UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Techniques for cattle detection, duplicate removal and counting in large pasture areas using multiple aerial images

Victor Hugo Andrade Soares

Doctoral Thesis of the Postgraduate Program in Computer Science and Computational Mathematics (PPG-CCMC)

ICMC USP SÃO CARLOS

Victor Hugo Andrade Soares

Techniques for cattle detection, duplicate removal and counting in large pasture areas using multiple aerial images

Thesis submitted to the Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, in partial fulfillment of the requirements for the degree of the Doctor in Science - Program in Computer Science and Computational Mathematics.

Concentration area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Ricardo José Gabrielli Barreto Campello

Coadvisor: Prof. Dr. Moacir Antonelli Ponti

Final version

USP - São Carlos

February 2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

S676t

Soares, Victor Hugo Andrade
Techniques for cattle detection, duplicate removal and counting in large pasture areas using multiple aerial images / Victor Hugo Andrade Soares; orientador Ricardo José Gabrielli Barreto Campello; coorientador Moacir Antonelli Ponti. -- São Carlos, 2024.
133 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. CONTAGEM DE ANIMAIS. 2. DETECÇÃO DE OBJETOS. 3. VISÃO COMPUTACIONAL. 4. PECUÁRIA DE PRECISÃO. 5. REMOÇÃO DE DUPLICATAS. I. Campello, Ricardo José Gabrielli Barreto, orient. II. Ponti, Moacir Antonelli, coorient. III. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Victor Hugo Andrade Soares

Técnicas para detecção de gado, remoção de duplicatas e contagem em grandes áreas de pastagem usando múltiplas imagens aéreas

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Doutor em Ciências - Ciências de Computação e Matemática Computacional.

Área de concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Ricardo José Gabrielli Barreto Campello

Co-orientador: Prof. Dr. Moacir Antonelli Ponti

Versão revisada

USP - São Carlos

Fevereiro 2024

Em memória de Jonatas Batista Costa das Chagas,
que por vezes contribuiu com ideias e conhecimento
que me ajudaram na construção deste trabalho.

ACKNOWLEDGEMENTS

Agradeço primeiramente a Deus pelo dom da vida e Sua infinita misericórdia para comigo.

Às instituições financeiras que apoiaram este projeto e o tornaram possível. Este trabalho foi realizado com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Também agradeço à empresa Nitryx, que apoiou o projeto financeiramente, forneceu equipamentos e acesso às fazendas que serviram de base para a coleta dos dados necessários ao desenvolvimento deste trabalho.

Ao meu orientador, Dr. Ricardo Campello, pela paciência, suporte intelectual e imensurável contribuição para o meu crescimento pessoal e profissional, desde o mestrado.

Ao meu co-orientador, Dr. Moacir Ponti, por todo o conhecimento compartilhado e pela disposição e suporte ao assumir a co-orientação deste trabalho.

Ao Dr. Rodrigo Gonçalves, que desempenhou um papel fundamental no surgimento e motivação para o desenvolvimento do tema proposto.

Aos meus pais, José Luiz e Marli, que desde cedo me motivaram a estudar e aprender mais sobre computadores, o que se tornou uma paixão.

Ao meu irmão, Vinícius, pela compreensão e apoio em minha vida acadêmica, muitas vezes tendo que cobrir minha ausência no trabalho devido à universidade.

Agradeço à minha esposa, Dominique Ferreira, minha companheira que sempre me suporta em amor. Sem ela, eu não teria ingressado no ensino superior e, consequentemente, não teria chegado até aqui.

Por fim, agradeço ao meu filho, Elliot Ferreira Soares, que nasceu durante este doutorado e, desde então, tem sido minha constante fonte de motivação para ser sempre alguém melhor.

"As coisas são mais belas quando vistas de cima."
Santos Dumont

ABSTRACT

SOARES, V.H.A. **Techniques for cattle detection, duplicate removal and counting in large pasture areas using multiple aerial images**. February 2024. 133p. Thesis (Doctorate in Science) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, USP - São Carlos, February 2024.

Among the production areas with largest impact on global economy, agriculture and livestock play a prominent role. Technologies have been developed in order to automate and increase the efficiency of these fields. The use of Unmanned Aerial Vehicles (UAVs) has been extensively investigated to improve the efficiency of agricultural production and in the monitoring of animals. One of the most important and challenging tasks in animal monitoring is cattle counting. Traditional manual counting methods are laborious and error-prone, while existing automated approaches struggle with duplicate animal detection. This work presents a method for detecting and counting cattle in aerial images acquired via UAVs. This method leverages Convolutional Neural Networks (CNNs) and employs a graph-based optimization technique to eliminate duplicate animal detection in overlapping images. Our results emphasize the importance of maximizing animal matching to mitigate duplicate counts. Additionally, we integrate multi-attributes, encompassing velocity, direction, state (lying down or standing), color, and distance, to enhance duplicate removal and counting precision. We conducted extensive experiments and training to seamlessly incorporate these attributes into our methodology. Furthermore, we provide a dataset comprising authentic images captured in extensive pasture areas, suitable for both training and testing/benchmarking cattle counting techniques. When evaluating detection and counting, our outcomes underscore the competitiveness of the proposed method while significantly reducing the computational cost of the overall counting process. When focusing solely on duplicate removal, our method surpasses state-of-the-art techniques, achieving an average percentage error of 2.34%. In summary, the proposed method marks a substantial stride towards more efficient cattle counting practices and enhanced livestock management in agriculture.

Keywords: Object Detection, Cattle Counting, Duplicate Removal, Precision Farming, Unmanned Aerial Vehicles (UAVs), Graph-based method, Livestock

RESUMO

SOARES, V.H.A. **Técnicas para detecção de gado, remoção de duplicatas e contagem em grandes áreas de pastagem usando múltiplas imagens aéreas**. February 2024. 133p. Tese (Doutorado em Ciências) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, USP - São Carlos, February 2024.

Nas áreas de produção com maior impacto na economia global, a agricultura e a pecuária desempenham um papel proeminente. Tecnologias têm sido desenvolvidas com o intuito de automatizar e aumentar a eficiência desses setores. O uso de Veículos Aéreos Não Tripulados (VANTs) tem sido amplamente investigado para aprimorar a eficiência da produção agrícola e o monitoramento de animais. Uma das tarefas mais importantes e desafiadoras no monitoramento de animais é a contagem de gado. Métodos tradicionais de contagem manual são trabalhosos e propensos a erros, enquanto abordagens automatizadas existentes lutam com detecções duplicadas de animais. Este trabalho apresenta um método para detectar e contar bovinos em imagens aéreas obtidas por meio de VANTs. Este método é baseado em Redes Neurais Convolucionais (CNNs) para detecção, e utiliza uma técnica de otimização baseada em gráfos para eliminar detecções duplicadas de animais em imagens sobrepostas. Nossos resultados destacam a importância de maximizar a correspondência de animais para mitigar contagens duplicadas. Além disso, integramos múltiplos atributos dos gados, incluindo velocidade, direção, estado (deitado ou em pé), cor e distância, para aprimorar a remoção de duplicatas e a precisão da contagem. Realizamos extensos experimentos e treinamentos para incorporar esses atributos em nossa metodologia. Além disso, fornecemos um conjunto de dados com imagens capturadas em extensas áreas de pastagem, adequadas tanto para o treinamento quanto para testes e avaliação de técnicas de contagem de gado. Ao avaliar a detecção e a contagem, nossos resultados destacam a competitividade do método proposto, ao mesmo tempo em que reduzem significativamente o custo computacional do processo de contagem como um todo. Ao focar exclusivamente na remoção de duplicatas, nosso método supera as técnicas mais avançadas, atingindo um erro médio percentual de 2,34%. Em resumo, o método proposto representa um grande avanço em direção a práticas de contagem de gado mais eficientes e ao aprimoramento da gestão do gado na agricultura.

Palavras-chave: Detecção de Objetos, Contagem de Gado, Remoção de Duplicados, Fazenda de precisão, Veículos Aéreos não Tripulados, Método baseado em grafos, Pecuária

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ABS	Absolute
alt	Relative Altitude
ANN	Artificial Neural Network
API	Application Programming Interface
AVG	Average
BFS	Breadth-first search
BR	Brazil
CMOS	Complementary metal–oxide–semiconductor
CNN	Convolutional Neural Network
DA	Data Augmentation
DC	Direction Classes
DEM	Digital Elevation Model
DT	Distance Threshold
$f35$	Focal In 35mm
fl	Focal Length
FN	False Negative
FP	False Positive
GB_{multi}	Graph-based Multi-attribute
GB_D	Graph-based Distance
GBT_{multi}	Graph-based Thresholded Multi-attribute
GIS	Geographic Information Systems
GNSS	Global Navigation Satellite System

GPS	Global Positioning System
GPU	Graphics Processing Unit
GSD	Ground Sample Distances
GT	Ground truth
H	Height
ILSVRC14	2014 ImageNet Large-Scale Visual Recognition Challenge
ISO	International Organization for Standardization
lat	Latitude
lng	Longitude
LOPO	Leave-One-Pasture-Out
$minR$	Minimum Threshold
MLP	Multilayer Perceptron
MOT	Multiple Object Tracking
MS	Mato Grosso do Sul
MS-COCO	Common Objects in Context
R-CNN	Region-based Convolutional Network
RAM	Random Access Memory
RGB	Red Green Blue
RTK	Real Time Kinematics
SfM	Structure from Motion
SGD	Stochastic Gradient Descent
STD	Standard Deviation
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UAV	Unmanned Aerial Vehicle
W	Width

CONTENTS

CHAPTER

1

INTRODUCTION

Livestock farming and agriculture have a significant impact on global food production and the economy (BANK, 2021). The sectors are responsible for the use of approximately 40% of the world's land area (ALSTON; PARDEY, 2014). Despite the high occupancy rate, in 2012 it was estimated that these sectors were responsible for 2.8% of gross world product and 19% of the world's workers population (World Bank, 2012).

Efficient management of cattle, particularly in large pasture areas, is crucial for optimizing production and ensuring animal welfare. Traditional manual counting methods are laborious, time-consuming, and prone to high error rates (FARJON; HUIJUN; EDAN, 2023). In recent years, Unmanned Aerial Vehicles (UAVs), commonly known as drones, have emerged as a highly promising technology for monitoring and management of cattle (REJEB *et al.*, 2022). Equipped with high-resolution cameras, these drones offer a significant advancement by capturing aerial images of vast territories, providing valuable information for various agricultural applications, including monitoring of plantations, pastures and livestock, to identify/control possible problems as well as to determine the location of these problems in the field (GÓMEZ-CANDÓN; CASTRO; LÓPEZ-GRANADOS, 2014; ALANEZI *et al.*, 2022).

The popularization of drones has been mainly driven by the consolidation of technologies such as the Global Positioning System (GPS), embedded microelectronics, miniature autopilot systems, mobile communication equipment, compact high-resolution digital cameras and high-power batteries. This makes drones low-cost, safe and easy to operate (GUO *et al.*, 2018).

The use of cameras and GPS embedded in drones allows it to estimate the geolocation of objects on the ground (JOHNSTON, 2006), mapping areas with high resolution and precision (GRAYSON *et al.*, 2018). GPS accuracy is related to external factors such as climate, wind and topography of the region overflown. In ideal flight conditions, such as a

sunny day with little wind and under flat terrain, the accuracy of the estimated geolocation of points on the ground can reach an average error of 2cm (BARRY; COAKLEY, 2013).

In spite of these remarkable advances in technology, there are still several factors that hinder the use of drones for monitoring animals, especially livestock. Regarding one of the most important drone related tasks in this context, namely, counting animals using aerial images, the main drawback is related to the movement of animals through the field, which poses major challenges (BARBEDO; KOENIGKAN, 2018). Ideally, one would like to obtain an image record of the entire area with just one photograph, however, there are laws that limit the flight altitude of drones in many countries. For large properties with 40ha or more, which includes the farms visited to collect the images used in this work, capturing the entire area with only one photograph is only possible via satellite. However, most satellites do not have sufficient resolution to allow the detection and monitoring of animals. Even satellites currently considered to be high resolution, such as Geo Eye 1 and WorldView (XUE; WANG; SKIDMORE, 2017), generate images in which each animal is represented by a few pixels, making precise detection virtually impossible. Additionally, the acquisition of images through these satellites has a high cost (BARBEDO; KOENIGKAN, 2018).

When it is not possible to capture the entire area in just one photograph, it is common practice to use the drone to register the entire area through multiple images (SHAO *et al.*, 2020; XU *et al.*, 2020a; XU *et al.*, 2020b). In this scenario, animals can be counted in each image separately, but this requires dealing with duplicated animals, since the same animal can appear in more than one image (BARBEDO *et al.*, 2020). Another approach is to generate a single image, where pixel combination algorithms are used to form a single mosaic from all the original images. However, current rendering algorithms do not prevent animals from appearing replicated in the final mosaic/image. In addition, the rendering process may cause image distortions and is highly computationally demanding, becoming prohibitive/infeasible in many application scenarios.

Careful consideration of cattle behavior and meticulous flight planning can increase the likelihood of capturing photos when the cattle are relatively stationary, thus reducing the chances of incorrect matching (KILGOUR, 2012; BARBEDO *et al.*, 2020). However, this is not always possible in large areas and it may only mitigate the problem to a limited extent. For instance, covering a large area while attempting to minimize cattle movement requires minimizing flight times, which in turn depends on the degree of overlap between images. Typically, there is a challenging trade-off involved in the definition of the image overlap degree, which plays a pivotal role in flight planning. High overlap is essential, e.g., for constructing mosaics (DANDOIS; OLANO; ELLIS, 2015). In general, some degree of overlap is also important for cattle detection and counting techniques more broadly, because the absence of a minimum degree of overlap may lead to undetected

cattle, which will end up unaccounted for, particularly if they are located near the edges of images and move between photograph shots. On the other hand, higher overlap can significantly extend the required flight times, providing more opportunities for cattle to move, potentially resulting in duplicate counts. In addition, using high overlap levels can be impractical in large areas, as drones may not cover the entire area within a single battery cycle. In this case, mid-flight battery changes may be required, increasing flight times and the uncertainty about cattle position and count even further.

For counting animals in the images (individual or mosaics), object detection techniques have been used, with emphasis on Convolutional Neural Networks (CNNs) (PENATTI; NOGUEIRA; SANTOS, 2015). These networks were designed as a solution to the problem of image recognition, initially for classification (CNNs) and subsequently for object detection and segmentation. These methods are designed as deep artificial neural networks, which use convolution operations to learn features in the space of input images (PONTI *et al.*, 2017) guided by an objective function that aims to achieve the desired task. In livestock, CNNs have been widely used in studies of dairy cows behavior (JIANG *et al.*, 2019), posture estimation (LI *et al.*, 2019), segmentation and contour extraction to calculate body condition score (QIAO; TRUMAN; SUKKARIEH, 2019), measure individual food consumption (BEZEN; EDAN; HALACHMI, 2020), breed recognition (WEBER *et al.*, 2020), disease diagnosis (MOHAN; RAJU; JANARTHANAN, 2019) and, particularly, counting animals using images from drones (CHAMOSO *et al.*, 2014; SHAO *et al.*, 2020; BARBEDO *et al.*, 2020; XU *et al.*, 2020a).

In this work, we propose methods for detecting and counting animals in large areas, using images obtained by drones and geolocation data. The methods include data acquisition, evaluation of deep neural networks for animal detection, and a novel graph-based counting algorithm. We also present the culmination of research efforts along multiple directions to improve effectiveness in cattle counting and duplicate removal. Our combined approach involves the analysis of multiple attributes beyond image pixel level. In particular, for each detected cattle, we take into account not only its estimated geolocation, but also its state (lying down or standing), color, velocity and direction. We investigate how such attributes can be properly combined and used to assign weights to the graph algorithm that we use to model and effectively solve the duplicate removal problem.

1.1 Contributions

The contribution of this thesis is three-fold:

- a novel method for cattle counting in extensive areas based on geolocations of animals and a graph-based algorithm to remove duplicates. This approach outperforms the state of the art, notably in terms of reducing duplicate counting, while also

significantly improving runtime efficiency.

- a thorough investigation into the use of cattle visual attributes to enhance the duplicate removal algorithm. By leveraging these attributes, we achieved more accurate and reliable results in the cattle counting process.

- a novel diverse and extensive dataset of real-world images captured using drones across various pastures. These images encompass a wide range of cattle breeds, backgrounds, altitudes, times of the day, and lighting conditions. This dataset serves as a valuable resource for the research community, aiding in the training of new methods and providing a benchmark for cattle counting tasks.

1.2 Outline of the Thesis

The remainder of this thesis is organized as follows:

In Chapter 2, we explore the background that form the basis of this research. This chapter provides an overview of key concepts, including a brief description of applications using drones in Precision Farming. It also details methods employed to Computer Vision, as Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs) for both, image classification and object detection tasks, which are important components of this study.

Chapter 3 presents an extensive review of related work in the field. Existing research is categorized into two main sections: works focusing on animal detection and counting from single images, and those addressing the task of counting animals from multiple images. This chapter sets the stage for the novel contributions in subsequent chapters.

Chapter 4 unveils the proposed methodology for cattle detection and counting across large areas, leveraging geolocation data and employing a graph-based algorithm to eliminate duplicate counts. Additionally, it introduces a new dataset for training and evaluation purposes, enriching the resources available to the research community.

Chapter 5 embarks on an in-depth exploration of multi-attribute analysis for cattle. It proposes a method to integrate attributes like color, state (lying down or standing up), direction, and velocity into the duplicate removal algorithm. Classifiers are employed to learn and detect the attributes within the counting pipeline, while additional evaluation methods, such as ablation studies, assess the contribution of each attribute.

The final chapter, Chapter 6, serves as a culmination of the research. It summarizes the main findings and contributions discussed throughout the thesis. Additionally, it acknowledges the limitations of the work and illuminates potential avenues for future research and exploration in this dynamic field of study.

CHAPTER

2

BACKGROUND

In this chapter, we present concepts that encompass the use of drones in precision agriculture and the field of computer vision. Specifically, we provide an overview of drone applications in precision farming. Furthermore, we delve into essential concepts in computer vision, with a particular focus on image classification and object detection tasks. The chapter offers a concise explanation of key concepts within Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs), and some examples of networks for images classification and object detection. These foundational insights set the stage for a comprehensive exploration of the ensuing chapters.

2.1 Drones for Precision Farming

In recent years, Unmanned Aerial Vehicles (UAVs), more commonly known as drones, have experienced a surge in popularity, particularly within the realm of precision farming. This newfound enthusiasm stems from the confluence of various factors, including advancements in drone technology, the availability of high-quality cameras, GPS precision, and a significant reduction in the cost of embedded components (GUO *et al.*, 2018).

Drones are increasingly used for a range of agricultural tasks, such as mapping extensive agricultural areas (VEROUSTRAETE, 2015), monitoring and surveying crops or livestock (HUANG *et al.*, 2022), delivering items (FRACHTENBERG, 2019), and crop dusting (KURKUTE *et al.*, 2018). The latter involves transporting tanks of fertilizers and pesticides for precise, targeted crop spraying, a task that drones perform with significantly higher accuracy compared to traditional tractors or manual labor (ME *et al.*, 2016).

In the realm of agricultural drones, two primary categories dominate the landscape: fixed-wing and multi-rotor drones (BUDIHARTO *et al.*, 2019). Each of these drone types offers distinct advantages. Multi-rotor drones are popular for their agility and ease of use. However, their batteries typically have shorter lifespans, limiting the amount of time they

can spend in the air. On the other hand, fixed-wing drones are rarer, bet they are better suited for larger farms due to their extended flight time capacity. With the ability to spend more time in the air, they can cover more substantial agricultural areas. Furthermore, fixed-wing drones are capable of carrying larger payloads, including multiple sensors. This broader sensor array significantly enhances data collection capabilities, making them a compelling choice for precision farming applications.

Central to the effectiveness of agricultural drones is their reliance on Global Positioning System (GPS) or Global Navigation Satellite System (GNSS) technology (GUO *et al.*, 2018). The integration of these positioning systems enables drones to operate autonomously and navigate the complex geographical landscapes of farms. Without GPS/GNSS, drones would merely function as radio-controlled aircraft, limiting their potential to perform autonomous and precise tasks (BUDIHARTO *et al.*, 2019)

Drones not only capture traditional photographs but also offer support to Geographic Information Systems (GIS) (BUDIHARTO *et al.*, 2021). This support allows for the production of orthophotos, which are geometrically corrected aerial images, free from distortions (CHEN *et al.*, 2023). Moreover, the sensor-rich environment of drones provides an abundance of data, facilitating precise estimation of the position of objects on the ground (HABCHI *et al.*, 2020). This wealth of information enables farmers to make informed decisions, manage resources efficiently, and enhance crop yields through data-driven insights.

By leveraging this advanced technology, precision farming has taken a giant leap forward, ensuring sustainable and productive agricultural practices that align with the demands of our modern world. The integration of drones into agricultural landscapes represents a important step toward smarter, data-driven, and sustainable farming practices (KRISHNA, 2018).

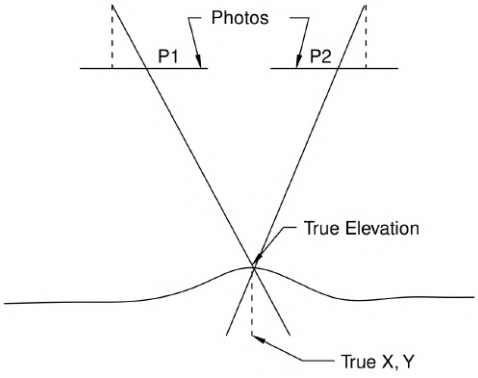
2.1.1 Mapping and Geolocation Estimation with Drone Imagery

Drone imagery plays a pivotal role in precision agriculture, supporting applications like mapping (RACHMAWATI *et al.*, 2021) and geolocation estimation (HABCHI *et al.*, 2020). The use of orthophotos and mosaics is widespread due to their capacity to provide undistorted, georeferenced images that serve as the foundation for a variety of applications.

Orthophotos are precise, orthorectified images that rectify the distortion effects present in standard aerial photos and in the relief. This correction is achieved through the amalgamation of geographical and elevation data, resulting in images where objects are uniformly scaled, and their positions precisely correspond to a constant coordinates (MORGAN; FALKNER, 2001), as illustrated in Figure 1.

Orthophotos are used in tasks like land mapping, crop health assessment, and

Figure 1 – How relief correction is applied in orthophoto generation.



Source: Morgan and Falkner (2001)

environmental monitoring.

Creating an orthophoto involves the following steps:

1. **Georeferencing:** Each aerial image is tagged with geographic coordinates, often through GPS data, ensuring precise spatial referencing.

2. **Camera Calibration:** The characteristics of the camera lens and sensor, including focal length and lens distortion, are meticulously calibrated to ensure accurate measurements.

3. **Digital Elevation Model (DEM):** The utilization of a Digital Elevation Model, often derived from sources like LiDAR, provides crucial terrain elevation data. The DEM is essential for correcting image distortions linked to terrain variations.

4. **Bundle Adjustment:** Bundle adjustment algorithms optimize camera positions and orientation parameters to enhance alignment with the acquired images.

5. **Orthorectification:** Geometric transformations are applied to correct perspective distortions. The orthorectification process can be mathematically represented as:

$$(X, Y, Z)_{\text{orthophoto}} = (X, Y, Z)_{\text{image}} + f(X, Y, Z)_{\text{DEM}}$$

Here, $(X, Y, Z)_{\text{orthophoto}}$ represents corrected orthophoto coordinates, $(X, Y, Z)_{\text{image}}$ denotes image coordinates, and $f(X, Y, Z)_{\text{DEM}}$ compensates for terrain elevation.

Mosaics, on the other hand, involve the composition of multiple images into a seamless, comprehensive view of a specific area. While orthophotos rectify individual images, mosaics focus on the amalgamation of these images to create a singular, large-scale

view. The process entails matching key points in overlapping images and blending them to yield a cohesive result.

In drone photography, overlap refers to the degree to which successive images share common features and details (MORGAN; FALKNER, 2001). Overlap is classified into two main categories:

- **Longitudinal Overlap:** This type of overlap occurs when images overlap along the flight direction, often referred to as "along-track" overlap.
- **Lateral Overlap:** Lateral overlap is created when images overlap side to side or "across-track".

A higher degree of overlap, typically around 60-80%, allows redundancy in information, leading to the production of accurate orthophotos and mosaics (DANDOIS; OLANO; ELLIS, 2015). The algorithms take into account this overlap, enabling corrections for even minor terrain variations, ensuring the highest level of accuracy. Although, high overlap results in more images capturing the same area, which lead to extended flight plans. In scenarios where the goal is detailed land mapping, high overlap is essential to produce precise orthophotos and mosaics. However, in scenarios where long flight plans are not possible or convenient, a more efficient approach can be adopted, leveraging metadata and geolocation estimation for accurate results with reduced computational demands.

Through an analysis of metadata, including drone location, sensor specifications, lens configuration, and camera pointing position, the geolocation of objects on the ground can be estimated (JOHNSTON, 2006). The precision of geolocation estimation depends on factors such as GPS accuracy, sensor calibration, and drone altitude (WOLF; DEWITT; WILKINSON, 2013). While orthophotos provide a high degree of accuracy due to rigorous correction processes, geolocation estimation from images offers rapid assessment capabilities but may not achieve the same level of precision. By optimizing the balance between geolocation accuracy and processing efficiency, drone-based geolocation estimation proves to be a valuable asset for tasks that do not require millimeter precision considering global coordinates, but with good relative precision across the images.

In summary, drone imagery emerges as a versatile resource for mapping, orthophoto generation, and geolocation estimation. These techniques find widespread application in precision agriculture, enhancing land management, resource allocation, crop health assessment, and animal monitoring.

2.2 Computer Vision

Humans have a remarkable ability to perceive and recognize the three-dimensional structure of the world around them. When looking at a photograph, for instance, we can

easily count (and label) all the individuals appearing in the image, while also discerning their emotions through facial expressions. This is made possible by our complex and efficient visual system. The field of computer science that aims to replicate the attributes of human vision using software and hardware is called *Computer Vision* (SZELISKI, 2010).

When we aim to solve a visual recognition problem using computational methods, it may be tempting to underestimate the challenge. However, among the visual tasks that we can instruct a computer to perform, analyzing a scene and recognizing all the objects within it remains one of the most challenging tasks (SZELISKI, 2010). This difficulty arises because the appearances of objects can vary considerably due to changes in lighting, viewpoint, deformations, and the natural intra-class variability, such as the "dog" class, which can contain visually distinct animals due to the presence of different breeds (FELZENSZWALB *et al.*, 2010).

The problem of image recognition can be mainly categorized into two distinct approaches. If we have an image with an object and we want to classify it based on a pre-defined set of classes, it is referred to as a *recognition* or *classification task* (SZELISKI, 2010). In such tasks, we identify key visual features and determine whether these features align with the geometric characteristics of any class in a predefined model created from known examples. Conversely, if we have an image and we know what we are looking for, and the objective is to traverse the image and determine the location of that object, this task is referred to as *Object Detection* (SZELISKI, 2010). In this work, our primary objective is the counting of animals using aerial images, making it an object detection task. However, we will also employ classification networks to differentiate specific attributes of the animals.

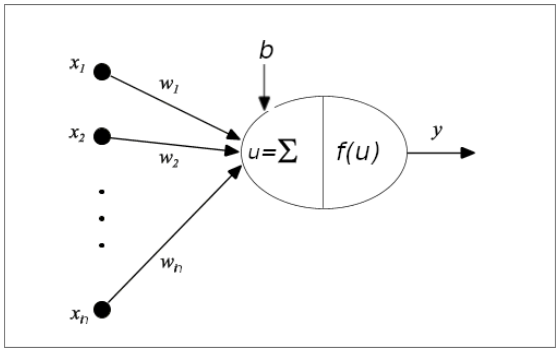
The following sections provide a foundational understanding of Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs), which are instrumental in the domains of image classification and object detection.

2.2.1 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are computational methods that present a mathematical model inspired by the neural structure of intelligent organisms (WITTEN; FRANK; HALL, 2011). An ANN is composed of simple units known as artificial neurons, which are interconnected and organized into layers (MELLO; PONTI, 2018). Figure 2 illustrates a neuron model known as McCulloch and Pitts (MCP) (MCCULLOCH; PITTS, 1943). In this model, where $x \in \mathbb{R}^n$ represents a multidimensional vector with input values, the neuron operates by associating these inputs with corresponding weights, forming a weighted sum $u = \sum_{i=1}^{n} w_i \cdot x_i$. Additionally, a bias term, $b \in \mathbb{R}$, is (optionally) introduced and added to the weighted sum, contributing to the overall activation of the neuron.

The activation function, represented here as $f(u)$, plays a crucial role in determining

Figure 2 – Model of a MCP Artificial Neuron



Source: Elaborated by the author.

the neuron's output y. There are various activation functions suitable for this task, but they are generally required to be differentiable. One widely used activation function is the sigmoid function, $f(u) = 1/(1 + \exp^{-u})$, which ensures that the output of the neuron falls within a bounded range between 0 and 1. The neuron processes its inputs and produces an output determined by the activation function applied to u (MCCULLOCH; PITTS, 1943).

The architecture of ANNs is typically organized in layers. These layers can be broadly classified into three groups: Input layers, where patterns are introduced into the network; Hidden layers, where the majority of processing takes place through weighted connections, which can be thought of as feature extractors; and Output layers, where the final result is presented (MELLO; PONTI, 2018).

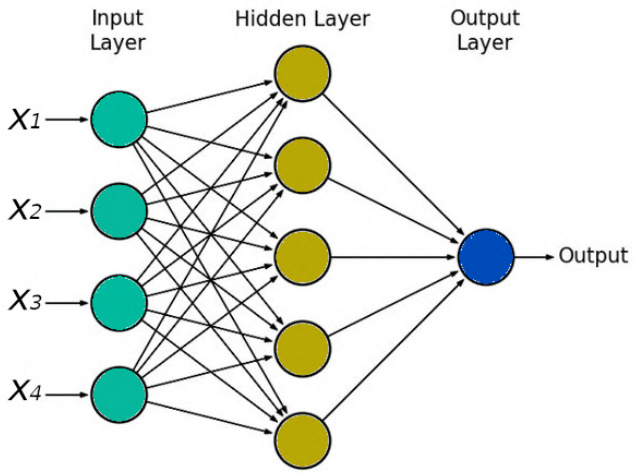
An ANN is specified primarily by its topology and the characteristics of its nodes and training rules. A Multilayer Perceptron (MLP), illustred in Figure 3, is a classic example of this structure, and it resembles a weighted graph that is directed and acyclic. In this network, the nodes represent neurons, and the edges signify the connections among them, allowing the output of one neuron to serve as the input to another (SHALEV-SHWARTZ; BEN-DAVID, 2014).

Notably, the "knowledge" of the MLP is stored in the weights of the neuron connections. These weights can be initially assigned random values or pre-learned from previous tasks, and they play a crucial role in shaping the network's ability to model complex relationships and solve specific problems.

2.2.1.1 Neural Network Training

The essence of neural networks' capabilities lies in their ability to "learn". In this context, learning signifies the network's quest to find a general solution within a specific problem class (SHALEV-SHWARTZ; BEN-DAVID, 2014). This journey of acquiring knowledge within neural networks takes place through various learning paradigms. The three primary learning paradigms are:

Figure 3 – Example architecture of a Multilayer Perceptron network (MLP).



Source: Elaborated by the author.

- **Supervised Learning:** In this paradigm, the network is provided with input data and corresponding correct output data. It aims to learn a mapping from inputs to outputs.

- **Unsupervised Learning:** Here, the network deals with unlabeled data and must discover patterns or structures in the data without explicit guidance.

- **Reinforcement Learning:** This paradigm involves an agent interacting with an environment, taking actions to maximize cumulative rewards, thus learning how to behave in various situations.

Training a neural network involves repeated iterations through the dataset, where each complete pass is termed an "epoch"(SARKAR; BALI; SHARMA, 2018). Each epoch allows the network to refine its internal parameters (usually, the weights of the connections between neurons) and inch closer to producing the correct output for a given input.

During training, there are two main modes for defining an epoch:

- **Standard Mode:** In this mode, the network computes the weights in each presentation of input data. This means that for each epoch, there are N weight updates, where N is the number of input data points.

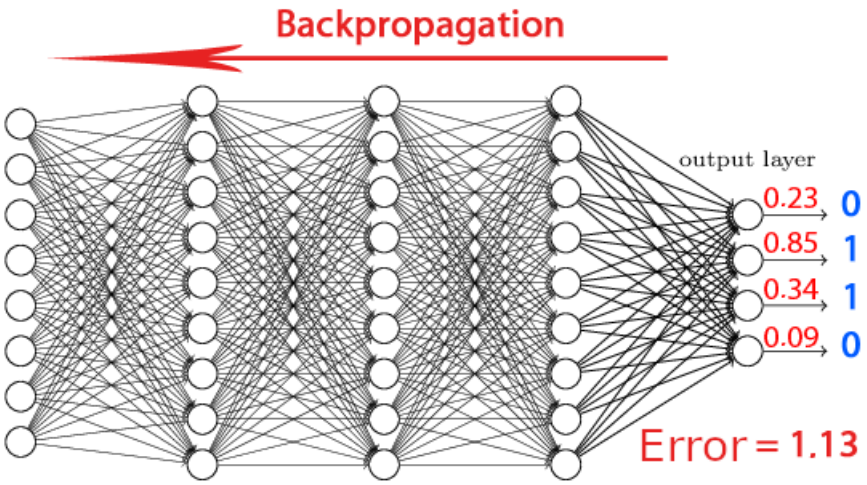
- **Batch Mode:** Contrastingly, in this mode, only one weight update is computed per epoch. The network processes all the input data, and only one update is performed for the entire epoch.

In our work, we focus on supervised training, which is particularly relevant for tasks like image classification. Now, let's delve into the specific training method used in supervised learning.

The most common method for training an Artificial Neural Network (ANN) is Backpropagation (LANG, 1988). This approach comprises two crucial phases: forward and weight update.

Figure 4 illustrates a fully connected network where the red values in the output layer result from forward propagation, while the blue values represent the model, i.e., the ideal values that should be obtained. The sum of the differences between the obtained and ideal values generates an error of 1.13. The error values are then backpropagated, and deltas are computed for each node or neuron in the network.

Figure 4 – Example of the backpropagation method. The red values are obtained through data propagation. The blue values are the expected (ideal) values. The sum of differences between obtained and expected values results in an error.



Source: Elaborated by the author.

To update the weights, gradients are calculated by multiplying the output delta (error) and the input activation. A learning rate is employed to determine the percentage of the gradient to subtract from the original weight, effectively updating the node weights (SARKAR; BALI; SHARMA, 2018).

These two steps, forward and weight update, are repeated several times through multiple epochs until satisfactory results are achieved. Typically, Backpropagation is used in conjunction with optimization algorithms or functions, with Stochastic Gradient Descent (SGD) being one of the most commonly used optimization methods in the field of Machine Learning (GOODFELLOW; BENGIO; COURVILLE, 2016).

This iterative process of forwarding input data, computing errors, backpropagating, and adjusting weights is what allows neural networks to "learn" and adapt their internal

parameters to achieve their intended goals.

2.2.2 Convolutional Neural Network (CNN)

Starting from 2012, when the competition ImageNet (RUSSAKOVSKY *et al.*, 2015), of image recognition, was won by the team of (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), the winning neural network, *AlexNet*, began to be successfully applied to various tasks in the field of Computer Vision (GIRSHICK *et al.*, 2014a; KARPATHY *et al.*, 2014; LONG; SHELHAMER; DARRELL, 2015; WANG; YEUNG, 2013). The success of the *AlexNet* network is considered the great milestone in the research of Convolutional Neural Networks (CNNs) and *Deep Learning* nomenclature (SZEGEDY *et al.*, 2015b). From it, several new neural networks emerged, causing great advances in the field of Computer Vision.

CNNs are a specialized type of neural network for data processing with a grid-like topology, similar to a grid (GOODFELLOW; BENGIO; COURVILLE, 2016). Common examples include time series data, which can be seen as a 1D grid with samples at regular time intervals, and images, which form a 2D grid of pixels.

2.2.2.1 Convolution

Convolution is an operation performed on a pixel's neighborhood, denoted as $I(i, j)$, by a filter or kernel K, resulting in a scalar value for pixel $S(i, j)$. Convolution between image I and kernel K is typically represented as an asterisk ($*$):

$$S(i, j) = (I * K)(i, j)$$

Assuming the kernel has dimensions of $m \times n$, where $m = 2a + 1$ and $n = 2b + 1$, the convolution of the image by the kernel is computed as follows:

$$S(i, j) = (I * K)(i, j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} I(i - s, j - t)K(s, t) \tag{2.1}$$

It is worth noting that convolution results in a scalar value representing only pixel (i, j). Consequently, the kernel slides over the image, computing convolution for all the pixels.

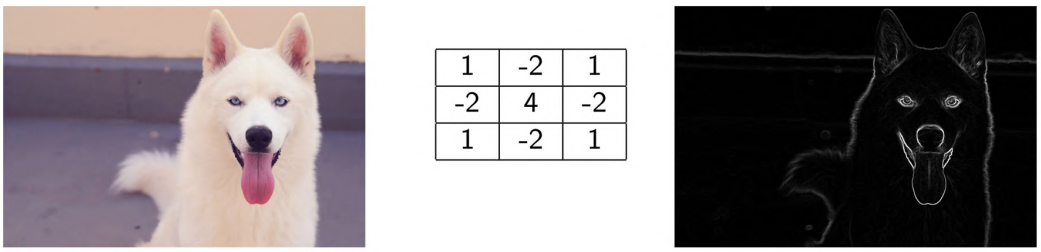
In the field of neural networks, for the sake of simplicity, convolution is often implemented as cross-correlation, which is equivalent to computing convolution with the kernel values mirrored:

$$S(i, j) = (I \star K)(i, j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} I(i + s, j + t)K(s, t) \tag{2.2}$$

Note that if the kernel is symmetric, convolution and cross-correlation are equivalent. In practice, this differentiation is not crucial for neural network implementations, as many libraries implement cross-correlation and call it convolution (GOODFELLOW; BENGIO; COURVILLE, 2016). Throughout this text, we consider both operations as convolution by convention.

Figure 5 illustrates a convolution result using a Laplacian kernel for edge detection. In the field of neural networks, the values of convolution kernels are learned by the network itself. This means that programmers usually don't need to define the specific convolutions, as the network adapts the kernels to make the filtering more discriminative.

Figure 5 – Example of convolution with a Laplacian kernel for edge detection in an image.



Source: Elaborated by the author.

2.2.2.2 CNN Structure

CNNs typically consist of three main components (SARKAR; BALI; SHARMA, 2018):

1. **Convolutional Layers:** These layers involve multiple filters (kernels) that convolve input data across all dimensions. Convolutions help identify spatial features in images, and each kernel generates a 2D activation map. These maps are stacked to form the final output of the convolutional layers.

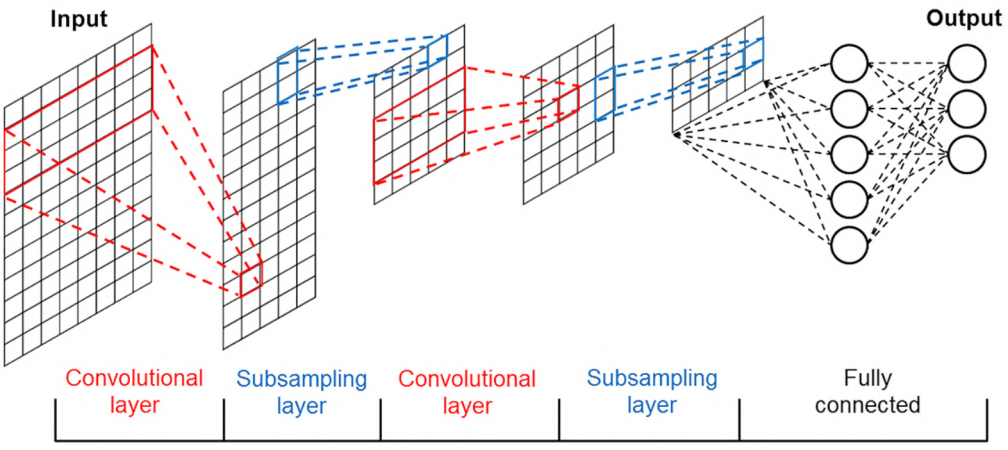
2. **Subsampling Layers:** These layers perform non-linear operations on input data and the outputs of convolutional layers. This subsampling aims to improve model generalization, prevent overfitting, and reduce computational cost. Common pooling operations include sum, average, and max pooling.

3. **Fully Connected Multilayer Perceptron (MLP):** This part is similar to fully connected layers in traditional ANNs. Each neuron is connected to all neurons in the previous layer. The values computed by these neurons reach the output layer, where image classification is achieved.

Figure 6 illustrates the common structure of a CNN. In broad strokes, convolutional layers transform the data to extract features. Subsampling layers reduce data sizes

while preserving important values, improving generalization, preventing overfitting, and enhancing computational performance. Finally, a fully connected network classifies the image.

Figure 6 – Common structure of a Convolutional Neural Network (CNN).



Source: Trakoolwilaiwan *et al.* (2017)

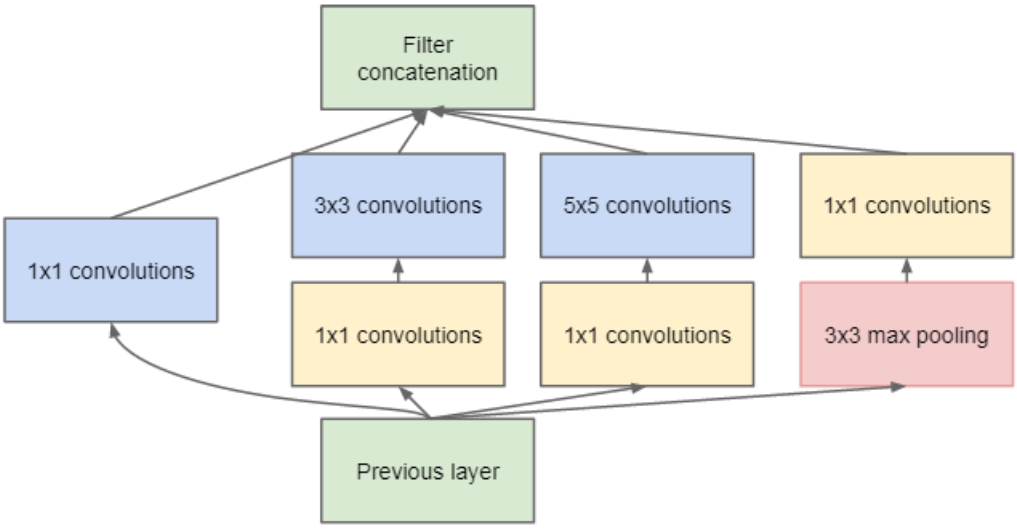
The number of new networks and applications for image recognition have been increasing dramatically. Some networks consist of hundreds or even thousands of layers. This characteristic has driven the usage of the term "Deep Learning" (SZEGEDY *et al.*, 2015b) in neural networks. The next sections introduce some of the primary CNNs for image recognition and object detection.

2.2.2.3 GoogLeNet - Inception

GoogLeNet, also known as Inception, is a CNN developed by Google for image recognition (classification). It won the 2014 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC14), the world's largest annual challenge for image recognition methods. In addition to achieving significantly better results than AlexNet, the 2012 winner, GoogLeNet reduced the number of parameters by a factor of 12 (SZEGEDY *et al.*, 2015a).

The GoogLeNet network consists of Inception convolution modules, where filters of sizes 1×1, 3×3, and 5×5 are applied. Figure 7 displays the architecture of an Inception module. Data comes from the *"Previous Layer"* and is then passed through four rows of convolution. The 1×1 convolutions are used to halve the dimensionality in the filter space, reducing computational cost. As such, they are applied before the 3×3 and 5×5 convolutions. In Figure 2.2.2.3, the light blue block (3×3 max pooling) subsamples the image, keeping only the maximum value for each 3×3 window in the image. The module's final output is a concatenation of these four convolution rows.

Figure 7 – Inception module of the GoogLeNet network with dimensionality reduction.



Source: Szegedy *et al.* (2015a)

Having defined the structure of an Inception module, the GoogLeNet network is built by stacking multiple Inception modules. Figure 8 illustrates the final structure of the GoogLeNet network. In blue, you can see the convolutions; in red, the sub-samplings; in green, normalizations; and in yellow, fully connected networks for classification. The light green rectangle highlights one of the Inception modules within the network. In total, nine Inception modules are stacked.

As seen in the light yellow blocks in Figure 8, besides the classifier at the end of the network, two additional auxiliary classifiers are connected to intermediate layers. Given the network's depth, the authors were concerned about the effectiveness of gradient propagation during training. Middle layers are expected to be highly discriminative as well. During training, the error of these auxiliary classifiers is added to the total error with a discount weight. However, these auxiliary classifiers are used only during the training phase. In the inference phase, they are discarded.

2.2.2.4 Region-based Convolutional Network (R-CNN)

The Region-based Convolutional Network (R-CNN) (GIRSHICK *et al.*, 2014b) is a CNN developed for object detection. While the R-CNN concept is simple, the method is robust. Figure 9 illustrates the R-CNN detection process. The method involves extracting approximately 2,000 regions of interest. Convolution operators are then used to compute the features for each extracted region, followed by classification. In the original proposal (GIRSHICK *et al.*, 2014b), a Support Vector Machine (SVM) is employed as the classifier.

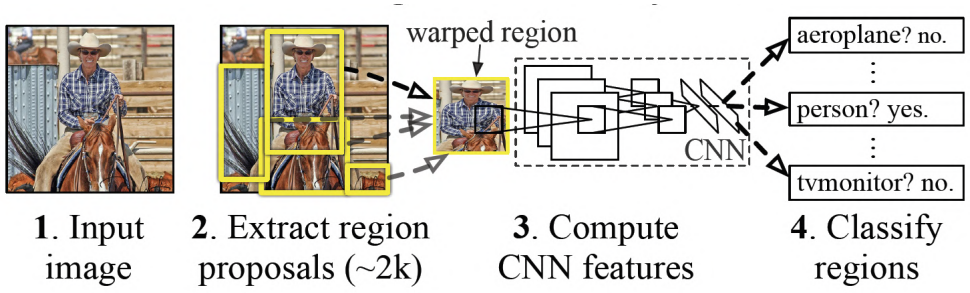
A more efficient version of R-CNN, called Fast R-CNN, was proposed in (GIRSHICK, 2015). In (HUANG *et al.*, 2017), several CNNs for object detection are compared, with Fast

Figure 8 – Complete structure of the GoogLeNet network.

Source: Szegedy *et al.* (2015a)

R-CNN yielding the best detection results but having the highest computational cost. Fast R-CNN, a network designed for object detection, is also used in conjunction with image

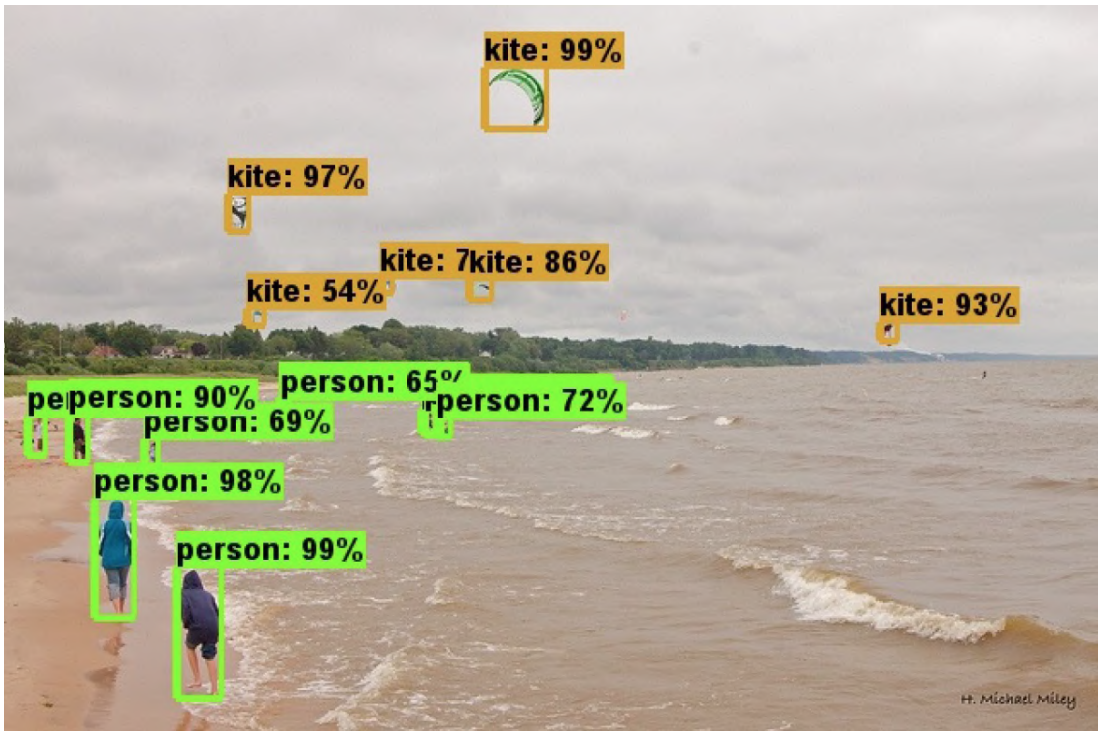
Figure 9 – R-CNN: Regions with CNN features.



Source: Girshick *et al.* (2014b)

recognition networks. For example, the GoogLeNet network can be used in the feature extraction step of the Fast R-CNN pipeline. In (HUANG *et al.*, 2017), the authors compare the usage of Inception (SZEGEDY *et al.*, 2015a), Inception v3 (SZEGEDY *et al.*, 2015b), Resnet (HE *et al.*, 2016), VGG (SIMONYAN; ZISSERMAN, 2014), MobileNet (HOWARD *et al.*, 2017), and Inception Resnet (SZEGEDY *et al.*, 2017) as feature extractors for the Fast R-CNN network. Figure 10 illustrates object detection results using the Fast R-CNN network combined with the Inception Resnet CNN for feature extraction. It demonstrates that the method can identify objects of various sizes with a high level of precision.

Figure 10 – R-CNN: Regions with CNN features.



Source: Huang *et al.* (2017)

2.3 Chapter Remarks

In this chapter, we have provided a concise exposition on the utilization of drones in precision farming. We've also introduced fundamental concepts within the realm of computer vision, along with the methodologies of Artificial Neural Networks and Convolutional Neural Networks, with a specific focus on image classification and object detection tasks. These concepts serve as vital background knowledge, preparing the foundation for the subsequent chapters where we delve deeper into the practical application of these technologies.

CHAPTER

3

RELATED WORK

Within the domain of computer vision and remote sensing, a body of research is dedicated to the detection and counting of animals in aerial imagery. These images can originate from various sources, including UAVs and satellites. While many studies concentrate on the essential tasks of animal detection and counting within single images or frames, another set of studies addresses the challenges of counting animals across multiple images, which remains an area that merits further exploration in the literature.

This chapter provides a comprehensive overview of pertinent research, delving into the landscape of related studies concerning the detection and counting of animals from aerial imagery. Section 3.1 examines works that primarily revolve around the detection and, to some extent, counting of animals in aerial images within single frames or images. It will spotlight studies that leverage Convolutional Neural Networks (CNNs) and other deep learning techniques for these purposes. On the other hand, section 3.2 places a specific focus on methodologies that tackle the intricate task of counting animals across multiple images. These approaches may involve the creation of image mosaics or methodologies designed to handle overlapping areas. This section addresses the significant challenges associated with counting animals in large, complex landscapes, requiring specialized methods distinct from those focused on single-image analysis.

3.1 Counting Animals From Single Images

In the domain of animal detection and counting from aerial imagery, a considerable portion of the related studies has predominantly focused on detecting animals or estimating counts in single images using detection or density estimation methods (XU *et al.*, 2020a; CHAMOSO *et al.*, 2014; BARBEDO *et al.*, 2019; LONGMORE *et al.*, 2017). This task presents several significant challenges, primarily revolving around the precise detection of animals within the image. These challenges encompass dealing with occlusion, shadows, background variations, and lighting changes, which can lead to potential false

negatives if not effectively addressed. Moreover, the detection systems, primarily relying on Convolutional Neural Networks (CNNs) for object detection, must accurately learn the intricate shapes of animals to prevent false positives, ensuring that objects on the ground are not misclassified as the animals to be detected. Despite these complexities, counting in this scenario becomes relatively straightforward once the animals are detected, as the final count is essentially determined by the number of successfully identified animals within a single image.

The use of drone images for animal monitoring and counting has gained popularity across different fields. It has been applied to wildlife conservation (GEDEON *et al.*, 2022; GEMERT *et al.*, 2015; KELLENBERGER; MARCOS; TUIA, 2018) and livestock management, including various species such as cattle (BARBEDO *et al.*, 2020; SHAO *et al.*, 2020), sheep (SARWAR *et al.*, 2021; SARWAR, 2022), goats (VAYSSADE; ARQUET; BONNEAU, 2019), and others.

Longmore *et al.* (2017) used astronomical software and images obtained by infrared sensors, installed in an UAV, to detect and monitor animals. The object detector identifies the "hot" points in an image, considered points of interest. Then, a point size criterion is applied to classify the object. The authors propose as future work the use of the method for systematic monitoring of parks and large fauna, such as the monitoring of rhinos.

In the context of cattle classification, the authors in (BARBEDO *et al.*, 2019) compared the accuracy of 15 CNNs performing classification of images of cattle captured by drones. The evaluation involved the classification of blocks of images cut off each photograph, where a block may or may not contain cattle. The experiments were conducted to evaluate the performance of the networks simulating different ground sample distances (GSD), that is, the altitude at which the photograph is taken, in addition to changes in lighting and shadows, which vary according to the season or time of the day that photographs are taken. The authors concluded that CNNs are robust to almost all variations of lighting, with the exception of rare situations where there are severe specular reflections, making the contrast between the animal and background very subtle, and detection more difficult. Regarding the GSD variation, the reported experiments used images taken at $30m$ high, which were also downsampled to simulate the resolution of images that would have been obtained at $60m$ and $120m$ high. For most CNNs, the images corresponding to $60m$ altitude resulted in better accuracy than the other images. CNNs NASnet Large (ZOPH *et al.*, 2018) and Inception ResNet V2 (SZEGEDY *et al.*, 2017) obtained acceptable results for all altitudes though, even for $120m$. This is important as altitude is the main mechanism for minimizing the amount of photos required to cover an entire area/pasture. However, this work assumes that the region to be classified (as cattle or not) has been previously cropped, which makes it not practical for counting multiple animals in large images.

In the context of cattle count, Xu *et al.* (2020a) used semantic classification and segmentation of animals with the objective of enhancing their biometric monitoring and the possibility of distinguishing them individually for tracking or studying behavior. The experiments reported accuracy close to the ground truth when counting animals in individual images, although counting scenarios that require traversal of large pastures, multiple images, and duplicate counting of animals have not been addressed. In addition, since the segmentation method related to annotation, training and detection is more complex than the traditional bounding box method, the images must be obtained at low altitude (the authors use images obtained from $6m$ to $25m$ high) and, preferably, in oblique angle, for better identification of shapes and correct semantic segmentation of animals. These requirements are particularly restrictive for applications involving large areas and possibly sparse animals.

The detection of animals from a bounding box is addressed in (CHAMOSO *et al.*, 2014), where the authors describe a system to monitor large areas of livestock production using drones. The system consists of controlling the drone and identifying animals using a communication mechanism between a laptop and the drone. Object recognition is performed in real-time through the analysis of individual frames received from the camera. The system then uses a sliding window approach to sequentially analyze small portions of the image. Each fragment of the image is evaluated by a trained CNN and an activation function, which returns a probability value that the window belongs to the target class "cattle". The probabilities of these windows are then aggregated on a 2D probability grid, which will indicate the positions where the animals were found, according to a predefined threshold. Although the authors report promising results for counting animals in an image frame, the work does not propose a strategy to estimate the number of animals in the entire pasture.

Weber *et al.* (2023) proposed the use of CNN models like YOLOv4 (BOCHKOVSKIY; WANG; LIAO, 2020) and YOLOv5 (JOCHER, 2020) for cattle detection and counting. The results showed that the YOLOv5-m model could achieve high values of precision (0.945) and recall (0.979), indicating the viability of using CNNs for certain cattle counting tasks. However, the study acknowledges limitations such as the focus on a single specific type of cattle (Nelore) and, mainly, the requirement to count from single images.

3.2 Counting Animals From Multiple Images

While the previous section delved into the challenges of detecting and counting animals in single images, this section shifts its focus to the intricate task of counting animals across multiple images or frames. The distinction lies in the need to track animals between these images or remove duplicates in overlapping areas, considering that animals may move between frames. This adds a layer of complexity to the counting process,

transforming it into a considerable challenge.

In contrast to the extensive body of work dedicated to counting within single images, the realm of counting from multiple images remains relatively less explored in the literature. This scarcity of existing research accentuates the wide-open field for investigations. The need for counting from multiple images typically arises when the area to be covered cannot be effectively captured within the confines of a single photograph.

A common technique employed in some existing methods involves creating a mosaic of a collection of images to obtain an overall view of the area (CHEN *et al.*, 2023; SHAO *et al.*, 2020). However, this approach presents challenges due to potential movements of animals between images, which can lead to distortions such as full or partial animal replications and/or omissions in the mosaic, with potentially detrimental impact on both the detection as well as the counting tasks. In this context, Chen *et al.* (2023) presented a method for automatic counting of *cranes* (Grus grus bird) using UAVs equipped with thermal cameras for night monitoring and visible light (RGB) cameras for daytime observations. The cranes assemble in a large communal roost at night, facilitating generation of an image mosaic for the counting process, once they are relatively static and grouped together. Image analysis and computer vision algorithms were developed to identify and count individual cranes, achieving an overall accuracy of 91.47% for nighttime thermal images and 94.51% for daytime RGB images. Although the algorithms were specifically designed for crane counting, they have potential applicability to other animal management scenarios. However, it should be noted that this method's suitability for cattle counting is limited since cattle behavior is often dynamic and the areas to cover can be much larger.

Regarding cattle counting in scenarios exceeding the coverage of a single photograph, Barbedo *et al.* (2020) proposed a multi-module method to address this challenge without using a mosaic approach. The method comprises four modules: a CNN for region of interest identification, color manipulation to segment white cattle from the background, mathematical morphology to separate animal clusters and remove spurious objects, and image matching to handle overlaps. This approach achieved a precision of approximately 93.3% for cattle counting across all datasets considered, which encompass variations in cattle density, backgrounds, and illumination. However, it lacks scenarios featuring animals of different colors within the same pasture, as the method is tailored for collections with animals of a single color. Additionally, the method employs a simplistic approach for duplicate removal, where animals from overlapping areas of the image are removed, without considering the possibility of animals moving in or out of the overlapping area between adjacent images.

Counting cattle in large areas is also addressed in (SHAO *et al.*, 2020). The authors propose a pipeline for detecting and counting animals in order to avoid duplicate counts. To that end, they use the YOLOv2 CNN method (REDMON; FARHADI, 2017)

to detect animals in individual images, and the results are combined by the use of the three-dimensional reconstruction model Structure from Motion (SfM) (WU *et al.*, 2011), where the position of animals in 2D images are projected and calculated on a merged 3D surface. The proposed method was tested in areas ranging from 1.5ha to 5ha and obtained an F-measure of 0.952 for the detection task in the training dataset. When tested on unseen images, the F-measure was 0.713. For the counting task, the method achieved results close to the ground truth, however, it is worth noticing that it depends on the SfM model, which is highly computationally demanding and may become infeasible depending on the number of images, which in turn depends on the size of the area to be covered.

A study by Sarwar (2022) employed drone-recorded videos to count sheep in paddocks. The approach involved multiple object tracking (MOT) and the Kalman filter (BROWN; HWANG *et al.*, 1992) to predict object trajectories. The Hungarian algorithm (MILLER; STONE; COX, 1997) was used to link trajectories to objects based on predicted states and object detector estimations. The drone's coverage of the entire paddock width allowed for straightforward removal of duplicates, as animals did not reappear once they left the frame. However, in our work, which deals with larger pasture areas that require more complex flight plans, the potential for cattle to reappear after leaving a frame presents unique challenges for the task of duplicate removal.

It is noteworthy that authors of most studies in this area do not provide access to their models or methods for reproduction of results and comparisons. Additionally, only (SHAO *et al.*, 2020) offers a publicly available dataset of images, which will be included as part of a larger collection of datasets to be used in the experiments conducted in this thesis.

3.3 Chapter Remarks

This chapter systematically explored the landscape of animal detection and counting in aerial imagery, encompassing both single-image and multi-image scenarios. The initial section, "Counting Animals from Single Images", shed light on extensive research dedicated to precise animal detection within individual images, employing CNNs and deep learning.

In contrast, the subsequent section, "Counting Animals from Multiple Images", emphasized the relatively limited research addressing the complexities of animal counting across multiple images. This included tracking and duplicate removal in overlapping areas. These insights set the stage for our comprehensive pipeline, detailed in the following chapters.

The next chapter introduces a novel pipeline for cattle detection and counting across large areas using multiple drone-obtained images. This approach incorporates the step of automatically cattle detection, and encompasses a graph-based algorithm to remove

duplicates, addressing one of the significant challenges associated with counting animals in large pastures.

CHAPTER

4

LIVESTOCK DETECTION AND COUNTING IN THE WILD

In the preceding chapters, we explored the core principles of Convolutional Neural Networks (CNNs) and introduced methodologies employing drone-captured images for ground mapping (Chapter 2). We also conducted a comprehensive survey of relevant literature in the field of animal detection and counting (Chapter 3). Leveraging this foundational understanding, the current chapter takes a substantial leap forward.

In this chapter, we detail the contributions to advance the research on automatic cattle counting from aerial images by developing methods that can be applied to large areas at a low computational cost, which were published in (SOARES *et al.*, 2021). As main contributions we propose an efficient graph-based method for detecting and counting animals from drone images, as well as a new benchmark image collection that covers larger areas than those currently available in the literature. The proposed method is based on geolocation parameters, which contrasts with the one proposed in (SHAO *et al.*, 2020), based on the SfM three-dimensional reconstruction model (WU *et al.*, 2011). We obtain a counting accuracy close to that obtained with the use of SfM but with a significantly lower computational cost that makes the method applicable to larger areas and image collections.

4.1 Materials and Methods

In this section, we outline the materials and methods used in this stage of the research. We detail the devices and software used for cattle image acquisition and introduce the datasets for model pretraining and cattle counting evaluation. Construction of this datasets was motivated by the scarcity of publicly available livestock aerial images. Lastly, we discuss the CNN configuration and training methodology, emphasizing key strategies and considerations.

Figure 11 – Drone DJI Mavic Pro

Source: Elaborated by the author.

4.1.1 Devices and Software

Below we list the materials used in this stage of work, in particular the drone and the software used to plan the flights, as well as the computer used to train the models and perform the experiments.

- **Image capture equipment:** DJI four-propeller drone (quadcopter), Mavic Pro model (Figure 11), with the following characteristics:
  - Camera:
    - Sensor: CMOS 1/2.3", 12.35 megapixels
    - Lens: 28mm
    - Maximum aperture: f 2.2
    - ISO range (photo): 100-1600
    - Maximum image size: 4000 × 3000 pixels
  - Battery: 3830 mAh (≈ 27 minutes of flight)
  - Controller: maximum range of 7 km
  - Maximum speed in flight plans: 36 km/h
- **Software:** for planning and executing flights we use the application "DroneDeploy - Mapping for DJI"[1], on an Apple iPhone 5s mobile phone.
- **Computer system:** to pre-process, develop and execute experiments for this project, a DELL XPS-8700 desktop computer was used, with an Intel Core i7-4790 3.60GHz×8 processor, with 16GB of RAM and 4GB Nvidia GeForce GTX-745 GPU .

[1] Information available at https://www.dronedeploy.com

Table 1 – Description of the location and characteristics of the farms where the images were obtained, as well as the times and altitudes at which flights were performed.

Farm	Location	Time	Altitude (m)	Animal color	Soil type
Água Boa	Rochedo, MS/BR 20^{o}07'22.3"S 54^{o}42'53.6"W	Morning Noon Afternoon	80, 90, 110, 120	Various	Red soil, Green Pasture
Bela Vista	Ribas do Rio Pardo, MS/BR 20^{o}40'02.1"S 53^{o}31'57.0"W	Noon Afternoon	90, 120	White	Sandy soil, Dry Pasture
Imec	Ribas do Rio Pardo, MS/BR 20^{o}40'53.0"S 53^{o}31'20.3"W	Morning Noon Afternoon	90, 120	White	Sandy soil, Dry Pasture
Lagoa	Ribas do Rio Pardo, MS/BR 20^{o}36'31.8"S 53^{o}09'22.5"W	Morning	90	White	Sandy soil, Dry Pasture
Primavera	Terenos, MS/BR 20^{o}25'14.4"S 55^{o}07'22.7"W	Morning	90	Various	Red soil, Dry Pasture
Duas Anas	Campinas, SP/BR 22^{o}48'53.1"S 47^{o}00'07.4"W	Morning Afternoon	90, 100, 120	Various	Red soil, Green Pasture

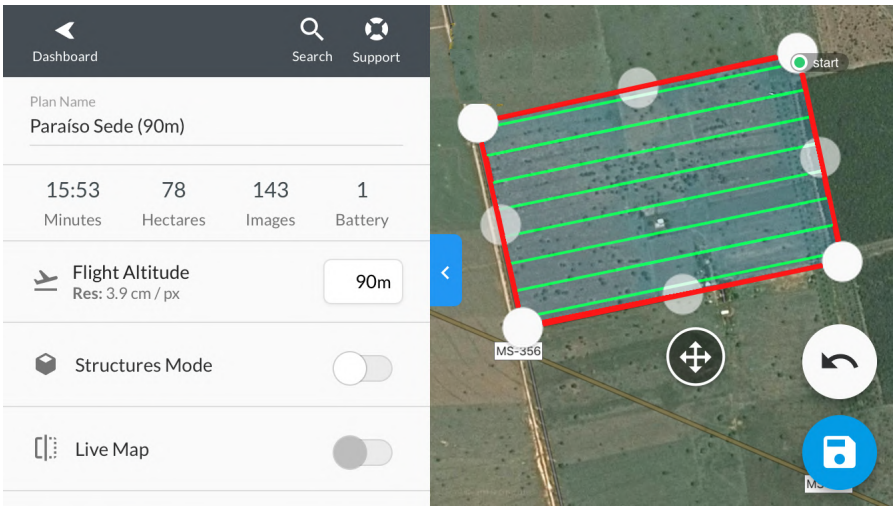
4.1.2 Novel image collection for training

In light of the current limitation of publicly available databases, a novel database of georeferenced images was acquired, intended to sample different types of soil, types of cattle (in terms of their visual aspects), altitudes, and times of the day, as shown in Table 1. The goal was to comprise image samples where animals are represented in different sizes, colors and backgrounds, in addition to scenarios with a variety of lighting and shadows, depending on the time of the day the images were taken. For which, authorization was obtained from farmers to fly over and capture images of their properties

If training was otherwise carried out, say, using only images of black animals on a green pasture, the classification algorithm would tend to learn that a cattle is just a black animal on a green background; in this case, when an image of a white cattle is given as input, the trained model would hardly classify it correctly. For example, in (SHAO *et al.*, 2020), in which a neural network was trained using a database of images from a single pasture, the accuracy of detection and counting of animals dropped considerably when performed on a new database of images not yet seen by the network, even though the images are from the very same pasture, but taken at different times (with different pasture coloring) and altitude. When flexible models such as neural networks are trained using a diversified dataset instead, they are driven to learn animal characteristics that go beyond the obvious (e.g. colours) and the result is a model with greater generalization capacity.

Table 1 provides information about the collection of images from 6 farms, at altitudes ranging from $90m$ to $120m$, and acquisition at different times of the day. In three of the farms there were only Nellore cattle, whose coat color is white. These farms have drier and sandy soil. In the other three farms, where soil is more reddish and the pasture is greener, the cattle is diverse, where there were animals of white, black, brown, red and spotted/piebald color.

Figure 12 – Flight plan creation screen (DroneDeploy application). The red line frame on the map is the area to be photographed. The route to be taken by the drone is represented by the green line. The left panel displays the flight settings and descriptions.



Source: Elaborated by the author.

The planning of autonomous flights to photograph the farms was made using the DroneDeploy application. Figure 12 shows the flight plan creation screen for the application. The left panel displays the flight description and the available settings. The example involves a flight plan over the Primavera farm. The pasture is marked on the map by the red frame with a total area of 78ha. The green line represents the route to be taken by the drone to photograph the entire area. In this case, the flight will be carried out at an altitude of $90m$, an estimated duration of 15min53s, and it will capture 143 images. Considering the resolution of the drone's camera used, each pixel of the images will represent 3.9cm on the ground. The flight will be completed without the need for a battery change. It is also possible to select the rate of overlap between images, with 30% being the minimum value. While overlays of 80% or higher are required for the purpose of creating maps and 3D surfaces, in animal counting we will use the smallest possible overlay, in order to minimize the amount of images and maximize the flight area with a single battery. In the example mentioned, the frontal and lateral overlap was set at 30%. All flight plans were performed in July 2017 and April 2018.

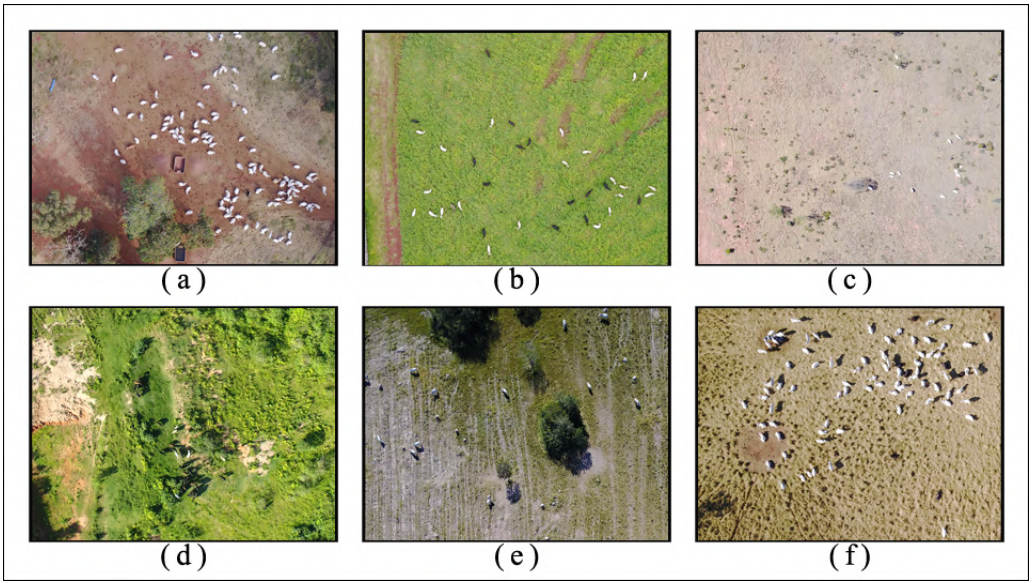
Table 2 describes the pastures imaged at each farm, including their area, the number of animals as declared by the farmers, as well as the number of photos recorded. Our database, which will be used for training, has a total of 5,058 images. As the flight plan covers the entire pasture region, a significant part of the images do not contain animals. This represents natural scenarios, in the wild, and is fundamental for training and validating methods with respect to false positives under realistic conditions.

Figure 13 exemplifies the variety of scenarios in the images collected: Figures 13(a)

Table 2 – Number and average size of pastures, approximate total number of animals, and number of photographs taken at each farm.

Farm	**# Pastures**	**AVG Area (ha)**	**# Animals**	**# Photos**
Água Boa	5	71	≈ 1000	3,178
Bela Vista	1	56	≈ 1000	106
Imec	2	28	≈ 250	268
Lagoa	1	90	≈ 300	46
Primavera	4	76	≈ 300	543
Duas Anas	1	50	≈ 50	917
			TOTAL:	5,058

Figure 13 – Example of the diversity of photos collected in this work: (a) and (b) were collected at Água Boa farm and taken at an altitude of 90m. (c), (d), (e) and (f) were collected at Bela Vista, Duas Anas, Imec and Primavera farms, at altitudes of 120m, 100m, 120m and 90m, respectively.



Source: Elaborated by the author.

and 13(b) show images captured at an altitude of 90m from the same property at 08:07am and 2:06pm, respectively. In the former, most of the cattle are white and are lying on the ground. In the latter, black, white, brown and gray animals are standing on a green pasture. The image shown in Figure 13(c) was captured at 9:14am, containing white animals on dry pasture. In Figure 13(d) the cattle have a diverse color and several animals are under the shade of a tree, which makes their identification more difficult. In Figure 13(e), the image was captured at 1:36pm at 120m altitude, and it is possible to see light colored stones, typical of places with sandy terrain, which can be confused with white animals. Finally, in Figure 13(f), captured from Primavera farm at 8:46am, most animals are white, while some have dark colors. The pasture is predominantly dry, with a mixture of animals lying and standing.

4.1.3 Datasets for Cattle Counting

This section provides an overview of the datasets employed for cattle counting tasks. The dataset compilation comprises 670 images from 22 flight sessions, sourced from the work of (SHAO *et al.*, 2020). Additionally, 667 new images were meticulously gathered from 4 flight sessions (referred to hereafter as "*BR_set*") and making them publicly available for the research community.

Table 3 offers a comprehensive breakdown of these datasets, including their flight session name, image count, cattle count, coverage area, and class. The 22 flight sections contributed by (SHAO *et al.*, 2020) contain a ground truth count of 218 animals in total. These sections were classified into two categories based on cattle movement within the images:

- **Motionless:** Sections where all cattle moved less than their body length during the flight plan.
- **Moving:** Sections in which some cattle moved more than their body length.

The *BR_set* collection, assembled for this research, consists of 4 flight sections. Due to the expansive nature of these sections, precise classification into "motionless" and "moving" categories was challenging. Therefore, such categorization was not applied. Table 3 provides details of this collection, which encompasses a total of 200 cattle and covers an average area of approximately 60 hectares. Notably, this scale represents a substantial increase compared to the datasets from (SHAO *et al.*, 2020), which typically covered areas ranging from 1.5 to 5 hectares.

The images constituting the *BR_set* were acquired from three of the farms described in Table 1: "Água Boa", "Primavera" and "Duas Anas". It is important to note that *none of the images* in the training datasets detailed in Section 4.1.2 are present in these datasets used for cattle counting, i.e., there is no overlap between these two dataset collections.

4.1.4 Labeling

The algorithm for detecting animals in the images used in this work is based on models trained from annotated data. The LabelImg tool (TZUTALIN, 2015) was used for this task, which consists of demarcating animals by boxes, then setting a label for that box (class). Figure 14 illustrates the LabelImg screen, where the animals in the image were properly demarcated and labeled with the class "cattle". The LabelImg tool then generates an `xml` file with the positions and labels of each box (animal) in the image. All images were labeled by a single person.

Table 3 – Datasets used in this work to evaluate the proposed animal counting method.

Datasets	**Flight Section**	**No. of Images**	**No. of Cattle**	**Area (ha)**	**Class**
From (SHAO *et al.*, 2020)	A	32	4	1.5 to 5.0	Motionless
	B	45	7	1.5 to 5.0	Motionless
	C	32	12	1.5 to 5.0	Moving
	D	40	5	1.5 to 5.0	Moving
	E	184	5	1.5 to 5.0	Moving
	F-1	20	8	1.5 to 5.0	Motionless
	F-2	20	20	1.5 to 5.0	Motionless
	F-3	20	24	1.5 to 5.0	Moving
	F-4	20	20	1.5 to 5.0	Motionless
	F-5	20	14	1.5 to 5.0	Motionless
	F-6	20	0	1.5 to 5.0	Motionless
	F-7	20	0	1.5 to 5.0	Motionless
	F-8	20	0	1.5 to 5.0	Motionless
	G-1	20	2	1.5 to 5.0	Motionless
	G-2	20	17	1.5 to 5.0	Motionless
	G-3	20	25	1.5 to 5.0	Motionless
	G-4	20	25	1.5 to 5.0	Motionless
	G-5	20	19	1.5 to 5.0	Moving
	G-6	20	5	1.5 to 5.0	Motionless
	G-7	20	0	1.5 to 5.0	Motionless
	G-8	20	0	1.5 to 5.0	Motionless
	Dataset 2	14	6	1.5 to 5.0	Moving
BR_set	SD_PV_90	143	5	78	-
	2A_90	325	32	56	-
	P1_AB_120	61	63	50	-
	PD_AB_90	138	100	54	-

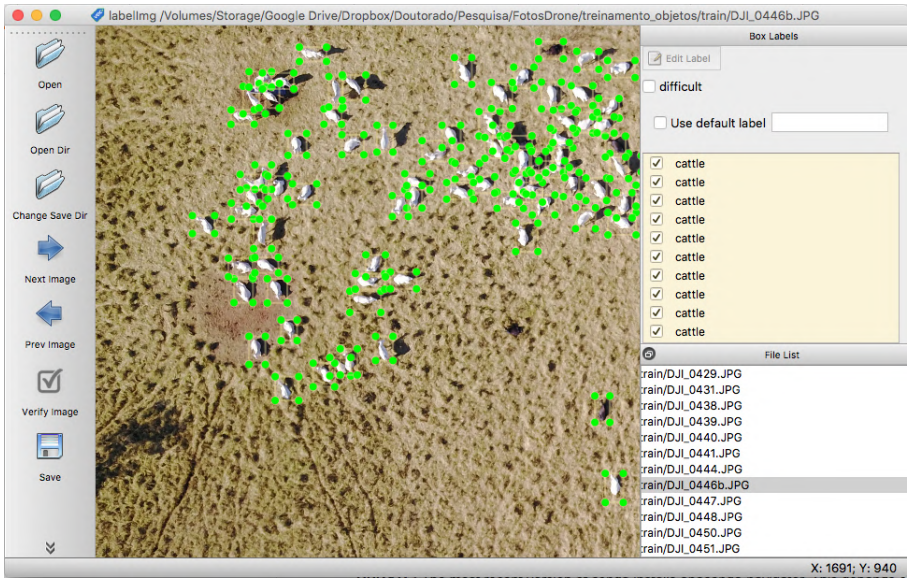
4.1.5 Training the Convolutional Neural Network (CNN)

The detection of animals in the images was performed using the deep neural network Faster RCNN Inception Resnet V2 (HUANG *et al.*, 2017), which is an object detector of the type *Faster R-CNN* that uses a mixture of units from the *Inception* (SZEGEDY *et al.*, 2015a) and *Resnet* (HE *et al.*, 2016) networks. We use a pre-trained model[2] learnt from the MS-COCO (Common Objects in Context)[3] dataset. The choice is justified by recent evidence showing that the better a model is in the database in which it was pre-trained, the better it tends to be when learning is transferred to other databases (KORNBLITH; SHLENS; LE, 2019). In this context, the Faster RCNN Inception Resnet V2 was one of the networks with the best accuracy for high-altitude cattle detection in recent experiments, reported in (BARBEDO *et al.*, 2019).

[2] Available at: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

[3] http://cocodataset.org

Figure 14 – Illustration of the LabelImg tool (TZUTALIN, 2015), used to manually demarcate and label all the animals in the image.



Source: Elaborated by the author.

We use Transfer Learning, which consists of initializing parameters with values obtained by training in a source database, and then adjusting only the layers of interest through a second training process commonly called fine-tuning. In this chapter, the output layer was redefined in order to address the cattle detection problem only, with its parameters randomly initialized.

To perform the training and test of the model, the 5,058 images described in Table 2 were used according to the following procedure:

1. **Split images into training and test sets:** the 5,058 labeled images are used in a 10-fold cross validation setup.

2. **Definition of parameters and training:** The API *Object Detection*[4] from TensorFlow v. 1.4 was used to train the final layer of the neural network. For this training, the following parameters were used:

   - **Mini-Batch:** 16;
   - **Epochs:** 100. An epoch is reached after training goes through all the images in the set. With batch size equal 16 and a 4,552 image training data set, it took 285 iterations to reach an epoch;
   - **Model selection for testing**: the last model, resulting from the last iteration, is used to perform the test;

[4] https://github.com/tensorflow/models/tree/master/research/object_detection

- **Optimization algorithm:** Gradient Descent with Momentum (SUTSKEVER *et al.*, 2013) of 0.9 and fixed learning rate of 0.0003.

Training was undertaken using the Desktop DELL XPS described in Section 4.1 and took approximately 27 hours.

4.2 Counting Method

The proposed counting method has as input the set of images that cover the entire pasture. The total number of animals with the list of their respective geographical coordinates is the output. To achieve this, 3 steps are performed, which will be detailed in the next sections:

- Computing the projections of the image vertices on the ground;
- Performing cattle detection in the images;
- Identifying/removing duplicates and counting.

4.2.1 Computing the Projections

For the counting method to be performed, it is necessary that the images contain metadata related to the GPS position and altitude of the drone in relation to the ground, as well as the position and technical information of the camera and lens used. Most commercial drones record this data on the images. This information is necessary to estimate the GPS coordinates of each detected animal and filter out duplicates using these coordinates. Table 4 describes the metadata available in drone's images that will be necessary to estimate the geolocation of the animals found in the images. In this work, all the images used were photographed with the camera orthogonal to the ground, that is, with the camera at a 90^o angle to the drone, as know as vertical photography (WOLF; DEWITT; WILKINSON, 2013). All equations for calculating the geolocation of animals have been adapted from the calculations proposed in Johnston (2006) and from the vertical photographic geometry explained in Wolf, DeWitt and Wilkinson (2013), to simplify and adjust the computation to the available metadata.

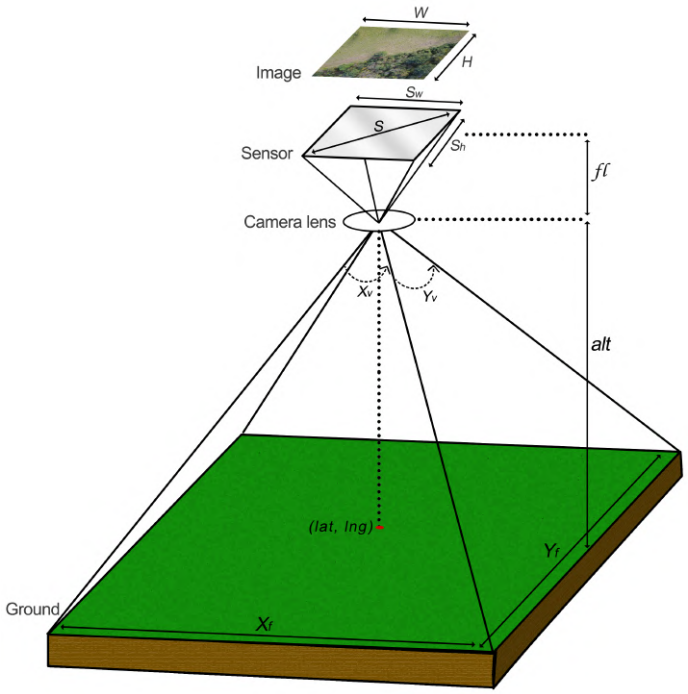
Figure 15 illustrates how the camera's field of view is calculated from a drone, depending on the altitude, focal length and sensor size. From the metadata obtained in the images, shown in Table 4, the diagonal size, S, of the camera sensor can be calculated according to Equation 4.1.

$$S = \frac{43.2666}{\frac{f_{35}}{fl}} \tag{4.1}$$

Table 4 – Description of the metadata needed to estimate the geolocation of the animals located in the images.

Metadata	**Abbr.**	**Description**
Latitude	lat	Drone latitude at the time of photography
Longitude	lng	Drone longitude at the time of photography
RelativeAltitude	alt	Relative altitude of the drone to the ground at the takeoff point
FlightYawDegree	yaw	Yaw angle of the drone, which refers to the flight directions
FocalLength	fl	Focal length of the lens
FocalIn35mm	f_{35}	Used to calculate the size of the camera's capture sensor
ImageWidth	W	Image width in pixels
ImageHeight	H	Image height in pixels

Figure 15 – Camera's field of view computation as a function of altitude (alt), focal length (fl) and sensor size (S).



Source: Elaborated by the author.

Provided that S is the diagonal size of the sensor, the height (S_h) and width (S_w) of the sensor can be obtained by the ratio of height (H) and width (W) of the image produced by the camera, as shown in Equation 4.2.

$$\begin{aligned} r &= \frac{\sqrt{H^2 \cdot W^2}}{S} \\ S_h &= H \cdot r \\ S_w &= W \cdot r \end{aligned} \tag{4.2}$$

Provided that the camera sensor has size $S_h \times S_w$, it is possible to calculate the angle of view for the two dimensions (X_v and Y_v) of the image, as shown in Equation 4.3.

$$\begin{aligned} X_v &= 2 \cdot \arctan \frac{S_w}{2 \cdot fl} \\ Y_v &= 2 \cdot \arctan \frac{S_h}{2 \cdot fl} \end{aligned} \tag{4.3}$$

Once the viewing angle has been calculated and given the relative altitude of the flight (alt), it is possible to calculate the area (in meters) in the camera's field of view, as shown in Equation 4.4. That is, a photograph taken at an altitude of alt meters, will result in an image that illustrates an area of $X_f \times Y_f$ m^2.

$$\begin{aligned} X_f &= alt \cdot \tan(X_v) \\ Y_f &= alt \cdot \tan(Y_v) \end{aligned} \tag{4.4}$$

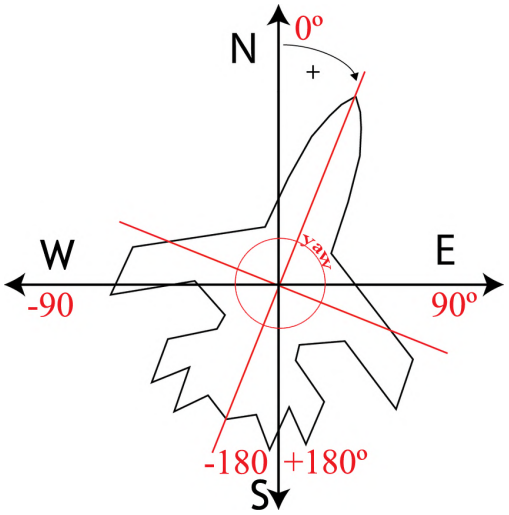
Since the images are obtained with the camera pointed at the ground, at an angle of 90^o, the latitude (lat) and longitude (lng) metadata represent the location of the central point of the image, which is the drone location. To estimate the geolocation of other points in the same image, it is necessary to consider the orientation of the image in relation to the cardinal points. To this, we will use the flight angle (YAW), obtained from the image metadata according to the values illustrated in Figure 16.

The YAW angle has a value of 0 when the drone points to the north, +90 to the east, −90 to the west and ±180 to the south. To facilitate the calculation of the location of the geographical coordinates of the animals within the images, the coordinates of the 4 vertices of the image, called *projections*, are computed. To that end, we first normalize the YAW angle within the interval $[0, 360]$, according to Equation 4.5.

$$yaw_N = \begin{cases} 360 - |yaw|\,, & \text{if } yaw < 0 \\ yaw, & \text{otherwise} \end{cases} \tag{4.5}$$

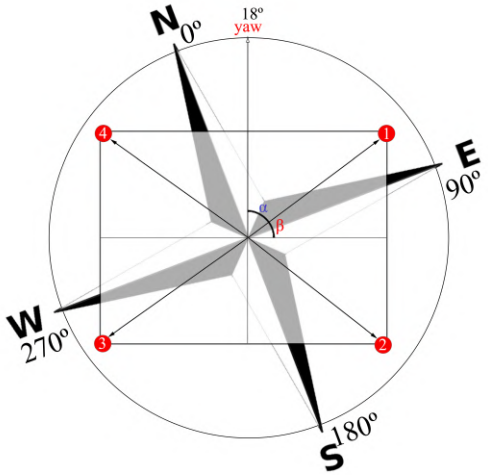
To calculate the coordinates of each vertex from the coordinates of the center of the image, we first need to obtain the yaw angle associated with each vertex. Figure 17 illustrates a scenario in which the image has an angle $yaw_N = 18^o$. In this case, the angle associated with vertex number 1 (top right) can be computed as $yaw_1 = yaw_N + \alpha$,

Figure 16 – YAW angle values obtained in the image metadata according to the direction in which the drone is flying.



Source: Elaborated by the author.

Figure 17 – Example where the *yaw* angle read from the metadata is 18^o. To calculate the geolocation of the 4 vertices of the image (projections) it is necessary to calculate the *yaw* angle of the direction of each vertex. To achieve this, displacements of α^o and β^o are applied to the initial *yaw* angle.



Source: Elaborated by the author.

whereas vertex 2's angle (bottom right) is given by $yaw_2 = yaw_1 + 2 \cdot \beta$, and so on, as shown in Equation 4.6. The operation (term) $\mod 360$ is used to set the reference back to zero (0^o) whenever the sum of angles and displacements exceeds 360^o.

$$\begin{aligned} yaw_1 &= (yaw_N + \alpha) \mod 360 \\ yaw_2 &= (yaw_1 + 2 \cdot \beta) \mod 360 \\ yaw_3 &= (yaw_2 + 2 \cdot \alpha) \mod 360 \\ yaw_4 &= (yaw_3 + 2 \cdot \beta) \mod 360 \end{aligned} \tag{4.6}$$

Once the vertex angles have been computed, the latitude and longitude of each vertex must be computed. The distance in meters from the center of the image to each vertex is calculated as follows:

$$D_v = \sqrt{\left(\frac{X_f}{2}\right)^2 + \left(\frac{Y_f}{2}\right)^2} \tag{4.7}$$

Since latitude and longitude are represented by decimal degrees and the distance from the center to the vertices is in meters, it is necessary to convert the distance to decimal degrees. For latitude (lat), each arc-minute corresponds to $1,852m$, which is equivalent to a nautical mile. As for longitude (lng), each arc-minute is equivalent to $\cos(lat) \cdot 1,852m$, due to the narrowing of the terrestrial sphere towards the poles. The calculation of lat and lng for each vertex is done as shown in Equation 4.8, where x in lat_x, lng_x and yaw_x refer to each one of the vertices (1, 2, 3 and 4). We first obtain a conversion factor:

$$\Omega = 1 \div 60 \div 1,852,$$

and then compute:

$$\begin{aligned} lat_x &= lat + (D_v \cdot \Omega) \cdot \cos(yaw_x) \\ lng_x &= lng + (D_v \cdot \Omega \div \cos(lat)) \cdot \sin(yaw_x) \end{aligned} \tag{4.8}$$

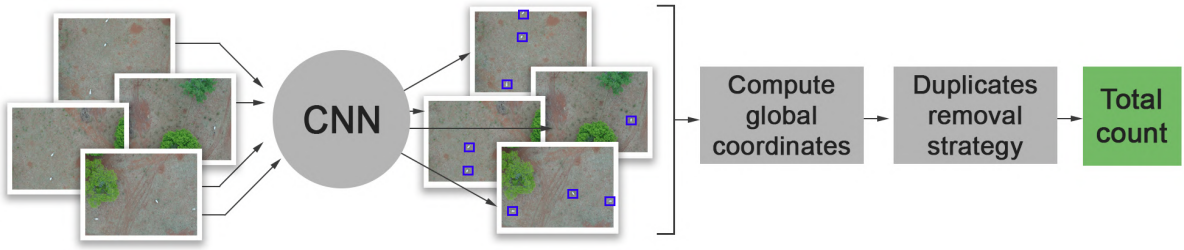
We than calculate the coordinates of any point within the image using a simple cross-multiplication. In the next section, we will describe how the detection of cattle in the images is done.

4.2.2 Performing Cattle Detection

Once the projections of the image have been calculated, the Faster R-CNN network, described in Section 4.1.5, is then used to detect the cattle in the image. The network processes the image and returns a list with the coordinates (x_1, y_1, x_2, y_2) of the bounding box for each detected animal.

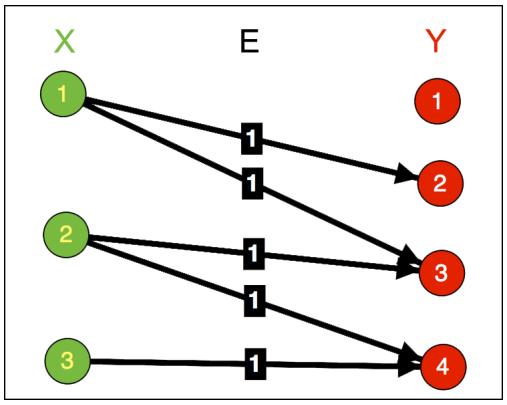
As illustrated in Figure 18, the set of images is provided as input to the CNN, which detects the cattle and returns the bounding boxes. Then, the coordinates of the bounding boxes in the images are used to compute the global coordinates of each animal, based on the image projections. Finally, duplicated animals are identified and removed before the total tally is finalized. Our proposed duplicate removal strategy is detailed in the next section.

Figure 18 – Pipeline for cattle counting from a set of images.



Source: Elaborated by the author.

Figure 19 – Example of a bipartite Graph G, where the vertices X are the animals $\in L_i$ and the vertices Y are the animals $\in L_t$. The edges E connect animals whose distance is less than the *threshold*.



Source: Elaborated by the author.

4.2.3 Cattle Counting and Removal of Duplicates

Due to the overlap between regions, an individual animal may appear in multiple images, so it is paramount to remove duplicates. Alongside with the detection of cattle, this represents the second most relevant step in the whole pipeline. In this chapter, we define a distance threshold to determine whether an animal should be considered the same or not when detected in different images with similar location. Algorithm 1 shows all steps to detect and remove duplicated cattle from all images, which are sequentially (chronologically) processed.

Algorithm 1 takes as input the list of images from a pasture where the cattle should be counted, as well as a distance *threshold* in meters, used to define the maximum distance within which animals located at similar coordinates but in different images are considered duplicates. Once all images have been provided as input, the algorithm outputs a final list L_t, with all valid animals and updated locations of detected animals in the entire image collection.

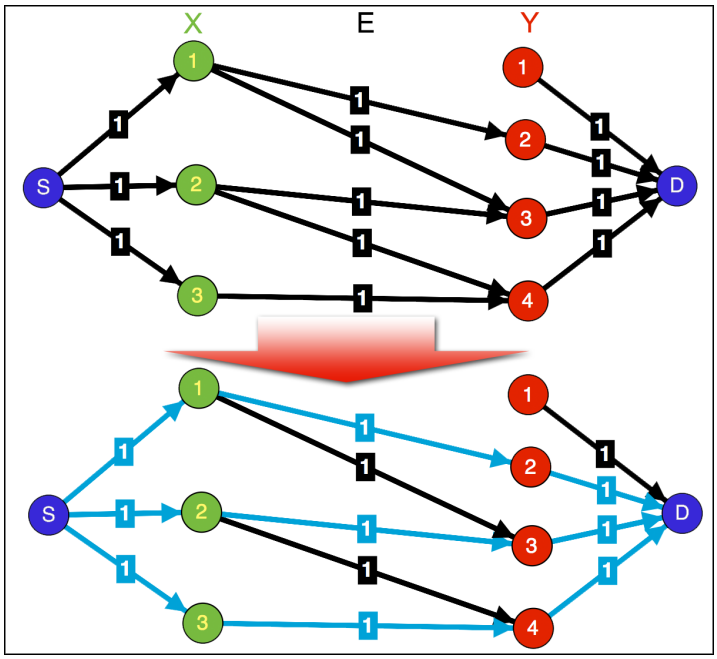
The algorithm starts with an empty L_t list. Then, for each image i, in chronological

Algorithm 1 Verify duplicated cattle

Input: *image_list*, *threshold*
Output: L_t

$L_t = \emptyset$
for each $i \in$ *image_list* in chronological order **do**
    L_i = get cattle coordinate list from image i
    $M = |L_i|$
    $N = |L_t|$
    if $M == 0$ **then**
        continue
    if $N == 0$ **then**
        add all cattle $\in$ L_i to L_t
    else
        $T_{(M,N)}$ = dist. matrix between all $x \in L_i$ and $y \in L_t$
        Create a bipartite graph $G = (X, Y, E)$, where E is the set of edges from X to Y with weight 1 when the distance $T_{(x,y)} \leq threshold$.
        Apply the maximum flow algorithm on G, generating a new graph $G' = (X, Y, E')$, where E' contains only edges selected as part of the maximum flow solution.
        for $x \in G'$ **do**
            if $deg(x) == 0$ **then**
                add $L_i(x)$ to L_t
                continue
            for $y \in G'$ **do**
                if $(x, y) \notin E'$ **then**
                    continue
                else
                    if $(T_{(x,y)} \leq T_{(x,y')} \; \forall \; y' \in G'$ where $deg(y') == 0)$ **then**
                        update location of $L_t(y)$ as $L_i(x)$
                    else
                        # A smaller distance $T_{(x,y')}$ where $y' \in G'$ and $deg(y') == 0$ was found:
                        replace (x, y) with (x, y') in E'
                        update location of $L_t(y')$ as $L_i(x)$
                    break
return L_t

Figure 20 – Example of solution for the maximum flow problem using Ford-Fulkerson algorithm. A vertex S (source) fully connected to the vertices X and a vertex D (destination) fully connected to the vertices Y are created, then the solution is the maximum flow from S to D (blue edges).

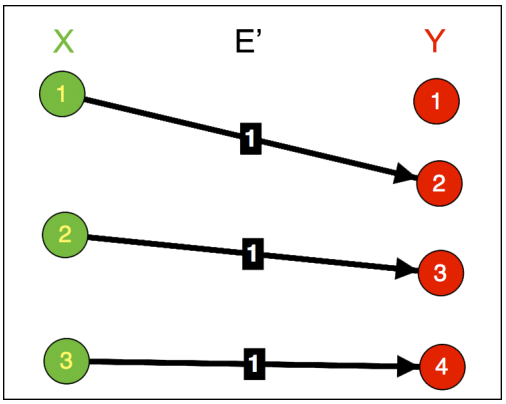


Source: Elaborated by the author.

order, a partial list L_i with the geolocation of detected animals in that image is created (line 3). If L_i is empty (line 6), which means no animals were detected in the current image, then the algorithm skips to the next image; otherwise, it checks whether L_t is empty (line 8). If so, either no image was processed so far, or no animals were detected in the previously processed image(s). In this case, we can initialize L_t with all animals in the current L_i. Once L_t is no longer empty ($N \neq 0$), the algorithm generates a distance matrix $T_{M,N}$ from all M animals $\in L_i$ to all N animals $\in L_t$ (line 11). From the distance matrix, in line 12 the algorithm creates a bipartite graph $G = (X, Y, E)$, where the sets of vertices X and Y refer to the cattle in L_i and L_t, respectively, and E is the set of edges generated from X to Y if the distance $T_{x,y}$ is less than the *threshold*, as illustrated in the Figure 19. All the edges have weight 1.

Selecting an edge $(x, y) \in E$ means that an animal $x \in X$ is deemed the same as $y \in Y$, as they are at a distance within the *threshold* apart. However, each animal in X can only be associated with at most one animal in Y and vice versa. Subject to this constraint, the objective is to select edges in order to match as many animals as possible. For this purpose, the Ford-Fulkerson (JR; FULKERSON, 1962) algorithm is employed to solve the maximum flow problem and obtain the maximum selection of edges of G, as illustrated in Figure 20, and described below.

Figure 21 – Example of a subgraph $G' \subset G$ (where G is the graph in Figure 19), where E' are the edges selected by the Ford-Fulkerson maximum flow solution, which connect the animals $X \in L_i$ considered to be the same as in $Y \in L_t$.



Source: Elaborated by the author.

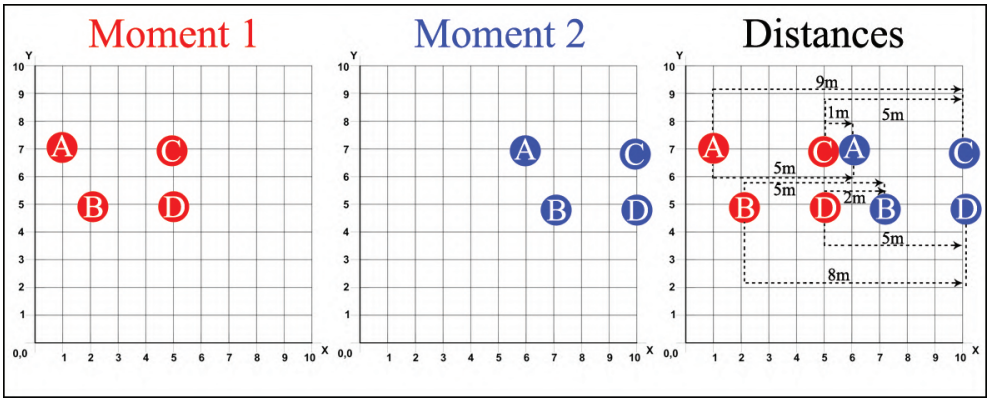
As illustrated in Figure 20, to solve the maximum flow problem we need to define vertices to be the source and the destination of the flow. For this, we create a vertex S (source) fully connected to the vertices X and a vertex D (destination) fully connected to the vertices Y. All edges are directed with unit weight. The Ford-Fulkerson algorithm considers that the flow cannot be greater than the capacity of each edge (weight) and, except for the source and destination vertices, the flow that enters a vertex must be equal to the flow that leaves the vertex. As all edges have weight 1 and each vertex in Y has only one outgoing edge connecting to D, each vertex in X can be connected to a maximum of 1 vertex in Y.

In Algorithm 1, line 13, the solution of the maximum flow problem is used to generate a new graph $G' = (X, Y, E')$, where E' has only the edges selected by the Ford-Fulkerson algorithm, as illustrated in Figure 21. The animals (vertices) $x \in X$ that have degree 0 in G', i.e. those that are not connected to any other vertex, represent cattle in L_i that do not correspond to any cattle in L_t (line 15), so these animals are added to L_t (line 16).

If a vertex $x \in X$ is adjacent (i.e. connected) to a vertex $y \in Y$ in G', it is necessary to check if there is some $y' \in Y$ with degree 0 in G' (i.e. without a paired vertex in X) whose distance $T_{(x,y')} < T_{(x,y)}$. If such a vertex exists, then the (x, y) edge in E' is replaced with (x, y') and the location of y' in L_t, $L_t(y')$, is updated with the location of the identified duplicate, $L_i(x)$ (lines 25 to 27). Otherwise, $L_t(y)$ is updated with $L_i(x)$ (lines 22 and 23).[5] This proximity check is needed because the Ford-Fulkerson algorithm considers all matches between animals to be the same (unit weight), so this step aims to prevent an x

[5] Notice that the location of duplicated cattle is updated due to the chronological order whereby images are processed: if an animal is moving, then it is necessary to keep the last location where it was detected.

Figure 22 – Example of displacement of four animals recorded in two moments. At moment 2 (blue) the 4 animals moved $5m$ to the right, when compared to moment 1 (red).



Source: Elaborated by the author.

to be matched to a y when there is a closer animal y' without a match.

In (SHAO *et al.*, 2020), the authors remove duplicates with the use of the Hungarian method (MILLER; STONE; COX, 1997), which addresses the assignment problem by seeking to minimize the weight (distances) between associations (matched animals). The authors apply a penalty defined by the value of the distance *threshold* to animals that are not paired to any other. However, we advocate that maximizing matches should be prioritized over minimizing distances. Figure 22 illustrates an example in which four animals (A, B, C and D) move $5m$ in the same direction between two images, when comparing their locations at moment 1 (1st image, in red) and moment 2 (2nd image, in blue). If we define the distance *threshold* for an animal to be considered the same in different images as $5m$, then all four animals at moment 1 should be matched to the corresponding animals at moment 2, because their displacement distance is within the *threshold*. However, in the Hungarian method, which seeks to minimize distances, the animals C and D at moment 1 (red) would be matched to animals A and B at moment 2 (blue), which are at distances of $1m$ and $2m$ from each other, respectively, while the other two animals would be unmatched, since they are at a distance above the *threshold*. Using our maximum flow method to maximize correspondences, all 4 animals would be correctly associated, since they all have a match at a distance within the defined *threshold*.

The Hungarian method complexity is $O(V^3)$, where V is the number of vertices on the smallest side of the bipartite graph, while the Ford-Fulkerson algorithm, used in our method, is bounded by $O(Ef)$, where E is the number of edges and f is the maximum flow in the graph (AHUJA; MAGNANTI; ORLIN, 1988).

The 3 major steps of our method (described above in this section) are performed for all input images in chronological order. After processing all images, L_t will contain

the geographical coordinates of all valid cattle, discarding duplicates. The next section describes the experiments carried out to study the behaviour of this method in practice.

4.3 Evaluation

This section describes the evaluation of the cattle counting method. We first evaluate the deep network for the cattle detection task. Then, we report experiments for cattle counting in whole areas, followed by a runtime analysis. Finally, we make available a novel benchmark dataset, where we propose two protocols for training and test, in order to contribute to the field of research in cattle counting in large areas. These experiments are detailed below.

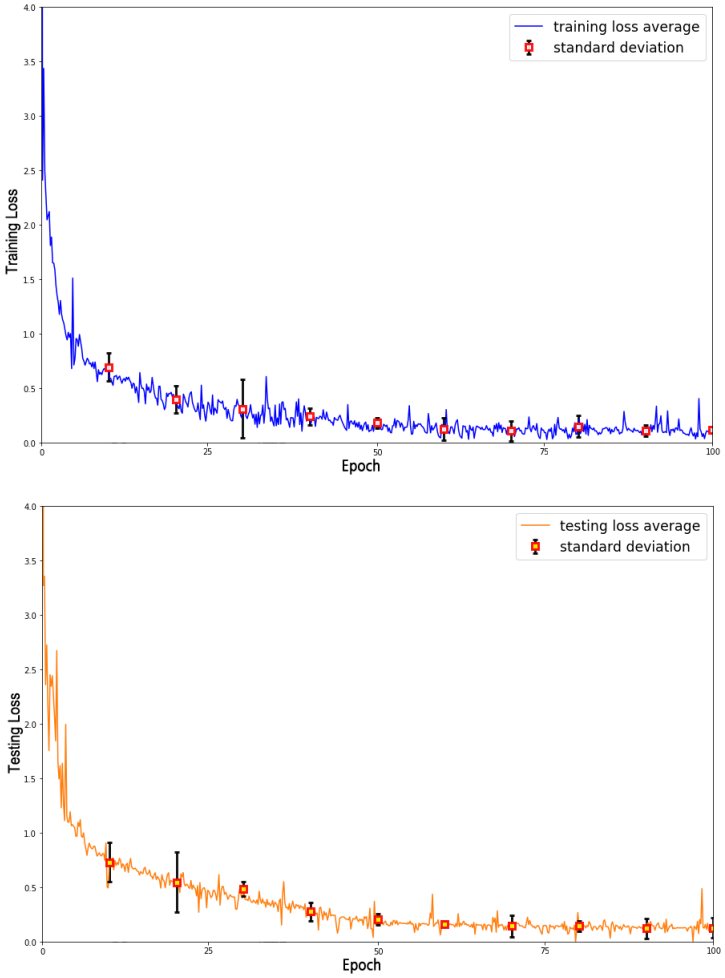
4.3.1 Cattle Detection

A critical step in the cattle counting method is detection. If there is no guarantee that the trained CNN model is capable of detecting cattle in the images, the accuracy of the counting method is seriously compromised. A 10-fold cross-validation was carried out to evaluate the deep network (Faster RCNN Inception Resnet V2), described in Section 4.1.5. The 5,058 labeled images described in Section 4.1.2 were divided into 10 subsets such that, at each cross-validated step, 9 subsets (90% of images) are used for training and 1 subset (10% of images) is used for testing. Performance on the 10 test subsets are aggregated as usual.

Figure 23 displays the average loss curves for training (blue) and testing (orange), respectively, after training was carried out during 100 epochs for each cross-validated training/test data split. In addition, the figure also displays the standard error around the mean value every 10 epochs. Note that in both graphs the curves have the same behavior, declining rapidly in the first 25 epochs (where reduction in bias is more prominent), continuing to decrease yet more slowly for approximately 25 epochs, and finally stabilizing after 50 training epochs. The low variance between the 10 experiments indicates that the model generalizes well within the dataset for all test subsets, and the absence of a test loss rebound suggests that no overfitting is taking place. For this reason, we just selected the model with the lowest training loss at the end of the 100 epochs to be used in our cattle detection and counting experiments, as described in the following sections. We will call this trained model Net_1.

In addition to cross-validation for model performance assessment, we also evaluate Net_1 using the $1,377$ new images described in Section 4.1.3, of which 670 are from two datasets provided by (SHAO *et al.*, 2020); and the remaining 667 new images from the "*BR_set*" dataset. It is worth noticing that none of these $1,337$ images were used in the CNN's training process.

Figure 23 – Average training (blue) and testing (orange) loss curves and standard deviation (reported every 10 epochs) — 10-fold cross-validation.



Source: Elaborated by the author.

For evaluation, we use Precision, Recall and F-measure (RIJSBERGEN, 1974) calculated according to Equations 4.9, 4.10 and 4.11, described below. We consider correctly classified cattle as true positives (TP), other objects misclassified as cattle as false positives (FP), and missed cattle as false negatives (FN).

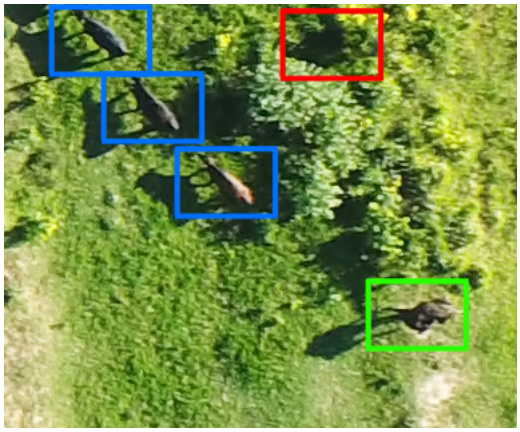
$$Precision = \frac{TP}{TP + FP} \tag{4.9}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.10}$$

$$F\text{-measure} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4.11}$$

Figure 24 shows an example of an image with 4 animals, where 3 correct detections

Figure 24 – Image with 4 animals, where 3 were detected correctly (TP - blue boxes), 1 was detected incorrectly (FP - red box), and one was missed (FN - green box).



Source: Elaborated by the author.

Table 5 – Detection results of the Net_1 network over $1,337$ images.

CNN	**Precision**	**Recall**	***F*-measure**
Net_1	0.928	0.935	0.932

(TP) are indicated by blue boxes, 1 incorrect detection (FP) is indicated by a red box, and 1 missed animal (FN) is indicated by a green box.

The Net_1 model was applied to detect the cattle in all of the $1,337$ test images. Table 5 shows the results, where it is clear that high performance was achieved across all measures, particularly considering that the test images were never seen by the network. This shows that the training set collected in this work covers a good variety of scenarios, allowing generalization with good accuracy for other sets obtained in areas not yet seen.

Once the CNN for cattle detection in the images has been validated, the next step is to evaluate the method of counting animals in an entire area, considering the overlap of the images and possible replication of animals.

4.3.2 Cattle Counting

To assess the method of counting livestock, we will use the 26 counting datasets described in Section 4.1.3. Our proposed animal counting method was applied to all flight sections using the Net_1 model, already described in Section 4.3.1. The default distance *threshold* for two detected animals to be duplicate candidates when located in different images (see Algorithm 1) is $6m$. This value was determined from the observation that a single animal usually does not exceed $3m$, and we empirically assume that an animal moves at most twice its size during image acquisition. This value has been set as default and used as such in our experiments, but it can be seen as an optional parameter that

Table 6 – Result of cattle counting for the flight sections classified as "Motionless" — estimated count against the ground truth (GT). The bottom bar displays the baseline results from (SHAO *et al.*, 2020).

Flight section	**GT**	**Estimated**	**FN**	**Duplicates**	**FP**
A	4	6	0	0	2
B	7	8	0	0	1
F-1	8	6	2	0	0
F-2	20	20	2	0	2
F-4	20	19	1	0	0
F-5	14	14	0	0	0
F-6	0	2	0	0	2
F-7	0	0	0	0	0
F-8	0	0	0	0	0
G-1	2	2	0	0	0
G-2	17	18	0	0	1
G-3	25	25	0	0	0
G-4	25	27	0	0	2
G-6	5	4	1	0	0
G-7	0	0	0	0	0
G-8	0	2	0	0	2
Overall	147	153	6	0	12
Reported by Shao *et al.* (2020)		143	9	0	5

Table 7 – Result of cattle counting for the flight sections classified as "Moving" — estimated count against the ground truth (GT). The bottom bar displays the baseline results from (SHAO *et al.*, 2020).

Flight section	**GT**	**Estimated**	**FN**	**Duplicates**	**FP**
C	12	17	2	6	1
D	5	6	0	0	1
E	5	7	0	0	2
F-3	24	25	0	0	1
G-5	19	20	1	0	2
Dataset 2	6	4	3	0	1
Overall	71	79	6	6	8
Reported by Shao *et al.* (2020)		86	14	19	10

could be adjusted for different application scenarios.

4.4 Results and Discussion

In order to compare the efficiency of our method against the state-of-the-art, we count the cattle in images provided by (SHAO *et al.*, 2020). Tables 6 and 7 display the results for the "Motionless" and "Moving" cases, respectively.

Table 8 – Results of cattle counting for the new flight sections *BR_set* — estimated count against the ground truth (GT).

Flight Section	**GT**	**Estimated**	**FN**	**Duplic.**	**FP**
SD_PV_90	5	5	1	0	1
2A_90	32	34	3	1	4
P1_AB_120	63	59	5	0	1
PD_AB_90	100	106	7	7	6
Overall	200	204	16	8	12

As shown in Tables 6 and 7, the proposed method slightly overestimated the amount of cattle in both scenarios, obtaining a counting value of 153 animals for the "Motionless" flight sections, where the ground truth is 147, and a counting value of 79 for the "Moving" sections, where the ground truth is 71. In contrast, (SHAO *et al.*, 2020) reported an underestimation (143) in the "Motionless" scenario and a larger overestimation (86) in the "Moving" scenario.

Although the difference between the total count errors made by each method is not major (14 for our method versus 19 for (SHAO *et al.*, 2020)), it is worth emphasizing that our Net_1 model was never faced with images from (or analogous to) the collection used for test in this experiment, which was provided by (SHAO *et al.*, 2020). In spite of this, we report significantly less false negatives than (SHAO *et al.*, 2020), while also improving on false positives in a number of flight sections, most noticeably in the "Moving" scenario.

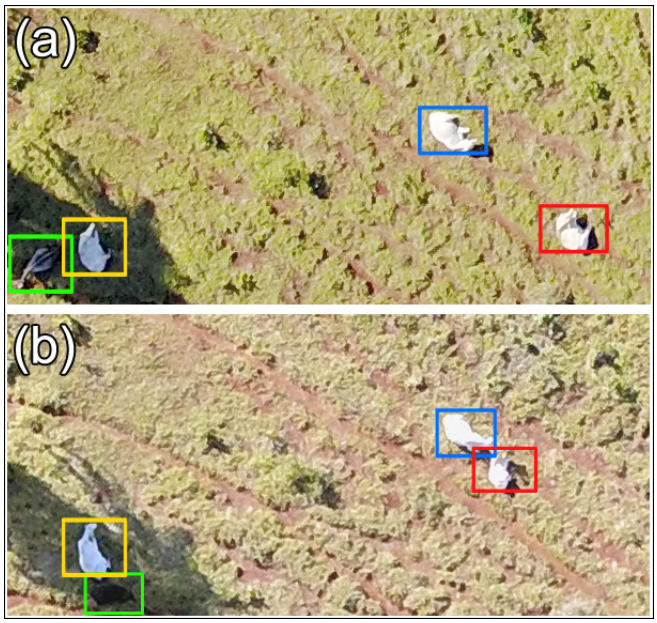
Regarding the strategy used to detect/remove duplicates, notice from Table 7 that our novel graph-based algorithm for duplicate removal outperformed the competing method by a large margin, partially failing only in flight section C, where there is a major movement of cattle involved.

The counting results for the 4 new flight sections *BR_set*, made available in this work, are shown in Table 8. The sum of the total count obtained by our method was 204 animals, whereas the ground truth is 200. Although the total result obtained by our method is only 2% above the ground truth, the numbers of FN and FP (16 and 12, respectively) represent 8% and 6% of the ground truth value, respectively.

In regard to the double count problem, in the SD_PV_90 and P1_AB_120 sections, the cattle movement is small because at the time of acquisition most animals were lying on the ground (as can be observed visually), and there was no duplicate counting. As for the 2A_90 section, only 1 of the 32 animals was counted in duplicate, because its movement was greater than the defined threshold. The PD_AB_90 section, which has the largest number of animals, was also the one with the largest amount of undetected duplicates (7 animals were counted more than once).

Figure 25 illustrates an area that is overlapped by two images taken at different times

Figure 25 – Two images of the same area taken at different time instants during the same flight plan. Matching color boxes represent matching animals according to our method.



Source: Elaborated by the author.

during the same flight plan. Boxes with the same color in Figure 25(a) and Figure 25(b) represent the same animals according to our method of duplicate cattle detection. As it can be seen, there is a small change in the location of the cattle marked with matching colour boxes from scene (a) to scene (b). However, the distance traveled by them is within the threshold defined in the method, enabling the correct detection and assignment of each animal to its new location.

4.4.1 Runtime Analysis: 3D Surface Location vs. Geolocation Estimates

As shown in Tables 6 and 7, our method obtains very competitive results when compared to (SHAO *et al.*, 2020), which uses 3D surface reconstruction of the photographed area to estimate the location of the detected animals. The code used in (SHAO *et al.*, 2020) is not publicly available, but in order to compare the runtime, we performed and timed the reconstruction of the 3D surface of each flight section to be used as a baseline. Note that the 3D reconstruction is required to produce the final result in (SHAO *et al.*, 2020) and represents the highest computational cost in that method, so it is a lower bound for its runtime and, as such, it allows a fair comparison against our complete detection pipeline. Table 9 shows the runtime in minutes to generate the reconstruction of the surface of each flight section using SfM as in (SHAO *et al.*, 2020), contrasted with the runtime to detect cattle using our geolocation estimates proposed in this chapter.

The 3D reconstruction alone takes, on average, 15 times longer to run than our

Table 9 – Comparison between the runtime of the proposed method against that of generating a 3D surface (using SfM) as required in (SHAO *et al.*, 2020).

Flight Section	**3D reconstruction runtime (min)**	**Our method runtime (min)**
A	189.54	9.06
B	203.15	12.21
C	175.9	9.02
D	194.26	10.32
E	563.12	46.07
F-1	102.53	5.75
F-2	99.7	5.63
F-3	97.52	6.45
F-4	101.64	5.68
F-5	100.3	5.65
F-6	95.12	5.7
F-7	97.01	5.73
F-8	93.12	5.81
G-1	96.15	5.88
G-2	101.22	5.86
G-3	96.14	5.71
G-4	105.93	5.75
G-5	99.54	5.92
G-6	99.11	5.61
G-7	100.42	5.58
G-8	102.3	5.68
Dataset 2	69.22	4.38
SD_PV_90	576.13	40.13
2A_90	1277.2	84.34
P1_AB_120	257.31	17.45
PD_AB_90	539.12	33.56

complete detection method. Considering only our own flight sections, on average our method takes $42 sec/ha$ to be performed, while the 3D reconstruction takes more than $11 min/ha$ on average.

In addition to the large runtime differences, the 3D surface reconstruction may fail if the images are not obtained with high overlap. According to (DANDOIS; OLANO; ELLIS, 2015), an overlap of at least 80% is desired to produce an accurate mapping. Another fundamental factor that limits its use is the flight time. The use of high overlap makes the flight plan time longer, impacting battery consumption, which may make it not possible to acquire images in a single flight. It also increases the possibility of cattle moving around, which often hinders correct counting. In our setup, a flight of ≈ 27 minutes suffices to cover an area of approximately 100ha using 30% overlap. For 80% overlap, as recommended for 3D reconstruction, and with the same acquisition system used in this chapter, a single flight would cover at most 15ha due to battery autonomy, which represents an area almost 7 times smaller.

Figure 26 – Example of unsuccessful orthophoto generation due to low overlap between images.

Source: Elaborated by the author.

Figure 26 illustrates the result of an orthophoto generation from the 3D mapping of a pasture at Primavera farm, using the 143 images obtained in the flight plan illustrated in Figure 12, where the overlap was set at 30%. As we can see in Figure 26, the mapping generated several gaps in the image due to the lack of matching points. It took 9 hours of processing to generate this mapping, while, for the same area, the counting method proposed in this work was able to finish in $\approx$ 40 minutes. Therefore, we believe our method represents a viable option for a realistic scenario and monitoring of large areas.

Considering the counting results presented in Tables 6 and 7 and the execution times presented in Table 9, it is noticeable that our cattle counting approach is competitive against the state-of-the art results (SHAO *et al.*, 2020), improving performance in terms of double counting with significantly reduced runtimes.

4.4.2 Experimental Framework for Benchmarking of Cattle Detection and Counting

As an additional contribution to the field of research of cattle counting in large areas, we make publicly available all the new image collections acquired for use in this work, namely, the training collection with 5058 images previously described in Section 4.1.2 as well as the *BR_set* test collection with 667 images previously described in Section 4.3.1.

These collections enable a variety of possible experimental protocols for future

benchmarking of cattle detection and counting. In the following, we illustrate two examples of such protocols, namely *Leave-One-Pasture-Out* (LOPO) and *Cross-Dataset*, both of which are focused on scenarios where the compared models are supposed to have been somehow pre-trained and are now entitled to self-tuning (using a more specialised collection of images) before they are assessed and compared on a collection that has not been used either for training or self-tuning.

Our first illustrative protocol, LOPO, takes a flight section with images from the same pasture as a *fold* and performs a *leave-one-out* type cross-validation across a collection of such flight sections by self-tuning a pre-trained model using all sections but one, which is used for test, then systematically repeating so that every flight section is used exactly once for test. Our second protocol, Cross-Dataset, takes two separate collections of images, each of which may consist of multiple flight sections, and uses one collection for self-tuning of a pre-trained model, while the other collection is used for test; then, the collections swap roles and the process is repeated.

We illustrate the two protocols above by using the pre-trained Net_1 model described in Section 4.3.1, subject to a fine-tuning procedure (ft), as follows:

- **LOPO** (using the four flight sections from BR_set)
  1. Net_1 with ft(SD_PV_90 + 2A_90 + P1_AB_120), test on PD_AB_90
  2. Net_1 with ft(SD_PV_90 + 2A_90 + PD_AB_90), test on P1_AB_120
  3. Net_1 with ft(SD_PV_90 + P1_AB_120 + PD_AB_90), test on 2A_90
  4. Net_1 with ft(2A_90 + P1_AB_120 + PD_AB_90), test on SD_PV_90
- **Cross-dataset** (using BR_set and the image collection from (SHAO *et al.*, 2020))
  1. Net_1 with $ft(BR\_set)$, test on dataset from (SHAO *et al.*, 2020)
  2. Net_1 with ft(dataset from (SHAO *et al.*, 2020)), test on BR_set

Fine-tuning is a procedure recommended when it is necessary to adapt a model already trained to new scenarios, in particular when the target dataset differs from the original training set (PONTI *et al.*, 2017). Fine-tuning was carried out during 50 training epochs for each one of the 6 scenario described above. Tables 10 and 11 show the cattle counting results obtained following the protocols LOPO and Cross-dataset, respectively.

For the LOPO protocol (Table 10), our method estimates a total number of 206 animals, while the ground truth is 200. Although the estimated number is farther from the ground truth than the number estimated by Net_1 without fine-tuning (Table 8), the fine-tuned model improved precision for cattle detection, reducing the number of FN and FP from 16 and 12 to 11 and 9, respectively. The number of duplicates remained at 8.

Table 10 – Results of cattle counting estimates against the ground truth (GT) for the Leave-one-pasture-out (LOPO) fine-tuning protocol.

Flight Section	**GT**	**Estimated**	**FN**	**Duplic.**	**FP**
SD_PV_90	5	5	1	0	1
2A_90	32	34	2	1	3
P1_AB_120	63	60	3	0	0
PD_AB_90	100	107	5	7	5
Overall	200	206	11	8	9

Table 11 – Results of cattle counting estimates against the ground truth (GT) for the Cross-dataset fine-tuning protocol.

Dataset	**GT**	**Estimated**	**FN**	**Duplic.**	**FP**
from (SHAO *et al.*, 2020)	218	231	9	6	16
BR_set	200	208	12	8	12
Overall	418	439	21	14	28

For the cross-dataset protocol, our method estimates a total of 439 animals, while the ground truth is 418, which represents an overestimation of $\approx 5\%$. Considering the results obtained from Net_1 without fine-tuning (Tables 6, 7 and 8), the fine tuned model also improves precision of cattle detection, reducing the number of FN and FP from 28 and 32 to 21 and 28, respectively.

4.5 Chapter Remarks

In this chapter, we have introduced an innovative approach to cattle counting in large areas, a method underpinned by the precise geolocation of animals and a graph-based algorithm designed for duplicate removal. Our outcomes have surpassed current state of the art practices, notably excelling in the realm of duplicate count reduction, all while significantly enhancing runtime efficiency by approximately 15 times. Additionally, we have put forward a novel cattle counting benchmark, featuring a fresh image collection and two experimental protocols. These resources serve as valuable tools for comparing and validating methods within real-world scenarios encompassing vast terrains.

This chapter underscores the feasibility of achieving comprehensive coverage with smaller image overlap, thereby providing accurate estimations of ground truth. Moreover, we have demonstrated that for the sole task of cattle counting, individual geolocated images suffice to generate precise estimates for both cattle location and count. This eliminates the necessity of constructing computationally intensive 3D maps or mosaics.

In the forthcoming chapter, we will delve into an in-depth analysis of cattle attributes, laying the foundation for an innovative approach to cattle counting and

duplicate removal.

CHAPTER

5

MULTI-ATTRIBUTE APPROACH FOR DUPLICATE LIVESTOCK REMOVAL AND COUNTING

In the previous chapter, we focused on cattle detection and counting in aerial images obtained by UAVs. The study proposed a method that employed Convolutional Neural Networks (CNNs) (PENATTI; NOGUEIRA; SANTOS, 2015) for cattle detection and a graph-based optimization technique to remove duplicated detections in overlapping images based on estimated animal geolocations and their distances. The results demonstrated the superiority of our proposed method as compared to the state-of-the-art, both in terms of counting accuracy as well as in reducing computational costs.

However, the previous approach relied solely on a fixed heuristic distance threshold to build the graph used for duplicate removal, based on the assumption that the same animal cannot move and appear further apart than the threshold in two different images. This assumption works reasonably well in many scenarios, but it is particularly over simplistic due to variable time gaps that inevitably occur between spatially adjacent images (especially in large areas), which can also be influenced by the flight plan configuration. As an example, Figure 27 illustrates a drone's flight plan designed to cover a pasture area. The green line represents the drone's path to survey the entire region, whereas the red dots indicate the locations where aerial photos are captured. In this plan, the drone starts from the top right green dot. The nearby yellow dotted box outlines the area imaged by the first photo. After the drone progresses from right to left and reaches the border, it descends and then returns covering an adjacent region from left to right, capturing additional photos. The blue dotted box indicates a photo whose area is adjacent to that of the first photo, taken after a time lapse that is significantly larger than the typical time lapse between any two consecutive photos. Of course, as the figure shows, this is just an extreme case of an effect that also occurs to different degrees involving other pairs of adjacent images.

Figure 27 – Drone's flight plan illustrating the path (green line) and photo capture locations (red dots) for a pasture area. The yellow and blue dotted boxes represent adjacent photos taken with a significant time lapse.

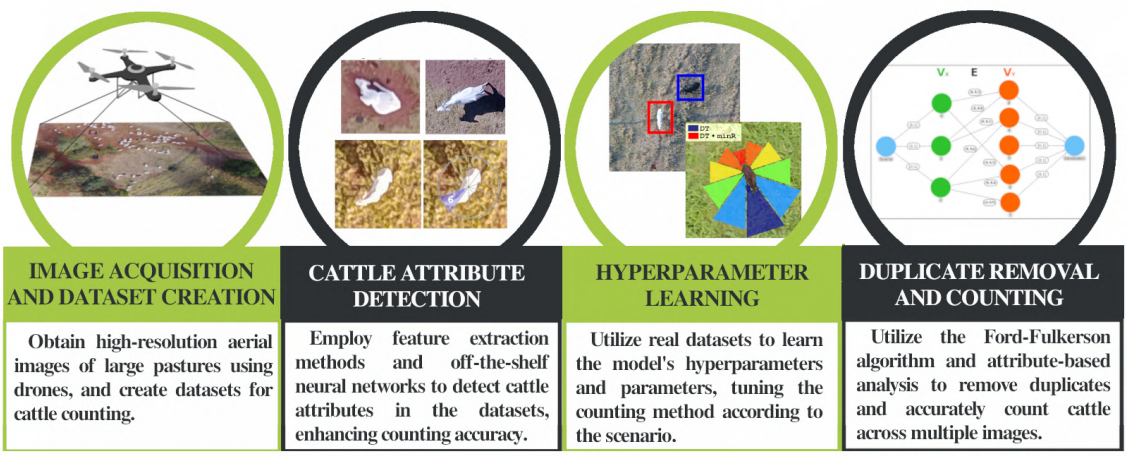
Source: Elaborated by the author.

Due to variable time gaps between spatially adjacent photos (and regardless of their level of overlap), using a fixed distance threshold as a basis for duplicate candidate removal is an important limitation of the previous approach. Relying solely on distances is another limitation, since there are many ambiguous scenarios that simply cannot be resolved with distances alone, as we will show later in this chapter.

To overcome these limitations and address the gaps in existing research, in this chapter we present the culmination of research efforts along multiple directions to improve effectiveness in cattle counting and duplicate removal. Our combined approach involves the analysis of multiple attributes beyond image pixel level. In particular, for each detected cattle, we take into account not only its estimated geolocation, but also its state (lying down or standing), color, velocity and direction. We investigate how such attributes can be properly combined and used to assign weights to a graph algorithm that we use to model and effectively solve the duplicate removal problem. Furthermore, thresholds on the graph weights are no longer set to heuristic default values as in Chapter 4, they are now learned from annotated training data following a cross-validation procedure instead.

We conducted an ablation study (MEYES *et al.*, 2019) to assess the contribution of each newly incorporated attribute to the overall accuracy of the cattle counting pipeline. This rigorous analysis ensures that every attribute incorporated into the counting methodology significantly contributed to improving the results. Also, to focus solely on counting and duplicate removal, we utilized manually annotated images for an independent evaluation of these tasks, ignoring potential object detection errors such as false positives and negatives that may precede such tasks in the pipeline. These errors constitute a separate concern that relates exclusively to the accuracy of the off-the-shelf convolutional networks we used for object detection, which is not the problem addressed in this chapter.

Figure 28 – Graphical abstract showcasing the main components of the proposed cattle counting approach. At a very high-level of abstraction, the methodology involves aerial image acquisition, cattle attribute detection, hyperparameter learning, and duplicate removal, leading to accurate and efficient cattle counting in large pasture areas.



Source: Elaborated by the author.

Figure 28 provides a concise overview of our cattle counting approach. It highlights key stages, such as aerial image acquisition, cattle attribute detection, hyperparameter learning, duplicate removal and counting. These critical components form the backbone of a comprehensive methodology employed in this work, designed to achieve accurate and efficient cattle counting in large pasture areas.

As an additional contribution, we introduce (and make it publicly available) a new collection of datasets comprising real images obtained from large pasture areas, which can be utilized to train and evaluate other cattle counting techniques to be developed in the future.

The remainder of this chapter is organized as follows. In Section 5.1, we present the materials and methods used in this study, including our new dataset collection. Section 5.2 details our proposed method for duplicate removal and cattle counting, emphasizing the attribute-based analysis. In Section 5.3, we delve into the hyperparameter specifications and parameter training. Moving forward, Section 5.4 offers a comprehensive account of our experiments and results, demonstrating the efficacy of our feature analysis and substantial improvements in counting accuracy when compared to prior work and state-of-the-art methods. Finally, in Section 5.5, we draw the chapter remarks.

5.1 Materials and Methods

This section provides a comprehensive overview of the materials and methods employed in our research. We first describe the devices used to acquire cattle images and

outline the datasets used for pretraining models and learning hyperparameters, as well as those used to evaluate the cattle counting process. Specifically, we introduce five new datasets meticulously collected for this study, ensuring diverse and representative samples. Additionally, we detail the evaluation methods utilized to assess the performance of our approach. In this section, we also detail the Convolutional Neural Network (CNN) used to identify key attributes of the cattle, such as state, color, and direction.

5.1.1 Devices

The aerial images were captured using a Drone DJI Phantom 4 (Figure29) equipped with the following characteristics:

- Camera:
  - Sensor: CMOS 1/2.3”, 12.4 megapixels
  - Lens: 28mm
  - Maximum aperture: f 2.8
  - ISO range (photo): 100-1600
  - Maximum image size: $4000 \cdot 3000$ pixels
- Battery: 5350 mAh ($\approx$ 28 minutes of flight)
- Controller: maximum range of 5 km
- Maximum speed: 20 m/s

The camera’s specifications allowed us to obtain high-resolution photographs of the pasture areas and the cattle therein, ensuring the clarity and detail required for our analysis. The application DroneDeploy[1] was employed on an Apple iPhone 6s mobile device to plan and execute our flights. We also utilized a laptop Dell Precision 5560 equipped with an i7 processor, 16GB RAM, and an NVIDIA T1200 graphics card to process and analyze the collected images efficiently.

5.1.2 Datasets for Training

This section provides an overview of the two image collections employed to train the machine learning models used in our work. The first dataset, *T2606*, is used for training three Convolutional Neural Network (CNN) models designed for cattle attribute detection (Section 5.1.4). The second dataset, *ADJ165*, is used for training a logistic regression model intended to assess the likelihood that a pair of detected cattle could be the same animal based on its movement across images (Section 5.3.3). Both datasets are detailed below.

[1] Information available at https://www.dronedeploy.com

Figure 29 – Drone DJI Phantom 4

Source: Elaborated by the author.

- ***T2606* dataset:** comprises a total of 2,606 individual cattle images. These images were cropped from aerial photographs taken in 2017, 2018 and 2020 at several farms in Brazil. The dataset encompasses cattle with diverse characteristics, including a wide range of coat colors (such as black, white, red, spotted, etc.), positions (standing or lying down), and backgrounds featuring red soil, sandy soil, dry pastures, and green pastures.

- ***ADJ165* dataset:** comprises a total of 330 full-frame aerial photographs, also captured in 2017, 2018 and 2020 at various farms in Brazil. In contrast to the *T2606* dataset, the images in this collection have not been cropped and they feature multiple cattle each. Specifically, the dataset consists of 165 pairs of spatially adjacent photos, where each pair contains at least one individual animal appearing in both images. The adjacent photos were taken at different time intervals; some were captured sequentially just a few seconds apart, while others involve a round-trip of the drone (lateral adjacency) with time gaps of up to 4 minutes.

5.1.3 Datasets for Cattle Counting

In this section, we present the datasets used for the cattle counting task, encompassing 4 datasets from (SOARES *et al.*, 2021) and 17 datasets from (SHAO *et al.*, 2020). Originally, the dataset collection from (SHAO *et al.*, 2020) consists of 22 datasets, but 5 of those datasets do not contain any cattle. As our focus is on cattle counting rather than detection, these 5 datasets were excluded from our study. In addition to the 21 datasets from these previous studies, we also introduce 5 novel datasets carefully curated for cattle counting, making them publicly available for the research community.

Table 12 – Overview of 26 datasets used in our experiments, featuring 5 new ones.

Source	Dataset	No. of Images	No. of Cattle	Area (ha)
From (SHAO *et al.*, 2020)	**A**	32	4	1.5 to 5.0
	B	45	7	1.5 to 5.0
	C	32	12	1.5 to 5.0
	D	40	5	1.5 to 5.0
	E	184	5	1.5 to 5.0
	F-1	20	8	1.5 to 5.0
	F-2	20	20	1.5 to 5.0
	F-3	20	24	1.5 to 5.0
	F-4	20	20	1.5 to 5.0
	F-5	20	14	1.5 to 5.0
	G-1	20	2	1.5 to 5.0
	G-2	20	17	1.5 to 5.0
	G-3	20	25	1.5 to 5.0
	G-4	20	25	1.5 to 5.0
	G-5	20	19	1.5 to 5.0
	G-6	20	5	1.5 to 5.0
	Dataset 2	14	6	1.5 to 5.0
BR_set	**SD_PV_90**	143	5	78
	2A_90	325	32	56
	P1_AB_120	61	63	50
	PD_AB_90	138	100	54
New Datasets	**Pasto4-16-10-7h**	146	109	71
	Sede-18-10-7h	50	22	20
	Brejo-19-10-12h	62	43	27
	Pasto2-17-10-7h	198	275	98
	Pasto1-15-10-12h	224	239	102

Our five novel datasets were acquired specifically for cattle counting at "Água Boa" farm, located in the state of Mato Grosso do Sul, Brazil, over five consecutive days in October 2020. The DroneDeploy application was used to plan flight paths with 30% front and side overlap between images. The mapping speed during the flight plan was set to 15m/s to ensure efficient image capture. All flight plans were conducted at an altitude of 90m. The images were acquired at two distinct hours of the day (7 am and 12 pm). The areas span from 20 to 102 hectares, with animal counts ranging from 22 to 275 and the number of photos varying between 50 and 224. These datasets offer a wide range of variation for training and evaluating cattle counting algorithms.

Table 12 provides an overview of all 26 datasets used in this work. It includes details such as the number of images, number of cattle and area covered in hectares (ha) for each dataset. The animals in these datasets exhibit varied color patterns, reflecting the diverse breeds observed in real-world cattle counting scenarios. Figure 30 showcases an example image from the Pasto1-15-10-12h dataset, featuring cattle of various colors, such

Figure 30 – Photo from the Pasto1-15-10-12h dataset, featuring diverse cattle colors: white, black, and spotted.

Source: Elaborated by the author.

as white, black, and spotted.

The datasets, exclusively consisting of cattle, can serve the purpose of a standard benchmark collection for evaluating cattle counting and cattle detection methodologies, providing researchers with a diverse set of real-world scenarios to further advance the field. Importantly, the cattle in these counting datasets were manually labeled. This ensures evaluation can focus exclusively on counting accuracy, rather than automatic detection performance, if so desired. This is the case in our study.

It is important to note that *none of the images* in the training datasets *T2606* and *ADJ165* (Section 5.1.2) are present in the datasets used for cattle counting (Table 12), i.e., there is no overlap between these two dataset collections.

5.1.4 Convolutional Neural Network (CNN)

Our approach employs a pretrained MobileNetV3 Large CNN architecture (HOWARD *et al.*, 2019) to address the tasks of color, state, and direction detection in cattle. This architecture strikes a balance between computational efficiency and high accuracy (PONTI *et al.*, 2021). As a successor to MobileNetV2 (SANDLER *et al.*, 2019), MobileNetV3 continues the lineage of efficient MobileNets, which have been widely adopted for various computer vision tasks due to their lightweight and fast nature. With MobileNetV3 Large, we were able to achieve accurate attribute detection for cattle counting while still maintaining a low computational footprint, making it ideal for real-world applications.

We train three attribute detection models using MobileNetV3 Large as backbone:

1. **State Detection:** Classifies cattle into two states: standing up or lying down.
2. **Color Detection:** Classifies cattle into four color categories: black, white, red/brown, and other.
3. **Direction Detection:** Determines cattle orientation in ten classes of 36-degree segments.

To optimize both time and accuracy, we fine-tuned each MobileNetV3 Large model on the *T2606* dataset over a total of 50 epochs, divided into two distinctive phases. During the initial 25 epochs, only the classification layer undergoes training. In the subsequent 25 epochs, all layers from the fifth (and last) block of the network are unfrozen and trained along with the classification layer. The choice of 50 epochs was determined through visual analysis of the training and validation loss curves averaged in a cross-validated fashion, whose details will be discussed in Section 5.3. The chosen optimizer is AdamW (LOSHCHILOV; HUTTER, 2017), featuring a scheduled learning rate. The learning rate embarked at 0.01 and remained constant for the initial 10 epochs. Subsequently, a gradual decay was initiated, adhering to an exponential reduction factor of $exp(-0.1)$ per epoch. For all models we employed the categorical cross-entropy loss function and batch size of 32.

The attribute detection models play an important role in our enhanced cattle counting pipeline, as we will discuss later in this chapter.

5.1.5 Evaluation Methods

To assess the performance of our counting method, we employ two evaluation measures: absolute error and percentage error, both computed in relation to the ground truth. The absolute error directly measures the deviation between the counted number of animals and the actual number present. The percentage error, in turn, represents this deviation as a percentage relative to the ground truth value, allowing for a more realistic interpretation and facilitating comparison across different scenarios, as it accounts for errors proportionally to the size of the dataset.

In addition to evaluating the overall performance of our counting method, we also conduct an ablation study to validate the effectiveness of individual features used in our approach. This study involves systematically analyzing the impact of removing specific components or attributes — namely, state, color, velocity, direction, and distance — from our method to assess their individual contributions to the overall performance.

To statistically compare different configurations of our method in the ablation study, as well as to compare our cattle counting method against state-of-the-art approaches,

we are utilizing the Wilcoxon signed-rank test (REY; NEUHÄUSER, 2011). It is a non-parametric statistical test that evaluates whether the differences in percentage errors between compared configurations are statistically significant or simply due to random chance. The test involves calculating the differences in percentage errors between paired samples, ranking the absolute values of these differences, then summing the ranks of positive differences. The resulting p-value measures the probability of obtaining the observed results by chance under the null hypothesis, where no difference is assumed to exist.

The Wilcoxon signed-rank test is well-suited for our work as it does not assume a specific data distribution, allowing for robust non-parametric evaluation of cases potentially involving outliers and/or non-normally distributed data. This test allows us to make data-driven decisions about the relative importance of each candidate attribute to be adopted in the counting pipeline as well as to compare our cattle counting method against state-of-the-art approaches.

5.2 Duplicated Removal and Counting Method

In this section, we present our proposed Duplicate Removal and Cattle Counting Method, which aims to accurately detect and remove duplicate cattle when counting the animals in a collection of images. The method encompasses three key components: Multi-Attribute Enhancement, Modified Ford-Fulkerson Algorithm for Duplicate Cattle Detection, and the Complete Counting Method.

The Multi-Attribute Enhancement involves computing weights for pairs of candidate cattle detected in two different images. The Modified Ford-Fulkerson Algorithm incorporates these computed attribute weights and adapts the concept of maximum flow in a bipartite graph to address the assignment problem, whereby potential duplicate cattle can be identified. Finally, the Complete Counting Method processes the image collection in chronological order, cross-referencing each image with the cumulative list of previously counted cattle, resulting in the final cattle count. These components will be detailed below.

5.2.1 Multi-Attribute Enhancement

In this section, we introduce a comprehensive multi-attribute enhancement that plays a crucial role as a weight measure in our graph-based cattle matching method. Extensive research and experimentation have been conducted to incorporate attributes into the cattle identification process, contributing to improved accuracy and robustness.

Given an animal x in an image X and an animal y in an image Y, where image X was taken chronologically before image Y, we define $w_{x,y} \in [0, 1]$ as the probability (weight) that x and y refer to the same animal. The weight formula is defined as follows:

$$w_{x,y} = state(x, y) \cdot color(x, y) \cdot P_v(x, y) \cdot D_t(x, y) \tag{5.1}$$

Each term in the weight formula represents a specific attribute contributing to the matching process, as follows:

- $state(x, y) \in \{0, 1\}$: Represents the (dis)agreement between animals x and y regarding their state (e.g., laying down or standing up).
- $color(x, y) \in \{0, 1\}$: Represents the (dis)agreement in terms of color category between animals x and y.
- $P_v(x, y) \in [0, 1]$: Models the probability that animals x and y are the same, based on their velocity, considering the movement from x's to y's coordinates.
- $D_t(x, y) \in \{0, 1\}$: Models whether or not the pair x and y is deemed a realistic/possible duplicate pair candidate based on their distance as well as on their orientation (and, accordingly, their most likely directions of movement).

Notably, $P_v(x, y)$ is the only attribute with a value ranging between 0 and 1. All other attributes are binary. Consequently, if any binary attribute is equal to 0, the overall weight $w_{x,y}$ in Equation 5.1 becomes 0. Conversely, if all binary terms are 1, the weight $w_{x,y}$ is then fully determined by $P_v(x, y)$. The intuition is that our velocity model will only effectively apply in those cases that cannot be promptly resolved (i.e., duplicate candidates cannot be discarded) with very high confidence based on the other attributes.

In the following sections, we delve into the details of each attribute and explain precisely how they are computed.

5.2.1.1 State Attribute

The state attribute requires a classifier to determine whether a cattle is standing up or lying down in each image. For an animal x, we denote its individual state as $state(x)$:

$$state(x) = \begin{cases} 0, & \text{if cattle } x \text{ is standing up} \\ 1, & \text{if cattle } x \text{ is lying down} \end{cases} \tag{5.2}$$

The degree of confidence (certainty) of this classification as returned by the corresponding CNN model is represented by $S_s(x) \in [0, 1]$. The individual states of two animals x and y in different images are combined to form attribute $state(x, y)$ in Equation 5.1. Since this binary attribute has the power to fully determine the weight $w_{x,y}$ as zero when indicating a disagreement between animals x and y regarding their state (i.e., when $state(x, y) = 0$), such an indication requires not only that the individual states are detected as different ($state(x) \neq state(y)$) but also that *both* classifications have been made with very high confidence, as follows:

$$state(x,y) = \begin{cases} 0, & \texttt{if } state(x) \neq state(y) \\ & \quad \texttt{and } S_s(x) > C_s \texttt{ and } S_s(y) > C_s \\ 1, & \texttt{otherwise} \end{cases} \tag{5.3}$$

In short, $state(x, y)$ is 0 if one animal is standing up whereas the other is lying down, and their classifications as such have a confidence score (degree of certainty) above a threshold C_s (otherwise, $state(x, y) = 1$). Threshold C_s is a hyperparameter that will be learned during experiments.

Incorporating the state attribute allows us to eliminate upfront candidate matches that involve animals in obviously conflicting (standing or lying) positions.

5.2.1.2 Color Attribute

As for the state attribute, a classifier is also used to detect the color patterns associated with different cattle breeds. By considering the color of the cattle in each image, we can discard potential matches that clearly do not belong to the same cattle, based on their distinctive colors.

The agreement for the color attribute is defined similarly to the state attribute. Let $color(x)$ represent the predicted color class of animal x, and $S_c(x) \in [0, 1]$ denote the degree of certainty about this classification. The color agreement between animals x and y is given by:

$$color(x,y) \begin{cases} 0, & \texttt{if } color(x) \neq color(y) \\ & \quad \texttt{and } S_c(x) > C_c \texttt{ and } S_c(y) > C_c \\ 1, & \texttt{otherwise} \end{cases} \tag{5.4}$$

The value of $color(x, y)$ is 0 if the predicted color classes (black, white, red/brown, other) for animals x and y are different, and their classifications have a degree of certainty above a threshold C_c, which will be learned during experiments. Otherwise, $color(x, y) = 1$.

While this attribute may not influence the results in pastures where all cattle share the same color, it becomes very useful in pastures with multiple cattle breeds, helping avoid mismatches during the identification process.

5.2.1.3 Velocity Attribute

The velocity attribute is designed to measure the speed of cattle presumably moving between consecutive images. Let Δt be the time interval between images X and Y, and let $\text{dist}(x, y)$ denote the geographical distance between cattle x and y based on their latitude and longitude coordinates, which can be derived from the GPS and camera metadata in the images as described in (SOARES *et al.*, 2021). The velocity $V(x, y)$ of an animal potentially moving from the coordinates of x to the coordinates of y can be calculated as

follows:

$$V(x, y) = \frac{\text{dist}(x, y)}{\Delta t} \tag{5.5}$$

To estimate the probability $P_v(x, y)$ that cattle x and y are the same based on their velocity, we use a classic Logistic Regression model, given by the following formula:

$$P_v(x, y) = \frac{1}{1 + \exp(-(\beta_0 + \beta_1 \cdot V(x, y)))} \tag{5.6}$$

where β_0 and β_1 are the model coefficients. These coefficients will be learned from the collection of manually annotated training data. The idea is to learn the match vs. mismatch probability profile from labelled data containing examples from both scenarios, in a way that properly accounts for the fact that the time lapse between images varies.

Unlike the other attributes, which yield binary values to determine whether a candidate match should be discarded or not, the velocity attribute stands out as the sole attribute providing a probability value between 0 and 1. This characteristic makes the velocity attribute particularly important in our graph-based cattle matching method, as it solely defines the weight value assigned to potential matches that have not been discarded by the other attributes in Equation 5.1.

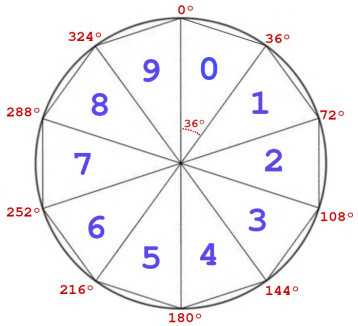
5.2.1.4 Direction Attribute

The direction attribute utilizes a classifier to detect the orientation of cattle x within image X, based on the direction its head is pointing to. It then compares this detected orientation with the direction along which cattle x would supposedly move from its location in image X to the location of a candidate match y in image Y, accounting for cattle movement rather than simply attributing it to (and indiscriminately handling it as) drone positioning errors.

Let DC be the number of direction classes chosen to train the classification model, where each class represents $(360/DC)$ degrees of the rotation circumference in the image perspective. We define $c_x \in \{0, 1, \ldots, DC - 1\}$ as the direction class value for cattle x. For instance, $c_x = 0$ corresponds to the first direction class, which starts at 0 degrees of rotation, whereas $c_x = DC - 1$ corresponds to the last direction class, which starts at $(360 - (360/DC))$ degrees of rotation. Figure 31 illustrates an example with $DC = 10$, where each class covers 36°. In this case, $c_x = 0$ corresponds to angles between 0° and 35°, while $c_x = 9$ corresponds to angles between 324° and 359°.

The direction class considers only the direction of the cattle relative to the image, not the compass direction. To obtain the compass (global) direction of a cattle x, we must account for both the direction class of this cattle as well as the direction of the drone, which is given by the *yaw* attribute obtained from the image's metadata. As shown in Section 4.2.1, the *yaw* value ranges between -180 and 180 degrees. For simplicity, we normalize the *yaw* angle within the interval $[0, 360]$ according to Equation 5.7.

Figure 31 – Example of $DC = 10$ Direction Classes for Cattle Head Rotation in Image Perspective, with each class representing 36° of the circumference.



Source: Elaborated by the author.

$$yaw_N(X) = \begin{cases} 360 - |yaw(X)|, & \text{if } yaw(X) < 0 \\ yaw(X), & \text{otherwise} \end{cases} \tag{5.7}$$

With the normalized yaw of image X ($yaw_N(X)$) and the cattle's direction class, c_x, we determine the global direction (ϕ) of cattle x as:

$$\phi_x = \left(c_x \cdot \left(\frac{360}{DC} \right) + yaw_N(X) \right) \mod 360 \tag{5.8}$$

After determining the global direction to which cattle x's head is pointing (ϕ_x), the next step involves calculating the direction of the supposed movement from cattle x to cattle y (ϕ_{xy}), where y is the candidate match in question. This computation relies on their geographic coordinates (lat_x, lng_x) and (lat_y, lng_y) as follows:

$$\phi_{xy} = \tan^{-1} \left(\frac{lat_y - lat_x}{lng_y - lng_x} \right) \cdot \frac{180°}{\pi} \tag{5.9}$$

Having obtained the global direction of cattle x (ϕ_x) and the direction of the supposed movement from x to y (ϕ_{xy}), the rotation difference between them ($\Delta r_{x,y}$) can be calculated as:

$$\Delta r_{x,y} = \min \left(|\phi_x - \phi_{xy}|, 360 - |\phi_x - \phi_{xy}| \right) \tag{5.10}$$

Using the rotation difference we can now compute the final threshold ratio, θ_{xy}, which indicates how aligned the supposed movement from x to y is with respect to cattle x's direction. The ratio is:

$$\theta_{xy} = 1 - \left(\left\lfloor \frac{\Delta r_{x,y}}{\frac{360}{DC}} + 0.5 \right\rfloor \cdot \frac{(1 - minR)}{\frac{DC}{2}} \right) \tag{5.11}$$

with $\theta_{xy} \in [minR, 1]$, where $0 < minR < 1$ is a minimum threshold imposed on θ_{xy}. Notice that θ_{xy} is assigned a value of 1 when the supposed movement correctly

aligns with the cattle's direction class (i.e., when $0 \leq \Delta r_{x,y} < (360/DC/2)$, such that $\lfloor(\Delta r_{x,y}/360/DC) + 0.5\rfloor = 0$). On the other extreme, θ_{xy} is assigned a minimum value of $minR$ when these are in opposite directions, namely, when $(180 - (360/DC/2)) \leq \Delta r_{x,y} < 180$, such that $\lfloor(\Delta r_{x,y}/360/DC) + 0.5\rfloor = DC/2$.

The use of a minimum threshold $minR \in (0, 1)$ prevents completely ruling out the possibility of x and y matching, despite the supposed movement from x to y and cattle x's head being in opposite directions. This minimum threshold allows accommodating potential GPS precision errors and inaccuracies in location estimation. Setting the value of $minR$ will be discussed later. For rotation deviations other than correct alignment or opposite alignment, the method proportionally decreases the ratio from 1 to $minR$ according to the number of classes (DC) between them.

5.2.1.5 Distance Threshold Attribute

Incorporating velocity as an attribute is important to account for the different time intervals between adjacent images, but it requires establishing a maximum distance threshold to account for the plausible range of cattle movement. When a drone executes a flight plan to cover a large pasture area, photos of certain adjacent regions may be captured with a significant time lapse between them. If we only consider velocity of cattle movement, this could increase the number of potential cattle mismatches, making it more challenging to accurately identify duplicates. The reason is that, while certain velocities may be realistic over short periods of time that are observed between most adjacent images, they would imply that very unlikely distances would need to be traversed over larger time lapses observed between some images, so a large number of unlikely candidate matches in those images would not be discarded based on velocity alone.

Let $\text{dist}(x, y)$ denote the distance between cattle x and y in adjacent images X and Y, respectively, and let DT be a maximum distance threshold beyond which a movement is very unlikely, so it is deemed not plausible. This threshold will be experimentally learned from data later in our study. Now, consider the direction-based threshold ratio, θ_{xy}, computed in the previous section. Then, the distance threshold agreement between cattle x and y is defined as:

$$D_t(x, y) = \begin{cases} 0, & \texttt{if } \text{dist}(x, y) > (DT \cdot \theta_{xy}) \\ 1, & \texttt{otherwise} \end{cases} \tag{5.12}$$

In this formulation, the maximum plausible distance DT is adjusted by the direction-based ratio θ_{xy}. Specifically, when the supposed movement from x to y correctly aligns with x's direction, then $\theta_{xy} = 1$ and the full threshold DT applies. When $\theta_{xy} < 1$, indicating misalignment and, accordingly, a less likely movement, the effective acceptable distance threshold $DT \cdot \theta_{xy}$ becomes tighter. If the distance between cattle x and y exceeds such an adjusted maximum distance, then $D_t(x, y) = 0$, indicating that cattle x cannot be

Figure 32 – Distance threshold variation according to cattle direction class: highest threshold (DT) for head direction (shaded in dark blue), proportional decrease for others (light blue, green, yellow, and orange), and minimum threshold ($DT \cdot minR$) for opposite direction (shaded in red).

Source: Elaborated by the author.

matched to cattle y. Otherwise, if the distance satisfies the threshold, $D_t(x, y) = 1$ (valid matching candidate).

Figure 32 illustrates an example of how the distance threshold behaves based on the direction of the cattle (as a categorical class variable with $DC = 10$ classes). The figure illustrates the relationship between the maximum distance threshold (DT) and the direction class to which the head of the cattle is pointed. The area shaded in dark blue represents the direction to which the cattle's head is facing, thence the effective threshold within that class (direction) is equal to DT. In contrast, the area shaded in red represents the opposite direction, where the minimum threshold is applied as $DT \cdot minR$. For the remaining directions (shaded in light blue, green, yellow, and orange), the threshold gradually decreases from DT down to $DT \cdot minR$. This dynamic thresholding mechanism ensures good adaptability to various cattle movement patterns, accounting for both perfect and opposite alignments, as well as anything in between, while accommodating GPS precision errors and location estimation inaccuracies.

Next, we present the modified Ford-Fulkerson algorithm that employs our final combined probability estimate in Equation 5.1 ($w_{x,y}$) as graph weights to detect duplicate cattle based on the assignment problem.

5.2.2 Modified Ford-Fulkerson Algorithm for Duplicate Cattle Detection

The Ford-Fulkerson algorithm (JR; FULKERSON, 1962) is a well-known method for solving the maximum flow problem in a network. In the Chapter 4, we employed the Ford-Fulkerson algorithm to solve the maximum flow problem in a bipartite graph with unweighted edges. This approach allowed us to determine the maximum selection of edges, effectively obtaining the maximum matching for duplicate removal.

In the present work, we have introduced weighted edges into the problem specification. In a weighted bipartite graph, it can be shown that by imposing a constraint that each node in one partition can be matched to at most one node in the other partition, the original maximum flow problem becomes fully equivalent to an assignment problem. Although there are alternative methods available to solving the assignment problem, such as the Hungarian algorithm (MILLER; STONE; COX, 1997) and the Jonker-Volgenant algorithm (CROUSE, 2016), both offering improved asymptotic complexity, for the sake of compatibility with the method proposed in Chapter 4, we have chosen to use an adapted version of the original Ford-Fulkerson maximum flow algorithm to handle the assignment problem (TRIPPI; ASH; II, 1974). It should be noted that all these algorithms are guaranteed to find the same (globally optimal) solution, so the choice of algorithm does not affect the results (FLORIAN; KLEIN, 1970).

Our implementation of the Ford-Fulkerson algorithm incorporates steps specifically tailored for detecting duplicate cattle in image analysis, as summarized in Algorithm 2. The key specializations lie in the definition of the graph, the calculation of edge weights, and the aforementioned constraint imposed on matches.

1. **Graph Construction**: We start by constructing a bipartite graph using two sets of nodes, V_X and V_Y, representing cattle in different photos (X and Y, where X denotes photos preceding Y in chronological order). The graph consists of a source node, a destination node, and edges (E) connecting nodes in V_X to nodes in V_Y. Each edge represents the probability that a cattle node in V_X is the same as a cattle node in V_Y. The graph is initialized with zero flow (lines 1 and 2).

2. **Edge Weight Calculation**: The edge weights are the probabilities $w_{x,y}$ resulting from the multi-attribute aggregation formula in Equation 5.1. The edge weights are assigned values between 0 and 1, where higher values indicate a higher likelihood of a match between the corresponding adjacent cattle nodes.

3. **Constraint Enforcement**: We impose a constraint during the flow augmentation process to ensure that each cattle node in V_X is matched with at most one cattle node in V_Y (and vice versa). At each step, when selecting an augmenting path, the algorithm considers only edges that maintain this constraint. This ensures that the resulting flow corresponds to a valid matching (assignment) of cattle nodes between the two photos.

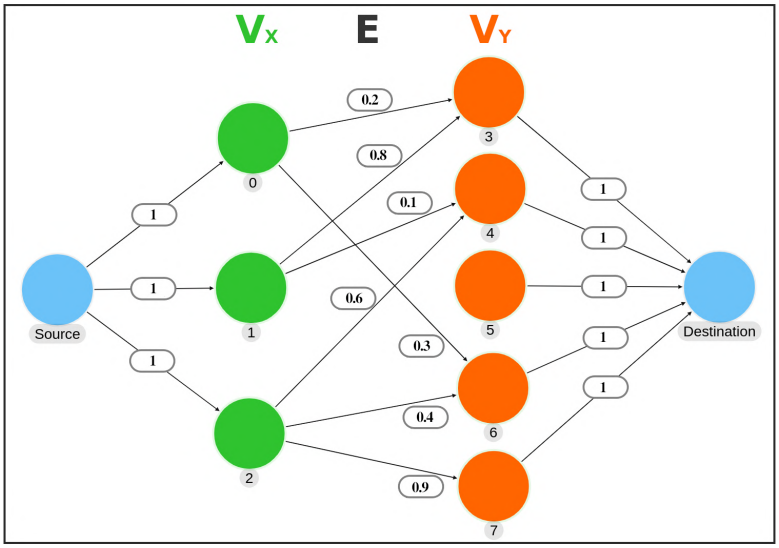
Figure 33 provides an illustration of a graph constructed to solve the cattle duplicate detection problem using the modified maximum flow algorithm. In this example, the graph consists of a source node, which is fully connected to all the three nodes in V_X. These nodes in V_X are connected to nodes in V_Y by weighted edges (E), describing the probability that

Algorithm 2 Modified Ford-Fulkerson Algorithm for Duplicate Cattle Detection

Input: V_X: Set of cattle nodes in image X
V_Y: Set of cattle nodes in image Y
E: Edges with probability weights $(w_{x,y})$
Output: E': Chosen edges

```
Create bipartite graph with nodes V_X, V_Y, source, and destination
Initialize graph with zero flow
while augmenting path exists do
    Find augmenting path using breadth-first search
    Calculate maximum flow along the path based on
    edge capacities and matching constraint
    Update flow and residual capacities
Determine maximum flow value
return E'
```

Figure 33 – Graph representation for duplicate cattle detection using maximum flow in networks. The graph consists of a source node, nodes in V_X and V_Y (representing cattle in images X and Y, respectively), and a destination (sink) node. Directed edges (E) contain matching probabilities as weights, representing maximum flow capacity between V_X and V_Y.



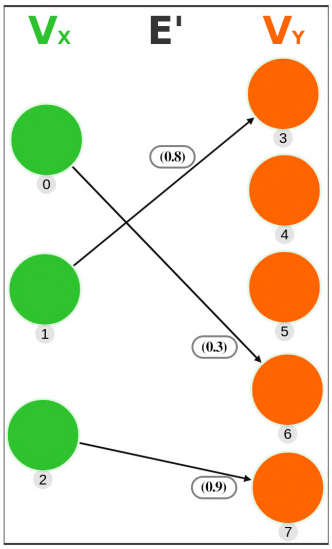
Source: Elaborated by the author.

the corresponding end nodes represent the same animal (the absence of an edge indicates a weight equal to zero). Additionally, all five nodes in V_Y are connected to a destination node. The graph is initialized with zero flow.

The Ford-Fulkerson algorithm adheres to the principle that the flow through an edge cannot exceed its capacity (weight). Additionally, except for the source and destination vertices, the inflow to a vertex must equal its outflow.

The algorithm iteratively finds augmenting paths in the graph using a breadth-first

Figure 34 – Result of the Ford-Fulkerson algorithm for duplicate cattle detection. The graph illustrates the selected edges (E'), (0, 6), (1, 3), and (2, 7), indicating the identified duplicates. Cattle 4 and 5 are deemed new observations in image Y rather than duplicates of animals previously observed in image X.



Source: Elaborated by the author.

search (BFS) (BUNDY; WALLEN, 1984) approach (line 4). Once an augmenting path is found, the algorithm calculates the maximum flow that can be sent along the path based on the edge capacities and the constraint on cattle matching (line 5). The flow is updated, and the process continues until no more augmenting paths can be found (line 6). At the end, given the equivalence between the result of this modified maximum flow algorithm and the solution to the assignment problem as previously discussed, the final chosen edges indicate the specific matches between cattle detected as duplicates in the two photos in question.

Figure 34 depicts the result of the modified Ford-Fulkerson algorithm applied to the scenario illustrated in Figure 33. The graph in this figure includes only the selected subset of edges E' resulting from solving the maximum flow problem. Specifically, the chosen edges are (0, 6), (1, 3), and (2, 7), indicating that the method identifies cattle 0 as the same as 6, cattle 1 as the same as 3, and cattle 2 as the same as 7. Conversely, cattle 4 and 5 are deemed new observations to be added to the total cattle count (see next section).

5.2.3 Complete Counting Method

In this subsection, we present our method for counting cattle in multiple images captured by a drone in a systematic flight pattern. The drone follows a predefined path that covers the entire area using a serpentine movement strategy, resulting in overlapping images. Due to the drone's forward and backward motions along this serpentine path,

cattle may appear in different locations across different images. Leveraging the metadata from drone images, including GPS location, camera angle, sensor size, focal length, rotation (yaw), and altitude, we can compute the coordinates of cattle on the ground as seen in the images (SOARES *et al.*, 2021).

The set of images is processed in chronological order, ensuring a consistent sequence. To handle duplicate cattle and accurately count the unique instances, we employ the Ford-Fulkerson algorithm, as described earlier. The Ford-Fulkerson algorithm assesses the probability of a cattle in one image being the same as a cattle in another image, considering attributes such as state, color, velocity, direction and distance. Algorithm 3 shows all steps of the cattle counting and duplicate removal method for a set of images.

Algorithm 3 Cattle Counting Algorithm

Input: *image_list*
Output: L_t

1: $L_t = \emptyset$
2: **for** each $i \in image\_list$ in chronological order **do**
3: L_i = get cattle coordinate list from image i
4: $M = |L_i|$
5: $N = |L_t|$
6: **if** $M == 0$ **then**
7: continue
8: **if** $N == 0$ **then**
9: add all cattle $\in$ L_i to L_t
10: **else**
11: Apply the Ford-Fulkerson algorithm on L_t and L_i, resulting in a graph $G' = (X, Y, E')$, where E' contains only edges selected as part of the maximum flow solution.
12: **for** $y \in G'$ **do**
13: **if** $deg(y) == 0$ **then**
14: add $L_i(y)$ to L_t
15: **else**
16: **for** $x \in G'$ **do**
17: **if** $(x, y) \notin E'$ **then**
18: continue
19: **else**
20: update location and timestamp of $L_t(x)$ as $L_i(y)$
21: break
22: **return** L_t

Algorithm 3 takes a list of images from a pasture as input, where cattle need to be counted, and outputs a final list L_t containing all valid animals with their updated locations throughout the image collection.

The algorithm starts by initializing an empty list L_t (line 1). For each image i in chronological order, a partial list L_i is created, containing the geolocation of the detected animals in that image (line 3). If L_i is empty (line 6), indicating no animals were detected in the current image, the algorithm moves to the next image. If L_t is empty (line 8),

it means either no images have been processed yet or no animals were detected in the previous processed images. In this case, we initialize L_t with all the animals from L_i.

As long as L_t is not empty ($N \neq 0$), the modified Ford-Fulkerson algorithm is employed to identify duplicates among the cattle in L_t and L_i (line 11). Solving the constrained maximum flow problem yields a graph $G' = (V_X, V_Y, E')$, where E' contains the optimal assignment edges. Here, V_X represents the vertices in L_t (animals already counted in all previously processed images, with their updated locations) and V_Y represents the vertices in L_i (animals detected in the current image). Vertices $y \in V_Y$ with degree 0 in G' correspond to cattle in L_i without a matching animal in L_t (line 13). As these animals are not deemed duplicates, we add all of them to L_t (line 14).

When a vertex $x \in V_X$ is adjacent to a vertex $y \in V_Y$ in G', which means they are considered duplicates, the location and timestamp of the corresponding cattle in L_t, denoted as $L_t(x)$, is updated with the coordinates from $L_i(y)$ (line 20). Importantly, the location of duplicated cattle is updated based on the chronological order of image processing. If an animal is in motion, the algorithm keeps track of its last detected location. Upon processing all images, the resulting list L_t contains the total number of cattle discounting duplicates.

5.3 Hyperparameter Determination and Attribute Learning

As it will be discussed in detail in the following subsections, the values of each hyperparameter are automatically learned from data alongside the corresponding attribute models, or they are justified taking into account relevant technical information and specific characteristics of the datasets of interest, or both.

5.3.1 State Attribute Learning

Recall that, according to the state attribute, we define the matching probability in Equation 5.3 as zero if the predicted classes for cattle x and y are different with a confidence score greater than or equal to a threshold C_s. To determine the value of C_s, we conducted experiments using the *T2606* dataset, presented in Section 5.1.2. In this dataset, we manually annotated the 2,606 cattle images in both standing up and lying down positions, as illustrated in Figure 35.

To address the challenge of imbalanced class distribution, we adopted data augmentation (DA) techniques (PEREZ; WANG, 2017) to effectively increase the number of training samples and achieve a balanced representation across all classes. Table 13 provides an overview of the image distribution for each state class in both the original dataset and the augmented dataset.

We employed a standard 10-fold cross-validation strategy, where each non-training

Figure 35 – Example of images representing both classes of cattle state in the *T2606* dataset: Laying down (left) and Standing up (right).

Source: Elaborated by the author.

Table 13 – Number of images for each state class before and after data augmentation (DA).

State Class	**Before DA**		**After DA**	
	# Img.	**%**	**# Img.**	**%**
Standing Up	2,103	80.70%	2,500	50%
Lying Down	503	19.30%	2,500	50%
Total	**2,606**	**100%**	**5,000**	**100%**

fold (containing 10% of the dataset) was then subdivided into two validation subsets. One of these validation subsets ($val1$) was used for computing the error curve and determining the optimal number of epochs for early stopping of the training process. The other validation subset ($val2$) was reserved for determining the C_s threshold. The remaining 9 folds (90% of the data) were used for training the MobileNetV3 large architecture, following the details provided in Section 5.1.4.

We performed the 10-fold cross-validation for 100 epochs, tracking the training and $val1$ losses for each fold and epoch. To determine the optimal epoch for early stopping, we analyzed the cross-validated average training and average $val1$ losses at each epoch. We found that at epoch 50 the $val1$ loss reaches a minimum; past that point, it begins to increase, while the average training loss is still decreasing, indicating overfitting. Thus, we selected 50 epochs as the best point to stop training the model.

For each (alternating) non-training fold, we obtained the average softmax score using the $val2$ validation subset with respect to the actual class labels (ground-truth annotation). The total average (AVG) and standard deviation (STD) of the softmax score across all 10 folds are shown in Table 14.

Based on the cross-validation results, we determined that the average softmax score for the state classifier is $C_s = 0.8571$. This value indicates the degree of confidence required for the agreement between classes of cattle x and y to be considered reliable. Following this rationale, if the confidence scores for the individual states of *both* x and y exceed C_s, and $state(x) \neq state(y)$, we confidently affirm that the states are different, so

Table 14 – 10-fold cross-validation results for the state and color classifiers within the *val*2 subsets across the non-training folds. The softmax score represents the classifier's confidence in predicting the correct (ground-truth) class.

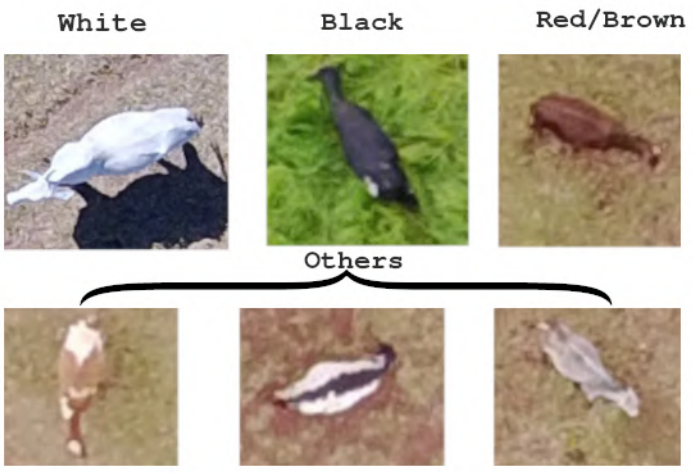
10-Fold	**Total AVG Softmax Score**	**Total STD Softmax Score**
State	0.8571	0.112
Color	0.7692	0.215

x and y are not the same cattle and they are promptly discarded as a candidate match.

5.3.2 Color Attribute Learning

To determine the confidence threshold for the color attribute in Equation 5.4, C_c, we followed the same methodology described above for the state attribute, using the *T2606* dataset. We manually annotated the 2,606 cattle images in this dataset into four color classes: white, black, red/brown, and others, as shown in Figure 36. As with the state classes, the number of images for each color class was imbalanced. To mitigate this imbalance, we once again applied data augmentation (DA), as shown in Table 15.

Figure 36 – Example of cattle images from each of the four annotated color classes in the *T2606* dataset: white, black, red/brown, and others.



Source: Elaborated by the author.

Analogous to the state attribute, we employed a 10-fold cross-validation strategy using the color augmented dataset for training the MobileNetV3 network across the alternating training folds and then using the results within the validation subset of each non-training fold as a basis to tune the confidence threshold C_c. Table 14 presents the total average (AVG) and standard deviation (STD) of the softmax score with respect to the actual class of the validation examples across all 10 folds.

Table 15 – Number of images for each color class before and after data augmentation (DA).

Color Class	Before DA		After DA	
	# Img.	%	# Img.	%
White	1,120	42.98%	2,000	25%
Black	760	29.16%	2,000	25%
Red/Brown	422	16.19%	2,000	25%
Others	304	11.67%	2,000	25%
Total	**2,606**	**100%**	**8,000**	**100%**

Based on the cross-validation results, we determined that the average softmax score for the color classifier is $C_c = 0.7692$. Following the same rationale as for the state attribute, if the confidence scores for the individual colors of *both* x and y exceed C_s, and $color(x) \neq color(y)$, we confidently affirm that the colors are different, so x and y are not the same cattle and they are promptly discarded as a candidate duplicate match.

5.3.3 Velocity Attribute Learning

The velocity attribute aims to model the movement speed of cattle between adjacent images. To determine the probability of a cattle pair being the same animal based on velocity, we use a classic logistic regression model. In this case, logistic regression models the probability of two cattle detected in different images being the same animal (as discriminated from being different animals), using velocity as independent variable. Using the *ADJ165* dataset (Section 5.1.2), we have data with examples of two classes for training the logistic regression model:

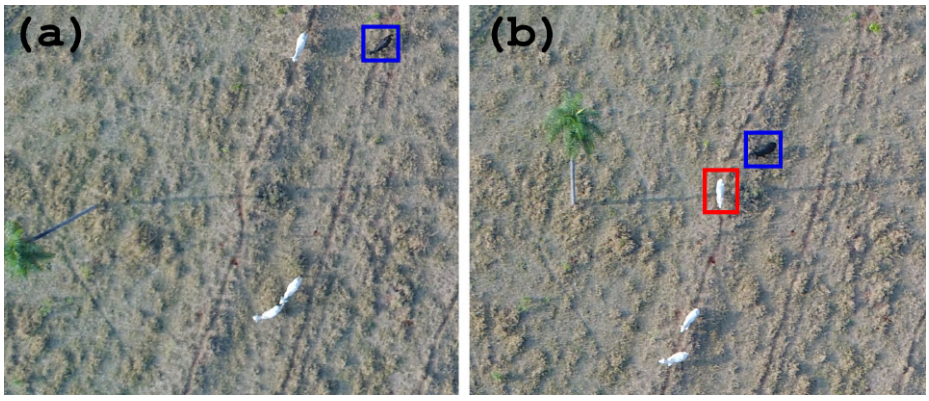
1. **Same Cattle (positive class)**: We have 165 examples consisting of pairs of images where the same animal appears in both. By using the cattle coordinate estimator and the timestamp of the images, we calculated the velocity of each animal moving between the coordinates in the first and second images where it appears. These 165 velocities constitute the data corresponding to the "Same Cattle" class.

2. **Other Cattle (negative class)**: For each pair of images containing an example of the "Same Cattle" class, we took another animal in the latest image with the closest coordinate to the animal corresponding to the "Same Cattle" example in that image. The coordinate of this closest non-matching animal was then used to compute the hypothetical velocity (should the corresponding movement have occurred, as a worst-case negative example that we want to be able to discriminate from). We collected 158 such hypothetical velocities belonging to the "Other Cattle" class. The reason why this is smaller than the number of positive examples (165) is that in 7 positive examples the pair of images contain a single (i.e., the same) animal.

To ensure that the animal in the negative class is not the same as the one in the first image, we selectively choose image pairs featuring cattle with distinct colors or visually identifiable characteristics. This deliberate selection minimizes the possibility of misidentifying the negative class as the same cattle in the initial image.

Figure 37 presents two images, (a) and (b), captured at different moments but depicting the same area with four cattle. In both images, there is a black animal marked with a blue bounding box. The timestamps of the photos allow us to compute the velocity of this black animal, resulting in a sample of the "Same Cattle" velocity class. Additionally, a white animal with a red bounding box appears in image (b) and is the closest to the location of the black cattle in that image. The velocity that would be required for the black animal to move from its blue bounding box in image (a) to the location of the red bounding box in image (b) represents a sample of the "Other Cattle" velocity class.

Figure 37 – Cattle velocity computation for the "Same Cattle" and "Other Cattle" classes. The velocity of the black animal with a blue bounding box in (a) and (b) belongs to the "Same Cattle" velocity class. The hypothetical velocity of the black cattle with a blue bounding box in (a), had it moved to the location of the white cattle with a red bounding box in (b), belongs to the "Other Cattle" velocity class.



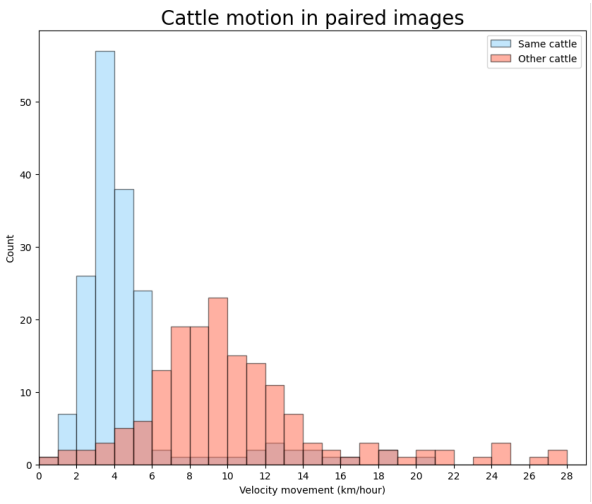
Source: Elaborated by the author.

The histogram in Figure 38 illustrates the distribution of velocities for both classes. The "Same Cattle" velocities (in blue) tend to be lower, indicating that cattle pairs that correspond to the same animal typically have lower movement speeds between adjacent images. Conversely, the "Other Cattle" velocities (shown in red) tend to be higher.

Using the "Same Cattle" and "Other Cattle" velocities as training data, we applied linear logistic regression adopting the `lbfgs` solver from `sklearn.linear_model` to learn the model coefficients β_0 (intercept) and β_1 (slope), as defined in Equation 5.6 in Section 5.2.1.3.

The logistic regression model fits a sigmoid curve to the data, allowing us to estimate the probability of any cattle pair in different images being the same animal

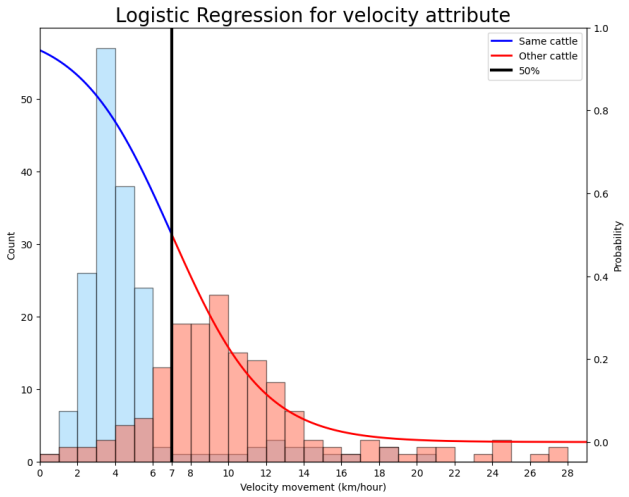
Figure 38 – Histograms of velocity for the "Same Cattle" (blue) and "Other Cattle" (red) classes.



Source: Elaborated by the author.

based on its velocity. Figure 39 shows the sigmoid curve learned by the logistic regression model, plotted over the velocity histograms. As expected, as the velocity decreases, the estimated probability of the pair being the same animal increases. The point where the curve intersects the 50% probability line (black line) is approximately $7km/h$. Cattle pairs with velocities below this threshold are more likely to be classified as "Same Cattle", while those with velocities above the threshold are more likely to be classified as "Other Cattle".

Figure 39 – Sigmoid curve learned by the logistic regression model for the velocity attribute: the black vertical line indicates a 50% probability cut-off value.



Source: Elaborated by the author.

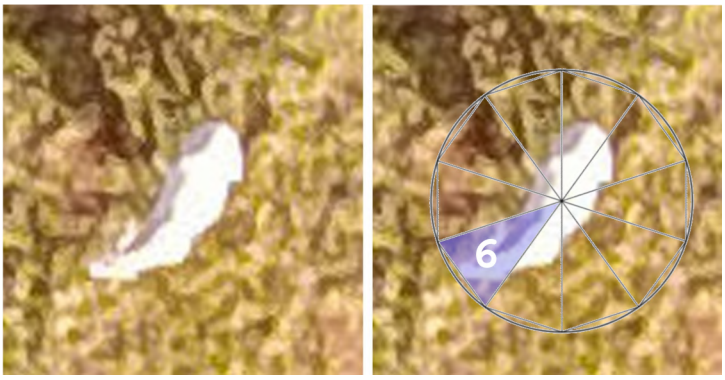
The logistic regression model yields a matching probability for each candidate cattle pair. Instead of performing a binary classification by applying a cut-off (threshold) value to this probability, as illustrated by the black vertical line in Figure 39, the real-valued

probability score is directly used as term $P_v(x, y)$ in the computation of the weights $w_{x,y}$ for the Ford-Fulkerson algorithm in Equation 5.1. Cattle pairs with higher probability scores are given higher weights, indicating a higher likelihood of being the same animal. The Ford-Fulkerson algorithm will use these weights to find the maximum flow through the graph, determining the optimal matching of cattle nodes and effectively identifying duplicate cattle in the image analysis process.

5.3.4 Direction Attribute Learning

The direction attribute plays a significant role in determining the orientation of cattle and enables the identification of their most likely movement directions. To represent the direction attribute, we divide the 360-degree circle into $DC = 10$ classes, with each class representing 36 degrees (see Figure 31). For training the direction classifier, we utilize the *T2606* dataset, which has been manually annotated to include the direction to which the animal's head is pointed within the image. The training data encompasses samples from all possible directions, ensuring comprehensive coverage for the classifier. Figure 40 illustrates an example of a cattle image annotated with class 6, representing the specific direction where the cattle's head is pointed.

Figure 40 – Example of a cattle image annotated as class 6 for the direction attribute. The class is defined by the direction where the cattle's head is pointed.



Source: Elaborated by the author.

To ensure a balanced representation of each direction class, we apply data augmentation techniques (DA). After data augmentation, each direction class has 530 samples, resulting in a total of 5,300 samples for training (Table 16).

To tackle the direction classification task, we again employed the MobileNetV3 network. The trained model is employed to detect the direction class of cattle with respect to the image frame, which is subsequently converted into compass direction through the image's GPS metadata, and then used to compute the direction ratio attribute in Equation 5.11, as previously detailed in Section 5.2.1.4.

Table 16 – Number of images for each direction class before and after data augmentation (DA).

Class	Direction (degrees)	Before DA		After DA	
		# Img.	%	# Img.	%
0	0° to 35°	296	11.3%	530	10%
1	36° to 71°	247	9.4%	530	10%
2	72° to 107°	162	6.2%	530	10%
3	108° to 143°	269	10.3%	530	10%
4	144° to 179°	339	13.0%	530	10%
5	180° to 215°	257	9.8%	530	10%
6	216° to 251°	226	8.6%	530	10%
7	252° to 287°	267	10.2%	530	10%
8	288° to 323°	336	12.8%	530	10%
9	324° to 359°	207	7.9%	530	10%
	Total	**2,606**	100%	**5,300**	**100%**

To compute the direction ratio attribute, we also need to define the hyperparameter $minR$, namely, the percentage of the maximum plausible distance threshold to be allowed even when the presumed movement direction of a candidate match between two images is opposite to the direction of the animal in question in the earliest image. The exact value of $minR$ will be determined in the next section, as it depends on another hyperparameter related to the distance attribute.

5.3.5 Distance Attribute Learning

As previously discussed in Section 5.2.1.5, establishing an appropriate distance threshold (DT) is essential to adjust tolerance to assumed cattle movements between adjacent images. If a distance threshold is not defined, and the timestamp between two photos is substantial, the potential matching range for cattle could become excessively large. To mitigate this issue, recall that we set a maximum distance threshold (DT) to limit the distance of movement deemed plausible.

In our approach, if the distance between a cattle instance in one photo and a candidate match in a subsequent adjacent photo is greater than DT, they are promptly discarded as a potential duplicate, regardless of the animal's orientation and most likely directions of movement.

For distances smaller than DT, recall from Serction 5.2.1.5 that this maximum threshold can be further adjusted (tightened) according to the orientation of the cattle, which in turn depends on a hyperparameter $minR$. We set this hyperparameter accounting for possible GPS imprecision when estimating the cattle's coordinates on the ground, as well as the accuracy of the drawn bounding box that detects the cattle. For example, a static cattle instance may appear to move backwards due to estimated location errors, leading to potential matches being discarded. By considering $minR$, we can account for

such errors and retain candidate matches that might otherwise be incorrectly discarded.

To define the value of $minR$, we refer to studies evaluating the accuracy of drone-based photogrammetry using GPS technology. In 2020phantom4precision, a DJI Phantom 4 RTK drone achieved an accuracy close to 2 cm. However, our drone is not equipped with RTK technology, a GPS correction system enabling centimeter-level positioning accuracy, due to its higher cost. A study on a DJI Mavic Pro Platinum, which contains similar GPS technology to ours, reported an average horizontal positional discrepancy of approximately 0.883 meters 2021mavic. Based on this information, we rounded up this value and set the smallest adjusted maximum distance threshold to $DT \cdot minR = 0.9$ meters in our experiments, i.e., $minR = 0.9/DT$.

To determine a value for DT, we undertake a training process using the dataset collection for cattle counting described in Section 5.1.3. However, this collection will also be used for the final evaluation (test) of our complete counting pipeline and, with only 26 datasets available, conventional train-test splits would entail sacrificing a significant portion of our valuable data just for the purpose of DT training, limiting our scope for evaluation (test) of the counting task itself. To address this issue, we employ a leave-one-out cross-validation methodology, as elaborated in Section 5.4.1. This well-known, principled approach enables us to leverage our entire dataset collection for both learning DT as well as for properly testing our complete counting pipeline in the final counting task, without violating the fundamental machine learning requirement that the same data cannot be used simultaneously both for learning as well as for testing.

5.4 Evaluation

In this section, we present a range of experimental results involving the cattle counting task. We begin by reporting the results of our cattle counting method using a leave-one-out cross-validation approach. Next, we conduct an ablation study to assess the significance of each attribute in our counting method. Finally, we compare our approach against state of the art baseline techniques to evaluate its effectiveness.

5.4.1 Leave-One-Out Cattle Counting Evaluation

Recall from Section 5.3.5 that, unlike other hyperparameters in our counting method, which were pretrained using specialized training datasets described in Section 5.1.2, the DT hyperparameter needs to be ultimately learned from a dataset collection that represents cattle counting scenarios. We conducted preliminary counting experiments on all 26 datasets in Section 5.1.3 using candidate DT values ranging from 3 to 20 meters (in steps of 1 meter). The resulting absolute errors, with respect to the actual number of cattle in each dataset (ground truth — GT), were recorded in Table 17.

Table 17 – Absolute counting error for each candidate value for the maximum distance threshold (DT), varying from 3 to 20 meters.

Dataset	**GT**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
A	**4**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	**7**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F-1	**8**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F-2	**20**	1	0	0	0	0	0	0	0	0	1	2	2	2	3	5	5	5	7
F-4	**20**	2	1	0	0	0	0	0	0	1	2	2	3	3	4	4	6	6	7
F-5	**14**	1	0	0	0	0	0	0	0	0	0	0	1	1	2	3	4	5	5
G-1	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G-2	**17**	1	1	0	0	0	0	0	0	0	0	1	1	1	2	3	3	3	4
G-3	**25**	0	0	0	0	0	1	0	0	0	0	0	0	0	1	2	4	4	4
G-4	**25**	0	0	0	0	0	0	1	0	1	1	1	1	2	2	2	3	5	5
G-6	**5**	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	**12**	6	6	6	6	6	6	4	4	4	5	5	5	5	5	5	5	5	6
D	**5**	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	**5**	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F-3	**24**	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
G-5	**19**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
Dataset2	**6**	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SD_PV_90	**5**	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	2
2A_90	**32**	7	7	5	5	4	4	3	2	2	2	2	2	3	3	3	3	4	5
P1_AB_120	**63**	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PD_AB_90	**100**	9	9	7	7	6	6	5	5	5	5	5	5	7	7	8	8	8	12
Pasto4-16-10-7h	**109**	8	8	7	8	8	7	6	5	5	5	6	6	7	7	7	7	7	9
Sede-18-10-7h	**22**	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	2	2
Brejo-19-10-12h	**43**	3	3	3	3	3	3	3	2	2	2	2	2	2	2	3	3	5	5
Pasto2-17-10-7h	**275**	19	17	15	14	13	9	7	6	6	7	7	7	8	8	8	10	10	10
Pasto1-15-10-12h	**239**	2	2	1	2	2	1	2	1	1	1	2	3	3	3	3	3	3	4
TOTAL ERROR		**69**	**62**	**46**	**47**	**44**	**39**	**32**	**26**	**28**	**32**	**36**	**39**	**45**	**51**	**59**	**67**	**74**	**89**

Once all the absolute counting errors for the various candidate DT values in Table 17 have been pre-computed, we can apply a standard *leave-one-out* cross-validated approach as follows: each time, we leave one dataset out for evaluation (test) of the counting error, using the best DT value with respect to the remaining 25 datasets (i.e., the value that provides the least total counting error, *excluding the error associated with the left-out dataset*). This approach ensures that the test error is assessed on a left-out dataset that has never been seen before in any stage during model training/learning. We repeated this process for all 26 datasets, leaving a different dataset out each time for test. The resulting absolute and percentage counting errors for each test dataset are shown in Table 18.

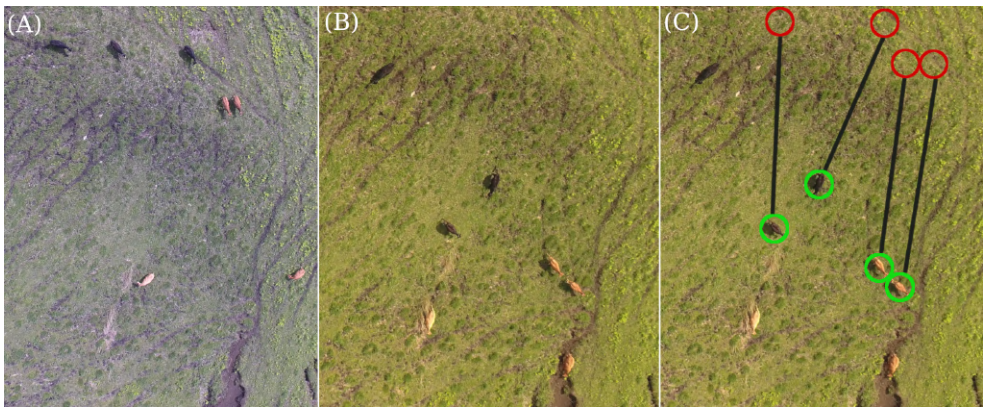
Table 18 – Absolute and percentage counting errors with respect to the ground truth (GT) in Leave-One-Out Cross Validation for each of the 26 validation datasets. For each dataset, the counting results are obtained using a distance threshold (DT) value learned from the other 25 datasets.

Dataset	**GT**	**Error**	
		ABS	**%**
A	**4**	0	0,00%
B	**7**	0	0,00%
F-1	**8**	0	0,00%
F-2	**20**	0	0,00%
F-4	**20**	0	0,00%
F-5	**14**	0	0,00%
G-1	**2**	0	0,00%
G-2	**17**	0	0,00%
G-3	**25**	0	0,00%
G-4	**25**	0	0,00%
G-6	**5**	0	0,00%
C	**12**	4	33,33%
D	**5**	0	0,00%
E	**5**	0	0,00%
F-3	**24**	0	0,00%
G-5	**19**	0	0,00%
Dataset2	**6**	0	0,00%
SD_PV_90	**5**	0	0,00%
2A_90	**32**	2	6,25%
P1_AB_120	**63**	0	0,00%
PD_AB_90	**100**	5	5,00%
Pasto4-16-10-7h	**109**	5	4,59%
Sede-18-10-7h	**22**	1	4,55%
Brejo-19-10-12h	**43**	2	4,65%
Pasto2-17-10-7h	**275**	6	2,18%
Pasto1-15-10-12h	**239**	1	0,42%
		SUM	**AVG**
TOTAL:		**26**	**2,34%**

The results reported in Table 18 represent the cross-validated performance of our proposed cattle counting method. The method achieved a total absolute counting error of 26 cattle, with an average percentage error of 2.34%. While accurately estimating the count in 18 out of 26 validation datasets, there is some variability in error across the remaining 8 datasets. This is mostly within the range $[0.42\%, 6.25\%]$, except for Dataset C, which exhibits a noticeable 33.33% error and calls for further analysis.

This substantial error can be attributed to the unique characteristics of Dataset C. Firstly, its small size, with only 12 cattle, means a discrepancy of four cattle results in a significant percentage error. Secondly, this dataset contains images where the movement of some animals is both atypical as well as in marked contrast to other animals appearing in the same images. For instance, consider Figure 41, where the images show four cattle

Figure 41 – Example of two images from Dataset C: (A) captures the initial moment, depicting the positions of seven cattle. (B) shows a subsequent moment less than a minute later, revealing significant movement by some cattle. (C) provides a visualization of the movement, illustrating the distance covered by four rapidly moving cattle within this short interval. Notably, other cattle in the image exhibit more typical behavior.



Source: Elaborated by the author.

that have moved considerably within a minute. Our method, designed for typical grazing scenarios, faced a challenge with this abnormal behavior. The rapid and uncharacteristic movement of this subset of animals, which is in contrast to the other cattle exhibiting more typical behavior captured in the same images, misled the algorithm to consider the former as distinct instances, contributing to the observed error.

In the following section, we delve into an ablation study, where we systematically evaluate the impact of each attribute employed in our proposed method.

5.4.2 Ablation Study

We conduct this study by systematically removing one attribute at a time and analyzing its impact on the counting performance. The attributes considered in this study are state, color, velocity, direction, and distance. Table 19 enumerates the configurations of all combinations of attributes for the ablation study. Configuration **A1** includes all attributes, while configurations **A2** to **A6** remove one attribute at a time. Specifically, **A2** removes only the distance threshold, **A3** removes only the direction attribute, **A4** removes only the velocity attribute, **A5** removes only the color attribute, and **A6** removes only the state attribute.

Table 20 presents the absolute counting error and percentage error for each ablation configuration on each of the 26 datasets, along with the total sum of absolute errors and the average percentage error for each configuration across all datasets. To compare the different model configurations, we focus mainly on the percentage error measure as it provides a more meaningful interpretation of the counting error, relative to the size of the

Table 19 – Ablation study configurations and their attribute combinations.

Config.	State	Color	Velocity	Direction	Distance
A1	✓	✓	✓	✓	✓
A2	✓	✓	✓	✓	
A3	✓	✓	✓		✓
A4	✓	✓		✓	✓
A5	✓		✓	✓	✓
A6		✓	✓	✓	✓

Table 20 – Ablation study results for the cattle counting method. It shows the absolute and percentage errors for different attribute configurations on 26 datasets, along with their total absolute sum and average percentage errors across all datasets.

Dataset	GT	A1		A2		A3		A4		A5		A6	
		ABS	%	ABS	%	ABS	%	ABS	%	ABS	%	ABS	%
A	**4**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
B	**7**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
F-1	**8**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
F-2	**20**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
F-4	**20**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
F-5	**14**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
G-1	**2**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
G-2	**17**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
G-3	**25**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
G-4	**25**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
G-6	**5**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
C	**12**	4	33.33	5	41.67	5	41.67	5	41.67	4	33.33	4	33.33
D	**5**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
E	**5**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
F-3	**24**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
G-5	**19**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
Dataset2	**6**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
SD_PV_90	**5**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
2A_90	**32**	2	6.25	3	9.38	2	6.25	3	9.38	3	9.38	2	6.25
P1_AB_120	**63**	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
PD_AB_90	**100**	5	5.00	5	5.00	5	5.00	6	6.00	5	5.00	5	5.00
Pasto4-16-10-7h	**109**	5	4.59	6	5.50	6	5.50	6	5.50	8	7.34	6	5.50
Sede-18-10-7h	**22**	1	4.55	2	9.09	1	4.55	2	9.09	2	9.09	2	9.09
Brejo-19-10-12h	**43**	2	4.65	2	4.65	2	4.65	4	9.30	3	6.98	3	6.98
Pasto2-17-10-7h	**275**	6	2.18	11	4.00	10	3.64	8	2.91	6	2.18	6	2.18
Pasto1-15-10-12h	**239**	1	0.42	34	14.23	3	1.26	2	0.84	3	1.26	2	0.84
		SUM	**AVG**	**SUM**	**AVG**	**SUM**	**AVG**	**SUM**	**AVG**	**SUM**	**AVG**	**SUM**	**AVG**
TOTAL:		**26**	**2.34**	**68**	**3.60**	**34**	**2.79**	**36**	**3.26**	**34**	**2.87**	**30**	**2.66**

dataset (in terms of its actual total number of cattle, GT).

Note that no error was observed across all configurations (A1 to A6) in 18 datasets. These datasets involve relatively stationary cattle in the overlapping areas, leading to consistently accurate counting results even after removing any attribute. However, in the remaining 8 datasets, the results noticeably changed depending on the attribute removed.

Among all configurations, the complete configuration (A1) achieved the best performance, with an absolute error of 26 counts across all datasets and the least percentage error of 2.34%. This indicates that all attributes, including state, color, velocity, direction, and distance, play a relevant role in contributing to the counting accuracy.

In contrast, configuration A2, which doesn't include the distance threshold attribute, yielded the worst result, with an absolute error of 68 counts (3.6%). This highlights the importance of maintaining a distance threshold when incorporating velocity information in the counting method. Without the distance threshold, by assuming constant cattle movement speed the method may generate mismatches between distant animals over long time intervals, leading to a negative impact on counting accuracy. By retaining the distance threshold, we can effectively address such mismatches and uphold the reliability of the counting process.

To statistically assess the differences between the results, we use the Wilcoxon signed-rank test on the percentage error from each dataset and configuration, which provides p-values with respect to each ablation setup when compared against the full model (A1) across the collection of datasets. The results are shown in Table 21. To avoid the pitfalls of assessing significance based on a fixed critical value while performing multiple pairwise tests, we only report the p-values without imposing an arbitrary significance level.

Table 21 – Wilcoxon signed-rank p-values for the ablation study.

Comparison	**P-value**
A1 vs. A2	0.027
A1 vs. A3	0.067
A1 vs. A4	0.011
A1 vs. A5	0.046
A1 vs. A6	0.079

Based on the p-values in Table 21, configurations A2, A4, and A5 exhibited the most significant deterioration in percentage errors compared to A1, while configurations A3 and A6 show less significant changes. It is important to note that, even though removing these attributes shows less significant differences, the total absolute error for the full configuration (A1) remains smaller. This suggests that attributes related to distance, velocity, and color have more significantly contributed to the accuracy of the cattle counting method.

Overall, the ablation study provides valuable insights into the importance of individual cattle attributes and helps understanding which aspects of the method could be further optimized to enhance counting accuracy.

5.4.3 Thresholded Variant: An Unweighted Version Using Multi-Attribute

In addition to the proposed complete method (A1), we present an alternative version based on the cattle counting approach introduced in Chapter 4. Referred to as the "Thresholded Variant," this version aims to explore the adaptability of multi-attribute analysis in an unweighted graph format. In the Thresholded Variant, edge connections are

binary, representing either the presence or absence of connections between cattle within the image data.

The Thresholded Variant acts as an experimental bridge between our weighted multi-attribute method and the prior unweighted algorithm. While in the complete A1 method, the state, color, direction, and distance attributes define the presence or absence of edges in the graph, with the velocity attribute providing weight even if it's low, the Thresholded Variant employs a different strategy. This variant retains an edge only if its weight exceeds 0.5, setting the weight to 1 in such cases. If the weight falls below this threshold, the edge is removed. Formally, this adjustment is described by Equation 5.1:

$$w_{x,y} = \begin{cases} 0, & \text{if } w_{x,y} \leq 0.5 \\ 1, & \text{if } w_{x,y} > 0.5 \end{cases} \tag{5.13}$$

Following this weight adjustment, the counting algorithm adopts a binary maximum flow approach, similar to the one outlined in Section 4.2.3. In the next section, we will compare the complete weighted method (A1) and this Thresholded Variant against two baseline methods.

5.4.4 Comparison Against Baselines

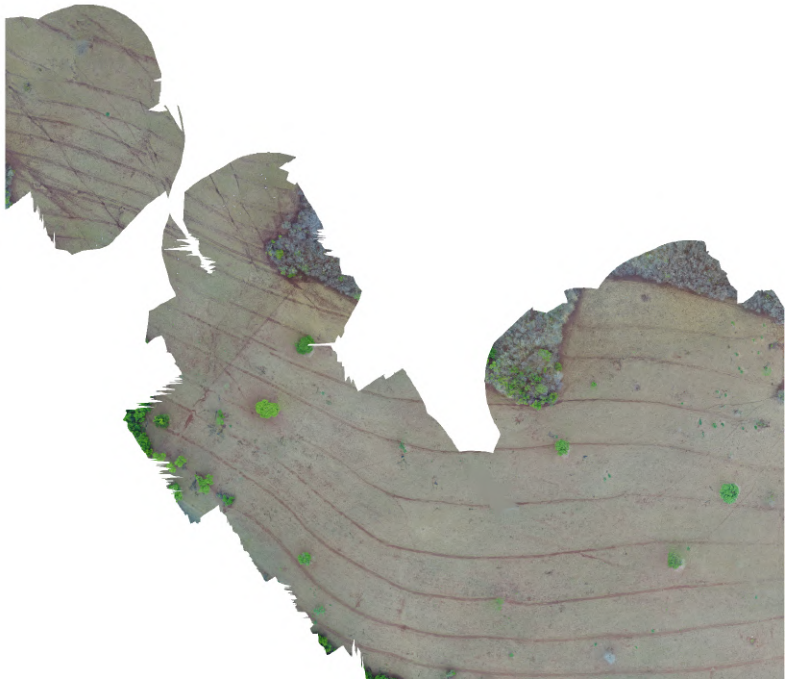
In this section, we present the counting results of our complete cross-validated method (A1), as well as the previously introduced Thresholded Variant. These are pitted against two state-of-the-art techniques. It is important to note that the field of counting cattle in large pastures, particularly regarding duplicate removal, is relatively underexplored, with very limited existing work that address this specific topic. As a consequence, there are currently very few code distributions available for reproduction and comparison.

As a first baseline competitor, we adopted our own implementation of the approach employed in (CHEN *et al.*, 2023). This method involves constructing a mosaic from multiple images for animal (cranes) detection and counting, which uses a proprietary software called PIX4D. Since we do not have access to the proprietary code, we employed a well-known open-source drone mapping tool called OpenDroneMap (ODM, 2020) to generate the bi-dimensional mosaics and count the cattle following the procedures described in the original work (CHEN *et al.*, 2023).

Since our primary focus is on the counting task itself, we generated a bi-dimensional mosaic from all images of each dataset and manually counted the cattle directly on the mosaic, without relying on the detection step. This allowed us to assess the accuracy of using mosaics independently from the detection process and, accordingly, a fair comparison with our approach.

Upon deploying the mosaic-based approach on all 26 datasets, technical limitations

Figure 42 – Unsuccessful mosaic generation for the dataset *Pasto1-15-10-12h*, illustrating gaps in the image due to a shortage of matching points.



Source: Elaborated by the author.

became apparent in three specific datasets (SD_PV_90, Sede-18-10-7h, and Pasto1-15-10-12h), where mosaic generation failed, rendering the baseline counting method infeasible for those datasets. Figure 42 visually illustrates an instance of such a failed mosaic generation attempt for the dataset Pasto1-15-10-12h. As displayed in the figure, the resulting mapping exhibited significant gaps, which is prone to occur due to a shortage of referential matching points in the collection of images.

When images lack distinctive objects that can be used to identify matching points (apart from mostly cleared pasture), construction of the mosaic as guided by these referential points may be compromised. This aligns with observations by the authors in (CHEN *et al.*, 2023), even when using proprietary software. For instance, the authors noted that in areas largely dominated by water, without other observable objects, the software tends to be less accurate.

Another reason that can cause mosaic construction to fail is the lack of enough overlap between images. According to (DANDOIS; OLANO; ELLIS, 2015), an overlap of at least 80% is recommended to produce an accurate mapping, whereas in our approach, we can afford to use much lower overlap levels (30% in our experiments). The use of higher overlap levels is very restrictive in large pastures due to practical constraints such as limited flight time. Adopting high overlap levels requires longer flight plans, leading to increased battery consumption, potentially preventing the acquisition of images in a single flight,

which increases the possibility of cattle movement during the survey, complicating accurate counting. For instance, Soares *et al.* (2021) substantiate this by evaluating that an 80% overlap level reduces the drone's surveyable area per battery charge from 100ha to only 15ha. These challenges highlight the importance of balancing the flight plan parameters and considering the specific characteristics of the pasture areas when utilizing mosaic-based methods for cattle counting.

As our second competitor, we consider our method proposed in Chapter 4. This method also uses the Ford-Fulkerson algorithm for duplicate removal but it is based solely on a fixed distance threshold of 6 meters as a binary attribute used to determine the (unweighted) edges of the bipartite graph. We use the very same manually labeled images given to our current method as input to this baseline.

In the subsequent discussions we refer to our method from Chapter 4 as Graph-based distance (GB_D), our current proposed complete method as Graph-based multi-attribute (GB_{multi}), the Thresholded Variant as Graph-based Thresholded multi-attribute (GBT_{multi})and the mosaic-based baseline following the approach in (CHEN *et al.*, 2023) as "mosaic".

Table 22 displays the absolute and percentage counting errors for GB_{multi}, GBT_{multi}, Mosaic, and GB_D for all 26 datasets. Notably, results marked with an asterisk (*) denote datasets for which counting using the mosaic-based method was infeasible. The table shows that, even if we just ignore these three datasets for the purpose of performance assessment of the mosaic-based method, our GB_{multi} method still performs significantly better, with a total absolute error of 26 counts across 26 datasets. In contrast, the mosaic-based method produced an absolute error of 53 counts across 23 datasets, whereas GBT_{multi} and GB_D exhibited an absolute error of 37 and 42, respectively, across 26 datasets. Considering each dataset individually, our complete method, GB_{multi}, consistently exhibits counting errors that are at least as good as the competitors' across all datasets.

Table 22 shows a 0% error rate across all four methods in 14 out of the 26 datasets. Notably, these datasets, referred to as "motionless" in (SHAO *et al.*, 2020), involve relatively stationary cattle within overlap areas. In such scenarios, all four methods yielded accurate and robust performance.

In contrast, among the remaining 12 datasets, where errors manifest in at least one method, GB_{multi} and GBT_{multi} systematically outperformed Mosaic (except for a single tie in dataset 2A_90) and they also outperformed GB_D in seven and five of these datasets, respectively, while tying with the others.

To assess the statistical significance of the differences between the counting methods, we performed the Wilcoxon signed-rank test for all three pairwise comparisons. Table 23 displays the corresponding p-values, which not surprisingly in this case are all noticeably

Table 22 – Absolute and percentage counting errors for the 4 compared counting methods on the 26 datasets. Results with * indicate datasets where counting could not be performed by the mosaic-based competitor due to unsuccessful mosaic generation.

Dataset	**GT**	**GB$_{multi}$**		**GBT$_{multi}$**		**Mosaic**		**GB$_D$**	
		ABS	**%**	**ABS**	**%**	**ABS**	**%**	**ABS**	**%**
A	**4**	0	0.00	0	0.00	0	0.00	0	0.00
B	**7**	0	0.00	0	0.00	0	0.00	0	0.00
F-1	**8**	0	0.00	0	0.00	0	0.00	0	0.00
F-2	**20**	0	0.00	0	0.00	0	0.00	0	0.00
F-4	**20**	0	0.00	0	0.00	0	0.00	0	0.00
F-5	**14**	0	0.00	0	0.00	0	0.00	0	0.00
G-1	**2**	0	0.00	0	0.00	0	0.00	0	0.00
G-2	**17**	0	0.00	0	0.00	0	0.00	0	0.00
G-3	**25**	0	0.00	0	0.00	0	0.00	0	0.00
G-4	**25**	0	0.00	0	0.00	0	0.00	0	0.00
G-6	**5**	0	0.00	0	0.00	0	0.00	0	0.00
C	**12**	4	33.33	4	33.33	9	75.00	6	50.00
D	**5**	0	0.00	0	0.00	2	40.00	0	0.00
E	**5**	0	0.00	0	0.00	1	20.00	0	0.00
F-3	**24**	0	0.00	0	0.00	0	0.00	0	0.00
G-5	**19**	0	0.00	0	0.00	0	0.00	0	0.00
Dataset2	**6**	0	0.00	0	0.00	0	0.00	0	0.00
SD_PV_90	**5**	0	0.00	0	0.00	*	*	0	0.00
2A_90	**32**	2	6.25	2	6.25	2	6.25	2	6.25
P1_AB_120	**63**	0	0.00	0	0.00	3	4.76	0	0.00
PD_AB_90	**100**	5	5.00	7	7.00	9	9.00	7	7.00
Pasto4-16-10-7h	**109**	5	4.59	6	5.5	13	11.93	7	6.42
Sede-18-10-7h	**22**	1	4.55	1	4.55	*	*	2	9.09
Brejo-19-10-12h	**43**	2	4.65	3	6.98	5	11.63	4	9.30
Pasto2-17-10-7h	**275**	6	2.18	8	2.91	9	3.27	8	2.91
Pasto1-15-10-12h	**239**	1	0.42	6	2.51	*	*	6	2.51
		SUM	**AVG**	**SUM**	**AVG**	**SUM**	**AVG**	**SUM**	**AVG**
TOTAL:		**26**	**2.34**	**37**	**2.65**	**53***	**7.91***	**42**	**3.60**

small. Overall, GB$_{multi}$ significantly outperformed GBT$_{multi}$ and both state-of-the-art competitors.

Table 23 – Wilcoxon signed-rank p-values for comparison of counting methods.

Comparison	**P-value**
GB$_{multi}$ vs. GBT$_{multi}$	0.043
GB$_{multi}$ vs. Mosaic	0.011
GB$_{multi}$ vs. GB$_D$	0.017
GBT$_{multi}$ vs. Mosaic	0.011
GBT$_{multi}$ vs. GB$_D$	0.067
Mosaic vs. GB$_D$	0.011

Despite this overall performance, the results on individual datasets indicate that there are still scenarios that may benefit from further research, such as dataset **C**, where the ground truth contains 12 animals, yet our proposed method GB_{multi} (as the top performer) produced a result that is 4 counts off (an error of 33.33%), which is proportionally high.

The results of our experiments show that while a group of datasets yielded low counting errors across all three methods, other datasets proved themselves more challenging. Interestingly, there is a strong association between the first group and the dataset collection referred to as "motionless" in (SHAO *et al.*, 2020). This suggests that accuracy of cattle counting is closely related to cattle behavior, in addition to the flight plan used for data acquisition. Some of the features of our proposed method, such as modelling the velocity profile of moving cattle, the traversed distances deemed realistic, and whether animals are standing up or laying down, have been designed aiming at explicitly considering cattle behavior into the counting problem. There still more to be done in this direction though, as discussed next.

5.5 Chapter Remarks

In this chapter, we embark on a comprehensive exploration of multi-attribute analysis, a novel approach to enhancing the efficiency of our duplicate removal method. Unlike the preceding chapter, our primary focus here is not on cattle detection but rather the task of cattle counting and duplicate removal. We delve into the intricate world of attribute analysis, considering factors such as state, color, direction, velocity, and distance, all of which prove instrumental in refining our existing methodology.

Furthermore, within this chapter, we introduce an array of new datasets tailored for training and cattle counting. These datasets significantly contribute to the evolution of our methods and the subsequent enhancement of our results.

The upcoming section will feature the conclusions of this thesis, where we will consolidate our findings and provide insights into the culmination of this extensive body of work.

CHAPTER

6

CONCLUSIONS AND FUTURE WORK

In this thesis, the challenge of counting cattle across extensive pasture areas using Unmanned Aerial Vehicles (UAVs) equipped with high-resolution cameras is addressed. Traditional manual counting methods, known for their labor-intensive nature and susceptibility to errors, and existing automated approaches, which often struggle with duplicate animal detections, were the motivation behind this research. Counting cattle in vast landscapes requires meticulous flight planning to capture comprehensive aerial imagery. However, this strategy frequently leads to multiple images of the same cattle, necessitating the development of robust methods for duplicate identification and removal.

Our research endeavors have been fundamentally motivated by the goal of enhancing cattle counting accuracy, particularly in extensive pastures employing geolocated aerial imagery. We successfully introduced innovative methodologies that combine Convolutional Neural Networks (CNNs) with graph-based optimization techniques. Our approach significantly improved cattle counting precision and, importantly, reduced processing times by approximately 15 times. This substantial advancement in efficiency, compared to traditional methods relying on image mosaics to address duplicate removal, underscores the contribution of our work.

Additionally, our research explored the use of CNNs for automated cattle detection in aerial images. Through meticulous model training on real-world images, we have demonstrated the effectiveness of CNNs in accurately detecting cattle within these images, particularly in scenarios with varying backgrounds and cattle colors.

Furthermore, our research has revealed that the problem of duplicate removal in multiple images is under-explored in the literature. We have embarked on an extensive exploration of this aspect by incorporating multi-attribute analysis into our methodology. By incorporating attributes such as velocity, direction, state (lying down or standing), color, and distance, we have significantly refined the matching process. This enhancement

not only contributed to improved duplicate removal but also elevated the overall accuracy of cattle counting. Our in-depth ablation study show the contributions of attributes such as velocity, color, and distance to the system's overall performance. Consequently, our multi-attribute analysis achieved an average error rate of only 2.34% across a diverse collection of 26 datasets, marking a significant advancement in the field.

Moreover, our research legacy extends to the creation of a comprehensive cattle counting benchmark and datasets for training. This benchmark includes a novel image collection and two experimental protocols, designed to enable exhaustive evaluations under real-world scenarios. The training datasets encompass an array of images, each featuring multiple variations. These variations span a range of cattle breeds, colors, backgrounds, lighting conditions, times of the day, altitudes, and distinct scenarios. Some datasets capture individual cattle in cropped shots, while others feature scenes with multiple cattle present in each frame. This thorough curation ensures that our datasets are not only diverse but also faithfully representative, positioning them as invaluable resources for training and rigorously testing cattle counting and classification methods across a multitude of real-world challenges.

In conclusion, we have tackled a substantial challenge in the field of cattle detection and counting, particularly relevant in the context of large pastures. Our innovative methods, which combine CNNs with graph-based optimization and multi-attribute analysis, have significantly improved the accuracy and efficiency of cattle counting in these extensive environments. Our contributions, ranging from the development of novel methodologies to the creation of comprehensive benchmark datasets, mark a significant advancement in this field.

6.1 Future Works and Limitations

While our research has indeed achieved significant milestones, we are cognizant of certain limitations that warrant consideration. Notably, we have not systematically evaluated our methods on datasets characterized by extensive cattle movements, a common occurrence when herds migrate to feeding and hydration areas. To address potential counting inaccuracies in such scenarios, we recommend scheduling counting sessions during periods when cattle predominantly graze. This strategic timing can mitigate the chances of erroneously identifying management-induced movements as duplicate instances.

Future research within our domain offers several promising directions. To further enhance counting precision, prospective investigations could delve into a comprehensive understanding of cattle behavioral patterns. These insights may then be leveraged to inform the development of optimal flight plans, which, in conjunction with the detection and counting method, can evolve into cohesive and effective counting strategies. These strategies may encompass factors such as the time of the day when cattle tend to move

less, preferred flight routes to enhance animal tracking, or adaptive adjustments to flight plans based on real-time cattle movement detection.

Alternative approaches may involve the exploration of variations in image overlap rates, drone flight paths, or even the deployment of multiple drones simultaneously. Moreover, the acquisition of more diverse datasets under a range of real-world conditions provides the opportunity to fine-tune CNN training and investigate conditions that have the potential to further enhance counting methodologies. This multidimensional exploration signifies the exciting and fruitful future prospects in our research field.

Additionally, future work could also explore the application and/or extension (or adaptation) of our method to video data. This extension could facilitate real-time and continuous cattle tracking and counting, adding a layer of practicality and versatility to the proposed approach. In this context, the scope could also expand to include datasets encompassing mixed animals in the same pasture, allowing for the separate counting of various livestock types.

REFERENCES

AHUJA, R. K.; MAGNANTI, T. L.; ORLIN, J. B. **Network flows**. [*S.l.: s.n.*]: Cambridge, Mass.: Alfred P. Sloan School of Management, Massachusetts, 1988.

ALANEZI, M. A. *et al.* Livestock management with unmanned aerial vehicles: A review. **IEEE Access**, v. 10, p. 45001–45028, 2022.

ALSTON, J. M.; PARDEY, P. G. Agriculture in the global economy. **Journal of Economic Perspectives**, v. 28, n. 1, p. 121–46, 2014.

BANK, W. Moving towards sustainability: The livestock sector and the world bank. **World Bank**, March 2021. Available at: https://www.worldbank.org/en/topic/agriculture/brief/moving-towards-sustainability-the-livestock-sector-and-the-world-bank.

BARBEDO, J. G. A.; KOENIGKAN, L. V. Perspectives on the use of unmanned aerial systems to monitor cattle. **Outlook on Agriculture**, SAGE Publications Sage UK: London, England, p. 0030727018781876, 2018.

BARBEDO, J. G. A. *et al.* Counting cattle in UAV images—dealing with clustered animals and animal/background contrast changes. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 20, n. 7, p. 2126, 2020.

BARBEDO, J. G. A. *et al.* A study on the detection of cattle in UAV images using deep learning. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 19, n. 24, p. 5436, 2019.

BARRY, P.; COAKLEY, R. Accuracy of UAV photogrammetry compared with network RTK GPS. **Int. Arch. Photogramm. Remote Sens**, v. 2, p. 2731, 2013.

BEZEN, R.; EDAN, Y.; HALACHMI, I. Computer vision system for measuring individual cow feed intake using RGB-D camera and deep learning algorithms. **Computers and Electronics in Agriculture**, Elsevier, v. 172, p. 105345, 2020.

BOCHKOVSKIY, A.; WANG, C.; LIAO, H. M. Yolov4: Optimal speed and accuracy of object detection. **CoRR**, abs/2004.10934, 2020. Available at: https://arxiv.org/abs/2004.10934.

BROWN, R. G.; HWANG, P. Y. *et al.* **Introduction to random signals and applied Kalman filtering**. [*S.l.: s.n.*]: Wiley New York, 1992. v. 3.

BUDIHARTO, W. *et al.* A review and progress of research on autonomous drone in agriculture, delivering items and geographical information systems (gis). *In*: IEEE. **2019 2nd world symposium on communication engineering (WSCE)**. [*S.l.: s.n.*], 2019. p. 205–209.

BUDIHARTO, W. *et al.* Mapping and 3d modelling using quadrotor drone and gis software. **Journal of Big Data**, Springer, v. 8, p. 1–12, 2021.

BUNDY, A.; WALLEN, L. Breadth-first search. *In*: _____. **Catalogue of Artificial Intelligence Tools**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1984. cap. 25, p. 13–13. ISBN 978-3-642-96868-6. Available at: https://doi.org/10.1007/978-3-642-96868-6_25.

CHAMOSO, P. *et al.* UAVs applied to the counting and monitoring of animals. *In*: RAMOS, C. *et al.* (ed.). **Ambient Intelligence - Software and Applications**. Cham: Springer International Publishing, 2014. p. 71–80. ISBN 978-3-319-07596-9.

CHEN, A. *et al.* Using computer vision, image analysis and uavs for the automatic recognition and counting of common cranes (grus grus). **Journal of Environmental Management**, Elsevier, v. 328, p. 116948, 2023.

CROUSE, D. F. On implementing 2d rectangular assignment algorithms. **IEEE Transactions on Aerospace and Electronic Systems**, v. 52, n. 4, p. 1679–1696, 2016.

DANDOIS, J. P.; OLANO, M.; ELLIS, E. C. Optimal altitude, overlap, and weather conditions for computer vision UAV estimates of forest structure. **Remote Sensing**, Multidisciplinary Digital Publishing Institute, v. 7, n. 10, p. 13895–13920, 2015.

FARJON, G.; HUIJUN, L.; EDAN, Y. Deep-learning-based counting methods, datasets, and applications in agriculture: a review. **Precision Agriculture**, Springer, p. 1–29, 2023.

FELZENSZWALB, P. F. *et al.* Object detection with discriminatively trained part-based models. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 32, n. 9, p. 1627–1645, 2010.

FLORIAN, M.; KLEIN, M. An experimental evaluation of some methods of solving the assignment problem. **INFOR**, v. 8, p. 101–106, 1970.

FRACHTENBERG, E. Practical drone delivery. **Computer**, IEEE, v. 52, n. 12, p. 53–57, 2019.

GEDEON, C. I. *et al.* Identification and counting of european souslik burrows from uav images by pixel-based image analysis and random forest classification: A simple, semi-automated, yet accurate method for estimating population size. **Remote Sensing**, MDPI, v. 14, n. 9, p. 2025, 2022.

GEMERT, J. C. van *et al.* Nature conservation drones for automatic localization and counting of animals. *In*: AGAPITO, L.; BRONSTEIN, M. M.; ROTHER, C. (ed.). **Computer Vision - ECCV 2014 Workshops**. Cham: Springer International Publishing, 2015. p. 255–270. ISBN 978-3-319-16178-5.

GIRSHICK, R. Fast R-CNN. *In*: **Proceedings of the IEEE international conference on computer vision**. [*S.l.: s.n.*], 2015. p. 1440–1448.

GIRSHICK, R. *et al.* Rich feature hierarchies for accurate object detection and semantic segmentation. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2014. p. 580–587.

GIRSHICK, R. *et al.* Rich feature hierarchies for accurate object detection and semantic segmentation. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2014. p. 580–587.

GÓMEZ-CANDÓN, D.; CASTRO, A. D.; LÓPEZ-GRANADOS, F. Assessing the accuracy of mosaics from unmanned aerial vehicle (UAV) imagery for precision agriculture purposes in wheat. **Precision Agriculture**, Springer, v. 15, n. 1, p. 44–56, 2014.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [*S.l.: s.n.*]: MIT Press, 2016. http://www.deeplearningbook.org.

GRAYSON, B. *et al.* GPS precise point positioning for UAV photogrammetry. **The Photogrammetric Record**, Wiley Online Library, v. 33, n. 164, p. 427–447, 2018.

GUO, X. *et al.* Application of UAV remote sensing for a population census of large wild herbivores—taking the headwater region of the yellow river as an example. **Remote Sensing**, Multidisciplinary Digital Publishing Institute, v. 10, n. 7, p. 1041, 2018.

HABCHI, A. E. *et al.* Cga: A new approach to estimate the geolocation of a ground target from drone aerial imagery. *In*: IEEE. **2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS)**. [*S.l.: s.n.*], 2020. p. 1–4.

HE, K. *et al.* Deep residual learning for image recognition. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2016. p. 770–778.

HOWARD, A. *et al.* **Searching for MobileNetV3**. 2019.

HOWARD, A. G. *et al.* Mobilenets: Efficient convolutional neural networks for mobile vision applications. **arXiv preprint arXiv:1704.04861**, 2017.

HUANG, J. *et al.* Speed/accuracy trade-offs for modern convolutional object detectors. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2017. p. 7310–7311.

HUANG, R. *et al.* Multi-uav collaboration to survey tibetan antelopes in hoh xil. **Drones**, v. 6, n. 8, 2022. ISSN 2504-446X. Available at: https://www.mdpi.com/2504-446X/6/8/196.

JIANG, B. *et al.* FLYOLOv3 deep learning for key parts of dairy cow body detection. **Computers and Electronics in Agriculture**, Elsevier, v. 166, p. 104982, 2019.

JOCHER, G. **YOLOv5 by Ultralytics**. 2020. Available at: https://github.com/ultralytics/yolov5.

JOHNSTON, M. G. Ground object geo-location using UAV video camera. *In*: IEEE. **2006 IEEE/AIAA 25TH Digital Avionics Systems Conference**. [*S.l.: s.n.*], 2006. p. 1–7.

JR, L. R. F.; FULKERSON, D. R. **Flows in networks**. [*S.l.: s.n.*]: Princeton University Press, 1962.

KARPATHY, A. *et al.* Large-scale video classification with convolutional neural networks. *In*: **Proceedings of the IEEE conference on Computer Vision and Pattern Recognition**. [*S.l.: s.n.*], 2014. p. 1725–1732.

KELLENBERGER, B.; MARCOS, D.; TUIA, D. Detecting mammals in UAV images: Best practices to address a substantially imbalanced dataset with deep learning. **Remote Sensing of Environment**, Elsevier, v. 216, p. 139–153, 2018.

KILGOUR, R. J. In pursuit of "normal": A review of the behaviour of cattle at pasture. **Applied Animal Behaviour Science**, Elsevier, v. 138, n. 1-2, p. 1–11, 2012.

KORNBLITH, S.; SHLENS, J.; LE, Q. V. Do better imagenet models transfer better? *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2019. p. 2661–2671.

KRISHNA, K. R. **Agricultural drones: a peaceful pursuit**. [*S.l.: s.n.*]: CRC Press, 2018.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *In*: **Advances in neural information processing systems**. [*S.l.: s.n.*], 2012. p. 1097–1105.

KURKUTE, S. *et al.* Drones for smart agriculture: A technical report. **International Journal for Research in Applied Science and Engineering Technology**, International Journal for Research in Applied Science and Engineering ..., v. 6, n. 4, p. 341–346, 2018.

LANG, K. The development of the time-delay neural network architecture for speech recognition. **Technical Report CMU-CS-88-152**, Carnegie Mellon University, 1988.

LI, X. *et al.* Deep cascaded convolutional models for cattle pose estimation. **Computers and Electronics in Agriculture**, Elsevier, v. 164, p. 104885, 2019.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2015. p. 3431–3440.

LONGMORE, S. *et al.* Adapting astronomical source detection software to help detect animals in thermal images obtained by unmanned aerial systems. **International journal of remote sensing**, Taylor & Francis, v. 38, n. 8-10, p. 2623–2638, 2017.

LOSHCHILOV, I.; HUTTER, F. Fixing weight decay regularization in adam. **CoRR**, abs/1711.05101, 2017. Available at: http://arxiv.org/abs/1711.05101.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, p. 115–133, 1943.

ME, S. M. *et al.* Quadcopter uav based fertilizer and pesticide spraying system. **Int. Acad. Res. J. Eng. Sci**, v. 1, n. 2016, p. 8–12, 2016.

MELLO, R. F. de; PONTI, M. A. Statistical learning theory. **Machine Learning**, Springer, p. 75–128, 2018.

MEYES, R. *et al.* **Ablation Studies in Artificial Neural Networks**. 2019.

MILLER, M. L.; STONE, H. S.; COX, I. J. Optimizing murty's ranked assignment method. **IEEE Transactions on Aerospace and Electronic Systems**, IEEE, v. 33, n. 3, p. 851–862, 1997.

MOHAN, A.; RAJU, R. D.; JANARTHANAN, P. Animal disease diagnosis expert system using convolutional neural networks. *In*: IEEE. **2019 International Conference on Intelligent Sustainable Systems (ICISS)**. [*S.l.: s.n.*], 2019. p. 441–446.

MORGAN, D.; FALKNER, E. **Aerial mapping: methods and applications**. [*S.l.: s.n.*]: CRC Press, 2001. v. 2.

ODM, O. A. **A command line toolkit to generate maps, point clouds, 3D models and DEMs from drone, balloon or kite images**. 2020. OpenDroneMap/ODM GitHub Page. Available at: https://github.com/OpenDroneMap/ODM.

PENATTI, O. A.; NOGUEIRA, K.; SANTOS, J. A. D. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition workshops**. [*S.l.: s.n.*], 2015. p. 44–51.

PEREZ, L.; WANG, J. **The Effectiveness of Data Augmentation in Image Classification using Deep Learning**. 2017.

PONTI, M. A. *et al.* Everything you wanted to know about deep learning for computer vision but were afraid to ask. *In*: IEEE. **2017 30th SIBGRAPI conference on graphics, patterns and images tutorials (SIBGRAPI-T)**. [*S.l.: s.n.*], 2017. p. 17–41.

PONTI, M. A. *et al.* Training deep networks from zero to hero: avoiding pitfalls and going beyond. *In*: IEEE. **2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. [*S.l.: s.n.*], 2021. p. 9–16.

QIAO, Y.; TRUMAN, M.; SUKKARIEH, S. Cattle segmentation and contour extraction based on mask R-CNN for precision livestock farming. **Computers and Electronics in Agriculture**, Elsevier, v. 165, p. 104958, 2019.

RACHMAWATI, S. *et al.* Application of drone technology for mapping and monitoring of corn agricultural land. *In*: **2021 International Conference on ICT for Smart Society (ICISS)**. [*S.l.: s.n.*], 2021. p. 1–5.

REDMON, J.; FARHADI, A. YOLO9000: better, faster, stronger. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2017. p. 7263–7271.

REJEB, A. *et al.* Drones in agriculture: A review and bibliometric analysis. **Computers and Electronics in Agriculture**, v. 198, p. 107017, 2022. ISSN 0168-1699. Available at: https://www.sciencedirect.com/science/article/pii/S0168169922003349.

REY, D.; NEUHÄUSER, M. Wilcoxon-signed-rank test. *In*: _____. **International Encyclopedia of Statistical Science**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. cap. 606, p. 1658–1659. ISBN 978-3-642-04898-2. Available at: https://doi.org/10.1007/978-3-642-04898-2_616.

RIJSBERGEN, C. J. V. Foundation of evaluation. **Journal of documentation**, MCB UP Ltd, 1974.

RUSSAKOVSKY, O. *et al.* Imagenet large scale visual recognition challenge. **International Journal of Computer Vision**, Springer, v. 115, n. 3, p. 211–252, 2015.

SANDLER, M. *et al.* **MobileNetV2: Inverted Residuals and Linear Bottlenecks**. 2019.

SARKAR, D.; BALI, R.; SHARMA, T. **Practical Machine Learning with Python**. [*S.l.: s.n.*]: Springer, 2018.

SARWAR, F. **Robust Livestock Detection and Counting Using an Unmanned Aerial Vehicle (UAV)**. 2022. Tese (Doutorado) — Auckland University of Technology, 2022.

SARWAR, F. *et al.* Detecting sheep in uav images. **Computers and Electronics in Agriculture**, Elsevier, v. 187, p. 106219, 2021.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding machine learning: From theory to algorithms**. [*S.l.: s.n.*]: Cambridge university press, 2014.

SHAO, W. *et al.* Cattle detection and counting in UAV images based on convolutional neural networks. **International Journal of Remote Sensing**, Taylor & Francis, v. 41, n. 1, p. 31–52, 2020.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SOARES, V. H. A. *et al.* Cattle counting in the wild with geolocated aerial images in large pasture areas. **Computers and Electronics in Agriculture**, Elsevier, v. 189, p. 106354, 2021.

SUTSKEVER, I. *et al.* On the importance of initialization and momentum in deep learning. *In*: **International conference on machine learning**. [*S.l.: s.n.*], 2013. p. 1139–1147.

SZEGEDY, C. *et al.* Inception-v4, inception-resnet and the impact of residual connections on learning. *In*: **AAAI**. [*S.l.: s.n.*], 2017. v. 4, p. 12.

SZEGEDY, C. *et al.* Going deeper with convolutions. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2015. p. 1–9.

SZEGEDY, C. *et al.* Rethinking the inception architecture for computer vision. **CoRR**, abs/1512.00567, 2015. Available at: http://arxiv.org/abs/1512.00567.

SZELISKI, R. **Computer vision: algorithms and applications**. [*S.l.: s.n.*]: Springer Science & Business Media, 2010.

TRAKOOLWILAIWAN, T. *et al.* Convolutional neural network for high-accuracy functional near-infrared spectroscopy in a brain–computer interface: three-class classification of rest, right-, and left-hand motor execution. **Neurophotonics**, v. 5, p. 5 – 5 – 15, 2017. Available at: https://doi.org/10.1117/1.NPh.5.1.011008.

TRIPPI, R. R.; ASH, A. W.; II, J. V. R. A mathematical approach to large scale personnel assignment. **Computers & Operations Research**, Elsevier, v. 1, n. 1, p. 111–117, 1974.

TZUTALIN. **LabelImg**. 2015. Git code. Available at: https://github.com/tzutalin/labelImg.

VAYSSADE, J.-A.; ARQUET, R.; BONNEAU, M. Automatic activity tracking of goats using drone camera. **Computers and Electronics in Agriculture**, v. 162, p. 767–772, 2019. ISSN 0168-1699. Available at: https://www.sciencedirect.com/science/article/pii/S0168169918312894.

VEROUSTRAETE, F. The rise of the drones in agriculture. **EC agriculture**, v. 2, n. 2, p. 325–327, 2015.

WANG, N.; YEUNG, D.-Y. Learning a deep compact image representation for visual tracking. *In*: **Advances in neural information processing systems**. [*S.l.: s.n.*], 2013. p. 809–817.

WEBER, F. de L. *et al.* Recognition of pantaneira cattle breed using computer vision and convolutional neural networks. **Computers and Electronics in Agriculture**, Elsevier, v. 175, p. 105548, 2020.

WEBER, F. de L. *et al.* Counting cattle in uav images using convolutional neural network. **Remote Sensing Applications: Society and Environment**, Elsevier, v. 29, p. 100900, 2023.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data Mining: Practical Machine Learning Tools and Techniques**. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN 0123748569, 9780123748560.

WOLF, P.; DEWITT, B.; WILKINSON, B. **Elements of Photogrammetry with Application in GIS, Fourth Edition**. McGraw-Hill Education, 2013. ISBN 9780071761116. Available at: https://books.google.com.br/books?id=bCx5rmWMHyAC.

World Bank. **World development indicators 2012 (English)**. 2012. World development indicators. Washington, DC: World Bank. Available at: http://documents.worldbank.org/curated/en/553131468163740875/World-development-indicators-2012.

WU, C. *et al.* Multicore bundle adjustment. *In*: IEEE. **CVPR 2011**. [*S.l.: s.n.*], 2011. p. 3057–3064.

XU, B. *et al.* Automated cattle counting using mask R-CNN in quadcopter vision system. **Computers and Electronics in Agriculture**, Elsevier, v. 171, p. 105300, 2020.

XU, B. *et al.* Livestock classification and counting in quadcopter aerial images using mask R-CNN. **International Journal of Remote Sensing**, Taylor & Francis, p. 1–22, 2020.

XUE, Y.; WANG, T.; SKIDMORE, A. K. Automatic counting of large mammals from very high resolution panchromatic satellite imagery. **Remote sensing**, Multidisciplinary Digital Publishing Institute, v. 9, n. 9, p. 878, 2017.

ZOPH, B. *et al.* Learning transferable architectures for scalable image recognition. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2018. p. 8697–8710.