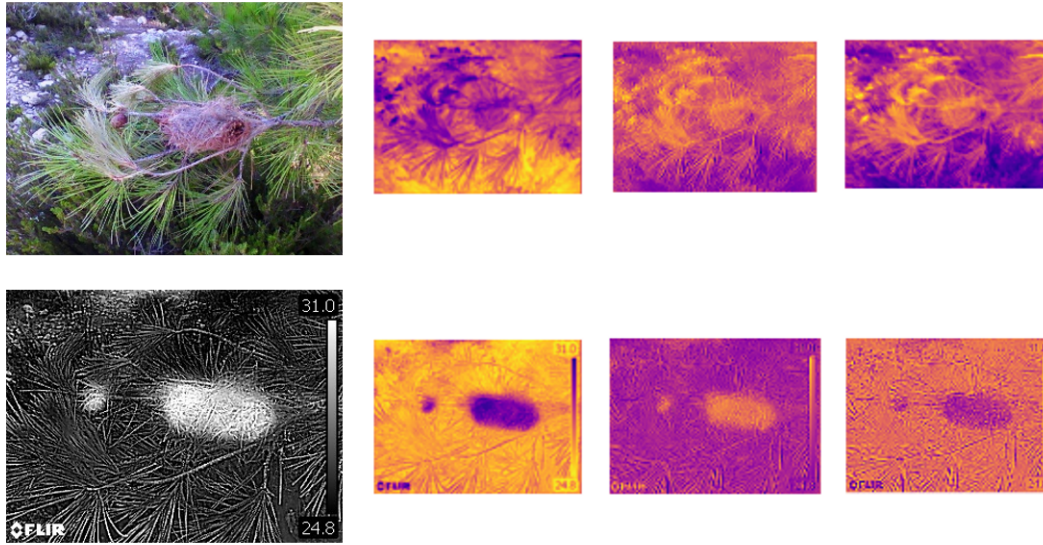


Figure 4.22: Output visualizations of three convolution layers of the ResNet101 feature extraction network.

perfect, since it only fired for the nest features in the activation layers; while in the case of the RGB image (Figure 4.24), the features extracted seem to be random and inaccurate, firing later for several objects in the activation layers.

### ResNet101 vs VGG16 Networks

The VGG16 network was trained and tested on the same dataset which the initial detector was trained and tested on before. As for the initial detector, the single streams (RGB and Thermal) were trained and tested each by its own before training the multi-channelled CNN to check the effectiveness of the new feature extraction network versus the old (Resnet). The two architectures are implemented using Pytorch [53]. Using an NVIDIA GeForce GTX 1060 GPU; the single stream training of each modality required an estimated time of three hours, and the multi-channelled network, initialized by the outputted pre-trained weights ( $W_{RGB}$  and  $W_{Thermal}$ ), required one hour of training to reach a minimum

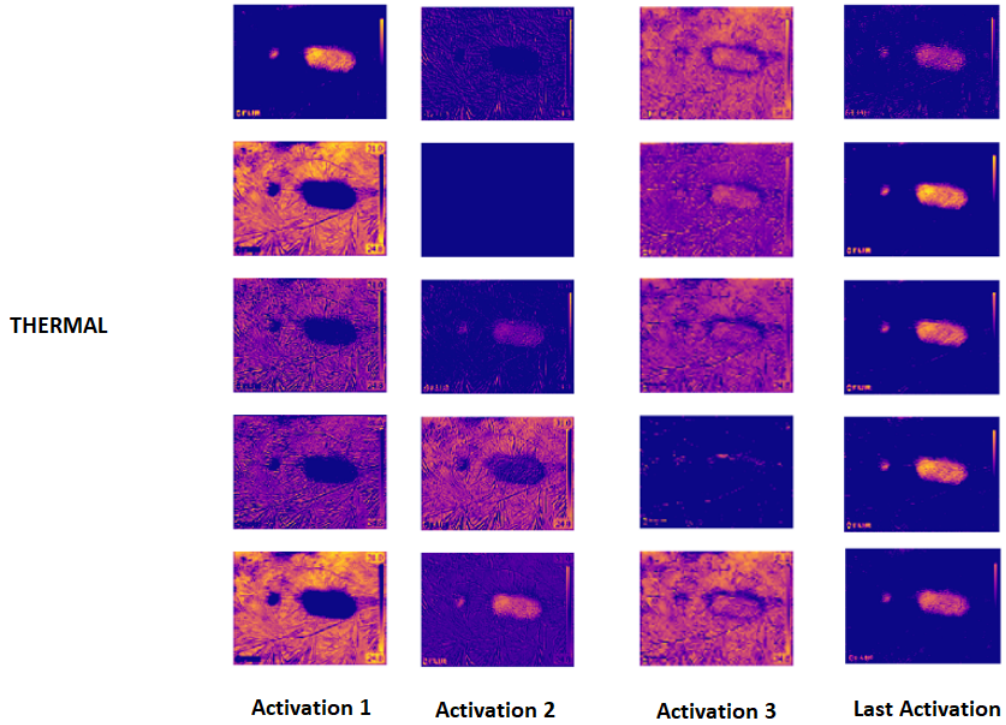


Figure 4.23: Output visualization of four ReLU-activation layers while testing a thermal image (ResNet).

loss.

$$Precision = \frac{TP}{TP + FP}. \quad (4.2)$$

$$Recall = \frac{TP}{TP + FN}. \quad (4.3)$$

$$F1Score = \frac{2 \times Precision \times Recall}{Precision + Recall}. \quad (4.4)$$

As shown in Figure 4.25, relying on the VGG16 feature extraction network showed improved visualizations where the features extracted from several convolution layers are more obvious compared to the ResNet network.

As shown in Table 4.6, accuracy, precision, recall, and F1 score - given in equa-

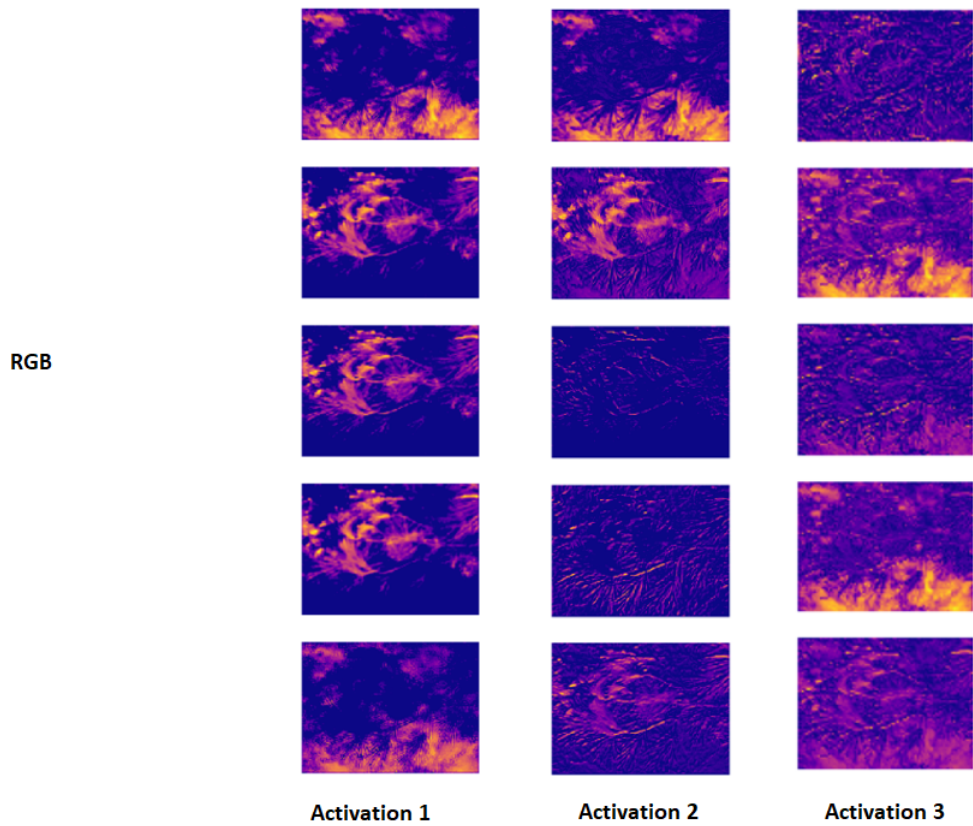


Figure 4.24: Output visualization of three ReLu-activation layers while testing an RGB image (ResNet).

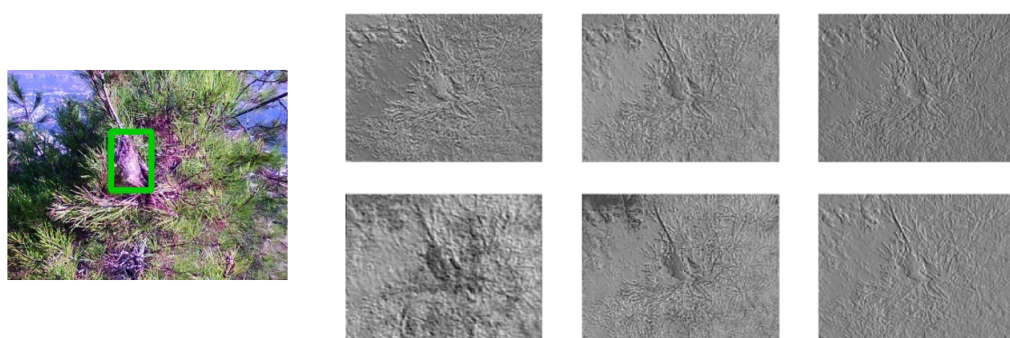


Figure 4.25: Output visualizations of six convolution layers of the VGG16 feature extraction network for an RGB image.

tions (4.1-4.4) - are used as evaluation metrics to assess the performance of each network . The VGG network outperforms the ResNet network when training the single streams alone, achieving a 95.9% accuracy versus only 81.9% for ResNet in the RGB case, and 88.4% accuracy vs 84.9% for ResNet in the thermal domain. Interpreting the results, ResNet recorded a higher number of false positive (FP) cases than the VGG 16, but almost the same number of true positives (TPs), indicating the improved feature extraction of the VGG network. When training the multi-channelled CNN, the ResNet101 feature extraction network outperforms the VGG network, showing that better feature learning is achieved when using both image types from the visual and thermal spectra. The visualization threshold for these experiments is set at 0.8, which indicates that only predictions that are 80% confident or above are visualized. A contribution lies in testing these two feature extraction networks in this multi-channel manner, with this performance comparison not being previously performed in such architecture.

Table 4.6: Detection results of the different experiments and algorithms.

	Stream	Nb. samples	TP	FP	TN	FN	Accuracy	Precision	Recall	F1 score
VGG 16	RGB	121	117	1	0	4	95.9%	99.1%	96.6%	97.8%
	Thermal	121	115	9	0	6	88.4%	92.7%	95 %	93.8%
	Multi	121	117	0	0	4	96.7%	100 %	96.6%	98.3%
ResNet101	RGB	121	118	23	0	3	81.9%	83.6%	97.5%	90 %
	Thermal	121	114	18	0	7	84.9%	86.3%	94.2%	90.1%
	Multi	121	121	0	0	0	100 %	100 %	100 %	100 %

## Final Experiments

Given its superior performance in multi-channelled streams, we rely on the ResNet101 feature extraction network and conduct four experiments to evaluate the detector’s performance in real-life conditions, with the results shown in Table 4.7.

Experiment-1 consists of a drone flight approaching only one nest that is successfully detected in all frames, recording zero false positive cases. Samples of

the detection are shown in Figure 4.26.



Figure 4.26: Three samples of the detections from Experiment 1.

In Experiment-2, the drone scans three trees, each containing one nest. The three nests are properly detected in 48 out of 57 frames, without recording any FP case. Three samples of the detection of each nest are shown in Figure 4.27.

Experiment-3 is represented by a flight scanning two trees, each containing one nest. The nests are faultlessly identified in all frames, also without recording any false detections. Some of the results are shown in Figure 4.28.

Experiment-4 involves scanning two nests on two different trees, detecting the nests in 142 out of the 144 frames, while recording a false positive detection of the same object in 12 consecutive frames. The average accuracy of all four experiments is 97%. Sample of the results are shown in Figure 4.29.

Table 4.7: Detection results of the four conducted experiments using ResNet 101.

Experiment	No. of Frames	TP	FP	TN	FN	Accuracy	Precision	Recall	F1 Score
1	46	40	0	6	0	100%	100%	100%	100%
2	209	48	0	152	9	95.7%	100%	84.2%	91.4%
3	121	27	0	94	0	100%	100%	100%	100%
4	187	142	12	33	2	92.6%	92.2%	98.6%	95.3%

## Evaluation Metrics

To have a more profound comparison and analysis of the obtained results, two widely used evaluation metrics were employed.

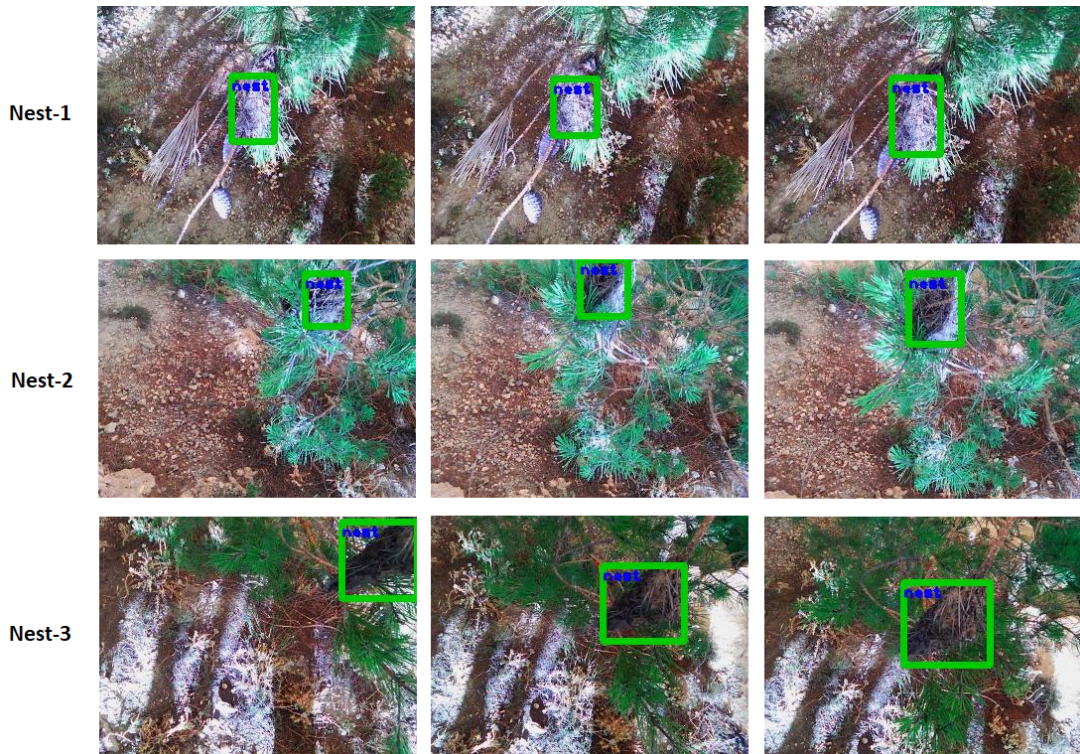


Figure 4.27: Three samples of the detections of each nest from Experiment 2.



Figure 4.28: Three samples of the detections of each nest from Experiment 3.

**ROC Curve:** An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a model at several thresholds. This curve plots two parameters:



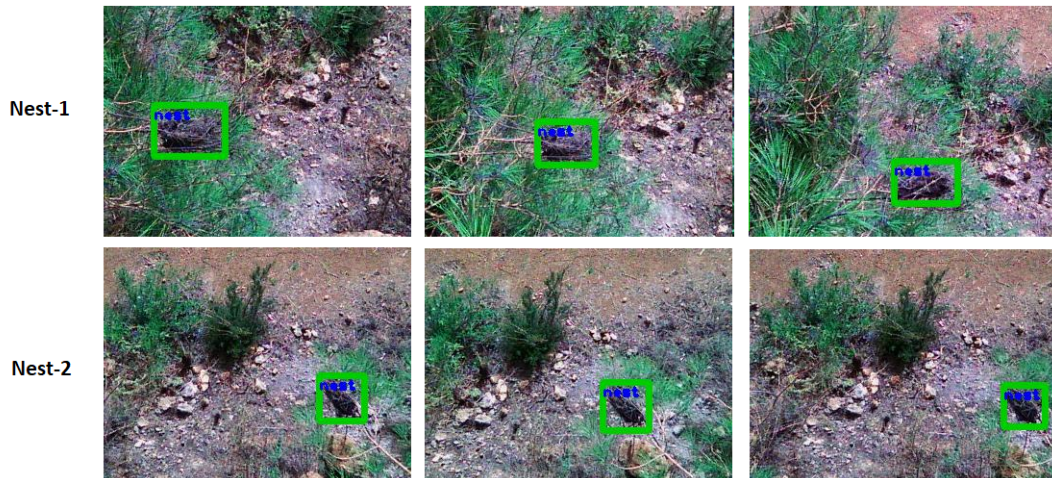


Figure 4.29: Three samples of the detections of each nest from Experiment 4.

$$1) TPR = TP / (TP + FN)$$

$$2) FPR = FP / (FP + TN)$$

An ROC curve plots TPR vs. FPR at different thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. Applying this metric on the conducted validation experiments, and testing the performance of the detector at several thresholds (0.05, 0.1, 0.2, 0.3, ..., 1) yielded a (TPR, FPR) at each value set. Table 4.8 represents the performance of the detector at the aforementioned thresholds while testing it on our original dataset. Tables 4.9 and 4.10 show the results of the same work applied on Experiments 2 and 4 from the validation experiments mentioned in section 4.4.1.

As shown in Table 4.8, a curve could not be generated obtaining the same (TPR, FPR) values at all thresholds and the reason behind that is lacking FN cases (missing nests) and TN cases (all images include a nest).

Figures 4.30 and 4.31 show the ROC Curves corresponding to Experiments 2 and 4 respectively. As shown in the figures, the ROC metric applied in this work

Table 4.8: Detector’s performance when tested on the original dataset (Dataset-C) at several thresholds for ROC metric.

Threshold	No. of Frames	TP	FP	TN	FN	TPR	FPR
0.1	121	121	3	0	0	1	1
0.2	121	121	3	0	0	1	1
0.3	121	121	3	0	0	1	1
0.4	121	121	2	0	0	1	1
0.5	121	121	2	0	0	1	1
0.6	121	121	2	0	0	1	1
0.7	121	121	2	0	0	1	1
0.8	121	121	2	0	0	1	1
0.9	121	121	1	0	0	1	1

Table 4.9: Detector’s performance when tested on Experiment-2 frames at several thresholds for ROC metric.

Threshold	No. of Frames	TP	FP	TN	FN	TPR	FPR
0.05	210	58	1	142	10	0.85	0.007
0.1	210	57	1	142	11	0.83	0.007
0.2	210	55	1	142	13	0.8	0.007
0.3	210	55	1	142	13	0.8	0.007
0.4	210	55	1	142	13	0.8	0.007
0.5	210	55	1	142	13	0.8	0.007
0.6	210	55	1	142	13	0.8	0.007
0.7	210	55	1	142	13	0.8	0.007
0.8	210	55	0	142	13	0.8	0
0.9	210	54	0	142	14	0.79	0
1.0	210	51	0	142	17	0.75	0

did not yield the proper curve. This method is widely used for evaluating the performance of classification models, not for object detection. Looking for the a more suitable metric for object detection, the Mean Average Precision (mAP) metric is adopted.

**Mean Average Precision (mAP)** is a popular metric in measuring the accuracy of object detectors like Faster R-CNN, SSD, etc. Average precision

Table 4.10: Detector’s performance when tested on Experiment-4 frames at several thresholds for ROC metric.

Threshold	No. of Frames	TP	FP	TN	FN	TPR	FPR
0.05	180	145	12	21	2	0.988	0.36
0.1	180	144	12	21	3	0.977	0.36
0.2	180	144	11	22	3	0.977	0.33
0.3	180	144	11	22	3	0.977	0.33
0.4	180	143	11	22	4	0.972	0.33
0.5	180	143	9	24	4	0.972	0.272
0.6	180	143	9	24	4	0.972	0.272
0.7	180	143	8	25	4	0.972	0.242
0.8	180	143	7	26	4	0.972	0.212
0.9	180	141	6	27	6	0.95	0.18
1.0	180	138	1	32	9	0.93	0.03

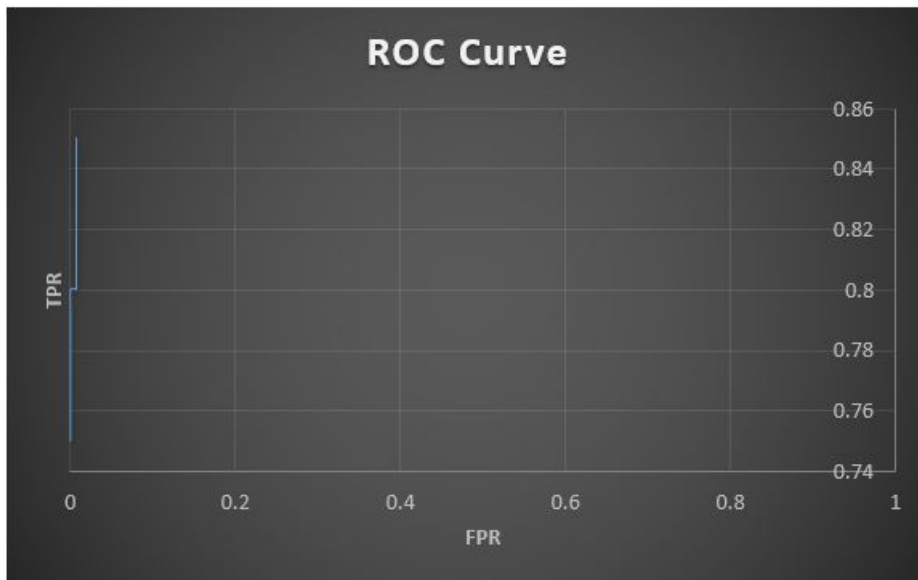


Figure 4.30: ROC Curve of Experiment-2. This curve is the plot of data corresponding to Table 4.9 where even at threshold=1.0, the detector still records TP cases, so it does not start from (0,0). Since the dataset does not include any TN cases (only images containing nests), the curve could not reach (1,1).

computes the average precision value for recall value over 0 to 1. The general definition for the Average Precision (AP) is finding the area under the precision-recall curve, where “Precision” and “Recall” are always between 0 and 1, and are

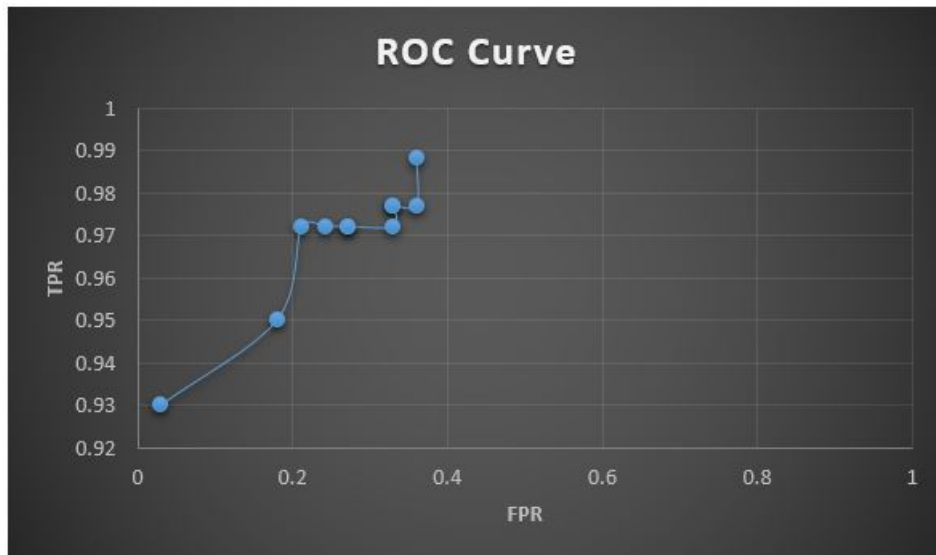


Figure 4.31: ROC Curve of Experiment-4. This curve is the plot of data corresponding to Table 4.10 where even at threshold=1.0, the detector still records TP cases, so it does not start from (0,0). Since the dataset does not include any TN cases (only images containing nests), the curve could not reach (1,1). The detector recorded a higher number of FP cases in this experiment than Experiment-2 at low threshold values, which then dropped at higher ones.

calculated using equations 4.2 and 4.3 . Therefore, AP falls within 0 and 1 also.

The mean average precision (mAP) metric, which evaluates the performance of each network over all visualization thresholds, was employed. The Intersection over Union (IOU) threshold is set to 0.5, which means that only predictions 50% overlaying with the ground truth label are deemed true positives, which is the default value used for several benchmark datasets [58]. The IOU concept is shown in Figure 4.32. This metric was applied to compare again the performance of VGG16 versus ResNet101 and for the validation experiments. A python code was prepared which automatically gives the mAP of the tested frames, given that all the tested frames should have a ground truth. The images of the dataset were already manually labelled to train and test the deep learning models on it, but the validation experiments' frames needed to be labelled as well, for a proper

evaluation.

Evaluating and comparing the performance of the VGG16 feature extraction network and the ResNet101 was done again, this time relying on the mAP. The results are captured in Table 4.11. The VGG16 network performs better feature extraction than ResNet, recording a significantly less number of FP cases thus a higher mAP (6 versus 48 in the RGB domain, and 22 versus 165 in the thermal domain), however both networks missed a similar number of nests (recording 4 versus 3 false negative cases (FN) in the RGB domain, and 6 versus 7 in the thermal domain). The dataset consists only of images which include a nest, so zero true negatives (TN) are of course recorded. Comparing the multi-channelled networks, ResNet again outperforms the VGG16 network, which performs better in single-stream trainings, achieving the highest mAP value of 98.3%.

Table 4.11: Mean Average Precision (mAP) of the two feature extraction algorithms tested on various image datasets (visualization threshold set to 0.5).

	Stream	Nb. samples	TP	FP	TN	FN	mAP
VGG 16	RGB	121	118	6	0	3	97.02%
	Thermal	121	115	22	0	1	94.35%
	Multi	121	116	9	0	1	94.2%
ResNet101	RGB	121	112	48	0	0	90.95%
	Thermal	121	112	165	0	0	83.71%
	Multi	121	119	5	0	0	98.31%

For the validation experiments, the frames of all four experiments were evaluated, and it yielded a 98.4% mAP. It is noteworthy to mention that a similar mAP value was obtained in the experiments as compared to the theoretical value (achieved from the image datasets), which demonstrates the detector’s reliability in real-life conditions.

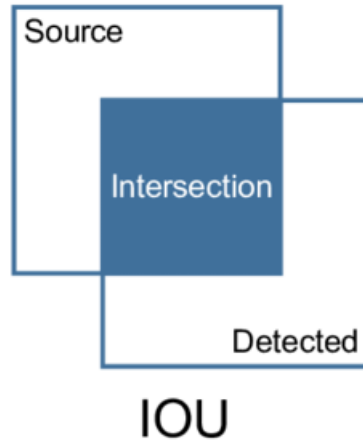


Figure 4.32: Concept of the Intersection over Union (IOU).

#### 4.4.2 PPM Nest Tracking

For single object tracking applications, accuracy is a critical metric that is commonly used for performance evaluation. Accuracy in object tracking is represented by calculating the IOU between the bounding box that is predicted by the tracker and the ground-truth (actual location of the object in the image). Accuracy is the average of all IoU values over the frames of the sequence. Tracking of the detected PPM nests is achieved in the four conducted experiments mentioned in section 4.4.1, where each nest is given a unique ID and FP are disregarded.

The tracking accuracy in each of the conducted experiments is captured in Table 4.12. Tracking is performed on the frame sequences of the validation experiments by setting the IOU threshold to 0.5, and setting the number of hits to 5. This implies that predictions above 50% of the actual detection are deemed successful, and attaining five consecutive correct predictions of a new tracked object gives it an ID.

Figure 4.33 shows three samples of the results of the tracking performance during the first experiment that includes one nests, where the detected nest is

given the same ID in subsequent frames achieving a tracking accuracy of 68.5% over all the frames' sequence. The tracker didn't lose the target anytime (i.e the IoU falls below a given threshold) throughout the entire sequence, yielding a continuous tracking

Figure 4.34 shows a sample of the results of the tracking performance during the second experiment that includes three nests, where each detected nest is given the same ID that is appropriated tracked as it appears in subsequent frames. In this experiment, an accuracy of 74.3% was achieved tracking three nests in the frames' sequence. Each nest showing up in several frames was given the same ID, however the tracker lost the second nest once before tracking it again successfully.

Table 4.12: Tracking Results of the four conducted experiments using the Kalman Filter.

Experiment	No. of Frames	Accuracy
1	46	68%
2	209	74%
3	121	71%
4	187	64%



Figure 4.33: Tracking using the Kalman Filter method - Experiment-1

In the third experiment, a continuous tracking was performed giving the first and the second nests the proper IDs, achieving an accuracy of 71.1%. Figure 4.35 shows samples of the tracking performance on two detected nests. Figure 4.36 shows samples of the tracking performance during Experiment-4, tracking each

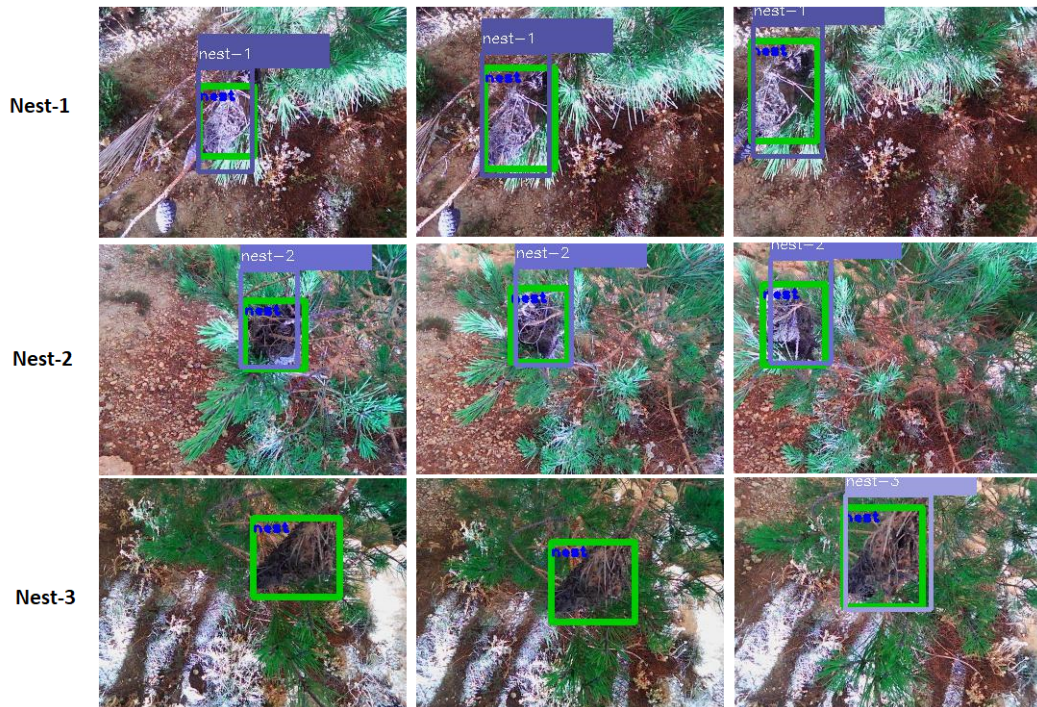


Figure 4.34: Tracking using the Kalman Filter method - Experiment-2.

of the two nests scanned in this experiment. In this experiment, the tracker lost the first nest once due to one wrong prediction before tracking it successfully in the rest of the frames, and tracked the second nest continuously. Both nests were given the proper IDs, as shown in the figure, with an accuracy of 64.8%.

In the fourth experiment, the detector records a false positive case during takeoff, which was detected in several frames. Figure 4.37 shows the performance of the two tracking methods illustrated before, where the first method gave the FP detection an ID number '1', and tracked this FP in the subsequent frames; giving the first detected nest later on an ID number '2'. While for the KF, and as shown in the figure, the tracker disregarded this FP case, and gave the ID number '1' to the first detected nest directly.



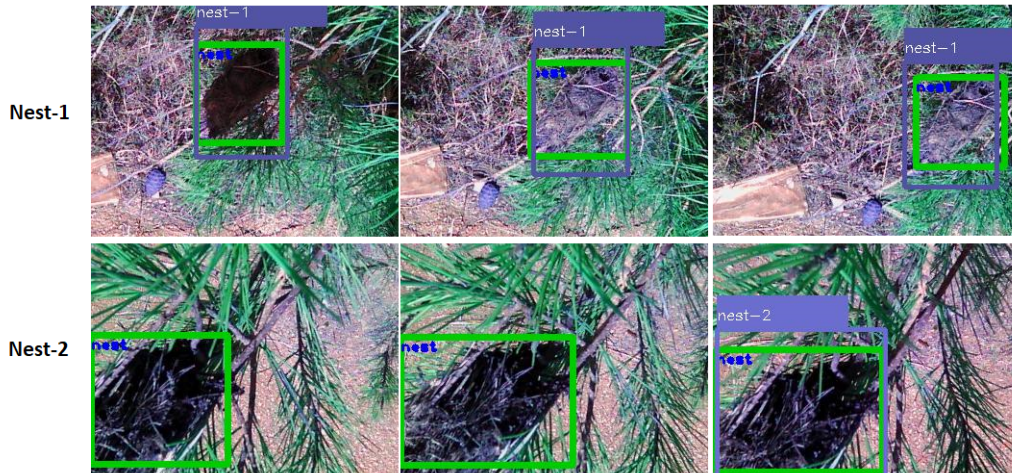


Figure 4.35: Tracking using the Kalman Filter method - Experiment-3.

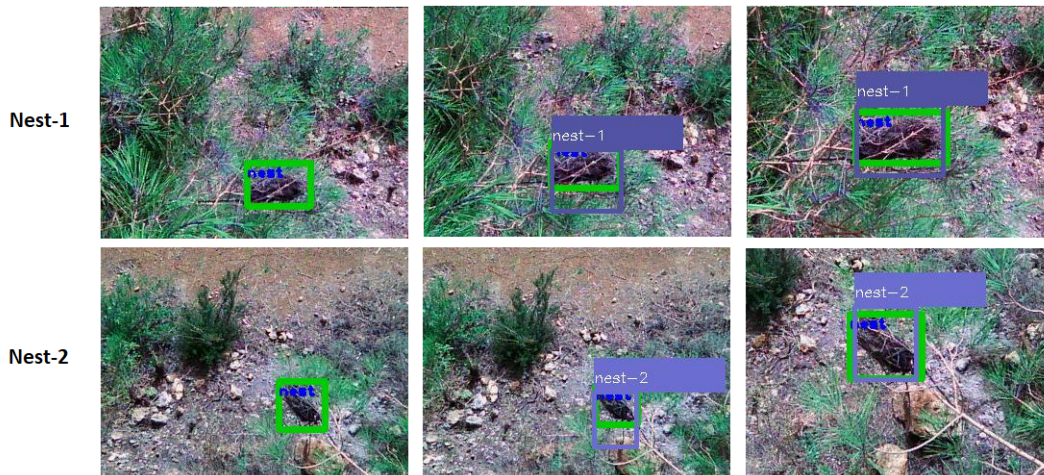


Figure 4.36: Tracking using the Kalman Filter method - Experiment-4.

### 4.4.3 PPM Nest Geo-Localization

As shown in Figure 4.38, experiments start by configuring the RTK-GPS in the field. The base station (RTK-base-GPS module) is configured (initialized and calibrated) and positioned at a location that receives the strongest signal from the satellites to attain minimal positioning error. In each of the four conducted experiments, the base is set to reach an accuracy of approximately 10cm. The rover-RTK-GPS module is mounted on board of the drone and receives input

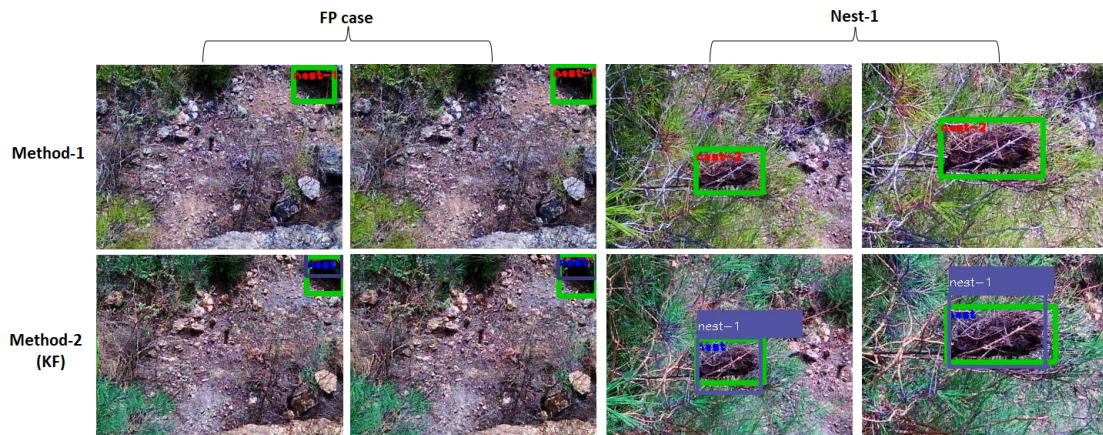


Figure 4.37: Comparing the performance of methods 1 and 2 on frames from Experiment-4.

corrections from the base via long range (LoRa) radio communication, achieving an accuracy of  $1cm$  given an interrupted signal during the entire flight.

To evaluate the performance of the proposed geo-localization scheme, the devised algorithm is tested in experiments where the drone approaches a nest from the top, and the algorithm estimates the nest's position before reaching it, whenever a detection occurs. Manual ground truthing is performed by taking advantage of the accurate RTK-GPS readings, where the user manually flies the drone and vertically approaches each nest in the trees being scanned. When the drone becomes exactly on top of the approached nest, i.e. when the nest appears in the center of the downward-pointing camera stream, the user records the position of the drone acquired from its on-board RTK rover-GPS module, which is stored as the nest's ground-truth position. This process is repeated for all trees being scanned in the experiments to record the ground-truth of all present nests. In scanning mode, the drone is flown on top of trees at low speeds that require small pitch angles (not to have a tilted camera view), and its flight data (RGB and thermal cameras, RTK-GPS) are recorded. The acquired data

are processed offline to detect and localize the nests, where the reported positions of the detections are compared to the ground truth values that were manually recorded beforehand.

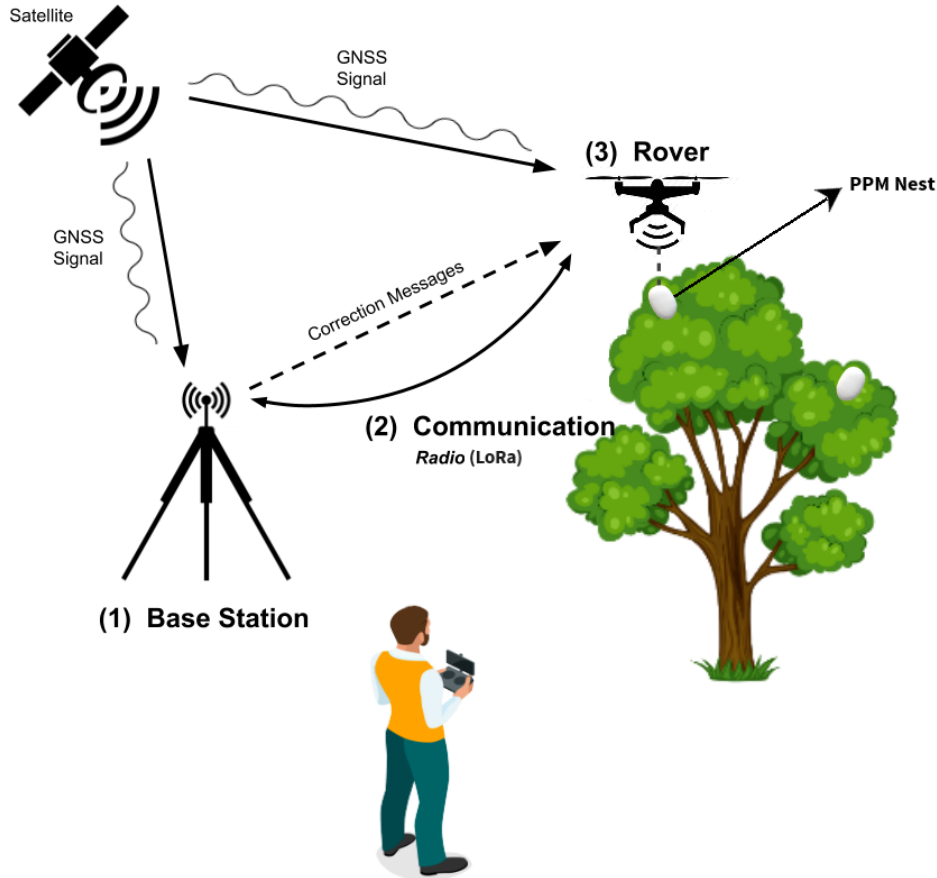


Figure 4.38: Schematic showing the experimental setup. The RTK-Base-GPS module is configured in the field receiving the satellite signals and sending corrections to the RTK-Rover-GPS module on board of the drone. The user drives the drone manually scanning the tree for PPM nests.

Figures 4.39 through 4.42 show the results of experiments 1,2,3 and 4 respectively, with the drone approaching each nest and estimating its position. The GPS data being captured is in the Longitude/Latitude/Height (LLH) format, but it has been converted by the GPS estimation code into the Universal Transverse Mercator-UTM format as shown in the figures of the results, where the values

are converted to meters.

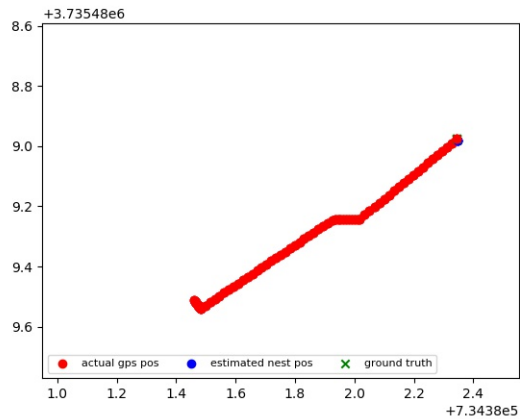
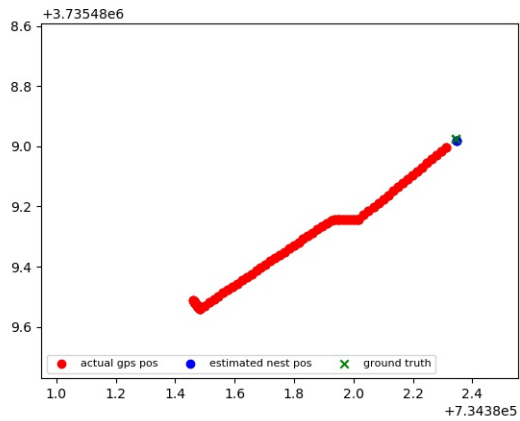
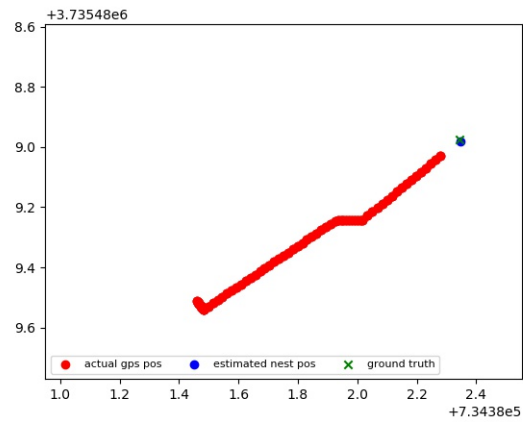


Figure 4.39: Three instances from Experiment-1 showing the estimated position of the nest at each corresponding detection showing the drone's moving position (red), the nest's ground truth position (green), and the estimated position of the nest (blue). Minimal error is observed between the true and estimated positions of the nest.

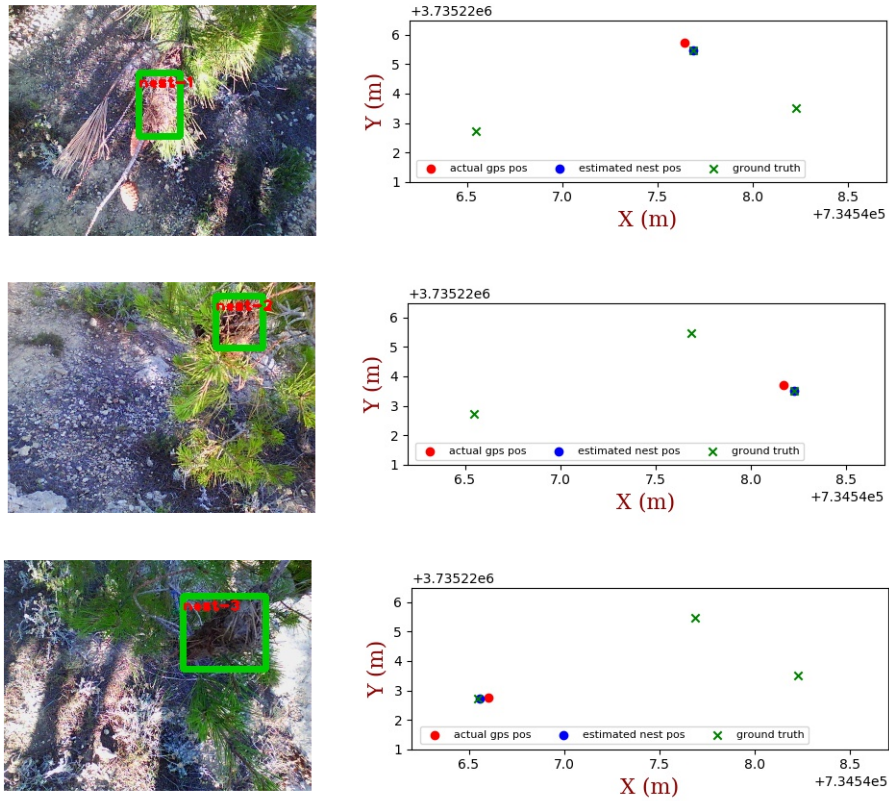


Figure 4.40: Localization result of Experiment-2 showing the drone’s moving position (red), the nest’s ground truth position (green), and the estimated position of the nest (blue). Rows 1,2, and 3 show the frame where the first, second, and third nests got detected for the first time respectively, and their estimated positions. Minimal error is observed between the true and estimated positions of each nest.

Table 4.13 shows the numerical results of four conducted experiments. Experiment-1 includes only one nest, where this nest showed up in 16 frames and the average estimation error versus its actual position (ground truth) is approximately 5 cm. Experiment-2 entails a drone flight over three nests, and the estimation process of the nests’ position was successful with an error not exceeding 10cm for each nest. In Experiment-3, the flight included two nests: the first nest showed up in 19 frames while the second nest showed up in eight (8) frames only. The average

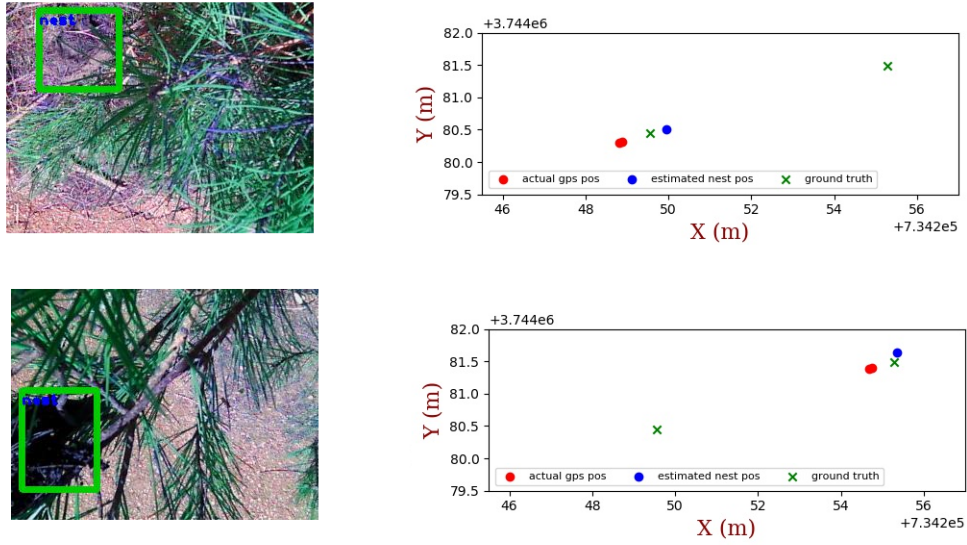


Figure 4.41: Localization result of Experiment-3 showing the drone’s moving position (red), the nest’s ground truth position (green), and the estimated position of the nest (blue). Rows 1 and 2 show the frame where the first and the second nest got detected for the first time and their estimated positions.

estimation error was approximately 24 cm and 12 cm for the first and the second nest, respectively. The fourth experiment entails detecting and estimating the positions of two nests, achieving accurate estimations and small average errors of approximately 7 cm from the ground truth, for each.

Table 4.13: Estimation of the nests’ positions in the four conducted experiments.

Experiment	Nest	Number of frames	Average Error
1	1	16	5 cm
2	1	19	2 cm
	2	30	2 cm
3	3	8	9 cm
	1	19	24 cm
4	2	8	12 cm
	1	84	7 cm
	2	59	7 cm

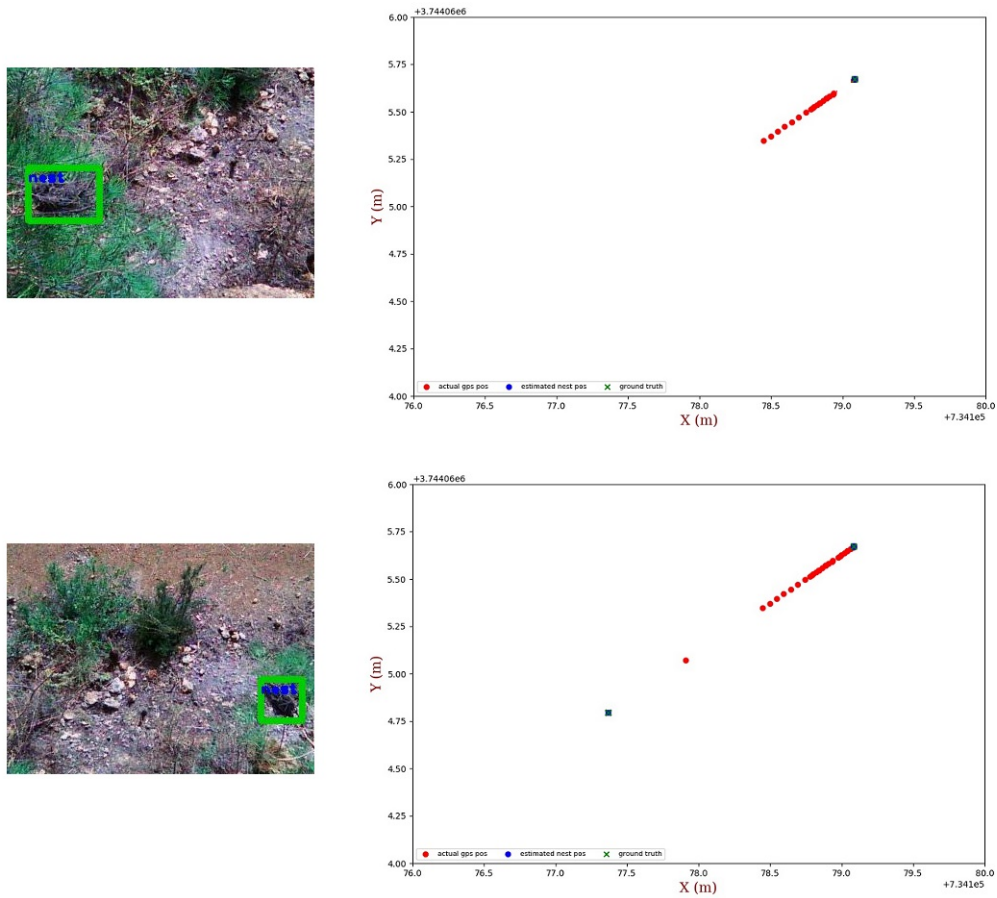


Figure 4.42: Two instances showing the estimated position of each nest when it got detected for the first time. The first row corresponds to the first nest, and the second row corresponds to the second nest. The drone's moving position (red), the nest's ground truth position (green), and the estimated position of the nest (blue).

## Output Maps

After performing accurate estimation of the nests' GPS positions, along with proper tracking, the desired output is yielded. The most accurate position to be reported of a nest is when the drone passes directly on top of it, which correlates to the drone's actual position. Thus, in the conducted experiments, the positions of nests passing at the center of the frame are directly associated with the drone's

position at that frame, without relying on the estimation. On the other hand, if a detected nest does not appear in the center of a frame, it is assigned the latest estimated position.

Figure 4.43 shows sample outputs of Experiment-2 (top) and Experiment-3 (bottom). The output plot of Experiment-2 shows the position of the three successfully detected nests by reporting back only three points assigned to each nest, indicating that tracking is successful. For Experiment-3 as well, a 2D map showing the two nest positions, which are detected during the drone's flight, is reported back.

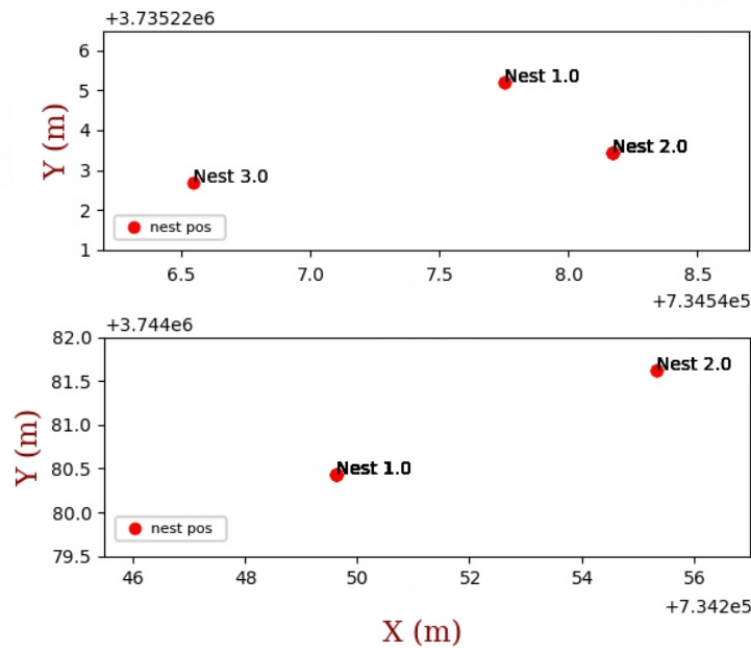


Figure 4.43: Result of Experiment-2 and 3 showing the final output as a 2D map.

### Orthomosaicing

Having a good detection system along with a proper localization method, a 2D plot showing the detected nests' locations is our output. A better visualization of the nest location is set to show the nests' positions on an image instead of a 2D



plot. Image mosaicing is considered the proper solution to do that, which stitches the frames together to form one complete image of the complete sight covered.

Such a process could be done using several commercial software programs such as “Reality Capture” [59], “Agisoft Photoscan” [56], “Pix4D mapper” [60]. Using these programs, acceptable results were obtained as shown in Figure 4.44.

Although the output shown in Figure 4.44 shows good visualization of the scene, but they are still not geo-referenced. Given that the flight’s frames are recorded synchronously, but independently from the GPS data, the captured frames are geo-tagged by adding each frame’s corresponding latitude and longitude positions to its metadata. This method is applied to Experiment-1, which yields the orthomosaic map shown in Figure 4.45, scaled by the proper GPS positions.

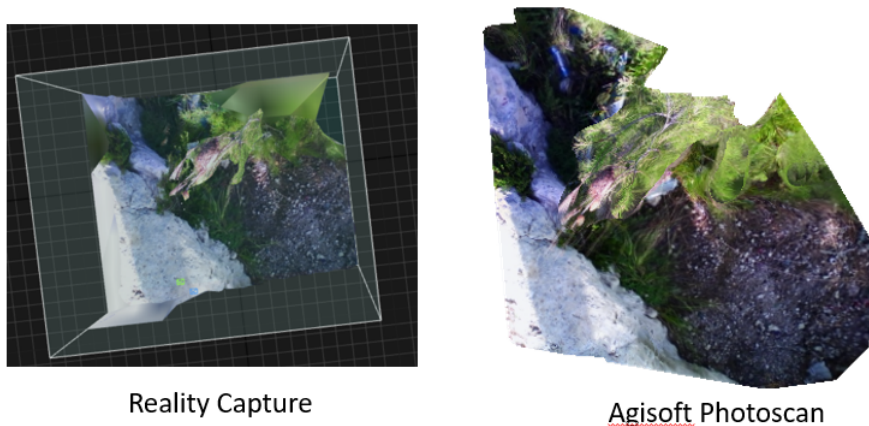


Figure 4.44: Mosaic Maps created from the frames of Experiment-1 using Reality Capture [59] and Agisoft Photoscan [56].

For better visualization of the complete surveyed site, another orthomosaic map is created. Lacking access to high resolution satellite images, frames of the experimented region are captured using our drone from high altitude (25 meters) in order to create our own high-resolution Birdseye-view image. These

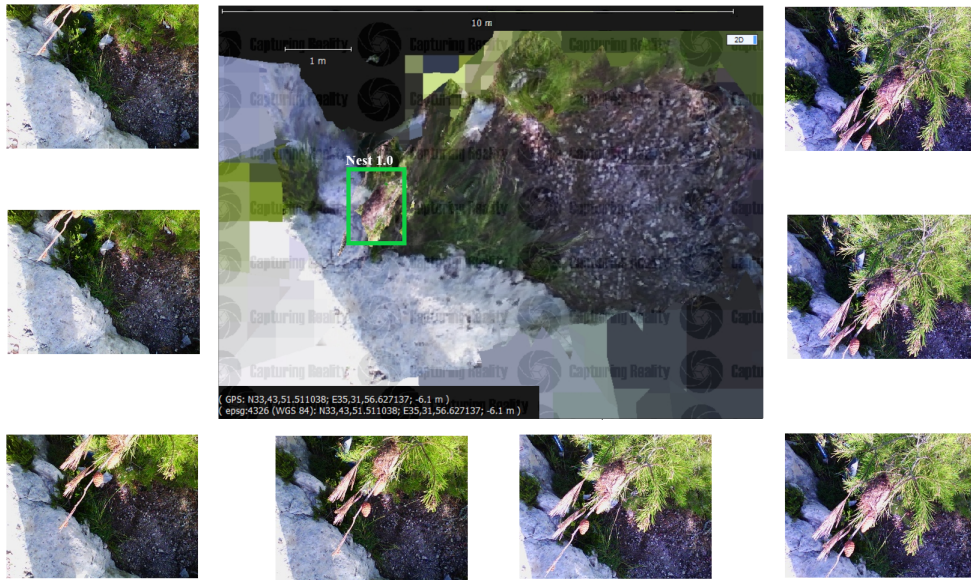


Figure 4.45: Orthomosaic map (center) created from Experiment-1 frames sequence (surrounding frames), with the GPS coordinates of the detected nest shown in the bottom left.

geo-referenced frames are imported into Pix4D mapper [60] to create this highly detailed orthomosaic image. Overlaying this image on Google Earth [61] and plotting the GPS positions of the three detected nests in Experiment-2 is shown in Figure 4.46.



Figure 4.46: Position of the three nests detected from Experiment-2 marked on our created orthomosaic image.

# Chapter 5

## Discussion

Given the gravity of the situation caused by PPM caterpillars, which are rapidly spreading [8] and destroying pine trees every season at large scales, a radical solution to eradicate is sought after in order to be applied as soon as possible. Any proposed system solution must feature a robust and accurate detection system, which identifies the presence of the moths before taking any measures of treatment. Options to detect PPMs include identifying the presence of their eggs, their caterpillars, or their nests. Detecting PPM eggs and caterpillars is technically challenging, thus tracking their nests is a more practical and feasible solution.

Measurement of defoliation in fields of pine trees due to PPMs is conventionally performed visually by forestry technicians on site. Visual examination is subjective and requires a lot of experience since there is minimal data available to validate the results. Using Computer Vision for the detection of PPM nests is a viable alternative to estimate the PPM defoliation in a tree. Such a system can also locate the nests within a tree environment, which enables targeting them with appropriate treatment options such as mechanical, chemical, biological, or other.

The goal of this research was to come up with a system which is able to detect a PPM nest properly, track it when it appears in several frames, and report back its best corresponding GPS position, within an accuracy of few centimeters ( $< 10cm$ ). The work done in this paper is different from prior research attempting to detect and localize these nests, creating a localized, tree-level scanning system which could be deployed in urban areas such as near houses, in private gardens, farms, road-side plantations etc. which their pine trees are also being attacked by PPM [62]; as much as in dense foliage forests.

In this work, a computer vision-based detection algorithm that leverages convolutional neural networks to detect the presence of a PPM nest and locate it within an image was proposed. A minimum detection accuracy of 90% was artificially imposed for the proposed solution to be accepted for later adoption by stakeholders. It is found that relying on RGB images in the visual spectrum alone for detecting PPM nests did not provide adequate accuracy (in the range of 60%), where a significant number of PPM nests went undetected in certain cases, with several false positive detections obtained in other cases. Since we aim to furnish an end-to-end localized solution with high precision to specifically target each PPM nest, false positive detections were considered a serious issue. Hence, a more accurate detection system was required to overcome the shortcomings of training the machine learning algorithms on RGB images alone. Inspired by the idea of using thermal images in agriculture for various detection and inspection purposes, we leverage the thermal characteristics of the silky material of the PPM nests that causes them to have a different temperature than their surroundings. This feature is employed to acquire a richer dataset for machine learning that can limit the number of false detections.

Datasets specifically collected for this work are acquired from different loca-

tions. Several altitudes are targeted in the aim of gaining a better understanding of PPM nests' characteristics. Visited areas at multiple altitudes (300, 450, 800, and 1100m) had PPM nests of similar sizes, with an average size of 15cm. With respect to thermal images, the silky material proved to entrap more heat than the tree branches and needles. Having most of the nests built on the outer-most branches of pine trees, little difficulty is faced during the collection of images and validation of the proposed system. That said, we discuss next some potential scenarios where the proposed system might not function as intended due to various challenges. A limited, yet possible, scenario may involve the nest getting affected by shade, which would result in a poor representation in the thermal images. Another challenge can occur when trying to capture thermal images of nests that are located in the lower part of the tree (lowest branches), where the background can become the ground, which may contain heat-absorbing objects that have similar temperature as the PPM nests. The proposed deep-learning architecture can deal with the above challenges and limitations given its learning of features from both spectra (visible and thermal), which can attenuate the effect of such unpredictable scenarios. In fact, the fusion of images from both spectra produced a significant improvement in detection accuracy (exceeding the specified threshold of 90%), which is achieved by reducing the number FP detections. In most cases, the individually trained branches yielded much lower detection accuracy and more FPs as compared to the dual-stream CNN. This gave confidence in the type of dataset (paired images), the suitability of the employed machine learning algorithm given the limited available dataset, and the dual-stream network architecture that we are proposing in this work. The conducted experiments demonstrate the applicability and performance of the proposed system in real-life conditions, as it achieved a similar mAP value to that obtained during the

training and testing phase on the custom yet limited dataset.

In this work, we conducted an interesting performance comparison by testing two feature extraction networks (ResNet 101 and VGG16) in a multi-channel system, something that has not been previously performed in such architectures. The interesting result of the comparison was in the fact that the ResNet 101 network, which performed poorer feature extraction in the single-channel architecture, outperformed the VGG16 network in the dual-channel architecture. This can be interpreted by the fact that ResNet tends to extract a wider range of features from the PPM nests present in the images, which is illustrated by the higher number of TP cases in the single-channel training, at the cost of recording a higher number of FP cases as well. However, in the multi-channel system, this wide range of extracted features by ResNet is fused with the features from the other modality, which produces better a richer set of features that represent the PPM nests, and thus better learning and training.

After achieving accurate detection of PPM nests, their global positions are reported back on a map with minimal error. Geo-localization is applied to accurately estimate the nests' positions as best as possible, especially when the scanning drone is not perfectly aligned on top of the nests. An RTK-GPS is used in the experimental setup for ground truth determination, which demonstrated the accuracy of the devised estimation scheme, even when estimating the position of nests that appear in the border of the processed frame, i.e., not right at its center. It is important to note that the achieved localization results were based on the assumption that the nest's size is around  $15cm$  (based on observations from field visits), and that the drone scans the trees from a height of  $2m$  above the region to be scanned.

This height restriction is due to the low resolution of the used thermal camera,

which requires the drone to be in close proximity of the scanned nests to make them perceivable in the captured thermal image. While these assumptions proved the concept of the proposed system, an enhanced solution can include using higher resolution thermal imaging devices and sensors that provide depth information (e.g. stereo camera or 2D LiDAR).

In addition to detection and localization, object tracking is performed to identify the same nests that appear in several subsequent frames, with the purpose of reporting back the best corresponding position of each nest, and not to plot the corresponding position of each frame detection. A Kalman filter is implemented to achieve accurate tracking by predicting the position of the detected objects in future frames, disregarding false positives, and giving the same ID to the correct predictions. The KF follows a constant velocity model, which yielded successful tracking in the conducted experiments. This constraint poses a limitation of the proposed system where sudden perturbations in the drone's motion may cause loss of tracking, or re-initialization of the tracker in less severe cases. This issue can be remedied by investigating more robust filters at a later stage if the system is to be deployed in harsh conditions.

The complete system is tested in real-life experiments and demonstrated its success at detecting, tracking, and localizing PPM nests. In the conducted experiments, the system was able to detect nests with high accuracy (97%), estimate the position of the detected nests properly ( $< 20cm$ ), and track the same nests that appear in several frames in order to report back the corresponding GPS location. For better visualization, two orthomosaic maps are created. The first one is from the geo-referenced frames captured while scanning the trees for nests, and the second one is from captured frames of the experimented area at high altitude, yielding a Birdseye-view satellite image showing the positions of the



detected nests.

In this thesis, the developed system is only able to scan a limited number of trees in one flight, due to issues faced with the system's hardware and configuration. In the validation experiments, we aimed to avoid causing damage to the octocopter UAV, which is an expensive research platform and is very hard to manually control and maneuver, thus we carefully tested the system in safe areas where trees stand solely without neighboring trees, and on PPM nests that are located on the lower level of the tree and could be easily reached. Another encountered issue was the discontinuous capturing of video frames and GPS data by the Raspberry Pi, due to its limited computational power that requires reconfiguration and repetition of the experiment all over again. We also faced issues with the RTK-GPS frequently losing its fixed signal upon experimentation, which required repeating of the experiment when this happens, even if the signal fixation loss was for few seconds of the flight. Although facing these hardware issues, the conducted experiments demonstrate that the system can detect several PPM nests, albeit not in a continuous manner. By addressing these limitations, the proposed system can be utilized for scanning large areas, while considering the design of a localized solution for eradicating these menacing pests.

# Chapter 6

## Conclusion

In this thesis, a multi-stream CNN architecture is proposed with the aim of designing a robust and accurate PPM nests detector. During preliminary experiments, it was discovered that the silky-like material of PPM nests lead to higher heat absorption and entrapment than the surrounding branches and leaves. As a result, the thermal spectrum was included as an additional channel in the data input, paired with the regular RGB channel; both channels provided complementary valuable features as inputs to the multi-stream CNN. The performance of two feature extraction networks was investigated when implemented in such a multi-channelled manner.

Experimental results validated the worth of the multi-channel approach over the single channel, especially in these unstructured environments where large variations in appearance are expected. Deep learning proved to be well-suited for applications with limited datasets when using a multi-stream framework, enabling the design of a robust detector that required relatively low computation and training time.

An advanced RTK-GPS was used in the experiments to guarantee an accurate

geo-localization process when passing on top of the scanned nests;, however, a GPS calculation method was implemented to geo-tag the position of the detected nests located in the drone's perimeter. In order to report back a map showing the position of the detected nests, a Kalman Filter was utilized as a nest tracker , to avoid reporting back the position of the same nest several times. Experimental results validated the successful combination of the aforementioned systems in performing a robust detection and localization of PPM nests, reporting back the desired 2D and orthomosaic maps.

# Appendix A

## Abbreviations

PPM	Pine Processionary Moth
DNN	Deep Neural Network
DBM	Deep Boltzmann Machines
SVM	Support Vectors Machines
KNN	K-NearestNeighbor
CNN	Convolutional Neural Network
R-CNN	Regional-Convolutional Neural Network
RPN	Region Proposed Network
FC	Fully Connected
BN	Batch Normalization
TF	TensorFlow
GPS	Global Positioning System
RTK	Real-time kinematic
KF	Kalman Filter
IOU	Intersection Over Union
mAP	Mean Average Precision

TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
TPR	True Positive Rate
FPR	False Positive Rate
GPU	Graphical Processing Unit
SFM	Structure from Motion

# Bibliography

- [1] A. Battisti, M. Avci, D. N. Avtzis, M. L. B. Jamaa, L. Berardi, W. Berrettima, M. Branco, G. Chakali, M. A. E. A. El Fels, B. Frérot, and Others, “Natural history of the processionary moths (*Thaumetopoea* spp.): new insights in relation to climate change,” in *Processionary moths and climate change: An update*, pp. 15–79, Springer, 2015.
- [2] “Mallorca’s Most Dangerous Small and Furry Insect.”
- [3] L. Cayuela, J. A. Hódar, and R. Zamora, “Is insecticide spraying a viable and cost-efficient management practice to control pine processionary moth in Mediterranean woodlands?,” *Forest Ecology and Management*, vol. 261, no. 11, pp. 1732–1737, 2011.
- [4] J. A. Hódar, J. Castro, and R. Zamora, “Pine processionary caterpillar *Thaumetopoea pityocampa* as a new threat for relict Mediterranean Scots pine forests under climatic warming,” *Biological conservation*, vol. 110, no. 1, pp. 123–129, 2003.
- [5] M. Kanat, M. H. Alma, and F. Sivrikaya, “Effect of defoliation by *Thaumetopoea pityocampa* (Den. & Schiff.) (Lepidoptera: Thaumetopoeidae) on annual diameter increment of *Pinus brutia* Ten. in Turkey,” *Annals of forest science*, vol. 62, no. 1, pp. 91–94, 2005.

- [6] P. S. Arnaldo, S. Chacim, and D. Lopes, “Effects of defoliation by the pine processionary moth *Thaumetopoea pityocampa* on biomass growth of young stands of *Pinus pinaster* in northern Portugal,” *iForest-Biogeosciences and Forestry*, vol. 3, no. 6, p. 159, 2010.
- [7] A. Cardil, U. Vepakomma, and L. Brotons, “Assessing pine processionary moth defoliation using unmanned aerial systems,” *Forests*, vol. 8, no. 10, p. 402, 2017.
- [8] A. Battisti, M. Stastny, S. Netherer, C. Robinet, A. Schopf, A. Roques, and S. Larsson, “Expansion of geographic range in the pine processionary moth caused by increased winter temperatures,” *Ecological applications*, vol. 15, no. 6, pp. 2084–2096, 2005.
- [9] A. Inal, D. U. Altintas, H. K. Güvenmez, M. Yilmaz, and S. G. Kendirli, “Life-threatening facial edema due to pine caterpillar mimicking an allergic event,” *Allergologia et immunopathologia*, vol. 34, no. 4, pp. 171–173, 2006.
- [10] A. I. Rodriguez-Mahillo, M. Gonzalez-Muñoz, J. M. Vega, J. A. López, A. Yart, C. Kerdelhué, E. Camafeita, J. C. Garcia Ortiz, H. Vogel, E. Petrucco Toffolo, and Others, “Setae from the pine processionary moth (*Thaumetopoea pityocampa*) contain several relevant allergens,” *Contact Dermatitis*, vol. 67, no. 6, pp. 367–374, 2012.
- [11] S. Akinci and A. H. Göktogan, “Detection and Mapping of Pine Processionary Moth Nests in UAV Imagery of Pine Forests Using Semantic Segmentation,”
- [12] L. Cayuela, R. Hernández, J. A. Hódar, G. Sánchez, and R. Zamora, “Tree damage and population density relationships for the pine processionary

- moth: Prospects for ecological research and pest management,” *Forest ecology and management*, vol. 328, pp. 319–325, 2014.
- [13] Ö. Güven, T. Aydın, I. Karaca, and T. Butt, “Biopesticides offer an environmentally friendly solution for control of pine processionary moth (*Thaumetopoea wilkinsoni* Tams) larvae and pupae in urban areas,” *Biocontrol Science and Technology*, pp. 1–18, 2020.
- [14] K. Nguyen, C. Fookes, and S. Sridharan, “Improving deep convolutional neural networks with unsupervised feature learning,” in *Image Processing (ICIP), 2015 IEEE International Conference on*, pp. 2270–2274, IEEE, 2015.
- [15] K.-K. Sung and T. Poggio, “Example-based learning for view-based human face detection,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 1, pp. 39–51, 1998.
- [16] P. Viola and M. J. Jones, “Robust real-time face detection,” *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.
- [17] C. P. Papageorgiou, M. Oren, and T. Poggio, “A general framework for object detection,” in *Computer vision, 1998. sixth international conference on*, pp. 555–562, IEEE, 1998.
- [18] K. A. Nugroho, “A comparison of handcrafted and deep neural network feature extraction for classifying optical coherence tomography (oct) images,” in *2018 2nd International Conference on Informatics and Computational Sciences (ICICoS)*, pp. 1–6, IEEE, 2018.
- [19] N. G. Paterakis, E. Mocanu, M. Gibescu, B. Stappers, and W. van Alst, “Deep learning versus traditional machine learning methods for aggregated



energy demand prediction,” in *Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), 2017 IEEE PES*, pp. 1–6, IEEE, 2017.

- [20] X. Li and S. Wang, “Object detection using convolutional neural networks in a coarse-to-fine manner,” *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 11, pp. 2037–2041, 2017.
- [21] T.-D. Truong, V.-T. Nguyen, and M.-T. Tran, “Lightweight deep convolutional network for tiny object recognition,” 2018.
- [22] S. Zia, B. Yüksel, D. Yüret, and Y. Yemez, “Rgb-d object recognition using deep convolutional neural networks,” in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 887–894, IEEE, 2017.
- [23] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1717–1724, 2014.
- [24] A. Cardil, K. Otsu, M. Pla, C. A. Silva, and L. Brotons, “Quantifying pine processionary moth defoliation in a pine-oak mixed forest using unmanned aerial systems and multispectral imagery,” *Plos one*, vol. 14, no. 3, p. e0213027, 2019.
- [25] A. Aakif and M. F. Khan, “Automatic classification of plants based on their leaves,” *Biosystems Engineering*, vol. 139, pp. 66–75, 2015.
- [26] M. Dyrmann, H. Karstoft, and H. S. Midtiby, “Plant species classification using deep convolutional neural network,” *Biosystems Engineering*, vol. 151, pp. 72–80, 2016.

- [27] J. Xiong, Z. Liu, S. Chen, B. Liu, Z. Zheng, Z. Zhong, Z. Yang, and H. Peng, “Visual detection of green mangoes by an unmanned aerial vehicle in orchards based on a deep learning method,” *Biosystems Engineering*, vol. 194, pp. 261–272, 2020.
- [28] S. Tu, Y. Xue, C. Zheng, Y. Qi, H. Wan, and L. Mao, “Detection of passion fruits and maturity classification using Red-Green-Blue Depth images,” *Biosystems Engineering*, vol. 175, pp. 156–167, 2018.
- [29] L. Fu, Y. Majeed, X. Zhang, M. Karkee, and Q. Zhang, “Faster R-CNN-based apple detection in dense-foliage fruiting-wall trees using RGB and depth features for robotic harvesting,” *Biosystems Engineering*, vol. 197, pp. 245–256, 2020.
- [30] B. Neupane, T. Horanont, and N. D. Hung, “Deep learning based banana plant detection and counting using high-resolution red-green-blue (RGB) images collected from unmanned aerial vehicle (UAV),” *PloS one*, vol. 14, no. 10, 2019.
- [31] A. Fuentes, S. Yoon, S. C. Kim, and D. S. Park, “A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition,” *Sensors*, vol. 17, no. 9, p. 2022, 2017.
- [32] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, “Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks,” *IEEE Access*, vol. 7, pp. 59069–59080, 2019.
- [33] H. Gan, W. S. Lee, V. Alchanatis, and A. Abd-Elrahman, “Active thermal imaging for immature citrus fruit detection,” *Biosystems Engineering*, vol. 198, pp. 291–303, 2020.

- [34] J. Wagner, V. Fischer, M. Herman, and S. Behnke, “Multispectral Pedestrian Detection using Deep Fusion Convolutional Neural Networks.,” in *ESANN*, 2016.
- [35] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool, “Deepfruits: A fruit detection system using deep neural networks,” *Sensors*, vol. 16, no. 8, p. 1222, 2016.
- [36] N. Hasler, B. Rosenhahn, T. Thormahlen, M. Wand, J. Gall, and H.-P. Seidel, “Markerless motion capture with unsynchronized moving cameras,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 224–231, IEEE, 2009.
- [37] P. Pourcelot, F. Audigié, C. Degueurce, D. Geiger, and J. M. Denoix, “A method to synchronise cameras using the direct linear transformation technique,” *Journal of Biomechanics*, vol. 33, no. 12, pp. 1751–1754, 2000.
- [38] P. Shrestha, M. Barbieri, H. Weda, and D. Sekulovski, “Synchronization of multiple camera videos using audio-visual features,” *IEEE Transactions on Multimedia*, vol. 12, no. 1, pp. 79–92, 2009.
- [39] P. Shrestha, H. Weda, M. Barbieri, and D. Sekulovski, “Synchronization of multiple video recordings based on still camera flashes,” in *Proceedings of the 14th ACM international conference on Multimedia*, pp. 137–140, 2006.
- [40] S. N. Sinha and M. Pollefeys, “Synchronization and calibration of camera networks from silhouettes,” in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 1, pp. 116–119, IEEE, 2004.

- [41] P. K. Rai, K. Tiwari, P. Guha, and A. Mukerjee, “A cost-effective multiple camera vision system using firewire cameras and software synchronization,” in *Proceedings of the 10th International Conference on High Performance Computing, Hyderabad, India*, vol. 1720, 2003.
- [42] B. Holveck and H. Mathieu, *Infrastructure of the GrImage experimental platform: the video acquisition part*. PhD thesis, INRIA, 2004.
- [43] T. Svoboda, H. Hug, and L. Van Gool, “Viroom—low cost synchronized multicamera system and its self-calibration,” in *Joint Pattern Recognition Symposium*, pp. 515–522, Springer, 2002.
- [44] V. R. Gaddam, R. Langseth, H. K. Stensland, C. Griwodz, M. Riegler, T. Kupka, H. Espeland, D. Johansen, H. D. Johansen, and P. Halvorsen, “Camera Synchronization for Panoramic Videos,” in *MediaSync*, pp. 565–592, Springer, 2018.
- [45] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, “Tensorflow: a system for large-scale machine learning,” in *OSDI*, vol. 16, pp. 265–283, 2016.
- [46] P. D. T. Ms, Y. Lin, “coco api.”
- [47] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.
- [48] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

- [49] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, pp. 91–99, 2015.
- [50] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [51] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [52] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and real-time tracking,” in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, IEEE, 2016.
- [53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.
- [54] E. Softwares, “Plant factory,” 2016.
- [55] E. Softwares, “Vue,” 2016.
- [56] “Agisoft PhotoScan.”
- [57] S. Upadhyaya, G. Pettygrove, J. Oliveira, and B. Jahn, “An introduction—global positioning system,” 2008.

- [58] L. Zhang, L. Lin, X. Liang, and K. He, “Is faster R-CNN doing well for pedestrian detection?,” in *European conference on computer vision*, pp. 443–457, Springer, 2016.
- [59] “RealityCapture.”
- [60] “Pix4Dmapper.”
- [61] “Google Earth Pro.”
- [62] J. Rousselet, C.-E. Imbert, A. Dekri, J. Garcia, F. Goussard, B. Vincent, O. Denux, C. Robinet, F. Dorkeld, A. Roques, and Others, “Assessing species distribution using Google Street View: a pilot study with the pine processionary moth,” *PLoS One*, vol. 8, no. 10, p. e74918, 2013.

