# ROBUST LIVESTOCK DETECTION AND COUNTING USING AN UNMANNED AERIAL VEHICLE (UAV)

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Supervisors

Prof. Peter Chong

Dr. Anthony Griffin

Prof. Timotius Pasang

28-10-2021

By

Farah Sarwar

School of Engineering, Computer and Mathematical Sciences

# Abstract

Fast pasture growth makes pastoral farming low-cost, sustainable and efficient for New Zealand. However, in recent years, sheep farming has declined due to the extra labor needed in handling these farms. One of the main tasks that need manual involvement is farm animal counting. Currently, most of the stock is counted manually with a gap of weeks, usually during a major stock movement or other events. The conventional method is time-consuming, tiring, and prone to errors. Also, this inability to constantly monitor stock numbers gives farm rustlers plenty of time to steal animals, hence causing a significant financial loss annually. Farmers desire to minimize their losses by having a daily count of their stock with the highest possible accuracy.

The research work presented in this thesis focused on solving livestock (sheep) counting problems using an unmanned aerial vehicle (UAV) and deep learning. It provides a solution to detect, count, and track livestock in a paddock using advanced methods, and gives extremely accurate information to farmers in a minimum time. The proposed system uses a UAV for counting farm stock while observing the code of conduct for UAV usage. It involves no disruption to the animals as a UAV will hardly be noticeable from the allowable height. Different deep learning algorithms were investigated in this regard and recorded videos were processed to provide the estimated sheep count in the UAV images. The next step of the system is livestock tracking, meaning that each sheep in the full paddock video should be assigned a virtual identification number to keep a track of movement in the video and to count the herd

size correctly. Fence detection was one of the main steps to avoid detecting sheep in nearby paddocks.

Stock counting using a UAV is a promising research area but offers various technical challenges, such as non-uniform illumination in images, occlusions, object scaling, rotation, noise, and the challenges in identifying objects from various visual perspectives. Previous studies have shown that there is a lot of potential in this field as not much work has been done so far.

For this research, a full dataset was created from different paddocks to make the proposed research work scalable to accommodate a variety of backgrounds and perimeters. The discussed network and the proposed method gave promising results and is crucial step forward towards a fully-automated livestock tracking system in sheep farms.

# Contents

# List of Tables

# List of Figures

# Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

Signature of candidate

# Publications and Presentations

## Journal Publications

**J1:** F. Sarwar, A. Griffin, S.U. Rehman and T. Pasang, "Detecting sheep in UAV images," *Computers and Electronics in Agriculture*, vol. 187, 2021.

## Conference Publications

**C1:** F. Sarwar, A. Griffin, P. Periasamy, K. Portas, and J. Law, "Detecting and counting sheep with a convolutional neural network," *in proceeding of 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2018, pp. 1-6.

**C2:** F. Sarwar, A. Griffin, S. U. Rehman, and T. Pasang, "Towards detection of sheep onboard a UAV," *arXiv preprint arXiv:2004.02758*, 2020.*(presented in the invited talk at British Machine Vision Conference (BMVC) workshop, 2019).*

**C3:** F. Sarwar, T. Pasang, A. Griffin, and S. U. Rehman, "Survey of livestock counting and tracking methods," *Nusantara Science and Technology Proceedings*, pp. 150–157, 2020.

**C4:** F. Sarwar, A. Griffin and T. Pasang, "Tracking livestock using a fully connected network and Kalman filter," *Geometry and Vision. ISGV 2021. Communications in Computer and Information Science,* vol 1386, pp. 247–261, 2021.

**C5:** F. Sarwar, A. Griffin, P. Chong and T. Pasang, "Pasture fence line detection in UAV videos", *in the proceedings of 36th International Conference on Image and Vision Computing New Zealand, (IVCNZ)*, 2021.

## Seminar Presentations

**P1:** F. Sarwar, T. Pasang and A. Griffin, "Livestock monitoring using an unmanned aerial vehicle (UAV)", *Presented in a collaborative tech-exchange seminar between AUT and AIMHI (Advanced Institute of Manufacturing with High-tech Innovations), National Chung Cheng University, Taiwan*. 31 August, 2020.

**P2:** F. Sarwar, A. Griffin and T. Pasang, "Livestock detection and tracking using an unmanned aerial vehicle (UAV)", *Presented in the postgraduate student seminar organised by IEEE NZ Signal Processing/Information Theory Joint Chapter in collaboration with the Acoustics Research Centre of the University of Auckland*. 27 October, 2020.

**P3:** F. Sarwar, A. Griffin, P. Chong and T. Pasang, "Detecting and counting sheep in UAV vidoes", *Presented in the seminar organised by IEEE NZ Signal Processing/Information Theory Joint Chapter in collaboration with the Department of Mechanical Engineering of the University of Auckland*. 2 June, 2021.

# Acknowledgements

I would like to express my appreciation to all those people who contributed in different ways to make this journey possible.
Special thanks to one of my supervisors, Dr. Anthony Griffin, who offered this research work to me and then helped me at every stage to reach my destination. I thank him for his patience, unmatched support, and proper guidance at every step. He always waited patiently to let me move to the next level and kept on supporting me through difficult times. Whether little or big, he always appreciated my work. Most of the data for this research work was collected by Dr. Anthony and I consider it as a major contribution towards my success. I would like to thanks the Palliser Ridge sheep farm owner and management team for letting us use their farm data for this research.
I would like to thanks Prof. Timotius Pasang to go beyond my expectations and provide me financial support at the time when it seems impossible. I was on the verge of deciding to discontinue my PhD work due to financial issues but he showed his full support and made this journey a lot easier.
Prof. Peter Chong and Dr. Saeed-Ur-Rehman guided me well in different ways and helped me to develop my understanding of this research work by asking different technical questions. They supported me at every possible step. Both of them kept on pushing me to move forward and to complete my targets in the most efficient way and within my time-frame.
I would also like to thanks Mr. Bumjun Kim to support me every time whenever I needed the GPU machine. He is such a humble person and always fulfill his duty by providing full technical support to the students.

# Dedication

I dedicate this thesis to my parents, who prayed for my success, my husband, who supported me throughout this journey, and my daughters, who were so patient with their busy mom, understood her well and still love her.

# List of Acronyms

| Acronym | Definition |
|---|---|
| AWA | Animals with Attributes |
| CAMShift | Continuously Adaptive Mean Shift |
| CNA | Cost of Non-Association |
| CNN | Convolutional Neural Network |
| CONV | Convolutional |
| DBSCAN | Density-Based Spatial Clustering of Applications With Noise |
| DCNN | Deep Convolutional Neural Network |
| Deep SORT | Deep Simple Online Real Tracking |
| DPM | Deformable Part-Based Model |
| EID | Electronic Identification |
| FAF | False Alarm per Frame |
| FASF | Feature-Selection-Anchor-Free |
| FC | Fully Connected |
| FCN | Fully Connected Network |
| FDX | Full-duplex |
| FHD | Fully High Definition |
| FN | False Negative |
| FP | False Positive |
| FPR | False Positive Rate |
| FRD-CNN | Feature Reuse Detection Convolutional Neural Network |
| GMM | Gaussian Mixture Model |
| HD | High Definition |
| HDX | Half-duplex |
| HOG | Histogram of Gradients |
| HOG-LBP | HOG Based Local Binary Patterns |
| ID | Identification |
| IOU | Intersection Over Union |
| MAE | Mean Absolute Error |
| mAP | Mean Average Precision |
| MAPE | Mean Absolute Percentage Error |
| MHT | Multiple Hypothesis Tracking |
| MOT | Multiple Object Tracking |
| MOTA | Multiple Object Tracking Accuracy |

| | |
|---|---|
| MOTP | Multiple Object Tracking Precision |
| MS | Mean Shift |
| MTGAN | Multi-Task Generative Adversarial Network |
| NAIT | National Animal Identification and Tracing |
| PICA | Person in Charge of Animals |
| PPF | Precision per Frame |
| R-CNN | Region-based Convolutional Neural Network |
| ReLU | Rectified Linear Unit |
| RFID | Radio Frequency Identification |
| RGB | Red, Green and Blue |
| RMSE | Root Mean Square Error |
| ROI | Region of Interest |
| RPF | Recall per Frame |
| SC | Sheep Count |
| SGD | Stochastic Gradient Descent |
| SIFT | Scale Invariant Feature Transform |
| SPM | Spatial Pyramid Matching |
| SSD | Single Shot Detector |
| SVM | Support Vector Machine |
| T-CNN | Tubelets with CNN |
| TN | True Negative |
| TNT | TrackletNet Tracker |
| TP | True Positive |
| UAV | Unmanned Aerial Vehicle |
| UHD | Ultra High Definition |
| WHD | Weighted Hausdorff Distance |
| YOLO | You Only Look Once |

# Chapter 1

# Introduction

This introductory chapter discusses the necessity of an easy solution for livestock counting in the pastoral farms of New Zealand and how this research contributes towards solving this particular issue. The main contributions of this research work are discussed and the chapter concludes with an outline of the overall structure of the thesis.

## 1.1 Rationale of the Study

The rainy weather of New Zealand is very beneficial for fast pasture growth and is favorable for farmers, as most of the food supplies can be covered from the green pasture, livestock handling is comparatively a low-cost task. Thus, dairy farming is a sustainable and cost-effective profession for New Zealand that helps it to compete as a good exporter of food and fibre [1]. Farmers take proper care of pasture quality by carefully selecting the grass and plant types to use in their area, and use different fertilizers to provide a portion of healthy food to the livestock.

For animals to have free movement in large areas, these grazing lands are fenced in many countries. These fenced grazing areas—also known as paddocks in New Zealand—have different herd sizes, that can go up to thousands of animals. Each farm

has a different number of smaller paddocks in it and paddock sizes also vary. Farmers shift the farm animals from paddock to paddock to let the grass grow and let animals wander in a different area for a while.

According to the statistical survey of New Zealand for the year 2020, there are a total of 52,293 farm holdings in New Zealand with an average area of 270 ha [2]. Among these farms, there are 23,403 sheep and beef farms that cover approximately 8,765 thousand ha and are 45% of the total farm holdings. Sheep farms have a major contribution of 49% in the total revenue and cattle farms are in second place with 26% contribution. For the year ended on 30 September 2019, approximately 18.5 million lambs, 3.4 million sheep and 2.7 million cattle were processed to produce 361, 91 and 697 thousand tonnes of meat, respectively. However, although deer and cattle farming has increased, stock unit in sheep farming has been declined by 15% in the last decade, and total sheep number has been reduced by 2.3% in the last year [3]. One of the main reasons behind this decline is the requirement of extra labour effort in handling sheep farms [2].

With the advancement in pastoral and agricultural techniques [4], a variety of data is available to farmers, such as the expected climatic changes, area temperatures, soil dampness values, pasture covers, and individual stock identification for growth rates monitoring and animal welfare. Most of the tasks done in livestock handling are quite labour-intensive, and livestock counting is one of these tasks. Counting is usually done with a gap of either weeks or months and thus makes stock count information occasional data. It also seems disruptive for the farm animals, because they have to pass through narrow choke points or drafting races for this purpose. One farmer manager has stated that they face a loss of around NZ$ 60,000 annually due to either sheep theft or missing stock (P. Kurt, personal communication, May 05, 2019). There are many cases where sheep and especially pregnant ewes were stolen from a farm, and farmers found out about their financial loss after 2 to 3 months on the next sheep count. Many

farmers are facing this issue [5, 6, 7], and it was reported that farm rustling is worth NZ$ 120 million a year [8, 9]. Approximately NZ$ 22.4 million were paid to farmers for insurance claims only over the time period of 2015 - 2019. Farmers learn about stolen or missing livestock weeks later, so, they usually do not report it to the police [10]. Farm rustlers are well aware of the fact that farmers do not count the farm animals frequently, therefore, they get plenty of time. If there was a way to identify animal theft earlier in any respective farm then farmers could inform the concerned authorities more promptly. An earlier discovery of the missing sheep may have positive results. To speed up the counting process of any livestock and to monitor the farm animals more easily, a farm manager has shown interest to get this system automated as much as possible without compromising on the accuracy of provided data (P. Kurt, personal communication, May 05, 2019). Easy and robust remote livestock counting and monitoring is one such system for handling the herds efficiently. Monitoring the distribution and population of animal species over time is also a key ingredient to successful nature conservation [11].

## 1.2 Objectives

The main objective of this research is to help farmers by designing a remote and sophisticated livestock counting system that can give them a stock count with minimum possible physical involvement. In this research, we are proposing a time-efficient, cost-effective and robust livestock counting and monitoring system. It uses an unmanned aerial vehicle (UAV) for either capturing paddock images or recording paddock videos. The gathered data will be processed using machine learning and deep learning algorithms to get the required output. Although there are a few methods available for animal counting, to the best of the researcher's knowledge, such a proposed system does not currently exist in the field of pastoral or dairy farming. Farmers will be able to see the whole paddock with the help of the UAV and can keep an eye on the required

statistical data. Stock count will be accessible to the farmers throughout the year, it will also be cost-effective and time-efficient.

This research can help the farming industry in New Zealand by excluding many physical tasks of the farmers needed for livestock monitoring and counting, and it is an efficient way to get all stock counted. In current stock counting method, farmers let animals pass through narrow chokes and prior to this step, gathering all animals at the entrance point needs a lot of physical involvement. This process is disturbing for animals and time-consuming for the farm manager. This whole process can be avoided using our proposed method. The cost of this system will always remain independent of the number of animals in the possession of the respective farm manager. The UAV can be bought only once for the respective farm and the counting task will need less effort as well as will be less disruptive for the animals.

To cover the above mentioned contributing factors, the following questions are answered in this thesis:

1. How can we efficiently and accurately detect hundreds of farm animals at various scales in the aerial images using a deep learning algorithm?
2. How can we track the detected animals in the respective paddock videos?
3. How can we detect fence lines robustly and define the boundary of a paddock to avoid an overlap in counting with the nearby paddock animals?

## 1.3 Challenges

### 1.3.1 Data Collection

Currently, no dataset of any livestock is publicly available for this kind of research work. Recording paddock videos under different illumination conditions and backgrounds was very challenging. The difference in the background was observed with the weather

variation, so data was collected at different times of the year to have as much variety as possible. A few videos were recorded in cloudy weather, such videos were less challenging in terms of identifying the object of our interest. While a few other videos were recorded in sunny conditions and at different times of a day to have a variety in shadow lengths. These sunny videos were very challenging due to reflection of sunlight from grass, tree branches and even leaves. However, a variation in collected data was necessary to design a robust system. The next step was ground truth labelling, requiring patience and hard work to label each animal and fence wooden posts of the paddock as a region of interest (ROI). At times it is difficult to identify the number of animals standing in a group, especially in the videos recorded from higher altitudes. And fence posts and fence wires were mostly vague in the video frames, causing a difficulty to identify their correct location. However, the task of ground truth labelling was done with great effort, and it was not completed in one go but was done as more data was collected from the farms. Collectively, it took months to complete the ground truth labelling over all data.

### 1.3.2 Small Object Detection

Detecting hundreds of small objects in any image is a challenging problem in the field of deep learning. Although many algorithms exist but they usually perform below average when it comes to detecting a tiny object [12]. Furthermore, a sheep looks like a whitish blob with negligible distinguishing features, and features are mainly needed by any computer vision algorithm to identify an object correctly. Providing an algorithm that can detect such tiny objects with high accuracy will be a promising contribution in this field. Along with livestock (sheep), paddock fence detection is a part of this research and at some points the fence posts get mixed with the background, making it even more difficult to define them properly. We were unable to find any literature relevant to the

paddock fence detection and this task was also achieved with good accuracy.

### 1.3.3 Multiple Object Tracking

After detecting objects successfully in an image, the next task is tracking multiple objects (sheep) in a video to get a total stock count in a full paddock. Tracking the objects of interest in any video is a very challenging problem of computer vision. Comparatively, multiple object tracking (MOT) is much more complex than single object tracking. Where single object tracking is for only one particular object that needs to be tracked throughout the video, MOT involves tracking the trajectories of multiple objects in a video. These objects can either be of the same category or different classes. Furthermore, a static or moving background will have different types of complexities. MOT involves the creation of new track identities as objects appear in the video frame, the use of a data associator for track assignment, matching tracks with the detected objects based on a cost factor and deleting the lost track. In the last decade, the use of different deep learning and machine learning algorithms for tracking multiple objects in videos has increased the efficiency of such systems. As the videos recorded by a UAV have more challenges, this offers room for extensive research. In the tracking-by-detection method, there are three main components: object detector, tracker and data associator. All these components play an equally important role and can be tuned to the highest efficiency for increasing the overall performance of the system. In our case, as there are hundreds of small objects, an object tracker is needed that can rapidly update all tracks simultaneously.

## 1.4 Structure of Thesis

The structure of this thesis is as follows: A detailed literature review about existing livestock counting methods, deep learning algorithms for object detection and MOT

algorithms are discussed in Chapter 2. Chapter 3 discusses the overall research methodology and details of the data collection. The deep learning based proposed methods for livestock detection are discussed in detail in Chapter 4. Chapter 5 shows the complete object detection, tracking and counting methodology. Details of fence detection, the last task of this research, is presented in Chapter 6. The experimental evaluation of all the methods and respective results are also discussed in the same chapters to avoid ambiguity in the end.

Chapter 7 concludes the work presented in this thesis with some discussion on the future work that can be done in this field.

# Chapter 2

# Literature Review

The proposed research aims at providing an appropriate solution to the livestock counting problem and this section discusses the existing literature that laid the basis of our work. A comprehensive study has been carried out to develop good background knowledge and find the research gap in this field. This also helped in pursuing the proposed methodology for this research.

This chapter is divided into five subsections: the existing methods for livestock counting and the state-of-the-art methods for object detection are discussed in Section 2.1 and 2.2, respectively. The algorithms for single or multiple object tracking are discussed briefly in Section 2.3. The details of fence detection, work done by other researchers in the relevant area and the current applications of the UAV along with its uses for livestock monitoring are also highlighted. A critical analysis and overall summary are then provided to enlighten the research gap.

The material mentioned in this chapter in terms of comparison of hardware and software-based methods and discussion about the need of this research is based on our paper [13]. More literature review from our other published papers is also added. The manuscript has been published as follows:

**C3:** F. Sarwar, T. Pasang, A. Griffin, and S. U. Rehman, "Survey of livestock counting

and tracking methods," *Nusantara Science and Technology Proceedings*, pp. 150–157, 2020.

## 2.1 Existing Livestock Counting Methods

The existing animal identification and counting methods involve the usage of different categories of radio frequency identification (RFID) and electronic identification (EID) ear tags [14], RFID bolus, and GPS collars [15]. Through the implantation of an RFID bolus, an animal can be tracked throughout its lifespan based on an assigned number. Although for each animal, this is a one-time process, good technological-based RF items can cost almost NZ$ 3.35 per animal which is very costly for farmers [14]. Similarly, RF ear tags can cost around NZ$ 0.49 for each animal but the only advantage they have is reusability. RFID and EID tags work in the same fashion but the latter one contains an electronic chip.

National animal identification and tracing (NAIT) have very strict regulations for cattle and deer farms. If a person in charge of animals (PICA) has only one cattle or deer than PICA should get them tagged using RFID half-duplex (HDX) or full-duplex (FDX) ear tags. All respective animals should get registered with NAIT immediately and status needs to be updated during any stock movement. These tags are not reusable and PICA should inform NAIT before replacing them in the case of the lost animal. All of these technologies require direct contact with farm animals in order to read the tag through a tag reader, and the price of a tag reader starts from NZ$ 1,620. However, these existing methods are not convenient and cannot give an automatic stock count to the farmers.

A GPS collar or harness can also be used to track the movement of animals and monitor their health continuously to automate animal counting [15]. Though they do offer different advantages, GPS collars are expensive with the minimum cost per item

of NZ$ 89. Maintenance of these collars in terms of replacing batteries can increase the involvement of farmers instead of reducing it.

In the last decade, UAVs are increasingly being employed for data collection because of their continuous technological advancements, constant decline in cost, size, and weight, evolution in attached sensors and high-tech cameras [16, 17]. Furthermore, the end-user can manage both the spatial and temporal resolution of recorded data. As a result, terrestrial as well as marine ecological surveys find them attractive and useful for collecting data. Researchers are using UAVs in different research areas like wildlife monitoring [18, 19, 20], ungulate survey [21], agricultural growth analysis [22], geographical mapping [23], 3D thermal mapping, detecting different deficiencies and stress levels in tress and crops [24, 25, 26], forest fire detection [27], classification of cultivated land [28], surveillance [29], security [30], shipping and delivery [31], rescue operation [32], and many more.

A few researchers have proposed the use of UAV to track [33] and monitor [34] farm animals, and a few farmers are effectively using UAVs as sheepdogs [35, 36]. Figure 2.1 shows the above-mentioned hardware-based technologies and software-based methodologies are discussed briefly in the following subsections.

(a) (b) (c) (d)

Figure 2.1: (a) RFID ear tags, (b) RFID bolus, (c) GPS collar and (d) UAV.

## 2.2 Object Detection

Processing real-time or offline image processing for animal detection in videos requires machine learning or deep learning based strategies, which come under the artificial intelligence domain. Artificial intelligence is actually the capability of a machine to imitate human behaviour intelligently and handling varieties of complicated tasks successfully. A few such tasks are understanding and interpreting different languages, making decisions under critical conditions, securing systems, identifying objects, following a given pattern and making new decisions to solve a problem. Machine learning is defined as a science of various techniques and approaches for problem-solving of artificially intelligent systems. Deep learning is a branch of machine learning, that is an inspiration of the human brain, and can have multi-layered neurons for data processing, and it is very effective for feature extraction on its own. Though it requires an enormous amount of data and very efficient high-end machines with discrete graphics cards for processing, once trained, deep learning algorithms can provide results very fast. The main difference between a machine learning and deep learning algorithm is that the latter one is a technique that does feature extraction and classification as an end-to-end process [37] as shown in Figure 2.2.

Object detection in images was initiated in the 1960s as pattern recognition for character recognition [38]. After that, multiple algorithms and techniques were proposed in the field of computer vision [39]. Thus, it actually started as pattern recognition, evolved towards feature extraction and matching, and then finally shifted to automatic feature extraction and classification. The latter technique is the data-driven method and requires a large training dataset. Though the object to be detected can belong to any living and non-living category, it faces many challenges due to occlusion, illumination, different noises, resolution and object scaling issues.

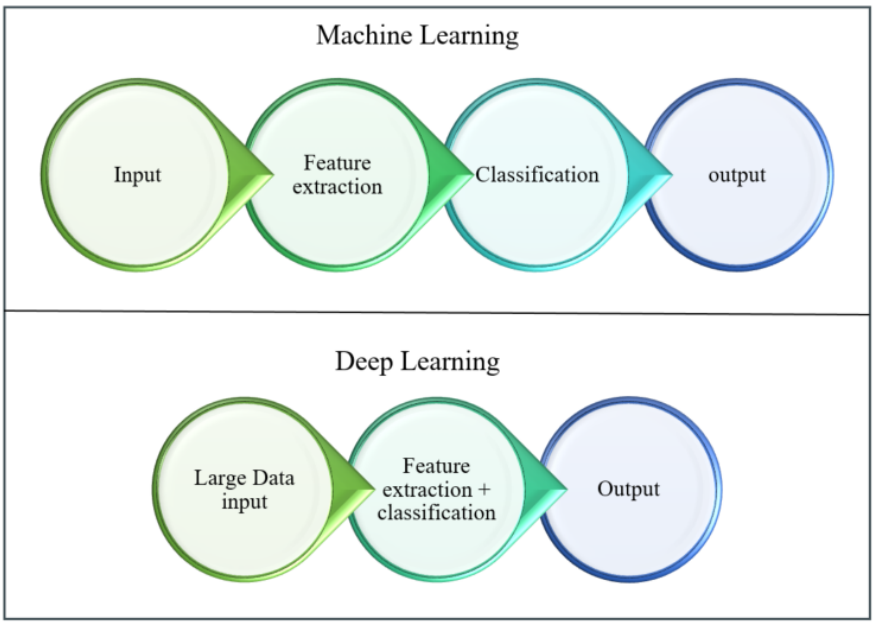



Figure 2.2: Comparison of machine learning and deep learning.

Handcrafted features extraction for object classification and detection were introduced in the late 2000s. This laid the foundation for techniques which are scale, translation, rotation, illumination and viewpoint invariants [37], and it started with the scale invariant feature transform (SIFT) [40], and support vector machine (SVM) [41]. Researchers investigated new ways of solving the object detection problem using object features and various algorithms were then proposed successfully. Some of the main methods are cascade networks [42], bag of words [43], histogram of gradients (HOG) [44], spatial pyramid matching (SPM) [45], deformable part-based model (DPM) [46], HOG based local binary patterns (HOG-LBP) [47], improved fisher kernels [48] and selective search [49].

In recent years, many researchers are aiming at unravelling object detection techniques using convolutional neural networks (CNNs) [50, 51, 52]. Ciresan *et al.* [53] achieved remarkable object classification result on the NORB and CIFAR-10 datasets. In 2012, Krizhevsky *et al.* [54] used deep CNN (DCNN) to perform the object classification task on ImageNet and obtained unmatched results. After that, different versions of

CNNs were proposed and a few of them are: Region-based CNN (R-CNN) [50], Fast R-CNN [55], Faster R-CNN [56], you only look once (YOLO) [57, 12, 58] single shot multibox detector (SSD) [59], Mask R-CNN [60], RetinaNet with focal loss [61] and U-Net with weighted Hausdorff distance (WHD) [62]. Later on multiple techniques were proposed as Cascade R-CNN [63], CornerNet [64], feature-selection-anchor-free (FSAF) framework [65] and CenterNet [66]. A brief timeline of object detection techniques is shown in Figure 2.3, and Table 2.1 shows a comparison of a few relevant CNN-based techniques.

Several modified versions of YOLO are available that are the fastest algorithms so far but the respective authors have also mentioned that it may give higher localization errors in comparison with Faster R-CNN and none of them can detect small objects well [12]. A very good survey of object detectors is given in [37] and shows a comparison of different existing techniques and highlights many research gaps like open world learning, more efficient detection frameworks, compact CNN feature generation, robust detection and 3D object detection.

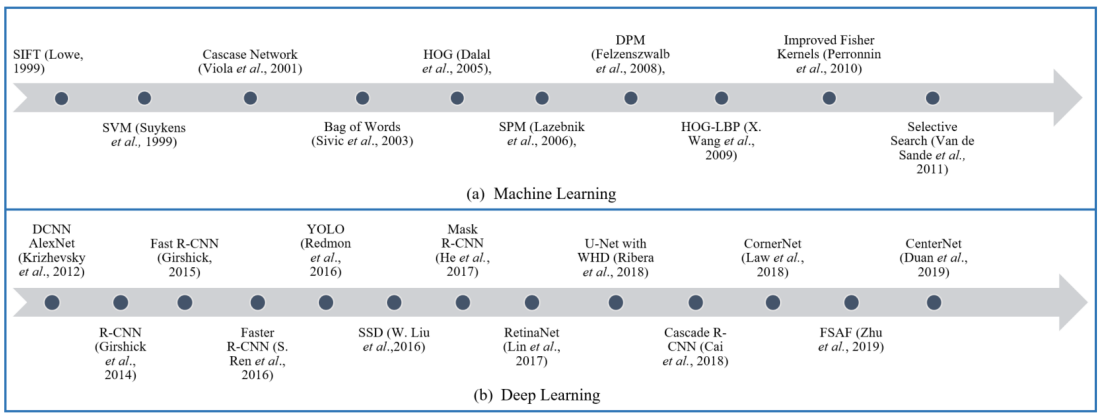



Figure 2.3: A brief timeline of object detection techniques.

Table 2.1: Comparison of a few deep learning based algorithms.

| **Method** | **Advantages** | **Limitations** |
|---|---|---|
| R-CNN [50] | Best two-stage object detector | Computationally expensive in terms of memory and time. |
| Fast R-CNN[55] | Faster than R-CNN, accept multiscale objects and no memory needed for feature caching | Long training time and computationally complex as compared to R-CNN |
| Faster R-CNN [56] | Faster than R-CNN and Fast R-CNN | The complex training process and needs improvement for optimization and still not good for real-time processing |
| SSD[59] | Processing speed is quite fast and performs detection at multiscale | Cannot detect small size objects |
| YOLO [12] | Fastest available detection and classification algorithm and it can be used for real-time applications | More localization errors and still needs improvement for small object detection |
| Mask R-CNN [60] | Simple and effective for object segmentation | Difficulty in handling real-time applications |
| U-Net [62] | Good for detecting small size objects and uses centroids of objects instead of bounding boxes | Training time is very long as compared to other algorithms |

### 2.2.1 UAV for Livestock Detection

Animal monitoring can equally help farmers and zoologists to identify various health issues in animals. Many researchers have been exploring this research area in the last two decades and the approach of wildlife monitoring started in 1991 [67] by Jachmann for estimating elephant densities using aerial samples. However, many challenges arise due to the diversity of background, species-specific characteristics, spatial clustering of animals [68]. Researchers use different statistical and biological methodologies to address these challenges [4] .

Machine learning based algorithms such as template matching algorithm [69], AdaBoost classifier [70], power spectral based techniques [71], DPM and SVM [11]

were used to detect animals in aerial images. In [34], Barbedo and Koenigkan provided an overview about the usage of unmanned aerial systems in monitoring some cattle farms, and later on published their relevant research results in [72, 73]. They collected a cattle dataset at 30 m altitude using a UAV and were able to achieve 87% in the most favourable conditions. Researchers also used a UAV for quantifying the spatial proximity in cattle [74], monitoring spatial heterogeneity of pasture [75] and livestock surveillance.

A study by Manzoor *et al.* [76], who collected livestock data at the height of 50 m, reported 76% mAP on the VGG16 network by employing SSD. In their dataset, they have gathered 3900 RGB images of 300 × 300 pixels each, at a height of 50 m. Using a modified version of U-Net and Google Inception v4, Han *et al.* [77] proposed a new algorithm in the same domain. Also, Rahnemoonfar *et al.* [78] assisted in the detection and counting of cattle but faced a problem with highly occluded objects. Meena and Agilandeeswari [79] achieved average precision of 92% while monitoring livestock through an autonomous unmanned ground vehicle. They tested their trained YOLOv2 network on three benchmark animal datasets, Missouri camera trap [80], North American camera trap [81], and Animals with Attributes (AWA) [82]. Tassinar *et al.* [83] used a fixed-position camera at the height of 3.5 m to detect cattle in a barn with roughly 86% recall and this is their ongoing research.

### 2.2.2 Small Object Detection

In this study, a UAV was used to capture videos of the entire paddock from the air in order to cover as much ground as possible in each frame. Consequently, the sheep become tiny objects without those detailed features that are required for successfully training a CNN. In many cases, machine learning and deep learning algorithms that identify large objects are incapable of identifying such small ones.

Researchers have proposed small object detection algorithms using multiple filter banks [84], infrared imaging [85] and contour models [86]. However, the improvement in camera resolution has made it possible for researchers to extract even more details of objects, and they started investigating the use of CNNs to do this. In a paper published in 2016 by Chen *et al.* [87], they combined R-CNN with a proposed region proposal generation and improved the mean average precision (mAP) by approximately 30% compared with the state-of-the-art R-CNN. Furthermore, Kong *et al.* [88] proposed the HyperNet, Eggert *et al.* [89] improved the performance of the Faster R-CNN, Li *et al.* [90] designed a perceptual generative adversarial network (Perceptual GAN) and Yancheng Bai *et al.* [91] worked on an end-to-end multi-task generative adversarial network (MTGAN). Li *et al.* [92] constructed a CNN using feature reuse detection (FRD-CNN) method and achieved 84% mAP on the PASCAL VOC2007 dataset. On the same dataset, Ma *et al.* [93] achieved 88% mAP using multiscale and multi-task algorithms. Some researchers also enhanced the performance of Faster-RCNN [94, 95, 96], SSD [97, 98, 99, 76] and YOLO [100, 101].

As far as author is aware, none of these algorithms are currently tested on aerial image datasets, which do not always include detailed information about objects. These results are based on already existing datasets, which include relatively large objects at high resolution, and the purpose of this study is to address one of these research gaps.

### 2.2.3 Fence Detection

Researchers have used different computer vision techniques for road line detection using cameras mounted on the car and power line detection using a UAV. However, to the best of the author's knowledge, no work has been reported particularly for fence line detection in the domain of pastoral farming. As road and power line detection resembles fence line detection to some extent, we will discuss the work done in these

fields.

Merlet and Zerubia [102] used dynamic programming where a cost function was defined and optimized to detect lines in the images. Lee *et al.* [103] used principal component analysis (PCA) to detect straight lines. Their algorithm separated row and column edges for the edge image, which were then labelled and PCA was performed for each labelled edge. Similarly, Guru *et al.* [104] used small eigenvalue analysis for detecting straight lines in an image with a lot of edges. A moving mask was used to scan the image from the top left corner to the right bottom corner. A small eigenvalue of the covariance matrix was then computed at each stage to link the centre points of the lines. Many different methods were used for road line detection such as the particle filter [105], vanishing-point-constrained edge detection [106], RANdom Sample And Consensus (RANSAC) algorithm [107], eignevalues [108], probabilistic Hough transform (HT) [109], Optical Correlator [110], and generalized HT [111].

Power lines detection for the UAV surveillance or inspection is also performed by a few researchers using HT [112], pulse couple neural filter and improved HT [113], HT and K-means [114], to name a few. The HT method is the most commonly used technique for line detection in different domains [115, 116, 117].

However, the detection of fence lines is quite a tricky task. It is a small part of a whole system, where the objects detected outside the fence should be ignored by the system. Also, the presence of livestock and many wooden logs in a few paddocks make it more complex. The above-mentioned techniques can perform well where the lines are well defined, however, the fence lines get blurred at most of the points at the altitude of 80 m. This leads us to the use of a combination of deep learning and machine learning algorithms.

## 2.3 Object Tracking

The main objective of object tracking in a video is to detect a region of interest in a video and keep track of its position and motion. It can also be defined as the approximation of the path of an object throughout the video by keeping track of objects' spatial and temporal changes like variations in position, size, and shape. It has multiple applications in the domain of video surveillance, robot vision, traffic monitoring, wildlife tracking, gesture identification, public security, animation, and marine life observation. Most commonly used algorithms are discussed briefly under this section to get a comparative analysis.

MOT can be done either an online or offline task; an online [118] or real-time tracking uses information from the previous frame to predict objects' motion and is usually used for a real-time application, while offline tracking [119] is usually used for recorded videos that can use the past frames as well as the future frames to finalize objects' motion. Both cases, however, present major challenges, including low-resolution camera noise, complex object contours, occlusion, variation in scale, and blurriness. UAVs can introduce additional challenges such as a sudden shift in the movement of an object due to windy circumstances and increased complexity associated with moving objects and backgrounds. Due to changes in altitude, objects also vary in size, and therefore their distinctive features appear to be lost. Despite these challenges, UAVs are gaining a similar amount of attention in computer vision and artificial intelligence research since they provide images and videos from various perspectives and therefore are better suited for object tracking and detection than fixed position cameras. Researchers are preferring to use the UAV in the research fields like data collection, object detection and object tracking [120].

Depending on the initialisation process, tracking algorithms are classified into two main categories: detection-based tracking [121, 122] and detection-free tracking [123].

The detection-based tracking—or tracking-by-detection—uses a pre-trained object detector as a preliminary step to detect objects in each frame. The detection values—either as a bounding box or a centroid—are then fed to the object tracker to initiate the tracking process. The main task of the object tracker is to predict the motion of the object in a future frame using the information from the previous frames. A data linker links the predicted state with the same detected object depending on different parameters. On the other hand, in detection-free tracking, a location of each object is initialised in the first frame and its features are used to track it throughout the video. Some recent survey papers of MOT [124, 125] present a systematic review of various existing algorithms and highlight the complexities of MOT in handling complex backgrounds and high occlusion.

Along with categorizing MOT based on the initialization and processing method [126], the existing techniques can also be grouped based on the algorithms used in the whole process. The objective of object tracking for this research is to keep track of livestock entering or leaving the video frame to avoid miscalculations. Literature available on object tracking categorizes it into two main categories: machine learning and deep learning. Within machine learning, it can be further subdivided into three broad categories: point tracking, kernel-based tracking, and silhouette-based tracking [127]. A hierarchical model is presented in Figure 2.4 and methods mentioned in this hierarchy are discussed briefly in following subsections.

### 2.3.1 Point-based tracking

In point-based tracking, features are used to detect objects in each frame and this is a simple approach in computer vision. If an object is successfully detected in an image, the point-based tracking algorithm is highly accurate. It can, however, incorrectly identify objects when objects are occluded [128]. The Kalman filter, particle filter and

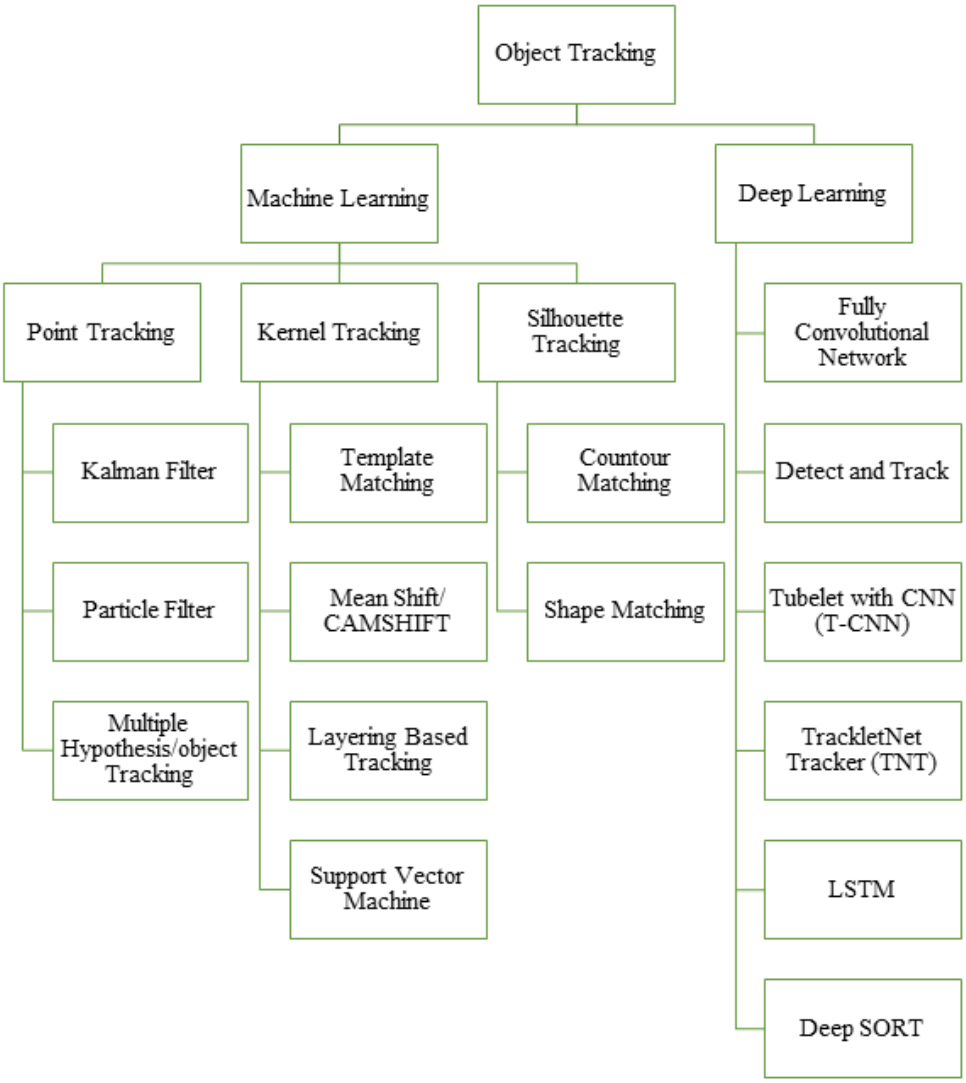



Figure 2.4: A brief hierarchy of object tracking techniques.

multiple hypotheses tracking algorithms fall in this category.

Kalman filter [129, 130] is an optimal recursive data processing algorithm that performs the task of restrictive probability density propagation. It uses a set of mathematical equations to efficiently estimates the state in an iterative manner. Irrespective of the nature of the system, it estimates the past, present and future states of an object. After estimation of the current state, it receives feedback and then recompute the values. The main drawback of the Kalman filter is the assumption that the distribution of state

variables is Gaussian by nature. Thus, it is not suitable for state variables that lack the property of Gaussian distribution. It is still in use along with many hand-crafted [123, 131] as well as deep learning techniques [132, 122] and showed good results where the object is moving, but the video is recorded from a fixed point. However, it is not suitable if the distribution of state variables is not Gaussian, and has difficulty in tracking multiple objects if initial states are computed incorrectly. To the best of the author's knowledge, so far the Kalman filter is used for tracking large objects, that have some defined features. Also, the parameter tuning of the Kalman filter has not been discussed in detail.

To overcome this drawback of the Kalman filter of assuming a normal distribution of all state variables, the particle filter was introduced [133]. This generates all the models for one variable before generating them for other variables and works dynamically in this regard. It uses contours, colour features and texture mapping of objects. The new location of an object is estimated after prediction and correction of previous weights. The particle filter with different modifications is used by many researchers for object tracking [121, 134, 135, 136]. However, this algorithm faces challenges in handling the problem of higher dimensionality, where more than one object needs to be tracked. Multiple object tracking needs a joint solution for data handling and correspondingly state estimation, which is difficult with this technique. So, it is best suitable for single object tracking.

Multiple Hypothesis/Object Tracking (MHT/MOT) [134] is an iterative algorithm that was initially introduced for tracking multiple signals in audio files and then found application in four-dimensional signals. It uses a set of existing track hypothesis for an object and then predicts the object's position in the next frame for each hypothesis. Each prediction is then compared with the original measurements using a distance measure. Depending on the calculated distance measure, a new hypothesis is introduced for the next iteration. It can be used for a new object, that just entered the scene, or a

previously tracked object. If the object has left the video frame then no hypothesis will be assigned to it. As the name predicts, it is used for tracking multiple objects in a video in conjunction with Kalman filter with deep learning algorithms [137] and is capable of handling occlusion and tracking multiple objects. However, it is computationally exponential both in memory and time.

Santosh and Mohan [138] compared the performance of the extended Kalman filter, Gaussian mixture model (GMM), and mean shift algorithm for MOT. Their comparative results showed that GMM can perform well in the case of occlusion and non-linear motion of objects. Hongyang Yu *et al.* [139] used the contextual information around the object of interests and proposed an Exchanging Object Context (ECO) model for online tracking of multiple objects. Siqi Ren *et al.* [140] divided detection results of MOT in three categories: low uncertain, high uncertain and false. To improve the system's performance the false values were penalised, the low uncertain ones were delayed until the end of the video and a tracking tree was constructed for less uncertain detections. Some of the researchers also used a particle filter [121] and multiple hypothesis tracker (MHT) [134, 137] with good tracking efficiency but these algorithms have a high computational cost.

### 2.3.2 Kernel Tracking

Kernel-based tracking [141, 142, 143] tracks the motion of the object from one frame to the next frame using the primary region of the object. Many researchers use this technique for tracking region of interests in consecutive frames. These algorithms are classified according to the method used to track objects, i.e., template, motion approximation, and initial representation of the object. But they also offer a limitation that part(s) of an object may be left outside the boundary of defined shape if it is merging with the background. This category of tracking algorithms includes template tracking,

layering tracking, mean shifting tracking, and SVM tracking.

Template matching searches for the predefined complete or partial template in each video frame to define the trajectories using different positions and rotations. It is a brute force technique, which makes it high in computational cost but is capable of dealing with partial occlusion of the object. The predefined prototype is a template of an object which is initially extracted from any of the video frames. Then small parts of an object or full object are searched in each video frame using all possible positions and rotations. A numerical index is then calculated that identifies how well the template fits in the image. It can deal with the partial occlusion of an object. Horng *et al.* [143] used a template matching technique to design a system for detecting driver's fatigue with 89% precision on test videos.

Layering based tracking is another kernel-based algorithm that is capable of multiple object tracking and can handle full object occlusion. Each layer consists of shape representation, intensity variation and motion parameters. These layers are designed after compensating the background motion and then estimating the object's motion.

Mean shift tracking uses gradient ascent to search for the maximum similarity score of the object in the video frame. This algorithm tries to reach that area of the video frame which is most similar to the initialized object's model using the gradient ascent method and searches for the maximum similarity score for the object under observation. The region of the image is defined as a colour histogram instead of a contour and the target model is represented by a probability density function. The advanced version of mean shift algorithm is Continuously Adaptive Mean Shift (CAMshift) algorithm, which can be used with CNNs for object tracking in videos. CAMShift [144] provides speed and robustness with minimal training and computational cost but the drawback of this algorithm is a lack of robustness when dealing with complex cases.

SVM [145] is a widely used classification method which requires training. After a proper training, SVM gives positive and negative values for tracked and non-tracked

objects in an image, respectively. And it is used successfully with CNN for object classification in images and also for multiple object tracking in videos. In [145], researchers used an SVM tracker as a kernelised structure and achieved tracking with high performance.

### 2.3.3 Silhouette Tracking

Tracking methods based on silhouettes are more suitable for tracking complex things like hands and fingers that cannot be described perfectly by simple geometric features [146]. Tracking objects occurs in the following frame using rough shapes of objects from the previous frame. Tracking methods that come under silhouette category are able to deal with complex shaped objects, occlusions, merging and splitting issues because they use predefined information for each object.

Silhouette tracking algorithms fall into two main categories [127]: contour and shape based tracking. It's the flexibility of these algorithms that makes them attractive because they can handle a very wide variety of objects. However, all objects must be known before tracking so that silhouettes tracking can succeed.

In contour-based tracking algorithms, a contour of the object is designed in the first frame, which is then predicted at a new position in the next frame [147]. Two different approaches are usually used for this purpose: statistics based on appearances like shape or an optimization technique of an energy function like gradient descent or a greedy method. The shape-based tracking, as the name specifies, is a bit similar to the template matching technique, it searches for object template in the existing frame and keeps track between frames. Shape or region based silhouette algorithms are more resilient to noise, while contour-based algorithms are computationally less expensive.

### 2.3.4 Deep Learning Based Tracking

Object tracking incorporated deep learning over the last decade, it is used for object detection, classification, segmentation, and tracking. For the latter part, it is mostly used in conjunction with different machine learning algorithms. A few of the recently proposed algorithms are discussed briefly in this section.

Instead of using any other tracking method along with CNN, Lijun Wang *et al.* [148] used the features of the fully convolutional network. Although they only tracked a single object per video, they were able to overcome many tracking challenges. Like Lijin Wang *et al.* [148], Christoph Feichtenhofer *et al.* [149] also proposed the concept of using a convolutional network to detect and track objects in a video simultaneously, named Detect and Track. Video frames were employed to completely train the network in the end-to-end mode. After computing cross-correlation between feature responses of adjacent frames, ROI pooling layer for classification and regression of proposal boxes were used. Object tracklets were then linked throughout the video using the detected ROI.

Kai Kang *et al.* [150] incorporated the temporal and contextual information of objects and proposed Tubelets with convolutional neural networks (T-CNN) to create an end-to-end deep learning framework for detecting and tracking objects. Their proposed method used R-CNN and Faster R-CNN for general object detection in each frame by using information from tubelets. A similar task was performed by Gaoang Wang *et al.* [119], TrackletNet Tracker (TNT) and in this method the bounding boxes from consecutive frames were linked to create tracklets. Similarly, Zhongdao Wang *et al.* [151] combined an object detection and appearance model to propose a shared MOT algorithm as a single-shot detector.

Kwangjin Yoon *et al.* [152] also discussed the importance of the data association method in tracking-by detection methods, and proposed a deep neural network-based

data association method for MOT. Bounding boxes and track histories were used as inputs to long short term memory (LSTM) networks. The end result was an association matrix which shows a correlation between tracks and all detected objects. An LSTM network was used by different researchers in order to improve the system's performance and accomplishing the MOT as an online tracking system [153, 154]. Similarly many researchers are trying to tackle the MOT tracking problem using recurrent neural networks [155] and CNNs [156, 157]. Shivani *et al.* [158] designed the Deep Simple Online Real Tracking (Deep SORT) algorithm using YOLOv3 and RetinaNet, and detected objects in a UAV recorded video.

Yingkun Xu *et al.* [159] analysed deep learning structures, reviewed deep network based strategies and provided an experimental comparison of tracking results on benchmark datasets. The discussed algorithms were also compared based on robustness, advantages in different conditions and effectiveness under various circumstances. The comparative analysis shows that there is still a room for improvement in order to achieve realistic results using deep learning in the field of MOT, as the highest reported accuracy was 71% by Tang *et al.* [160] on the MOT16 benchmark dataset. Table 2.2 shows a comparison of discussed techniques for object tracking.

As per the literature reviewed, machine learning based algorithms still lead in this field in terms of a system's overall efficiency.

### 2.3.5 Livestock Tracking

Kaixuan and Dongjian [161] used background subtraction as a target detection for the moving cows. This was a single object tracking problem and a fixed camera was used to collect the data. They achieved 88% true detection rate when cows were walking straight under normal illumination conditions and there was a high contrast difference with background. The use of edge refinement with optical flow estimation was proposed

Table 2.2: Comparison of object tracking algorithms.

| Algorithms | Types | Advantages | Limitations |
|---|---|---|---|
| **Point Tracking** | Kalman Filter [130] | - Can track many objects in noisy images | - Assumes Gaussian distribution of all state variables |
| | Particle Filter [133] | - Does not need Gaussian distribution of state variables | - Difficulty in tracking multiple objects |
| | MHT/MOT [134] | - Capable of handling multiple objects as well as to object occlusion | - Difficulty in tracking multiple objects |
| **Kernel Tracking** | Template Matching [143] | - Relatively simple and can deal with partial occlusion | - High computational cost |
| | Mean shift [144] | - Does not need any prior shape information | - Cannot distinguish an object from the background in case of the same colour |
| | SVM [145] | - Suitable for multiple object tracking | - Needs training |
| | Layering Based [128] | - Suitable for multiple object tracking | - High computational cost |
| **Silhouette Tracking** | Contour Matching [147] | - Can handle multiple complex shaped objects | - Computationally expensive |
| | Shape Matching [127] | - Resistant to noise and can handle multiple complex shaped objects | - Need prior information of all shapes and needs training |
| **Deep CNN Based Tracking** | Fully Convolutional Network [148] | - Can handle different challenging factor | - Track one object at a time |
| | Detect and Track [149] | - End-to-end training and failed detections can be recovered | - Computationally expensive algorithm |
| | T-CNN [150] | - Low computational cost and good for tracking a few objects | - Difficulty in tracking many objects in video |
| | TrackleNet Tracker [119] | - Can handle occlusion much efficiently | - Face tracking issues in the fast camera motions |
| | LSTM [153] | - High speed make it applicable for online MOT | - High memory usage and computational cost |
| | Deep SORT [158] | - Can be used for real-time MOT | - Face difficulty in occlusion cases |

in [162] to improve the tracking performance. This proposed method was tested on different KITTI datasets and also on a publicly available livestock (cattle) dataset [163]. Al-Thani *et al.* [164] used Raspberry Pi module V2 on-board of a quad-copter drone to perform online as well as offline processing of the videos for sheep counting and monitoring at 10 m and 15 m altitude. Each image had maximum 9 sheep and they achieved 60% accuracy when counting was done offline, and a bit higher accuracy when it was done online on the drone. However, one of the main issues reported in this case is the high power consumption of the designed system and the other issue was the difficulty in covering paddock efficiently. The paddock coverage at the mentioned altitudes was a difficult task and livestock crossing between frames was unavoidable.

Livestock detection, tracking and counting is performed at altitudes lower than 50 m in the mentioned literature. This makes an efficient UAV flight a challenge too. By efficient, we mean the coverage of whole paddock while avoiding the passing of livestock between frames. which can only be avoided by recording the videos at higher altitudes. Furthermore, the only publicly available dataset has only one object per image and was actually designed for cattle type identification. It cannot be used for this research and our recorded videos have an average of 150 objects per frame that need to be tracked.

## 2.4 Research Gap

Existing hardware-based methods inevitably involve direct contact between farmer and animal, and are therefore classified as manual counting. Manual counting can take time and may be susceptible to a variety of psychological phenomena that can cause error or optical illusion [165]. Although there are machines that can be installed near holding pens to eliminate the need for manual checking, obtaining daily stock information would prove impractical. Therefore, it is imperative to make good use of technology for

the benefit of farmers.

Surveillance or motion sensor cameras can be installed around the whole farm. Assuming a common paddock size of 20 ha, many cameras will be required for proper coverage. This will arise problems of continuous power supply, networking issues for linking all these cameras together and data transmission to a central system, which will cost a lot more than the annual financial loss.

These factors support automatic detection, counting, and remote monitoring of livestock with the help of a UAV. This can give some relief to farmers and relevant information can be extracted in much less time. The design of this automatic system is still complex, even when all the right conditions are in place for data collection. There will be challenges in terms of light and background variations, as well as variations in shadows with changes in weather and sunlight. There is therefore great complexity in formulating a general technique to address these issues, while incorporating various paddock shapes and sizes.

Though there is a large variety in drones, ranging from the ones that can be used for recreational purposes to those that are good for professional use, researchers usually prefer to use UAVs with functions like GPS-enable autopilots, emergency landing technology, stability, good camera resolution, long battery life, and easy flight mode. This gives ease in access to images and videos, and can be easily linked with either smartphones or tablets. The latest UAV can go up to the range of 7 km from the control box and have a flight time of 25 minutes. The Civil Aviation Authority of New Zealand allows UAV flight in the line of sight, which is usually very less than 7 km. The main challenge, which UAVs offer for this research, is the coverage of the whole paddock from a good allowable height and in a minimum possible time. This coverage time can vary from paddock to paddock as some paddocks have plain ground with no trees while others have uneven terrains and many bushes. The current research is focused on those paddocks which have flat terrains and have a minimum number of trees. Later on, it

can be extended to all others. Currently, no large dataset is available for farm animals, so, creating a good and large dataset will be an important contributions towards this research.

According to literature, livestock aerial data collection is most effective at an altitude lower than 80 m; however, such videos are difficult to cover a paddock area efficiently. In farms that spread over many hectares, it is often not possible to capture the width of the entire paddock in a single frame. Because such farms also have large paddocks and a video needs to be taken from a height where the width of the paddock can be captured. As a result, we have collected data as high as 120 m in order to make using of UAVs for livestock counts more convenient for farm managers.

In addition, it is essential that the livestock counting system is as accurate as possible so that the estimated count is as close to 100% as possible to the true count. This is due to the fact that any 1% error is the equivalent of 10 sheep on a herd of 1000, and this is a significant monetary loss. The previously mentioned systems are incapable of achieving such high accuracy when handling small objects.

Deep learning and machine learning are two key fields for dealing with image-processing tasks. We will utilize deep learning and machine learning techniques to detect and track animals, respectively. While the detection system needs a large dataset to be trained, once trained, it has great potential in detecting and classifying objects. Additionally, since the designed network can be trained using aerial videos of the paddocks, it can be used for other animal detection. Our goal is to detect and track hundreds of animals per frame, each comprises a mere 10 pixels and appearing like a colored blob above 80 m. However, currently available algorithms deal with images and videos with only a few animals per frame and 100s of pixels per animal. The task of tracking multiple objects has been performed by many researchers using the Kalman filter and and different data association methods [118, 122, 123, 166, 167]. To the best of the author's knowledge so far, the Kalman filter is used for tracking large objects,

that have defined features. Furthermore, as farm fence detection has not been done in pastoral farming, it is a new research area. From the mentioned altitudes the fence lines look very vague and we mostly have to guess if there is a fence line or not. Though there are many line detection algorithms, due to the overall scene complexity, those algorithms cannot be used for fence detection.

The literature review reveals that counting and tracking hundreds of objects, along with fence detection from the mentioned altitude, especially farm animals [34], has not been targeted by any researchers until now.

# Chapter 3

# System Modelling and Data Collection

A literature review has revealed that there is a need for a robust, cost-efficient and comparatively easy-to-use livestock counting system to help pastoral farm managers. In New Zealand, sheep rustling has the highest rate among all farm rustling cases, so this research work is focused on sheep farms. Designing such a system needed a complete understanding of farms and the respective limitations. This issue was handled by collecting data from different farms, in different seasons and weather. As the research proceeded each year, the process of data collection and dataset design was matured, and was used in our different research articles. However, the details of data collection and only the finalized datasets are presented in this chapter. These finalized datasets were used in the following manuscripts:

**C2:** F. Sarwar, A. Griffin, S. U. Rehman, and T. Pasang, "Towards detection of sheep onboard a UAV," *arXiv preprint arXiv:2004.02758*, 2020.

**C4:** F. Sarwar, A. Griffin and T. Pasang, "Tracking livestock using a fully connected network and Kalman filter," *Geometry and Vision. ISGV 2021. Communications in Computer and Information Science,* vol 1386, pp. 247–261, 2021.

**J1:** F. Sarwar, A. Griffin, S.U. Rehman and T. Pasang, "Detecting sheep in UAV images," *Computers and Electronics in Agriculture*, vol. 187, 2021.

**C5:** F. Sarwar, A. Griffin, P. Chong and T. Pasang, "Pasture fence line detection in UAV videos", *in the proceedings of 36th International Conference on Image and Vision Computing New Zealand, (IVCNZ)*, 2021.

## 3.1 System Model

This section focuses on the design methodology for this research work. Design science is used as a research methodology, which is an iterative research method. In this method, the researcher keeps on improving the system after successful evaluation until the desired performance is achieved. Figure 3.1 shows the overall structure of this study. Quantitative evaluation, system modelling and simulation form an iterative process that helps in improving the output of the system. The most important parameters for evaluating the system are precision, cost, time and memory efficiency, multi-scale animal detection, and most importantly, implementation feasibility. These things will cover the research gap in this study that was raised after the literature review and will also keep up the motivation for improving the system design and respectively, the required results.

Figure 3.2 shows the main steps of research design flow followed in this thesis. Categorically the whole research can be divided into five main parts: data collection, datasets design, livestock (sheep) detection, fence detection and sheep tracking. The first step was to record the paddock videos using a UAV. Multiple RGB frames were

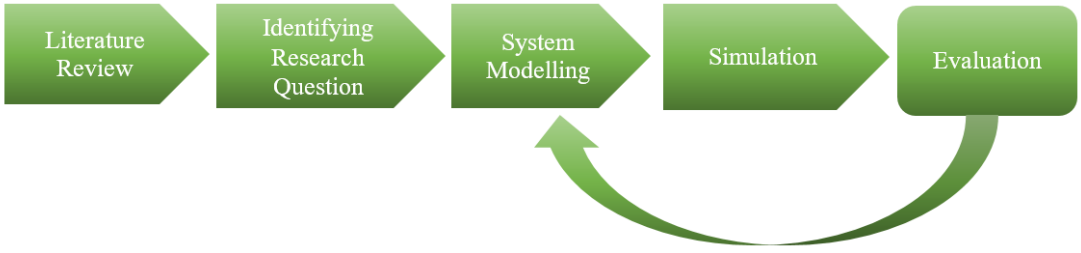



Figure 3.1: Research methodology.

extracted as paddock images, and ROIs were labelled in all these images. As there were two main tasks of object detection, i.e. sheep detection and fence detection, the ground truth labelling was required for both of these cases. From each frame, sub-images of dimension 256 × 256 were extracted as per respective object (sheep or fence). Datasets were designed for training the deep CNNs targeting each task of the system. The livestock tracking task was then expected to be performed within fenced paddocks to avoid unnecessary detection outside the fences.

The main focus of this chapter will be on data collection and datasets design, whereas the details of livestock detection, fence detection and livestock tracking are discussed separately in later chapters.

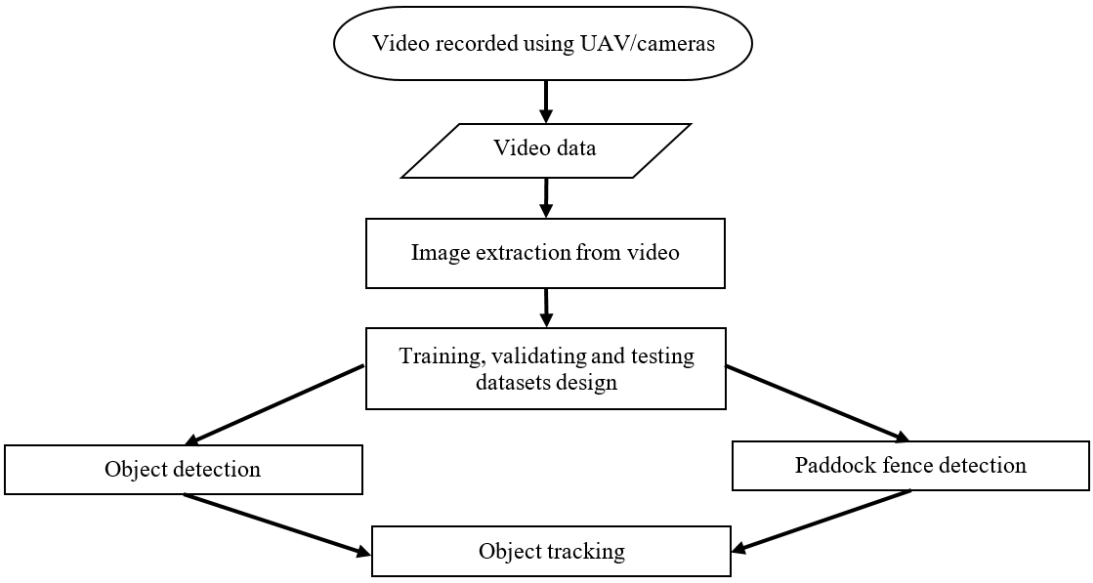



Figure 3.2: Research design flow.

## 3.2 Data Collection

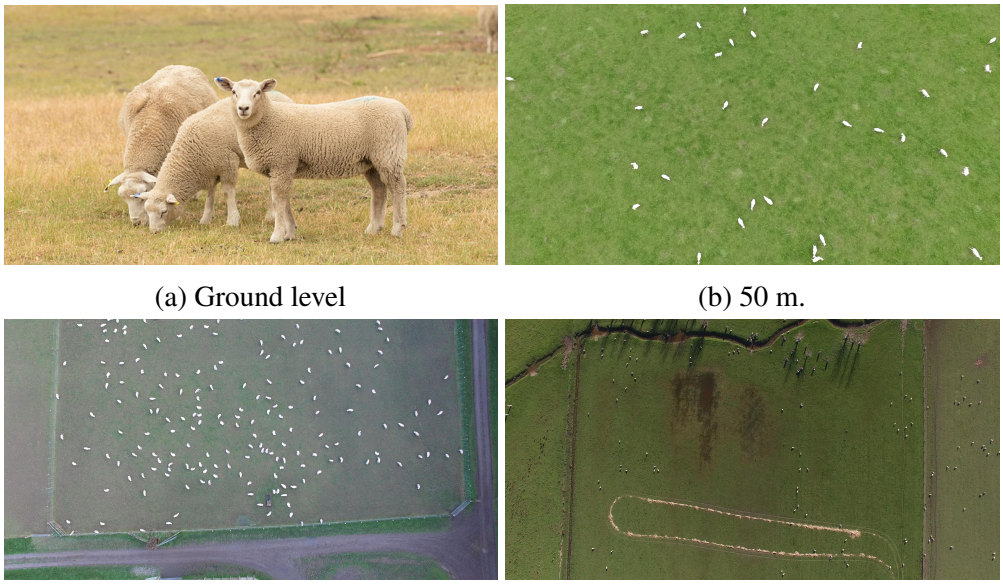
A DJI Phantom 3 Pro was used for recording the paddock videos, shown in Figure 3.3. This UAV can be flown at a maximum speed of 16 m/s and has a flight time of approximately 23 minutes. The controller has a USB port that can be attached to any

tablet and smartphone for an image or video output. It comes with an already attached camera, which has $1/2.3$" CMOS sensor and lens of FOV $94^{\circ}$ 20 mm. The videos can be recorded in ultra high definition (UHD), full high definition (FHD) and high definition (HD) mode.

The data was collected from different sheep farms of Palliser Ridge, Wellington, New Zealand. It has a total area of 1300 ha, has multiple paddocks, and holds approximately 8,000 sheep and 1,700 cattle. The UAV was used to record the UHD videos in the sheep paddocks only. Initially, three different altitudes were used i.e. 50 m, 80 m and 120 m, where 120 m is the maximum allowable altitude as per the New Zealand Civil Aviation Authorities. Data was collected on multiple days and videos were recorded during different weather conditions and times of day. This technique of data collection helped in capturing a variation in the background and illumination naturally. Figure 3.4(a) shows a sheep image that looks like a whitish blob in (b), (c) and (d) in the aerial views from respective altitudes.

Figure 3.4 shows that the higher the altitude, the smaller the sheep size, as well as the lower the quality of the sheep details in the captured imagery. Thus, it is quite tempting to collect data at a lower altitude like 50 m. Indeed, at this height, sheep are quite easy to distinguish. However, at a lower elevation, it is much more complicated to capture an entire paddock properly, and avoiding sheep movement between passes

Figure 3.3: DJI Phantom 3 Pro.



(a) Ground level (b) 50 m.

(c) 80 m. (d) 120 m.

Figure 3.4: Sheep images taken by the UAV at the mentioned altitudes.

is not possible. A UAV can mostly fly from one end to another in the entire paddock efficiently from 80 m and above, and the fences on either side can be seen on video in most cases. It was discovered that the UAV was disturbing for the sheep at lower altitudes but an object detector detects the sheep accurately. Additionally, it was found that the UAV can be used to track the sheep between frames once they are detected at the desired altitude.

This research aims to make livestock counting as robust, non-disruptive, and less time-consuming a task as possible, which means that compromises had to be made between the details that can be captured in each frame and other factors. An altitude of 80 m and above was decided to keep it non-disruptive for animals and the video recording time for larger paddocks was still for a few seconds. All data was collected at 80 m and 120 m altitude and each frame captured approximately 144 m × 76 m and 216 m × 114 m area on the ground for respective altitudes. Table 3.1 shows the necessary details of the UAV and data collection.

Table 3.1: Details of data collection and UAV.

| | Parameters | Details |
|---|---|---|
| **Farm** | Location | Pirinoa, New Zealand |
| | Sheep Breeds | Romney, Texel |
| | Grass Types | Rye, Clover, Chicory, Cocksfoot, Plantain |
| | Total farm area | 1300 ha |
| | Average paddock area | 20 ha |
| | Sheep per paddock | 350 approx. |
| **Data Collection** | Drone | DJI Phantom 3 Pro |
| | Dates | 2018, 2019, 2020 |
| | Weather | Cloudy and sunny |
| | Video time | 10 to 20 s (each) |
| | Drone altitude | 80 m and 120 m |
| | Area per frame | 144 m × 76 m (80 m)<br>216 m × 114 m (120 m) |
| | Ground spatial resolution | 35 cm (80 m)<br>52 cm (120 m) |
| **Camera Specifications** | Sensor | $1/2.3$" CMOS |
| | Lens | 20 mm focal length |
| | Horizontal FOV | $82.4°$ |
| | Vertical FOV | $61.9°$ |
| | Video recording mode | UHD: $4096 \times 2160$ p |
| | Frame rate | 30 fps |

## 3.3 Image Datasets

Irrespective of the object of interest, the main requirements for designing a useful dataset is to answer a few very important questions like:

- For which kind of network should it be designed?
- What kind of ground truth data is needed?
- What should be the size of the training images?

Different networks were initially tested for sheep detection [168, 169], that needed different kind of ground truth labelling. One type is bounding boxes around all ROIs in each training image, which has four values: (x,y,w,h), referring to (x coordinate, y coordinate, the width of the object, the height of the object). The other type of ground truth is a centroid for the respective object which is just a centre point of the bounding

box too.

As the sheep and fence posts both look like tiny objects in the aerial videos, the centroid based ground truth looked more appropriate and after the first year of research, only latter values were used. For both sheep and fences, centroid points were assigned manually and bounding boxes were generated using a simple algorithm. The task of ground truth labelling spanned a few months for both objects. This task was not done at once and the dataset was increased as more videos were recorded every year.

Defining a bounding box or centroid for a sheep was easier at 80 m as it has a defined blob shape. However, at 120 m it was a very difficult task because at times the sheep were difficult to distinguish from the background. Furthermore, two or more sheep standing close to each other appear as one big whitish blob with no intermediate boundary. One such case can be seen in Figure 3.5, which shows the cropped sub-images from frame numbers 1, 50 and 100 from a video recorded at 120 m. The group of sheep in the top left corner that looks like to have five sheep in the first frame has six sheep, as confirmed in the 100-th frame. In such cases, we had to update the data of the previous 100 frames to correct the labelling task.

Figure 3.6 shows a few examples of paddock fences that are commonly seen in New Zealand. The wooden posts were expected to be visible at the desired altitudes but the wires are a bit difficult to identify even in the shown images. This made the ground truth

(a) Frame #1 (b) Frame #50 (c) Frame #100

Figure 3.5: Cropped frames from a 120 m video.

labelling for the fences even trickier at 120 m, as in most of the cases, the fence lines merged completely with the background. It can also be seen in Figure 3.4 (c) and (d) that the top view of the wooden fence posts is a tiny line in each image and there is a vague fence line that needs to be identified correctly. Figure 3.7 gives a closer look of the fence lines as seen in the dataset at 80 m and 120 m, respectively. We had trouble guessing the wooden posts and wires in most of the images at 120 m. For example, in Figure 3.7 (b) the fence posts and wires are hardly noticeable in first and second image. However, we completed this task and the approximate centre of each wooden post was used as the centroid at both altitudes.

There are two ways of creating an effective dataset with a large number of images, either capture a high number of paddock images using the UAV or use data augmentation efficiently for good dataset creation. The second option was used and for each category, thousands of training images were generated using only a few full-sized images per dataset. Translation of 10 pixels as well as rotation at 90, 180 and 270-degree angles were used as data augmentation methods.

The main dataset for each altitude was created by extracting RGB frames as images from the respective paddock videos. The images had a resolution of 4096 × 2160 pixels and cannot be used directly for the training of CNN network(s). Different datasets were designed for each altitude. Videos recorded at 80 m altitude were used to create the dataset that can be used to train and test the networks for respective object detection at 80 m, and the same process was repeated for 120 m. All images were down-sampled

Figure 3.6: Examples of a conventional paddock fence.

(a) 80 m.

(b) 120 m.

Figure 3.7: Fence images taken by the UAV at the mentioned altitudes.

by two and were divided into many overlapping sub-images using a commonly used $256 \times 256$ pixels dimension. Various data augmentation techniques, like translation and rotation at three different angles, were used to create an effective and large dataset. For any kind of CNN, each training image should have an ROI in it for proper training. The training, validation and test sets were designed with a ratio of $80 : 10 : 10$ respectively. The network that gave the best results was used for fence detection and later as a backbone for object tracking task in each frame of the respective video.

Four different datasets were designed as shown below:

1. SheepSet80: Dataset of livestock (sheep) at 80 m.
2. SheepSet120: Dataset of livestock (sheep) at 120 m.
3. FenceSet80: Dataset of fence at 80 m.
4. FenceSet120: Dataset of fence at 120 m.

Ground truth labels of both types were created for each one. Irrespective of the ground truth value types, these datasets will be referred to with the above-mentioned names when needed. The details of each dataset along with the number of images in training, validation and test set are shown in Table 3.2. The reason behind the variation in the total numbers is that in some cases more than half of the frame was wasteful as it does not contain any sheep or fence. Only those sub-images were kept in the datasets that have at least one respective ROI in them.

A few centroid based annotated samples from each training set are shown in Figure 3.8. In each image, the red dot shows the centroid of the respective object. The difference in the size of sheep and fence is very obvious in these images, and the respective object can hardly be seen under the centroid in Figure 3.8 (c) and (d). Last image in Figure 3.8 (c) shows one of those cases where it was difficult for the author to detect sheep properly for ground truth labelling as the bleached out wooden logs resemble the sheep at a few places.

Table 3.2: Details of sheep and fence datasets.

| **Name** | **Actual Images** (4096 × 2160× 3) | **Images** ($256 \times 256 \times 3$) | | | |
|---|---|---|---|---|---|
| | | Training | Validation | Testing | Total |
| SheepSet80 | 12 | 18,195 | 2,274 | 2,274 | 22,742 |
| SheepSet120 | 20 | 14,198 | 1,774 | 1,774 | 17,746 |
| FenceSet80 | 12 | 49,867 | 6,233 | 6,233 | 62,333 |
| FenceSet120 | 12 | 30,904 | 3,862 | 3,862 | 38,628 |

## 3.4 Video Datasets

The object tracking algorithm used in this research is from the machine learning domain and does not need training or validation. However, a few videos were used to tune this algorithm for tracking tiny objects between frames and a few more were used to verify if the tracking algorithm was adjusted correctly. As all sheep look alike from 80 m

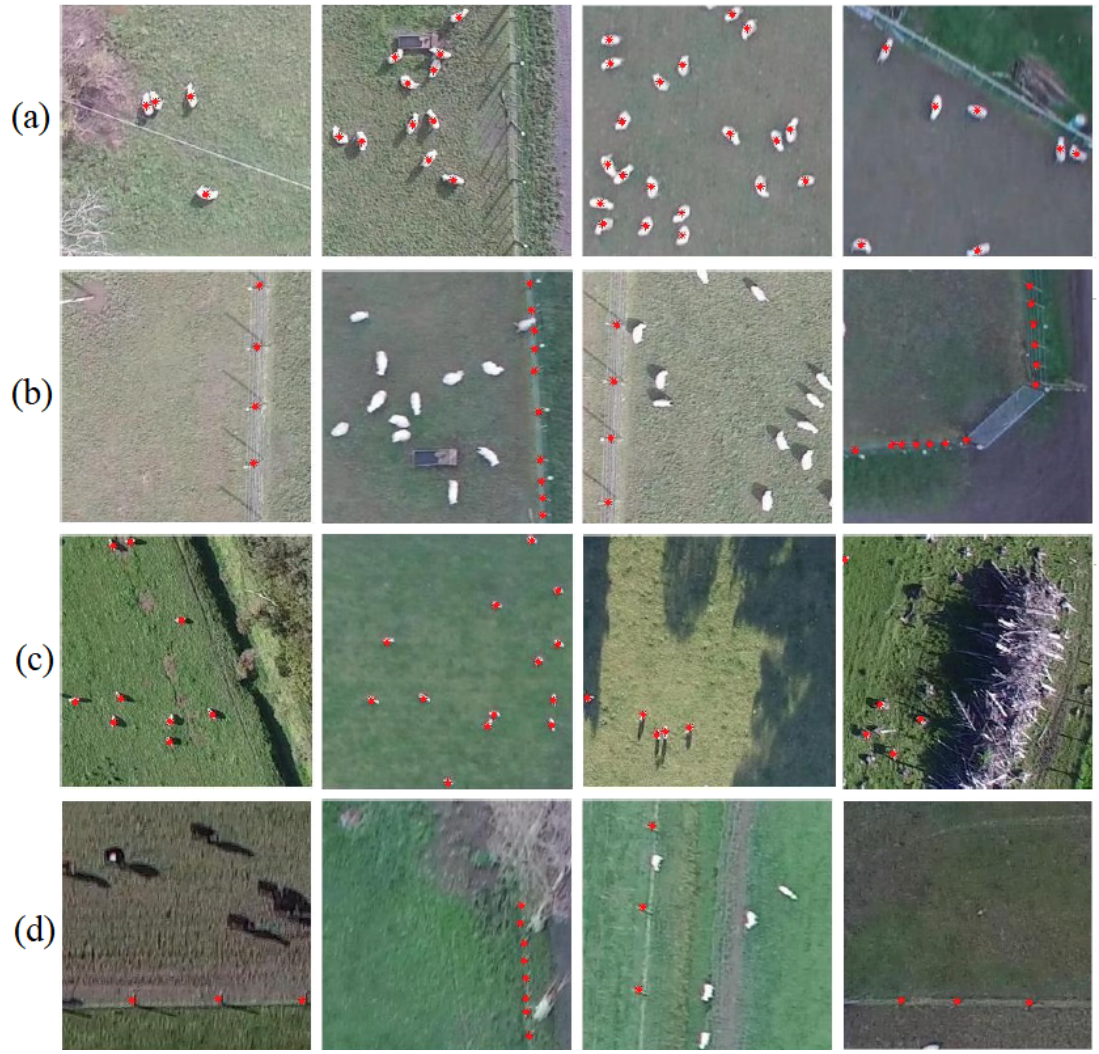



Figure 3.8: Examples of annotated training images from (a) SheepSet80, (b) FenceSet80, (c) SheepSet120 and (d) FenceSet120.

and 120 m, five videos were recorded from different paddocks. Each paddock had 300 to 400 sheep in it and each frame had a headcount of 100 to 250. Even with a fewer number of videos the cumulative sheep count in the videos recorded at 80 m and 120 m were 34,843 and 76,786 respectively. The details of the video data are summarized in Table 3.3.

Two of these videos were recorded at 80 m altitude in the same paddock in cloudy and sunny weather and were labelled as 80a and 80b respectively. Other four videos, 120a, 120b, 120c and 120d, were from 120 m altitude, where first three videos were

recorded in cloudy condition. The last video, 120d, was recorded in sunny weather, and it was the most challenging video in terms of ground truth labelling, object detection and object tracking.

Table 3.3: Details of recorded video and respective sheep count.

| **Video name** | **No. of frames** | **Total count** | **Sheep in paddock** |
|---|---|---|---|
| 80a | 112 | 15,045 | 352 |
| 80b | 137 | 19,798 | 352 |
| **Total** | 249 | 34,843 | 704 |
| 120a | 119 | 21,458 | 276 |
| 120b | 180 | 18,501 | 254 |
| 120c | 150 | 24,373 | 362 |
| 120d | 159 | 12,454 | 203 |
| **Total** | 459 | 76,786 | 911 |

# Chapter 4

# Livestock Detection

In this chapter, the work done for sheep detection in paddock images is explained in detail. For a better understanding of the object detection networks, the mathematical details of a CNN architecture are discussed. Currently, multiple algorithms and CNN networks are available to detect objects in images. As our objects—sheep—are tiny, a careful study was carried out to select a network for this task. The performance was improved by varying some hyper-parameters and methods to use the output provided by the network. Most of the material mentioned in this chapter was published in different articles [168, 169, 170]. The said manuscripts were published as:

**C1:** F. Sarwar, A. Griffin, P. Periasamy, K. Portas, and J. Law, "Detecting and counting sheep with a convolutional neural network," *in proceeding of 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2018, pp. 1-6.

**C2:** F. Sarwar, A. Griffin, S. U. Rehman, and T. Pasang, "Towards detection of sheep onboard a UAV," *arXiv preprint arXiv:2004.02758*, 2020. *(presented in the invited talk at British Machine Vision Conference (BMVC) workshop, 2019).*

**J1:** F. Sarwar, A. Griffin, S.U. Rehman and T. Pasang, "Detecting sheep in UAV images," *Computers and Electronics in Agriculture*, vol. 187, 2021.

## 4.1 Convolutional Neural Network

A CNN makes the task of feature extraction and classification an end-to-end process. The main components used for a basic structure are explained in this section. Feature extraction is usually carried out using different combinations of convolutional (CONV), pooling and nonlinear layers. Classification is then performed by the fully connected (FC) layers and a loss function.

### 4.1.1 Convolutional Layer

A simple convolution layer convolves a filter or kernel with the input image or with the output of the previous layer in the following way:

$$O[x,y] = \sum_{i=1}^{m}\sum_{j=1}^{m} I[x-i, y-j]K[i,j], \tag{4.1}$$

Or in simple form,

$$\mathbf{O} = \mathbf{I} * \mathbf{K}. \tag{4.2}$$

Where $m$ is the kernel size, **O** is the output feature map, **I** is the output from the previous layer or the input image and **K** is the kernel or filter. **I** is convolving with **K** in Equation 4.1 and 4.2 to give output feature map **O**. The smallest size of the kernel can be 3 × 3, and it can go up to 21 × 21. However, the higher the size, the more time and memory it will take to compute the features. The values of these kernels can be initialised either randomly or using normal/Gaussian distribution. The features such as edges, colour contrast, object's contour are extracted at this convolution operation. Conventionally, the low-level features are captured by the first layer, where subsequently added layers are used to extract high-level features of the object and its surroundings. There are further two types of convolution operation depending on whether the dimensionality of the feature matrix should be reduced or kept the same in

comparison to the input. This difference in dimensionality is achieved by using different padding values while performing convolution.

### 4.1.2 Pooling Layer

The pooling layer is used to reduce the spatial size and number of parameters of the feature matrix, which was obtained after the convolution operation. This reduction method helps to design a condensed feature map and decrease the computational power required for data processing. Furthermore, it extracts those dominant features that are invariant to rotation, translation and position, thus helps in training the model more effectively. There are different types of pooling layers, i.e. max, average, and L2 pooling. The max pooling method returns the maximum value from a selected part of an image, average pooling computes the average of all values from the same selected image portion and L2 pooling takes L2 norm of the selected part. All these layers reduce the dimension of the feature matrix and also help reducing the noise component to some extend. Average pooling layer was mostly used when CNNs found their way in image processing but was then replaced by max pooling layer, which gives a better performance [171].

Commonly a 2 × 2 max pool layer with a stride of 2 is used. The following equations show the working of max, average, and L2 pooling respectively.

$$P[x,y] = \max_{1\leq i\leq n} \max_{1\leq j\leq n} O[i,j] \tag{4.3}$$

$$P[x,y] = \frac{1}{2n} \sum_{1=1}^{n} \sum_{1=j}^{n} O[i,j] \tag{4.4}$$

$$P[x,y] = \sqrt{\sum_{1=1}^{n} \sum_{1=j}^{n} (O[i,j])^2} \tag{4.5}$$

Where $n$ is the dimension of the max pool layer, $O$ is the input feature map and $P$ is the resultant feature map after the pooling layer. A combination of convolutional and pooling layers usually fulfill the requirement of one layer of a CNN. However, depending on the data complexity, captured details requirement and processing power of the machine, more supporting layers can be used.

### 4.1.3 Activation Layer

The output received from each layer is passed through an application appropriate activation function before making it an input for the next layer. Various activation functions are available and are used as per the requirement of the overall system. One of the main reasons for using these activation functions is to help the network learn some complex patterns from the data. They introduce another factor of non-linearity, otherwise the CNN will act like a linear regression model and will have limited learning power. The main task of this activation function is to map the input values to a certain narrow range or remove negative values.

The most commonly used activation functions are sigmoid, hyperbolic tangent and rectified linear unit (ReLU). Sigmoid functions are very commonly used in logistic regression and simple architectures of neural networks. For deep learning, the use of a sigmoid function is not preferred due to its drawback of vanishing gradient for bigger networks. As it narrows down the output of each layer to some range, there is an issue of information loss as derivative will be computed over a small range of values. As more layers will be added, more information will be compressed and lost at each layer. This loss is amplified at each layer and results in major overall data loss. Starting with integer values in the first layer and ending up with values less than 1 at the end will not be revertable. When the system tries to update the parameters of the whole network in the back pass of the backpropagation step, then it will not be able to update the

parameters of the starting layers properly. Hence, the main low-level features will not be extracted from the input image up to the requirement. In short, for larger networks, it will compress the information to an extent that any update through the backpropagation step will not update the parameters completely.

A sigmoid function always has positive output for any input values and is not zero-centered. It is not considered ideal for many neural networks. The tanh activation function covers this problem of not being zero-centered by moving the output range between -1 and 1 but still faces the problem of vanishing gradient [54]. However, it is considered to be more optimal as compared to the sigmoid function. The use of ReLU completely removes the problem of limiting the output to a narrow range and acts as a hard-limiter. The ReLU layer has a very simple equation of $max(0, x)$ at each pixel location to remove the negative values. ReLU also adds non-linearity in a CNN model, and increases the speed of training and testing of any model.

### 4.1.4 Fully Connected Layer

The last layer in an object detection network is usually a fully connected layer, where the features from the last layer are flattened and fed to a simple neural network for the task of classification. As the name specifies, each neuron of this layer is connected with every neuron of the previous layer. There can be one or multiple layers, but the last one should be of the same size as is the number of detectable object classes. In training, a loss function layer is used after this to compare the predicted and actual class of the input. There are many loss functions i.e. square, absolute, logarithm, average, cross-entropy, exponential, focal, weighted Hausdorff distance and 0-1 loss function, and the most commonly used loss function is cross-entropy. As it works with the probabilities of predictions, so the loss function usually follows a softmax layer after the last fully connected layer.

### 4.1.5 CNN Training Method

The CNN network designed by using different combinations of layers mentioned earlier is trained iteratively to optimize the loss or objective function. There are usually two main steps for network training, a forward pass and a backward pass. In the forward phase, the input is fed to the network, is passed through different layers of the network and estimated output is computed. As it is supervised learning, the output is then compared with the actual output and an error value is computed using an application-specific loss or optimization function. Then the backward pass is started, where the error is propagated back to update the weights or network parameters using gradients.

There are many optimization algorithms like gradient descent, Adagrad, Adadelta, Adam, AdaMax and Nadam to name a few. Among all, gradient descent and its variants are the most popular and commonly used algorithms to optimize deep CNNs. Currently, state-of-the-art deep learning libraries give the option of selecting an optimization algorithm of the user's choice.

Gradient descent minimizes the loss through an objective function. The parameters of the whole network are updated in order to reduce the error between actual values and outputs estimated by the network. There are two more variants of gradient descent, stochastic gradient descent (SGD) and batch gradient descent. These three algorithms differ in terms of the amount of data or samples used for computing the gradient and the time taken accordingly. The convergence rate of SGD is faster than gradient descent. In gradient descent, the full training data is used in each iteration and cannot be used for larger datasets. However, SGD uses some training data to update the network parameters for the current iteration.

## 4.2 Object Detector

In this research, we used both one-stage and two-stage DCNN based detectors. Object detection in one-stage is handled as a regression problem, in which the pixels of training images are used as input data. The bounding box coordinates of the detected object(s) and the corresponding class probability are the main outputs of such detectors. In this class, the method of dealing with the multi-scale aspect involves either dividing the input image into grids or using anchors.

As part of the two-stage detection algorithm, each training image is used to generate thousands of positive and negative region proposals in the first stage. After these region proposals have been designed, they are used in the second step for feature extraction, bounding box regression, and classification. Both classes of detectors differ in the speed at which they learn to infer, their training methodology, and performance.

We propose a one-layered and a seven-layered two-stage CNN network and also used a one-stage U-Net model [169]. For performance evaluation and comparison, AlexNet [54] GoogLeNet [172], VGG16, VGG19 [173] and ResNet50 [174] are also fine-tuned for livestock detection.

### 4.2.1 One-Stage Detectors

In the approach used by Ribera *et al.* [62, 175], the fully convolutional network was trained without using grids or anchors. Instead of bounding boxes, they used centroids of objects as ground truths and returned a probability map and a regression term. This slight change in ground truth definition makes it more convenient to handle objects with complex shapes, small sizes, and overlapping areas. The estimated probability map of the input image gives higher probabilities at the location of the detected objects.

In order to calculate the training loss, the modified weighted Hausdorff distance (WHD) is used as the distance function between the estimated and actual centroids.

An L1 loss function is used to estimate the number of objects using the feature map from the deepest layer and the probability map. The FCN model used by Ribera *et al.* [62]—also known as a U-Net model [176]—is similar to the one shown in Figure 4.1. The blocks shown in the lower part of this figure show the layer-wise configuration in the respective part of the network.

Each layer in the first block—the contraction block—has repeated application of 3×3 CONV layers with padding value of 1, each followed by a batch normalization operation and an ReLU layer. After the second ReLU layer, a 2×2 max pooling layer with stride 2 was applied for downsampling. This operation of downsampling reduces the dimension of each feature channel matrix where the number of feature channels depend on the number of filters used at the respective layer. For example, 64 filters were used in the first layer, which resulted in 64 feature channels at the output of this layer and downsampling from max pool reduces the matrix dimension from 256×256 to 128×128. From first layer onward, the feature channels double up every consecutive layer until

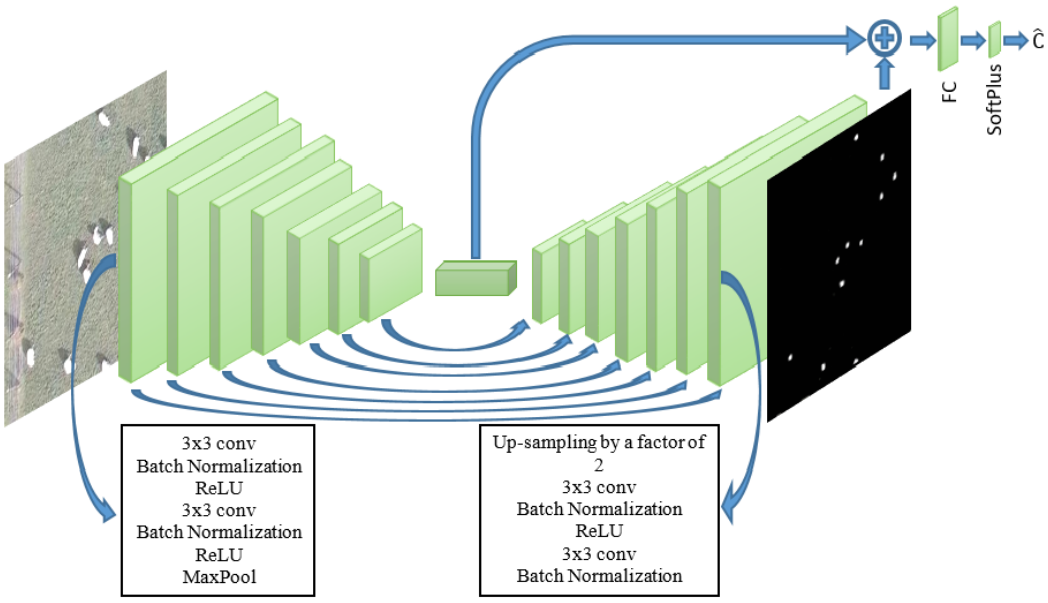



Figure 4.1: The architecture of the U-Net network.

the third layer and remain at 512 in subsequent layers. However, the dimension of each feature channel was halved by downsampling and the deepest layer has 512 feature channels with dimension of 1×1. If the input image has a different dimension then these values will also vary accordingly.

The same architecture is used in the opposite direction in the expansion block to go from 1×1×512 features to a full feature map of 256×256×1 dimension. In this expansion block, the features from the previous layer pass through bilinear upsampling process and are concatenated with the feature map from the contraction block. These concatenated features then pass through repeated application of 3×3 CONV layer, each followed by a batch normalization and an ReLU layer. Last, we map the 128-component feature from the second-to-last layer as an output probability map using a $1 \times 1$ convolution layer.

Concatenating the deepest layer features and the probability map allows an estimation of the number of the detected objects. It is then processed by an FC layer, which produces a single output $g$. The Softplus function is used to rectify this output of the FC layer and the number of detected objects are estimated as follows:

$$\hat{C} = \log(1 + e^{g}). \tag{4.6}$$

In the case of object estimation not being needed, this last step can be eliminated without affecting the performance of detection. The object count can also be determined based on the centroid values of detected objects.

The WHD proposed in [62] is as follows:

$$L_T(p, Y) = \frac{1}{|X| + \epsilon} \sum_{x \in \Omega} p_x \min_{y \in Y} d(x, y) + \frac{1}{|Y|} \sum_{y \in Y} \min_{x \in \Omega} \frac{d(x, y) + \epsilon}{p_x^{\alpha} + \frac{\epsilon}{d_{\max}}} + L_1(C - \hat{C}), \tag{4.7}$$

where

$$|X| = \sum_{x \in \Omega} p_x. \tag{4.8}$$

Here $p_x \in [0, 1]$ and is the probability of an estimated object at pixel $x$, $X$ are the estimated object locations, $Y$ are the actual object locations, $d(x, y)$ is the Euclidean distance between $x$ and $y$, and $d_{\text{max}}$ is the maximum distance between extreme points of $X$ and $Y$. The quantity $\epsilon$, which was set to $10^{-6}$, is a correction factor to avoid infinite loss in case of no estimated objects.

In the first term of Equation 4.7, the multiplication of $p_x$ penalizes the cost function in those areas of the image where there is no ground truth value but the network has given high probability value. Furthermore, to reduce the effect of the distance function values between ground truths and those estimated points where there is no object, the second term is divided by $p_x$. The value of $\alpha$ used is greater than 1 to make the second term larger. If this value was set to 1 then the network does not search for objects which were not detected in the previous training step. After a few experiments, this value was set to 4 during the training phase of U-Net model to keep a balance between recall and precision.

The third term in Equation 4.7 is a smooth L1 loss function and is defined as follows:

$$L_1(l) = \left\{ \begin{array}{ll} 0.5\, l^2, & \text{for } |l| < 1 \\ |l| - 0.5, & \text{for } |l| \geq 1 \end{array} \right\}. \tag{4.9}$$

### 4.2.2 Two-Stage Detectors

Originally proposed by Girshick *et al.* [50], R-CNN consists of three modules: defining region proposals by selective search [177], extracting regional features from these proposals using CNN, and classifying the region proposals using SVM. In this study, a cross-entropy loss function, instead of an SVM classifier, is used to calculate the confidence values of the generated region proposals.

The network used by Girshick [50] has four CONV layers and two FC layers. In order to observe the impact of network architecture on performance metrics, we

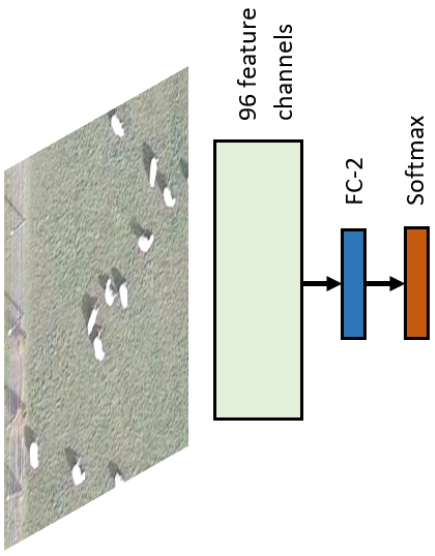



Figure 4.2: The proposed one layered network, Network-I.

designed different networks. The first network—named Network-I—had only one CONV layer with 11×11×96 kernels followed by an ReLU layer and one FC layer along with a softmax function, and is shown in Figure 4.2. This is the simplest architecture that can be used for object detection in an image and was designed to test whether smaller networks can effectively detect small objects. Performance metrics of networks with different topologies, variation in CONV and FC layers, and different kernel sizes were presented in [168]. However, except for Network-I, training and testing was done on smaller datasets, and all details with respective results are presented in Appendix A.

In lieu of increasing the size of the training images, a more complicated network—named Network-II—with seven layers was proposed, as shown in Figure 4.3. This network is designed for objects that mainly exhibit contour information. Such objects may not have much information other than their shape and color. Each layer in the network has a block of $3 \times 3$ convolution with stride 1, batch normalization, an ReLU layer, and a maxpool layer. The maxpool layer has a stride 2 and it helps in halving the dimension of feature channels. The features keep on getting denser towards the end

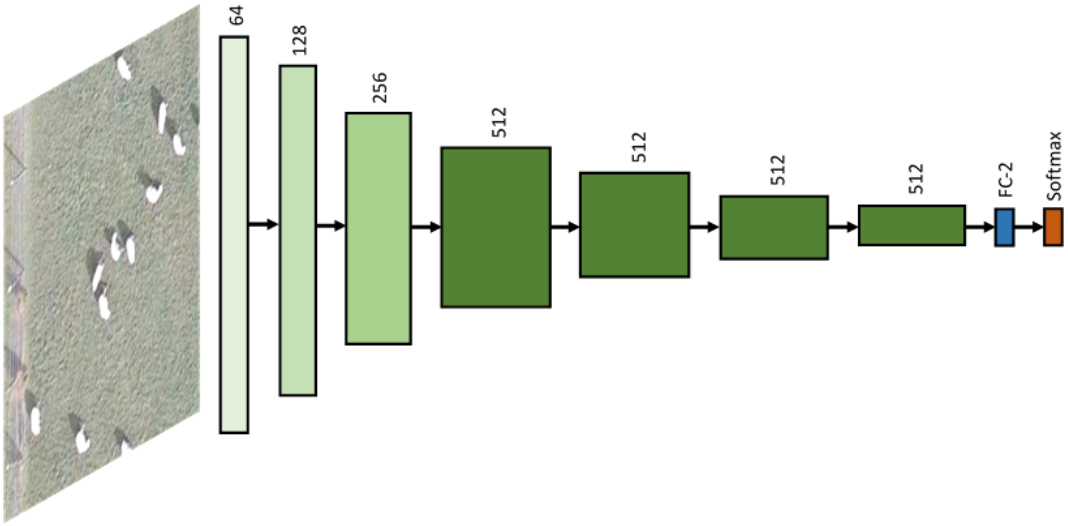



Figure 4.3: The proposed seven layered network, Network-II.

of the network and the dimension of feature channels is halved after each layer. Up until the fourth layer, the number of feature channels was doubled and then remained constant at 512. Designed to separate backgrounds from objects, the last FC layer has a dimensionality one greater than the number of object classes being detected. This layer is then processed by the softmax layer, which tries to classify the data it retrieves from the FC layer.

Training and testing images are not restricted in terms of their dimensions. Although neither need be the same size, however, for the FCN discussed in Section 4.2.1, the images should be of same size. When comparing performance metrics, the training images, settings of the network, and parameters of the training are kept constant.

## 4.3 Training

The data collection and datasets design details were discussed in Chapter 3, but in order to recall the details, Figure 4.4 shows the data collection and the dataset design method, which then leads to the network(s) training. The discussed networks were all trained

with a batch size of 10 images and for 500 epochs using SGD with 0.9 momentum and $1 \times 10^{-4}$ learning rate. Total eight networks were trained: Network-I, Network-II, AlexNet [54], GoogLeNet [172], VGG16, VGG19 [173], ResNet50 [174] and U-Net, on the SheepSet80. Network-I, Network-II and U-Net were trained from scratch and other networks were fine-tuned. The networks which were fine-tuned on our dataset, had pre-trained on ImageNet and were already capable for classifying objects of 1000 categories. After every epoch, each trained network was validated on the full validation set.

In contrast with the two-stage network, the one-stage network's training method and loss function differ. For the one-stage detector used in this research, the WHD loss function is used to compute the distance between the actual centroid values and the estimated points of the probability map. L1 loss calculates the difference between the number of objects estimated and the number actually observed. All other networks were trained using the R-CNN method. The first stage of R-CNN involves generating 1000 region proposals with respective positive and negative overlap values of $[0.5, 1]$ and $[0.1, 0.2]$.

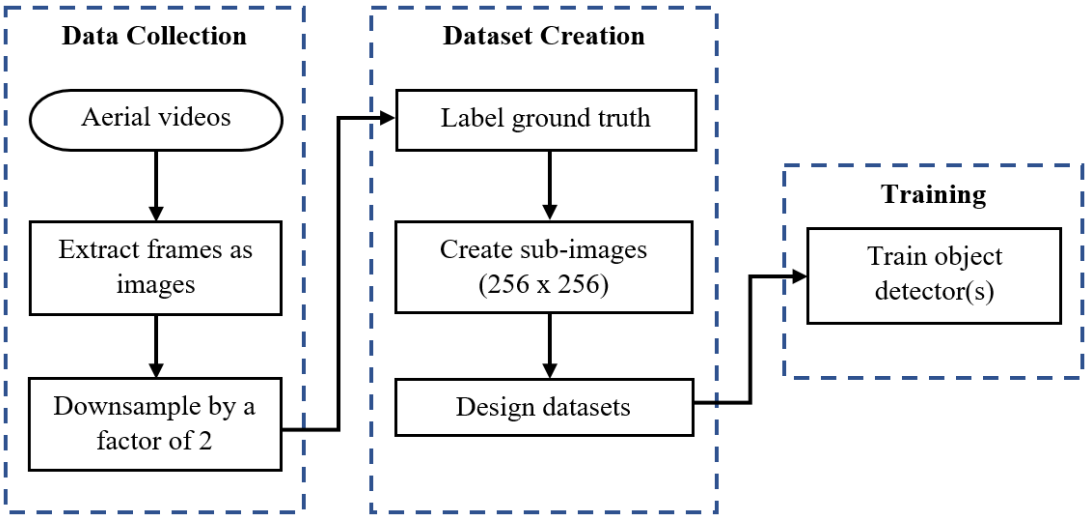



Figure 4.4: A system diagram for network(s) training.

ROIs are small in the images, and greater positive overlap values enhance object detection. The region proposal is then classified as either an object or background class by using the softmax and cross-entropy functions. The softmax gives the class probabilities, whereas the cross-entropy uses these probabilities for computing the loss to the actual class labels. The network that performed the best was then re-trained and tested using the SheepSet120 dataset.

Table 4.1 summarizes the networks used in this research and methods used to train them. This study makes use of an NVIDIA GeForce RTX 2080 with 3072 NVIDIA CUDA cores and 8 GB of GDDR6 memory for training, validation, and testing.

Table 4.1: List of object detectors and the respective training method.

| **Object Detector** | | **Training Method** | |
|---|---|---|---|
| Type | Name | Trained from Scratch | Fine-Tuned |
| Two-Stage | ResNet50 [174] | | ✓ |
| | GoogLeNet [172] | | ✓ |
| | AlexNet [54] | | ✓ |
| | VGG16 [173] | | ✓ |
| | VGG19 [173] | | ✓ |
| | Network-I | ✓ | |
| | Network-II | ✓ | |
| One-Stage | U-Net-GMM [62] | ✓ | |
| | U-Net-MS | ✓ | |

## 4.4 Testing

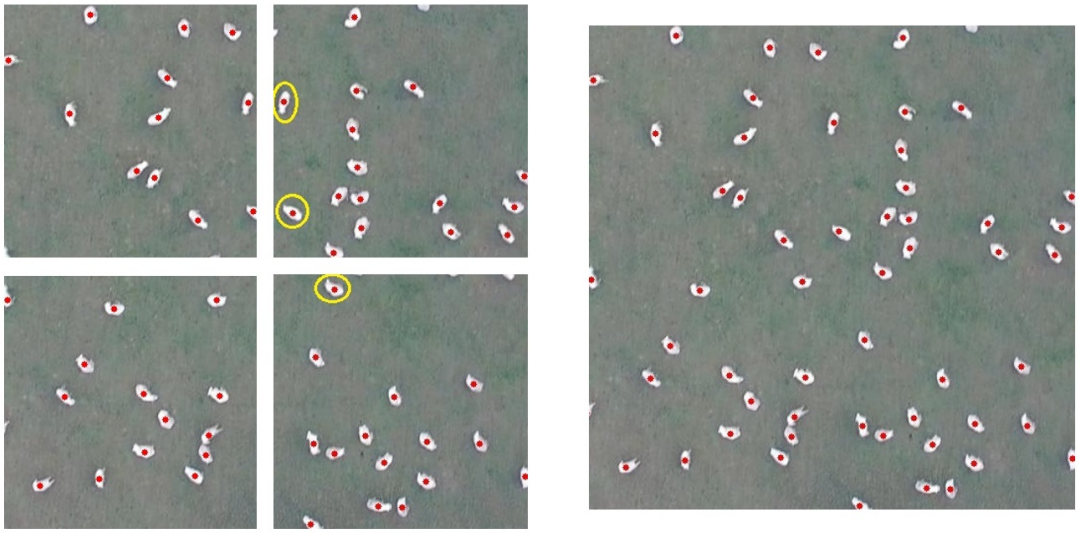
The test results of a few trained network on the test set of SheepSet80 are presented in Appendix B. This was only 10% of the entire set, and full frames need to be tested so that farmers can get a true count of their livestock more easily. For testing, 64 new frames were introduced from different videos recorded at 80 m and 120 m. Each test set has 64 images, testing steps are discussed briefly in the following subsections.

### 4.4.1 Image Stitching

The U-Net model used in this research has a limitation: the training and testing images should be of the same size. If the network is trained on 256×256×3 images then it can be tested only on the test images that have the same dimension. As we need to test the final model on each video frame, there were two options to cover this limitation. One option was to use full video frames for training too but this comes with a requirement of large memory to store all the parameters of the network. The GPU machine that we used for this research had limited memory, so this option was not possible. The other option was to train the network on smaller images and later during the testing phase, divide the test frame into multiple parts of the same dimension and stitch back the results. The second option was selected and RGB sub-images of 256×256 pixels were used to train the network, as mentioned in Chapter 3. For testing, a test frame was divided in a way that each sub-image has an overlapping edge of 20 pixels on each side with the neighboring sub-images. It was observed that the value of 20 pixels is the average size of sheep in each frame. This selection of overlap areas ensured that successive sub-images will have either partial or full sheep in them. After testing, those sheep centroid values were reduced to one value, which were repeating in the overlapping areas. There are four such sub-images in Figure 4.5 where some sheep appear twice. Counting these sheep only once is achieved by stitching these four images together.

### 4.4.2 Bounding Box Estimation

The inference process of the two-stage networks is very similar to the training. It involves the generation of many region proposals which are then used by the CNN for feature extraction. Probabilities and class labels are computed, respectively, by Softmax and cross-entropy. The regions with confidence value less than 0.5 were eliminated using non-max suppression, and the strongest bounding boxes were taken into account.



(a) Four sub-images with 20 overlapping pixels at boundaries (b) Stitched image with sheep counted once at boundary

Figure 4.5: Sub-images of test dataset with centroids shown as red circles.

### 4.4.3 Centroid Estimation

As shown in the rightmost image of Figure 4.1, a cluster will be displayed in the probability map where the U-Net model detects an object. Pixels of object tend to have the highest probability values and centroid values of such clusters can be computed using different clustering algorithms. In this study, we explored two clustering algorithms: Gaussian mixture models (GMM) and mean shift (MS). According to the data and the probability map, either of these methods can be used. One disadvantage of the GMM is that it calculates the centroids only when the number of groups is known in advance. With the Softplus function, $\hat{C}$ (4.6), the estimated number of objects can be obtained and become an input to the GMM. A difference between the true object count and the estimated one can lead to either false positive or false negative result. The MS algorithm does not require prior knowledge of the number of the estimates. It is important to specify the cluster radius and we used 6 pixels for 80 m and 4 pixels for 120 m video frames.

Riberia *et al.* [175] also tried both GMM and MS, and reported that detection results

were improved with the use of GMM algorithm. We found the opposite result, but note that using MS. Because our focus is on achieving a better recall measure, the inference time of our system was compromised for higher sensitivity and precision and examples are shown in Figure 4.6. When compared to the values in the first row, which are calculated by GMM, the centroid values calculated by MS in Figure 4.6 (b) are more accurate.

Figure 4.7 illustrates the steps for testing the video frames with the one-stage and two-stage object detectors. Since there is a lesser number of steps for two-stage CNN, these networks have a shorter inference time. The one-stage detector takes more time to produce the results mainly due to using a clustering algorithm and image stitching.

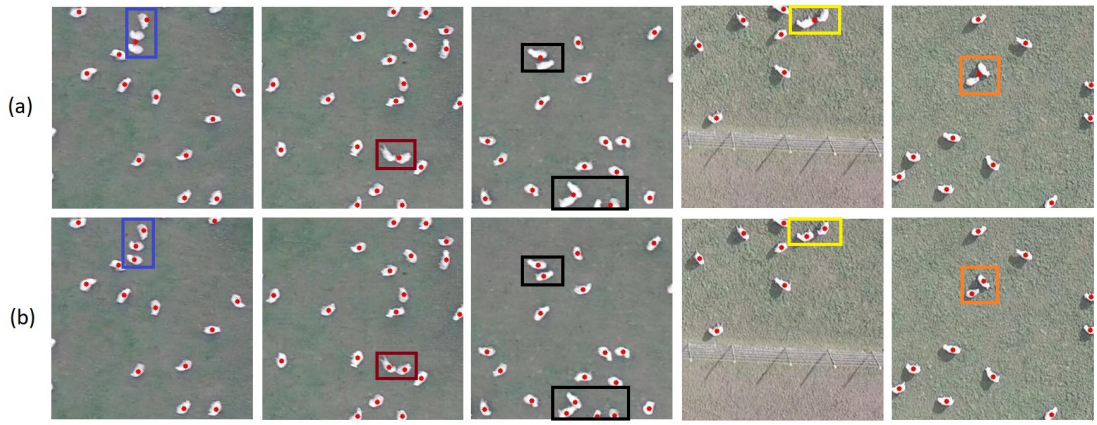



Figure 4.6: Centroids are plotted as red circles over respective sheep using (a) GMM and (b) MS.

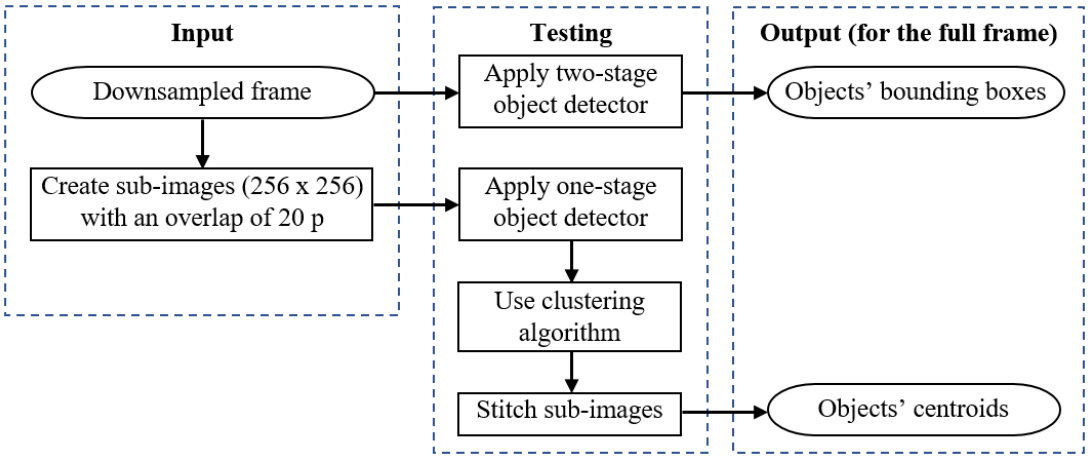



Figure 4.7: Main steps for the livestock detection using different object detector(s).

### 4.4.4 Metrics

The performance metrics we used are: precision, recall, $F_1$-score, and mean absolute percentage error (MAPE), and are defined as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{4.10}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{4.11}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \tag{4.12}$$

$$\text{MAPE} = 100 \cdot \frac{1}{N} \sum_{n=1}^{N} \left| \frac{C_n - \hat{C}_n}{C_n} \right|, \tag{4.13}$$

where, the numbers of true positive, false positive, and false negative detection are represented by TP, FP, and FN, respectively. The actual and estimated numbers of items in the $n$-th image are represented by $C_n$ and $\hat{C}_n$, respectively. Correct detection of sheep in any image is represented as a TP detection. If the estimated object is not matched with a ground truth value, it is reported as an FP. The FN value is produced when no actual object is detected. Precision is compromised by a higher FP value, while recall is affected by a higher FN value.

As the recall shows the sensitivity of the trained network, it is a crucial factor in livestock detection. The precision identifies the accuracy in the detection of objects, and the $F_1$-score, the first harmonic mean of both precision and recall, is the final measure of how well the trained network performed overall. Accordingly, MAPE is a statistical measure of network accuracy and is used to compare the actual and estimated object counts per image. We computed these metrics based on a parameter, intersection over union (IOU), which compared the boundaries of detected and actual objects. IOU computes the ratio of the total overlapping area of the estimated and actual bounding boxes to the combined area of both.

## 4.5 Results

### 4.5.1 Sheep Detection at 80 m

The performance metrics for all CNN networks are shown in Table 4.2. These networks were trained on SheepSet80 but were tested on different video frames. We present experimental results based on our U-Net model when the computation of centroids were carried out using GMM and MS, which we refer to as U-Net-GMM and U-Net-MS, respectively. Regardless of which clustering algorithm was used, U-Net consistently outperformed the rest. By using MS, the value of FN and FP has decreased because centroid values are given for all sheep detected by the network. In terms of performance, Network-II performed better than all other networks except for the U-Net. Considering that sheep in these images are very small, deep networks were expected to lose some necessary information about objects at the deeper layers. Experiments demonstrate that this is indeed the case.

In order to improve the accuracy, we combined U-Net detections with Network-II detections, which are shown in the bottom two rows of Table 4.2. But since this

Table 4.2: Performance metrics of all networks.

| Network | Mean Precision | Mean Recall | Mean $F_1$-score | MAPE |
|---|---|---|---|---|
| ResNet50 [174] | 87.46% | 81.03% | 84.12% | 8.16% |
| GoogLeNet [172] | 88.62% | 81.01% | 84.64% | 9.08% |
| AlexNet [54] | 93.12% | 80.83% | 86.54% | 13.09% |
| VGG16 [173] | 93.37% | 80.75% | 86.60% | 13.50% |
| VGG19 [173] | 95.72% | 80.70% | 87.57% | 15.68% |
| Network-I | 97.70% | 80.21% | 88.09% | 17.90% |
| Network-II | 98.14% | 81.08% | 88.88% | 17.38% |
| U-Net-GMM [62] | 98.31% | 96.60% | 97.44% | 2.58% |
| U-Net-MS | **98.58%** | **98.02%** | **98.30%** | **1.89%** |
| U-Net-GMM+Network-II | 98.35% | 97.62% | 97.98% | 1.67% |
| U-Net-MS+Network-II | **99.02%** | **98.25%** | **98.64%** | **1.63%** |

significantly increases computational complexity, it may not be justified in a real system for the modest benefit it offers. Even so, the results of 99.02% precision and 98.25% recall are very good and, at least to our knowledge, surpass all previous results published for livestock detection in UAV captured images.

Figure 4.8 shows the sheep detection results for one frame by (a) ResNet50, (b) GoogLeNet, (c) AlexNet, (d) VGG16, (e) VGG19, (f) Network-I, (g) Network-II, (h) U-Net-GMM, (i) U-Net-MS and (j) U-Net-MS + Network-II. As the main purpose was to show the sheep detection results in these images, the extra space from the left and right sides was removed for clarity. However, full frames were used in the testing phase. Those sheep that were not detected by the U-Net and were successfully detected by Network-II are shown in the yellow bounding boxes. In Figure 4.8 (g), (h) and (i), to facilitate visual comparison, the bounding boxes of 25×25 pixels are designed using the centroid values generated from the U-Net model.

According to Figures 4.9 and 4.10, the recall and precision for respective networks have decreased as the IOU ratio between estimated and actual bounding boxes has increased. There will be fewer detections classified into the positive class as the IOU

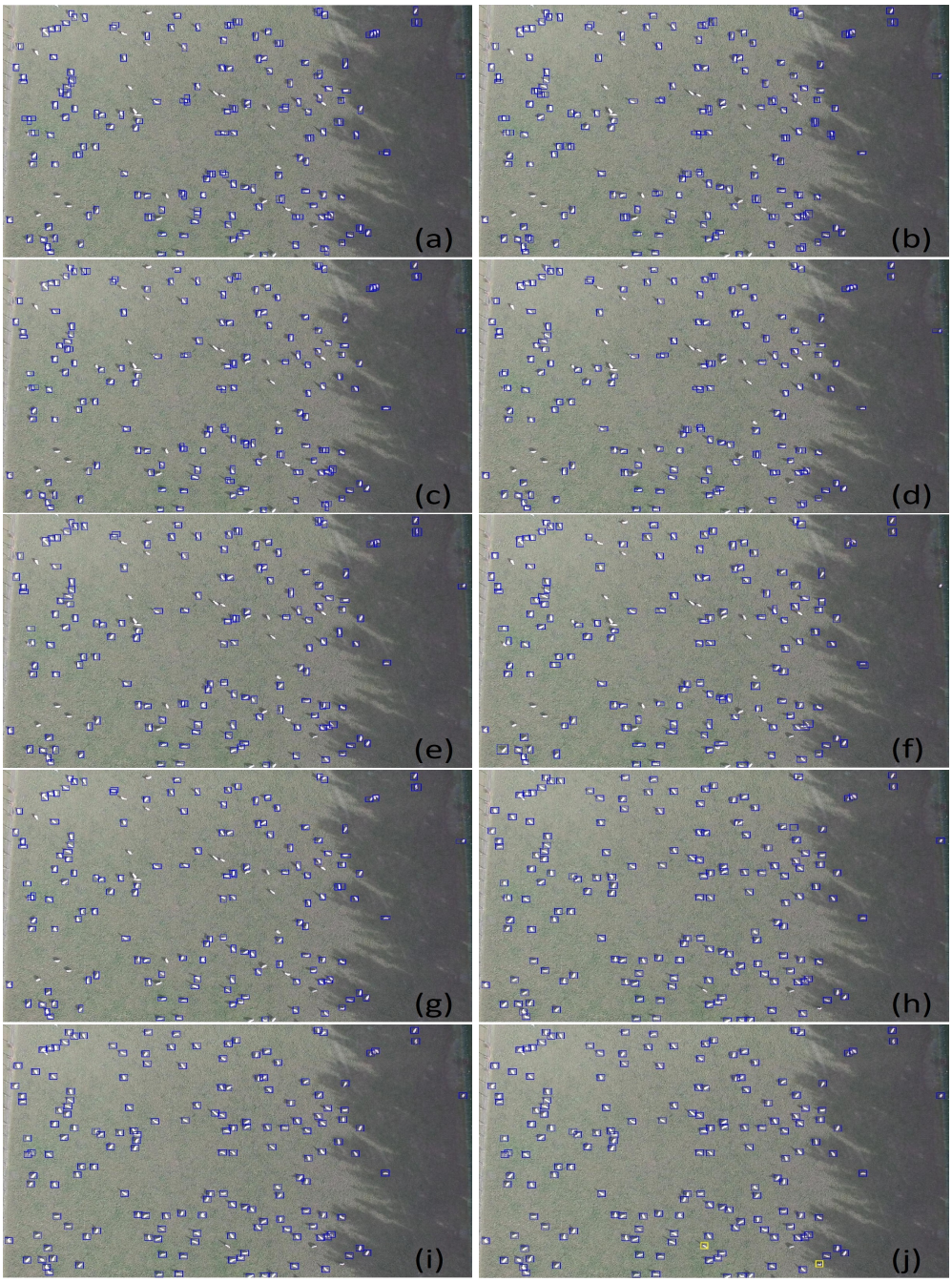



Figure 4.8: An example frame taken from the video recorded at 80 m height tested by (a) ResNet50, (b) GoogLeNet, (c) AlexNet, (d) VGG16, (e) VGG19, (f) Network-I, (g) Network-II, (h) U-Net-GMM, (i) U-Net-MS and (j) U-Net-MS+Network-II.

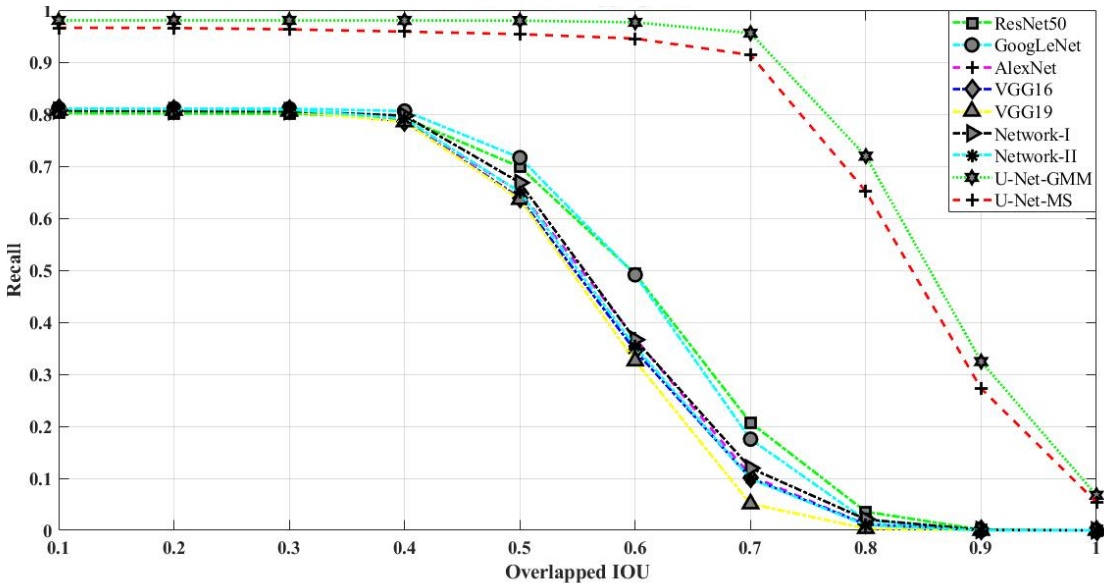



Figure 4.9: Recall vs IOU.

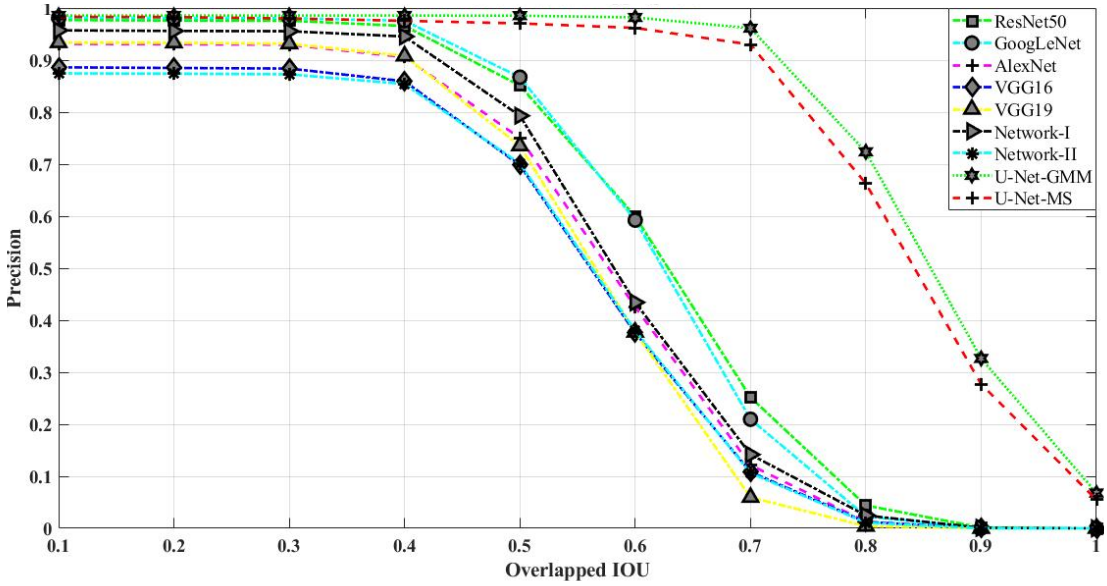



Figure 4.10: Precision vs IOU.

increases. Precision and recall were computed over the IOU range $[0.1, 0.2, 0.3, \ldots, 1]$ and it was found that they remained unchanged until the value of 0.3 but decreased afterward. The U-Net results appear to be much more robust to changes in the IOU. We used an IOU threshold of 0.1 in Table 4.2 to compute results.

### 4.5.2 Sheep Detection at 120 m

Despite getting encouraging sheep detection results in the videos recorded at 80 m, flying the UAV at 120 m is more practical as larger paddocks can be covered easily. The left and right fences can still be kept in each frame for wider paddocks. It is also the highest point of the legal limit in many countries. But this altitude selection will come with a compromise on the sheep size. The sheep appear even smaller from 120 m and it will be challenging to detect them with a high accuracy.

The sheep dataset at 120 m, SheepSet120, was designed in the same way as SheepSet80 was created. Initially, it was tested with the U-Net-MS network that was trained on SheepSet80. We wanted to check if the trained network was invariant to any scale variation. A drop of 36% in the mean $F_1$-score was observed, as shown in the second row of Table 4.3. The results were improved to some extent by training the U-Net model on a dataset consisting of both 80 m and 120 m training images, but not quite to the level of when the training and testing datasets are both from the same altitude. As long as the images are tagged with the altitude at which they are taken, and if they are then analyzed using a trained network matching their altitude, this restriction is not significant. Creating SheepSet120 was a bit more challenging since it contained many other objects such as wooden logs, tree branches, cattle, farm gates, etc., that could

Table 4.3: Mis-Matches effect on the performance of the U-Net-MS, with different training and testing datasets.

| Training Dataset | Testing Dataset | Mean Precision | Mean Recall | Mean $F_1$-score | MAPE |
|---|---|---|---|---|---|
| 80 m | 80 m | 98.58% | 98.02% | 98.30% | 1.89% |
| | 120 m | 49.45% | 84.13% | 62.28% | 103.87% |
| 80 m & 120 m | 80 m | 95.67% | 99.32% | 97.46% | 10.93% |
| | 120 m | 91.87% | 97.88% | 94.78% | 19.01% |
| 120 m | 80 m | 43.53% | 72.15% | 54.30% | 99.05% |
| | 120 m | 96.53% | 98.20% | 97.36% | 5.77% |

have been misinterpreted as sheep. Since the multiple objects in these images were harder to distinguish, it was much more challenging to create the ground truth data at this altitude. The network effectively handled this issue and performed unexpectedly well. Figure 4.11 shows an example of sheep detection result at the mentioned height and it can be seen that the sheep contain little information about their contours. The image can be seen to have a few cattle, which have not been detected. The network was trained specifically to detect sheep due to the limited data available. The results are very encouraging, considering that 120 m is a more practical height for UAVs, and the performance is only slightly affected compared to that of 80 m.

Moreover, the performance of the U-Net-MS was better on the images taken under cloudy conditions at both altitudes. Bright sunny weather causes the sheep boundary to blend with the background as the grass reflects sunlight and makes the sheep detection a tricky task. It is not the case in cloudy weather, where the boundary of the sheep is very clear, appearing as a whitish blob in front of the grassy background and can be detected more confidently by the network. There are test images from both sunny and cloudy

Figure 4.11: Sheep detection result at 120 m altitude.

weather in Tables 4.2, 4.3 and 4.4. These results show that the overall performance of the networks is acceptable. Based on the successful performance of the U-Net-MS network, it may be used to develop a UAV-based application for helping farmers detect and count livestock.

Table 4.4: Performance metrics of sheep detection using U-Net-MS.

| Metric | Datasets | |
|---|---|---|
| | Sheep at 80 m | Sheep at 120 m |
| Precision | 98.58% | 96.53% |
| Recall | 98.02% | 98.20% |
| $F_1$-score | 98.30% | 97.36% |
| MAPE | 1.89% | 5.77% |

## 4.6 Concluding Remarks

In this chapter, we discussed the architecture and performance of different networks for livestock detection in UAV video frames. We proposed two networks for livestock detection, and compared the results with U-Net, AlexNet, GoogLeNet, VGG16, VGG19 and ResNet50. The last five networks were fine-tuned on our dataset, SheepSet80. Combining finalized results of one of our networks, Network-II, and U-Net-MS led to the highest recall, but at a higher computational cost. The one-stage detector—U-Net-MS—outperformed the existing state-of-the-art detectors. As far as the author is aware, performances at this level have never been reported before in this domain. By using this trained network, a farmer or a farm manager can count livestock directly from UAV images. This is a very important step towards designing an automated counting system for livestock.

# Chapter 5

# Livestock Tracking and Counting

In a video recorded by a fixed position or moving camera, the multiple object tracking (MOT) algorithms follow trajectories of multiple objects. Depending on the recording mode, the video can have various foreground objects and moving or fixed backgrounds. As deep learning for the task of MOT in video recordings by UAVs has grown, its challenges have also risen. The three main components, the object detector, the object tracker, and the data associator, play equally vitals role in the tracking-by-detection method. The individual components need to operate at maximum efficiency in order to achieve the best performance as a whole. This research also focused on selecting the optimal parameters of the Kalman filter and the Hungarian algorithm for tracking multiple sheep in different paddock videos. Using an experimental evaluation, it is shown that a slight variation in the system components can change the capability of tracking results if the object detector is already working up to the required level [169]. The complete system and the results that we are going to discuss in this chapter were presented in [178], and were published as:

**C4:** F. Sarwar, A. Griffin and T. Pasang, "Tracking livestock using a fully connected network and Kalman filter," *Geometry and Vision. ISGV 2021. Communications in Computer and Information Science,* vol 1386, pp. 247–261, 2021.

## 5.1 Multiple Livestock Tracking

Figure 5.1 shows a system diagram of our proposed method. Although all the recorded videos were only about 10 seconds long, some preprocessing was done to reduce the effect of inference time taken by the U-Net-MS model [176]. Each video was downsampled in the temporal and spatial domains by a factor of 2, and then the downsampled frame was processed by the object detector as well as the Kalman filter [179]. This helped in reducing the video time by half and inference time by a factor of 8. After downsampling, the respective original RGB frame of 4096×2160 pixels was reduced to 2048×1080 pixels. The U-Net-MS was used to detect sheep on the downsampled frames and as some paddocks share fences, we ignored the the sheep that were outside the fences. Then the Kalman filter predicts the trajectories, and the Hungarian algorithm links these trajectories to objects based on the predicted state of the tracks and estimations provided by the object detector. In the last step, tracks were corrected using the estimations provided by the object detector and these corrected tracks were used as the basis of next cycle.

### 5.1.1 Object Detection

The object detector has an important part in keeping up the overall system's performance of the tracking-by-detection method. By enhancing the detection accuracy, the task of motion prediction and track-to-detection association becomes significantly easier. In the proposed method, the U-Net-MS model detects sheep in the UAV's recorded videos and performs this task with a high recall [169]. The network architecture of the U-Net-MS model and its full working was discussed in Chapter 4.

When each frame is fed to the U-Net-MS model trained for sheep detection it is initially divided into sub-frames of 256×256 dimension with an overlap of 20 pixels on each side. Due to the limited memory of the system used, inference on the full frame

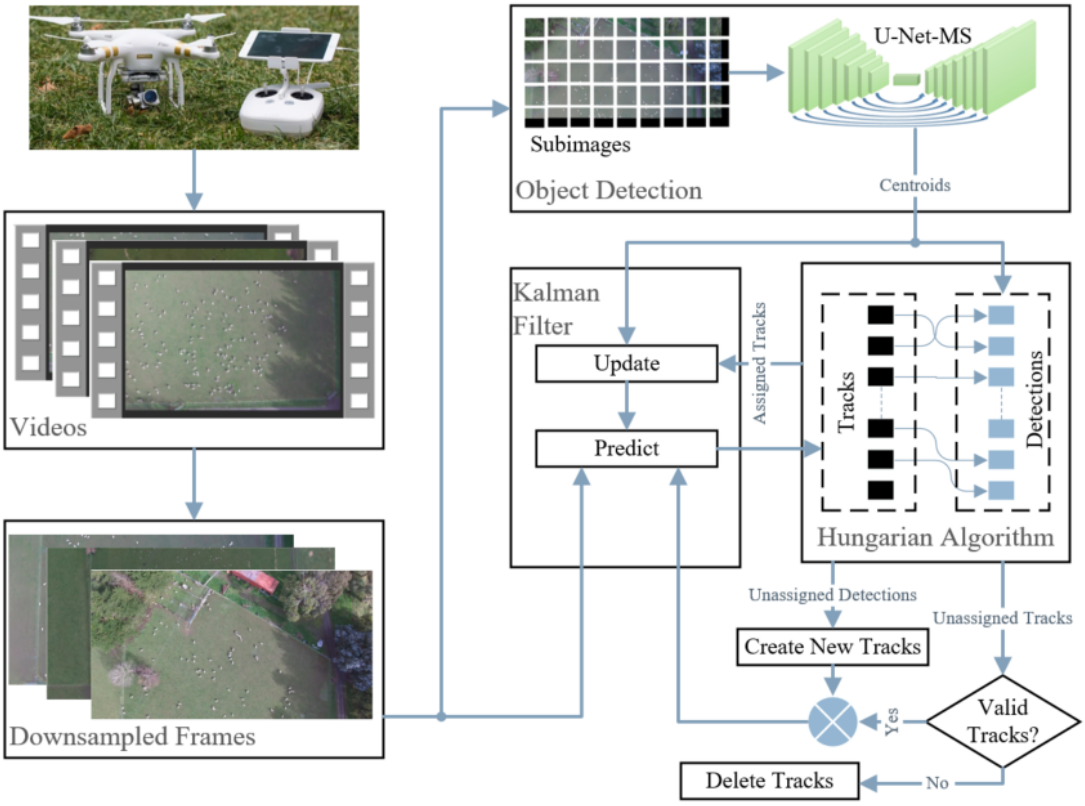



Figure 5.1: A system diagram of sheep detection and tracking.

was not possible and the centroids computed for each sub-image were concatenated in a way that there is no repetition in the overlapped area [170].

### 5.1.2 Object Tracker

On average a sheep moves the same distance in pixels between frames as all the other objects and background. The average movement of all the objects was between 10 to 18 pixels. This range was computed by comparing the pixels coordinates for the respective objects in two consecutive frames.

As the initial value of each object's state, the Kalman filter is provided with the centroid of each sheep in the full-frame, along with its velocity in the $x$- and $y$- direction. The sheep's coordinates are directly provided by the U-Net-MS model and the velocity

can be either set to zero or equal to the average velocity of a sheep. Initially, the velocity was set to zero to tune the tracker and data linker in a noisy environment. So, in the $i$-th frame, the state vector for the $m$-th object is provided as:

$$\mathbf{s}_i^m = \left[c_{\mathrm{x},i}^m,\, v_{\mathrm{x},i}^m,\, c_{\mathrm{y},i}^m,\, v_{\mathrm{y},i}^m\right]^T, \tag{5.1}$$

where $c_{\mathrm{x},i}^m$ and $c_{\mathrm{y},i}^m$ are the $x$- and $y$-coordinates of the true centroid of the $m$-th object, and $v_{\mathrm{x},i}^m$, $v_{\mathrm{y},i}^m$ are its velocities in the respective directions. With the following equation, the states of each object in the $i$-th frame were estimated from the state values in the $(i-1)$-th frame:

$$\mathbf{s}_i = \mathbf{A}\mathbf{s}_{i-1} + \mathbf{w}_i, \tag{5.2}$$

where $\mathbf{A}$ is the state transition matrix and $\mathbf{w}_i$ is the process noise. The state transition matrix was designed as per the linear motion of the tracked objects. The process noise has a normal distribution with zero mean and covariance $\mathbf{Q}$, representing the unexpected noise in the system. This covariance value can be determined based on the quality of the video and the unexpected variation in the linear motion of the objects. The importance of this factor can also be explained with respect to aerial videos recorded by the UAV. In windy conditions, the UAV flight will not be smooth and may need frequent adjustments to stay on course. For this particular case, the covariance values help the Kalman filter to predict the next position close to the expected position.

As the Kalman filter is a recursive algorithm, it can predict optimal states before the observations are read and update them based on the values estimated by the object detector for the respective current frame. For any frame, each state measurement is modelled as follows:

$$\mathbf{z}_i = \mathbf{H}\mathbf{s}_i + \mathbf{v}_i, \tag{5.3}$$

where $\mathbf{H}$ is the measurement model matrix to link the object detection values and states

of the Kalman filter. If there is a difference of unit between both then $\mathbf{H}$ converts one value to the other without affecting the system. In our case, as $\mathbf{z}_i$ and $\mathbf{s}_i$ are using the same unit of pixels, $\mathbf{H}$ is just an identity matrix. Here $\mathbf{v}_i$ is the observation noise and has a normal distribution with zero mean and covariance $\mathbf{R}$. The value of this covariance is adjusted as per the certainty of the detector's output.

In order to estimate the average velocity of the sheep for each track, we took the two-dimensional fast Fourier transform of $(i-1)$-th frame, $\mathbf{F}_{i-1}$, and $i$-th frame, $\mathbf{F}_i$, so that

$$\mathbf{G}_{i-1} = \mathcal{F}\left\{\mathbf{F}_{i-1}\right\}, \tag{5.4}$$

$$\mathbf{G}_i = \mathcal{F}\left\{\mathbf{F}_i\right\}. \tag{5.5}$$

We then form the cross-power spectrum as

$$\mathbf{R}_i = \frac{\mathbf{G}_{i-1} \odot \mathbf{G}_i^*}{|\mathbf{G}_{i-1} \odot \mathbf{G}_i^*|}, \tag{5.6}$$

where $\odot$ denotes the Hadamard product (the element-wise product), and the absolute value in $\mathbf{R}_i$ is also element-wise.

The estimated $x$ and $y$ motion between $\mathbf{F}_{i-1}$ and $\mathbf{F}_i$ can then be computed as

$$\hat{\mathbf{m}}_i = \arg\min_{x,y}\left[\mathcal{F}^{-1}\left\{\mathbf{R}_i\right\}\right]. \tag{5.7}$$

The value of $\hat{\mathbf{m}}_i$ was used to set the velocity of the new tracks designed for this frame.

### 5.1.3 Track-to-Detection Association

The Hungarian algorithm [180] was used to solve the assignment problem between existing tracks and all objects detected by the U-Net-MS model in each frame. A

cost matrix was calculated between predicted states of all tracks and the estimated centroids of all objects. Each column of this cost matrix represents the Euclidean distance between the estimated centroid and predicted positions of all tracks, and similarly, each row of the matrix is Euclidean distance between the predicted position of a track and all estimated centroids. This cost matrix is then used by the Hungarian algorithm to complete the task of track-to-detection assignment and a factor, cost of non-association (CNA), is used to decide the leniency of this algorithm. If CNA is a small number then it may not link estimations and predicted states efficiently, and categorise most of them as unassigned values. This can result in a large number of FPs and FNs in the tracking system, however, if the value of CNA is set to a higher value then this algorithm will make more pairs. The decision of the CNA value depends on the application and average distance between objects, and is adjusted experimentally in our research.

Often not all tracks and detections can be linked and some unassigned tracks, as well as detections, are also identified. The tracks that were successfully linked are considered reliable tracks and were sent to the Kalman Filter for correcting and updating the respective states as per the centroid values provided by the U-Net-MS. Tracks categorized as unassigned tracks were then checked with different conditions to conclude whether keeping them in the system will be beneficial or not. The Kalman filter creates new tracks for the unassigned detections, as these are usually due to those objects which have just entered the frame.

### 5.1.4 Unassigned Tracks

As shown in Figure 5.2, the paddock videos were recorded in a way such that sheep always leave the frame from the bottom and only enter from the top. Whenever a sheep enters the frame, it is categorised as an unassigned detection and a new track with a

unique ID is created for it by the Kalman filter. However, a few tracks need to be deleted as per the following three main conditions:

1. The track visibility was less than 60% and it was visible for less than two frames,
2. The track's state was predicted beyond the lower boundary of the frame, and
3. The object detector did not detect the respective object in consecutive $F_{\text{lost}}$ frames.

The track visibility factor was calculated by dividing the number of frames for which the object was detected and number of frames the Kalman filter was predicting the position for the same object. The first condition helps to reduce the FP detections as they do not exist for long in any video, while the second condition is for those sheep that left the frame from the bottom. Deleting these tracks avoid the unbound growth of the tracks and reduces the use of computational resources. The third condition is for those crucial cases where the object detector fails momentarily and does not detect the exact number of sheep properly in a group. Although this does not occur frequently, the respective tracks were then kept in the loop for at least $F_{\text{lost}}$ frames and were deleted afterwards if they were FP detections.

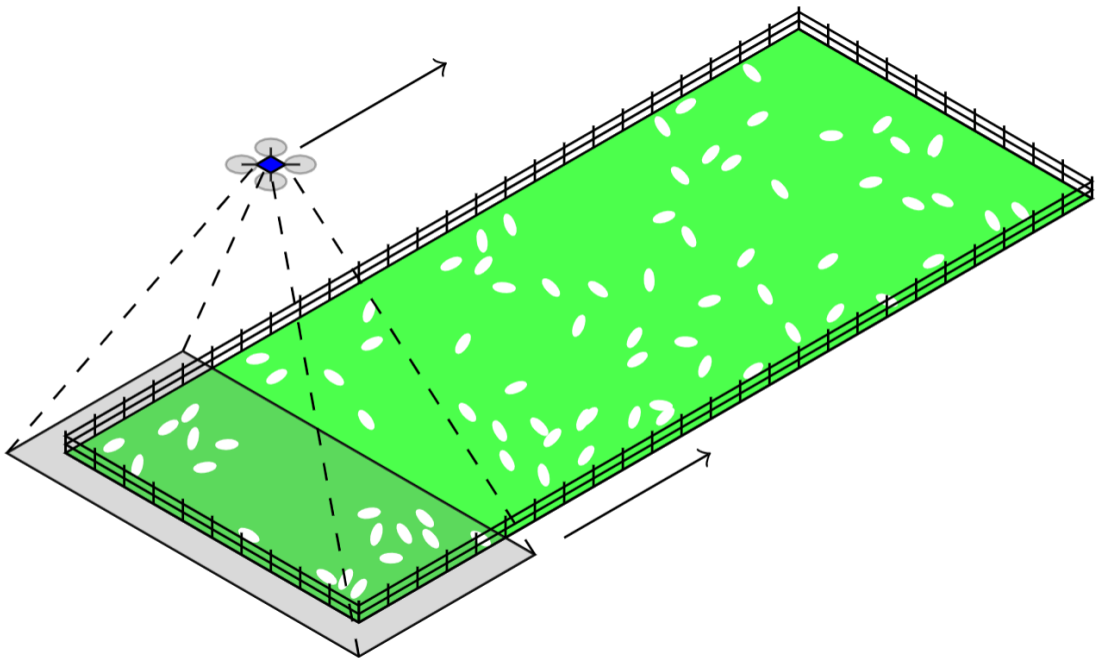


Figure 5.2: An illustration of how a UAV is used to record a video of a paddock, where sheep are shown as white ellipses.

## 5.2 Experimental Evaluation

### 5.2.1 Parameters

In the recorded videos most of the sheep move with a constant velocity and exhibit a linear motion, as discussed earlier. However, there are a few sheep that showed some random movements and do not move towards the expected position. The parameters of the Kalman filter and the Hungarian algorithm were adjusted to cover such issues. The values of the main variables used in the prediction step are as follows:

$$\mathbf{A} = \begin{bmatrix} 1 & T_x & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_y \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Q} = q\, I_{4\times4}, \quad \mathbf{R} = r\, I_{4\times4}, \tag{5.8}$$

where $I_{4\times4}$ is the 4×4 identity matrix, and $T_*$ is the sampling interval which was set to 1. The process and noise covariance values—$q$ and $r$—which represent the unexpected noise in the overall system and the object detector's response, were varied between $[1, 2, 3, 4, 5, 10]$. The large covariance ensures that a track will be linked with a nearby detected object but it can worsen the condition where there are many groups of closely standing objects, and we selected the final values experimentally.

A variation in the tracking performance was also observed with variation in the values of the CNA. If the objects detected in each frame are not linked properly with the existing tracks, this may lead to the creation of new tracks, hence increasing the numbers of unassigned tracks. Another factor that impacts the system's performance is $F_{\text{lost}}$, the number of frames for which the unassigned track should stay in the system. If any unassigned track is not predicted close to the respective undetected sheep then it will be an FP case, and a FN otherwise. One of the targets was to have accurate sheep

count for respective paddock, so decision making had to be limited to whether to create a new track for an object missed earlier or to simply keep the previous track until the object appeared.

These parameters are summarized as follows:

- process and observation noise covaraince, $q$ and $r$
- cost of non-association, CNA,
- threshold frame value, $F_{\text{lost}}$.

The effect of changing the values of these parameters were experimentally investigated in the recorded videos.

### 5.2.2 Performance Metrics

As the tracking-by-detection method was used for livestock tracking and counting in a paddock, the performance metrics for both the detector and the full system are reported here. The main metrics used to observe the performance of the U-Net-MS model throughout the video are precision per frame (PPF), recall per frame (RPF) and false alarm per frame (FAF). All these values were then averaged over the respective full video sequence. The PPF shows the ability of the U-Net-MS model to identify the true cases, RPF shows the ability of the detector to find all the relevant cases, and FAF is the average number of FPs in each frame. These values were calculated as follows:

$$\text{PPF} = \frac{1}{M}\sum_{i=1}^{M}\frac{\text{True Detections}}{\text{All Detections}} \times 100, \tag{5.9}$$

$$\text{RPF} = \frac{1}{M}\sum_{i=1}^{M}\frac{\text{True Detections}}{\text{All Ground Truth}} \times 100, \tag{5.10}$$

$$\text{FAF} = \frac{1}{M}\sum_{i=1}^{M}\frac{\text{Incorrect Detections}}{\text{All Ground Truth}} \times 100. \tag{5.11}$$

where $M$ is the total number of frames.

Different metrics can be used to evaluate a multiple object tracking system and it usually depends on the requirement of the respective problem. Obtaining an accurate sheep count in the respective paddock is the primary task for this system, so achieving lower values of FP and FN is essential. A lower value of FP indicates that there were fewer cases where the tracker was predicting an object's position at the wrong place. While a lower FN supports the fact that there were fewer cases where a track was not assigned to the respective sheep. The cases of FN usually appeared if either the tracker was unable to follow the unexpected movement of the sheep or the detector did not detect the sheep in the last few frames. The tracker was then corrected accordingly and this usually happens in a group of sheep.

Among other standard and commonly used MOT metrics are MOT accuracy (MOTA), MOT precision error (MOTPE), and false positive rate (FPR) [181, 182, 183]. However, we were unable to compare our results with any other livestock tracking work due to the unavailability of any relevant publicly available dataset. The MOTA encompasses the FPs and FNs during the whole tracking process and is computed as:

$$\text{MOTA} = \left(1 - \frac{\sum_i \left[\text{FP}(i) + \text{FN}(i)\right]}{\sum_i \text{GTC}(i)}\right) \times 100, \tag{5.12}$$

where $\text{FP}(i)$, $\text{FN}(i)$ and $\text{GTC}(i)$ represent the number of false positives, false negatives, and ground truth counts in the $i$-th frame, respectively.

The MOTPE parameter measures the ability of the tracker to estimate the precise positions of the objects, irrespective of the other errors. It is the ratio of the total position error between the matched objects to the total number of matches made. The value of MOTPE (in pixels) is computed as follows:

$$\text{MOTPE} = \frac{\sum_i d(\hat{\boldsymbol{c}}_i, \boldsymbol{c}_i)}{\sum_i \text{MC}(i)}, \tag{5.13}$$

where $\hat{\boldsymbol{c}}_i$ and $\boldsymbol{c}_i$ are 2×1 vectors containing the x- and y-coordinates of the predicted and ground truth centroids of the objects, respectively. $d(\boldsymbol{a}, \boldsymbol{b})$ is the Euclidean distance between $\boldsymbol{a}$ and $\boldsymbol{b}$, and $\text{MC}(i)$ is the number of the matched centroids between $\hat{\boldsymbol{c}}_i$ and $\boldsymbol{c}_i$. A lower Euclidean distance between the sheep's predicted and true centroids results in a lower MOTPE because their predicted and true centroids are closer together.

The FPR is the ratio of negative events wrongly categorized as positive to the actual number of ground truth counts in all frames. In other words, it is a measure of the number of FPs as a fraction of the total objects in all frames and is defined as

$$\text{FPR} = \Big( \frac{\sum_i \text{FP}(i)}{\sum_i \text{GTC}(i)} \Big) \times 100. \tag{5.14}$$

As the UAV moves much faster than the sheep, and fences on the left and right sides were within each frame, a sheep only enters any frame from the top and exits via the bottom. A counter, $K_i$, was used for this task and was incremented with the number of those tracks that were deleted under the second condition of valid tracks.

Thus, assuming that the video stops when the fence at the top is visible, there are $M$ frames in the video and the last frame has $N_M$ active tracks, the total sheep count was computed as

$$\text{SC} = N_M + \sum_{i=2}^{M} K_i. \tag{5.15}$$

Table 5.1 summarises all these detection and tracking metrics. The direction of arrows following each metric shows whether the corresponding value should be higher or lower to indicate better performance.

Table 5.1: Performance metrics used for evaluating livestock detection, tracking and counting.

| Category | Metrics | Description |
|---|---|---|
| Object Detector | PPF ↑ | Precision per frame |
| | RPF ↑ | Recall per frame |
| | FAF ↓ | False alarm per frame |
| MOT | MC | Matched centroids |
| | FPR ↓ | False positive rate |
| | MOTPE ↓ | Multiple object tracking precision error |
| | MOTA ↑ | Multiple object tracking accuracy |
| Counting | SC | Sheep count |

## 5.3 Results

The results of sheep detection and tracking are presented from six different videos. Two of these videos were recorded at an altitude of 80 m in the same paddock in cloudy and sunny weather and were labelled as 80a and 80b respectively. The remaining four videos—120a to 120d—were from an altitude of 120 m in cloudy and sunny weather and were all from different paddocks. In the first stage, sheep were detected in all frames of each video separately and Table 5.2 shows these results. The high recall at both altitudes and corresponding low false alarm rate shows that the detection rate of the U-Net-MS model is good enough to be used as a reliable detector.

At both altitudes, the best results were observed in the overcast videos. In the two videos recorded in sunny weather—80b and 120d—a degradation in the detector's performance is clear from Table 5.2. In 120d, the sheep shadows were observed very closely in order to verify if the ground truth labels were correct. A few examples are shown in Figure 5.3 to illustrate the challenge for this particular video. It is clear that it is difficult to identify the exact number accurately. In Figure 5.3(c) three sheep are standing near a tree stump, and one of them is under the shadow of this tree stump. Then in Figure 5.3(e) the sheep is in top center but is hardly visible.

Table 5.2: Sheep detection results on all recorded videos.

| Video Data | Metrics | | |
|---|---|---|---|
| | PPF ↑ | RPF ↑ | FAF ↓ |
| 80a | 99.97% | 99.51% | 0.03% |
| 80b | 99.84% | 96.44% | 0.15% |
| 120a | 99.88% | 96.80% | 0.12% |
| 120b | 99.80% | 98.92% | 0.20% |
| 120c | 99.41% | 98.40% | 0.59% |
| 120d | 94.80% | 87.48% | 4.85% |

### 5.3.1 Tracking Livestock at 80 m

The paddock used for data collection at 80 m had a total of 352 sheep and each frame of these videos contained between 100 to 250 sheep. The paddock recordings using the UAV was done on different days, the livestock spread was different which caused a variation in the total ground truth count. Thus the cumulative ground truth counts in the 80a and 80b videos were 15,045 and 19,798, respectively.

As both $q$ and $r$ are varied simultaneously, these values will be referred to as covariance. Initially, values of $F_{\text{lost}}$ and CNA were not changed and kept constant

(a) Four sheep (b) Three sheep (c) Five sheep

(d) Three sheep (e) One sheep (f) Eight sheep

Figure 5.3: Sub-images from different frames of the 120d video, where the actual sheep number is stated below.

at 10 and 20, respectively, where the values of the covariance was varied through $[1, 2, 3, 4, 5, 10]$. As well as adjusting the track-to-detection linking process, the system was modified to delete invalid tracks less strictly and results are shown in Table 5.3.

For the 80a video, the UAV velocity fluctuated in the second half of the video which disturbed the complete tracking system. The lower value of covariance and CNA resulted in the creation of multiple tracks for the same sheep in each successive frame, causing very high number of FPs and FNs. The negative values of MOTA clearly show the poor performance of the system. But this was not the case for the other video, 80b, where the velocity was almost constant. From these results, it was not possible to determine the required values of covariance that can be fixed for the full system. Instead of finalising them in this first experiment, we initially changed the CNA against each covariance and after performing extensive investigation a covariance of 2 was selected

Table 5.3: Tracking performance metrics for different values of the process and measurement noise covariances, $q$ and $r$, where CNA = 20 and $F_{\text{lost}}$ = 10. The total sheep count in this paddock was 352.

(a) 80a ( Ground truth = 15,045)

| | **Metrics** | | | | | | |
|---|---|---|---|---|---|---|---|
| $q$ **and** $r$ | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 1 | 848 | 27623 | 43 | 15002 | 183.60% | 0.04 | -83.89% |
| 2 | 793 | 20359 | 43 | 15002 | 135.32% | 0.03 | -35.60% |
| 3 | 778 | 16760 | 49 | 14996 | 111.40% | 0.03 | -11.72% |
| 4 | 728 | 15463 | 47 | 14998 | 102.78% | 0.03 | -3.09% |
| 5 | 681 | 14109 | 46 | 14999 | 93.78% | 0.03 | 5.92% |
| 10 | 528 | 5862 | 48 | 14997 | 38.96% | 0.03 | 60.72% |

(b) 80b ( Ground truth = 19,798)

| | **Metrics** | | | | | | |
|---|---|---|---|---|---|---|---|
| $q$ **and** $r$ | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 1 | 375 | 634 | 591 | 19207 | 3.20% | 0.08 | 93.81% |
| 2 | 353 | 276 | 587 | 19211 | 1.39% | 0.08 | 95.64% |
| 3 | 353 | 231 | 581 | 19217 | 1.17% | 0.08 | 95.90% |
| 4 | 352 | 232 | 582 | 19216 | 1.17% | 0.08 | 95.89% |
| 5 | 352 | 234 | 582 | 19216 | 1.18% | 0.08 | 95.88% |
| 10 | 350 | 207 | 589 | 19209 | 1.05% | 0.08 | 95.98% |

Table 5.4: Tracking performance metrics with respect to variation in CNA, where $q$ = 2, $r$ = 2, $F_{\text{lost}}$ = 10 and the actual sheep count is 352.

(a) 80a (Ground truth = 15,045)

| CNA | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 20 | 793 | 20359 | 43 | 15002 | 135.32% | 0.03 | -35.60% |
| 40 | 363 | 111 | 45 | 15000 | 0.73% | 0.03 | 98.96% |
| 60 | 353 | 34 | 39 | 15006 | 0.23% | 0.03 | 99.51% |
| 80 | 353 | 34 | 39 | 15006 | 0.23% | 0.03 | 99.51% |
| 100 | 353 | 34 | 39 | 15006 | 0.23% | 0.03 | 99.51% |

(b) 80b ( Ground truth = 19798)

| CNA | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 20 | 353 | 276 | 587 | 19211 | 1.39% | 0.08 | 95.64% |
| 40 | 350 | 176 | 578 | 19220 | 0.89% | 0.08 | 96.19% |
| 60 | 350 | 184 | 578 | 19220 | 0.93% | 0.08 | 96.15% |
| 80 | 350 | 175 | 578 | 19220 | 0.88% | 0.08 | 96.20% |
| 100 | 350 | 177 | 582 | 19216 | 0.89% | 0.08 | 96.17% |

for both videos.

The results reported in Table 5.4 were observed when CNA was varied between $[20, 40, 60, 80, 100]$, and other parameters were kept constant as mentioned in the table. As the CNA was increased, the system gets stricter in terms of linking the existing tracks with detected objects in each frame. New tracks were created either for those objects that appeared in the frame for the first time or for those which were not detected in the previous frames. A significant increase in the MOTA was observed for 80a, as the detector was performing better in this case, the proper adjustment in other parameters improved the overall performance, along with that of 80b.

It was observed that in the case of a sudden movement in the UAV flight, the tracks in the starting rows and ending rows of the frame were mostly affected. Keeping a high value of CNA made it difficult to create or delete tracks in such cases only, and track IDs of sheep that were leaving the frame were assigned to sheep that entered the

Table 5.5: Effect of varying $F_{\text{lost}}$ on tracking, where $q$ = 2, $r$ = 2, and CNA = 80. The actual sheep count is 352.

(a) 80a (Ground truth = 15,045)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 2 | 350 | 17 | 51 | 14994 | 0.11% | 0.02 | 99.55% |
| 4 | 352 | 22 | 40 | 15005 | 0.15% | 0.03 | 99.59% |
| 6 | 352 | 28 | 40 | 15005 | 0.19% | 0.03 | 99.55% |
| 8 | 353 | 32 | 39 | 15006 | 0.21% | 0.03 | 99.53% |
| 10 | 353 | 34 | 39 | 15006 | 0.23% | 0.03 | 99.51% |

(b) 80b ( Ground truth = 19798)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 2 | 344 | 57 | 648 | 19150 | 0.29% | 0.07 | 96.44% |
| 4 | 349 | 104 | 612 | 19186 | 0.52% | 0.07 | 96.38% |
| 6 | 350 | 132 | 599 | 19199 | 0.67% | 0.08 | 96.31% |
| 8 | 352 | 156 | 589 | 19209 | 0.79% | 0.08 | 96.24% |
| 10 | 354 | 175 | 582 | 19216 | 0.88% | 0.08 | 96.18% |

frame from the top. However, the system should create new tracks for new objects and delete tracks for those objects which leave the frame, and setting the CNA to a lower value caused the creation of new track IDs for existing tracks. To overcome this issue efficiently, the values of CNA were kept low for some upper and lower rows and 80 for the rest of the frame. However, we will refer to it as a fixed 80 value to avoid confusion.

Values of FPs and FNs could be reduced further by deleting the unassigned tracks earlier, which were allowed to stay for 10 frames. Two parameters were fixed after the previous experimental observation and now the $F_{\text{lost}}$ was varied on the range $[2, 4, 6, 8, 10]$. A different impact was observed for both videos, as shown in Table 5.5. While the value of MOTA decreased after 4 and 2 for 80a and 80b respectively, the value of SC varied oppositely. An SC close to the actual sheep count, 352, was obtained when the unassigned tracks were allowed to stay in the system longer than 6 frames. As

there is not much variation in the value of MOTA at this stage, an $F_{\text{lost}}$ of 6 was selected after this experiment. The reported MOTA of 99% and 96% and the accurate sheep counts are the best values reported so far in the field of livestock counting.

Figure 5.4 shows a cropped part from an intermediate frame of 80a and shows an example case of FN and FP. The Kalman filter predicted the next position of the sheep at a wrong position, shown in cyan color, that caused both an FP (cyan) and FN (blue) instance. Such cases were rare and happened when the sheep exhibited an unexpected movement. But there were cases too where the detector did not detect the sheep but the position was predicted correctly by the Kalman filter, as shown in the red color. It was marked as a matched track.

Figure 5.5 shows a particular case from 80b where the U-Net-MS model failed to detect a sheep in some intermediate frames but the same track IDs were assigned to the sheep in the later frames. The object detector did not detect the sheep that has a tracking ID of 117 on it, however, the Kalman filter tracked it successfully in this frame. In the next frame, the U-Net-MS model detected it, and the Hungarian algorithm assigned the same track ID to this sheep. For the third and fourth frames, the sheep with ID 116 faced the same problem. In both Figures 5.4 and 5.5 the best values of the discussed parameters were used.

Refer to Figure 5.6 for the graphs of matched object counts and ground truth, also


Figure 5.4: An example of different errors.

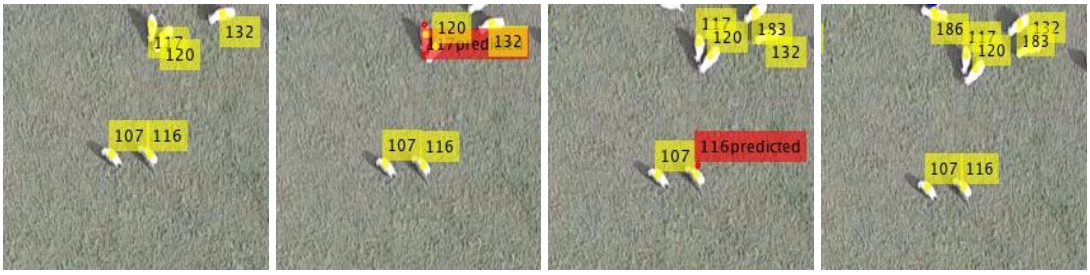



Figure 5.5: Cropped sub-images from four successive frames (left to right).

named as matched count and true count, respectively. Two graphs were plotted for each video: one graph shows an overlap between true and matched counts in each frame and the other shows per frame error between these counts. For 80a, it can be seen that there was a difference of only one sheep in some frames and overall tracking results were very encouraging. However, for video 80b, the highest error of 9 was observed in a few frames. As only those sheep were counted in each frame that were leaving the frame, this error did not affect the SC for the full video.

### 5.3.2 Livestock Tracking at 120 m

We recorded a few videos at 120 m in larger paddocks with a cumulative headcount in all frames equal to 76,786. At this altitude, the object detector failed in a few cases, especially when sheep were standing close together in a group. Due to a higher FNs in this case, the performance of the object tracker was affected. As the sheep look like a very small whitish blob from this altitude, the task of ground truth labelling was very challenging. A few frames needed to be inspected very closely to identify how many sheep a particular blob contained.

For the videos, 120a to 120d, the same process was repeated for selecting parameters and other factors to increase the tracking performance of the system. Initially, the covariance values were changed, while keeping the CNA and $F_{\text{lost}}$ constant. These results are reported in Table 5.6 (tables from this one onward, are placed at the end of

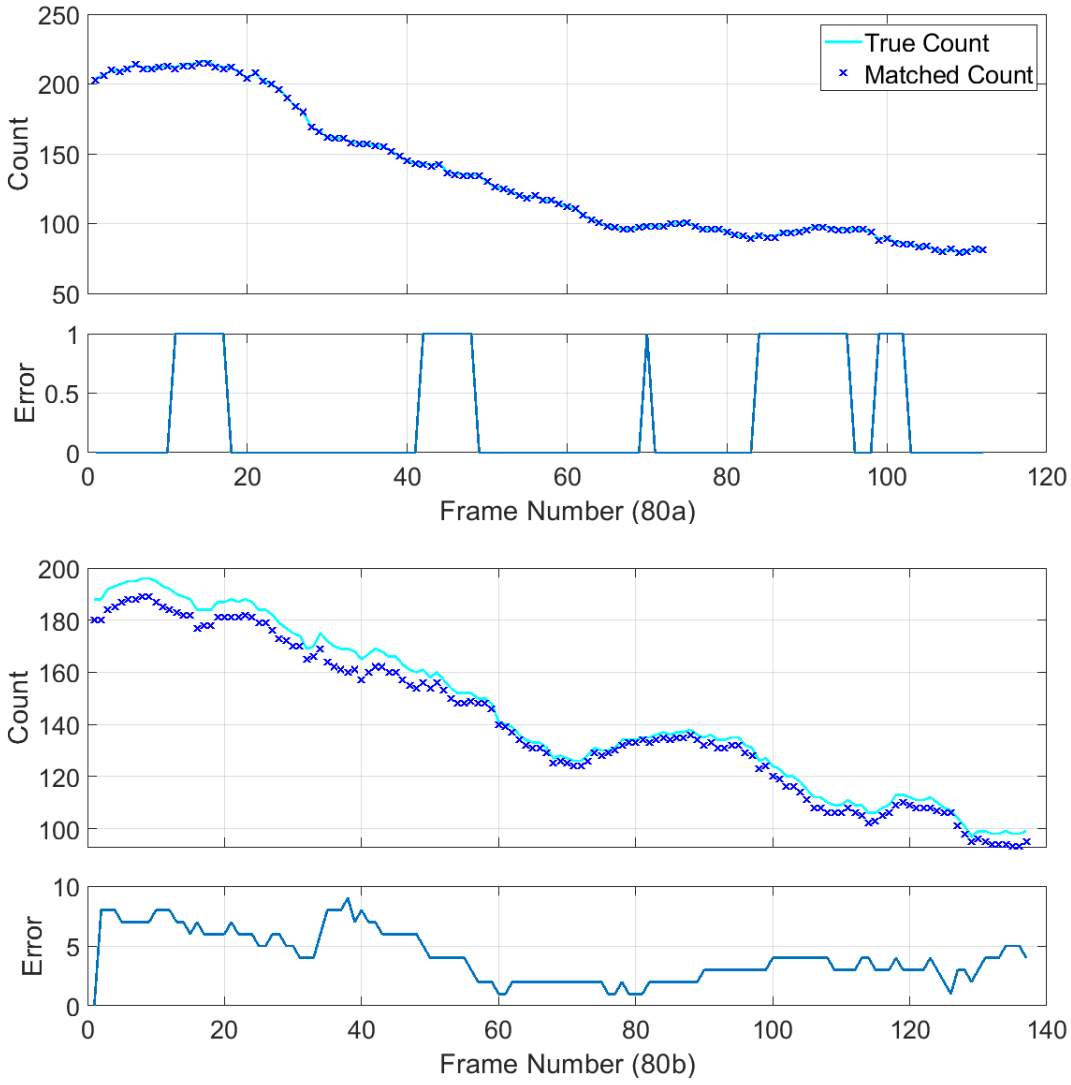



Figure 5.6: True and matched counts in respective frames for the videos 80a and 80b (top to bottom), the lower plot in each highlights the per frame difference between these two values. The estimated and true sheep count had a difference of zero and two for the full videos 80a and 80b respectively.

this chapter), and less variation was observed for the covariance values of 2 and above. To verify that we can use the same $q$ and $r$ for 80 m and 120 m, we repeated the task of varying CNA for covariance values of 2 and 3. As it was already observed for 80 m videos that a higher CNA links the tracks and detections more accurately, not much variation in the performance of the system was observed.

We have presented the results in Table 5.7, where $q$ and $r$ were kept constant at 2 and CNA is varied. The results for covariance values at 3 and varying CNA were very similar. The performance metrics for these videos with respect to varying $F_{\text{lost}}$ are provided in Table 5.8. Identifying and tracking these many objects with high accuracy is a significant progress in this field.

The same parameters were finalised for 120 m, as were decided for 80 m. The covariance value of 2 and $F_{\text{lost}}$ equal to 6 were chosen. As for CNA, for the starting and ending few frames of the video the value of 5 was used and 80 for the rest of the frame.

A few more factors were changed after finalising these parameters. One of them was the use of average velocity using (5.7) to initiate the object's state in (5.1). This decreased the number of FN and FP and hence increased MOTA. The second factor was the use of a different cost matrix calculation function. The function we were initially using to compute the distance between the detections and tracks for designing the cost matrix incorporated the noise covariance values. This is an important factor to involve when there is a chance of occlusion and objects can go behind other objects in the video. In the case of aerial videos, there is no occlusion and objects get only close to each other. Omitting the use of covariance values for cost calculation and replacing them with the Euclidean distance function improved the results a little. Although it was not a very significant difference as the values of MOTA and MOTPE were already better, improving the last few percentage points is the main challenge. We used these options to reduce that difference a bit more. The result of object tracking with the best settings identified is shown in Table 5.9 for all the recorded videos. Table 5.6 to 5.9 are added at the end of this chapter.

Figures 5.7 and 5.8 show graphs of true count and matched count, and the difference between these values for 120a to 120d. Videos 120a, 120b, and 120c were recorded in cloudy weather and as the U-Net-MS performs better in such weather, the overall tracking performance of our system was outstanding. In 120a, there were many groups

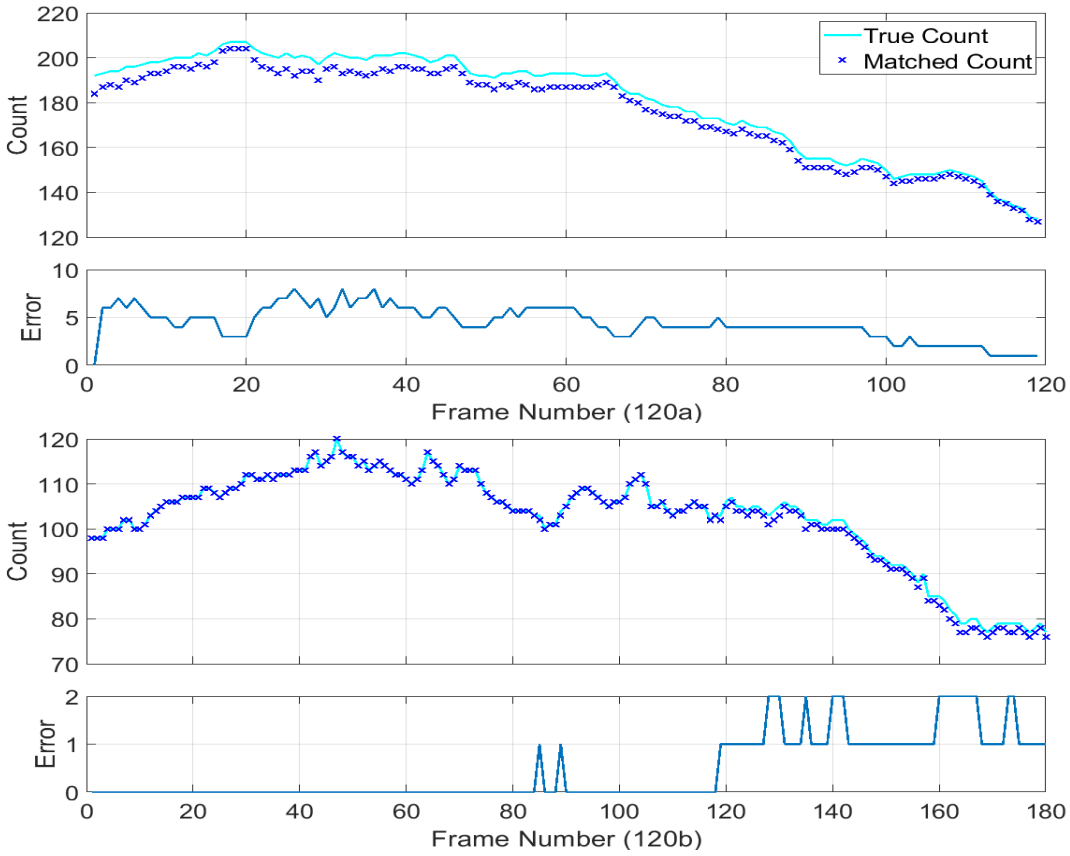



Figure 5.7: True and matched counts in respective frames for the videos 120a and 120b (top to bottom), each lower plot highlights the per frame difference between these two values.

of sheep, and it was difficult for the author to identify the correct number of sheep in such groups. However, the per frame error did not exceed the value of 9 and SC was very close to actual sheep number. We did not face this challenge in 120b and 120c, and the highest per frame difference of 2 and 4 was observed respectively. 120d was recorded in summer, with full sunlight, and had many challenging items within paddock which were considered as sheep by both author and the object objector, as shown in Figure 5.3. These factors affected the tracking performance of the proposed system in this video.

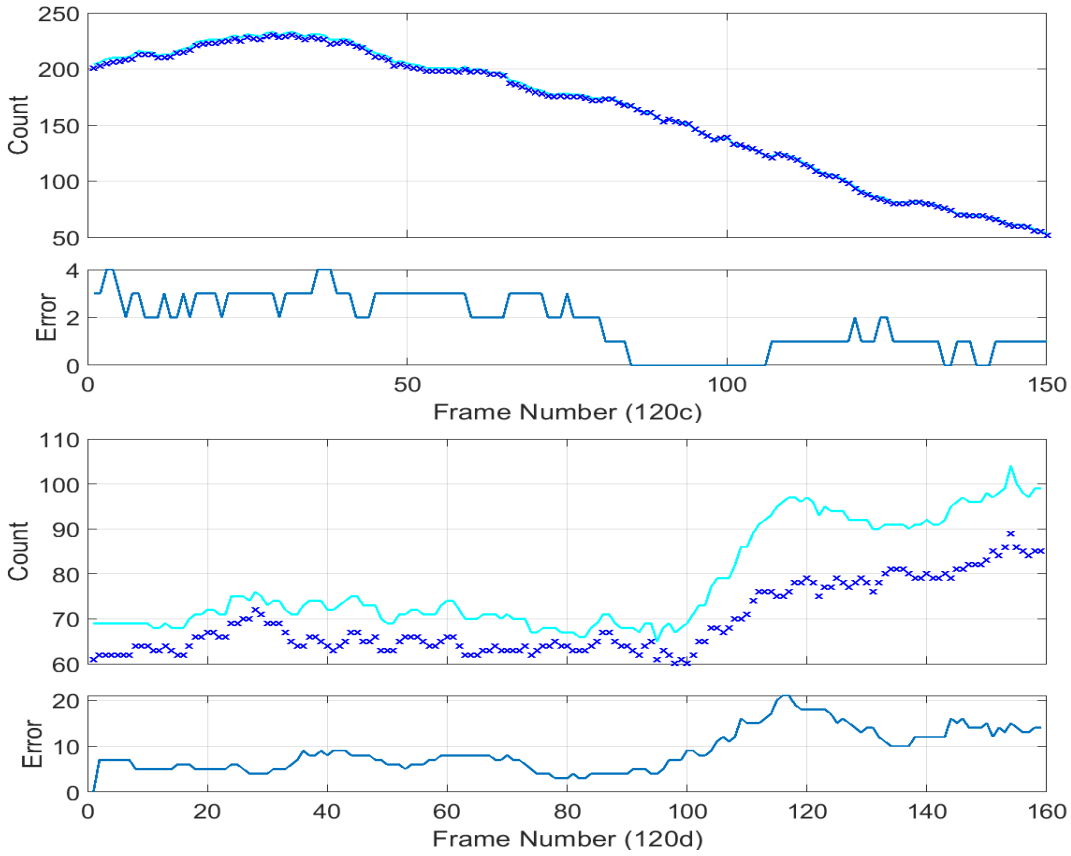



Figure 5.8: True and matched counts in respective frames for the videos 120c and 120d (top to bottom), with per frame difference shown in the second graph.

## 5.4 Concluding Remarks

A tracking-by-detection method for tracking and counting livestock in the offline aerial videos of different paddocks was presented in this chapter. We used the U-Net-MS model, which detected the livestock with high recall at both altitudes. Other parameters could be tuned as a result. The main factors of the Kalman filter as well as the Hungarian algorithm were varied over different ranges to see the improvement in the overall tracking system. The main performance metrics are RPF, MOTA and SC. High RPF shows that a high number of sheep are detected in the image by the object detector and there are fewer FN detections. As MOTA accounts for the overall errors of the tracker, so higher MOTA is required by our system. Using the most appropriate

combinations of the tested parameters, a high MOTA of around 98% was achieved at both altitudes. However, such performance may not be possible if the failure rate of the object detector increases beyond a certain limit. The metric SC should be close to zero as it refers to the difference between actual and estimated sheep count. It was less than 5 for every paddock, so the proposed system can be used for sheep counting in different paddocks using a UAV.

Table 5.6: Tracking performance metrics for various values of the process and measurement noise covariances, $q$ and $r$, where CNA = 20 and $F_{\text{lost}}$ = 10.

(a) 120a (Ground truth = 21,458, Actual sheep count = 276)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 1 | 273 | 87 | 516 | 20942 | 0.41% | 0.08 | 97.19% |
| 2 | 273 | 88 | 521 | 20937 | 0.41% | 0.08 | 97.16% |
| 3 | 273 | 90 | 522 | 20936 | 0.42% | 0.08 | 97.15% |
| 4 | 273 | 90 | 523 | 20935 | 0.42% | 0.08 | 97.14% |
| 5 | 273 | 90 | 524 | 20934 | 0.42% | 0.08 | 97.14% |
| 10 | 273 | 96 | 521 | 20937 | 0.45% | 0.08 | 97.12% |

(b) 120b (Ground truth = 18,501, Actual sheep count = 254)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 1 | 262 | 231 | 76 | 18425 | 1.25% | 0.03 | 98.34% |
| 2 | 255 | 113 | 76 | 18425 | 0.61% | 0.03 | 98.98% |
| 3 | 255 | 94 | 76 | 18425 | 0.51% | 0.03 | 99.08% |
| 4 | 255 | 95 | 76 | 18425 | 0.51% | 0.03 | 99.08% |
| 5 | 255 | 94 | 76 | 18425 | 0.51% | 0.03 | 99.08% |
| 10 | 255 | 91 | 77 | 18424 | 0.49% | 0.03 | 99.09% |

(c) 120c (Ground truth = 24,373, Actual count = 362)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 1 | 410 | 3229 | 261 | 24112 | 13.25% | 0.05 | 85.68% |
| 2 | 361 | 620 | 264 | 24109 | 2.54% | 0.05 | 96.37% |
| 3 | 361 | 422 | 264 | 24109 | 1.73% | 0.05 | 97.18% |
| 4 | 361 | 376 | 271 | 24102 | 1.54% | 0.05 | 97.34% |
| 5 | 361 | 387 | 270 | 24103 | 1.59% | 0.05 | 97.30% |
| 10 | 361 | 374 | 270 | 24103 | 1.53% | 0.05 | 97.36% |

(d) 120d (Ground truth = 12,454, Actual sheep count = 203)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 1 | 227 | 1582 | 1414 | 11040 | 12.70% | 0.14 | 75.94% |
| 2 | 207 | 1231 | 1422 | 11032 | 9.88% | 0.14 | 78.70% |
| 3 | 208 | 1173 | 1422 | 11032 | 9.42% | 0.14 | 79.16% |
| 4 | 208 | 1142 | 1427 | 11027 | 9.17% | 0.13 | 79.37% |
| 5 | 206 | 1111 | 1428 | 11026 | 8.97% | 0.13 | 79.56% |
| 10 | 199 | 1033 | 1443 | 11001 | 8.29% | 0.12 | 80.11% |

Table 5.7: Tracking performance metrics with respect to variation in CNA, where $q$ = 2, $r$ = 2, $F_{\text{lost}}$ = 10.

(a) 120a (Ground truth = 21,458, Actual sheep count = 276)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 20 | 273 | 88 | 521 | 20937 | 0.41% | 0.08 | 97.16% |
| 40 | 273 | 88 | 521 | 20937 | 0.41% | 0.08 | 97.16% |
| 60 | 273 | 88 | 521 | 20937 | 0.41% | 0.08 | 97.16% |
| 80 | 273 | 88 | 521 | 20937 | 0.41% | 0.08 | 97.16% |
| 100 | 273 | 88 | 521 | 20937 | 0.41% | 0.08 | 97.16% |

(b) 120b (Ground truth = 18,501, Actual sheep count = 254)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 20 | 255 | 113 | 76 | 18425 | 0.61% | 0.03 | 98.98% |
| 40 | 254 | 82 | 76 | 18425 | 0.44% | 0.03 | 99.15% |
| 60 | 254 | 82 | 76 | 18425 | 0.44% | 0.03 | 99.15% |
| 80 | 254 | 82 | 76 | 18425 | 0.44% | 0.03 | 99.15% |
| 100 | 254 | 82 | 76 | 18425 | 0.44% | 0.03 | 99.15% |

(c) 120c (Ground truth = 24,373, Actual sheep count = 362)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 20 | 361 | 620 | 264 | 24109 | 2.54% | 0.05 | 96.37% |
| 40 | 361 | 341 | 263 | 24111 | 1.40% | 0.05 | 97.53% |
| 60 | 362 | 335 | 263 | 24110 | 1.37% | 0.05 | 97.55% |
| 80 | 362 | 335 | 263 | 24110 | 1.37% | 0.05 | 97.55% |
| 100 | 361 | 333 | 263 | 24110 | 1.37% | 0.05 | 97.55% |

(d) 120d (Ground truth = 12,454, Actual sheep count = 203)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 20 | 207 | 1231 | 1422 | 11032 | 9.88% | 0.13 | 78.70% |
| 40 | 201 | 1010 | 1427 | 11027 | 8.11% | 0.13 | 80.43% |
| 60 | 201 | 1004 | 1427 | 11027 | 8.06% | 0.12 | 80.48% |
| 80 | 200 | 996 | 1435 | 11019 | 8.00% | 0.12 | 80.48% |
| 100 | 200 | 996 | 1435 | 11019 | 8.00% | 0.12 | 80.48% |

Table 5.8: Effect of varying $F_{\text{lost}}$ on tracking, where $q$ = 2, $r$ = 2, and CNA = 80.

(a) 120a (Ground truth = 21,458, Actual sheep count = 276)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 2 | 272 | 40 | 639 | 20819 | 0.19% | 0.06 | 96.83% |
| 4 | 272 | 62 | 588 | 20870 | 0.29% | 0.07 | 96.97% |
| 6 | 273 | 85 | 565 | 20893 | 0.39% | 0.07 | 96.97% |
| 8 | 273 | 84 | 536 | 20922 | 0.39% | 0.08 | 97.11% |
| 10 | 273 | 99 | 523 | 20935 | 0.46% | 0.08 | 97.10% |

(b) 120b (Ground truth = 18,501, Actual sheep count = 254)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 2 | 252 | 56 | 114 | 18387 | 0.30% | 0.03 | 99.08% |
| 4 | 253 | 71 | 94 | 18407 | 0.38% | 0.03 | 99.11% |
| 6 | 253 | 79 | 89 | 18412 | 0.43% | 0.03 | 99.09% |
| 8 | 254 | 83 | 83 | 18418 | 0.45% | 0.03 | 99.11% |
| 10 | 254 | 82 | 76 | 18425 | 0.44% | 0.03 | 99.15% |

(c) 120c (Ground truth = 24,373, Actual sheep count = 362)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 2 | 359 | 170 | 336 | 24037 | 0.70% | 0.04 | 97.92% |
| 4 | 359 | 230 | 304 | 24069 | 0.94% | 0.04 | 97.81% |
| 6 | 359 | 270 | 278 | 24095 | 1.11% | 0.05 | 97.75% |
| 8 | 362 | 307 | 272 | 24101 | 1.26% | 0.05 | 97.62% |
| 10 | 363 | 327 | 264 | 24109 | 1.34% | 0.05 | 97.57% |

(d) 120d (Ground truth = 12,454, Actual sheep count = 203)

| $F_{\text{lost}}$ (frames) | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SC** | **FP↓** | **FN↓** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 2 | 196 | 961 | 1509 | 10945 | 5.55% | 0.11 | 82.33% |
| 4 | 198 | 809 | 1464 | 10990 | 6.50% | 0.12 | 81.75% |
| 6 | 199 | 886 | 1445 | 11009 | 7.11% | 0.12 | 81.28% |
| 8 | 200 | 948 | 1439 | 11015 | 7.62% | 0.12 | 80.83% |
| 10 | 201 | 995 | 1435 | 11019 | 8.00% | 0.12 | 80.48% |

Table 5.9: Performance metrics for all videos where $q$, $r$, CNA and $F_{\text{lost}}$ were set at 2, 2, 80 and 6 respectively. Euclidean distance was used for cost matrix computation and average velocity of objects were provided during tracks initialization.

| **Metrics** | **Video Data** | | | | | |
|---|---|---|---|---|---|---|
| | 80a | 80b | 120a | 120b | 120c | 120d |
| Actual Count | 352 | 352 | 276 | 254 | 362 | 203 |
| Sheep Count | 352 | 351 | 273 | 253 | 359 | 199 |
| Count Difference | 0 | 1 | 3 | 1 | 3 | 4 |
| Total GT | 15,045 | 19,798 | 21,458 | 18,501 | 24,373 | 12454 |
| Total MC | 15,009 | 19,199 | 20,924 | 18,422 | 24,099 | 11030 |
| Matched Difference | 36 | 599 | 534 | 79 | 274 | 1,424 |
| Total FP | 19 | 93 | 50 | 81 | 245 | 811 |
| Total FN | 36 | 599 | 534 | 79 | 274 | 1424 |
| FPR | 0.13% | 0.47% | 0.23% | 0.44% | 1.01% | 6.51% |
| MOTPE | 0.03 | 0.07 | 0.08 | 0.03 | 0.05 | 0.11 |
| MOTA | 99.63% | 96.50% | 97.28% | 99.13% | 97.87% | 82.05% |

# Chapter 6

# Paddock Fence Detection

In New Zealand, a fenced pasture is known as a paddock. The average farm area in New Zealand is 270 ha [2] and most of the sheep farms have multiple paddocks. A farm manager has to keep on shifting livestock among different paddocks. This gives time to the grass and other plants for growth, which is then used as food by sheep. Farm managers are most interested in knowing the number of livestock within a particular fenced pasture, which raised the necessity of detecting paddock fences in a UAV video.

A research paper based on the method and results presented in this chapter will be published in a conference as follows:

**C5:** F. Sarwar, A. Griffin, P. Chong and T. Pasang, "Pasture fence line detection in UAV videos", *in the proceedings of 36th International Conference on Image and Vision Computing New Zealand, (IVCNZ)*, 2021.

## 6.1 Fence Detection Method

In sheep farms of New Zealand, there are mostly two different types of fence designs. One type is a multi-wire, multi-batten design and the other one is a post and wire (no battens) design. In the first design, there are wooden posts spaced 3.5 m to 4 m apart,

and 3 to 5 battens between two wooden posts, while in the second design, there are only wooden posts with wires, and no battens are placed between them. The paddocks that were used for data collection have fences of both these types and the datasets created for fence detection—FenceSet80 and FenceSet120—have centroids on wooden posts only. However, one of the paddock fences had a different top structure and there was a wooden plank on the top of the full fence. The network was trained on all the above-mentioned fence types and a few examples are shown in Figure 6.1 from FenceSet80. A mixture of all these fence designs made our system robust for accurate fence detection.

The details of data collection and dataset are explained in Chapter 3. Two datasets were designed for detecting fence lines at 80 m and 120 m, and both datasets were divided into $80:10:10$ proportion for training, validation and testing the CNN network. As the U-Net model outperformed other networks for sheep detection, the centroids of the wooden posts were used as ground truth values to train the U-Net model for fence detection too. The main steps for training this network are similar as mentioned in Chapter 4 and will be briefly discussed here. The U-Net model was trained to detect fence posts at 80 m and 120 m using FenceSet80 and FenceSet120 separately. The training method and parameters for the U-Net model were kept the same as were mentioned in Section 4.3. A batch size of 10 images was used and the U-Net model was trained for 500 epochs using SGD with 0.9 momentum and $1 \times 10^{-4}$ learning rate. The

(a) Multi-wire and multi-batten fence

(b) Post and wire fence design

(c) Fence with a wooden plank on top

Figure 6.1: Fence types in FenceSet80 dataset.

trained network was validated after every epoch over the full validation set. Training the U-Net model was just the first step in fence detection and the overall system is much more complex than sheep detection. However, the testing phase is different and Figure 6.2 shows the flowchart of fence detection in a video frame.

Similar to the testing phase of sheep detection, each test image or frame was

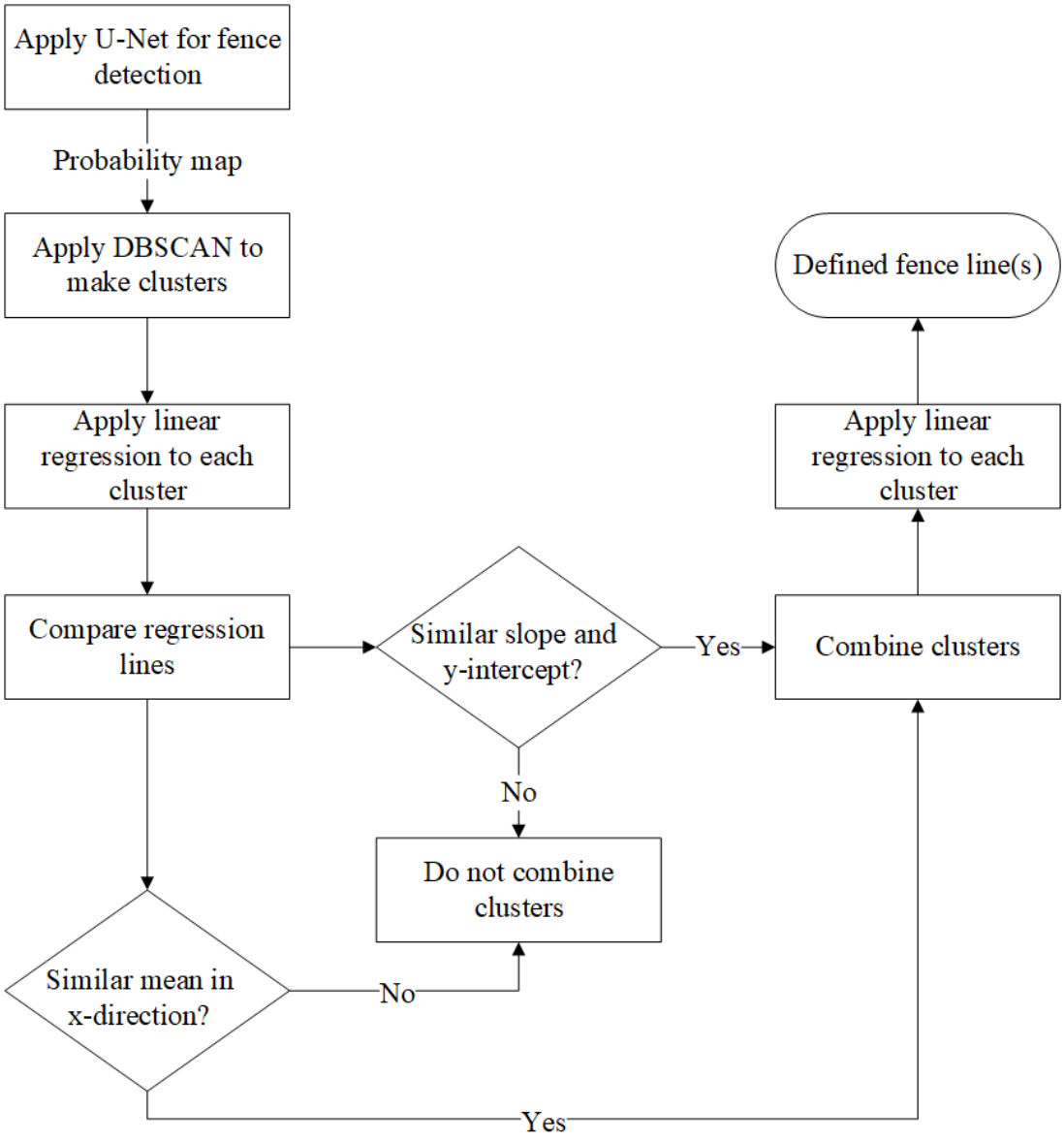



Figure 6.2: A system diagram for paddock fence detection.

divided into sub-images of 256×256×3 dimension with an overlap of 20 pixels. Here, the overlapping pixel value can be changed to decrease the inference time per frame. However, we used the same value to keep the initial testing phase similar to sheep detection. The probability maps provided by the U-Net model for all sub-images for the respective frame were then stitched together using the same method as mentioned in Section 4.4.1. One example frame from video 80a and respective stitched probability map computed from the above-mentioned method are shown in Figure 6.3(a) and (b) respectively.

The highest probability values are usually at or near the centroids of the objects and it can be seen in Figure 6.3 (b) that this respective frame has three fence lines. In this case, the data in the form of centroid values of respective fence posts are not sufficient for creating fence lines either in the sub-image or the full frame. To define fence lines on different sides, we need to devise a method that can work well even if the network fails to detect some intermediate fence posts.

The density-based spatial clustering of applications with noise (DBSCAN) [184] is an effective approach to create clusters in this case. This clustering algorithm helps to remove some of the FP detections by categorizing them as a noise cluster. Linear regression is then applied repeatedly to the clusters created by DBSCAN to create complete fence lines. The major steps of fence detection are further explained in detail in the following subsections.

### 6.1.1 Clustering Using DBSCAN

DBSCAN is a well-known data clustering algorithm that was proposed by Ester *et al.* [184] in 1996. It is commonly used in machine learning and data mining problems, and groups different points in clusters depending on the density and distance between them. Those points, which are far away from higher density regions, are marked as

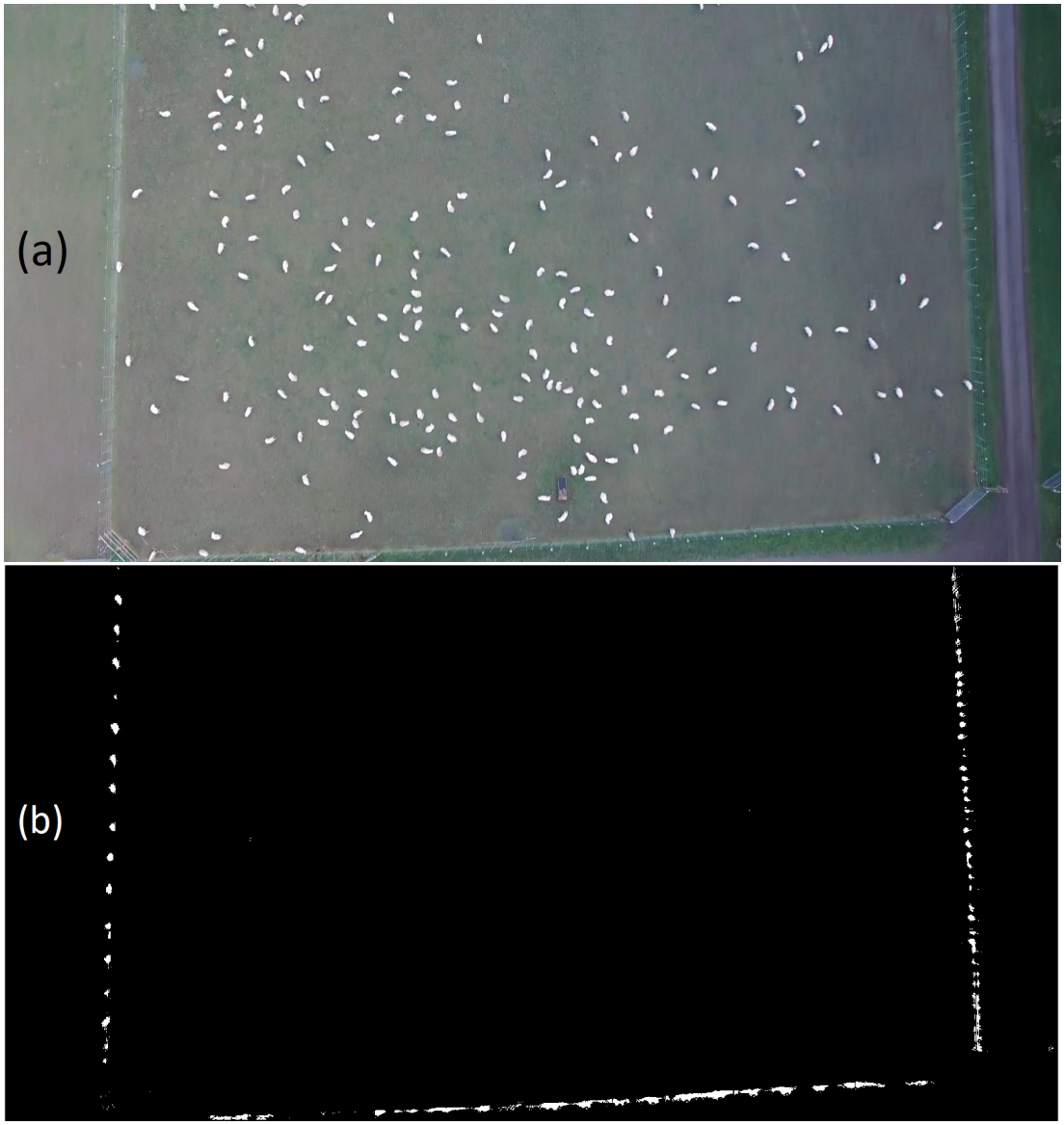



Figure 6.3: A probability map shown in (b) generated by the U-Net model for a frame from 80a video shown in (a).

noise.

DBSCAN selects a starting point at random and starts exploring the neighbouring points, if there are at least a minimum number of points in that area then this starting point is marked as a core point. This process starts the formation of a cluster. If there are not enough points then this starting point is marked as noise. However, if the cluster

formation starts, all those neighbouring points that are within allowable distance become a part of this cluster. If some other core points lie within this neighbourhood then all those points will also become the part of this cluster that are within allowable distance from new core points. This process keeps on repeating until DBSCAN visits all points. Some border points can be labelled as noise by DBSCAN if they are not within the provided distance from core points. When one such cluster is defined, another point is selected randomly to repeat the same process. DBSCAN keeps on making clusters in this way and this process is finished once all points are visited and labelled.

To define such clusters, two parameters must be provided to this algorithm:

- Minimum points: the minimum number of values for defining the dense region. If this value is set to 100, then there must be at least 100 points to make a cluster.
- Allowable distance: the maximum distance between points to link them to any cluster. If the distance between points is less than or equal to this value then these points will be a part of the respective cluster.

If any point does not lie within allowable distance from a higher density region, then it is declared as noise. The selection of the above-mentioned parameters depends on the required output. For our dataset, each video frame had a maximum of three fence lines: a left fence line, a right fence line and a bottom or top fence line. If we knew there were only a left and right fence line then the allowable distance could be set higher to define only two clusters. But such selection of the distance value will not identify enough clusters when there is either a bottom or top fence within the respective frame. After a few experiments, it was decided to keep a cluster of a minimum of 200 points and an allowable distance of 50 pixels. This helped to create multiple clusters throughout the image, as shown in Figure 6.4(a). Increasing the allowable distance at times merged the corner points of different fences, making it one big cluster. Below this value, the clusters were too small and more border points were labelled as noise. The probability map showed in Figure 6.3 is plotted over the respective RGB frame to give a more clear

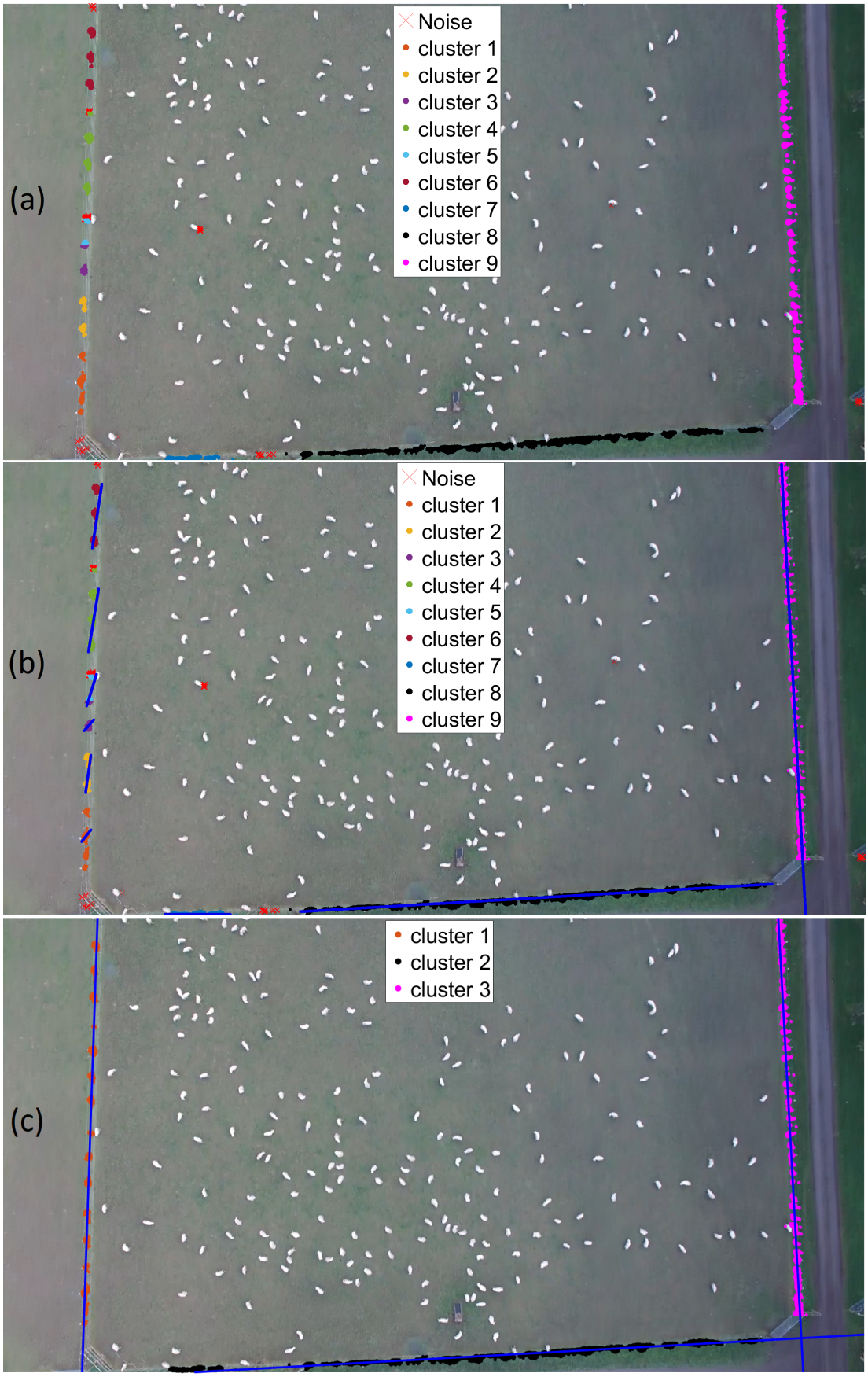



Figure 6.4: (a) Clusters obtained after DBSCAN, (b) Linear regression applied to each cluster and (c) Clusters combination to define respective fences.

picture in Figure 6.4(a), (b) and (c). As per the above-mentioned settings, a total of ten clusters were identified by DBSCAN, where one of them was a noise cluster. The points categorised as noise were not used while designing the fence lines.

### 6.1.2 Linear Regression

The paddocks used for data collection had relatively straight lines. If some points can be detected on a respective fence then a polynomial of the first order can be used to define the fence lines on each side. At the start of the video, there can be fences on the left, bottom and right side of the frame(s). As the UAV moves forward, only the left and right fences will be visible in the video frames. Near the end of the video, the top fence will appear and again there will be a total of three fences.

For defining lines from the estimated clusters, linear regression is used. Linear regression creates a predictive mathematical model for the estimated data. The relationship between the dependent and independent data values is modelled using a linear predictor function. The values of this unknown linear predictor are estimated from the provided data. Given the predictor values, linear regression focuses on the conditional probability of the response. In simple words, linear regression tries to fit a linear equation of first-order between dependent and independent variables by minimizing the error. Some of these existing approaches are the least-square fitting method, bisquare weights method, Cauchy, Huber, talwar, etc. The least-square method can be used to even fit those models that are not linear. If there are multiple curves in the fence then polynomial equations of higher order can be fitted to get the required lines.

A linear regression model has an equation $y = \beta_0 + \beta_1 x$, where $x$ is the independent variable. This equation has a slope $\beta_0$ and y-intercept $\beta_1$. We separately applied linear regression to each cluster using a robust least square fitting approach. This is an iterative method that keeps on removing the outliers to minimize the error between actual and

predicted values. There is no exact definition for an outlier data value and it usually depends on the respective system. In our case, those data points are categorized as outliers that are at more than the allowable distance from the centroid of the cluster in the x-direction for left and right fences and in y-direction for the bottom and top fence. Note that the core points defined by DBSCAN and the centroid values of all clusters can be different values.

Such outlier points pull the fitted model far from the centre by receiving more weight, hence distorting the result. The use of the robust least square method overcomes this problem. The results obtained after applying linear regression are shown in Figure 6.4(b). The x-coordinates of each cluster were used as an independent variable, whereas y-coordinates were predicted and compared.

For each such frame, we get a fitted model for each cluster, but all these models still need further processing and cannot be used efficiently. The next step was to combine all these lines using some user-defined conditions.

### 6.1.3 Combining Clusters

To combine clusters on the same side of the frame, a few conditions were defined which are as follows:

1. Combine clusters if the difference between slope is negligible and y-intercepts are also close to each other.
2. Combine clusters if the difference between the mean value of their independent variables, x-coordinates, is below a threshold.
3. Remove those points from these newly designed clusters which are far from the generated lines to get more accurate results.

The first condition helped by combining the lines that are on the same side of the image. A similar case of such line equations are for cluster 4 and cluster 6 in

Figure 6.4(b). Due to different density regions, the values of the slope and y-intercept between two nearby clusters can also have a large difference, as can be seen for cluster 1 and cluster 2 in Figure 6.4(b). For such cases, the second condition is used. So, even if the slopes are different but the clusters are on the same fence, then these clusters should be combined.

The first two conditions are applied twice on the full set of clusters to combine as many as possible. After combining them, linear regression was applied again. As a robust least-square fitting method was used, this fulfilled the third condition and generated straight lines. Three fence lines are shown in Figure 6.4(c).

### 6.1.4 Performance Metrics

The performance metrics for fence detection are different in comparison to what we used for sheep detection. In the case of sheep, centroid values were used and improving recall was the main objective. However, in fence detection, we are comparing the actual and estimated fence lines. The actual or ground truth fence line was created using the ground truth fence post points. Linear regression was applied only once for each side to generate the linear model, while the estimated fence line was created using the procedure shown in Figure 6.2. These lines are compared according to the predicted y-coordinate values when the same input matrix of x-coordinates is provided. The slopes were also compared for each fence to observe a difference between estimated and actual lines.

The main metrics used were mean absolute error (MAE), root mean square error (RMSE) and mean average percentage error (MAPE). MAE is a way of comparing estimated and actual values based on the same scale as measured data. It is the simplest

way of comparing points and is defined as:

$$\text{MAE} = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N}, \tag{6.1}$$

where $y_i$ and $\hat{y}_i$ are the actual and estimated y-coordinate values for the respective fence line.

The RMSE measures the standard deviation of the prediction error. This prediction error, also known as the residual, measures how far the data points are from the regression line. Or in other words, how much the data is concentrated around the estimated line. It is computed as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N}}. \tag{6.2}$$

MAPE is explained in Section 4.4.4 and is used to show the accuracy of the predictor.

We have used MAE, RMSE and MAPE to compare the estimated and actual points on respective lines. We have also compared slope values for all fences and used MAE and RMSE as the performance metrics. MAPE was not used for slope comparison as the slope for left and right fences were less than one, which show a drastic rise in MAPE. Hence, it is not a good measure for very small data points.

All these performance metrics were averaged across all frames to get per frame metric values.

## 6.2 Results

### 6.2.1 Fence Detection at 80 m

The U-Net model trained on the training set was validated on the validation set after each epoch during the training process. Once the training was complete, the whole

system was used to estimate fence lines on the frames of 80a and 80b using the process shown in Figure 6.2. The frames from the start of the video and near the end of the video had three fence lines. At the start, there are three fence lines, left, bottom and right. After a few frames only left and right fence lines can be seen and then near the end of the video, there are left, top and right fence lines. As there was less change in the left and right fences, the results were computed for every second frame until the bottom or top fence is visible, at which point every frame was processed.

Metrics were computed separately for the left, right, bottom and top fences. The results for fence data points and slopes are shown in Tables 6.1 and 6.2 respectively. The results obtained at this altitude are encouraging, showing that this system can be used for quite accurate fence detection. The slope errors were computed using the minimum angular distance [185] to avoid the biased results due to vertical and horizontal slopes.

These videos were recorded in the same paddock and a higher MAPE was observed for the top fence in both videos. The proportion of the fence images shown in Figure 6.1(c) was lower as compared to the other types as only there is one such fence line. The top wooden structure makes the actual fence posts less visible, making them difficult to be identified correctly by the U-Net model. The U-Net model still detected the fence posts in most of the images but an error in a few frames has impacted the

Table 6.1: Performance metrics for fence line points in the 80a and 80b videos.

| **Video** | **Fence Line** | **MAE** | **RMSE** | **MAPE** |
|---|---|---|---|---|
| 80a | Left | 2.52 | 2.71 | 1.51% |
| | Right | 3.10 | 3.31 | 0.17% |
| | Bottom | 2.57 | 3.00 | 0.24% |
| | Top | 1.63 | 3.40 | 2.06% |
| 80b | Left | 1.61 | 1.76 | 0.63% |
| | Right | 2.24 | 2.40 | 0.13% |
| | Bottom | 6.94 | 8.43 | 0.64% |
| | Top | 0.62 | 1.58 | 1.35% |

Table 6.2: Metric values for fence line slope for the videos 80a and 80b.

| Video | Fence Slope | MAE | RMSE |
|---|---|---|---|
| 80a | Left | 0.19 | 0.22 |
| | Right | 0.21 | 0.24 |
| | Bottom | 0.51 | 0.59 |
| | Top | 2.80 | 5.75 |
| 80b | Left | 0.15 | 0.18 |
| | Right | 0.15 | 0.19 |
| | Bottom | 1.21 | 1.53 |
| | Top | 2.38 | 6.19 |

performance metric for the full video.

The values of MAE and RMSE for respective fence lines are shown in Figure 6.5 and 6.6 for 80a and 80b respectively. For both videos, the left and right fences are visible for the full videos, while the bottom fence was visible for the first few frames. The top fence appeared near the end of the videos. For 80a, the left and the right fences are visible for the full video, while bottom fence was visible for first seven frames. Top fence appeared at frame # 102. Similarly, for 80b, the left and the right fences are visible for the full video, while bottom fence was visible for the first five frames, and the top fence appeared at frame # 122.

As there are many fence posts for each side, the ground truth labeling of the centroids may have some errors. A slight change in a few points on the respective side can change the regression line parameters as well as the actual line values. However, the highest MAE error of 6.94 among 1080 estimated and predicted values for one of the fences is considered acceptable. This high error is not for all fence lines, as can be seen in the graphs shown in Figures 6.5 and 6.6. The regression lines may vary with a variation in the estimated and actual clusters when only a small portion of bottom or top fence is visible in the frame. An example of a few such worst case scenarios are shown in Figure 6.7 where the actual fence line is very small, but the estimated fence line goes along the full lower boundary of the frame, hence causing a higher difference between

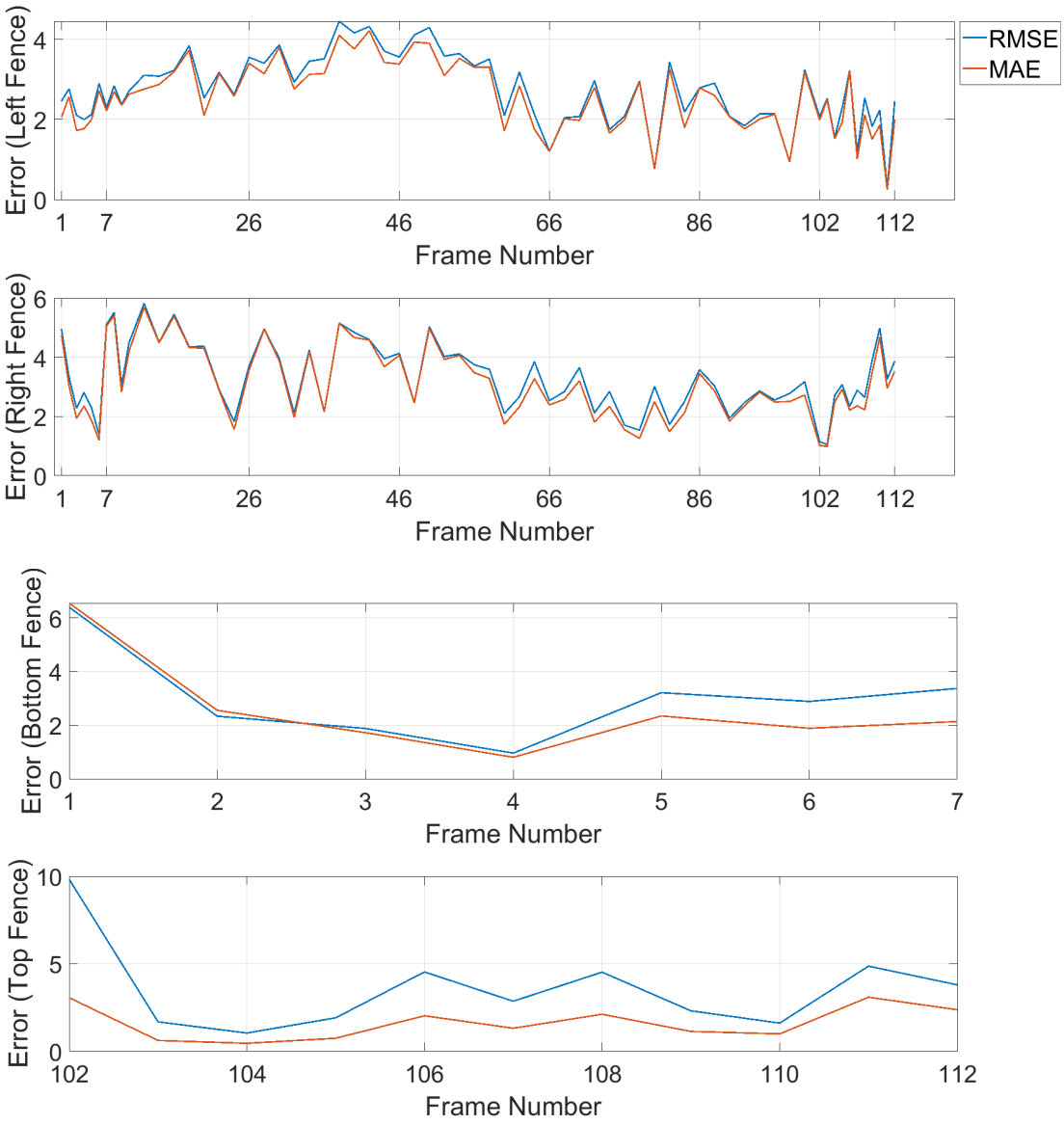



Figure 6.5: RMSE and MAE per frame for left, right, bottom and top fence line points (top to bottom) for video 80a.

fence points and the slope. Similarly, when a small portion of the top fence was visible, the estimated and actual fence lines did not match, as shown in Figure 6.8.

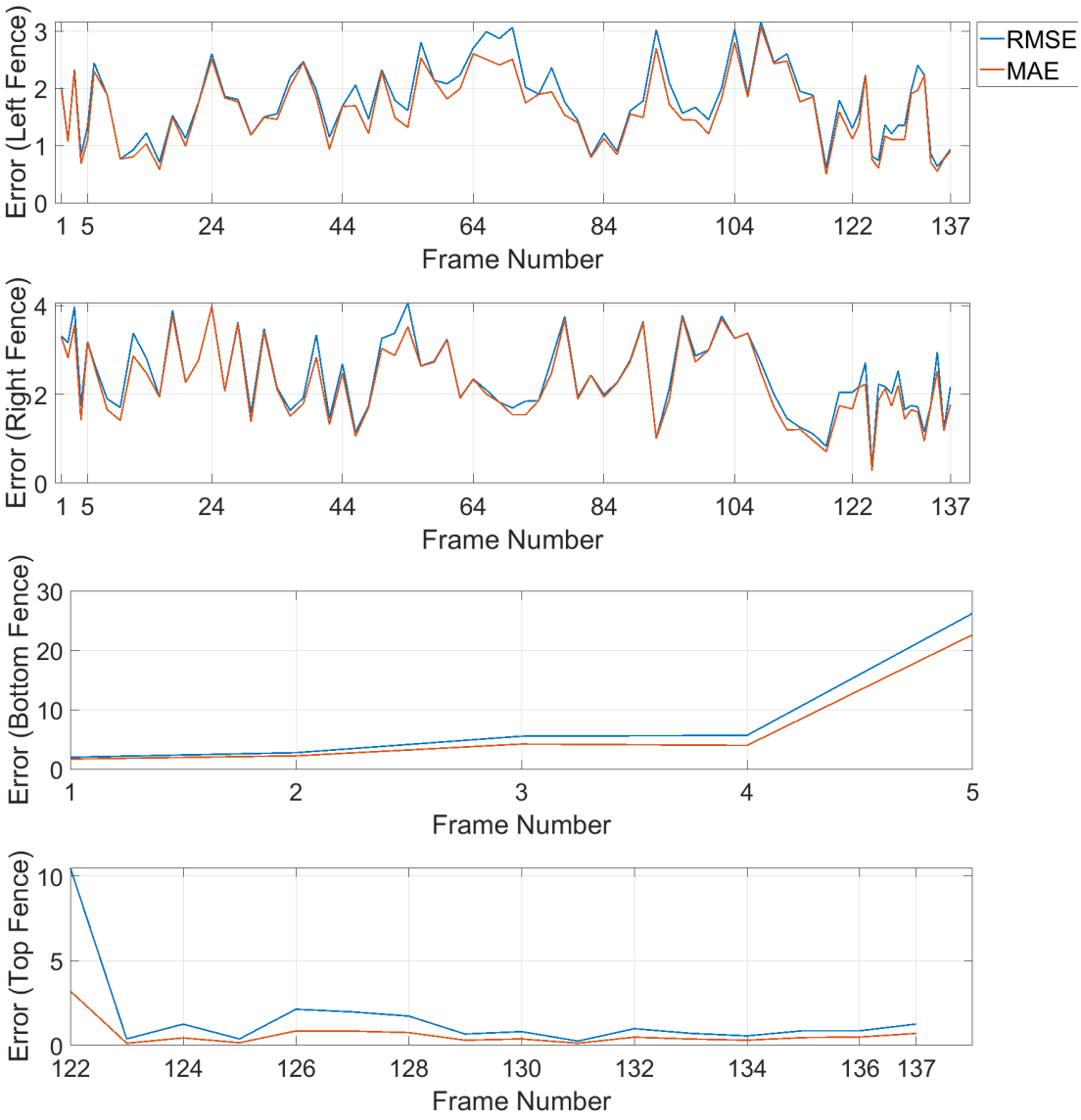



Figure 6.6: RMSE and MAE per frame for left, right, bottom and top fence line points (top to bottom) for video 80b.

### 6.2.2 Fence Detection at 120 m

The same U-Net model was trained for fence detection at 120 m. As the fence posts get merged with the background in most of the cases, it was not expected that the network will be able to detect fence lines accurately. The FenceSet120 was created many times and the network was trained on each dataset from scratch. A similar

Figure 6.7: Estimated and actual fence lines shown in blue and green respectively for the $5^{th}$ frame of 80b.

Figure 6.8: The $122^{th}$ frame of 80b with estimated and actual fence lines shown in blue and green, respectively.

process was used to create FenceSet120 as was used for all other datasets, i.e. a frame was downsampled by a factor of two and then RGB sub-images of 256×256 pixels were created. Data augmentation techniques like translation and rotation were used to

increase the size of dataset. However, as fence posts were already not visible at different spots, downsampling blurred the fences even more. Then another dataset was created using high resolution frames directly and except for downsampling, other steps were the same. The network was trained again but no noticeable improvement was observed in the results. It was then noticed that the network was over-fitting i.e. giving good results on validation set but bad results on the test set. Another dataset was created that was three times smaller than the previous set. Again, the results were not up to the required level.

The trained U-Net model was unable to detect the fence line correctly in any of the 120 m videos and gave very high FP detections. One such example is shown in Figure 6.9, where 6.9(a) shows one of the frames from 120d. The probability map estimated by the U-Net is shown in Figure 6.9(b) and 6.9(c) shows the probability map over main image. It can be seen that although fence lines are visible to some extent in the map, no clustering algorithm we investigated could help to properly identify any fence line. If locating fence posts in 120 m videos was a challenging task for the researcher, then it is difficult to expect good performance from any network. Due to a high number of FP values in the full frame, DBSCAN and linear regression failed to give appropriate clusters and lines, respectively.

## 6.3 Concluding Remarks

In this chapter, we discussed the fence detection method at 80 m and 120 m using the U-Net model. The network was separately trained on FenceSet80 and FenceSet120 to perform detection task. The probability map provided by the U-Net model was divided into clusters using DBSCAN to create fence areas. Linear regression along with some conditions was used to create fence lines from the designed clusters. Promising results were obtained at 80 m, showing that this task can be performed to ignore the sheep

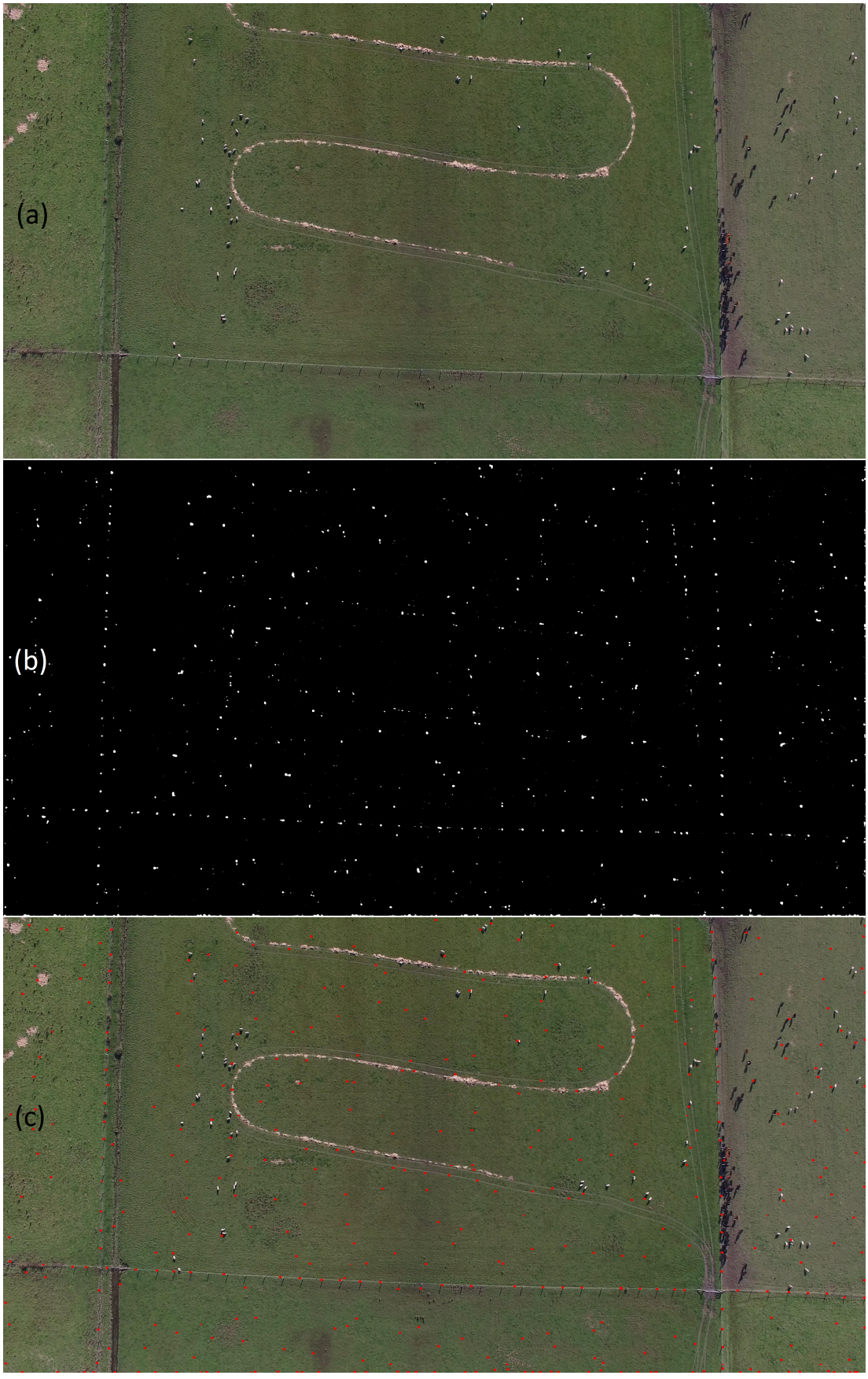



Figure 6.9: (a) Original frame, (b) Probability map, and (c) Probability map over original image.

detection outside the defined boundary. However, the same task cannot be performed at 120 m due to less visibility of fence posts and wires.

# Chapter 7

# Conclusions and Future Work

A brief synopsis of the research work done during the three years of this PhD study, discussion on results and future work are presented in this chapter. The challenges that were faced at different stages of this research, as well as the limitations of the proposed system, are also highlighted.

## 7.1 Synopsis

In New Zealand, pastoral farms vary in size and can contain large number of paddocks in each pasture. Larger farms make livestock counting a bit hectic for farmers, leading to financial loss in some cases. This research work mainly focused on providing an alternative solution for livestock (sheep) counting for pastoral farms of New Zealand. The proposed system is not limited to New Zealand only, it can be used anywhere for different object detection, tracking and counting tasks. We proposed an offline tracking-by-detection method using a unique combination of deep learning and machine learning techniques. A U-Net model was used for detecting sheep and fences in each video frame, and the sheep were tracked using the Kalman filter. The task of associating detections with tracks was done by the Hungarian algorithm. The high recall of sheep

detection and promising accuracy of the tracking makes the proposed system applicable for sheep counting.

### 7.1.1 Sheep Detection

In the context of object detection, two new two-stage CNNs were proposed. These new networks and a one-stage network, U-Net, were trained from scratch and a few state-of-the-art two-stage networks were fine-tuned. During the training phase of the CNN networks, there were a few important things we observed. The learning rate of SGD and the batch size were two hyperparameters that affected the network's performance. Depending on the SGD learning rate, networks can either reach the local minima only or achieve the global minima during the loss optimization process. The networks were not converging to any minima below $1 \times 10^{-4}$ and it was taking a very long time to converge above this value in our training phase. As a result of an in-depth study, the value stated here was selected after a detailed analysis of the network training process. The performance of the network is significantly affected by changes in these values, just as with batch size. To verify this theoretical point, we applied the same hyperparameters to train the U-Net model for the plant detection [62] and found that the recall was improved by 5.65%, which is a significant improvement. The U-Net-MS outperformed all other one-stage and two-stage CNN networks for detecting sheep by giving the highest recall, and to the best of the author's knowledge, the presented results are the first in this research area to report such values.

Paddock videos were recorded from higher altitudes so that they would be covered more effectively. Compared to the sheep detection at 120 m, U-Net-MS performed slightly better at 80 m altitude due to the contours of the sheep being easier to distinguish from this height.

Moreover, the performance of the U-Net-MS was better on the images taken under

cloudy conditions at both altitudes. Bright sunny weather causes the sheep boundary to blend with the background as the grass reflects sunlight and makes the sheep detection a tricky task. It is not the case in cloudy weather, where the boundary of the sheep is very clear, appearing as a whitish blob in front of the grassy background and can be detected more confidently by the network. There are test images from both sunny and cloudy weather in Tables 4.2, 4.3 and 4.4. These results show that the overall performance of the networks is acceptable. Based on the successful performance of the U-Net-MS network, it may be used to develop a UAV-based application for helping farmers detect and count livestock. As this PhD was the first step to develop such a thing, there are still a few limitations that can be reduced in the future.

The time involved in testing each frame is one of the limitations associated with the underlying real-time application. On the NVIDIA GeForce RTX 2080, the testing time of the U-Net is one second per frame, and the MS algorithm takes two seconds to compute the centroids of all sub-images in that frame. As long as GPU systems continue to improve, this issue will be somewhat overcome in the future. Inference time of 3 seconds per frame is too high for real-time applications and must be reduced. Instead of making it an online application, another option is to make this a system that recognizes livestock offline.

It becomes increasingly difficult to count a few sheep when they are increasingly close together as a group. Both the labeller and CNN networks experience the same limitation in this case. Fortunately, sheep tend to spread themselves out rather than congregate in one place, counting is easier when they are not disturbed.

### 7.1.2 Object Tracking

The final system presented is a tracking-by-detection system. As there were 100s of sheep per frame in the recorded aerial videos, a simple algorithm was needed which

will put a less overhead delay in the output of the overall system. That led to the use of the Kalman filter and Hungarian algorithm. However, both algorithms need some tuning in order to perform well for tracking a large number of tiny objects between frame. Both algorithms were tuned for 80a and 80b, and the finalized parameters were used for tracking sheep in 120a to 120d videos. To verify that the process is valid, we tested the whole process for 120 m videos too.

The main points observed after this extensive research of tuning this tracking system are:

- The values of process and observation noise covariance are highly dependent on the video quality and the object's size. Small values should be selected in the case of smooth videos and small objects. This factor was supported by the results reported in Table 5.9 where the same settings that worked best at 80 m gave good results at 120 m too.

- The track-to-detection associating system should be strict enough to avoid the unnecessary creation of new track IDs. ID switches may occur but as all livestock look similar, such switches can be ignored in this case. A farmer is more interested in the final sheep count and will not be bothered if a sheep has the same ID till the end of its existence in the video.

- As letting the unassigned tracks remain longer in the system does not degrade the system that much, the system can be kept relaxed in terms of deleting such tracks.

- Sheep tracking and counting at the mentioned altitudes is helpful to limit the UAV flight times to a few seconds and it will not disturb the farm animals. This will make it easier for farm managers to get paddock videos at the desired altitude and use the respective trained model for sheep detection and tracking.

- The main task of this research was to get livestock counts in the respective

paddocks and it worked extremely well in most of the videos.

However, this system works better in cloudy conditions as the object detector, the U-Net model, performs better in this weather. Due to the reflection from the grass in sunny weather, the system fails at a few points. Also, it gets difficult to perform ground truth labelling in sunny weather.

Additionally, there are still physical limitations: the proposed UAV system may not detect sheep under bushes or trees in a paddock with many trees, because such obstructions make it more difficult to see them. It was also noted that sometimes the sun-bleached wooden logs and old tree stumps are mistakenly detected as sheep by the CNN network. Another constraint is the weather: the UAVs cannot be used in rainy or very windy conditions and limit the use of a UAV in other days. However, farmers would also avoid manual sheep counting in rain as this would only increase their effort and make the result more prone to errors.

### 7.1.3 Fence Detection

The literature found relevant to line or edge detection has applications for research problems like power line and road lane detection. The algorithms proposed in those research articles were used for those kind of images which have objects with visible lines or edges. In the initial stages of our research, we tried to use Hough Transform for fence line detection but due to the higher scene complexity and vague fence lines at higher altitudes, we were unable to proceed further. However, as the results of the U-Net model for sheep detection were promising, the same U-Net model was trained for fence posts detection at both altitudes.

A unique combination of the U-Net model, DBSCAN and linear regression was used to get the fence lines in respective frames of the video. Fence lines were identified correctly at 80 m but unfortunately, the results were not encouraging for 120 m. As

discussed in Section 6.2.2, fence posts were hardly visible in most of the images. The labeller often had to guess the location of particular fence posts. Sometimes an impression of a fence line is clear when zoomed out, but the individual fence posts were not clear when zoomed in.

For 80 m videos, fence lines were detected for each frame and then only those sheep detections were kept in the system that were within the fence lines of the respective frame. However, for 120a to 120d, this task was done manually, i.e. the objects detected outside the fences were not kept in the system.

The proposed method was tested only on paddocks with rectangular fencing. Depending on the geographical area the paddocks can be fenced in some arbitrary way. The last step, linear regression, could be replaced with a polynomial regression of a higher degree to define those fence lines that are not straight.

## 7.2 Future Research Scope

The work presented in this thesis has set a basis for a real-time and highly accurate livestock detection, tracking and counting system. To increase detection and tracking accuracy, high-tech cameras can be used to capture images with higher resolution. The contours of the sheep as well as other objects will be more clear in high-resolution images and it may increase the detection accuracy. Similarly, the problem of vague fence lines at higher altitudes could also be resolved using higher resolution images and videos. However, this is likely to increase the inference and processing time per image which can cause a delay in the overall system. There is a need for a careful trade-off between the accuracy of this counting system and the time taken by the whole process.

Also, thermal imaging may help in the cases where sheep hide under the trees and cannot be seen through other cameras. Thermal imaging cameras usually have low resolution as compared to other cameras and a sheep may appear on a few pixels only.

If the sheep is not sheared and has a wool layer on it, then the thermal camera may only show the heat signal from the head of the sheep. It is expected that it can result in only one or two pixels per sheep on the thermal image. To overcome this issue, results from both normal images and thermal images can be combined to increase detection and counting accuracy. For this purpose, images from both cameras should be taken at the same time and the drone should be custom-designed to hold both cameras together. This will increase the cost of the system but can improve the results too.

One of the limitations of the U-Net model is an inaccurate detection when sheep are in a group of three or more. In most of such cases, the U-Net model do not detect all of them. This needs an improvement, maybe either using the pixel density per sheep or training the network in such a way that it can distinguish between one sheep and multiple sheep.

Another approach to counting sheep within a paddock fence is to use an orthomosaic RGB map bounded by previously geo-referenced fence lines to constrain the sheep count per paddock. An orthomosaic map needs to be created using orthorectification for each farm and then for each paddock on that farm. This requires some more expertise in map reading and orthorectification. The farm manager also needs to define the path for drone flight each time for the respective paddock. This simply means that we are expecting a farm manager to be professional in handling and managing the drone flights, which may not be true. This is one of the limitations of using this method. The other limitation is that the paddock images should be stitched to use for sheep detection on orthomosaic map but as the sheep movements cannot be controlled image stitching may not be an appropriate approach for moving objects.

This research can be extended to those paddocks which have uneven terrain and have many bushes and trees on premises. This will introduce challenges like variation in the sheep sizes at different altitudes and difficulty in flying the UAV for efficient coverage of paddocks. However, such type of data which will have variation in object

sizes may train the network even better, possibly improving the overall performance. A lot of work is likely to be needed to reach that stage.

A similar detection method can be used for other livestock like cattle, buffaloes, horses, deer, llamas etc. The object detector can be trained for the respective animal and other parts of the system can be kept the same. Like sheep farms, cattle farms also cover a major part of the pastoral farms in New Zealand and can be targeted first. However, unlike sheep, cattle have different colors and need a bigger dataset for training the neural network efficiently. Another very prominent factor is the shadow during daytime and especially for black cattle, it can be a bit tricky to differentiate between the actual object and the shadow from 80m or above. The trained network might count the animal and its shadow as two different objects and the network should be trained very carefully.

As our proposed system can be used for wildlife detection and counting, the U-Net model can be trained to detect multiple objects. Some wildlife data is collected from fixed cameras, while some others are collected using drones. The proposed system can be used for both kinds of data. Our current method is trained to detect an object of one class and will need a thorough understanding to modify it properly for multi-class detection.

Currently, paddock videos can be recorded and sent to a low-powered CPU that can be used to process them and give a sheep count minutes later. The use of a small GPU machine, Jetson Nano, was proposed in [169] to upgrade this offline tracking-by-detection system to an online real-time application. However, most of the current mobile phones and tablets come with a built-in GPU machine. The proposed system can be implemented to run on the device that is connected to the drone at the time of recording videos. Instead of transferring it to a low-powered CPU machine, the same videos can process using the device (tablet or mobile phone) GPU in much less time. It will decrease the inference time and will need some expertise in Android or iOS

application development techniques. As more and more image and video processing applications are emerging on edge devices, the proposed system can be implemented too and will be a good step forward.

Summarizing the above points, this research can be improved further in different ways and has multiple applications. Researchers with an interest in similar projects can use this system as a base and explore other improvements to meet future challenges.

# References


[1] S. Morris and P. Kenyon, "Intensive sheep and beef production from pasture—a New Zealand perspective of concerns, opportunities and challenges," *Meat science*, vol. 98, no. 3, pp. 330–335, 2014.

[2] "Beef + lamb New Zealand, compendium of New Zealand farm facts 2020 (44 ed.)," Accessed: 07 May, 2021. [Online]. Available: https://beeflambnz.com/sites/default/files/data/files/Compendium-2020.pdf

[3] "Beef + lamb New Zealand, new season outlook 2020-21," Accessed: 19 May, 2021. [Online]. Available: https://beeflambnz.com/sites/default/files/data/files/B%2BLNZ%20New%20Season%20Outlook%202020-21.pdf

[4] J. McKinlay, C. Southwell, and R. Trebilco, "Integrating count effort by seasonally correcting animal population estimates (ICESCAPE): A method for estimating abundance and its uncertainty from count data using Adélie penguins as a case study," *CCAMLR Science*, vol. 17, pp. 213–227, 2010.

[5] D. Beck, "Ram and sheep slaughtered and stolen on Rotorua farm," *NZ Herald*, 23 May, 2020. [Online]. Available: https://www.nzherald.co.nz/rotorua-daily-post/news/ram-and-sheep-slaughtered-and-stolen-on-rotorua-farm/7FCDMUOMD7T4FIMTLZJ3CAT3S4/

[6] M. Solignac, "'Heartbreaking' theft of prime lambs leaves farmer reeling," *Stuff*, 07 Dec, 2020. [Online]. Available: https://www.stuff.co.nz/marlborough-express/marlborough-top-stories/300176708/heartbreaking-theft-of-prime-lambs-leaves-farmer-reelingl

[7] A. Redmore, "Hundreds of sheep stolen in Southland robbery," *Newshub*, 20 May, 2019. [Online]. Available: https://www.newshub.co.nz/home/rural/2019/05/hundreds-of-sheep-stolen-in-southland-robbery.html

[8] M. Wright, "Why is it so hard to catch stock thieves?" *Stuff*, 15 April, 2017. [Online]. Available: https://www.stuff.co.nz/business/farming/91511748/why-is-it-so-hard-to-catch-stock-thieves

[9] R. Davison, "More than 300 sheep rustled from Waimumu farm," *Rural Life*, 21 May, 2019. [Online]. Available: https://www.odt.co.nz/rural-life/rural-life-other/more-300-sheep-rustled-waimumu-farm

[10] "Early season lambs stolen from Takapau farm," *NZ Herald*, 14 Aug, 2018. [Online]. Available: https://www.nzherald.co.nz/hawkes-bay-today/news/early-season-lambs-stolen-from-takapau-farm/XBJYEUSMBIILVDKGZEPAP6PHKA/

[11] J. C. Van Gemert, C. R. Verschoor, P. Mettes, K. Epema, L. P. Koh, and S. Wich, "Nature conservation drones for automatic localization and counting of animals," in *Workshop at the European Conference on Computer Vision*. Springer, 2014, pp. 255–270.

[12] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[13] F. Sarwar, T. Pasang, A. Griffin, and S. U. Rehman, "Survey of livestock counting and tracking methods," *Nusantara Science and Technology Proceedings*, pp. 150–157, 2020.

[14] M. Kassler, "LIVE. 106 automatic counting of sheep," 2001. [Online]. Available: https://www.mla.com.au/contentassets/4312471af05f4f2ebd74484c901dd5df/live.106_final_report.pdf

[15] R. Handcock, D. Swain, G. Bishop-Hurley, K. Patison, T. Wark, P. Valencia, P. Corke, and C. O'Neill, "Monitoring animal behaviour and environmental interactions using wireless sensor networks, GPS collars and satellite remote sensing," *Sensors*, vol. 9, no. 5, pp. 3586–3603, 2009.

[16] K. Anderson and K. J. Gaston, "Lightweight unmanned aerial vehicles will revolutionize spatial ecology," *Frontiers in Ecology and the Environment*, vol. 11, no. 3, pp. 138–146, 2013.

[17] H. Nawaz, H. M. Ali, and S.-u.-R. Massan, "Applications of unmanned aerial vehicles: a review," *3C Tecnología. Glosas de innovación aplicadas a la pyme*, pp. 85–105, 2019.

[18] L. K. Blight, D. F. Bertram, and E. Kroc, "Evaluating UAV-based techniques to census an urban-nesting gull population on Canada's Pacific coast," *Journal of Unmanned Vehicle Systems*, vol. 7, no. 4, pp. 312–324, 2019.

[19] A. H. A. Mutalib, N. Ruppert, S. Akmar, F. F. J. Kamaruszaman, and N. F. N. Rosley, "Feasibility of thermal imaging using unmanned aerial vehicles to detect bornean orangutans," *Journal of Sustainability Science and Management*, vol. 14, no. 5, pp. 182–194, 2019.

[20] A. Singh, M. Pietrasik, G. Natha, N. Ghouaiel, K. Brizel, and N. Ray, "Animal detection in man-made environments," *arXiv preprint arXiv:1910.11443*, 2019.

[21] J. Witczuk, S. Pagacz, A. Zmarz, and M. Cypel, "Exploring the feasibility of unmanned aerial vehicles and thermal imaging for ungulate surveys in forests - preliminary results," *International journal of remote sensing*, vol. 39, no. 15-16, pp. 5504–5521, 2018.

[22] R. Näsi, E. Honkavaara, P. Lyytikäinen-Saarenmaa, M. Blomqvist, P. Litkey, T. Hakala, N. Viljanen, T. Kantola, T. Tanhuanpää, and M. Holopainen, "Using UAV-based photogrammetry and hyperspectral imaging for mapping bark beetle damage at tree-level," *Remote Sensing*, vol. 7, no. 11, pp. 15 467–15 493, 2015.

[23] Z. Sun, D. Wang, and G. Zhong, "Extraction of farmland geographic information using OpenStreetMap data," in *Proceedings of the 7th International Conference on Agro-geoinformatics (Agro-geoinformatics)*. IEEE, 2018, pp. 1–4.

[24] R. Avtar, S. A. Suab, A. P. Yunus, P. Kumar, P. K. Srivastava, M. Ramaiah, and C. A. Juan, "Applications of UAVs in plantation health and area management in Malaysia," in *Unmanned Aerial Vehicle: Applications in Agriculture and Environment*. Springer, Cham, 2020, pp. 85–100.

[25] S. Hogan, M. Kelly, B. Stark, Y. Chen *et al.*, "Unmanned aerial systems for agriculture and natural resources," *California Agriculture*, vol. 71, no. 1, pp. 5–14, 2017.

[26] W. Wu, M. A. Qurishee, J. Owino, I. Fomunung, M. Onyango, and B. Atolagbe, "Coupling deep learning and UAV for infrastructure condition assessment automation," in *Proceedings of the IEEE International Smart Cities Conference (ISC2)*. IEEE, 2018, pp. 1–7.

[27] H. Dang-Ngoc and H. Nguyen-Trung, "Evaluation of forest fire detection model using video captured by UAVs," in *19th International Symposium on Communications and Information Technologies (ISCIT)*. IEEE, 2019, pp. 513–518.

[28] P. Xu, W. Xu, Y. Luo, Z. Zhao *et al.*, "Precise classification of cultivated land based on visible remote sensing image of UAV." *Journal of Agricultural Science and Technology (Beijing)*, vol. 21, no. 6, pp. 79–86, 2019.

[29] T. Yang, Z. Li, F. Zhang, B. Xie, J. Li, and L. Liu, "Panoramic UAV surveillance and recycling system based on structure-free camera array," *IEEE Access*, vol. 7, pp. 25 763–25 778, 2019.

[30] D. Hein, T. Kraft, J. Brauchle, and R. Berger, "Integrated UAV-based real-time mapping for security applications," *ISPRS International Journal of Geo-Information*, vol. 8, no. 5, p. 219, 2019.

[31] M. Di Perna and L. Rodrigues, "A UAV software flight management system using arinc communication protocols," *IEEE Aerospace and Electronic Systems Magazine*, vol. 33, no. 9, pp. 18–28, 2018.

[32] J. Scherer, S. Yahyanejad, S. Hayat, E. Yanmaz, T. Andre, A. Khan, V. Vukadinovic, C. Bettstetter, H. Hellwagner, and B. Rinner, "An autonomous multi-UAV system for search and rescue," in *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*. ACM, 2015, pp. 33–38.

[33] P. Chamoso, W. Raveane, V. Parra, and A. González, "UAVs applied to the counting and monitoring of animals," in *Ambient Intelligence-Software and Applications*. Springer, Cham, 2014, pp. 71–80.

[34] J. G. A. Barbedo and L. V. Koenigkan, "Perspectives on the use of unmanned aerial systems to monitor cattle," *Outlook on Agriculture*, vol. 47, no. 3, pp. 214–222, 2018.

[35] M. Burry, "Barking drones used on farms instead of sheep dogs," *Radio New Zealand*, 7 March, 2019. [Online]. Available: https://www.rnz.co.nz/national/programmes/checkpoint/audio/2018685575/barking-drones-used-on-farms-instead-of-sheep-dogs

[36] ——, "Drones and dog combo prove efficient for farmer," *Radio New Zealand*, 18 May, 2018. [Online]. Available: https://www.rnz.co.nz/news/country/357662/drone-and-dog-combo-prove-efficient-for-farmer

[37] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[38] L. Roberts, "Pattern recognition with an adaptive network," in *Proceedings of the institute of radio engineers*, vol. 48, no. 3, 1960, pp. 398–398.

[39] A. Andreopoulos and J. K. Tsotsos, "50 years of object recognition: Directions forward," *Computer vision and image understanding*, vol. 117, no. 8, pp. 827–891, 2013.

[40] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. IEEE, 1999, pp. 1150–1157.

[41] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[42] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR)*, vol. 1. IEEE, 2001, pp. 511–518.

[43] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 3. IEEE Computer Society, 2003, pp. 1–8.

[44] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[45] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 2169–2178.

[46] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, 2008, pp. 1–8.

[47] X. Wang, T. X. Han, and S. Yan, "An HOG-LBP human detector with partial occlusion handling," in *Proceedings of the IEEE 12th international conference on computer vision*. IEEE, 2009, pp. 32–39.

[48] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proceedings of the European conference on computer vision*. Springer, 2010, pp. 143–156.

[49] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders, "Segmentation as selective search for object recognition," in *Proceedings of the International Conference on Computer Vision*. IEEE, 2011, pp. 1879–1886.

[50] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[51] 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*, 2014, pp. 1717–1724.

[52] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[53] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3642–3649.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[55] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[56] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[57] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[58] 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*, 2016, pp. 779–788.

[59] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proceedings of the European conference on computer vision*. Springer, 2016, pp. 21–37.

[60] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[61] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[62] J. Ribera, D. Güera, Y. Chen, and J. D. Delp, Edward, "Weighted Hausdorff distance: A loss function for object localization," *arXiv preprint arXiv:1806.07564*, 2018.

[63] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6154–6162.

[64] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 734–750.

[65] C. Zhu, Y. He, and M. Savvides, "Feature selective anchor-free module for single-shot object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 840–849.

[66] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6569–6578.

[67] H. Jachmann, "Evaluation of four survey methods for estimating elephant densities," *African Journal of Ecology*, vol. 29, no. 3, pp. 188–195, 1991.

[68] G. Sileshi, "The excess-zero problem in soil animal count data and choice of appropriate models for statistical inference," *Pedobiologia*, vol. 52, no. 1, pp. 1–17, 2008.

[69] F. A. Wichmann, J. Drewes, P. Rosas, and K. R. Gegenfurtner, "Animal detection in natural scenes: critical features revisited," *Journal of Vision*, vol. 10, no. 4, pp. 1–27, 2010.

[70] T. Burghardt and J. Calic, "Real-time face detection and tracking of animals," in *8th Seminar on Neural Network Applications in Electrical Engineering*. IEEE, 2006, pp. 27–32.

[71] M. Parikh, M. Patel, and D. Bhatt, "Animal detection using template matching algorithm," *International Journal of Research in Modern Engineering and Emerging Technology*, vol. 1, no. 3, pp. 26–32, 2013.

[72] J. G. A. Barbedo, L. V. Koenigkan, T. T. Santos, and P. M. Santos, "A study on the detection of cattle in UAV images using deep learning," *Sensors*, vol. 19, no. 24, p. 5436, 2019.

[73] J. G. A. Barbedo, L. V. Koenigkan, and P. M. Santos, "Cattle detection using oblique UAV images," *Drones*, vol. 4, no. 4, p. 75, 2020.

[74] J. T. Mufford, D. J. Hill, N. J. Flood, and J. S. Church, "Use of unmanned aerial vehicles UAVs and photogrammetric image analysis to quantify spatial proximity in beef cattle," *Journal of Unmanned Vehicle Systems*, 2019.

[75] K. Kawamura, J. Lim, Y. Kurokawa, T. Obitsu, M. Yayota, and S. Ogura, "Monitoring spatial heterogeneity of pasture within paddock scale using a small unanned aerial vehicle sUAV," *J. Integrated Field Sci*, vol. 14, pp. 61–66, 2017.

[76] M. Razaak, H. Kerdegari, V. Argyriou, and P. Remagnino, "Multi-scale feature fused single shot detector for small object detection in UAV images," in *Proceedings of the International Conference on Computer Vision Systems*. Springer, Cham, 2019, pp. 778–786.

[77] L. Han, P. Tao, and R. R. Martin, "Livestock detection in aerial images using a fully convolutional network," *Computational Visual Media*, vol. 5, no. 2, pp. 221–228, 2019.

[78] M. Rahnemoonfar, D. Dobbs, M. Yari, and M. J. Starek, "DisCountNet: Discriminating and counting network for real-time counting and localization of sparse objects in high-resolution UAV imagery," *Remote Sensing*, vol. 11, no. 9, p. 1128, 2019.

[79] S. D. Meena and L. Agilandeeswari, "Smart animal detection and counting framework for monitoring livestock in an autonomous unmanned ground vehicle

using restricted supervised learning and image fusion," *Neural Processing Letters*, pp. 1–33, 2021.

[80] Z. Zhang, Z. He, G. Cao, and W. Cao, "Animal detection from highly cluttered natural scenes using spatiotemporal object region proposals and patch verification," *IEEE Transactions on Multimedia*, vol. 18, no. 10, pp. 2079–2092, 2016.

[81] M. A. Tabak, M. S. Norouzzadeh, D. W. Wolfson, S. J. Sweeney, K. C. VerCauteren, N. P. Snow, J. M. Halseth, P. A. Di Salvo, J. S. Lewis, M. D. White *et al.*, "Machine learning to classify animal species in camera trap images: Applications in ecology," *Methods in Ecology and Evolution*, vol. 10, no. 4, pp. 585–590, 2019.

[82] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 951–958.

[83] P. Tassinari, M. Bovo, S. Benni, S. Franzoni, M. Poggi, L. M. E. Mammi, S. Mattoccia, L. Di Stefano, F. Bonora, A. Barbaresi *et al.*, "A computer vision approach based on deep learning for the detection of dairy cows in free stall barn," *Computers and Electronics in Agriculture*, vol. 182, p. 106030, 2021.

[84] U. B. Desai, S. N. Merchant, M. Zaveri, G. Ajishna, M. Purohit, and H. Phanish, "Small object detection and tracking: Algorithm, analysis and application," in *Proceedings of the International Conference on Pattern Recognition and Machine Intelligence*. Springer, 2005, pp. 108–117.

[85] W. Meng, T. Jin, and X. Zhao, "Adaptive method of dim small object detection with heavy clutter," *Applied optics*, vol. 52, no. 10, pp. D64–D74, 2013.

[86] A. Vard, K. Jamshidi, and N. Movahhedinia, "Small object detection in cluttered image using a correlation based active contour model," *Pattern Recognition Letters*, vol. 33, no. 5, pp. 543–553, 2012.

[87] C. Chen, M.-Y. Liu, O. Tuzel, and J. Xiao, "R-CNN for small object detection," in *Proceedings of the Asian conference on computer vision*. Springer, 2016, pp. 214–230.

[88] T. Kong, A. Yao, Y. Chen, and F. Sun, "Hypernet: Towards accurate region proposal generation and joint object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 845–853.

[89] C. Eggert, S. Brehm, A. Winschel, D. Zecha, and R. Lienhart, "A closer look: Small object detection in faster R-CNN," in *Proceedings of the IEEE international conference on multimedia and expo (ICME)*. IEEE, 2017, pp. 421–426.

[90] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan, "Perceptual generative adversarial networks for small object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1222–1230.

[91] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem, "SOD-MTGAN: Small object detection via multi-task generative adversarial network," in *Proceedings of The European Conference on Computer Vision (ECCV)*, September 2018.

[92] W. Li, K. Liu, L. Yan, F. Cheng, Y. Lv, and L. Zhang, "FRD-CNN: Object detection based on small-scale convolutional neural networks and feature reuse," *Scientific reports*, vol. 9, pp. 1–12, 2019.

[93] D. W. Ma, X. J. Wu, and H. Yang, "Efficient small object detection with an improved region proposal networks," in *IOP Conference Series: Materials Science and Engineering*, vol. 533, no. 1. IOP Publishing, 2019.

[94] G. X. Hu, Z. Yang, L. Hu, L. Huang, and J. M. Han, "Small object detection with multiscale features," *International Journal of Digital Multimedia Broadcasting*, vol. 2018, 2018.

[95] J. Noh, W. Bae, W. Lee, J. Seo, and G. Kim, "Better to follow, follow to be better: Towards precise supervision of feature super-resolution for small object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9725–9734.

[96] Y. Ren, C. Zhu, and S. Xiao, "Small object detection in optical remote sensing images via modified Faster R-CNN," *Applied Sciences*, vol. 8, no. 5, p. 813, 2018.

[97] L. Bo, T. Hai, and F. Qiang, "A small object detection method based on local maxima and SSD," in *AOPC 2019: AI in Optics and Photonics*, vol. 11342. International Society for Optics and Photonics, 2019, p. 113420Q.

[98] L. Chao, Y.-d. Ding, and D.-b. Wang, "Dense-SSD for detecting small objects," *DEStech Transactions on Engineering and Technology Research, ECAR*, 2018.

[99] J.-S. Lim, M. Astrid, H.-J. Yoon, and S.-I. Lee, "Small object detection using context and attention," *arXiv preprint arXiv:1912.06319*, 2019.

[100] Z. Du, J. Yin, and J. Yang, "Expanding receptive field YOLO for small object detection," in *Journal of Physics: Conference Series*, vol. 1314, no. 1. IOP Publishing, 2019, p. 012202.

[101] H.-W. Luo, C.-S. Zhang, F.-C. Pan, and X.-M. Ju, "Contextual-YOLOV3: implement better small object detection based deep learning," in *Proceedings of the International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*. IEEE, 2019, pp. 134–141.

[102] N. Merlet and J. Zerubia, "New prospects in line detection by dynamic programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 426–431, 1996.

[103] Y.-S. Lee, H.-S. Koo, and C.-S. Jeong, "A straight line detection using principal component analysis," *Pattern Recognition Letters*, vol. 27, no. 14, pp. 1744–1754, 2006.

[104] D. Guru, B. Shekar, and P. Nagabhushan, "A simple and robust line detection algorithm based on small eigenvalue analysis," *Pattern Recognition Letters*, vol. 25, no. 1, pp. 1–13, 2004.

[105] J. Alarcon, R. Salvador, F. Moreno, P. Cobos, and I. Lopez, "A new real-time hardware architecture for road line tracking using a Particle filter," in *Proceedings of the 32nd Annual Conference on IEEE Industrial Electronics (IECON)*, 2006, pp. 736–741.

[106] H. Kong, J.-Y. Audibert, and J. Ponce, "General road detection from a single image," *IEEE Transactions on Image Processing*, vol. 19, no. 8, pp. 2211–2220, 2010.

[107] D. Fontanelli, M. Cappelletti, and D. Macii, "A RANSAC-based fast road line detection algorithm for high-speed wheeled vehicles," in *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference*, 2011, pp. 1–6.

[108] J.-W. Kim, T.-H. Kim, and K.-H. Jo, "Traffic road line detection based on the vanishing point and contour information," in *SICE Annual Conference*, 2011, pp. 495–498.

[109] Y. Lin and S. Saripalli, "Road detection from aerial imagery," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2012, pp. 3588–3593.

[110] T. Harasthy, J. Turán, and L. Ovseník, "Road line detection based on Optical Correlator," in *36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2013, pp. 298–300.

[111] W. Liu, Z. Zhang, S. Li, and D. Tao, "Road detection by using a generalized Hough transform," *Remote Sensing*, vol. 9, no. 6, p. 590, 2017.

[112] Z. Li, Y. Liu, R. Hayward, J. Zhang, and J. Cai, "Knowledge-based power line detection for UAV surveillance and inspection systems," in *Proceedings of the 23rd International Conference Image and Vision Computing New Zealand*, 2008, pp. 1–6.

[113] Z. Li, Y. Liu, R. Walker, R. Hayward, and J. Zhang, "Towards automatic power line detection for a UAV surveillance system using pulse coupled neural filter and an improved Hough transform," *Machine Vision and Applications*, vol. 21, no. 5, pp. 677–686, 2010.

[114] J. Zhang, L. Liu, B. Wang, X. Chen, Q. Wang, and T. Zheng, "High speed automatic power line detection and tracking for a UAV-based inspection," in *Proceedings of the International Conference on Industrial Control and Electronics Engineering*, 2012, pp. 266–269.

[115] C. Galamhos, J. Matas, and J. Kittler, "Progressive probabilistic Hough transform for line detection," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 1, 1999, pp. 554–560.

[116] N. Aggarwal and W. Karl, "Line detection in images through regularized Hough transform," *IEEE Transactions on Image Processing*, vol. 15, no. 3, pp. 582–591, 2006.

[117] L. A. Fernandes and M. M. Oliveira, "Real-time line detection through an improved Hough transform voting scheme," *Pattern recognition*, vol. 41, no. 1, pp. 299–314, 2008.

[118] I. Iraei and K. Faez, "Object tracking with occlusion handling using mean shift, Kalman filter and edge histogram," in *Proceedings of the 2nd IEEE International Conference on Pattern Recognition and Image Analysis (IPRIA)*. IEEE, 2015, pp. 1–6.

[119] G. Wang, Y. Wang, H. Zhang, R. Gu, and J.-N. Hwang, "Exploit the connectivity: Multi-object tracking with TrackletNet," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 482–490.

[120] J. Kwak, J. H. Park, and Y. Sung, "Emerging ICT UAV applications and services: Design of surveillance UAVs," *International Journal of Communication Systems*, vol. 34, no. 2, p. e4023, 2021.

[121] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Robust tracking-by-detection using a detector confidence particle filter," in *Proceedings of the IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1515–1522.

[122] Y. Zhang, J. Wang, and X. Yang, "Real-time vehicle detection and tracking in video based on Faster R-CNN," in *Journal of Physics: Conference Series*, vol. 887, no. 1, 2017, p. 012068.

[123] X. Li, K. Wang, W. Wang, and Y. Li, "A multiple object tracking method using Kalman filter," in *Proceedings of the IEEE international conference on information and automation*. IEEE, 2010, pp. 1862–1866.

[124] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61–88, 2020.

[125] S. Zhou, M. Ke, J. Qiu, and J. Wang, "A survey of multi-object video tracking algorithms," in *Proceedings of the International Conference on Applications and Techniques in Cyber Security and Intelligence*. Springer, Cham, 2018, pp. 351–369.

[126] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple object tracking: A literature review," *Artificial Intelligence*, p. 103448, 2020.

[127] H. S. Parekh, D. G. Thakore, and U. K. Jaliya, "A survey on object detection and tracking methods," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 2, pp. 2970–2979, 2014.

[128] J. J. Athanesious and P. Suresh, "Systematic survey on object tracking methods in video," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 1, no. 8, pp. 242–247, 2012.

[129] M. S. Grewal and A. P. Andrews, "Applications of Kalman filtering in aerospace 1960 to the present [historical perspectives]," *IEEE Control Systems Magazine*, vol. 30, no. 3, pp. 69–78, 2010.

[130] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[131] H. A. Patel and D. G. Thakore, "Moving object tracking using Kalman filter," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 4, pp. 326–332, 2013.

[132] F. Bu, Y. Cai, and Y. Yang, "Multiple object tracking based on faster-RCNN detector and KCF tracker," Tech. Rep., 2016.

[133] G. Kitagawa, "Non-Gaussian state—space modeling of nonstationary time series," *Journal of the American statistical association*, vol. 82, no. 400, pp. 1032–1041, 1987.

[134] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004.

[135] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *Proceedings of the European conference on computer vision*. Springer, 2004, pp. 28–39.

[136] J. Yang, D. Schonfeld, and M. Mohamed, "Robust video stabilization based on particle filter tracking of projected camera motion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 7, pp. 945–954, 2009.

[137] M. A. Zulkifley and B. Moran, "Robust hierarchical multiple hypothesis tracker for multiple-object tracking," *Expert Systems with Applications*, vol. 39, no. 16, pp. 12 319–12 331, 2012.

[138] D. H. Santosh and P. G. K. Mohan, "Multiple objects tracking using extended Kalman filter, GMM and mean shift algorithm - a comparative study," in *Proceedings of the IEEE International Conference on Advanced Communications, Control and Computing Technologies*, 2014, pp. 1484–1488.

[139] H. Yu, L. Qin, Q. Huang, and H. Yao, "Online multiple object tracking via exchanging object context," *Neurocomputing*, vol. 292, pp. 28–37, 2018.

[140] S. Ren, Y. Zhou, and L. He, "Multi-object tracking with pre-classified detection," in *Proceedings of the International Conference on Robot Intelligence Technology and Applications*. Springer, Cham, 2017, pp. 503–513.

[141] M. Bertalmio, G. Sapiro, and G. Randall, "Morphing active contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 7, pp. 733–737, 2000.

[142] M. A. A. Dewan, E. Granger, G.-L. Marcialis, R. Sabourin, and F. Roli, "Adaptive appearance model tracking for still-to-video face recognition," *Pattern recognition*, vol. 49, pp. 129–151, 2016.

[143] W.-B. Horng, C.-Y. Chen, Y. Chang, and C.-H. Fan, "Driver fatigue detection based on eye tracking and dynamic template matching," in *Proceedings of the IEEE International Conference on Networking, Sensing and Control, 2004*, vol. 1. IEEE, 2004, pp. 7–12.

[144] D. Exner, E. Bruns, D. Kurz, A. Grundhöfer, and O. Bimber, "Fast and robust CAMShift tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*. IEEE, 2010, pp. 9–16.

[145] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, "Struck: Structured output tracking with kernels," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2096–2109, 2015.

[146] J. Athanesious and P. Suresh, "Implementation and comparison of kernel and silhouette based object tracking," *International Journal of Advanced Research in Computer Engineering & Technology*, pp. 1298–1303, 2013.

[147] C. I. Patel and R. Patel, "Contour based object tracking," *International Journal of Computer and Electrical Engineering*, vol. 4, no. 4, pp. 525–528, 2012.

[148] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3119–3127.

[149] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3038–3046.

[150] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang *et al.*, "T-CNN: Tubelets with convolutional neural networks for object detection from videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2896–2907, 2017.

[151] Z. Wang, L. Zheng, Y. Liu, and S. Wang, "Towards real-time multi-object tracking," *arXiv preprint arXiv:1909.12605*, 2019.

[152] K. Yoon, D. Y. Kim, Y.-C. Yoon, and M. Jeon, "Data association for multi-object tracking via deep neural networks," *Sensors*, vol. 19, no. 3, pp. 559–673, 2019.

[153] K. Fang, Y. Xiang, X. Li, and S. Savarese, "Recurrent autoregressive networks for online multi-object tracking," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 466–475.

[154] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 300–311.

[155] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[156] L. Chen, H. Ai, C. Shang, Z. Zhuang, and B. Bai, "Online multi-object tracking with convolutional neural networks," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 645–649.

[157] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online multi-object tracking using CNN-based single object tracker with spatial-temporal attention mechanism," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4836–4845.

[158] S. Kapania, D. Saini, S. Goyal, N. Thakur, R. Jain, and P. Nagrath, "Multi object tracking with UAVs using Deep SORT and YOLOv3 RetinaNet detection framework," in *Proceedings of the 1st ACM Workshop on Autonomous and Intelligent Mobile Systems*, 2020, pp. 1–6.

[159] Y. Xu, X. Zhou, S. Chen, and F. Li, "Deep learning for multiple object tracking: a survey," *IET Computer Vision*, vol. 13, no. 4, pp. 355–368, 2019.

[160] S. Tang, M. Andriluka, B. Andres, and B. Schiele, "Multiple people tracking by lifted multicut and person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3539–3548.

[161] Z. Kaixuan and H. Dongjian, "Target detection method for moving cows based on background subtraction," *International Journal of Agricultural and Biological Engineering*, vol. 8, no. 1, pp. 42–49, 2015.

[162] B. Liao, J. Hu, and R. O. Gilmore, "Optical flow estimation combining with illumination adjustment and edge refinement in livestock UAV videos," *Computers and Electronics in Agriculture*, vol. 180, p. 105910, 2021.

[163] W. Andrew, C. Greatwood, and T. Burghardt, "Visual localisation and individual identification of holstein friesian cattle via deep learning," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2850–2859.

[164] N. Al-Thani, A. Albuainain, F. Alnaimi, and N. Zorba, "Drones for sheep livestock monitoring," in *Proceedings of the IEEE 20th Mediterranean Electrotechnical Conference (MELECON)*. IEEE, 2020, pp. 672–676.

[165] S. Longmore, R. Collins, S. Pfeifer, S. Fox, M. Mulero-Pázmány, F. Bezombes, A. Goodwin, M. De Juan Ovelar, J. Knapen, and S. Wich, "Adapting astronomical source detection software to help detect animals in thermal images obtained by unmanned aerial systems," *International Journal of Remote Sensing*, vol. 38, no. 8-10, pp. 2623–2638, 2017.

[166] Jong-Min Jeong, Tae-Sung Yoon, and Jin-Bae Park, "Kalman filter based multiple objects detection-tracking algorithm robust to occlusion," in *Proceedings of the SICE Annual Conference (SICE)*, 2014, pp. 941–946.

[167] S. Vasuhi, M. Vijayakumar, and V. Vaidehi, "Real time multiple human tracking using Kalman filter," in *Proceedings of the 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, 2015, pp. 1–6.

[168] F. Sarwar, A. Griffin, P. Periasamy, K. Portas, and J. Law, "Detecting and counting sheep with a convolutional neural network," in *Proceedings of the 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2018, pp. 1–6.

[169] F. Sarwar, A. Griffin, S. U. Rehman, and T. Pasang, "Towards detection of sheep onboard a UAV," *arXiv preprint arXiv:2004.02758*, 2020.

[170] ——, "Detecting sheep in UAV images," *Computers and Electronics in Agriculture*, vol. 187, p. 106219, 2021.

[171] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Proceedings of the International conference on artificial neural networks*. Springer, 2010, pp. 92–101.

[172] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[173] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[174] 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*, 2016, pp. 770–778.

[175] J. Ribera, D. Güera, Y. Chen, and E. J. Delp, "Locating objects without bounding boxes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[176] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[177] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[178] F. Sarwar, A. Griffin, and T. Pasang, "Tracking livestock using a fully connected network and Kalman filter," *Geometry and Vision. ISGV 2021. Communications in Computer and Information Science*, vol. 1386, pp. 247–261, 2021.

[179] G. Welch, G. Bishop *et al.*, "An introduction to the Kalman filter," 1995.

[180] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[181] K. Bernardin, A. Elbs, and R. Stiefelhagen, "Multiple object tracking performance metrics and evaluation in a smart room environment," in *6th IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*, vol. 90, 2006, pp. 91–99.

[182] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the CLEAR MOT metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, pp. 1–10, 2008.

[183] A. Milan, K. Schindler, and S. Roth, "Challenges of ground truth evaluation of multi-target tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 735–742.

[184] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[185] A. Griffin, A. Alexandridis, D. Pavlidi, Y. Mastorakis, and A. Mouchtaris, "Localizing multiple audio sources in a wireless acoustic sensor network," *Signal Processing*, vol. 107, pp. 54–67, 2015.

[186] H. W. Sorenson, *Kalman filtering: theory and application*. IEEE, 1985.

[187] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of the Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. IEEE, 2000, pp. 153–158.

[188] G. Jocher, A. Stoken, A. Chaurasia, J. Borovec, NanoCode012, TaoXie, Y. Kwon, K. Michael, L. Changyu, J. Fang, A. V, Laughing, tkianai, yxNONG, P. Skalski, A. Hogan, J. Nadar, imyhxy, L. Mammana, AlexWang1900, C. Fati, D. Montes, J. Hajek, L. Diaconu, M. T. Minh, Marc, albinxavi, fatih, oleg, and wanghaoyang0106, "ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support," Oct. 2021.

[189] M. Broström, "Real-time multi-object tracker using YOLOv5 and deep sort," https://github.com/mikel-brostrom/Yolov5_DeepSort_Pytorch, 2020.

[190] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

# Appendix A

This research was started with a very small dataset and we investigated different-sized networks to observe the impact of network architecture on performance metrics. Performance metrics of networks with different combinations of CONV and FC layers, kernel sizes and network topology are discussed here and were presented in the first conference paper [168] as follows:

**C1:** F. Sarwar, A. Griffin, P. Periasamy, K. Portas, and J. Law, "Detecting and counting sheep with a convolutional neural network," *in proceeding of 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2018, pp. 1-6.

## A.1 Smaller Dataset

For training and testing, the initial dataset designed for this research is shown in Table A.1, where there a total of 4 images used for training and then 8 for testing [168]. In both training and testing, half images were captured in sunny conditions and half were in cloudy weather.

Different images were used for the design of training and then testing datasets. To ensure that our algorithm could be applied to different species of animals in the future, we used RGB images. From each training image, multiple sub-images were extracted.

Table A.1: Details of the smaller dataset, where all images are ($2048 \times 1080 \times 3$) pixels in size.

| **Dataset** | **Training** | | **Testing** | |
|---|---|---|---|---|
| | No. of Images | Total Sheep | No. of Images | Total Sheep |
| Sunny | 2 | 267 | 4 | 582 |
| Overcast | 2 | 319 | 4 | 520 |
| Mixed | 4 | 586 | 8 | 1102 |

Each RGB sub-image was of 250×250 pixels and was flipped upside down and left to right to increase the size of the training set. It can also be said that the sub-images were rotated at 90, 180, and 270 to create a similar effect. A dataset of 400 images was created from four full-sized images and among them, 82 lacked any object of interest and only had backgrounds. Then, the rest were used to create three different training datasets: only sunny images, only overcast images, and both combined. The details of these three datasets in terms of the number of images and total sheep in the respective set are shown in Table A.2. A few images had some areas of tree shades in them and were included in the mixed dataset only. This fact makes a bit different number of total images in the mixed dataset and it is not just the sum of the other datasets.

Each training image was labeled manually and bounding boxes were drawn around each sheep. Each ground truth bounding box has same format: $(x, y, w, h)$, where $x$ and $y$ represents the starting coordinates and $w$ and $h$ are the width and height of respective bounding box. Because the number of sheep per training image ranges between 1 to 18, the process was quite time-consuming. Our full dataset consists

Table A.2: Details of augmented training dataset, where all images are (250×250×3) pixels in size.

| **Dataset** | **No. of Images** | **Total Sheep** |
|---|---|---|
| Sunny | 146 | 884 |
| Overcast | 169 | 811 |
| Mixed | 318 | 1706 |

of 1706 objects or ROIs (sheep). Labels weren't applied to objects which weren't completely enclosed in an image or were at the edges. When we retained such objects, the networks gave more false positives detections and degraded the results. We observed this effect experimentally, removed labels from the sheep on edges, and then retrained the networks.

## A.2 Different Networks

For examining how different combinations of convolutional and FC layers affect the result, we designed different architectures. The first network created for this purpose is shown in Figure A.1. Other networks were then designed by systematically eliminating convolutional layers (from right to left) from the network shown. Each designed network was trained using the same algorithm and the results did not vary that much. The precision was affected by 5%, and the recall rate was affected by 6%, when the network was decreased to only one convolutional layer, followed by one FC layer and softmax layer.

More networks were also designed by changing some other factors [168] and the details of all of them are as follows:

1. 96 filters with different kernel sizes of 5×5, 11×11, 21×21, 25×25 and 29×29 in a network of 1 convolution layer followed by 1 FC and softmax layer.
2. R-CNN network with 2, 3, 4, 5 and 6 network layers using a 6-layer network structure as shown in Figure A.1.
3. A 6-layer R-CNN network using different kernel sizes in each convolutional layer.


Figure A.1: Main architecture of R-CNN network.

## A.3 Results

The accuracy of the results is also affected if the same network is trained by a different training dataset. However, despite having half as many images as the full dataset, the training set with only sunny images was able to train the network almost equally well. Results were relatively poor when only overcast images were used during training. Details of the test images are shown in Table A.1 and all networks were tested on them.

Figure A.2 shows results computed with all networks that have been trained on the full dataset. There are three curves of precision and three curves of recall for three different datasets in each figure. The results obtained from the two-layered R-CNN network (1 conv + 1 FC + softmax) were used to generate the top graph. Kernel sizes for the convolutional layer are indicated on the x-axis of this graph. On the bottom left, there are precision and recall curves for 2-, 3-, 4-, 5- and 6-layer R-CNN networks. Using a 6-layer R-CNN network, the results can be seen at the bottom right in different kernel sizes across each layer. There were fewer illumination variations in cloudy test images which resulted in the highest recall. On all types of test datasets, the highest recall was observed using a two-layer convolutional network with 96 filters of 21×21 dimension.

From these graphs, it was observed that the networks with a larger kernel size of around 21×21 and 25×25 gave good results as compared to the other ones. As shown in the bottom right graph with the same figure, different filter combinations in larger networks didn't produce significant variations. And the variation in filter size was done

only for the smallest network to observe the performance. Smaller filters are more adept at capturing tiny details in objects, and sheep look like an oval-shaped white blob at the mentioned altitude. It can be concluded from these curves that smaller networks can be used for detecting small objects, and it can also save time and memory.

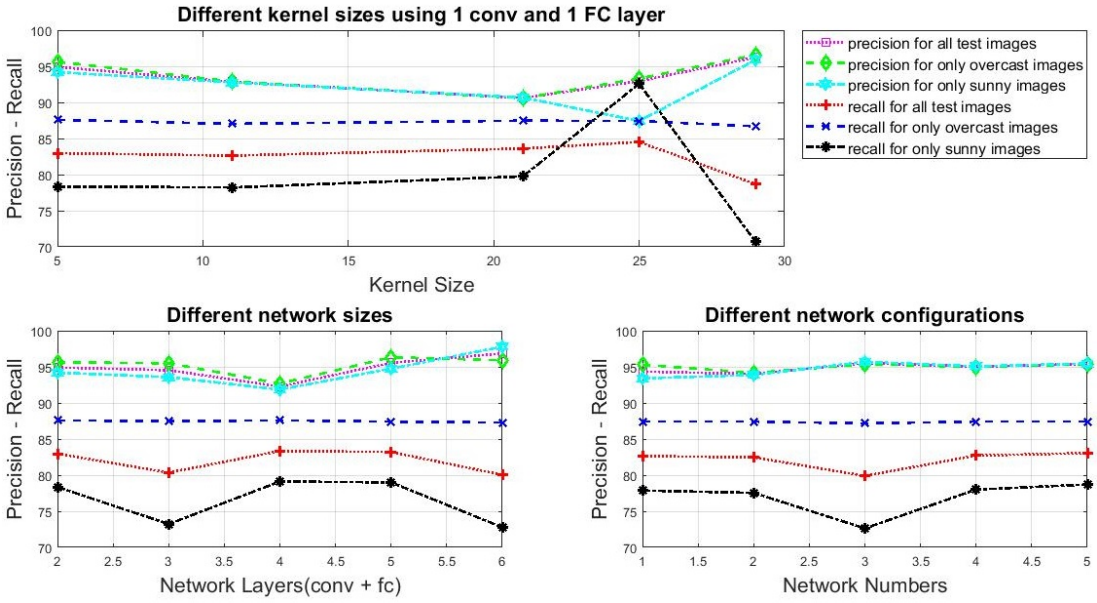



Figure A.2: Precision and Recall curves using different settings.

# Appendix B

The results of different CNN networks, which were trained, validated and tested on the SheepSet80 are presented here. These results were presented in an invited talk of the British Machine Vision Conference, 2020 and was later published as the following manuscript:

**C2:** F. Sarwar, A. Griffin, S. U. Rehman, and T. Pasang, "Towards detection of sheep onboard a UAV," *arXiv preprint arXiv:2004.02758*, 2020. *(presented in the invited talk at British Machine Vision Conference (BMVC) workshop, 2019).*

## B.1 Results on Test Set of SheepSet80

We evaluated the performance of five different networks: GoogLeNet, AlexNet, Network-I, Network-II and U-Net-GMM on the test set of SheepSet80. The first two networks were already trained on ImageNet and were fine-tuned on our dataset after modifying the last two layers. The last three networks i.e. Network-I, Network-II and U-Net-GMM were trained only on SheepSet80.

Precision, recall and $F_1$-score were used as main parameters for performance evaluation, and they are defined in Section 4.4.4. These performance metrics for the mentioned trained networks are shown in Table B.1. Among the competing networks, U-Net-GMM provides a superior performance on all metrics, making it a clear choice

Table B.1: Performance metrics of the networks.

| **Network** | **Mean Precision** | **Mean Recall** | **Mean $F_1$-score** |
|---|---|---|---|
| GoogLeNet | 80.79% | 81.67% | 81.23% |
| AlexNet | 78.74% | 81.37% | 80.03% |
| Network-I | 88.77% | 80.45% | 84.38% |
| Network-II | 91.21% | 81.91% | 86.31% |
| U-Net-GMM | **96.20%** | **90.14%** | **93.07%** |

for our task. Although Network-II is the second-best method in terms of overall accuracy, it is substantially slower than the U-Net-GMM [169].

# Appendix C

The results of sheep detection and tracking are presented from six different videos. Two of these videos were recorded at an altitude of 80 m in the same paddock in cloudy and sunny weather and were labelled as 80a and 80b respectively. The remaining four videos—120a to 120d—were from an altitude of 120 m in cloudy and sunny weather and were all from different paddocks. For performance comparison, different state-of-art object detectors like VGG19 [173], ResNet50 [174], the original U-Net [176] (referred to as U-Net-GMM in [170]), and Network-II [170] were also used to detect sheep in every frame of the recorded videos. The performance of the overall tracking system was observed using the above-mentioned object detectors and the Kalman filter as well as the extended Kalman filter [186], unscented Kalman filter [187] and Particle filter [133]. YOLOv5 [188] with Deep Sort [189] was also used for sheep detection and tracking in the UAV videos. This extensive research for comparing the performance of different state-of-art methods support our proposed object tracking system for tracking small or tiny objects.

The discussed object detectors and object trackers were first evaluated on both videos that were recorded at 80 m altitude. Only those tracking systems, which gave the best results for this task at 80 m, were then used for livestock tracking at 120 m.

## C.1 Livestock Detection and Tracking at 80 m

In the first stage, sheep were detected in all frames of 80a and 80b separately and Table C.1 shows the respective results. The U-Net-MS model gave the best results in all aspects for both videos. YOLOv5 and U-Net-GMM showed similar performance for both videos. The main difference between U-Net-MS and U-Net-GMM is the slight difference in the network architecture and the usage of MS clustering algorithm in the U-Net-MS but GMM in the U-Net-GMM. The U-Net-GMM model mostly merges the closely standing sheep and considers them as one sheep only, hence degrading the performance to some extent. However, YOLO has come a long way from poorly detecting small objects [58] to performing very well for detecting tiny objects in images [190]. Network-II outperformed the remaining three object detectors for 80a, however, ResNet50 performed better for 80b. Simplifying the results, the U-Net-MS model performed exceptionally well for both sunny and cloudy video recorded at 80 m and will be used for livestock detection at 120 m too. Also, the parameters for the Kalman filter and Hungarian algorithm will be first tuned using the U-Net-MS as the object detector. Then the results for all combinations will be presented and discussed in

Table C.1: Sheep detection results on videos recorded at 80 m respectively.

| **Video** | **Object Detector** | **Metrics** | | |
|---|---|---|---|---|
| | | PPF ↑ | RPF ↑ | FAF ↓ |
| 80a | U-Net-MS [170] | **99.97%** | **99.51%** | **0.03%** |
| | U-Net-GMM [62] | 95.17% | 88.28% | 4.46% |
| | Network-II [170] | 99.45% | 88.01% | 0.51% |
| | ResNet50 [174] | 91.45% | 76.69% | 7.21% |
| | VGG19 [173] | 99.39% | 87.75% | 0.55% |
| | YOLOv5 [188] | 95.81% | 89.91% | 4.75% |
| 80b | U-Net-MS [170] | **99.84%** | **96.44%** | **0.15%** |
| | U-Net-GMM [62] | 98.06% | 90.17% | 1.79% |
| | Network-II [170] | 98.61% | 77.79% | 1.11% |
| | ResNet50 [174] | 94.79% | 87.96% | 4.92% |
| | VGG19 [173] | 97.97% | 75.70% | 1.58% |
| | YOLOv5 [188] | 96.84% | 92.23% | 3.76% |

detail.

To verify that our proposed combination works well in most of the cases, other tracking algorithms were used and results are compared, as shown in Tables C.2 and C.3 for both videos. We have abbreviated Kalman filter as KF at a few places to save the space in the mentioned tables. It was observed that the overall performance of the system was consistently good, which justifies the finding from existing literature that machine learning algorithms perform well in MOT. Deep SORT is an advanced version of SORT and incorporates the object's appearance information through a pre-trained association metric. YOLOv5 was initially trained on our dataset to generate association metrics for our objects. It was observed that in terms of tracking performance, YOLOv5 with Deep SORT again performed equivalent to U-Net-GMM. The Particle filter was computationally expensive and it was almost 10 times slower than the other algorithms. The performance of different versions of the Kalman filter was pretty much similar with those object detectors that have a good performance.

## C.2 Livestock tracking at 120 m

Table C.4 shows the sheep tracking results at 120 m using the U-Net-MS as object detector and other machine learning object trackers. It was observed that at this altitude the values of SC, MOTPE and MOTA show high similarity for respective videos for all combinations. This supports our proposed system and the results are quite promising.

Table C.2: Performance metrics comparison of object tracking algorithms on 80a video, where ground truth centroids are 15,045 and actual sheep count is 352.

| **Detector** | **Tracker** | **Metrics** | | | | |
|---|---|---|---|---|---|---|
| | | **SC** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| U-Net-MS | Kalman Filter | 352 | 15,009 | 0.13% | 0.03 | 99.63% |
| | Extended KF | 352 | 15,004 | 0.19% | 0.03 | 99.53% |
| | Unscented KF | 352 | 15,004 | 0.19% | 0.03 | 99.50% |
| | Particle Filter | 354 | 14,997 | 2.07% | 0.03 | 97.61% |
| U-Net-GMM | Kalman Filter | 389 | 13,717 | 8.89% | 0.61 | 82.28% |
| | Extended KF | 370 | 14,035 | 14.01% | 0.60 | 79.28% |
| | Unscented KF | 370 | 14,035 | 14.01% | 0.60 | 79.28% |
| | Particle Filter | 478 | 13,842 | 50.43% | 0.64 | 41.58% |
| Network-II | Kalman Filter | 354 | 14,393 | 1.62% | 4.07 | 94.04% |
| | Extended KF | 355 | 14,526 | 1.86% | 4.00 | 94.69% |
| | Unscented KF | 355 | 14,526 | 1.86% | 4.00 | 94.69% |
| | Particle Filter | 470 | 14,430 | 37.05% | 4.12 | 58.86% |
| ResNet50 | Kalman Filter | 432 | 14,253 | 13.27% | 4.89 | 81.47% |
| | Extended KF | 405 | 14,513 | 21.89% | 4.84 | 74.57% |
| | Unscented KF | 405 | 14,512 | 21.90% | 4.84 | 74.56% |
| | Particle Filter | 497 | 14,404 | 58.75% | 4.95 | 36.99% |
| VGG19 | Kalman Filter | 354 | 14,349 | 2.5% | 5.68 | 92.84% |
| | Extended KF | 352 | 14,496 | 3.07% | 5.60 | 93.28% |
| | Unscented KF | 352 | 14,496 | 3.07% | 5.60 | 93.28% |
| | Particle Filter | 460 | 14,336 | 36.09% | 5.72 | 59.20% |
| YOLOv5 | Deep SORT | 387 | 13,347 | 5.11% | 2.75 | 84.82% |

Table C.3: Performance metrics comparison of object tracking algorithms on 80b video, where ground truth centroids are 19,798 and actual sheep count is 352.

| **Detector** | **Tracker** | **Metrics** | | | | |
|---|---|---|---|---|---|---|
| | | **SC** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| U-Net-MS | Kalman Filter | 351 | 19199 | 0.47% | 0.07 | 96.50% |
| | Extended KF | 350 | 19188 | 0.74% | 0.07 | 96.18% |
| | Unscented KF | 350 | 19187 | 0.76% | 0.07 | 96.15% |
| | Particle Filter | 346 | 19153 | 0.80% | 0.07 | 95.94% |
| U-Net-GMM | Kalman Filter | 365 | 18266 | 4.52% | 0.58 | 87.74% |
| | Extended KF | 354 | 18419 | 6.40% | 0.59 | 86.63% |
| | Unscented KF | 354 | 18417 | 6.42% | 0.59 | 86.61% |
| | Particle Filter | 360 | 18479 | 9.05% | 0.61 | 84.29% |
| Network-II | Kalman Filter | 363 | 17711 | 6.06% | 4.09 | 83.40% |
| | Extended KF | 342 | 18512 | 6.09% | 4.07 | 87.41% |
| | Unscented KF | 342 | 18512 | 6.09% | 4.07 | 87.41% |
| | Particle Filter | 349 | 18531 | 8.36% | 4.09 | 85.24% |
| ResNet50 | Kalman Filter | 479 | 17048 | 19.89% | 4.91 | 66.22% |
| | Extended KF | 391 | 18334 | 31.76% | 4.96 | 60.85% |
| | Unscented KF | 391 | 18333 | 31.76% | 4.96 | 60.84% |
| | Particle Filter | 410 | 18437 | 39.90% | 4.95 | 53.22% |
| VGG19 | Kalman Filter | 379 | 17205 | 8.34% | 5.00 | 78.56% |
| | Extended KF | 345 | 18296 | 9.34% | 4.99 | 83.07% |
| | Unscented KF | 345 | 18296 | 9.35% | 4.99 | 83.07% |
| | Particle Filter | 350 | 18322 | 13.17% | 5.01 | 79.37% |
| YOLOv5 | Deep SORT | 371 | 17,740 | 4.57% | 2.78 | 85.89% |

Table C.4: Performance metrics comparison of object tracking algorithms for videos 120a to 120d, where ground truth centroids are 21,458, 18,501, 24,373 and 12,454 respectively.

| **Video (Actual Sheep Count)** | **Tracker** | **Metrics** | | | | |
|---|---|---|---|---|---|---|
| | | **SC** | **MC** | **FPR↓** | **MOTPE↓** | **MOTA↑** |
| 120a (276) | Kalman Filter | 273 | 20,954 | 0.29% | 0.08 | 97.30% |
| | Extended KF | 273 | 20,923 | 0.39% | 0.08 | 97.12% |
| | Unscented KF | 273 | 20,923 | 0.40% | 0.08 | 97.11% |
| | Particle Filter | 273 | 20,933 | 0.34% | 0.08 | 97.21% |
| 120b (254) | Kalman Filter | 254 | 18,426 | 0.45% | 0.03 | 99.13% |
| | Extended KF | 254 | 18,418 | 0.45% | 0.03 | 99.10% |
| | Unscented KF | 254 | 18,418 | 0.45% | 0.03 | 99.10% |
| | Particle Filter | 255 | 18,409 | 0.97% | 0.03 | 98.53% |
| 120c(362) | Kalman Filter | 362 | 24,110 | 1.11% | 0.05 | 97.77% |
| | Extended KF | 362 | 24,101 | 1.26% | 0.05 | 97.62% |
| | Unscented KF | 362 | 24,101 | 1.26% | 0.05 | 97.62% |
| | Particle Filter | 360 | 24,103 | 1.56% | 0.05 | 97.33% |
| 120d (203) | Kalman Filter | 201 | 11,045 | 6.51% | 0.11 | 81.87% |
| | Extended KF | 201 | 11,015 | 7.58% | 0.12 | 80.87% |
| | Unscented KF | 201 | 11,015 | 7.58% | 0.12 | 80.87% |
| | Particle Filter | 202 | 11,043 | 8.06% | 0.13 | 80.61% |