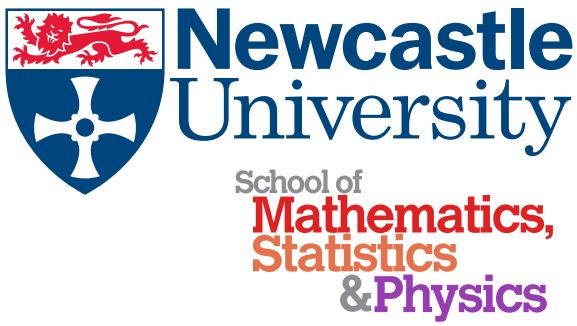
Constraining Models of Collective Motion in Biological Systems

Hayley Elizabeth Louise Moore

Thesis submitted for the degree of
Doctor of Philosophy



School of Mathematics, Statistics & Physics
Newcastle University
Newcastle upon Tyne
United Kingdom

October 2021

Acknowledgements

Firstly I would like to thank my supervisors Andrew Baggaley and Colin Gillespie for the opportunity to undertake this work. Thank you also to the supportive staff in the CDT for Cloud Computing for Big Data, Jen Wood and Paul Watson in particular. Thanks should also go to the wonderful IT staff in the maths department: Michael Beaty and John Nicholson. I would also like to thank Nigel Hill and his family for letting me film his sheep on a number of occasions.

I want to thank all the friends I have made in the maths department over the last 10 years: Clarissa Barratt, Robbie Bickerton, Tom Bland, Marios Bounakis, Kieren Charles, David Cushing, Can Evirgen, Francesca Fedele, Tom Fisher, Sophie Harbisher, Emma Johns, Emma Jones, Sarah Jowett, Scott Lilico, Joe Matthews, Mae Mezgarneshad, Abby Miller, Robert Pattinson, David Robertson, Em Rickinson and Vicki Smith. Special mention to the original MathBio group: Jack Walton and Laura Wadkin for making our trips to Nottingham, Bath and Montreal so much fun.

Outside of university there are a number of people I would like to thank. A big thanks to Tasha Hall and Rafe Skippings. They have stuck by me since I was 9 and without their continued support and laughter I would not be the person I am today. Thanks also to Tom Boyle, Emma Burrow, The Fieldings, Cory Fowden, Imogen Fowler, Zoe Harrison, Gavin Logie and The Staggs for giving me encouragement and laugher when I most needed it. Thanks also to Helen and Nirmal Seth for sparking my interest in computers.

To my parents, Grandma and Grandpa, Bethan, Georgia, Lucy and Freddie: Thank you for all the love and laughter. Thank you for all the Sunday dinners, long phone calls and video game sessions.

Finally, I give my thanks to George Stagg, for his vast amount of love, patience, support and understanding. I can't wait to start the next chapter!

Abstract

Animals moving together as one is a commonly seen spectacle in both the sky, with flocks of birds, and in the oceans, with school of fish. Mathematical models have been developed over the last 50 years to gain a deeper understanding into how such coordination occurs or to recreate the behaviour digitally. There has been extensive numerical simulation and analysis done for these models but little comparison to actual data. This is due to the complexity of obtaining high quality data suitable for analysis.

We were able take advantage of lightweight high definition cameras and drone technology to collect footage of collective behaviours. In this thesis we describe a computer vision algorithm we devised to detect and track individual sheep in the drone footage we collected. The algorithm emphasises the differences in the colours of the sheep and the grass background in order to locate the sheep. It then tracks the individuals throughout the video. In total the trajectories of 45 or more sheep were extracted from 14 videos ranging from 150 frames to 593 frames. In some of these videos the quadbike and farmers herding the sheep were also tracked. From these trajectories we were able to extract quantities such as average speed and global alignment which can then be used to compare to simulated data.

We describe a number of models from the literature which aim to reproduce the types of behaviours we observed in our sheep flocks and some of these we expand on to make them include new features such as allowing agents speeds to change or allow agents to interact with a predator whist in an enclosed area.

We go on to compare our observational data to two different types of these models. The first of these was a family of models which were able to replicate the emergent flocking behaviour seen in some of the observations. The second was a model able to simulate data to compare to our observations of "steady-state" flocking as well as being able to include the movement of the quadbike or farmer herding the animals. We will compare our observational data to simulated data using an approximate Bayesian computation rejection scheme to calculate an approximate joint posterior distribution for the parameters in each of the models. The parameters of these models were sampled from a Latin hypercube meaning we are able to cover the full parameter space efficiently.

Contents

III Models And Data Comparisons

List of Tables

List of Algorithms

List of Figures

I

Introduction

1

Introduction

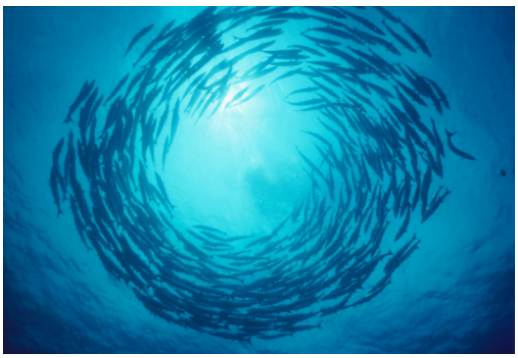
1.1 Collective Animal Motion

People have always been fascinated by the displays of animals moving collectively. In ancient times, the Greeks and Romans practised ornithomancy and augury, the practice of reading omens from the flights of birds [1, 2]. These aggregations of groups happen across organisms of all size, from bacteria colonies up to pods of whales [3, 4]. The behaviour is seen across virtually all types of habitat, species, degrees of mobility and many more biological traits [5, 6]. Some examples demonstrating this wide range of scales are the rotating of ant colonies, roosting starlings producing breathtaking aerial displays (Figure 1.1a), and herds of wildebeest stampeding across the African plains, (Figure 1.1b). It is particularly interesting when groups self-organise into complex patterns with no external stimulus. School of fish move in an orderly fashion but also change direction amazingly abruptly. They can also swirl so frantically that they confuse any nearby predators [4], Figure 1.1c. Looking closer to home we also see collective motion in humans. In busy streets people spontaneously form "traffic lanes" to allow people to pass more easily [7], Figure 1.1d. Knowledge of this behaviour can be used to influence the design of new shopping centres and other public places. For example, in places like train stations and airports there is often a barrier separating the left and right side of a corridor to encourage the formation of “traffic lanes” in humans and passively control the crowds that pass through these areas.

Physical aggregation can be thought of as part of a continuous scale of group interaction [4]. At one end of the scale you have individuals who are highly integrated, they develop long-term relationships with other members of the group and have a high rate of information passed between individuals, both directly and indirectly. At the other end of the scale are territorial animals who have little need to transfer information or to develop group structure. Dolphin pods, honeybees, and humans are examples of highly integrated groups [8] and tigers [9], European badgers [10] and real lemurs [11] are examples of animals who reside in groups which are less integrated. In the highly integrated groups, pathways of

(a) A flock of starling creating spectacular aerial displays [12].

(b) A herd of wildebeest migrating across the Serengeti in Tanzania [13].



(c) A school of fish swimming in a torus shape [14].

(d) A crowd of tourists in Bavaria, south east Germany [15].

Figure 1.1: A selection of images showing collective behaviour on ground, in the air and underwater

long-term communications are often established between known individuals (which could be clones, siblings or other reciprocating members) and at designated locations (these could be the hive, the breeding grounds or for humans at the family home).

There exists a third type of group interaction which lies between these two extremes. Parrish et al. [4] claimed that this type of interaction should be considered "prototypical" animal aggregations as it includes behaviours seen in herds, swarms, flocks and schools. Over 50% of bird species form feeding flocks [16] and over 50% of species of fish display coordinated schooling at some point in their lives, and an even higher percentage exhibit coarser aggregation [17]. These "prototypical" animal aggregations display coordinated motion, but individual members of the group generally never form long lasting relationships and are often unrelated to one another. The number of members in these groups can vary greatly in size, ranging from a number as small as 12, in sheep flocks [18], up to billions of individual herring in a single school [19]. The individuals in large groups are likely to interact with only a small proportion of the total, forming a neighbourhood around themselves in which they interact. The number of individuals they interact with

could be fixed, or depend on the destiny of individuals within the neighbourhood. In some species individuals react to the sound emitted by the group [20]. Groups have been observed forming repeated arrangements similar to those seen in crystalline structures. It is thought that these structures maximise sensory contact between individuals, reflecting the ambient conditions and the organisms' shape [4, 21]. These group-level characteristics are generally believed to have significant biological consequences on the wellbeing of its group members [5], such as the success of foraging [22, 23], defence against predation [24–27], and the success of reproduction [16, 28].

It is this "prototypical" group interaction that this thesis will be exploring further. Here we will observe and analyse the behaviour of medium sized ($n \approx 40$) flocks of sheep which are reacting to an external stimulus, such as a quadbike or a farmer. The sheep start with the appearance of incohesion, however, once the sheep become aware of the quadbike or farmers they quickly coalesce and form a group which displays synchronous and coordinated movement. This transition from incohesion to cohesion is an example of a type of "prototypical" aggregation and is known as emergent flocking behaviour. Emergent flocking has been observed in many of the species known to flock in these types of aggregations [4]. Another type of "prototypical" aggregation that we will be analysing is steady-state flocking. This is the behaviour of a group when acting together as one.

1.2 Mathematical models

The modelling of flocking behaviour has been attempted by a wide range of communities, from computer graphics specialists, biologists and physicists [29]. Mathematical modelling is an integral part of exploring the connection between properties of the individual and properties of the group as a whole. Using models, researchers are not only able to test interaction mechanisms and observe the resulting group behaviour but they can also use these models to program autonomous drones. Although particular implementations may differ dramatically, interactions are generally modelled as a combination of any or all of the following three influences: repulsion away from individuals which are too close, attraction to individuals which are far away, and alignment with neighbours which are close by.

Emergent collective motion as described above is often studied using self-propelled particle models. The self-propelled particle models are a subset of agent-based models where the term "agent-based" means that the model represents the individuals of a swarm or flock as separate entities [30, 31]. This is in contrast to continuum models [32–34] and those that coarse-grain the behaviour of the individuals to reach the continuous limit [31, 34, 35]. In this thesis we will refer to self-propelled particle models as agent-based models for simplicity.

Agent-based models are incredibly versatile. The models are not limited to modelling a single species of animal nor are they limited to a specific research area or modelling a single scenario. For example, in 2015 Carter et al. [9] described the use of an agent-based model which aims to aid in the conservation of tigers (*Panthera tigris*); in 2002, Helbing et al. [36] discussed an agent-based model approach to help understand the forces on pedestrian crowds in evacuation situations; and in 2015 Choudhary et al. [37] used agent-based modelling to see the effect of inertia on schools in a turbulent flow.

In the agent-based models we consider, individuals are reduced to featureless points which change their direction of motion and speed in response to others' and their surroundings according to a set of fixed rules [38]. One such simple rule could be a repulsion force acting on two agents if they were to get too close to one-another. While complex interactions can be defined as combinations of a number of simple rules, remarkably, it is known that simple rules of interaction are sufficient to produce complex emergent behaviour [39–41].

Each of these rules are assumed to take a particular form, sometimes changing as a function of a parameter. These parameters can then be estimated by using experimental data [3, 30–34, 42]. In the case of the repulsion force mentioned earlier there are two common forms that this rule can take. The first of these is if two agents get too close together, a smaller distance than R, the repulsion force changes their direction of motion to point directly away from its neighbour [42–44]. Whereas the other approach is to allow the force to be continuous so that the closer two agents get to one another the greater the magnitude of the repulsion force [45]. By analysing experimental data it is possible to see which of these approaches is more appropriate to recreate the observed data. It is also possible to infer from observational data appropriate values that the parameter R should take.

1.2.1 Continuum Models

One of the drawbacks of using agent-based models is that they typically have to be studied numerically. In contrast continuum models may allow us to gain a deeper understanding of the mathematics that underpins collective motion. Continuum models give a macroscopic description of the flocking behaviours seen in the "prototypical" aggregations. Rather than using a perhaps more obvious coarse-grained hydrodynamical approach, in the late 1990s Toner and Tu [46, 47] instead derived the most general continuum equations of motion for a velocity field and density consistent with the conservation laws and symmetries of their problem. Using this setup, they introduced phenomenological parameters whose numerical values depended on the observed microscopic behaviours. Unlike agent-based models each of the terms describing the large-scale behaviour only depend on the conservation laws and symmetries and do not depend on the rules for the microscopic phenomena.

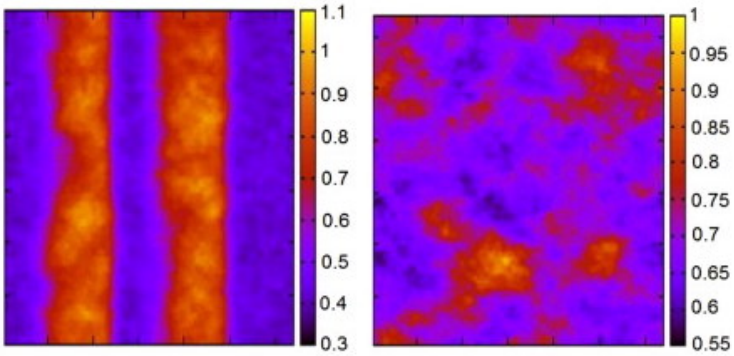


Figure 1.2: Results of numerical simulations of the Bertin hydrodynamic model [29] showing a) A polarised density profile demonstrating striped flocking motion. b) A density profile in the transient phase showing fluctuating flocking.

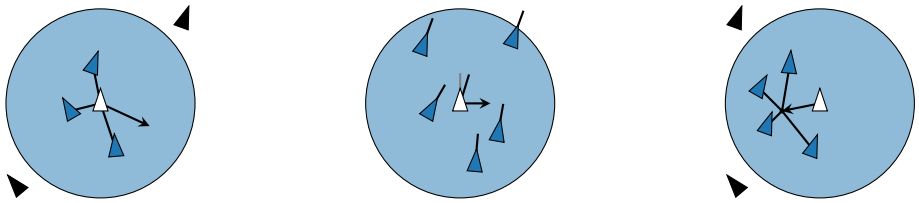
Bertin et al. [48] made a significant step in the area of continuum models of collective motion when they derived hydrodynamic equations for the velocity field and density of a gas of self-propelled particles which had binary interactions, Figure 1.2. These interactions came from the corresponding microscopic rules. The results of numerical simulations of this model closely agreed with the standard model of self-propelled particles.

A primary difference between agent-based and continuum models is the fact that the degree of aggregation is defined over the entire 2D or 3D field of interest, whilst agent-based models are point-based where simple properties of the system are defined only on particles. Agent-based models are much more flexible in terms of their interaction rules, and also simpler to integrate into statistical inference. Therefore, given that observational data in the form of (x, y)-coordinates are already in the correct form for comparison with agent-based models in this thesis we will not investigate continuum models.

1.2.2 Agent-Based Models

In contrast to continuum models, agent-based models describe the microscopic rules which result in flocking behaviours. One of the earliest well-known flocking simulations was published by Reynolds [43] who was motivated by the visual spectacle of coherently flying birds. In his model bird-like objects, which he referred to as "boids", moved according to a set of differential equations which took into account three different types of interactions:

1. Avoiding collisions.
2. Heading in the same direction as its neighbours.
3. Staying close to the centre of mass of the flock.



(a) Rule 1: Avoid collisions.

(b) Rule 2: Head in the same direction as neighbours.

(c) Rule 3: Stay close to the centre of mass of the flock.

Figure 1.3: The three microscopic rules in the Reynolds "boids" model. The △ individual interacts with its neighbours ▲ within a set distance, shown here by the pale blue circle. The remainder of the neighbours, ▲, are outside of this area and are not interacted with. The current direction of travel of these "boids" are shown by a short straight line from the tip of the triangle. The new direction of motion of the △ is shown by a black arrow.

Reynolds' model uses similar microscopic rules to an earlier model by Aoki [42]. A schematic of these three rules can bee seen in Figure 1.3. The speed and direction of each agent in Reynolds' model is stochastic but the direction of motion is influenced by the location and direction of its neighbours. Building on the work of Aoki [42] and Reynolds [43] in 1995, Vicsek et al. [38] established a quantitative interpretation of large flock in the presence of perturbations. This model is widely known as the "Vicsek Model" cf. Cucker and Smale [49], Ha et al. [50], Yates et al. [51], Kattas et al. [52], Khurana and Ouellette [53], Baggaley [54] and Armbruster et al. [55]. Unlike the Reynolds' model for "boids" where the speed is allowed to vary, in the standard Vicsek model particles move with fixed speed in the average direction of its neighbours within a fixed distance. The Vicsek model is investigated in more detail in Chapter 4. This simple model allows the simulation of hundreds of flocking agents and by varying some of the parameters or initial conditions it is possible to recreate a vast range of collection motion behaviours, such as mills, rotating chains, bands and more [29].

The model by Vicsek et al. [38] has been generalised over the years. In 2002 Couzin et al. [44] and in 2007 Cucker and Smale [49] the simple two-dimensional Vicsek model was expanded into higher dimensions and Cucker and Smale [49] allowed the interaction between agents to decay as the distance between them increased. It was then generalised further by Ha et al. [50] in 2010. Other modifications of the Vicsek model have been to make the model more applicable to certain scenarios: Yates et al. [51] introduced a leader/follower dynamic which is able to reproduce behaviours seen in pack animals where there is hierarchical structure to the flock. In these situations when the leader moves the others will follow; Khurana and Ouellette [53] and Baggaley [54] replaced the normal noise term with noise similar to that of a realistic turbulent flow; Armbruster et al. [55] enclosed the swarm with hard boundaries. Changes to the model were also made after experimental data was collected: Kattas et al. [52] observed the movement of homing

Figure 1.4: The human eye can easily separate the daffodil from the background, whilst being able to detect the shape of each of its petals [57].

pigeons and modified the model accordingly whereas Ginelli et al. [56] observed sheep grazing in a field the model was modified in a different manner.

1.3 Computer Vision

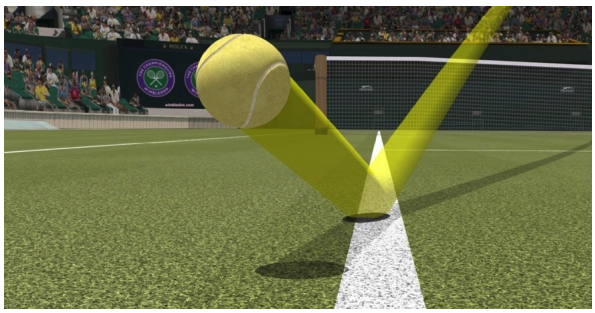
Having discussed mathematical models of collective motion we shall now focus on computer vision techniques, which provide background for work presented later in this thesis. As humans we are easily able to interpret the three-dimensional structures of the world we live in. When presented with an object we know its shape and any areas of translucency from the light and shading patterns across its surface, Figure 1.4. Taking advantage of biological features such as binocular vision, we can also easily separate the object from its background. When looking at a group photograph, we can easily count the number of individuals present and name each of them. Furthermore, we are even able to speculate the emotions of each person by studying their facial expressions and body language. Even though perceptual psychologists are able to devise optical illusions to investigate some of the principles of how we perceive the world, a full model of the complex human visual system remains elusive.

Computer vision researchers have developed mathematical techniques to recreate 3D shapes and their appearance from an image. There are now reliable methods for creating a simple 3D model of an environment from large numbers of partially overlapping photographs, Figure 1.5b [58]. Using stereo matching (the process of finding the pixels in the different views that correspond to the same 3D point in the scene) we can take a large number of photographs of a building taken at different views and under different lighting conditions to create a detailed 3D surface model of the building, Figure 1.5d [59]. Computer vision is

now widely used in movie production in Hollywood, with the use of motion capture techniques [60, 61] to translate the movement of actors to computer animation, Figure 1.5c. However, despite these advances made over the last 50 years, we are still struggling to get a computer to interpret an image better than a toddler could. Szeliski [62] states that this difficulty is due to vision being an "inverse problem", where we seek to discover unknowns with insufficient information to specify any solution.

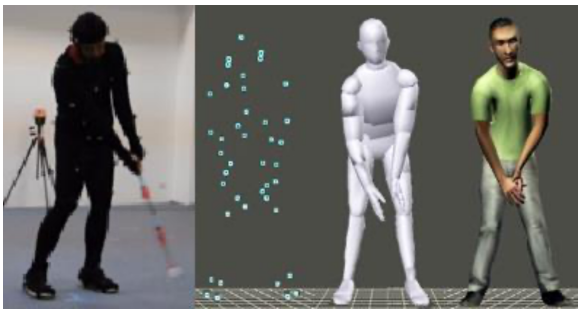
Some computer vision methods are simply trying to recreate what we can see and are usually developed using physics and computer graphics. These fields of research focus on modelling one small aspect of our 3D environment: how objects move, how light scatters in the atmosphere, or how light refracts in a camera lens. The results of computer graphics models are often not perfect; animated films such as *Final Fantasy: The Spirits Within* and *The Polar Express* are often referred to as being in "uncanny valley" due to the animated humans imperfectly resembling a human, provoking feelings of uneasiness of revulsion [64, 65].

However, more often than not, the phrase "computer vision" is not used to refer to the above process. It is used to refer to the process of describing the contents of an image.



(a) The recreated trajectory of a tennis ball using Hawk-Eye technology [63].

(b) A simple 3D model of Notre Dame Cathedral, Paris [58].



(c) A person being tracked using motion capture methods and the final animation [61].

(d) A detailed 3D model of the Statue of Liberty, New York [59].

Figure 1.5: A selection of images showing different computer vision techniques, including creating 3D models from multiple photographs, and motion capture methods.

Whilst humans and animals can do this naturally with great ease, computer algorithms are error prone or have low accuracy [62]. When computer vision first started in the 1960s a professor at MIT expected his undergraduate to "spend the summer linking a camera to a computer and getting the computer to describe what it saw" [66]. Researchers are still working on solving this problem over 50 years later.

Computer vision is now being used regularly in a variety of applications

- **Motion Capture**: As mention previously computer vision is used regularly in motion capture techniques so that an actor's movements can be recreated in computer animations [60, 61], Figure 1.5c.
- **Optical Character Recognision**: This reads postcodes on handwritten envelopes to then reduce the amount of manual sorting needed [67]. OCR is also used in number plate recognition (ANPR) [68].
- **Match Move**: This process allows computer generated imagery to be placed into live action films between the background and the foreground. This is used in films such as *Jurassic Park* and *The Curious Case of Benjamin Button* [69, 70].
- **Sports**: Computer vision is used to support decision making in critical sports matches. In big tennis matches Hawk-Eye is often used to determine whether or not the ball landed in or not, Figure 1.5a [71, 72]. The technology has recently started being used in football matches as well.

From the list above we can see that the number of applications for computer vision is vast. It is now starting to be used more widely in data collection to be able to compare the behaviours of different animals with simulations. Lukeman et al. [73] used computer vision methods to identify and track a flock of surf scooters on the surface of a body of water and Croze et al. [3] used it to calculate the density of algae passing through a region of interest in a pipe flow. In Chapter 2 we will use similar methods to firstly identify individual sheep in frames of a video, and then track the individuals through time. The methods we will use to identify the sheep are similar to those used in image segmentation techniques when counting cells in a petri dish [74–76] or detecting the nucleus of cells [77].

We took inspiration from previously published image segmentation techniques and extended them to include novel processing steps based on statistical methods. Further discussion of the relevant computer vision literature and details about the methods introduced in this thesis can be found in Chapter 2. Our new methods reduce the noise in the detected locations and reduce the difficulty in tracking over long periods of time.

The advantage of creating a new algorithm rather than using currently available (both commercial and free) software [78–81] is that we are able to fix the number of sheep we

wish to detect at the beginning of the process and the algorithm using previous information keep keep this number constant.

1.4 Thesis Overview

This section gives an overview of the structure of this thesis, as well as a short description of each of its chapters. This thesis is split into four main parts.

Part I - Introduction

Chapter 1 introduces the idea of collective animal motion in nature and different approaches to mathematically modelling the observed behaviour. We also give a brief overview of the history of computer vision and its applications.

Part II - Data Collection With Computer Vision

Part II focuses on the extraction of observational data from a number of drone videos and the observational data itself.

Chapter 2 starts by describing the location we chose to film sheep being herded and the equipment we used to film them. It then goes into depth about the decisions we made creating our computer vision algorithm to locate and track sheep through each of the drone videos.

Chapter 3 then explores the information we were able to infer from the information extracted from the drone footage.

Part III - Models And Data Comparisons

Part III introduces a number of mathematical models which can recreate some of the behaviour seen in our observational data.

Chapter 4 outlines some agent-based models from the literature which can simulate behaviours seen in sheep flocks which are being herded. In this chapter we propose a number of modifications to the existing models and combine two existing models to make them more applicable for comparison with the data we collected.

In Chapter 5 we simulate data from modification of the well-known Vicsek model for collective motion. Using an ABC rejection scheme, we compare the simulated data to the observational data in order to determine the posterior distribution of the parameters for each modification.

Similarly, in Chapter 6 we simulate data from the combined sheep model which approximates the behaviour of sheep in the presence of a predator with solid boundaries. As before using an ABC rejection scheme we compare the simulated data to the observational data to determine the posterior distribution for the parameters in this new model.

Part IV - Conclusions and Appendices

Chapter 7 contains the conclusions and main results of this thesis. It also describes a number of scenarios that would be good for future work.

Appendix A defines a number of mathematical properties which are used in the thesis. It also describes and shows standard proofs of a number of numerical time stepping methods. Appendix B illustrates how to calculate the Mahalanobis distance as a way of separating the background of an image from the foreground. Appendix C contains the trajectory plots for each of the captured drone videos. Appendix D contains a table of notation and symbols used throughout this thesis.

II

Data Collection With Computer Vision

2

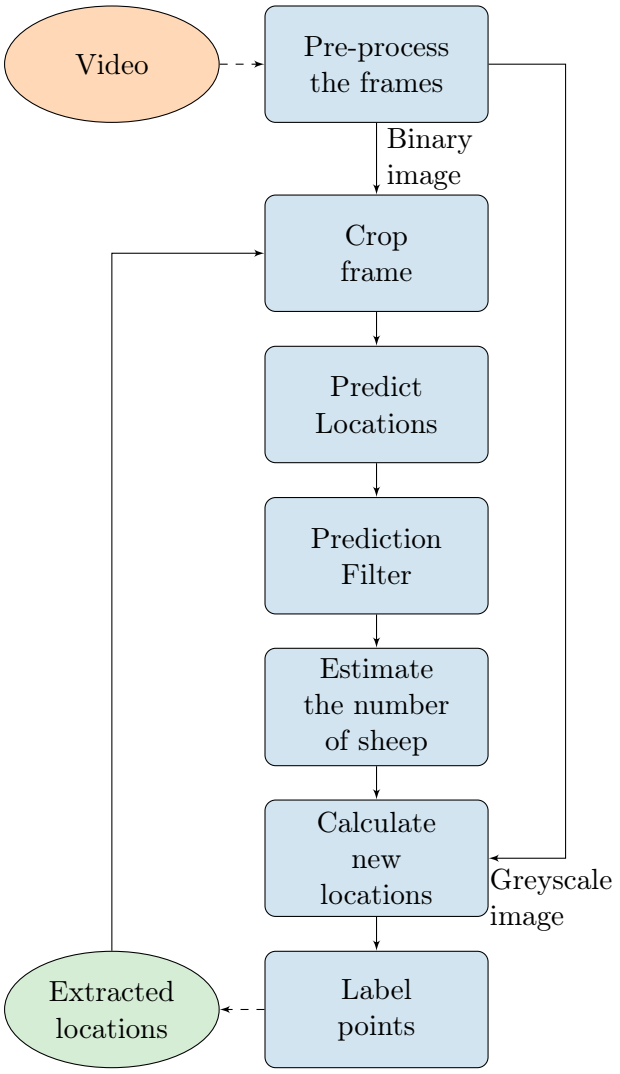
Detecting And Tracking The Location Of Sheep Using Drone Footage

2.1 Introduction

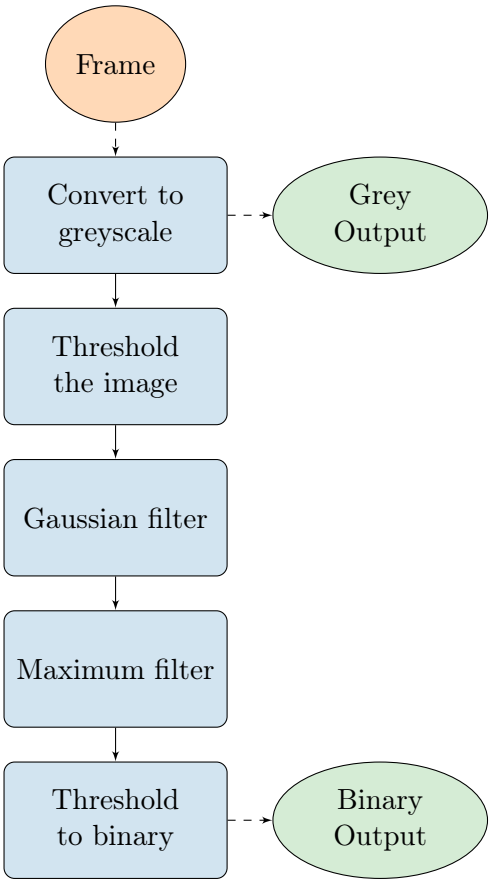
In order to evaluate how well a mathematical model mirrors real life, it is necessary to compare the simulations to real biological systems. The data we need in order to do this comparison must be in the form of (x, y) coordinates of animals over time, since this is the form of the data from simulations. In this chapter we will go through the steps we took to collect our own data; the process of filming a flock of sheep being herded by a quadbike using a drone and then extracting the coordinates of the sheep using computer vision.

Computer vision is a way of automating tasks that the human visual system can do [82]. The algorithm we used to do the extraction comprises of three stages; the pre-processing stage is a combination of standard image manipulation processes, the second stage we employ novel image processing steps that requires knowledge from the previous frames, the final stage is the extraction stage where we extract the location of the sheep in each frame. The entire process is demonstrated in Flowcharts 2.1 and 2.2, this chapter will go through each stage individually starting with each step in the pre-processing stage.

A similar process was used by Lukeman et al. [73] to extract the locations of surf scooters on the surface of a body of water, however in their method using a camera at an angle over the water some surf scooter go in and out of shot and hence are missing in the dataset for that period of time. A similar issue was found in the method used by Ginelli et al. [56]. Therefore their methods were not sufficient for us as we want to know the locations of the sheep at all times. Croze et al. [3] used computer vision techniques to determine the density of algae passing through a pipe, however their method did not result in (x, y)-coordinates of individual agents over time and hence that method was not applicable in our situation as we require the full trajectories of our agents.



Flowchart 2.1: Overview of sheep tracking algorithm.



Flowchart 2.2: Pre-Processing frames algorithm.

We took inspiration from the image segmentation techniques when counting cells in a petri dish [74–76] and detecting the nucleus of cells [77]. We extended these above computer vision methods to include novel processing steps based on statistical methods, such as segmentation based on bivariate normal distributions so as to improve the precision of the determined agent locations, and improved prediction of trajectory paths using a Kalman filter [83]. The statistical methods introduced reduces the amount of noise in the resulting agent trajectories and reduces the difficulty in producing consistent results over long periods of time. The details of the new steps are described later in this chapter in Section 2.5.1 and Section 2.6.

We combined the above methods to accurately find a fixed number of agents in an image

and track their locations. From this we are able to examine their individual trajectories, each agent associated with their own (x, y)-coordinates unbroken over time. The datasets produced using our algorithm are the largest datasets of collective animal motion known to the author at the time of writing.

2.2 Video Collection And Equipment

In this section we describe our experimental setup of capturing behaviour on video. We observed the herding of large groups of sheep ($N > 40$) by a quadbike in flat enclosed fields. We achieved our filming with a DJI Phantom 3 drone equipped with a built in camera attached to the underside, Figure 2.1. By filming with a drone we are able to have the camera directly above the sheep. This removes the risk of sheep being obscured by other sheep due to the camera angle. Similar observational work has previously been performed where the research team fixed a camera to the top of a tall pole that took a photo every second. Due to the camera angle, the sheep regularly obscured each other resulting in all location extraction needing to be done by hand [56, 84]. The drone is equipped with onboard GPS, allowing it to hover in one location without any input from the user, providing stable footage.

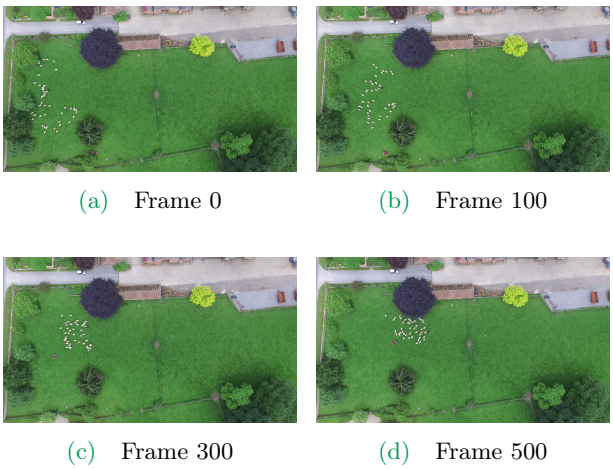
The drone can fly up to a hight of 120m but at this height the sheep are no larger than a couple of pixels on the resulting video, so we film as low as we can without causing the sheep undue stress due to the noise of the drone and the air displacement. Thus a balanced height is maintained at approximately 90m. The drone can use the on-board GPS to lower the 120m height restriction if it is flying in an area where drone flight is restricted, for example, under a flight path. This GPS function is also useful if the drone gets in trouble. Before a flight starts a location can be set as 'home' and if you were to lose connection with the drone or it were to run low on battery the drone would fly itself back to the 'home' location and safely land [85]. The drone only has a 25 minute battery life so all flights must be short, we aim to get 2-3 herding sessions with a single battery.

Figure 2.1: The filming equipment consisted of a DJI Phantom 3 with built in camera [85].

The camera used for filming was the original built in camera. By using the original camera we did not have to consider the additional weight of a different camera when considering the battery life of the drone, as carrying heavy loads can cause increased battery consumption. The camera up shoots in 2.7K (2704×1520 pixel image) and films with 24 frames per second (FPS). This frame rate is sufficient to accurately reconstruct the sheep's trajectories.

2.3 Description Of Sheep Videos

We have fourteen videos of length varying from 9 seconds to 25 seconds from which we want to extract the location of the sheep. The filming for all videos was carried out at a farm outside of the town of Northallerton, North Yorkshire (54.302237, -1.446258) in the north east of England in June 2018, Figure 2.3. On the first day of filming it was bright but with almost complete cloud cover and with almost no wind. This cloud cover ensured that each sheep could be seen clearly but there were very few shadows that should have hampered the locating of the sheep. The second day of filming it was just as bright and cloudy but on this day a substantial amount of wind which caused the drone to shift during filming. The data extracted from the footage on this day has more noise present. The farm has a number of large flat fields that we used for filming. The fields were not completely homogeneous due to a small number of trees near the edge, these can be seen in Figure 2.2. But as we are interested in the response of the sheep reacting to the quadbike



(a) Frame 0 (b) Frame 100

(c) Frame 300 (d) Frame 500

Figure 2.2: A subset of frames taken from Video 1. The sheep can be seen moving from left to right across the image and then up towards a large tree.

Figure 2.3: Map of the U.K. showing the location of the farm used for our experiments [86].

we trimmed each video to contain only the parts where each sheep is clearly visible.

The height we film at is fixed at 90m so that the size of the sheep in the images is approximately constant. In Videos 1-7, 10-14 there are 44 white sheep and a singular black sheep, giving $N = 45$, that are always in the view of the camera. In Videos 8 and 9 there is a different flock of sheep, this time containing 88 white sheep. In Video 1 the sheep start approximately stationary, Figure 2.2a, but most are facing toward the centre of the field. The farmer is initially behind a few trees where it is out of sight of the sheep and so it is the noise of the quadbike which makes the sheep move. The sheep move from the trees at the edge of the field towards the centre, Figure 2.2b. The sheep pause when they get used to the noise quadbike before they can see it and then start moving away from the bike once it gets in sight, Figure 2.2c. The farmer herds the sheep up towards the larger tree at the top of the field, Figure 2.2d.

2.4 Image Manipulation: Preprocess Stage

The locations of the sheep can be extracted from each frame using image manipulation and image segmentation tools, the algorithm is depicted in Flowchart 2.2. We initially split the video into the original frames. There are a number of standard ways to do this: programs such as VLC player allow you to save a video as frames; or you can convert a video into a set of still images using command line tool, such as ffmpeg; or you can load the video the majority of modern programming languages and from there loop through each frame. We chose the latter of these methods because we want to have a single work flow. We use Python 2.7 [87] with the numpy 1.13.1 [88] package and load the video into memory using the OpenCV package [89]. A subset of these frames from Video 1 can be seen in Figure 2.2.

The extraction process starts by passing each of the frames from the video through a number of image manipulation steps. These are essential to reduce the noise in the image, leaving a much simpler image to work with.

2.4.1 Colour Image Format

Each of our extracted frames is a colour image, with three colour channels: red, green and blue. Each pixel (i, j) in the image has three numeric values, $\mathcal{F}_{i,j} = (R_{i,j}, G_{i,j}, B_{i,j})$, so a colour image can be thought of as a 3D array. In an RGB image the range of values each colour can take is conventionally limited to integers that lie between 0 and 255 inclusive giving 2^8 possible values per colour channel. This convention arises so that a single channel can be easily represented by a single byte in memory. However, when discussing the values

Colour	RGB Value	Greyscale value, 0–1 Weighted sum	Gamma Correction, $\gamma = 5$
White	(255, 255, 255)	1	1
Black	(0, 0, 0)	0	0
Red	(255, 0, 0)	29.9	0.33
Green	(0, 255, 0)	0.59	0.33
Blue	(0, 0, 255)	0.11	0.33
Orange	(255, 120, 0)	0.58	0.34
Purple	(120, 0, 255)	0.25	0.34
Yellow	(255, 255, 0)	0.89	0.66

Table 2.1: A selection of different colours with the equivalent RGB values and greyscale values using a subset of methods.

of RGB in this document they will be rescaled to lie in the range 0–1 to make comparisons easier. Table 2.1 shows a number of observed colours with their RGB values.

2.4.2 Converting To Greyscale

To make manipulating the image easier the first step is to convert the image to greyscale. This reduces the image to a 2D scalar array, where value $\mathcal{I}_{i,j}$ is the value of the pixel in the greyscale image, $\mathcal{I}$, at position (i, j). There are a number of conversion methods available and in the following section we will investigate some of the more widely-used of these methods. The value of a pixel in a greyscale image varies from 0 to 1, which is equivalent varying from black to white.

(a) Original colour image. (b) Weighted average greyscale image.

Figure 2.4: Converting an image of Tynemouth priory from colour to greyscale using a weighted average of the three colour channels; red, green and blue [90].

Converting To Greyscale Using A Weighted Average

The default method is to take a weighted average of the three colour channels. Given a pixel in the RGB image, $\mathcal{F}_{i,j} = (R_{i,j}, G_{i,j}, B_{i,j})$, the corresponding pixel in the greyscale image is calculated using

$$\mathcal{I}_{i,j} = 0.299 \times R_{i,j} + 0.587 \times G_{i,j} + 0.114 \times B_{i,j}. \tag{2.1}$$

The weights shown here are the ones used by OpenCV [89] and Matlab [91]. Given that each colour channel is given a different weight when calculating the average two objects that look equally as bright (e.g. the grass and the sky) can look quite different after conversion. This is because this method of conversion does not retain luminance, meaning that bright areas can get dulled to a pale grey.

Examples of a number of RGB colour values converted to grey using this method can be seen in Table 2.1. An example of converting a colour image to greyscale using this method can be seen in Figure 2.4. Even though this method does not retain luminance, it does retain many features of the image. In Figure 2.4b one can still see definition in the stonework of the priory and some of the texture of the grass.

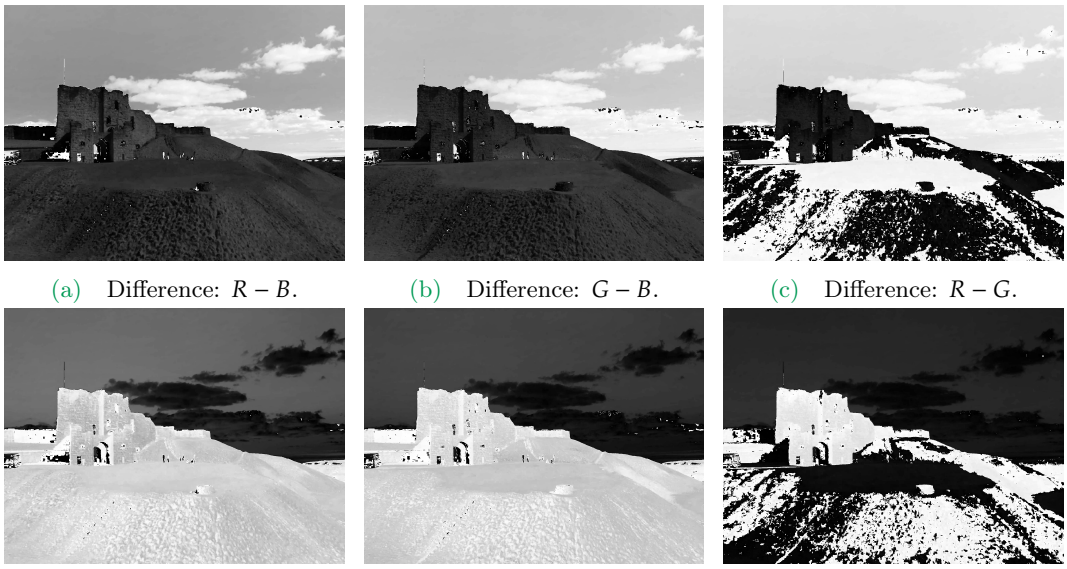
Converting To Greyscale By Using A Single Colour Channel

Another common method is to use one of the three colour channels. That is that given that the colour is is $\mathcal{F}_{i,j} = (R_{i,j}, G_{i,j}, B_{i,j})$ the greyscale image becomes

$$\mathcal{I}_{i,j} = \begin{cases} R_{i,j} & \text{if red channel is chosen,} \\ G_{i,j} & \text{if green channel is chosen,} \\ B_{i,j} & \text{if red channel is chosen.} \end{cases} \tag{2.2}$$

(a) Red colour channel. (b) Green colour channel. (c) Blue colour channel.

Figure 2.5: Converting an image of Tynemouth priory to greyscale by using just one of the three colour channels; red, green or blue [90].



(a) Difference: $R - B$. (b) Difference: $G - B$. (c) Difference: $R - G$.

(d) Difference: $B - R$. (e) Difference: $B - G$. (f) Difference: $G - R$.

Figure 2.6: The six different combinations of calculating the difference between two of the three colour channels in an R, G, B image.

Depending on which channel you choose you can get drastically different results. Figure 2.5 shows the differences of extracting each of the three colour channels of a single image of Tynemouth priory [90]. The sky significantly differs in each of the three images. In Figure 2.5a the sky is darker than in Figure 2.5b, but the main difference comes from looking at the blue channel, Figure 2.5c. In the image of the blue colour channel, the sky appears almost white. This is because the value of the pixels for the sky in RGB space will be close to $(0, 0, 1)$, so when we look solely at the blue colour channel the sky have a value close to 1, which is the colour white. Therefore it becomes harder to distinguish between the sky and the clouds.

Converting To Greyscale By Calculating The Difference Between Colour Channels

An alternative to selecting just one of the colour channels is to use the difference between two channels. That is the greyscale pixel takes the value

$$\mathcal{I}_{i,j} = \mathcal{F}_{i,j,k} - \mathcal{F}_{i,j,l} \tag{2.3}$$

where $\mathcal{F}_{i,j,k} \in \{R_{i,j}, G_{i,j}, B_{i,j}\}$ but $k \neq l$. With a three channel image there are three different combinations you can calculate, and their negatives. The values this method returns are not guaranteed to be between 0–1 so they have to be rescaled to fit into this range. Figure 2.6 shows the effects of calculating the six different combinations of

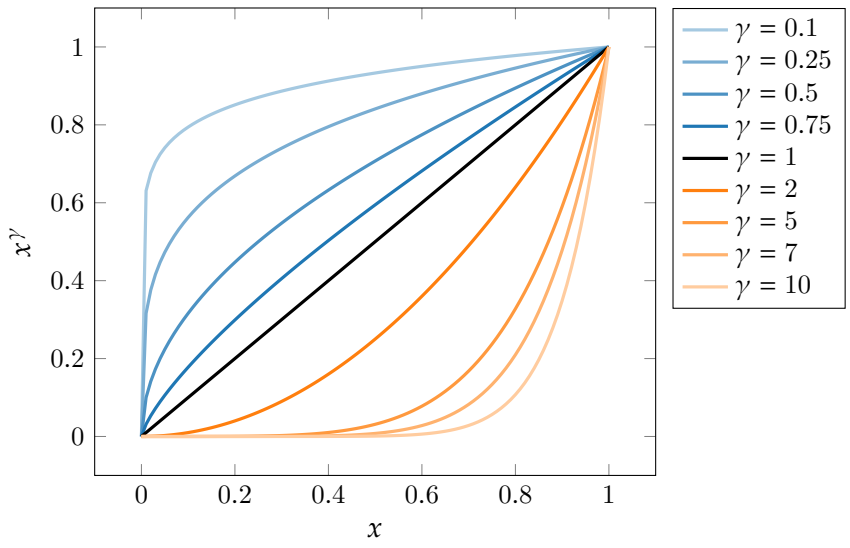


Figure 2.7: The curves of x^γ where γ ranges from 0.05 to 100.

differences that can be made from an RGB photo of Tynemouth priory [90]. Figures 2.6d-f are the negative images of Figures 2.6a-c respectively.

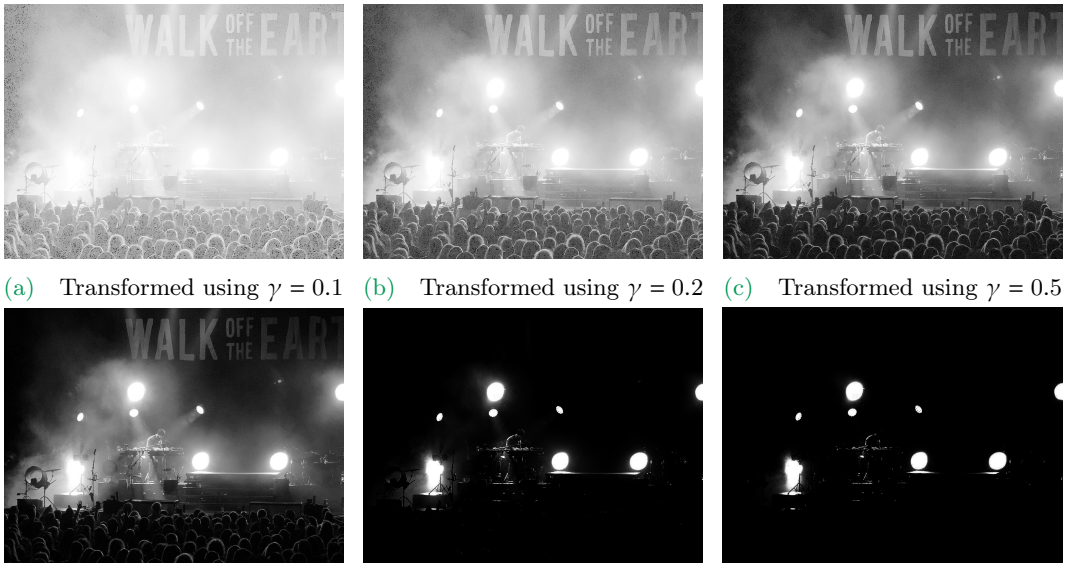
Using Gamma Correction When Converting To Greyscale

Following from the method where a weighted average is used, we can take a transformation of each colour channel then take the average of the modified channel. The transformation method we use here is known as gamma correction [92]. Gamma correction can make shadows darker or lighter depending on the size of parameter, γ. Gamma correction can take many forms [92], here we use one of the simplest cases, a power-law expression:

$$V_{\text{out}} = AV_{\text{in}}^{\gamma}. \tag{2.4}$$

Setting $A = 1$ Figure 2.7 shows the effect of gamma correction on a linear input between 0-1. When gamma $= 1$ no change to the input occurs. When $\gamma < 1$ the input is distorted so as to emphasise values. Similarly when $\gamma > 1$ the input is distorted so as to diminish the values. This distortion follows a power law. The overall effect is to increase or decrease the average value whilst maintaining a continuous smooth output between 0 and 1.

We can see how the image changes as γ varied from 0.1 to 10 on the greyscale image in Figure 2.8. When $\gamma < 1$ the image gets much brighter as the pixel values get lifted to 1, Figs 2.8a–2.8c. If γ gets too small then features start to disappear as everything goes to white. When we have $\gamma = 1$ we return to our original image, Figure 2.8d. When $\gamma > 1$ the shadows get emphasised to the point that features can start to disappear. In



(a) Transformed using $\gamma = 0.1$ (b) Transformed using $\gamma = 0.2$ (c) Transformed using $\gamma = 0.5$

(d) Original image, $\gamma = 1$ (e) Transformed using $\gamma = 5$ (f) Transformed using $\gamma = 10$

Figure 2.8: The effects of gamma correction on a concert photo with lots of shadows. When $\gamma < 1$ the shadows become lighter, the crowd becomes clearer but the keyboard player becomes washed out by the lights. On the other hand when $\gamma > 1$ the shadows become much darker to the point when $\gamma > 5$ the crowd can no longer be seen and when $\gamma \approx 10$ the keyboard player also starts to vanish [93].

Figure 2.8e the crowd is almost completely obscured. This is due to the pixels value now all being crushed to 0 instead of lifted to 1. When the γ value gets high enough most of the features practically vanish and we are left with an image showing where the spotlights were. Gamma correction is used in encoding and decoding luminance or tristimulus values in video or still image systems [92]. The standard value for gamma when encoding is 0.45, this compresses the image. For decoding the image the standard value is 2.2, this will expand the image.

Converting To Greyscale By Calculating The Product Of The Three Colour Channels

Another more unconventional method for converting the colour image to greyscale is to multiply the colour channels together, that is, given the pixel has RGB value of $\mathcal{F}_{i,j} = (R_{i,j}, G_{i,j}, B_{i,j})$, the same pixel in the greyscale image will obtain the value

$$\mathcal{I}_{i,j} = (R_{i,j} \times G_{i,j} \times B_{i,j}). \tag{2.5}$$

This works similarly to the gamma correction in that the darker pixels get crushed towards black whereas the brighter pixels get lifted in the direction of white. Figure 2.9 shows the

Figure 2.9: The resulting image of calculating the product of the red, green and blue colour channels.

effect of converting to greyscale using the product of the three colour channels on an image of Tynemouth priory [90]. While most of the image has been darkened, the bright areas (such as the clouds) stand out clearly in white.

2.4.3 Converting Drone Images To Greyscale

In this section we will go through all the conversion methods mentioned in Section 2.4.2 and discuss if the method is suitable for use when looking at the frames from the drone footage. Each of the methods will use the same colour image taken from Video 1 for conversion.

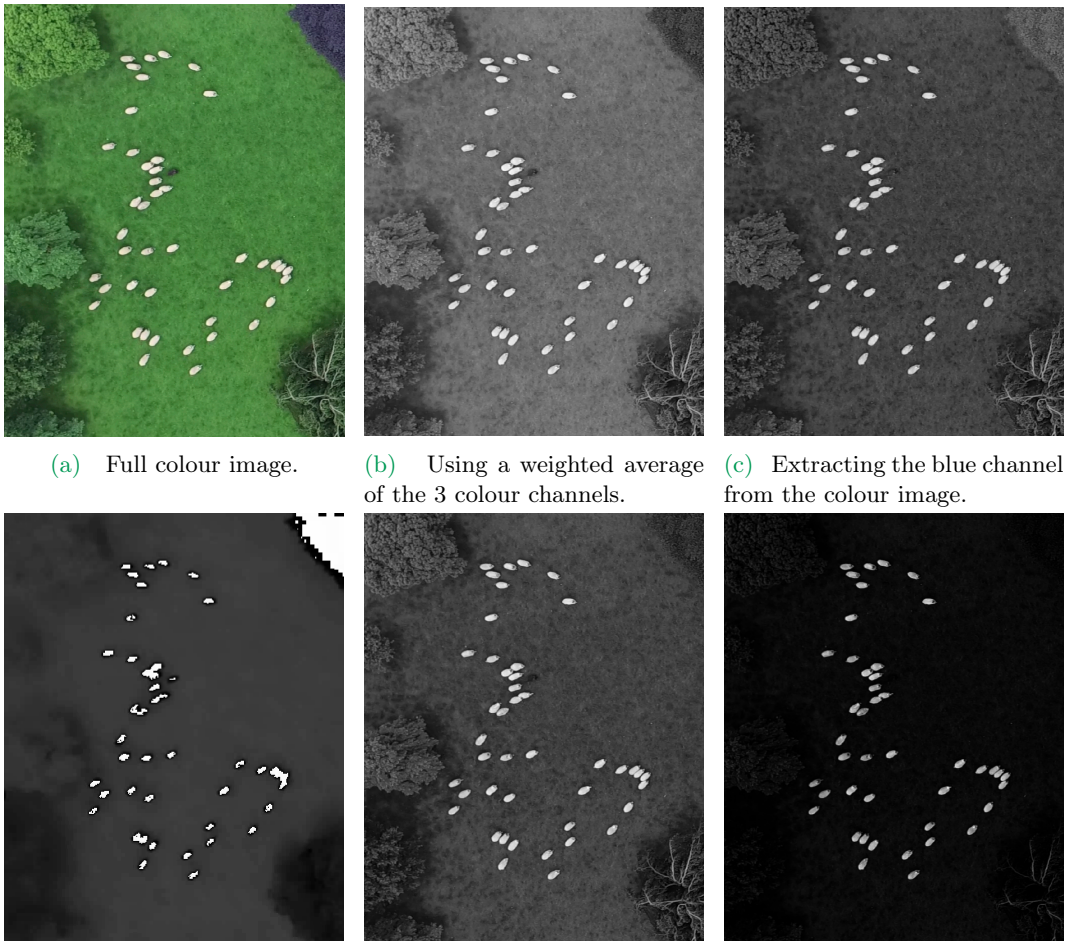
The first method is a weighted average of the colour channels to convert to greyscale. When converting the drone image the loss of luminance reduces the contrast between the sheep and the grass, Figure 2.10b. This contrast is vital in determining what is sheep and what is grass when looking solely at the values of the pixels. Despite the loss of luminance this method is very good at preserving the features of the image; a human could still locate each sheep in the image.

When using just one of the colour channels we have to decide which channel to choose. It is best to choose the colour channel which retains the most information. In this case we would choose the blue channel over the other two because the external objects, such as the fence bounding the field, becomes much less pronounced while preserving a lot of the features of the sheep without too much distortion, Figure 2.10c. This method retains the original contrast between the sheep and the grass.

Converting to greyscale by calculating the difference between colour channels removes most of the features, including some of the sheep, so this method cannot be used for detecting the sheep in the images. Figure 2.10d shows the difference between the green and red

channels. However, when trying to extract the location of the quadbike this method gives the best results, Section 2.9.2. This is because the quadbike is red and the surrounding grass is green, so when the difference between green and red is rescaled to lie between 0 and 1 an image of the silhouette of the quadbike is left behind.

When using gamma correction with the images from the drone we set the parameter $A = 1/255$ in order for the pixel values to be rescaled to the range $(0 - 1)$. We then set the gamma correction parameter be $\gamma \approx 1.3$. As shown in Figure 2.10e a gamma value of this size subdues most of the dark elements, the grass, to black leaving only bright sheep behind. This is useful as it returns a large contrast between the sheep and grass and still preserves details such as the shape and size of each sheep.



(a) Full colour image.

(b) Using a weighted average of the 3 colour channels.

(c) Extracting the blue channel from the colour image.

(d) Calculating the difference between the green and the red colour channels.

(e) The average of the colour channels transformed using gamma correction, $\gamma = 1.35$.

(f) The product of the three colour channels rescaled to lie between $0 - 1$.

Figure 2.10: A frame from Video 1 cropped to show just the area containing the sheep converted to greyscale using Equations (2.1)–(2.5).

Converting to greyscale using the product of the three colour channels has a similar effect to using gamma correction. So it is not a surprise that we obtain similar outcomes to converting the drone image to greyscale using these methods, Figure 2.10f. The potential issue with this method is that some of the sheep decrease in size making them harder to locate in the image.

Selecting A Method

To quantitatively determine which of these methods is best to use, we look at the kernel density of each of the resulting images. Figure 2.11 shows that the resulting distributions are bimodal. Each peak represents the two main objects in our images; the grass and the sheep. In order for us to remove the background we want to be able to find a suitable cut off point where all pixels with values less than that value are grass. The further apart these peaks are the easier it is to find this value and therefore, the easier it will be to separate the sheep from the grass. The method that will be most useful for detecting the sheep will be a compromise between peak separation and features retained in the image.

The method with the most separated peaks is the difference between green and red channels, however, as mentioned earlier this method is not feasible for the detection of sheep, by human or computer vision, due to many features being lost. This compromise cannot be made.

The next best methods due to peak separation in the kernel density plot is gamma cor-

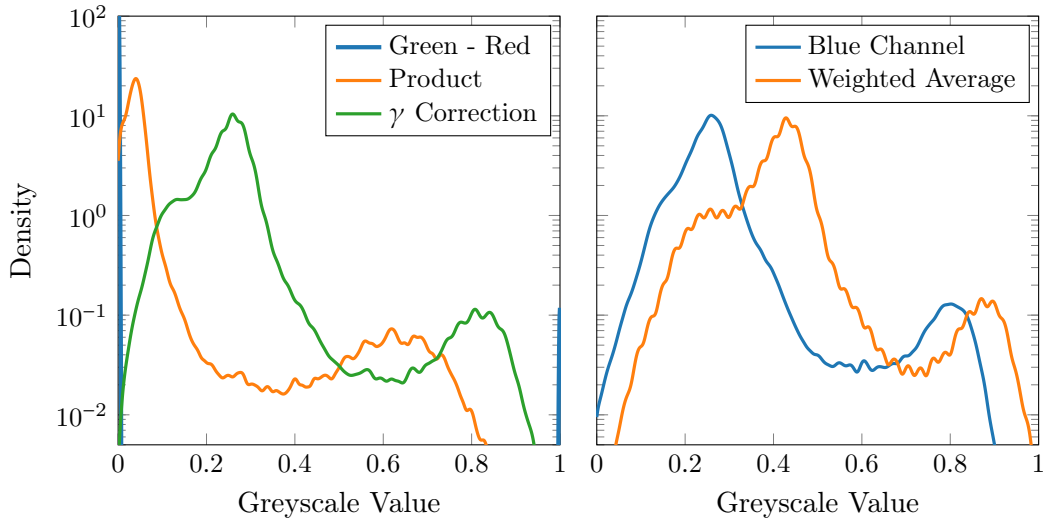


Figure 2.11: Kernel density distributions of the 5 methods use to make the greyscale images in Figure 2.10 plotting on a log axis. The each distribution appears to be bimodal, representing the grass and the sheep.

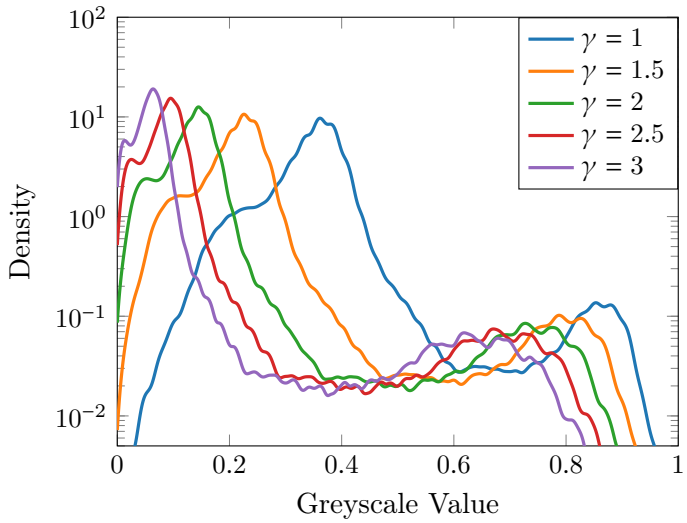


Figure 2.12: Kernel density distributions of the drone image converted to greyscale using the gamma correction method for several values of γ. The distribution shifts in the negative x direction as γ increases. The distance between the two peaks does not change when $\gamma > 1$.

rection or using the product of the three colour channels. They both have a large narrow peak at the small greyscale values and a smaller peak at the higher greyscale values. The area between the two peaks represents areas where it is unclear whether the pixel belong to a sheep or part of the grass. In the image this could be areas where grass has reflected more light than expected or shadows cover part of the sheep. Since the value of γ effects brightness of the sheep it is also worth looking at the kernel density plot of the greyscale images for multiple gamma values. Figure 2.12 shows γ effects the location of the two peaks but the distance between them remains approximately unchanged.

The kernel density distribution for the blue channel and for the greyscale image created using the weighted average method also have two very well defined peaks, but since the distance between them is smaller then the gamma correction or product methods, these methods might not perform as well.

Due to the slightly increased distance between peaks we conclude that the method of taking the average of the colour channels that have been transformed using gamma correction is the best method to use in the case our drone images.

2.4.4 Thresholding An Image

Now that the image is greyscale we can pass it through through two thresholds, as we want to classify each pixel as belonging to the background grass or the sheep. The first threshold replaces all pixels with a value less than t_l with the value v_l and the second replaces all pixels with a value greater than t_u with v_u. Therefore for the value of the

(a) The original greyscale image of a grey and white cat

(b) A lower threshold: replaces all pixel values lower than 0.35 with 0

(c) An upper threshold: replaces all pixel values higher than 0.65 with 1

(d) Both lower and upper thresholds at 0.35 and 0.65

Figure 2.13: The effects of doing two thresholds on a greyscale image of a cat [94]; one to replace the smaller values with 0 and one to replace the higher values with 1.

pixel at coordinates (i, j), in the thresholded image $\bar{\mathcal{I}}_{i,j}$, gets calculated using

$$\bar{\mathcal{I}}_{i,j} = \begin{cases} v_l & \mathcal{I}_{i,j} < t_l, \\ \mathcal{I}_{i,j} & t_l < \mathcal{I}_{i,j} < t_u, \\ v_u & \mathcal{I}_{i,j} > t_u. \end{cases} \tag{2.6}$$

where $\mathcal{I}_{i,j}$ is the value of the pixel before the thresholding, i.e. the value of the pixel in the greyscale image at (i, j). Figure 2.13 shows the effects of doing these steps on a picture of a grey and white cat [94] where the lower value is $v_l = 0$, black, and the upper value is $v_u = 1$, white.

The values of the threshold are obtained from the kernel density plot of values in the greyscale image. Figure 2.14 shows the histogram of the values in the image created by the gamma correction methods mentioned in Section 2.4.2. The lower threshold should be set to be at the base of the first peak. Most of the points with a value less than this will be grass and so we can set them to 0 — black. The second threshold should be set at the base of the second peak where the density starts to increase, since the second peak is the pixels covering the sheep. Pixel with values greater than this are set to 1 — white. This idea is similar to the logic behind the Otsu method [95].

To determine the values of these thresholds we fit a Gaussian mixture model; hence we assume our data consists of a weighted sum of Gaussian distributions. Since our data has two well defined peaks that are joined together with an area of uncertainty we try fitting a Gaussian mixture model with 3 distributions, Figure 2.14. We set our first threshold at a high percentile value of the lowest distribution (somewhere between 95% and 99.9%)

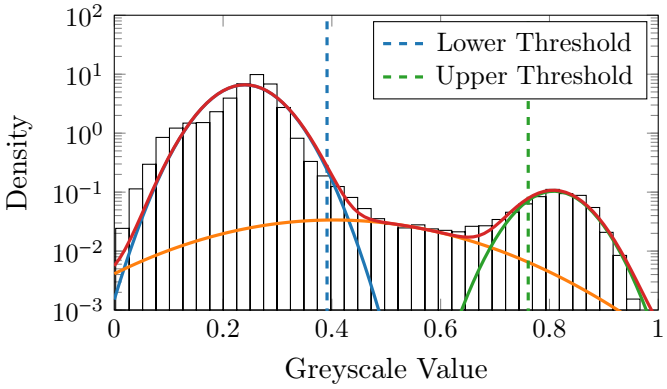


Figure 2.14: Histogram of the greyscale value in Figure 2.10e. Overlaid are the resulting 3 normal distributions after fitting a 3 component Gaussian mixture model, ((-), (-), (-)) and the mixture model itself (-). The vertical lines represent the 95% percentile for the lowest distribution (--) and the 5% percentile for the highest distribution (--).

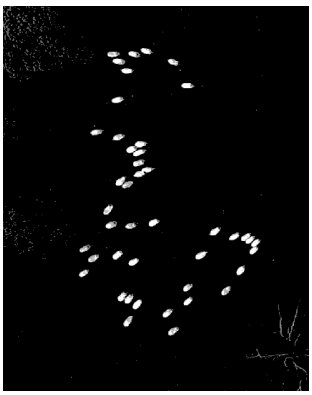


Figure 2.15: The effect of an upper and lower threshold on the greyscale image produced in Section 2.4.3 leaves only a small proportion of grey pixels.

and the second threshold, t_u, at a lower percentile of the third distribution (somewhere between 1% and 40%). We call the percentiles chosen p_{t_l} and p_{t_u} for the upper and lower thresholds respectively. By setting the values of t_l and t_u to be at a fixed percentile, p_{t_l} and p_{t_u} respectively, the values of t_l and t_u are dynamic, as the distributions of the greyscale values will change over the course of the video but the percentiles remain constant. The algorithm automatically changes the values of t_l and t_u through time as the image changes, as it is able to react to how bright the image is.

2.4.5 Determining Parameter Values Using A Simulation Study: γ, p_{t_l}, p_{t_u}

Since the location of the peaks are dependent on γ when converting to greyscale we completed a simulation study to determine which were the optimal values for γ, p_{t_l} and p_{t_u}. The values mentioned in this section refers the parameter values for Video 1. We ran our tracking algorithm for 20 frames and observed whether the correct number of sheep were identified. For a single value of γ we investigated 6 values for p_{t_l} and 5 for p_{t_u}. We test more values for p_{t_l} since the distribution for the smaller values has a greater variance.

For each $\gamma = 1, 1.025, \ldots, 2.475, 2.5$ the proportion of parameter combinations that identify the correct number of sheep can be calculated. The plot in Figure 2.16 shows that all values of $\gamma < 1.6$ have a proportion of 50% or greater. The value selected for gamma correction should be in this range ensuring that the locations outputted from the algorithm are consistently placed in the centre of each sheep. For Video 1 we chose the value of $\gamma = 1.5$ since as the sheep become more tightly packed, a large gamma value ensured

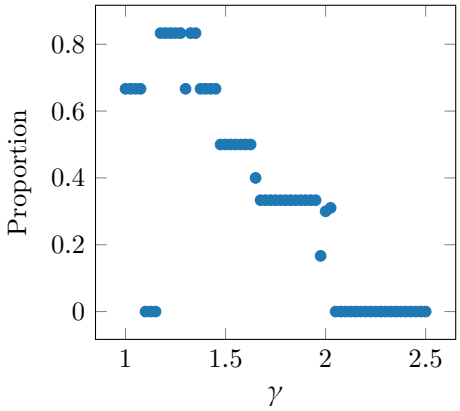


Figure 2.16: The proportion of p_{t_l} and p_{t_u} values that correctly identify 44 sheep in the first 20 frames of Video 1 for each γ value, $p_{t_l} \in (95\%, 99\%, 99.25\%, 99.5\%, 99.75\%, 99.9\%)$ and $p_{t_u} \in (1\%, 10\%, 20\%, 30\%, 40\%)$.

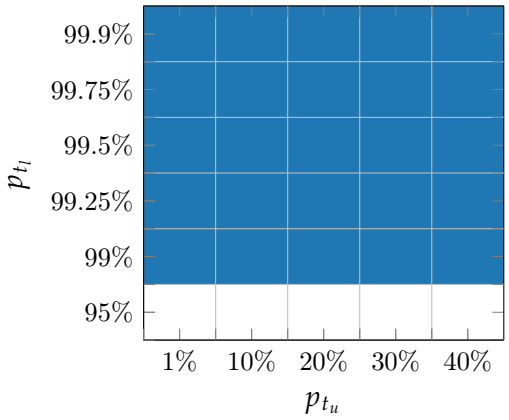


Figure 2.17: Percentile combinations that correctly identify the 44 sheep are shown in blue otherwise are shown as white for $\gamma = 1.25$. Here 83% of the combinations tested obtained the correct number.

that the sheep could still be correctly identified.

Once the value of γ is fixed we can then establish the value of p_{t_u} and p_{t_l} for the upper and lower thresholds respectively. For a single γ value we can produce a plot of which percentile values identified the correct number of sheep and which did not, Figure 2.17. For the percentile for the lower threshold the range of values that identify the correct number of sheep is from 99% to 99.9% so we set the parameter p_{t_l} to be the mid point $p_{t_l} = 99.5\%$. Similarly for the upper threshold, p_{t_u} is set to be the midpoint of the values that identity the correct number of sheep, i.e. $p_{t_u} = 20\%$.

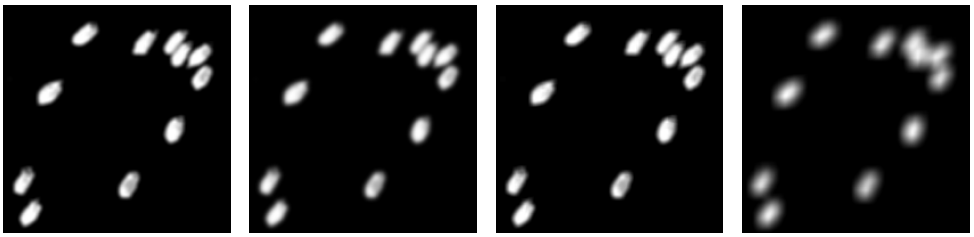
2.4.6 Blurring An Image Using A Gaussian Blur

Once an image has been simplified using a threshold, the image is blurred using a Gaussian blur. A Gaussian blur is the convolution of a Gaussian function with standard deviation σ_G centred on each individual pixel of an image. The new value of the pixel at (i, j), $\tilde{\mathcal{I}}_{G\,i,j}$, can be calculated using

$$\tilde{\mathcal{I}}_{G\,i,j} = \sum_{i*=i-L}^{i+L} \sum_{j^*=j-L}^{j+L} \bar{\mathcal{I}}_{i^*,j^*} \frac{1}{2\pi\sigma_G^2} \exp^{-(i^{*2}+j^{*2})/2\sigma_G^2}. \tag{2.7}$$

where $L = (\ell_G - 1)/2$.

A Gaussian blur reduces high frequency white noise from an image but while doing this it also obscures some of the finer detail. The Gaussian blur has two parameters that need to be set; the width of the kernel, ℓ_G, and the standard deviation of the kernel, σ_G. There



(a) $\ell_G = 5, \sigma_G = 2$ (b) $\ell_G = 13, \sigma_G = 2$ (c) $\ell_G = 5, \sigma_G = 20$ (d) $\ell_G = 13, \sigma_G = 20$

Figure 2.18: Effect of the Gaussian blur when varying the kernel size, ℓ_G, and standard deviation, σ_G.

are restrictions on what values the width of the kernel can take. 1) It must be a positive integer. 2) It must be odd. The first of these restrictions ensures that that the resulting kernel can be represented as an $\ell_G \times \ell_G$ matrix of values. The latter ensures that the kernel has a well defined centre, making it easy to place the centre of the kernel over each pixel in the image. When $\ell_G = 1$ the Gaussian blur has no effect as this is equivalent to a weighted average of just one value.

Given that most of the noise gets removed by the threshold only a small width is required, $\ell_G \in \{3, 5, 7\}$. For Video 1 the value $\sigma_G = 5$ is used. Since the kernel size is small and the image is close to binary the effect of the size of the standard deviation is minimal; one would have to look at the individual pixel values to see a difference, however, when looking at a large kernel size the effect of the standard deviation is much clearer. The larger the value of σ_G the more obscured each sheep gets, Figure 2.18.

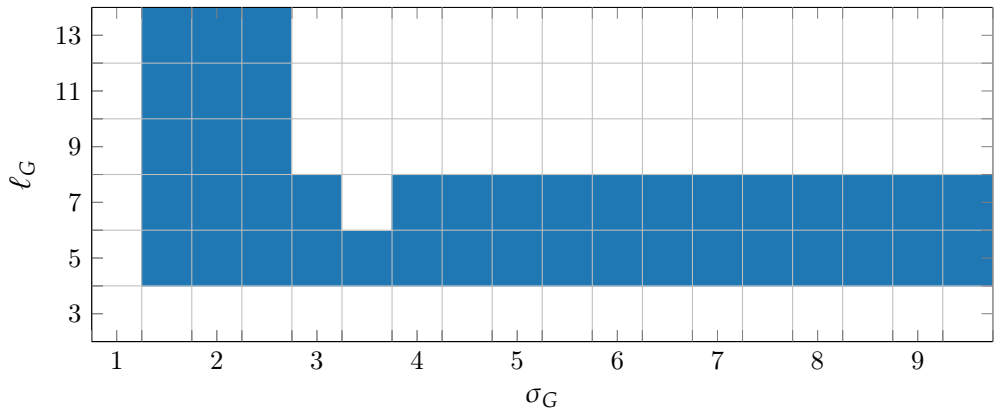


Figure 2.19: Combinations of the Gaussian blur parameters, ℓ_G and σ_G, that correctly identify the correct number of sheep are shown in blue otherwise are shown as white.

Determining Parameter Values Using A Simulation Study: ℓ_G, σ_G

Using a similar method to when determining the parameter values for the threshold we use a simulation study to determine the value of the parameters used in the Gaussian blur. We ran our tracking algorithm for 20 frames observing whether the correct number of sheep were identified while varying the values of ℓ_G and σ_G. Since we are now only changing two parameters we can display the full parameter sweep in one plot, Figure 2.19. The figure shows us that when ℓ_G is small i.e. $\ell_G < 9$, the value of σ_G does not alter the number of sheep detected. This is because the size of the filter limits the impact of the strength of the blur. Looking back at Figure 2.18 we can see that effect of increasing the filter size from $\ell_G = 5$ to $\ell_G = 13$ whilst keeping the strength of the blur constant at $\sigma_g = 2$, has a greater impact than increasing the strength of the filter from $\sigma_G = 2$ to $\sigma_G = 20$ and keeping the size of the filter restricted to $\ell_G = 5$. However, when the filter size is larger $\ell_G = 13$ you can see that increasing the blur strength from $\sigma_G = 2$ to $\sigma_G = 20$ the effect is much bigger.

This plot shows that the value of ℓ_G with the highest proportion of simulations where the correct number of sheep identified is $\ell_G = 5$. When using this value for ℓ_G we can then select the value for σ_G.

There are 3 values for σ_G that have highest proportion of simulations where the correct number of sheep were identified, $\sigma_G = 1.5, 2, 2.5$. Therefore we select the mean to help reduce the chance of under blurring the image or over blurring the image as time goes on as both of these could result in the tracking algorithm failing due to the wrong number of sheep being detected.

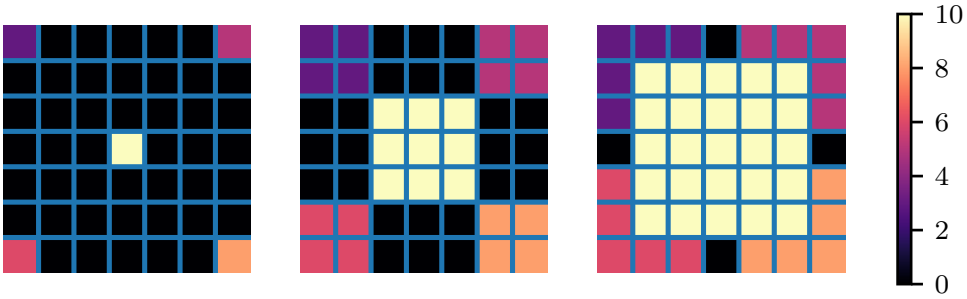
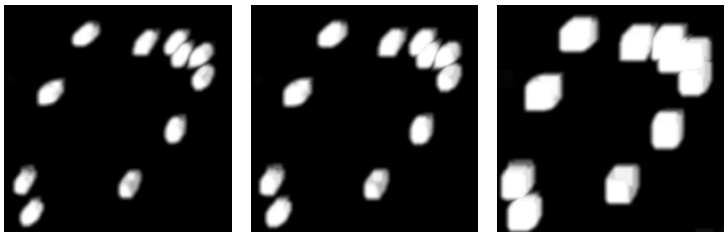


Figure 2.20: Schematic of the maximum filter a) original image where black is 0 and as the colour increases brightness the value increases b) maximum filter with $\ell_M = 3$ c) maximum filter with $\ell_M = 5$



(a) $\ell_M = 3$ (b) $\ell_M = 5$ (c) $\ell_M = 13$

Figure 2.21: Effect of the maximum filter when varying the kernel size ℓ_M. As the kernel size increases the features retained by the image decreases, to the point where the original subject of the image can be lost.

2.4.7 Blurring An Image Using A Maximum Filter

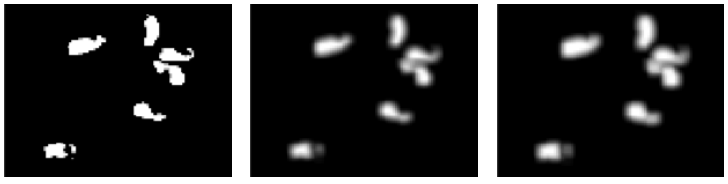
After the Gaussian filter has been applied, it is then manipulated further using a maximum filter. This filter is important since it reduces the chance of a silhouette of a sheep fracturing into many parts. This filter replaces each pixel with the maximum value of its neighbouring pixels within a kernel size, ℓ_M. A demonstration of the maximum filter can be seen in Figure 2.20. Similar to the Gaussian filter the maximum filter has a kernel size, ℓ_M. The larger the kernel size the more distorted the image becomes.

The new value of the pixel at (i, j), $\tilde{\mathcal{I}}_{M\,i,j}$, can be calculated using

$$\tilde{\mathcal{I}}_{M\,i,j} = \max(\tilde{\mathcal{I}}_{G\,i^*,j^*}). \tag{2.8}$$

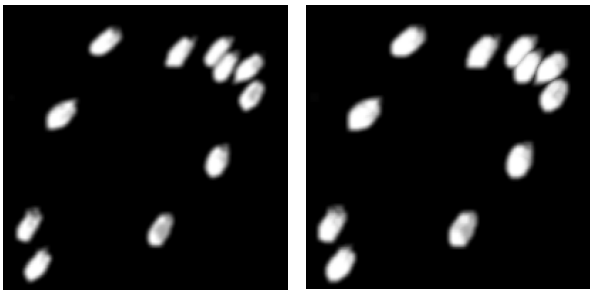
where $i^* = i - (\ell_M - 1)/2, i - (\ell_M - 1)/2 + 1, \cdots, i + (\ell_M - 1)/2 - 1, i + (\ell_M - 1)/2$ and $j^* = j - (\ell_M - 1)/2, j - (\ell_M - 1)/2 + 1, \cdots, j + (\ell_M - 1)/2 - 1, j + (\ell_M - 1)/2$.

When looking at the image taken after applying the Gaussian blur, and performing the maximum filter with varying sizes of ℓ_M (Figure 2.21) we can see that in order to preserve a realistic shape and size of the sheep in the image we need ℓ_M to be small, i.e. $\ell_M < 13$. For Video 1 we set $\ell_M = 3$ so that as many features as possible are retained in each of the images.



(a) Fractured sheep silhouette (b) Gaussian blur: $\ell_G = 5, \sigma_G = 2$ (c) Maximum filter: $\ell_M = 3$

Figure 2.22: The effect of the maximum filter on a sheep silhouette which fractures into two distinct parts during the first threshold.



(a) Gaussian blur: $\ell_G = 5, \sigma_G = 2$ (b) Maximum filter: $\ell_M = 3$

Figure 2.23: The differences between the result of the Gaussian blur and the following maximum filter may not visually be that different. But without the maximum filter the process of extracting the location of the sheep becomes much trickier.

The combination of these two filters helps reduce the chance of the sheep silhouettes fracturing into multiple parts. This can happen when shadows obscure part of the sheep and when the image is passed through the thresholding procedure these darker areas are incorrectly identified as background. An example of a sheep silhouette fracturing into two part, then being but back together using the Gauss and Maximum filter can be seen in Figure 2.22. The differences between the image after the Gaussian filter and after the maximum filter may be small and hard to see visually (Figure 2.23) but we find that the ability to detect the sheep using computer vision without the maximum filter is much tricker.

Determining Parameter Values Using A Simulation Study: ℓ_M

Since we are now only looking to determine one parameter value we can look at the number of sheep identified for each value of ℓ_M in the range $1, 3, 5, 7, 9$, Figure 2.24. As discussed in the previous section, if ℓ_M get too large then the image gets highly distorted and the sheep cannot be identified. Figure 2.24 shows that when $\ell_M > 3$ the algorithm fails to detect some of the sheep.

We therefore have the choice between two values for the kernel size of the maximum filter, $\ell_M = 1, 3$. However, a value of $\ell_M = 1$ is equivalent to the maximum filter being removed from the process. So therefore since we do not want to risk the silhouettes of the sheep fracturing into multiple parts we chose $\ell_M = 3$.

Both the image after converting to greyscale and the image after the maximum filter has been implemented get passed on to the extraction stage of the algorithm. The result from the maximum filter allows us to deduce the approximate area of the image the sheep are in, and the image from the greyscale image retains a lot of this original information so therefore is useful for extracting precise locations of the sheep in the image once we have

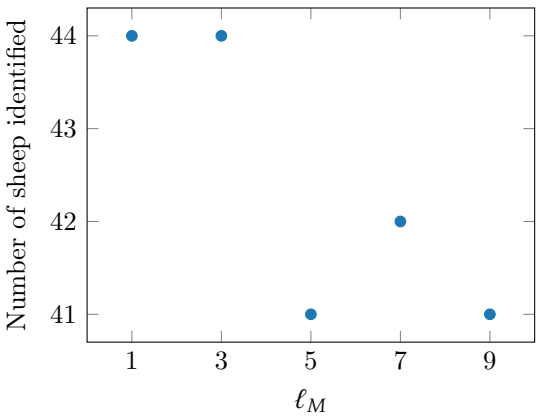


Figure 2.24: The number of sheep identified for a small number of ℓ_M tested. When $\ell_M = 1$ the maximum filter is effectively not used as you would be selecting the maximum of one value.

the approximate location from results using the maximum filter.

This is the end of the pre-processing stage of the algorithm to detect and extract the locations of sheep in drone footage.

2.5 Image Manipulation Using Prior Information

In the previous section we focused on processing techniques on single images in isolation. The locations of the sheep in each frame are highly correlated through time, so we can use the information we have from previous frames to inform our processing of the image, and subsequent location extraction.

2.5.1 Predicting Locations Using Kalman Filter

From Flowchart 2.1 the next step is to predict the locations of the sheep and then if necessary use our predictions to filter the image further. Since there is some noise in the previous detected locations (i.e. the location is not always in the centre of the sheep) a Kalman filter [83] is used to predict where the sheep will be next. The Kalman filter is based on a linear time-invariant dynamical system of the form:

$$\mathbf{z}^*_{t+1} = A\mathbf{z}^*_t + \mathbf{w}_t, \tag{2.9}$$

$$\mathbf{z}_{t+1} = H\mathbf{z}^*_t + \mathbf{v}_t, \tag{2.10}$$

where $\mathbf{z}^*_t$ is the underlying state and $\mathbf{z}_t$ is the measurement output at time t. The matrix A is known as the state transition matrix and it controls the typical expected behaviour of the state variables. The other matrix in the system is H, the observational matrix, and

it maps the true state space to the observed space. The final two terms, $\mathbf{w}_t$ and $\mathbf{v}_t$, are noise terms in both the process and the measurement and are given the bivariate normal distributions $N(0, Q)$ and $N(0, R)$ respectively, where Q is the dynamical covariance matrix and R is the measurement covariance matrix. The initial state and the noise vectors are assumed to be mutually independent.

It is assumed that the underlying state has four components, the x, y component of its location and the x, y components of its velocity. We also assume that it is not possible to measure the underlying velocities so the measurement has the form

$$\mathbf{z}_t = (z_{x,t}, z_{y,t})^T.$$

To formulate the transition matrix, A, we write approximate equations to describe the behaviour of the system. We relate the location of a sheep at time $t+1$ to the location and velocity of a sheep at previous time t. We make the simplification to assume that a sheep moves at constant velocity. As such the acceleration and higher order terms are 0 and so we can use an Euler time step, Section A.2.1, to approximate the position of a sheep at time $t+1$.

Therefore our equations are

$$\begin{aligned}
z^*_{x,t+1} &= z^*_{x,t} + \dot{z}^*_{x,t}, \\
z^*_{y,t+1} &= z^*_{y,t} + \dot{z}^*_{y,t}, \\
\dot{z}^*_{x,t+1} &= \dot{z}^*_{x,t}, \\
\dot{z}^*_{y,t+1} &= \dot{z}^*_{y,t}.
\end{aligned}$$

Hence A takes the form

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Similarly looking at the equations for the observations

$$\begin{aligned}
z_{x,t} &= z^*_{x,t} \\
z_{y,t} &= z^*_{y,t}
\end{aligned}$$

it is possible to extract the form of H,

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The covariance matrix for the dynamical noise here is:

$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{q_1}^2 & 0 \\ 0 & 0 & 0 & \sigma_{q_2}^2 \end{pmatrix},$$

i.e. we assume that the noise only affects the velocity of the sheep rather than the position. This noise in the velocity is what causes further small deviations in the position of the sheep in addition to the dynamical process.

The covariance matrix describing the distribution of measurement noises can be written as

$$R = \begin{pmatrix} \sigma_{r_1}^2 & \sigma_{r_1,r_2} \\ \sigma_{r_1,r_2} & \sigma_{r_2}^2 \end{pmatrix},$$

where σ_{q_i} and σ_{r_i} are parameters in the model.

The Kalman filter has two stages: a time update stage and a measurement update stage. The first of these gives us a prediction for the current point and what the predicted covariance matrix looks like. The equations for this are

$$\tilde{\mathbf{z}}_t = A\mathbf{z}_t, \tag{2.11}$$

$$\tilde{P}_t = AP_tA^T + Q, \tag{2.12}$$

where $\tilde{\mathbf{z}}_t$ is the state estimate and $\tilde{P}_t$ is the state covariance estimate based on previous locations and the previous estimated state covariance,

$$P_t = \begin{pmatrix} \sigma_{x,x,t} & \sigma_{x,y,t} & \sigma_{x,\dot{x},t} & \sigma_{x,\dot{y},t} \\ \sigma_{x,y,t} & \sigma_{y,y,t} & \sigma_{y,\dot{x},t} & \sigma_{y,\dot{y},t} \\ \sigma_{x,\dot{x},t} & \sigma_{y,\dot{x},t} & \sigma_{\dot{x},\dot{x},t} & \sigma_{\dot{x},\dot{y},t} \\ \sigma_{x,\dot{y},t} & \sigma_{y,\dot{y},t} & \sigma_{\dot{x},\dot{y},t} & \sigma_{\dot{y},\dot{y},t} \end{pmatrix}.$$

The second stage takes the estimated values and covariance of the underlying state at time t and uses the measurement taken at time t to give a better estimate of the current state.

This starts by calculating the Kalman gain

$$K_t = \tilde{P}_t H^T [H\tilde{P}_t H^T + R]^{-1}. \tag{2.13}$$

which is then used to give the improved value of the underlying state estimate at time t

$$\mathbf{z}^*_{t+1} = \tilde{\mathbf{z}}_t + K_t(\mathbf{z}_{t+1} - H\tilde{\mathbf{z}}_t). \tag{2.14}$$

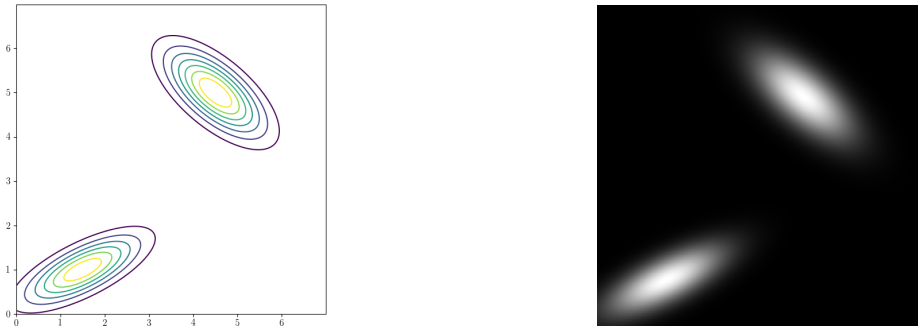
The final step is to update the state covariance similarly to above

$$P_{t+1} = [I - K_t H]\tilde{P}_t. \tag{2.15}$$

The resulting predictions, $\mathbf{z}_p$, are calculated by passing all locations up until time t into the filter as the measurements and then using Equation (2.11) to give a prediction on where to expect the sheep next.

We define the measurement covariance matrix R using the size of the sheep, i.e. we assume our measurement location could be placed anywhere on the silhouette of the sheep. The dynamical noise covariance matrix is similarly defined in relation to the size of the sheep, we set $\sigma^2_{q_1}$ and $\sigma^2_{q_2}$ to be $1/4$ of the size of the sheep in the x direction. We use the full size of the sheep for R and only $1/4$ of the size of the sheep for Q because we expect more noise in our measurements than in the underlying model.

To show these assumptions are correct we completed a simulation study where we simulated two silhouettes of different sizes moving towards each other. The silhouettes were created by calculating the probability density function for two bivariate normal distributions. The contour plot of these distributions and the resulting image can be seen in Figure 2.25. This can be converted into a binary image by converting areas where the density had a value greater than 0.5, to white, and areas where the value of the density



(a) Contour plot. (b) Greyscale image.

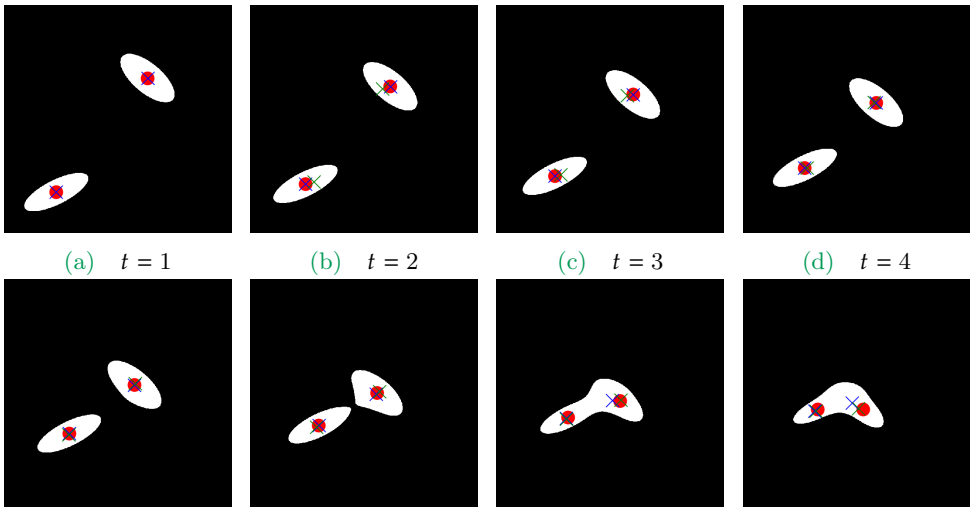
Figure 2.25: The p.d.f. of two bivariate normal distributions.

is less than 0.5 to black. This allows our simulated silhouettes to merge as they get close just as the silhouettes of the sheep in a data videos do.

The mean of each distribution is changed over time where the velocity is subject to normal distributed noise, with a small variance. We set the distribution at the lower left to move upwards and right, and the distribution at the top right to move down and left. Figure 2.26 shows the movement of the two simulated silhouettes and their eventual merging. The green markers in this figure show the result of using the Kalman filter, with the parameters mentioned above, in our algorithm. When the silhouettes first start to move the Kalman filter still contains a lot of noise from our initial state so the algorithm without the filter is more accurate. However, after only 3 time steps have passed the algorithms now give the same results. When the two silhouettes merge at $t = 7$ the algorithm without the Kalman filter no longer performs as well as the resulting fitted mean is further away from the true value, shown in red.

2.5.2 The Prediction Filter

The predictions from the Kalman filter are used to create a filter that removes any areas of where sheep are not expected. The prediction filter is created by fitting a bivariate normal distribution centred on each of our predictions. Each distribution has a covariance matrix so that the spread of the distribution is similar to that of the size and shape of



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$ (d) $t = 4$

(e) $t = 5$ (f) $t = 6$ (g) $t = 7$ (h) $t = 8$

Figure 2.26: Two simulated sheep silhouettes moving towards each other until they merge. The red point shows the mean of the distributions used to create the silhouette, the blue are the resulting means from running our sheep tracking algorithm without the Kalman filter, and the green is the resulting mean of running the algorithm with Kalman filter implemented. There is no green marker for $t = 1$ since the Kalman filter requires an initial velocity.

each sheep.

To determine the size and shape of the sheep, we take a single silhouette and extract a large number of points from each pixel. We want the coordinate of each pixel to be extracted E_n times, where E_n is proportional to the value of the pixel in the greyscale image. This will result in a high density of points where the image is white, where the sheep is, and a low density of points where the image is black. By taking the covariance of these points we obtain a matrix that can be used to describe the size and shape of the individual sheep. We can calculate the covariance of them using

$$C_j = \begin{pmatrix} \frac{1}{n}\sum_1^n p_{x,i}^2 - \bar{p_x}^2 & \frac{1}{n}\sum_1^n p_{x,i}p_{y,i} - \bar{p_x}\bar{p_y} \\ \frac{1}{n}\sum_1^n p_{x,i}p_{y,i} - \bar{p_x}\bar{p_y} & \frac{1}{n}\sum_1^n p_{y,i}^2 - \bar{p_y}^2 \end{pmatrix} \tag{2.16}$$

where n is the number of points we extract from the silhouette and C_j is the resulting covariance matrix for sheep j. This procedure is repeated for each sheep, the resulting bivariate normal can then be overlaid on the original image to produce Figure 2.27.

To use this filter we sum each of the distributions together to produce a prediction image, Figure 2.28. This image is then multiplied by the image produced by the maximum filter. This is our final chance for removing extraneous objects before converting the image to binary. This is done by running it through a final threshold but this time all points with a value more than t_u get set to $v_u = 1$ and all with a value lower than t_l are set to $v_l = 0$, where $t_l = t_u$, Figure 2.29.

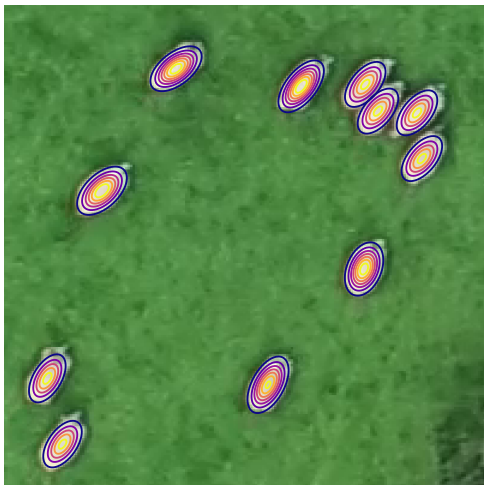


Figure 2.27: A probability density function of a bivariate normal with mean set as the predicted location of the sheep and covariance set to be the size of the sheep.

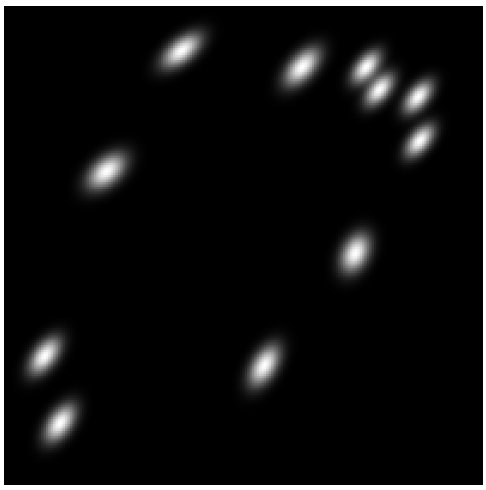


Figure 2.28: The prediction filter obtain by fitting bivariate normal distributions to each predicted location.

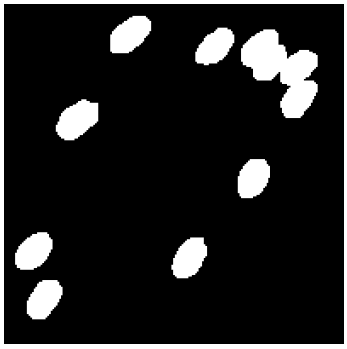


Figure 2.29: The binary image, showing a number of sheep silhouettes.

2.6 Extracting Locations From Each Silhouette

The binary image is used to find sheep in the full image. However, it does not contain enough information to give accurate location values. Therefore, each silhouette is considered individually so that only the local features are taken into account.

When we look at each silhouette individually we start by determining the number of sheep that are contained within the silhouette, as each silhouette may not contain only one sheep. To do this we look at the predicted points and count the number of points that lie within the silhouette.

In the previous section we discussed extracting points from pixels within a silhouette that depended on the value of the greyscale pixel. We repeat that process here, but this time we fit a two dimensional gaussian mixture model to the points. This is the same process from Section 2.4.4 where we fit a one dimensional mixture model to the greyscale value pixel value. Whereas before we knew the number of components needed for the mixture model, here we have to use the number of predicted points within the silhouette to determine the number of components our mixture model requires. Doing this we obtain a mean for each component which we take as the location of that sheep.

The final step of the extraction process is to match up locations through time. So that if necessary we could follow a single sheep though the full video. The Hungarian algorithm was designed to connect two sets of points together by minimising the cost of doing so. Here we set the cost to be the squared distance between the locations at time t and the locations at time $t + 1$. We square the distances to ensure that the correct sheep are matched together. Figure 2.30 shows the trajectories of the sheep in Video 1.

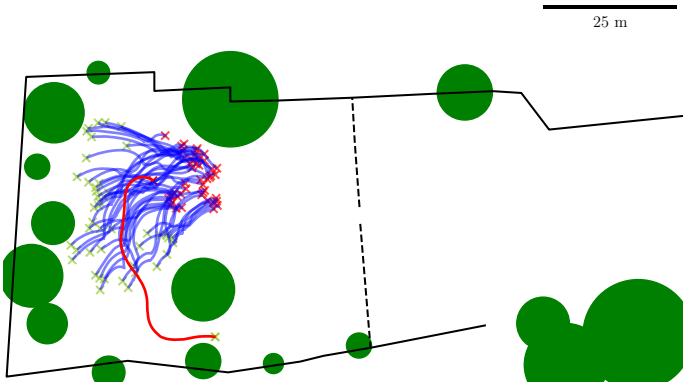


Figure 2.30: The trajectories of the sheep over 323 time steps in Video 1

2.7 Accounting For Drone Movement

In some of the drone videos the camera does not stay at a fixed location or angle, as it is in Video 1. This is due to stronger winds and the drone following the sheep on the ground to keep them in view. In these videos before detection can take place we must stabilise the footage in order to minimise the amount of movement of the sheep that is down to the movement of the drone. The OpenCV package has many builtin functions to allow us to achieve a more stable video.

To initialise the stabilisation we use the goodFeaturesToTrack function, to which we pass the greyscale image, the number of features we wish to track n_f, the relative quality of these features (q_l), the minimum distance between features and finally an image mask to define where we want these features to lie. This last part is the most important, we want the stabilisation code to focus on areas of the images which are not moving. This function works by finding the most prominent features in the specified image region using the method described in the paper by Shi and Tomasi [96]. Their method contains the following steps:

1. For each pixel (i, j) in the region of the image defined by the mask, calculate the quality measure, $Q_{i,j}$.
2. Then perform a non-maximum suppression, where only the local maximum in a 3×3 neighbourhood gets retained.
3. Points where $Q_{i,j} < (q_l \times \max\{Q_{i,j}\})$ are rejected.
4. The function then removes points for which there is a stronger point within the minimum distance.
5. Finally only the n_f points with the highest quality measure are returned to the user.

(a) Raw image: frame 0 (b) Raw image: frame 150

(c) Stabilised image: frame 0 (d) Stabilised image: frame 150

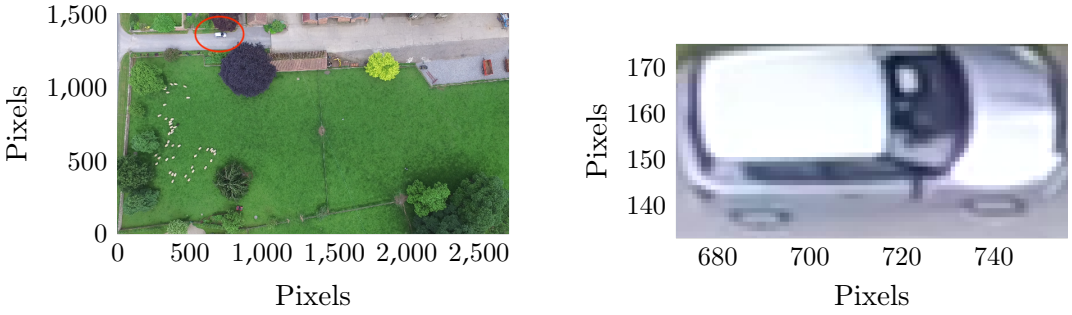
Figure 2.31: The effect of stabilisation on frame 0 and frame 150 of Video 8

For the first frame we only run this initialisation and add a border around the image to allow for subsequent movement, Figure 2.31a. For all subsequent frames we have a few calculations to perform. We need to see how many of the points detected by `goodFeaturesToTrack` can still be seen. We can then calculate the rigid transformation matrix using the `estimateRigidTransform` function, which describes how the new image should be rotated with respect to the previous frame. We also calculate a transformation matrix to describe how much the image has been warped using `warpAffine`, in our footage this could be due to the drone filming from a slightly different height or angle. Using matrix multiplication we can then rotate, resize or shift the raw image from the drone to give us stable footage. If we were to lose more than a third of the points identified in the initialisation due to camera shift, we can reinitialise using the first frame where the number of tracked points dropped below this threshold.

2.8 Transforming Extracted Coordinates

To convert the distances and locations from the reference frame of the size of pixels to length scales with units meters, we exploit the fact that a stationary car can be see throughout Video 1 and since all the footage was shot from the same height we can assume that this conversion works for all thirteen videos. Figure 2.32 shows a frame from Video 1, at the top of the image, a car can be seen parked up at the side of the road.

Using the graphical interface of Python we can crop the image down to show just the car, Figure 2.32b. We calculate the width of the resulting image to be 85 pixels and from manufacturers we can get the length of the car, 3992mm [97]. So we calculate that there



(a) A car can be seen parked up at the side of the road, shown circled in red.

(b) By showing just the car we can extract the length of the car in pixels.

Figure 2.32: We use the known size of the car shown in the frames to convert the lengthscales from pixels to meters.

is approximately $3992/85 \approx 47$mm per pixel.

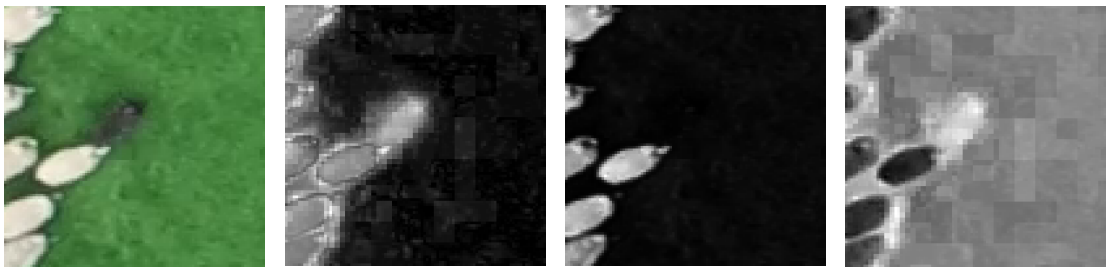
2.9 Extracting Other Features Of The Fields

The white sheep are not the only things we need to locate in the video for comparison with the mathematical models. We also need to track any black sheep, and anything acting as a predator, such as the quadbike or the farmers. We also require the location of any solid boundaries the sheep may encounter, hence we need to extract the location of the fences and trees. In the following sections we will describe the extraction process for these other features.

2.9.1 The Black Sheep

Detecting and consequently extracting the location of the black sheep in the drone footage works in similar way to that of the white sheep. At the start of the process of detecting and tracking the white sheep there is a preprocess stage, where the colour image is converted to greyscale in a way as to enhance the difference between the white sheep and the green grass. Now that we have a black sheep this process is much more convoluted. We start by cropping the image down around the black sheep, Figure 2.33a. We then calculate the Mahalanobis distance between the value for each pixel and the average colour of the grass background. How to calculate this distance, and why it was not appropriate for the white sheep can be seen in Appendix B. This distance can be displayed as a greyscale image, Figure 2.33b, where the black sheep can be seen as a pale grey shadow, but the white sheep are bright white.

To remove the bright white areas that contain the white sheep we also need to calculate



(a) Cropped colour image (b) Mahalanobis distance (c) Product of colour channels (d) Difference between (b) and (c)

Figure 2.33: Detecting the black sheep using the difference between the Mahalanobis distance and the product of the colour channels as our greyscale image with the black sheep the brightest point of the image.

the product of the three colour channels, Figure 2.33c. This allows us to identify any white sheep that may be present in the image, but does not show the black sheep. To remove the presence of any white sheep we need the difference between the Mahalanobis distance and the product of the colour channels as our greyscale image, Figure 2.33d. This process leaves pixels where the white sheep are with negative values, and the area where the black sheep is with positive values. We then threshold the image so that all negative pixel are reassigned the value 0.

This leaves us with a greyscale image where the brightest part is the object we are wanting to detect. We are in the same scenario as we had previously with detecting the white sheep. The rest of the detection and tracking of the black sheep follows as before with the white.

2.9.2 The Quadbike

Tracking the quadbike follows a simplified procedure compared to tracking the white sheep. When detecting the sheep we used gamma correction as it emphasised the difference between the sheep and the grass when looking to extract the location of the quadbike, as mentioned earlier, we take the difference between the red and green colour channels, Figure 2.34. This enhances the difference between the red quadbike and the green grass. We then threshold the image using the method described in Section 2.4.4.

Once the image has thresholded we have a number of pixels which are white. We use the mean coordinate of these pixels to be the centre of the quadbike. We can do this now, when we could not do this before with the sheep, because we are now only tracking a single object, so all brightened pixels are part of the same object, the quadbike.

(a) Cropped colour image

(b) Difference between red and green colour channels

Figure 2.34: Detecting the quadbike using the difference between the red and green colour channels as the greyscale image.

2.9.3 Fences And Trees

The trees and fences are the simplest objects to detect as they do not move so this operation does not need to be automated. We use the graphical interface of python to display the full image and locate coordinates to the closest pixel. When looking to extract the location of the fences we assume that the fences are perfectly straight between each corner or end point. Therefore we only have to find the corners of the field and where the fence ends. Figure 2.35 shows the location of these points in red, and where these points are connected by a fence a solid black line connects them.

In the following Chapter we will explore what information can be extracted from the trajectories about the movement of the sheep such as their speed and the global alignment of the flock.

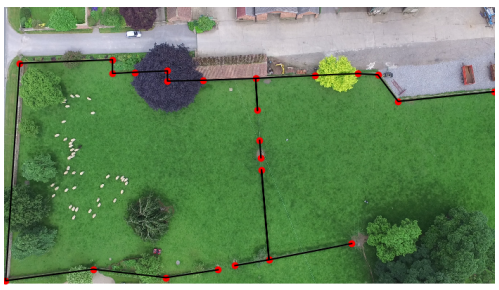


Figure 2.35: The fence sections located in Video 1 using the graphical interface of python, overlaid on top of a frame from the data video.

Figure 2.36: The 16 trees located in Video 1 using the graphical interface of python, overlaid on top of a frame from the data video.

3

Analysis of Observations

3.1 Introduction

In this Chapter we will be exploring the information we can extract from the sheep trajectories we extracted using the computer vision algorithm described Chapter 2. By examining the trajectories, information about the individual sheep's speed, acceleration and direction of travel can be extracted. In this chapter we will also look at aspects of the flock as a whole such as the area covered by the flock, and how well aligned the sheep are within the flock.

3.2 Overview

We have 14 videos in total, filmed over two days across two fields. For each video we extracted the location of the sheep, and quad bike if visible. The shortest dataset we obtained was for Video 11, where there are 45 sheep across 192 frames of footage. The longest dataset, Video 3, is about 3 times larger and contains the information of the movements of 45 sheep for 593 frames. Table 3.1 shows the number of sheep in each video and how many frames we were able to track them for[1].

In addition to tracking the sheep, with Videos 1-8, 10-13 we were able to track the quad bike for the duration of the full video. In Video 9, however, the bike is off screen until frame 80, approximately 1/4 of the way through the video. In this case we were only able to track the quad bike for 220 frames out of a possible 300 frames. In the final video, Video 14, the quad bike never becomes visible, however two farmers can be seen herding the sheep. In this video we use the location of the farmers as the source of the stimulus causing the sheep to move. The farmers appear in the footage as a collection of very pale pixels, and so in this case we track the location of the farmers using the same method we used to track the sheep. For other videos where the farmers appear to be interacting with the sheep they are also tracked.

[1] There are 24 frames per second.

		Video Duration		⟨Speed⟩ (ms^{-1})	
Video	# Sheep	Seconds	Frames	Sheep	Quad
1	45	17.50	420	1.20	2.66
2	45	14.67	352	1.35	3.17
3	45	24.71	593	2.46	2.69
4	45	13.46	323	2.19	2.56
5	45	9.88	237	2.08	3.63
6	45	10.25	246	1.41	1.84
7	45	11.33	272	3.20	1.80
8	70	6.25	150	1.78	3.49
9	70	12.50	300	3.70	*4.00
10	45	9.71	233	2.81	3.06
11	45	8.00	192	4.55	6.32
12	45	20.38	489	2.43	2.15
13	45	10.17	244	4.00	1.85
14	45	17.54	421	3.18	-

Table 3.1: Summary of data videos. There is no visible quad bike in Video 14 so the speed could not be recorded. *Quad bike tracked for only part of the video.

3.3 Estimating Sheep Size

In the final step of the detection process, Section 2.6, in order to to extract a location, we fit a 2D gaussian mixture model to each silhouette, which at time t can be written as $s_i^{(t)}$, where $s_i^{(t)}$ is simply the notation we have given for identifying silhouette i at time t. Each silhouette may cover several sheep, so this fitting results in $n_{s_i^{(t)}}$ means and covariance matrices, where $n_{s_i^{(t)}}$ is the number of sheep within the i^{th} silhouette at time t. For each sheep, we have a Gaussian distribution which describes the size and shape of the individual sheep at each moment in time.

When sheep are aligned with the x axis or the y axis then the covariance matrix has zero off-diagonals. The width and length of the sheep would then directly relate to the standard deviation, σ_x and σ_y, in the x and y direction.

Assuming the sheep is aligned we first aim to relate an ellipse, (aligned so that the major and minor axes align with the x and y axes) with the Gaussian fitted to the sheep data. Geometrically, such an ellipse is written in standard form as

$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 = 1 \tag{3.1}$$

where r_x is the radius in the x direction and r_y the radius in the y direction.

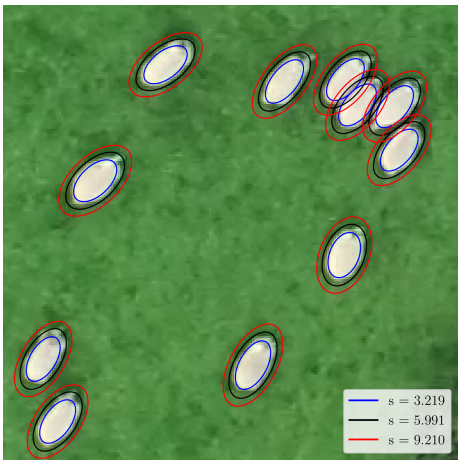


Figure 3.1: Confidence levels of $s = 3.219$ underestimates the area of the sheep. When $s = 9.210$ it over estimates the area. When $s = 5.991$ it consistently gives a good visual approximation of the area covered by each sheep.

We begin by writing an ellipse with major and minor axes in the same ratio as the standard deviations σ_x and σ_y of our Gaussian distribution, and with variable size. Such an ellipse will have the form

$$\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2 = s, \tag{3.2}$$

parameterised by the scale factor, s, known as the Mahalanobis radius of the ellipsoid [98].

We then choose s such that an interval, p, is formed by the shape of the ellipse. To find s, consider that since x and y have a bivariate normal kernel, the left hand side of Equation (3.2) has the form of a χ^2 distribution, with two degrees of freedom [99].

This can be easily solved using Python or other statistical computing packages such as R. Visual inspection of fit, Figure 3.1, suggests that $s = 5.991$ is a reasonable choice. This corresponds to 95% of the sheep falling into the ellipse.

In the general case where the covariance between the x and y directions for the sheep's fitted distribution is non-zero we still use this method largely unchanged, but in a rotated coordinate system. Instead of using the standard deviations in the x and y direction as the ratio of the radii of the ellipse we use the eigenvalues of the covariance matrix. These eigenvalues represent the spread in the direction of the eigenvectors, which under a rotated coordinate system are aligned with the x, y axes.

By definition a covariance matrix is positive definite therefore all eigenvalues are positive and can be seen as a linear transformation to the data. The radii of the standard ellipse in this general case are $\sqrt{s\lambda_1}$ and $\sqrt{s\lambda_2}$, and the resulting ellipse is rotated via the covariance matrix eigenvectors.

	⟨Area⟩		⟨Width⟩		⟨Length⟩	
Video	Pixels^2	m^2	Pixels	m	Pixels	m
1	307.	0.68	15.	0.70	26.	1.23
2	320.	0.71	13.	0.63	31.	1.43
3	286.	0.63	13.	0.59	29.	1.37
4	332.	0.73	15.	0.70	28.	1.34
5	282.	0.62	13.	0.59	29.	1.34
6	285.	0.63	13.	0.59	29.	1.36
7	308.	0.68	13.	0.61	30.	1.41
8	358.	0.79	15.	0.72	30.	1.40
9	438.	0.97	17.	0.82	32.	1.50
10	286.	0.63	14.	0.66	26.	1.22
11	321.	0.71	12.	0.59	33.	1.54
12	323.	0.71	13.	0.63	31.	1.45
13	305.	0.67	12.	0.57	32.	1.49
14	257.	0.57	11.	0.54	29.	1.34

Table 3.2: Summary of the average size of sheep in each data video, using the 95% confidence ellipse of the bivariate normal fitted to each sheep.

We denote the major axis and the minor axis of the ellipse by r^+ and r^-, these are equivalent to

$$r^+ = \sqrt{s\lambda^+}, \quad \text{and} \quad r^- = \sqrt{s\lambda^-}$$

where $\lambda^+ = \max\{\lambda_1, \lambda_2\}$ and $\lambda^- = \min\{\lambda_1, \lambda_2\}$. The result of this process is shown in Figure 3.1.

To obtain the values in Table 3.2 we calculated the length of the major and minor axis for each sheep across the video and took the average. The area covered by the sheep is calculated using $A = \pi r^+ r^-$.

3.4 Post Processing

A small amount of post processing is required to the data, once it has been extracted from the videos. The tracking process adds a small amount of noise to the locations detected. When locating the sheep, the extracted coordinate may not be the centre of the sheep's back, it might be the top of its head or down by its tail. This is due to the way we find sheep by looking from bright white pixels, it might be the case that for a single image the brightest part of the sheep is its head whereas in the next image the brightest part to be its tail. When plotting a trajectory of this it would appear as though the sheep moved backwards for a short time, Figure 3.2. We denote the direction of sheep i at time t to be

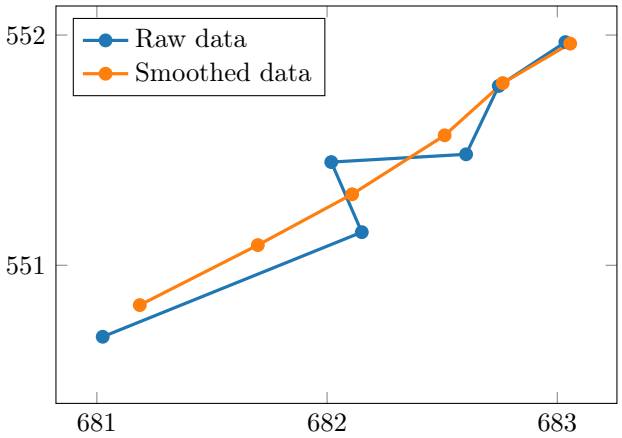


Figure 3.2: The original trajectory (–) of a single sheep in video 1 from frame 49 to frame 55 which has an artefact. The smoothed trajectory (–) no longer has this.

$\phi_i^{(t)}$ then an artefact is defined as when

$$\left|\phi_i^{(t)} - \phi_i^{(t-1)}\right| > \frac{\pi}{2}.$$

Given we are looking at sharp changes in direction between two consecutive frames, $\frac{1}{27}$ of

	Before P.P.		After P.P.	
Video	# Artefacts	%	# Artefacts	%
1	3	0.02	0	0.00
2	78	0.49	0	0.00
3	127	0.48	2	0.01
4	31	0.21	0	0.00
5	53	0.50	0	0.00
6	33	0.30	0	0.00
7	95	0.78	2	0.02
8	92	0.88	2	0.02
9	39	0.19	2	0.01
10	2	0.02	2	0.02
11	20	0.23	0	0.00
12	137	0.62	3	0.01
13	47	0.43	0	0.00
14	38	0.20	0	0.00

Table 3.3: The number of tracking artefacts in the trajectories of the sheep for each video when all the sheep are moving with a speed greater than 0.5 pixels/frame (Case numbers will be removed for final version)

a second, it is unrealistically quick to move that way. We can only detect artefacts when the sheep is moving, i.e. $\dot{\mathbf{z}} > 0.5$ so the number of artefacts we can detect in the video datasets can be seen in Table 3.3.

To reduce the number of these artefacts in our data we use a moving average process with a window size of 5. This window size needs to be minimised so that artefacts are removed while keeping as much of the true movements of the tracked objects. The window size of the moving average process should be odd, therefore we have a limited number of options for what ours can be. We take 5 as this removed most of the artefacts that occurred while the sheep were in motion, whereas 3 was prone to leaving some behind. Table 3.3 shows the number of artefacts that were present before and after the post-processing process. Hence, all summaries based on the location of tracked objects will therefore be calculated using the smoothed data.

3.5 The Movement Of The Quad Bike

From the (x, y) coordinates of the quad bike, denoted $q(t)$, we can see how the speed and acceleration changes over the course of each of the videos. We calculate an estimate of the speed of the bike via,

$$\dot{q}_t = q_t - q_{t-1}. \tag{3.3}$$

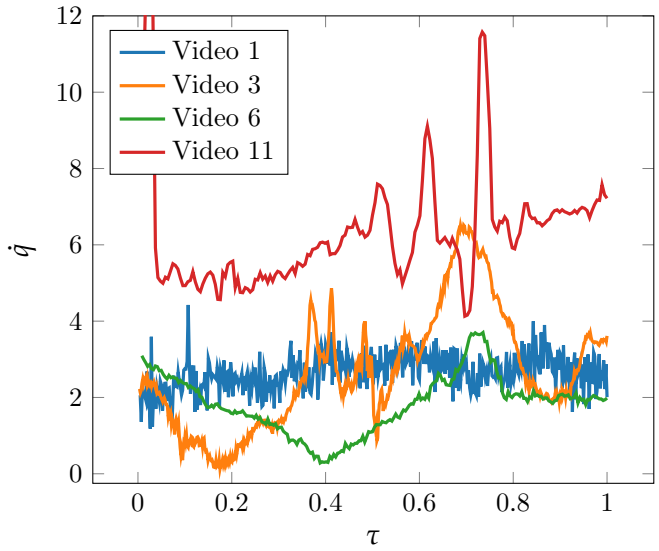


Figure 3.3: The speed of the quad bike (ms^{-1}) plotted against dimensionless time for videos 1, 3, 6 & 11. The orange line shows the typical behaviour for videos where the quad bike comes to rest.

We chose not to smooth the trajectories of the quad bike as we do not want to infer any information about the "behaviour" of the bike, we collected this information to give context to the behaviour of the sheep and for general completeness.

As can be seen in Table 3.1 in most videos the quad bike travels around 3ms^{-1}. Videos 6, 7 & 13 have an average speed less than this due to the quad bike slowing to be almost stationary, so that the bike does not get too close to the sheep. For Video 11, where the quad bike is going at twice the speed of the other videos, the sheep appear to be herding themselves across the field, so the bike has to catch them before they escape. It is useful to have a dimensionless time variable, so that videos of different lengths can be compared. Our dimensionless time variable is defined as

$$\tau = t/\max(t) \tag{3.4}$$

The speed of the quad bike four of the videos can be seen in Figure 3.3. The shape of the speed over time in Video 1 (–) and 3 (–), show that even though that have similar average speeds, they can look quite different. Video 6 (–), has a typical speed profile for videos when the quad almost comes to rest. Also shown is the speed of the quad bike throughout Video 11 (–), where the average speed is 2 times larger than that of Video 1.

3.6 The Movement Of The Sheep

Following the analysis of the quad bike, we can also look at the movement of the sheep over each video. By looking at the coordinates of the sheep, $\mathbf{z}$, we can compare many aspects of the flock, including speed and acceleration, and also how the size of the flock changes over time, or how the direction of travel of the flock changes.

3.6.1 Speed Of The Sheep

Visually inspecting the speed of the sheep across the video we can cluster the videos into three different groups. Looking first at the speed of the sheep over the course of the videos, we see that in videos 1-8 the speed has a wavelike pattern. For videos 9-12 the speed of the sheep decreases over time, and for the final two videos we see an increase in speed. Therefore in videos 1-8 we will see emergent flocking behaviour, i.e. what happens when the sheep start approximately still and start to flock. In the remaining videos we have sections where the speed is approximately constant, here we will see what happens when the flock is established. Figure 3.4 shows an example of each of these three different behaviours; (–) shows the average speed of the sheep in Video 1. In this video we see a wavelike speed profile implying that this video will show small segments of emergent

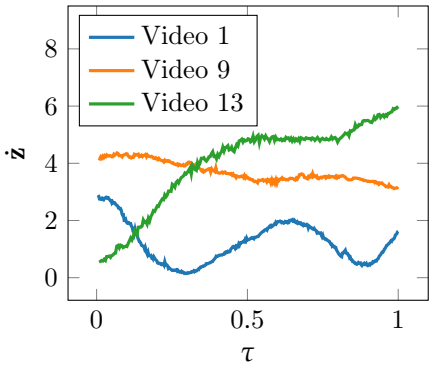


Figure 3.4: The mean speed of the sheep plotted against dimensionless time for videos produces three different shapes; (–) the flock's speed fluctuates, (–) the speed decreases slightly, (–) the speed of the flock increases.

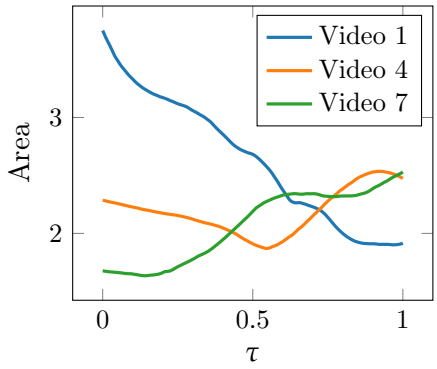


Figure 3.5: The area covered by the flock produces three distinct behaviours; (–) the flock gets more compact, (–) the area of the flock fluctuates, (–) the flock disperses.

flocking behaviours when the speed is increasing. Video 9 (–) is an example of slightly decreasing speed, in videos with speed profiles like this we would expect to see established flocking behaviours. Finally an example of an increasing speed profile can be seen in Video 13 (–) which will contain a single emergent flocking event.

3.6.2 Area Of The Flock

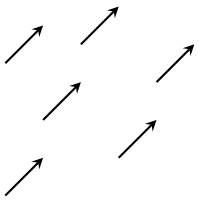
We see three different behaviours when looking at the area covered by the flock, Figure 3.5. We calculate the area of the flock by computing the area of the convex hull of the coordinates of the sheep, Section A.4. For videos 1-3,10 we see that the area the flock covers over time decreases (–), hence the flock becomes more densely packs as they move. We see the opposite behaviour when looking at videos 7-9,11,13-14, the area the flock covers increases over time (–), the sheep spread out. For the remaining 4 videos we see that the area covered by the flock follows a wave like pattern (–).

3.6.3 Alignment Of Sheep

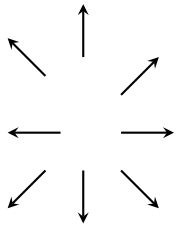
How well aligned the sheep are at any given time can be calculated using

$$\psi^{(t)}\left(\mathbf{z}_{[i]}\right) = \sqrt{\frac{1}{N_i}\left(\sum_{j=1}^{N}\cos\left(\theta_j^{(t)}\right)^2 + \sum_{j=1}^{N}\sin\left(\theta_j^{(t)}\right)^2\right)}, \tag{3.5}$$

where N_i is the number of sheep in Video i and $\theta_j^{(t)}$ is the direction of sheep j at time t in Video i. The alignment values vary from 0, when the sheep are perfectly unaligned, to



(a) Alignment is maximised, $\psi^{(t)} = 1$, when all sheep have the same direction.



(b) Alignment is minimised, $\psi^{(t)} = 0$, when the directions are random.

Figure 3.6: Schematics showing maximal and minimal alignment values

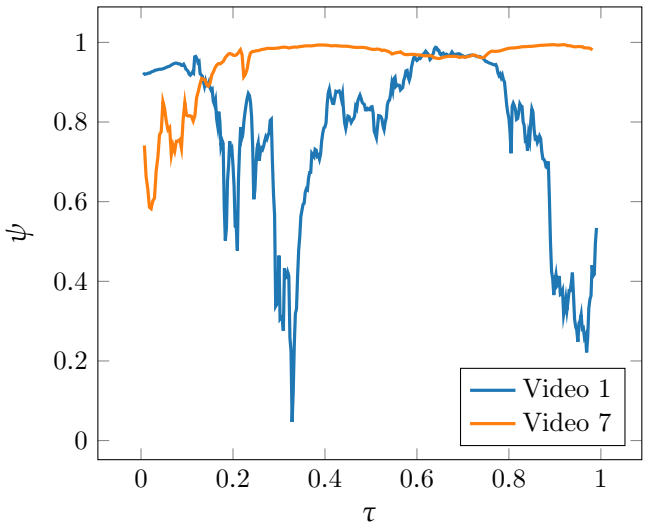


Figure 3.7: The alignment plotted against dimensionless time for videos 1 and 7. These are stereotypical of the two different behaviours that are observed.

1, where the sheep are perfectly aligned. Diagrams showing these behaviours can be seen in Figure 3.6.

Look at the resulting alignment over time plots of all the videos we can see two different behaviours. For videos 1-6 we see that the alignment fluctuates between almost perfectly aligned, $\psi^{(t)} \approx 1$, and and much lower values of alignment, $\psi^{(t)} < 0.3$. For the remaining videos we see that most of the time we have high alignment values, $\psi^{(t)} > 0.8$, for the full video, Figure 3.7.

3.6.4 Velocity correlation

The velocity correlation function often used in fluid dynamics, $f(r)$, for fixed t is defined as

$$f(r,t) = \frac{\langle v_x(\mathbf{r},t)\, v_x(\mathbf{r}+r\hat{\mathbf{e}}_x,t)\rangle}{\langle v_x(\mathbf{r},t)^2\rangle}, \tag{3.6}$$

where $\mathbf{r}$ is the position vector and v_x the velocity in the x direction [100, 101]. The notation $\langle\cdot\rangle$ denotes average over fixed distance r. In conventional simulations the points lie on a square lattice with spacing r.

Since we do not have points on a grid, i.e. the sheep are free to take any location, we define our velocity correlation function slightly differently. We start by defining the velocity in

the direction from agent i to its neighbour agent j at time t. We denote the velocity in this direction $v^{(t)}_{i\to j,i}$ for agent i and $v^{(t)}_{i\to j,j}$ for agent j. The velocity component in this direction for agent i is

$$v^{(t)}_{i\to j,i} = \frac{\mathbf{v}^{(t)}_i \cdot \left(\mathbf{z}^{(t)}_j - \mathbf{z}^{(t)}_i\right)}{\left|\mathbf{z}^{(t)}_j - \mathbf{z}^{(t)}_i\right|}, \tag{3.7}$$

and agent j is

$$v^{(t)}_{i\to j,j} = \frac{\mathbf{v}^{(t)}_j \cdot \left(\mathbf{z}^{(t)}_j - \mathbf{z}^{(t)}_i\right)}{\left|\mathbf{z}^{(t)}_j - \mathbf{z}^{(t)}_i\right|}. \tag{3.8}$$

where $\mathbf{v}^{(t)}_i$ and $\mathbf{z}^{(t)}_i$ is the speed and location of agent i at time t.

We separate each pair of sheep (i, j) into M distinct bins $B_0, B_1, \ldots B_{M-1}$, based on the distance measure between them, $r^{(t)}_{ij}$. We have,

$$B_m = \left\{(i,j) \;\middle|\; r^{(t)}_{ij} \geq \frac{m}{M}\max\left(r^{(t)}_{ij}\right), \quad r^{(t)}_{ij} < \frac{m+1}{M}\max\left(r^{(t)}_{ij}\right)\right\}$$

We then define the the correlation function,

$$f(x_m, t) = \frac{\left\langle v^{(t)}_{i->j,i} v^{(t)}_{i->j,j}\right\rangle_{B_m}}{\left\langle {v^{(t)}_{i->j,i}}^2\right\rangle_{B_m}} \tag{3.9}$$

where $\langle\cdot\rangle_{B_m}$ denotes averaging the sheep in the m^{th} bin B_m. The average distance between sheep in bin B_m is defined as $x_m = \left\langle r^{(t)}_{ij}\right\rangle_{B_m}$.

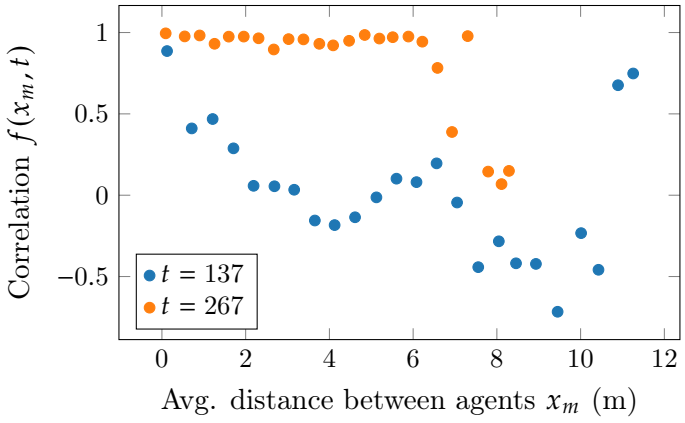


Figure 3.8: The velocity correlation when $t = 137$ and $t = 267$ for Video 1.

In Video 1 the alignment increases from approx 0 at frame 137 to approx 1 at frame 267. Looking at the velocity correlation at $t = 137$ we obtain the blue points (–) and for $t = 267$ we obtain the orange points (–) in Figure 3.8. Other than a few outliers, the correlation is low for most distances when $t = 137$ which reinforces the idea that the directions of the sheep are random. When the alignment increases the speed also increases so we see that at $t = 267$ the velocity correlation, f is approximately 1 for most measured distances between agents.

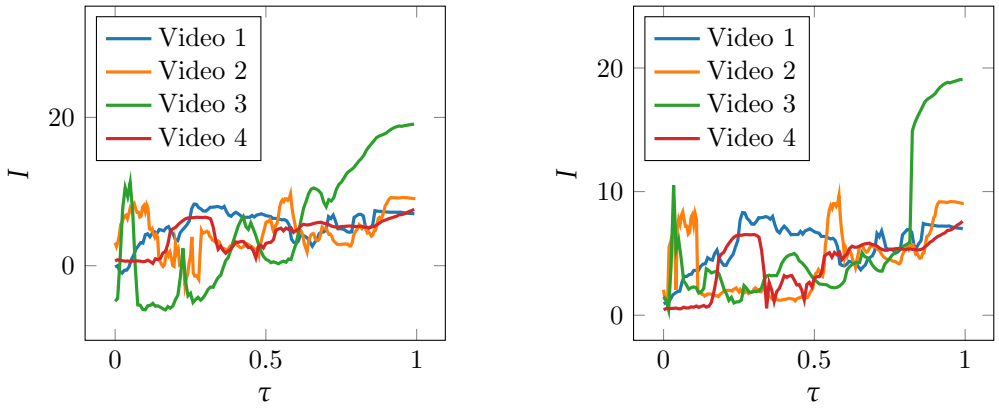
3.6.5 Integral Lengthscale

We can calculate the distance over which the velocities are correlated using the integral length scale, which is defined at time t for the dataset $\mathbf{z}$ as

$$I(t) = \int_0^{\max\left(r_{ij}^{(t)}\right)} f(x,t)\,\mathrm{d}x. \tag{3.10}$$

We approximate the integral $I(t)$ numerically for each point in time t by using the trapezium rule over our M bins at $x = x_0, \ldots, x_m$.

This can be calculated for each of our overall time points. The integral lengthscale for Videos 1,2,3 and 4 is shown in Figure 3.9a where τ is time rescaled as earlier in this Chapter, but instead of covering the whole Video, $\tau = 0$ now shows the time where the alignment is low and is starting to increase, and $\tau = 1$ is the time where the alignment is high and has finished increasing. Due to this period of time being chosen we expect the integral lengthscale to increase over time.



(a) The integral lengthscale for four videos using Equation (3.10).

(b) The integral length scale for four videos, using Equation (3.11).

Figure 3.9: The integral lengthscale of four videos, Videos 1 to 4, over the time period where alignment goes from approximately zero until the time point where alignment stops increasing using two different equations to calculated I.

The calculation for the integral lengthscale can be modified to make sure that the length it produces is always positive. This is done by only integrating up to a critical distance, x_c, i.e.

$$I(\mathbf{z}, t) = \int_0^{x_c} f(x, t)\, \mathrm{d}x, \tag{3.11}$$

where the critical value is defined as the x_m associated with the first bin B_m where the correlation goes below 0. Figure 3.9b shows the result of using this formation of the integral lengthscale for the same four videos as seen in Figure 3.9a over the same time period.

Now that we have observational data such as average speed, average global alignment and average distance between nearest neighbours we can start to compare these quantities to simulated data from mathematical models of collection motion. In Part III we will introduce in detail a number of mathematical models which could be used to model sheep being actively herded across a field. We will then compare some of these models with the appropriate observations.

III

Models And Data Comparisons

4

Mathematical Models For Collective Motion

4.1 Introduction

In the introduction to this thesis, we introduced a number of ways that have previously been proposed to model collective motion. In this chapter we will look at four different models from the literature that have been suggested that the resulting trajectories could be comparable to the data we collected. Each of these models incorporates a different scenario into the model such as agents interacting with other agents with a different set of behavioural rules, agents interacting with a predator and agents interacting with walls. In addition to the models from the literature we will also introduce a number of models of our own. These are extensions of these previously documented models of collective motion.

The first of the models from the literature is the Vicsek model. In the original paper [38] output from this model was visually examined to determine that the behaviour was similar to that seen in biological systems. However, since then many people have modified the simple Vicsek model adding in new interaction and compared the output to observational data. In 2002 Couzin et al. [44] modified the Vicsek model and compared that to observations of birds flocking. The second of our model in this chapter, Ginelli model, also uses the Vicsek model to approximate the way sheep move while grazing in a field. They compared their model to a number of observations they collected of sheep grazing. The full Ginelli model differs from the Vicsek model by having different interactions between agents based on their state: standing, grazing, running. It also incorporates how an agent would transition between these behavioural states. Our third model, the Chen model [45], is used to approximate the movement of animals in a predator prey situation. They did not have observational data to compare their results too, so they visually compared the shape of empty region surrounding the predator in their model to photographs of shepherds herding sheep in Argentina. This model differs from the previous two in two main ways. The most obvious of these is the fact that the model comprises of predator and

prey agents. The second way is that all agents are interacting with each other at all times. In the original Vicsek model agents only interacted with those within a fixed radius. In this model the neighbours closer to an agent interact more strongly, but even those very far way with contribute to the agents movements. The final model from the literature in this chapter is the Helbing model which looks at the way people move when trying to escape a room in panic. Helbing et al. [36] collected observational data of people leaving a room and compared this to the simulated data from their model. This model is the only one of the four models discussed here to incorporate physical size to the agents and to include physical boundaries. In the previous models the agents are modelled as point-like particles, but in this model the interactions changes based on whether or not and agent is pressed up against other agents or the boundaries of their environment.

4.2 Vicsek Model

A model devised by Vicsek et al. [38] is constructed in a simple way to model collective group behaviour where agents align with their nearest neighbours. The Vicsek model is an evolution of the Ising model which represents magnetic dipole moments of atomic "spins" in a lattice structure with can only be one of two states +1 and -1 [102]. The Ising model allows each spin to interact with its neighbours. When the neighbouring spins match the energy is lower than when the spins are opposed. The system then tries to minimise the total energy. The Ising model allows simulation of this system to be run in order for phase transitions to be identified [102]. The Vicsek model is similar to this but where agents are no longer constrained to a lattice. Since this requirement was lifted the phase transitions seen in simulations of the Vicsek model do not break the Mermin–Wagner theorem which states that continuous symmetries cannot be broken at finite temperatures when in two dimension [103]. The agents in this model are point-like objects where agent i at time t

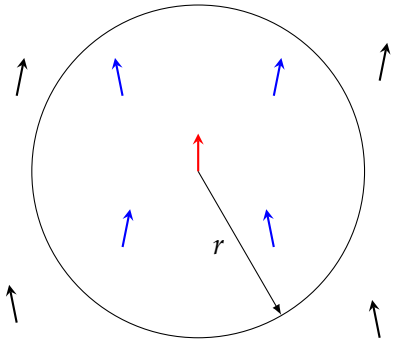


Figure 4.1: The set of neighbours (blue) of the centred agent (red) lie inside the circle of radius r. The remaining agents (black) outside the circle are not part of the set $\mathcal{N}_i$

has a location $\mathbf{x}_i^{(t)}$, and direction $\theta_i^{(t)}$. The position of agent i going from time t to time $t+1$ is calculated using the Euler time step approximation, Section A.2.1, with a step size $dt = 1$, and velocity of the form

$$\dot{\mathbf{x}}_i^{(t)} = v\left(\cos\theta_i^{(t)}, \sin\theta_i^{(t)}\right), \tag{4.1}$$

where v is the speed of the agents. The speed is fixed, both in time and over all of the agents. The direction of an agent, $\theta_i^{(t)}$, is updated by finding the average of the direction of its neighbours within a set distance r, Figure 4.1, with some small noise, i.e.

$$\theta_i^{(t+1)} = \left\langle \theta_j^{(t)} \right\rangle_r + \xi, \tag{4.2}$$

where $\xi \sim U(-\eta\pi, \eta\pi)$. The magnitude of this noise is controlled by changing the size of η. Vicsek et al. [38] varied this quantity between 0–5. In the original 1995 Vicsek paper this quantity represented the temperature of the model. In our simulations of the Vicsek model, and its modifications, we have removed the noise term to reduce the dimensions of the parameter space. Vicsek et al. [38] define the average direction to be

$$\begin{aligned}\left\langle \theta_j^{(t)} \right\rangle_r &= \arctan\left[\frac{\left\langle \sin\left(\theta_j^{(t)}\right)\right\rangle_r}{\left\langle \cos\left(\theta_j^{(t)}\right)\right\rangle_r}\right], \\ &= \arctan\left[\frac{\frac{1}{\left|\mathcal{N}_i^{(t)}\right|}\sum\limits_{\mathcal{N}_i^{(t)}} \sin\left(\theta_j^{(t)}\right)}{\frac{1}{\left|\mathcal{N}_i^{(t)}\right|}\sum\limits_{\mathcal{N}_i^{(t)}} \cos\left(\theta_j^{(t)}\right)}\right].\end{aligned}$$

as due to the cyclic nature of angles the numerical mean of the directions can not be used to calculate this quantity. Where $\mathcal{N}_i^{(t)}$ is the set of indices of the neighbours of agent i at time t. The quantity $\left|\mathcal{N}_i^{(t)}\right|$ is the cardinality of this set which is the number of agents which agent i interacts with at time t. This is the same as calculating the weighted average of the agents directions using

$$\left\langle \theta_j^{(t)} \right\rangle_r = \arctan\left[\frac{\sum\limits_{j=1}^{N} w_{ij}^{(t)} \sin\left(\theta_j^{(t)}\right)}{\sum\limits_{j=1}^{N} w_{ij}^{(t)} \cos\left(\theta_j^{(t)}\right)}\right] \tag{4.3}$$

where the weights are calculated using a heaviside step function

$$w_{ij}^{(t)} = H\left[r_{ij}^{(t)} - r\right] = \begin{cases} 0, & r_{ij}^{(t)} - r \geq 0, \\ 1, & r_{ij}^{(t)} - r < 0, \end{cases} \tag{4.4}$$

and $r_{ij}^{(t)}$ is the distance between agent i and agent j at time t.

In the following four sections we discuss modifications to the Vicsek model to enhance the model's ability to recreate emergent collective behaviours. We modify the Vicsek model in order to be able to compare to the observational data which we collected and discussed in Chapter 3. For example the simulated agents in the Vicsek model are constrained to moved with fixed speed, whereas in most of our observational datasets the sheep do not move with constant speed. Therefore we require a model which allows the speed of the simulated agents to change over time. Other models [44, 49–56, 104–109] have been created by modifying the model by Vicsek et al. [38]. Couzin et al. [44] introduced two more regions of interaction: a small inner zone of repulsion, then a larger alignment interaction zone and finally a large zone of attraction. By having these three zones of interaction they were able to reproduce behaviours such as shoaling, rotational motion and highly aligned motion. Other models have including restricting agents to interact with only a fixed number of neighbours [108] or having a restricted field of view [107, 109].

4.2.1 Modification 1: Variable Speed

In biological systems the speed of the animals is rarely constant, so we require a model which allows agents to have variable speed. We allow the speed of the agents to change by using,

$$v_i^{(t+1)} = \sum_{j=1}^{N} w_{ij} v_j^{(t)}. \tag{4.5}$$

In this model modification we use the same equation, Equation (4.4), for calculating the weights of the interactions. So when the initial speed of the agents are the same we retain the original Vicsek model. However, when the agents are initialised with differing speeds then the average speed of the agents will change over time.

When the agent's speed changes in this manner the resulting behaviour can model scenarios such as stampedes or emergent flocking. One way to get this type of behaviour would be to initialise a single agent with a faster speed, when it encounters other agents their speed will also increase.

4.2.2 Modification 2: Gaussian Interactions

The second modification we make is to the weights function. When using the heaviside function to calculate the weight of the interaction between agents we are in a situation where an agent's neighbour is either included or excluded depending on the distance between them. Therefore by changing the function to calculate the weights to a Gaussian kernel we allow an agent's speed and direction to be effected by all other agents but those further away have less influence. This is similar to the modifications Cucker and Smale [49] made to the Vicsek model in 2007. The new weights are calculated by

$$w_{ij}^{(t)} = \exp\left\{-\frac{r_{ij}^2}{\sigma^2}\right\} \tag{4.6}$$

where the variable $\sigma > 0$ plays an analogous role to that of the variable r in the original Vicsek model. Rio et al. [110] found that interaction strength, for both direction and speed, between a pedestrian and its neighbour decreased with distance.

4.2.3 Modification 3: Interactions Dependent On Neighbours Speed

Until now neighbouring agents at the same distance have been given the same weight, regardless of their speeds. We want the weighting of neighbours to depend on speed. As it is logical to assume that faster neighbours will influence an agents behaviour more than slower ones. This allows us to model the idea of neighbours which are moving faster must have more information about danger or food than the individual. So the individual is more likely to follow them than neighbours which are moving slower. The formula for the

Modification	Weight function
1	$w_{ij}^{(t)} = \begin{cases} 0, & r_{ij}^{(t)} - r \geq 0, \\ 1, & r_{ij}^{(t)} - r < 0, \end{cases}$
2	$w_{ij}^{(t)} = \exp\left\{-\frac{r_{ij}^2}{\sigma^2}\right\}$
3	$w_{ij}^{(t)} = \exp\left\{-\frac{r_{ij}^2}{\left(v_j^{(t)}+1\right)^{\alpha}\sigma^2}\right\}$
4	$w_{ij}^{(t)} = \exp\left\{-\frac{r_{ij}^2}{\left(v_j^{(t)}+1\right)^{\alpha}\frac{\sigma^2}{\left(v_i^{(t)}+1\right)^{\beta}}}\right\}$

Table 4.1: The weight functions for the four Vicsek model modifications.

weight is

$$w_{ij}^{(t)} = \exp\left\{-\frac{r_{ij}^2}{\left(v_j^{(t)}+1\right)^{\alpha}\sigma^2}\right\}. \tag{4.7}$$

The parameter α allows us to vary the strength of the importance of the speed of the neighbour. When $\alpha = 0$ we return back to the case we have in Section 4.2.2, where speed does not play a role in the weights of neighbours. By incorporating the neighbours speed in this way when $\alpha > 0$ it is equivalent to making a faster agent appear closer that it would otherwise. Similarly when $\alpha < 0$ this is equivalent to making a faster agent appear further away, and therefore it would have a smaller weight in the averages. There are no constraints on the value of α

4.2.4 Modification 4: Interactions Dependent On All Speeds

Once the speed of the neighbours has been included in the weights it is only natural for the speed of the agent itself to also be included. It is logical to assume that when one is moving faster less of the surroundings get accounted for. Hence the formulation of the weights is now

$$w_{ij}^{(t)} = \exp\left\{-\frac{r_{ij}^2}{\left(v_j^{(t)}+1\right)^{\alpha}\frac{\sigma^2}{\left(v_i^{(t)}+1\right)^{\beta}}}\right\}. \tag{4.8}$$

The parameter β plays a similar role to that of α, it controls the strength of the importance of an agents own speed in the weights of its neighbours.

4.3 Ginelli Model

In 2015, Ginelli et al. [56] published a paper called "Intermittent collective dynamics emerge from conflicting imperatives in sheep herds", which describes a model for simulating the movement of sheep herds. They noticed that in modelling collective animal movements the existence of individual-level behavioural shifts, which may trigger a transition at the collective level, is often neglected; for example, in many species of fish, the group alternated regularly between a schooling behaviour, in which fish tend to move in the same direction and are aligned, and an aggregate behaviour, in which they have low levels of alignment. This behavioural change is due to a sudden change in velocity of a small number of individuals that propagates throughout the entire group [56, 111].

The model [56] devised was based on a quantitive study on the collective behaviour of

large groups of sheep in which the movement of the sheep was not influenced by any external stimuli. In their study they performed five trials, each an hour long, where 100 sheep were enclosed in a flat 80m × 80m spatially homogeneous field. During each trial the movement of the sheep was recorded by photographing the field at a rate of 1 frame per second. The camera was fixed at the top of a 7m tower located outside the field at the corner. They noticed that while the sheep were grazing the herd would spread apart into small groups. This dispersion was then broken by an individual, typically at the group periphery, running towards the centre. This individual would then cause other sheep to start to run until stopping leaving a compact herd which then begins to graze. This behaviour of slow dispersal and fast clumping was the key phenomena that they tried to replicate mathematically in their models.

In the mathematical model the sheep are represented by point-like agents which move according to a set of rules similar to that of the Vicsek model. In this model the i^{th} agent at time t is described by its position, $\mathbf{x}_i^{(t)}$, its direction, $\theta_i^{(t)}$, and its behaviour state, $q_i^{(t)}$. The behavioural state can take only 3 values: 0 when the agent is stationary; 1 when the agent is grazing/walking; or 2 when the agent is running. The rules used to update the location and direction of the agent depend on which state the agent is currently in. When the agent is stationary, $q_i^{(t)} = 0$, the location remains constant. In this model we also enforce that the direction of motion of stationary agents do not change. For agents that are moving, i.e. $q_i^{(t)} > 0$, we update the location according to an Euler time step, Section A.2.1, and where

$$\mathbf{v}_i^{(t)} = v\left(q_i^{(t)}\right)\left(\cos\theta_i^{(t)}, \sin\theta_i^{(t)}\right) \tag{4.9}$$

is the velocity of the agent. The speed of the agents v depends on the agents behavioural state, $v(2) \gg v(1) > v(0) = 0$.

The walking agents follow dynamics similar to the Vicsek model has been discussed in this chapter, Section 4.2. The walking agents in this model aim to align with their neighbours closer than r while travelling at a fixed speed. The update equation for the direction of motion is therefore,

$$\theta_i^{(t)} = \arctan\left(\frac{\left\langle \sin\left(\theta^{(t)}_{j\in\mathcal{N}_i^{(t)}}\right)\right\rangle}{\left\langle \cos\left(\theta^{(t)}_{j\in\mathcal{N}_i^{(t)}}\right)\right\rangle}\right) + \xi \tag{4.10}$$

where ξ is a noise term which perturbs the angle by a random amount. In this model the noise is uniformly distributed in the range $[-\eta\pi, \eta\pi]$. The set $\mathcal{N}_i$ contains the indices of the neighbours of agent i which are within the radius r.

The dynamics of the running agents are much more complex. Instead of interacting with

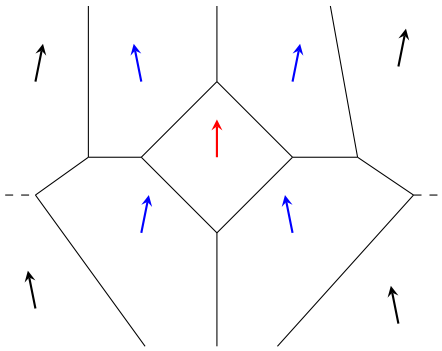


Figure 4.2: Set of first shell Voronoi neighbours of an agent, →, are shown as →. They are the agents which reside in the regions adjacent to the region containing →. The regions are calculated using a Voronoi tessellation, Section A.5, [112]. The → agents are not part of the set $\mathcal{V}_i$

neighbours within a fixed distance running agents only interact with the first shell of their Voronoi neighbours [112], Figure 4.2, using a Voronoi tessellation calculated using the procedure shown in Section A.5. The indices of these neighbours make up the set $\mathcal{V}_i^{(t)}$. For each of these neighbours we calculate the effect it has on agent i using

$$\mathbf{e}_{ij}^{(t)} = \delta_{2,q_j^{(t)}} \hat{\mathbf{v}}_j^{(t)} + \beta f\left(r_{ij}^{(t)}\right)\left(\mathbf{x}_j^{(t)} - \mathbf{x}_i^{(t)}\right), \tag{4.11}$$

where $\hat{\mathbf{v}}_i^{(t)}$ is the unit velocity vector, $\delta_{2,q_j^{(t)}}$ is the Kronecker delta and $f(r_{ij}^{(t)})$ is the attraction/repulsion force, which depends on the distance between neighbours, $r_{ij}^{(t)}$.

The Kronecker delta, $\delta_{i,j}$, is a function of two variables i and j. When the two variables are equal then it takes the value 1, otherwise it has the value 0. So when calculating the effect of agent j on agent i, $\mathbf{e}_{ij}^{(t)}$, if the behaviour of the neighbouring agent j is running, i.e. $q_j^{(t)} = 2$, then the term containing the Kronecker delta, $\delta_{2,q_j^{(t)}} \hat{\mathbf{v}}_j^{(t)}$, is nonzero and the velocity of the neighbour is accounted for in the calculation of $\mathbf{e}_{ij}^{(t)}$. However, when the neighbour is not running, $q_j^{(t)} \neq 2$ subscripts on the Kronecker delta are no longer the same so $\delta_{2,q_j^{(t)}} = 0$. The parameter β scales the strength of the attraction/repulsion force which is defined as

$$f\left(r_{ij}\right) = \min\left(1, \frac{r_{ij} - r_e}{r_e}\right), \tag{4.12}$$

where r_e is the equilibrium distance at which agent i is neither attracted to nor repulsed by its neighbour. If you wanted the attraction/repulsion force to only be enacted up to a finite difference you could use

$$f\left(r_{ij}\right) = \begin{cases} \min\left(1, \frac{r_{ij} - r_e}{r_e}\right) & r_{ij} < r_c, \\ 0 & r_{ij} \geq r_c. \end{cases}$$

Once the effect of agent j on agent i has been calculated for all $j \in \mathcal{V}_i^{(t)}$ we can find the total effect on agent i by finding the sum of these individual effects

$$\mathbf{E}_i^{(t)} = \left(E_{i,x}^{(t)}, E_{i,y}^{(t)}\right) = \sum_{j \in \mathcal{V}_i^{(t)}} \mathbf{e}_{ij}^{(t)}. \tag{4.13}$$

The new direction of agent i is simply the direction this effect is pointing in. So therefore we have

$$\theta_i^{(t+1)} = \arctan\left(\frac{E_{i,y}^{(t)}}{E_{i,x}^{(t)}}\right), \tag{4.14}$$

where $E_{i,x}$ and $E_{i,y}$ are the x and y components of the total force respectively.

Using an Euler time step and Equation (4.9) we can update the location and using Equations (4.11)–(4.14) the direction of the agents over time using the equations above. Now we will describe the equations of how to transition between behaviour states. Transitions between behavioural states are stochastic, but the probability of the change occurring depends on a number of variables. The probability rate of a transition from stationary to walking for agent i at time t takes the form

$$p_{0,1}(i)^{(t)} = \frac{1 + \alpha N\left(\mathcal{N}_i^{(t)}, 1\right)}{\tau_{0,1}} \tag{4.15}$$

where the function $N\left(\mathcal{N}_i^{(t)}, q\right)$ calculates the number of neighbours in the set $\mathcal{N}_i^{(t)}$ which have behaviour q. As laid out in the schematic in Figure 4.1, the set $\mathcal{N}_i^{(t)}$ is set of indices of the agents within a distance r from agent i at time t. The parameter $\tau_{0,1}$ is the transition time, and α is the strength of the mimetic effect.

The function which calculates the rate of going from walking to stationary has a similar form to Equation (4.15)

$$p_{1,0}(i)^{(t)} = \frac{1 + \alpha N\left(\mathcal{N}_i^{(t)}, 0\right)}{\tau_{1,0}} \tag{4.16}$$

where $\tau_{1,0}$ is the transition time from walking behaviour to still behaviour.

Still and walking sheep transition to running with the same rate function

$$p_{q,2}(i)^{(t)} = \frac{1}{\tau_{q,2}}\left[\frac{l_i^{(t)}}{d_R}\left(1 + \alpha N\left(\mathcal{V}_i^{(t)}, 2\right)\right)\right]^{\delta} \tag{4.17}$$

where q here is either 0 or 1, $l_i^{(t)}$ is the average distance to all neighbours in $\mathcal{V}_i^{(t)}$, d_R is a characteristic length scale and δ controls the allelomimetic effect. The size of characteristic

lengthscale d_R determines at which density the clustering event occurs. When $\frac{l_i^{(t)}}{d_R} > 1$ then the separation between the sheep is too high and the chance of a clustering event occurring increases. Similarly when $\frac{l_i^{(t)}}{d_R} < 1$ then the separation between the sheep is small so the chance of a clustering event occurring decreases.

For simplicity the behaviour of running sheep can only transition to stationary. So the final rate is

$$p_{2,0}(i)^{(t)} = \frac{1}{\tau_{2,0}} \left[\frac{d_S}{l_i^{(t)}} (1 + \alpha N(\tilde{\mathcal{V}}_i^{(t)}, 0)) \right]^{\delta}, \tag{4.18}$$

where d_S is a second characteristic length scale with the property $d_S < d_R$. The set $\tilde{\mathcal{V}}_i^{(t)}$ is the set of first shell Voronoi neighbours which stopped between time $t-1$ and time t. Including the term $\tilde{\mathcal{V}}_i^{(t)}$ leads to a positive feedback of agents stopping, which causes the herd to come to a sudden halt. The role of the average distance to the nearest neighbours here plays the opposite role to that in Equation (4.17). The more dense the group becomes, the more likely it is that the agent will transition to stationary.

We convert the rate of each of the transitions into a probability of transitioning from behaviour r to behaviour s by using

$$P_{r,s} = 1 - e^{-p_{r,s} dt} \tag{4.19}$$

where $p_{r,s}$ is the probability rate of switching from behaviour r to behaviour s, and $r \neq s$.

Parameter	Value	Units	Description
N	100.00	–	Number of agents
dt	1.00	s	Size of the time step in the Euler method
$v(1)$	0.15	ms^{-1}	Speed of walking agents
$v(2)$	1.50	ms^{-1}	Speed of running agents
$\tau_{1,0}$	8.00	s	Transition time from 1 to 0
$\tau_{0,1}$	35.00	s	Transition time from 0 to 1
$\tau_{0,2}$	N	s	Transition time from 0 to 2
$\tau_{1,2}$	N	s	Transition time from 1 to 2
$\tau_{2,0}$	N	s	Transition time from 2 to 0
d_S	6.30	m	Characteristic length scale to enable stopping
d_R	31.60	m	Characteristic length scale to enable running
r_e	1.00	m	Equilibrium distance for attraction/repulsion force
r	1.00	m	Radius of interaction for walking agents
β	0.80	–	Strength of attraction/repulsion force
η	0.13	–	Strength of the noise term

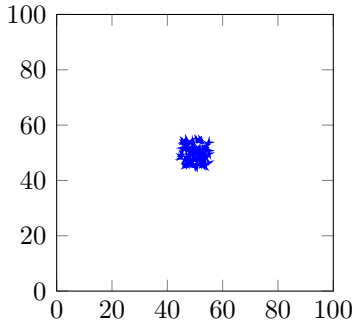
Table 4.2: Parameters used by Ginelli et al. [56] to recreate their experimental data.

We can now update all aspects that define each agent.

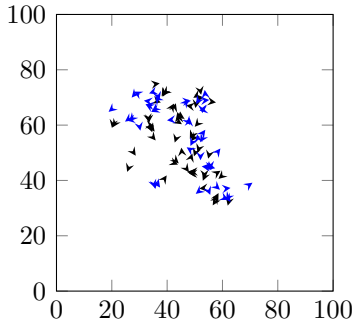
4.3.1 Model Simulations

Ginelli et al. [56] found that many parameters of the model can be deduced from obervering their experimental data, and by order of magnitude considerations [56], Table 4.2. This leaves the models with only two unknowns, the allelomimetic parameters α and δ. They found in order to have the slow spreading followed by fast paced clustering, which they had seen in their experiments, the allelomimetic parameters had lower bounds, $\alpha > 5$ and $\delta > 2$. They found that values of $\alpha = 15$ and $\delta = 4$ best described the experimental data.

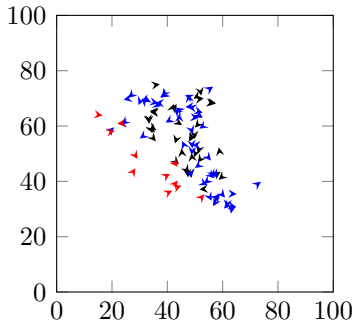
The agents in our simulations are initialised with random locations within a 50×50 box. The agents each start with a random direction between 0 and 2π and a behaviour state of $q_i^{(0)} = 1$ for all i (walking). Snapshots taken from one of these simulations are given in Figure 4.3.



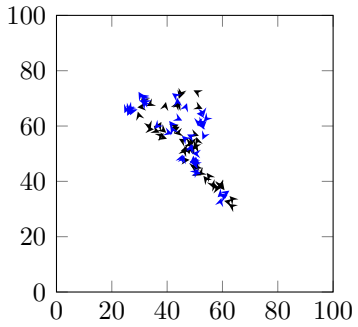
(a) $t = 0$: The flock start all with the walking behaviour.



(b) $t = 500$: The flock slowly disperses while individuals alternate between walking and staying still.



(c) $t = 521$: When the flock spreads out the probability of agents at the edge starting to run increases.



(d) $t = 586$: The flock is much denser after a clustering event.

Figure 4.3: Snapshots from a simulation of the model by Ginelli et al. [56] using their preferred parameters. The different colours show the different behaviours ➤ stationary, ➤ walking, ➤ running.

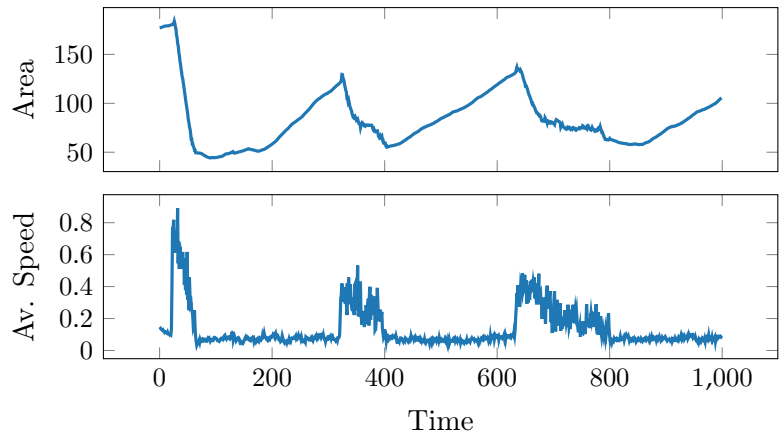


Figure 4.4: The area of the convex hull of locations of the agents from a single simulation of the Ginelli model.

Figure 4.5: The average speed of the sheep from a single simulation of the model by Ginelli et al. [56].

When looking at how the area covered by the flock changes over time we can see that the "clumping events" described by Ginelli et al. [56] can be identified by a sharp decrease in area, Figure 4.4. We can approximate the area of the flock by looking at the area of the convex hull of the locations of the agents [113]. The convex hull of a set of points defined as the smallest convex polygon which includes all of the points.

It is also possible, though harder, to see these same clumping events by looking at the mean speed over time, Figure 4.5. When the flock clumps together more of the sheep are in the running behaviour, therefore they have a mean speed closer to $v(2)$, which as described in Equation (4.9) is greater than that of the walking speed, $v(1)$.

It is not possible to compare this model to our collected data, as the sheep in this model are not being actively herded. The sheep being observed by Ginelli were left to graze in the field whereas the sheep we observed were being actively moved by the farmer [56]. Additionally when looking at the trajectories of the sheep simulated by the Ginelli model, Figure 4.6, we can see that the differences between the behaviour we observed and the behaviour shown by the Ginelli simulations are not compatible. We did look at what would

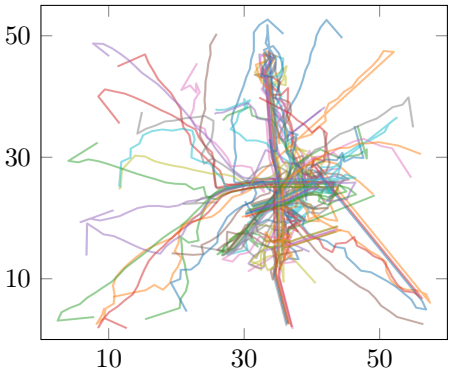


Figure 4.6: Trajectories of the sheep simulated by the Ginelli model after 200 time steps with the colour distinguishing between individuals.

happen if a single agent was initialised with fixed behaviour and direction to approach a flock of sheep simulated by the Ginelli model. However, this simply resulted in a clumping event in the simulation when the agent with fixed behaviour approached the rest of the flock, and then the Ginelli agents behaved as seen before without the agent with fixed behaviour.

4.4 Chen Model

In our collected data we have either a quadbike or a number of farmers interacting with the sheep, so therefore we wish to be able to compare our data to a model which can incorporate these. One way of doing this would to include the quadbike or farmer in the model as a predator.

There are many models which aim to reproduce the dynamics of predator – prey interactions, most of these model are too complex to study analytically [45] so Chen and Kolokolnikov [45] produced a minimal model which was amenable to mathematical analysis and captures essential features of predator – prey interactions. As with Ginelli model, each agent in this model is represented by a point-like particle which moves with respect to the location of the other agents. This is the case for both different types of agent: predator and prey.

In this model there are N prey agents and a single predator. Chen and Kolokolnikov [45] assume that the prey agents, positions at time t, $\mathbf{x}_i^{(t)}$, follow Newton's second law of motion. Implying that the force on each individual can be written as

$$\mathbf{F}_{i,\text{total}} = m\frac{\mathrm{d}^2\mathbf{x}_i^{(t)}}{\mathrm{d}t^2} + \mu\frac{\mathrm{d}\mathbf{x}_i^{(t)}}{\mathrm{d}t}, \tag{4.20}$$

where m is the mass of the prey agent and μ is the size of the frictional force acting upon it. Chen and Kolokolnikov [45] remove the acceleration term assuming that the mass of the prey agents, m, is negligible compared to that of the size of the frictional force, μ. The relation in Equation (4.20) is then rescaled so that $\mu = 1$ which simply leaves the model as

$$\frac{\mathrm{d}\mathbf{x}_i^{(t)}}{\mathrm{d}t} = \mathbf{F}_{i,\text{total}}, \tag{4.21}$$

where the total force can broken down into a sum of two parts: the force from other prey $\mathbf{F}_{i,\text{prey}}$; and the force coming from the predator $\mathbf{F}_{i,\text{pred}}$. This simplification reduces the second-order model to a first-order model, which aids with mathematical analysis. The

Parameter	Value	Description
N	100	Number of prey agents
a	1.0	Size of the attraction force between prey
b	0.2	Size of the repulsion force on the prey from the predator
p	3.0	Power law exponent for force on predator from prey

Table 4.3: Parameters fixed by Chen and Kolokolnikov [45].

force between prey is defined as

$$\mathbf{F}_{i,\text{prey}} = \frac{1}{N} \sum_{j=1, j\neq i}^{N} F\left(\left|\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\right|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\right), \tag{4.22}$$

where

$$F(r) = \frac{c}{r^2} - a.$$

The first term is a Newtonian-type short-range repulsion force, and the second term is a linear long-range attraction force with strength $a > 0$. Both of these forces act in the direction from agent i to agent j.

The predator's position is denoted $\mathbf{p}^{(t)}$ and the model assumes that it acts as a repulsive particle on the prey agents. Therefore the force between the two types of agent can we written as

$$\mathbf{F}_{i,\text{pred}} = G\left(\left|\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}\right|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}\right) \tag{4.23}$$

where

$$G(r) = \frac{b}{r^2}.$$

The parameter b is controls the strength of this repulsive force. We now have all elements needed to calculate the velocity of the individual prey agents in this model. The velocity of the predator agent follows the same logic, but since there is only one predator agent in this model there is only have one term, the predator–prey interactions. The force acting on the predator is the average over all predator–prey forces, where each individual interaction is a power law that decays at large distances. The predator moves in the direction of that force resulting in the following update equation

$$\frac{\mathrm{d}\mathbf{p}^{(t)}}{\mathrm{d}t} = \frac{c}{N} \sum_{i=1}^{N} \frac{\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}}{\left|\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}\right|^p} \tag{4.24}$$

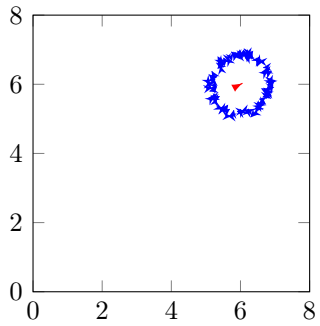
where the c determines the strength of the force from the prey agents on the predator.

4.4.1 Model Simulations

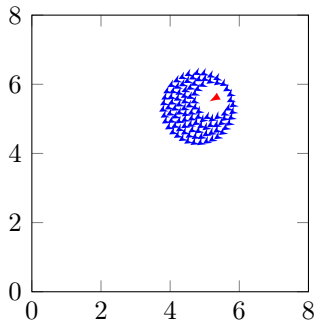
Chen and Kolokolnikov [45] fix all parameter values apart from c, which controls the strength of the response of the predator. They used c as a control parameter and varied from $c = 0.15$ up to $c = 2.5$. See Table 4.3 for the values of the parameters fixed by Chen and Kolokolnikov [45]. For both types of agent we use the fourth order Runge-Kutta method, Section A.2.5, to update the location of the agents where we fix the time stepping size to $dt = 1$.

Slow Predator Responses

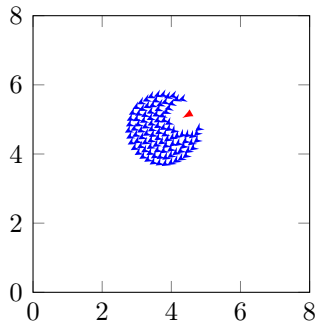
The prey are initialised with random locations within a torus shape. To initialise the locations in this fashion each of the locations is originally described using polar coordinates, (r, θ). Each agent draws its r coordinate from a $\mathcal{U}(0.3, 0.7)$ and its θ coordinate from a $\mathcal{U}(0, 2\pi)$. These polar coordinates are then transformed into Cartesian coordinates leaving us with the agents initialised in a torus where the radius of the hole in the centre



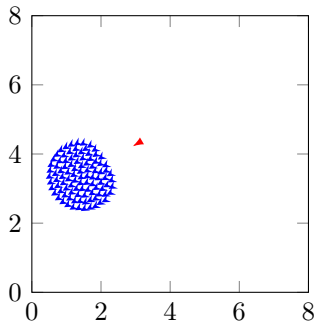
(a) $t = 0$: The flock start in a torus shape, with the predator in the centre.



(b) $t = 50$: The flock is aligned and moves away from the predator.



(c) $t = 60$: The flock eventually manages to expel the predator from within the group.



(d) $t = 80$: The flock is now free of the predator which has been left behind.

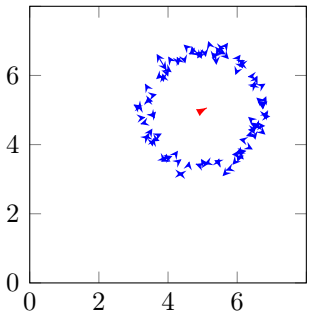
Figure 4.7: Reproduced snapshots from a simulation of the model by Chen and Kolokolnikov [45] using their preferred parameters and $c = 0.15$. The different arrowheads show whether the agent is representing ▸ prey or ▶ predator.

of the torus is 0.3 and the width of the torus is 0.2. The shape allows the predator to be initialised in the centre without encountering a spike in the velocities of ether the prey agents or the predator agent. Since the velocities of all agents, prey and predator, depend on the locations we do not give the agents an initial velocity. The velocities are calculated using Equations (4.22) and (4.23) for the prey and Equation (4.24) for the predator.

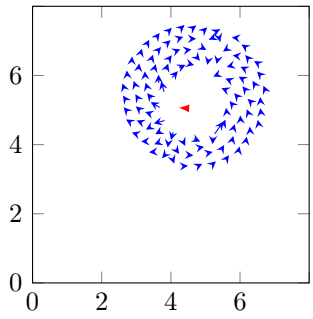
When the predator response variable is small, $c = 0.15$, the speed at which it can chase after the prey is insufficient for the predator to catch the prey. The initial location of the predator is in the centre of the prey; after a short period of time the predator is left trailing behind. This progression can be seen in Figure 4.7.

Fast Predator Responses

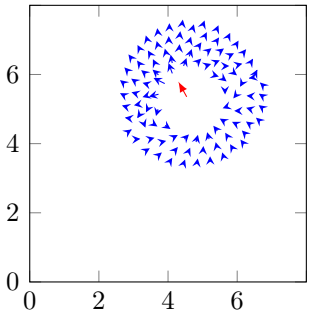
When simulating with a predator with fast responses, $c = 0.8$, we initialise the agents in the same way we did with the slow response rate. The prey agents are initialised with locations in a torus shape, with the predator located in the centre. By having the predator initialised in the centre of the flock, we see the typical behaviour of the agents earlier in



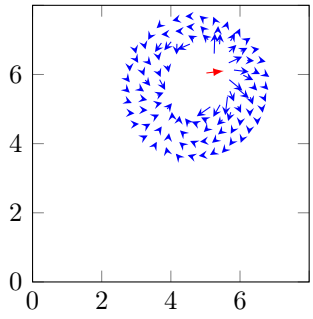
(a) $t = 0$: The flock start in a torus shape, with the predator in the centre.



(b) $t = 50$: The flock is aligned and tries to move away from the predator but cannot.



(c) $t = 60$: The predator changes direction to face the closest prey agents.



(d) $t = 80$: The flock then oscillates unable to get away from the predator.

Figure 4.8: Reproduced snapshots from a simulation of the model by Chen and Kolokolnikov [45] using their preferred parameters and $c = 0.8$. The different arrowheads show whether the agent is representing ▸ prey or ▶ predator.

the simulation then if the predator was slightly removed from the flock.

The predator now has sufficient speed to get in amongst the flock, and enough momentum to chase the prey in the inner ring of the torus, Figure 4.8. The forces acting on all the agents are no longer balanced, and the torus starts to oscillate. This movement is a culmination effect of the prey trying to move away from predator as one cohesive flock. However, this causes the prey agents at the rear of the flock to move closer to the predator. This then causes the flock moves in the opposite direction. This behaviour repeats itself until the end of the simulation.

With larger values of c we see a more exaggerated version of this behaviour. Since this model does not allow the predator to catch and hence remove the prey, we have seen all three different types of behaviour that this model can produce.

4.4.2 Acceleration Model

Chen and Kolokolnikov [45] also devised a second model which, rather than calculating the new velocity of agents, calculates the new acceleration instead. The derivation of the model follows the same steps as the previous model, but they no longer assume that the mass of the agents is negligible compared to the coefficient of friction. The equation for calculating the acceleration of the prey agents and for the predator are as follows:

$$m_i \frac{\mathrm{d}^2\mathbf{x}_i^{(t)}}{\mathrm{d}t^2} = -\mu_i \frac{\mathrm{d}\mathbf{x}_i^{(t)}}{\mathrm{d}t} + \frac{1}{N}\sum_{i=1, i\neq j}^{N} F\left(|\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\right) + G\left(|\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}\right), \tag{4.25}$$

$$m_0 \frac{\mathrm{d}^2\mathbf{p}^{(t)}}{\mathrm{d}t^2} = -\mu_0 \frac{\mathrm{d}\mathbf{p}^{(t)}}{\mathrm{d}t} + \frac{q}{N}\sum_{i=1}^{N} H\left(|\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}\right), \tag{4.26}$$

where the variable q controls how quickly the predator agent can react to the prey agents and where we have

$$F(r) = \frac{c}{r^2} - a, \tag{4.27}$$

$$G(r) = \frac{b}{r^2}, \tag{4.28}$$

$$H(r) = \frac{1}{r^s + r}, \tag{4.29}$$

with the variables a, b and c are the parameters which control the behaviour of the prey agents. The variable s determines the strength of the attraction from towards the prey. The function F determines how the prey agents interact with each other whist the functions

Parameter	Value	Description
N	500	Number of prey agents
$m_i = m_0$	1	Mass of prey agents and predator respectively
$\mu_i = \mu_0$	1	Coefficient of friction of prey and predator respectively
a	1	Size of the attraction force between prey
b	1	Size of the repulsion force on the prey from the predator
c	1	Size of repulsion between prey agents
q	1	Reaction speed for the predator
s	6	Size of attraction towards the prey for the predator

Table 4.4: Parameters fixed by Chen and Kolokolnikov [45] for their acceleration model [45].

G and H determine how the prey and predator interact. All three of these functions are based on the distance between interacting agents. The parameters for this model are shown in Table 4.4.

4.5 Helbing Model

In all the models we have seen so far the agents have been unconfined. However, in our data collection, Chapter 2, and in the observations by Ginelli et al. [56] flocks were confined to a field, so we require a model which takes into account the interactions agents have with walls. In 2000 Helbing et al. [114] published a model to approximate the movement of agents moving through a narrow exit. Similarly to the Chen and Kolokolnikov [45] model, they started of with a generalised force model, using Newton's second law of motion

$$m\frac{\mathrm{d}^2\mathbf{x}_i^{(t)}}{\mathrm{d}t^2} = \mathbf{F}_{i,\text{total}}. \tag{4.30}$$

Unlike the model by Chen and Kolokolnikov [45] this model does not reduce to first order ordinary differential equations. This total force, $\mathbf{F}_{i,\text{total}}$ can be broken down into three different types of forces: a driving force $\mathbf{F}_{i,\text{driving}}$, the force between agents $\mathbf{F}_{i,j}$, and the force between an agent and a wall $\mathbf{F}_{i,w_k}$, where w_k denotes the k^{th} wall. The driving force is simply the force which acts on the agent because they each have a desired speed, $v_i^{(t)}$, and a desired direction, $\mathbf{e}_i^{(t)}$. Both the desired speed and direction can be set to change over time. It can be calculated by

$$\mathbf{F}_{i,\text{driving}} = \frac{m_i}{\tau_i}\left(v_i^{(t)}\mathbf{e}_i^{(t)} - \mathbf{v}_i^{(t)}\right), \tag{4.31}$$

where m_i is the mass of agent i and τ_i is the characteristic time in which agent i adapts its velocity.

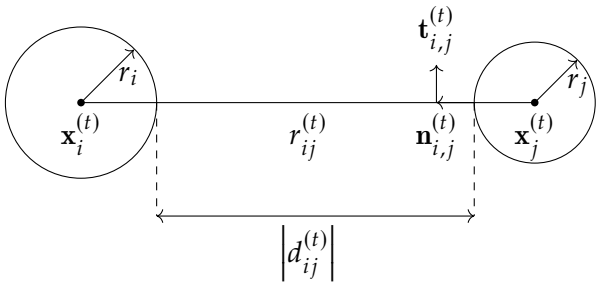


Figure 4.9: Schematic showing variables used when calculating the force acting on an agent coming from other agents, $\mathbf{F}_{i,j}$.

Helbing et al. [114] define the force between agents as the combination of a repulsive force, a body force, and a friction force. This model includes a frictional force between agents because in this model the agents have physical size, and therefore they can rub against each other. It is this force cause them to slow down when that happens. This gives us

$$\mathbf{F}_{i,j} = A_i \exp\left\{\frac{d_{ij}^{(t)}}{B_i}\right\}\mathbf{n}_{i,j} + K_1 g(d_{ij}^{(t)})\mathbf{n}_{i,j} + K_2 g(d_{ij}^{(t)})\left[\left(\dot{\mathbf{x}}_j^{(t)} - \dot{\mathbf{x}}_i^{(t)}\right)\cdot t_{i,j}\right] t_{i,j}, \tag{4.32}$$

where $\dot{\mathbf{x}}_i^{(t)}$ is the velocity of agent i at time t, r_{ij} is the distance between the centre of mass of the agents at time t and $d_{ij}^{(t)} = r_i + r_j - r_{ij}$ is the distance between the agents given physical size. If d_{ij} is positive then this represents the amount the agents overlap and if it is negative then the size of this represents the distance between the agents but taking into account their physical size, Figure 4.9. The function $g(d)$ is defined as

$$g(d) = \begin{cases} 0 & \text{for } d < 0 \\ d & \text{for } d \geq 0 \end{cases} \tag{4.33}$$

The vector $\mathbf{n}_{i,j}^{(t)}$ is the normal vector pointing from agent j to agent i at time t. It is be calculated by

$$\mathbf{n}_{i,j}^{(t)} = \frac{\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}}{r_{ij}} \tag{4.34}$$

and finally $\mathbf{t}_{i,j}^{(t)}$ is tangental to this normal vector. So if we have normal vector with components $\mathbf{n}_{i,j}^{(t)} = \left(\mathbf{n}_{i,j}^{(t)}(x), \mathbf{n}_{i,j}^{(t)}(y)\right)$ the tangental normal is $\mathbf{t}_{i,j}^{(t)} = \left(-\mathbf{n}_{i,j}^{(t)}(y), \mathbf{n}_{i,j}^{(t)}(x)\right)$. All the variables mentioned here in the formulation of $\mathbf{F}_{i,j}$ can be seen in the schematic in Figure 4.9. Since these forces include a crushing force, $K_1 g(d_{ij}^{(t)})\mathbf{n}_{i,j}$, if agents are too close to one another or a wall, the agents in this model have physical size, they are no longer point like agents which were defined in the Chen model [45].

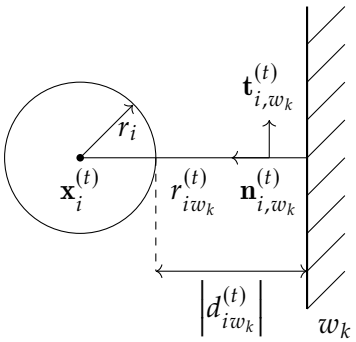


Figure 4.10: Schematic showing variables used when calculating the force acting on an agent coming from an individual wall, $\mathbf{F}_{i,w_k}$.

The interactions with the wall is analogous: $r_{i,w_k}^{(t)}$ is the distance between agent i and wall k at time t taking into account the size of the agent, $\mathbf{n}_{i,w_k}^{(t)}$ is the normal perpendicular to it, and $\mathbf{t}_{i,w_k}^{(t)}$ is the vector tangental to it. Therefore the corresponding force from wall k to agent i is

$$\mathbf{F}_{i,w_k} = A_i \exp\left\{\frac{d_{iw_k}^{(t)}}{B_i}\right\}\mathbf{n}_{i,w_k} + K_1 g(d_{iw_k}^{(t)})\mathbf{n}_{i,w_k} + K_2 g(d_{iw_k}^{(t)})\left[\dot{\mathbf{x}}_i^{(t)} \cdot \mathbf{t}_{i,w_k}^{(t)}\right]\mathbf{t}_{i,w_k}^{(t)}. \tag{4.35}$$

We now have defined all three different types of force therefore the Helbing model is

$$m_i \frac{\mathrm{d}^2\mathbf{x}_i^{(t)}}{\mathrm{d}t^2} = \mathbf{F}_{i,\text{driving}} + \sum_{j=0,j\neq i}^{N} \mathbf{F}_{i,j} + \sum_{k=0}^{K} \mathbf{F}_{i,w_k}. \tag{4.36}$$

where N is the number of agents in the model, and K is the number of walls in the model.

4.5.1 Defining The Walls

In the simulation shown in Figure 4.11 we have four walls. Each wall is defined by the coordinates of its start and end point. From these coordinates we can calculate the minimum distance between an agent and the wall, $|d_{iw_k}|$, and also the normal vector for the line at the point where the distance is minimal, $\mathbf{n}_{iw_k}$.

To be able to find both of these we need to be able to define the equation of line defined by the start and end point. If the start point of our wall is denoted $P_1 = (x_1, y_1)$ and the end point $P_2 = (x_2, y_2)$, then the equation for a line passing through these two points is

$$(y_2 - y_1)x - (x_2 - x_1)y + y_2(x_2 - x_1) = 0.$$

Since we know the equation for the line in the form $ax + by + c = 0$ the normal vector $\mathbf{n}_{iw_k} = (a, b)$.

In some situations the length of the wall might not be infinitely long. Therefore to calculate the distance between the wall and the agent we first need to calculate u where

$$u = \frac{(x_0 - x_1)(x_2 - x_1) + (y_3 - y_1)(y_2 - y_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

When the agents lies in line with the wall then $0 < u < 1$ and the point of intersection between the agents and the wall is at the point $P_{Int} = (x_{Int}, y_{Int}) = (x_1 + u(x_2 - x_1), y_1 + u(y_2 - y_1))$ but if $u < 0$ the point of intersection is $P_{Int} = P_1$ and if $u > 1$ the point of intersection is $P_{Int} = P_2$. The distance between the wall and the agent then can be calculated:

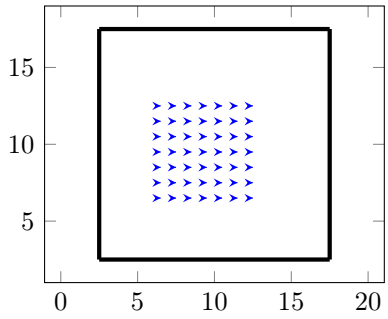
$$|d_{iw_k}| = \sqrt{(x_0 - x_{Int})^2 + (y_0 - y_{Int}^2)}. \tag{4.37}$$

4.5.2 Model Simulations

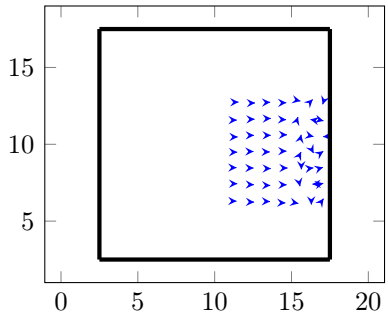
In the paper by Helbing et al. [114] they simulated $N = 200$ agents in a 15m×15m room with identical desired speed, $v_i^{(d)}$, and where the desired direction, $\mathbf{e}_i^{(t)}$, was the unit vector towards a 1m wide exit in one of the walls. In our simulations of their model we set $N = 49$ and set their desired direction to be the vector $\mathbf{e}_i^{(d)} = (1, 0)$. Since we are not interested in the behaviour of the agents as they leave the room, we remove the exit. In both our simulations and those by Helbing et al. [114] the radius (in meters) of the agents comes from uniform distribution, $r \sim \mathcal{U}(0.25, 0.35)$. Allowing the radius' to vary across agents reduces the chances of model artefacts appearing. The remaining parameters of the model remain unchanged from the values used by Helbing et al. [114] in their simulations,

Parameter	Value	Units	Description
m_i	80.00	kg	Mass of agents
$v_i^{(0)}$	0.80	m s^{-1}	Desired speed of agents
A_i	2,000.00	N	The strength of the wall
B_i	0.08	m	The size of the wall
τ_i	0.50	s	Characteristic time
K_1	120,000.00	kg s^{-1}	The size of body compression force
K_2	240,000.00	kg m^{-1} s^{-1}	The size of friction force

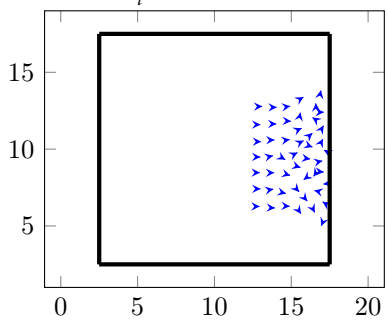
Table 4.5: Parameters fixed by Helbing et al. [114] in their model simulations. Due to the lack of experimental data available owing to the fact that escape panics are rare and unexpected and the ethic implications of manufacturing these behaviours Helbing et al. [114] used information from Weidmann [115] to estimate their parameter values.



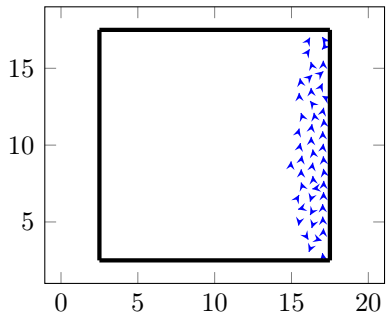
(a) $t = 0$: The agents start in a lattice, with desired direction $\mathbf{e}_i^{(d)} = (1, 0)$.



(b) $t = 7$: When they reach the wall the lattice starts to breakdown.



(c) $t = 9$: The agents spread out along the wall face.



(d) $t = 30$: After some time the agents fill the space across the wall.

Figure 4.11: Reproduced snapshots from a simulation of the model by Helbing et al. [114] using their preferred parameters where ➤ represent the agents.

Table 4.5.

Snapshots from our simulations of the Helbing model can be seen in Figure 4.11. In these snapshots we observe that the agents start of moving in their desired direction until they get to the wall. Once at the wall the forces tangental to it makes them to spread out and the lattice structure they originally had breaks down. After a while the agents then fill the space across the entirety of the wall and the forces balance causing the agents to come to rest.

4.6 Combined Sheep Model (CSM)

We combined two models from the literature in order to create a combined model: a predator–prey model by Chen and Kolokolnikov [45] and a pedestrian model including walls by Helbing et al. [114]. A similar study was done by Miller and Ouellette [105] where they simulated agents following Vicsek dynamics hitting an impermeable wall. In their research they looked into the surface tension of the agents. The model by Chen and Kolokolnikov [45] provided the base for our combined sheep model, as it was a minimal

model containing interactions between a herd of animals and a predator. The equation to calculate the acceleration of both types of agents, as seen in the previous Section 4.4.2, is as follows

$$m_i \frac{\mathrm{d}^2\mathbf{x}_i^{(t)}}{\mathrm{d}t^2} = -\mu_i \frac{\mathrm{d}\mathbf{x}_i^{(t)}}{\mathrm{d}t} + \frac{1}{N}\sum_{i=1,i\neq j}^{N} F\left(|\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\right) + G\left(|\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}\right), \tag{4.38}$$

$$m_0 \frac{\mathrm{d}^2\mathbf{p}^{(t)}}{\mathrm{d}t^2} = -\mu_0 \frac{\mathrm{d}\mathbf{p}^{(t)}}{\mathrm{d}t} + \frac{q}{N}\sum_{i=1}^{N} H\left(|\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}\right). \tag{4.39}$$

where the variable q controls how quickly the predator agent can react to the prey agents and where we have

$$F(r) = \frac{c}{r^2} - a,$$
$$G(r) = \frac{b}{r^2},$$
$$H(r) = \frac{1}{r^s + r},$$

The Helbing model restricts agents to interact within a confined area and also originates from Newton's second law. Helbing et al. [114] define the change in velocity to be

$$m_i \frac{\mathrm{d}^2\mathbf{x}_i^{(t)}}{\mathrm{d}t^2} = m_i \frac{v_{d,i}^{(t)}\mathbf{e}_{d,i}^{(t)} - \dot{\mathbf{x}}_i^{(t)}}{\tau_i} + \sum_{j\neq i}\mathbf{f}_{ij} + \sum_{k=0}^{K}\mathbf{f}_{i,w_k} \tag{4.40}$$

where the first term of the right hand side of the equation is simply the finite difference between the velocity the agent is moving at and the speed and direction it wishes to travel in. The desired speed and direction are denoted $v_{d,i}^{(t)}$ and $\mathbf{e}_{d,i}^{(t)}$. The second term, $\sum_{j\neq i}\mathbf{f}_{i,j}$, is an internal force between agents. The force is a combination of attraction and repulsion forces, but also represents agents getting crushed if they get too close together. The final term, $\sum_{k=0}^{K}\mathbf{f}_{i,w_k}$ is a similar combination of attraction and repulsion forces, but instead these forces being between agents they are between an agent and the walls. The agents in this model have physical size, they are no longer point like agents which were defined in the models by Chen and Kolokolnikov [45] and Vicsek et al. [38].

In order to join these two models together we replace the first two terms in Equation (4.40) with Equation (4.38) as the acceleration calculated in Equation (4.38) determines the agents desired direction and speed due to the forces acting on it by is surroundings. This

combination gives

$$\frac{\mathrm{d}^2\mathbf{x}_i^{(t)}}{\mathrm{d}t^2} = -\mu_i \frac{\mathrm{d}\mathbf{x}_i^{(t)}}{\mathrm{d}t} + \frac{1}{N}\sum_{i=1,i\neq j}^{N} F\left(|\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\right) + G\left(|\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}\right) + \frac{1}{m_i}\sum_{k=0}^{K}\mathbf{f}_{i,w_k}, \tag{4.41}$$

$$\frac{\mathrm{d}^2\mathbf{p}^{(t)}}{\mathrm{d}t^2} = -\mu_p \frac{\mathrm{d}\mathbf{p}^{(t)}}{\mathrm{d}t} + \frac{q}{N}\sum_{i=1}^{N} H\left(|\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}\right) + \frac{1}{m_0}\sum_{k=0}^{K}\mathbf{f}_{i,w_k}. \tag{4.42}$$

We define our crushing forces to be a simpler version of those used by Helbing et al. [114] in their paper to reduce the time it takes to simulate from this model. We use

$$\mathbf{f}_{iw} = A_i \mathbf{n}_{iw} \exp\left\{\frac{d_{iw}}{B_i}\right\} \tag{4.43}$$

were A_i and B_i are constants that determine the size of the force felt by agent i.

When looking at the two models individually, Section 4.4.2 and Section 4.5, the location of the agents (both predator and prey) were updated using different time stepping methods with different requirements on the size of the time step. Therefore, we will be using a variable time stepping method so that the dt adapts to the complexity of the environment, Section A.2.4. This method will result in a larger step size being used when the agents are all in the centre of the domain, but as they start to approach the walls the step size will decrease.

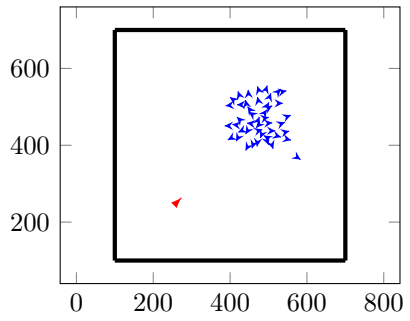
Parameter	Value	Description
N	45.0	Number of prey agents
$m_i = m_0$	60.0	Mass of prey agents and predator respectively
$\mu_i = \mu_p = \mu_0$	1.0	Coefficient of friction of prey and predator agents
a	1.0	Size of the attraction force between prey
b	1,000.0	Size of the repulsion force on the prey from the predator
c	12,000.0	Size of repulsion between prey agents
q	12,000.0	Reaction speed for the predator
s	2.4	Size of attraction towards the prey for the predator

Table 4.6: Parameters used to create Figure 4.12. These values results in average speed and distance to nearest neighbour which are comparable to collected data.

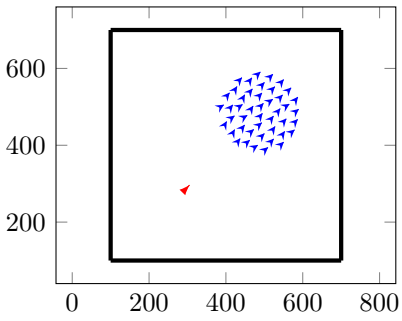
4.6.1 Model Simulations

An example of the type of behaviour the combined sheep model can produce can be seen in Figure 4.12. The simulation shown here was ran with 45 agents confined to a square with corners at $(100, 100), (100, 700), (700, 700)$ and $(700, 100)$. By doing a parameter search the model parameters m_i, mu_i, a, c were selected to result in speed similar to those observed in our observational data and the remaining parameters were set to obtain spacing between agents and between agents and the predator that was comparable to the collected data, Table 4.6.

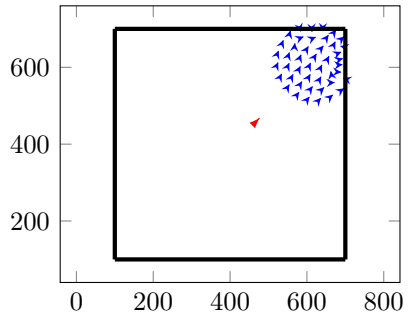
We can see in these snapshots that the prey agents are initially disordered, Figure 4.12a, but quickly they find order and start to move away from the predator agent, Figure 4.12b. Due to the initial location of the predator the agents are forced into the corner of the walls where they become crushed against it, Figure 4.12c. This then gives the predator a chance to catch up to the prey agents where it then becomes surrounded by the prey, Figure 4.12d. When the simulation was ran longer (snapshots not shown) one could see the prey then moving away from the predator and the walls and only to repeat the same behaviour again once it hit a different wall.



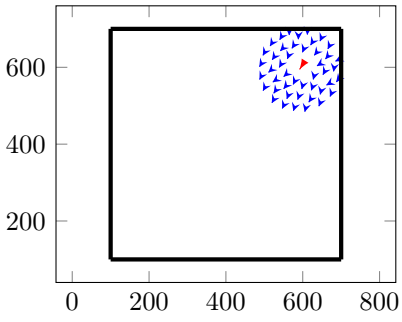
(a) t = 0: The flock starts in a disorganised group away from the predator.



(b) t = 10: The flock moves away from the predator.



(c) t = 55: The prey hits the walls and the predator starts to get closer.



(d) t = 90: The flock then surrounds predator causing the forces to be balanced.

Figure 4.12: Snapshots from a simulation of the combined sheep model using parameters shown in Table 4.6. The different arrowheads show whether the agent is representing ➤ prey or ▶ predator.

In the following two chapters we will compare our observational data described in Chapter 3 to data simulated from the Vicsek model modification in Chapter 5 and to simulated data from the combined sheep model in Chapter 6. These comparisons will produce a posterior distribution for the values of the parameters in each of the models.

5

Comparison Of Vicsek Models To Observational Data

5.1 Introduction

In Chapter 2 we described the process we used to obtain observational data which we wished to compare to the mathematical models discussed in the previous chapter. In this chapter we will compare a subset of that data to the Vicsek model modifications described in Section 4.2. The observational data used in this chapter will contain cases where collective behaviour is emergent. The Vicsek models presented here are not appropriate for comparison to state-state flocking as there is no term in any of the models that keep the flock together in one cohesive group. Examples of this can be seen in the trajectory plots at the end of this chapter, Figure 5.25. Simulated flocks may split into smaller groups and each of those will exhibit steady state behaviour. We cannot compare the behaviour of these sub-groups to our observational data s the number of agents in the group will be smaller than the number of sheep in our observations. This chapter will start by describing approximate Bayesian computation. It will then lay out the observational datasets used for comparing to the simulations. Finally we to aim to determine whether the family of Vicsek models can reproduce the behaviour we see in our observations.

5.2 Bayesian Inference

Bayesian inference is a statistical inference method where Bayes' theorem is used to update the probability of an event occurring as more information becomes available. Bayes' theorem can be written as

$$\pi(\phi|\mathbf{D}) = \frac{\pi(\phi)\pi(\mathbf{D}|\phi)}{\pi(\mathbf{D})} \tag{5.1}$$

where ϕ denotes the parameters of the model whose probability is affected by data, denoted by $\mathbf{D}$. The distribution $\pi(\phi)$, is known as the prior probability and is the probability of

the quantity $\boldsymbol{\phi}$ before the current data has been observed. The probability of $\boldsymbol{\phi}$ once the data has been observed is denoted $\pi(\boldsymbol{\phi}|\mathbf{D})$, and is known as the posterior probability. Therefore, $\pi(\mathbf{D})$ is the probability of observing the data. The final term in Bayes' equation $\pi(\mathbf{D}|\boldsymbol{\phi})$ is the probability of the data given the parameters and is called the likelihood. Using the same notation the term $\pi(\mathbf{D})$ is the probability of observing the data. This is often difficult to compute, so Bayes' theorem is often written as

$$\pi(\boldsymbol{\phi}|\mathbf{D}) \propto \pi(\boldsymbol{\phi})\pi(\mathbf{D}|\boldsymbol{\phi}) \tag{5.2}$$

i.e. the posterior is proportional to the prior times the likelihood [116]. The probability of observing the data is constant with respect to the parameters chosen. It can therefore absorbed into the constant of proportionality.

In Bayesian inference we want to infer the parameter(s), $\boldsymbol{\phi}$, of the mathematical model for our observed data. In other words what range of parameter(s) would we need in the model to best recreate the observed data using our mathematical model.

5.3 Approximate Bayesian Computation

Approximate Bayesian computation (ABC) is used to approximate the posterior distribution. ABC is a powerful and flexible tool and has been used in many fields, such as genetics [117], evolution [118] and agent based models [119]. The technique is used to determine the distribution of each of the model's parameters which best fit the observed data. This method is used in many statistical applications where it is difficult or not possible to calculate the likelihood, but it is easy to simulate from the model.

Consider a model $f(\mathbf{D}|\boldsymbol{\phi})$ where $\mathbf{D}$ is observed data and $\boldsymbol{\phi} = (\phi_1, \phi_2, \ldots, \phi_n)$ are the model parameters. Suppose we wish to sample from the posterior distribution, $\pi(\boldsymbol{\phi}|\mathbf{D})$, without evaluating the likelihood function. If model simulation is straightforward, then the following algorithm provides exact samples from the posterior

1. Sample $\boldsymbol{\phi}^*$ from the prior distribution $\pi(\boldsymbol{\phi})$.
2. Simulate a realisation, $\mathbf{x}$, from the model $f(\cdot|\boldsymbol{\phi}^*)$.
3. Accept $\boldsymbol{\phi}^*$ as a draw from the posterior distribution $\pi(\boldsymbol{\phi}|\mathbf{D})$ if $\mathbf{x} = \mathbf{D}$.

This rejection algorithm samples from the posterior distribution, $\pi(\boldsymbol{\phi}|\mathbf{D})$. This can be shown as follows. Let

$$I = \begin{cases} 1, & \mathbf{x} = \mathbf{D} \\ 0, & \mathbf{x} \neq \mathbf{D} \end{cases}$$

Then we have that $Pr(I = 1|\phi) = f(\mathbf{D}|\phi)$. So given that

$$\begin{aligned} Pr(I = 1) &= \int Pr(I = 1|\phi)\pi(\phi)\, \mathrm{d}\phi, \\ &= \int f(\mathbf{D}|\phi)\pi(\phi)\, \mathrm{d}\phi. \end{aligned}$$

Hence

$$\begin{aligned} \pi(\phi|I = 1) &= \frac{Pr(I = 1|\phi)\pi(\phi)}{\int Pr(I = 1|\phi)\pi(\phi)\, \mathrm{d}\phi}, \\ &= \frac{f(\mathbf{D}|\phi)\pi(\phi)}{\int f(\mathbf{D}|\phi)\pi(\phi)\, \mathrm{d}\phi}, \\ &= \pi(\phi|\mathbf{D}). \end{aligned}$$

When the observational data is discrete then this algorithm is exact and the rate at which proposed values, $\boldsymbol{\phi}^*$, are accepted would be non-zero. On the other hand, in most complex problems the rate at which $\boldsymbol{\phi}^*$ are accepted is unsuitably low. For example when using continuous data the rate of acceptance is typically zero as it is highly unlikely for the simulated data, $\mathbf{x}$, to be exactly the same as the observed data, $\mathbf{D}$.

This is where the "approximation" part of approximate Bayesian computation comes in. Instead of only accepting values of $\boldsymbol{\phi}^*$ when $\mathbf{x} = \mathbf{D}$, values are now only accepted provided that $\mathbf{x}$ is sufficiently close to $\mathbf{D}$. Hence the ABC rejection algorithm is now as follows

1. Draw $\boldsymbol{\phi}^*$ from the prior distribution $\pi(\cdot)$.
2. Simulate a realisation, $\mathbf{x}$, from the model $f(\cdot|\boldsymbol{\phi}^*)$.
3. Accept $\boldsymbol{\phi}^*$ as a draw from the posterior distribution if $\rho(\mathbf{x}, \mathbf{D}) \leq \varepsilon$,

where $\rho(\mathbf{x}, \mathbf{D})$ is some distance between simulated and observed data. The quantity ε is a small pre-defined tolerance level. For a suitable choice of ε, the ABC rejection scheme will approximate the true posterior distribution. However, if ε is too small then the rejection rates will be higher. A typical choice for $\rho(\mathbf{x}, \mathbf{D})$ is the Euclidean distance

$$\rho(\mathbf{x}, \mathbf{D}) = \sqrt{(x_1 - D_1)^2 + \cdots + (x_n - D_n)^2}$$

where n is the dimension of $\mathbf{x}$ (and hence $\mathbf{D}$).

The rejection rate increases with the dimension of the data, so in this case it is easier to compare summary statistics, $S(\mathbf{X})$. Summary statistics are used to summarise the high-dimensional data as simply as possible [118]. The common summary statistics are the

arithmetic mean, to measure the location of the observations, and the standard deviation, to measure the spread of the observations. In 2008 Joyce *et al.* proposed a method of choosing summary statistics systematically [120]. The criteria for parameter acceptance becomes:

$$\rho(S(\mathbf{x}), S(\mathbf{D})) \leq \varepsilon. \tag{5.3}$$

The efficiency increase obtained by using summary statistics does not introduce any error if the summary statistics are sufficient [121]. A sufficient statistic is defined as all information in $\mathbf{D}$ about ϕ is contained in $S(\mathbf{D})$. Finding a sufficient summary statistic can be difficult and with our data and models one could not be found. So to reduced bias and reduce the amount of information lost three summary statistics will be used and the average distance between observational data and simulated data will be used in the criteria for parameter acceptance, Equation (5.3).

5.4 Observational Data

The simulations from all four of the modified Vicsek models result in the average speed, alignment and integral lengthscale increasing over the course of the simulation, Figures 5.1, 5.2 and 5.3. These aspects are typical in behaviour seen in emergent collective

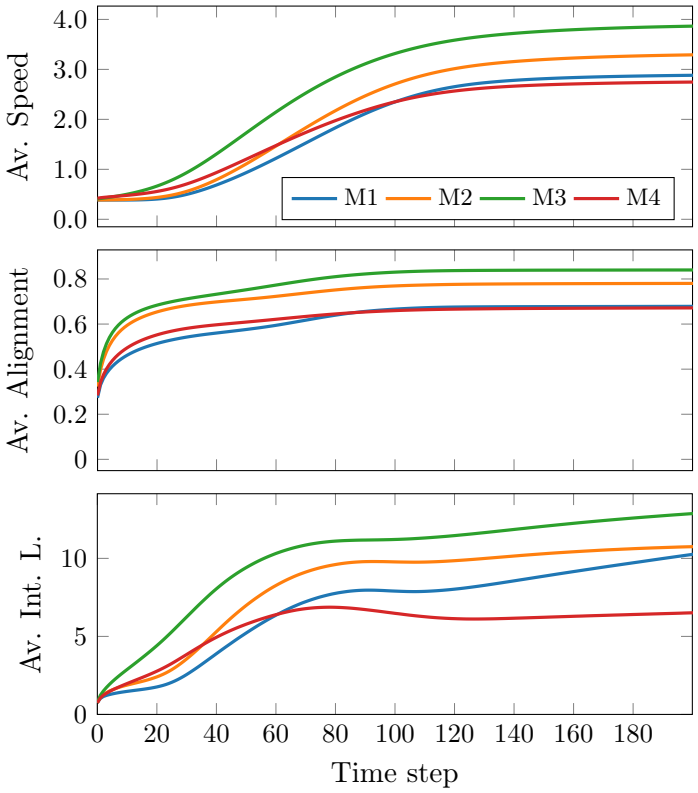


Figure 5.1: Average speed of the model modification simulations over time.

Figure 5.2: Average alignment of the model modification simulations over time.

Figure 5.3: Average integral lengthscale of the model modification simulations over time.

behaviour. Therefore we compare the simulated data to observational data which has similar features. The observational datasets we will use for comparison comes from Videos 1, 2, 3 and 4. The trajectory plots for these datasets can be seen Figures C.1, C.2, C.3 and C.4. We denote the observational datasets as $\mathbf{z}_{[i]}$ where $i \in [1, 2, 3, 4]$ is the video number. In the observational videos we see the emergence of collective behaviour in the flock of sheep in periods of time where the average speed, alignment and the integral lengthscale increase. Therefore by using the family of modified Vicsek models we aim to reproduce this emergent collective behaviour.

The three observational datasets have different lengths, Table 5.1, so we normalise the time period so that time starts at $\tau = 0$ and ends at $\tau = 1$. By doing this is it much easier to compare the videos to each other but also to the simulated data. Once time has been normalised we can look at each of the quantities mentioned above over τ. The normalisation factor for each of these observational datasets is simply the number of frames in the section of the video which we are going to compare $T_{[i]}$, shown in Table 5.1. Three of the observational datasets have a very similar number of frames which we want to compare to the simulations, $T_{[i]} \approx 120$ but Video 2 has almost twice as many, $T_{[2]} = 220$. Due to this it is important for us to normalise the time period using the method described above to make comparisons across videos easier.

The observational datasets were chosen because the average speed, alignment and the integral lengthscale increase over time. The average speed is calculated using

$$\langle |\dot{\mathbf{x}}^{(t)}| \rangle = \frac{1}{N} \sum_{j=1}^{N} |\dot{\mathbf{x}}_j^{(t)}| \quad \text{and} \quad \langle |\dot{\mathbf{z}}_{[i]}^{(t)}| \rangle = \frac{1}{N} \sum_{j=1}^{N} |\dot{\mathbf{z}}_{[i],j}^{(t)}|, \tag{5.4}$$

where $|\dot{\mathbf{x}}_j^{(t)}|$ is the speed of simulated agent j at time t, and similarly $|\dot{\mathbf{z}}_{[i],j}^{(t)}|$ is the speed of the j^{th} observed sheep at time t in observational dataset i. The average speed profiles for the observational datasets are shown in Figure 5.4.

The global alignment of the agents/sheep at time t, as described in Chapter 3, is defined

Obervational dataset $\mathbf{z}_{[i]}$	Length of observation Frames, $T_{[i]}$	Seconds	Number of sheep in oberservation
1	130	5.42	45
2	220	9.17	45
3	120	5.00	45
4	120	5.00	45

Table 5.1: The number of frames and time covered in each of the observational datasets, $\mathbf{z}_{[i]}$ where $i \in [1, 2, 3, 4]$.

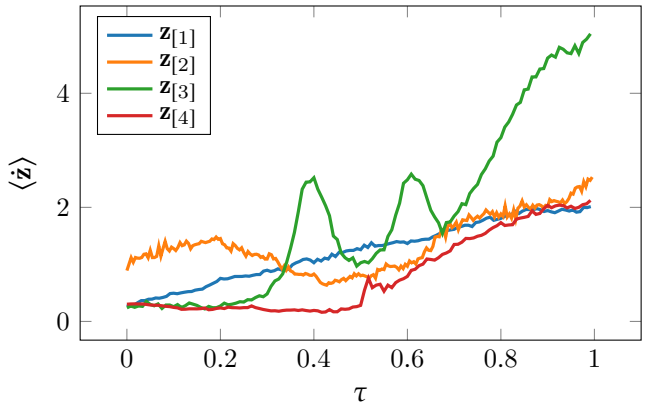


Figure 5.4: The average speed over rescaled time. Three of the observational datasets show that the average speed at the end of the simulation is higher than at the start of the observation. Whereas in $\mathbf{z}_{[3]}$ the profile of average speed has very distinct waves.

as

$$\psi^{(t)}(\mathbf{x}^{(t)}) = \frac{1}{N}\left|\sum_{j=1}^{N} \dot{\mathbf{x}}_j^{(t)}\right| \quad \text{and} \quad \psi^{(t)}(\mathbf{z}_{[i]}^{(t)}) = \frac{1}{N}\left|\sum_{j=1}^{N} \dot{\mathbf{z}}_{[i],j}^{(t)}\right|, \tag{5.5}$$

where $\dot{\mathbf{x}}_j^{(t)}$ is the velocity of agent j at time t, as defined in Section 3.6.4, and similarly $\dot{\mathbf{z}}_{[i],j}^{(t)}$ is the velocity of sheep j at time t in observational dataset i. We have the same number of agents in the simulations as the number of sheep in all four of the observational dataset so the i subscript on the N as seen in Chapter 3 has been dropped for conciseness. The global alignment over time for each the observational datasets is shown in Figure 5.5.

Finally the integral lengthscale, which gives an idea of the distance over which the velocities

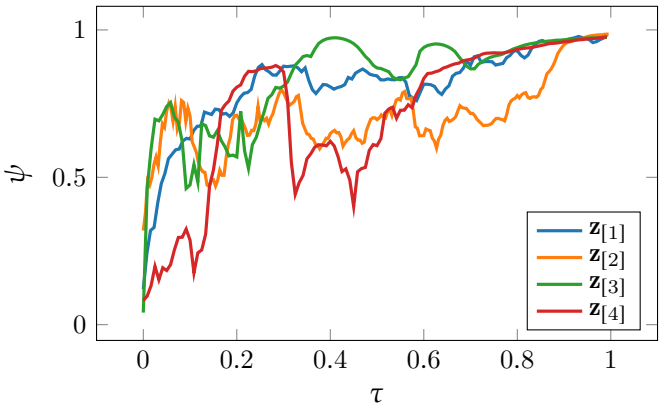


Figure 5.5: The global alignment over normalised time. All four observations exhibit similar increases in global alignment but with varying amounts of noise.

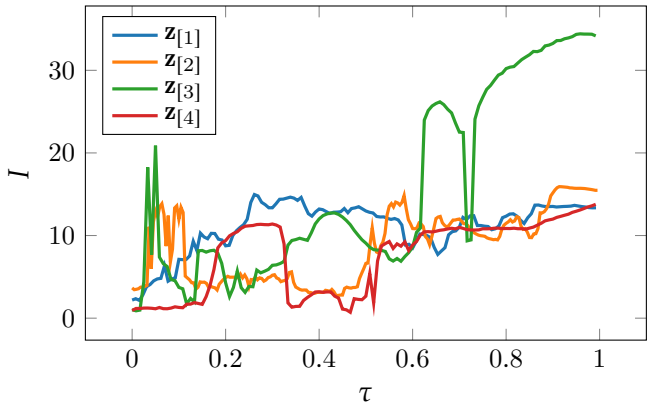


Figure 5.6: The integral lengthscale over normalised time for our four observational datasets. On average this quantity increases overtime, but it is the noisiest of the three summary statistics chosen to represent the observed data.

of the agents or sheep are correlated, as defined in Section 3.6.5

$$I^{(t)}\left(\mathbf{x}^{(t)}\right) = \int_0^{x_c} f(x,t)\,\mathrm{d}x \quad \text{and} \quad I^{(t)}\left(\mathbf{z}_{[i]}^{(t)}\right) = \int_0^{x_c} f(x,t)\,\mathrm{d}x, \tag{5.6}$$

where $f(x,t)$ is the correlation between the velocity of agent j and its n^{th} nearest neighbour at time t, and x_c is defined as the neighbour where the velocity correlation goes below zero, Section 3.6.4. This can be seen in Figure 5.6 for each of out observational datasets.

The remaining ten videos either do not have periods of time where these quantities are increasing or the period of time is too short (number of viable frames less than 100). These three quantities will be used as summary statistics of both the simulated data and of our observational data.

The four observational videos have visually similar behaviour and this is reflected in each of the quantities increasing over time. However, the average speed in Video 3 has a more sinusoidal profile compared to the other three videos, Figure 5.4. The alignment in Video 4 peaks around $\tau = 0.25$ before decreasing back to $\psi \approx 0.5$, this could be due to the drone shifting position during filming which the stabilisation process was not able to fully rectify, Figure 5.5. These two videos have a more varied relationship between I and τ because of the noise seen in the other quantities, Figure 5.6. Due to the additional noise in these datasets the minimal $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values for $i = 3, 4$ may not be as small as the minimal $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ when $i = 1, 2$.

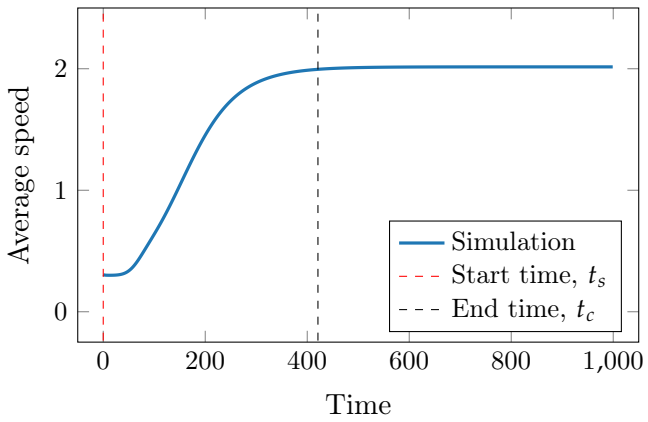


Figure 5.7: The average speed of simulated agents over time, showing the cut off points used to trim simulations (Simulation of modification 2 with $\sigma \approx 70$).

5.5 Determining t_j

Our simulations contain a longer time period than that of the observed datasets. Most of the simulations include a short period at the beginning of the simulation where the summary statistics are not increasing and small, and a larger period of time at the end of the simulation where the summary statistics are at a constant larger value. We are interested in comparing the period of time where the summary statistics are increasing so we truncate the simulations. The simulations are trimmed to start as the first of the summary statistics starts to increase, t_s, and end as the last of the summary statistics reaches within 1% of its final value t_c, Figure 5.7. We now have a curve for each of the summary statistics which increases over time for each simulation.

The sampling rate of the observational data and simulated data do not match. Figure 5.7 shows that we will have approximately 400 remaining simulated data points once it has been trimmed. This is almost twice as much information than we have in our longest observational dataset. So we use linear interpolation to obtain a the value of τ in the simulation for each time point where we have an observation. The time point $\tau = t_j$ is simply the time at which the j^{th} observation was made.

5.6 Using ABC With Vicsek Flocking Models

The ABC rejection scheme algorithm for approximating the posterior distribution using our collected data follows the same procedure as set out in the previous section, Algorithm 5.1.

All three summary statistics were chosen because they all measure the behaviour better

than the raw trajectories. The raw trajectories could be heading in different directions and be in different places at a certain time so a naive comparison would not show the similarities between simulation and raw data. the summary statistics so as to capture emergent flocking behaviour information without relying on similarly of raw trajectories. For example specific flocks of sheep becoming ordered over time from a state of disorder might demonstrate trajectories that do not align at any point, however, the statistics we use are consistent throughout several datasets and simulations. This consistency allows us to use them in the ABC parameter inference.

The three summary statistics that will be used to decrease the dimensionality of our data, both simulated and observed, are the mean speed of the agents, the global alignment of the agents and the integral lengthscale of the agents. For the ABC scheme we first calculate the distance between the summary statistics of the observed data and the summary statistics of the simulated data using

- The distance between simulated mean speed and observed mean speed over all time is defined as

$$\rho_v(\mathbf{x}, \mathbf{z}_{[i]}) = \frac{1}{T_{[i]}} \left| \sum_{j=1}^{T_{[i]}} \left\{ \frac{\langle |\dot{\mathbf{x}}^{(t_j)}| \rangle - \langle |\dot{\mathbf{z}}_{[i]}^{(t_j)}| \rangle}{\langle |\dot{\mathbf{z}}_{[i]}^{(t_j)}| \rangle} \right\} \right| \tag{5.7}$$

  where $\langle |\dot{\mathbf{x}}^{(t)}| \rangle$ and $\langle |\dot{\mathbf{z}}_{[i]}^{(t)}| \rangle$ is the mean speed of the agents in the simulated data and observed data at time t respectively and are calculated using Equation (5.4). The quantity $T_{[i]}$ is the number of observations in observational dataset i.

- The distance between simulated and observed global alignment over all time is calculated using

$$\rho_\psi(\mathbf{x}, \mathbf{z}_{[i]}) = \frac{1}{T_{[i]}} \left| \sum_{j=1}^{T_{[i]}} \left\{ \frac{\psi(\mathbf{x}^{(t_j)}) - \psi(\mathbf{z}_{[i]}^{(t_j)})}{\psi(\mathbf{z}_{[i]}^{(t_j)})} \right\} \right| \tag{5.8}$$

  where $\psi(\mathbf{x}^{(t)})$ and $\psi(\mathbf{z}_{[i]}^{(t)})$ is the global alignment of the simulation and observed data at time t respectively, computed using Equation (5.5). As with the distance between mean speeds, $T_{[i]}$ is the number of observations in observational data i.

Algorithm 5.1: Using ABC with the Vicsek model modifications and our observational data.

1. Draw parameters, ϕ^*, from the prior distribution;
2. Simulate $\mathbf{x}$ from the Vicsek model modification using ϕ^*;
3. Compare $\mathbf{x}$ to $\mathbf{z}_{[i]}$;
4. Accept ϕ^* if $\rho(\mathbf{x}, \mathbf{z}_{[i]}) < \varepsilon_{[i]}$;

- The distance between the integral lengthscales averaged over time using

$$\rho_I(\mathbf{x}, \mathbf{z}_{[i]}) = \frac{1}{T_{[i]}} \left| \sum_{j=1}^{T_{[i]}} \left\{ \frac{I(\mathbf{x}^{(t_j)}) - I(\mathbf{z}_{[i]}^{(t_j)})}{I(\mathbf{z}_{[i]}^{(t_j)})} \right\} \right| \tag{5.9}$$

  where $I(\mathbf{x}^{(t_j)})$ and $I(\mathbf{z}_{[i]}^{(t_j)})$ is the integral lengthscale of the simulation and observed data at time t respectively, Equation (5.6). As seen with the other two distance metrics, $T_{[i]}$ is the number of observations in observational data i.

These distances calculates the difference between the simulated and observed quantity and then normalise this by dividing by the observed quantity. By normalising in this fashion when calculating the average difference over time all three distance metrics are in the same order of magnitude despite the raw alignment being almost 3 orders of magnitude smaller than the integral lengthscale.

Once all three of the distances between observed and simulated summary statistics are calculated we use the average distance of those calculated in Equations (5.7), (5.8) and (5.9) as the distance used in the ABC rejection scheme when comparing to the i^{th} dataset, $\mathbf{z}_{[i]}$,

$$\rho(\mathbf{x}, \mathbf{z}_{[i]}) = \frac{1}{3} \left(\rho_v(\mathbf{x}, \mathbf{z}_{[i]}) + \rho_\psi(\mathbf{x}, \mathbf{z}_{[i]}) + \rho_I(\mathbf{x}, \mathbf{z}_{[i]}) \right). \tag{5.10}$$

By using the arithmetic mean of the three distance metrics we ensure that all three of the summary statistic distances have to be close in order for the parameter combination $\boldsymbol{\phi}^*$ to be accepted.

5.7 Initialising The Simulations

The initialisation of the simulations is vital to being able to create simulations where the three summary statistics increase. In order for the simulations to match our collected data we require a stimulus. For this we use agent N to mirror the stimulus the sheep in the data videos received from the quad bike. In order to do this we fix the speed to be faster than the remaining agents and also fix direction of the agent over time.

5.7.1 Initialising Agents

The initial locations of $N-1$ of the agents in the simulations are randomly allocated. The (x, y) coordinates for each agent is selected by independently randomly sampling from a continuous uniform distribution for both x and y. The width of the distribution is governed by the quantity $\bar{s}$, which is the average distance between an agent and its nearest

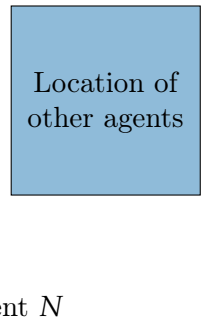


Figure 5.8: The starting location of agent N compared to the starting locations of the rest of the agents.

neighbour. We require $\bar{s}$ to be approximately the same as the average distance between a sheep and its nearest neighbour in the observed data, to ensure that we are comparing like-for-like. For Video 1 the average separation between a sheep and its nearest neighbour is approximately 30 pixels. When randomising the initial coordinates of the simulated agents using independent samples from a continuous $\mathcal{U}(0, 350)$ distribution for x and y the average separation between an agent and its nearest neighbour is also approximately 30.

In our observations the alignment starts small, meaning that the initial directions of the observed sheep are approximately random. Therefore we initialise the initial directions of the agents from a continuous uniform distribution $\mathcal{U}(-\pi, \pi)$.

The initial speeds of the agents are sampled from a normal distribution. The mean and standard deviation of this distribution are taken from the observed data at time $\tau = 0$. When comparing to Video 1 the normal distribution used as the prior for the initial speed is $\mathcal{N}(0.26, 0.12)$. The values realised from a normal distribution are not limited to positive values. If an agent is initialised with a negative speed, $-v$, in the direction θ then this is equivalent to the agent being initialised with direction $-\theta$ and speed v. Since the starting directions are random this is not a problem for initialisation.

5.7.2 Initialising Agent N

We initialise this agent's location to be slightly displaced from the rest of the agents, Figure 5.8. This ensures that our simulations have a burn in period, so we always capture the moment where the rest of the agents start to react to the stimulus. The direction of agent N is fixed at the beginning of the simulation to point directly towards the rest of the agents, Figure 5.8. By doing this we can be confident that the other agents will always interact with agent N.

Since we require the average speed of the simulated flock to increase once exposed to agent

Parameter	Bounds Lower	Upper	Modification			
r	0	120.00	1^*			
σ	0	120.00		2^*	3	4
α	0	4.00			3	
α	0	0.05			3	
α	0	1.00			3^*	4
β	0	4.00				4^*
β	0	0.05				4
β	0	10.00				4

Table 5.2: The upper and lower bound for the uniform prior distributions of parameters and which of the modified models it appears in. If the model number is written 1^* then this is the prior distribution used for model analysis.

N we fix the speed to be higher than the initial speed of the other agents. This is similar to how in the observational videos when the sheep become aware of the quadbike their speeds increase.

To allow for an easier comparison between simulated and observed data we fix the speed of agent N to be the mean speed of the observed sheep at the end of our dataset (when comparing to Video 1 this speed is $\langle|\dot{\mathbf{z}}_{[1]}^{T_{[1]}}|\rangle \approx 2$, where T_1 is the final time in dataset 1, $\mathbf{z}_{[1]}$). By doing this the average speed for both simulated and observed data should start at approximately 0 and end when the average speed is approximately $\langle|\dot{\mathbf{z}}_{[i]}^{T_{[i]}}|\rangle$, for all datasets, $\mathbf{z}_{[i]}$ where $i \in [1, 2, 3, 4]$. We could instead to use the average speed of the quadbike as the fixed speed of agent N, this would produce similar behaviour in the simulated agents as the average speed of the quadbike is approximately $\langle|\dot{\mathbf{z}}_{[i]}^{T_{[i]}}|\rangle$.

5.8 Prior Distributions Of The Model Parameters

ABC requires the parameter(s) of the Vicsek modifications, $\boldsymbol{\phi}^*$, to be sampled from the prior distribution. In the following sections we will discuss the prior distributions used for each parameter in $\boldsymbol{\phi}^*$. The parameters are:

- r in modification 1,
- σ in modifications 2, 3, and 4,
- α in modifications 3 and 4,
- β in modification 4.

Each of the model modifications as set out in the previous chapter are: modification 1 is the Vicsek model with variable speed, Section 4.2.1; modification 2 is the Vicsek model with variable speed and weighted Gaussian interactions, Section 4.2.2; modification 3 is a continuation of modification 2 but where the size of the Gaussian now depends on the agents neighbours' speed, Section 4.2.3; finally modification 4 changes the size of the Gaussian used for the weighted interactions based on both the agents speed and its neighbours' speed, Section 4.2.4.

For these parameters we assume a uniform prior and it is these parameters that we are interested in approximating the posterior distribution. For continuity when the parameter is used in multiple modifications we use the same continuous uniform distributions in each. We use a latin hypercube, defined in Section A.7, to sample from the prior distribution, the marginal prior distributions are shown in Table 5.2, to minimise the number of simulations needed to cover the entire parameter space.

5.9 Modification 1: Variable Speed

The first modification to the Vicsek model allows the agents to change their speed in the same way they update their direction, Section 4.2.1. The weights used for interactions in this modification at time t are

$$\begin{aligned} w_{ij}^{(t)} &= H\left[r_{ij}^{(t)} - r\right] \\ &= \begin{cases} 0, & r_{ij}^{(t)} - r \geq 0, \\ 1, & r_{ij}^{(t)} - r < 0, \end{cases} \end{aligned} \tag{5.11}$$

as previously shown in Equation (4.4), where $r_{ij}^{(t)}$ is the distance between agents i and j at time t. Hence this modification contains a single parameter, the radius of interaction, r.

For clarity in the following sections we focus how the simulations compare to Video 1, but will show the resulting posterior distributions from comparison with the other three observational datasets. Figure 5.9 shows the mean and and the mean plus/minus one standard deviation of the $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values from one ABC scheme, Algorithm 5.1, with the $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ split into bins of width $r = 1$. We can see that when r is close to 0, $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ is at its highest, therefore the model is not matching the observational data very well. This makes sense because in the simulations the agents will be too far apart to interact. As r increases the value of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ starts to decrease suggesting that at for the larger values of r the model is closest to the collected data with respect to $\rho(\mathbf{x}, \mathbf{z}_{[1]})$. Once $r > 100$

$\mathbf{z}_{[i]}$	Tolerance level $\varepsilon_{[i]}$
1	0.19
2	0.60
3	0.47
4	0.85

Table 5.3: Tolerances values used to obtain 1% of simulations with the smallest $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values for $i \in [1, 2, 3, 4]$ when comparing to modification 1.

the $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values do not decrease anymore which shows that by increasing r further the fit does not improve. We can take from this that approximately $r = 100$ all agents are interacting.

Figure 5.10 shows the histogram and the kernel density estimation distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values we obtain. The $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ distribution has three peaks, at approximately $\rho(\mathbf{x}, \mathbf{z}_{[1]})$= $0.3, 0.5$ and $\rho(\mathbf{x}, \mathbf{z}_{[1]})$= 0.8, for both runs of the ABC scheme. This shows that we are running enough simulations to cover the parameter space sufficiently. We run our ABC scheme, Algorithm 5.1, twice each with 20,000 simulations sampled across our parameter space in order to check we are running a sufficient number of simulations.

As can be seen in Table 5.2, the prior distribution of r is $\mathcal{U}(0, 120)$. The approximate posterior distribution for r is obtained by looking at best 1% of simulations produced by the ABC scheme. We define the best 1% of simulations as the 1% of the simulations with the smallest $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ value. The best 1% of simulations, when comparing to Video 1, is equivalent to a tolerance level of $\varepsilon_{[1]} = 0.19$. The tolerance levels for comparing to the other three datasets can be seen Table 5.3

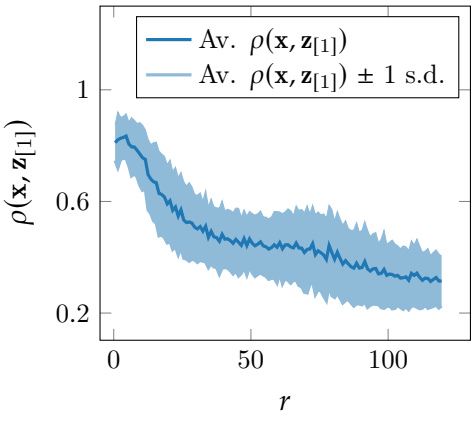


Figure 5.9: The relationship between $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ and r using all 20,000 simulations categorised into 120 bins when using modification 1.

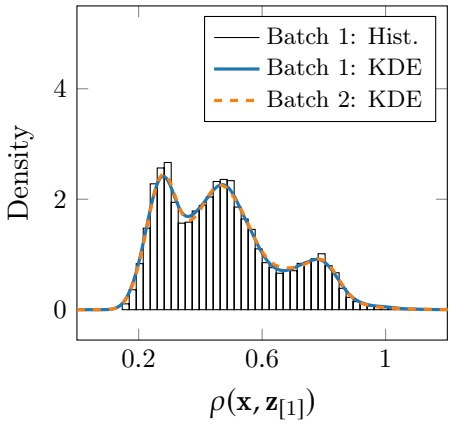


Figure 5.10: The distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values from all simulations of the ABC schemes for the first modification of the Vicsek model.

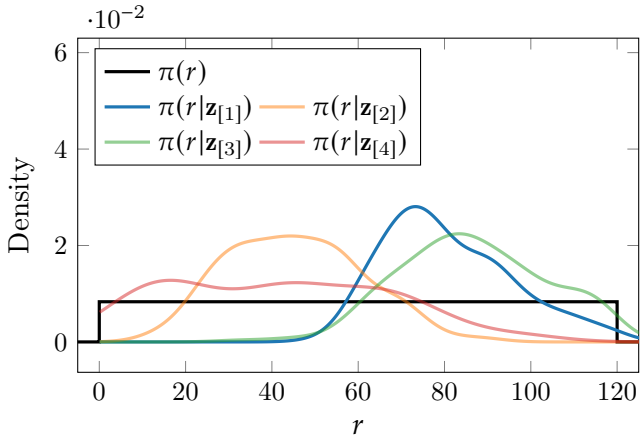


Figure 5.11: The prior distribution and the approximate posterior distributions from comparison with all four datasets for modification 1.

Figure 5.11 shows the prior distribution and approximate posterior distributions for r when comparing to all four observation datasets. We can see that in the approximate posterior distribution for r when comparing to Video 1, the values vary from about 50 up to 120, with a peak at 75. The density of r values at $r = 75$ is higher than the density in the prior distribution at this point. The density is now approximately 0 for all values of $r < 50$. The distributions for r when comparing to Video 1 and Video 3 are very similar, they each have r varying from 50 up to 120 and peak close to $r = 80$. The other two posterior distributions cover much smaller values of r. We see that the broadest distribution comes from comparison with Video 4 which had a large amount of noise in the summary statistics.

5.10 Modification 2: Gaussian Interactions

As with the first modification of the Vicsek model there is only a single parameter in the model with prior distribution $\sigma \sim \mathcal{U}(0, 120)$. We use the same limits for the uniform prior distribution for σ that was used for the radius of interaction r, since when $\sigma \approx r$ the influence an agent receives from its surroundings is similar. The weights used for interactions in the second modification at time t are

$$w_{ij}^{(t)} = \exp\left\{-\frac{(r_{ij}^{(t)})^2}{\sigma^2}\right\} \tag{5.12}$$

as previously shown in Equation (4.6), where $r_{ij}^{(t)}$ is the distance between agents i and j at time t. The equation to calculate the weights used in averaging now gives a larger weight

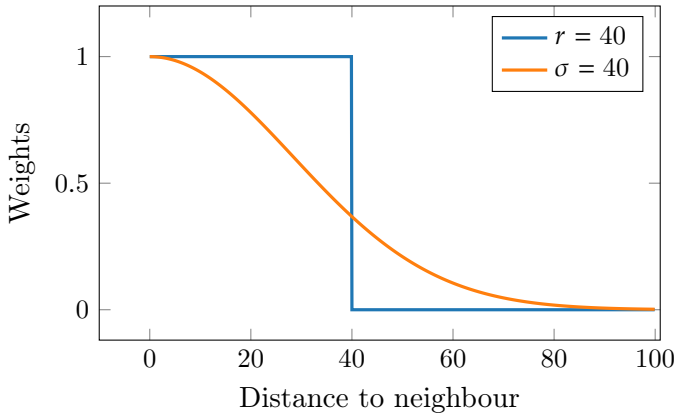


Figure 5.12: The area under the curve representing the radius of interaction (–) is comparable to the area under the curve for the Gaussian interactions (–).

to neighbours at further distances.

Figure 5.12 shows how the weight changes as the distance to the neighbour increases for each model, calculated using Equations (5.11) and (5.12). The area under the curve gives us an idea to how much an agent can be influenced by its neighbours. The area under the curve for the previous modification (–) is 40, whereas the area under the curve for this modification (–) is slightly smaller at 35.5. From this we might expect that the peak in the posterior distribution for σ to be at a slightly larger value that the location of the peak in the posterior distribution for r.

Figure 5.13 shows the mean $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ value and the mean plus and minus one standard deviation of the $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values for data split into bins of width 1. When σ is close to 0 the value of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ is at its highest and once σ gets larger then 100 the values of

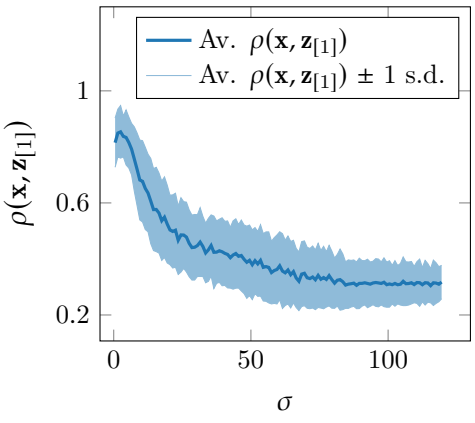


Figure 5.13: The relationship between $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ and σ using one batch of 20,000 simulations split into 120 bins.

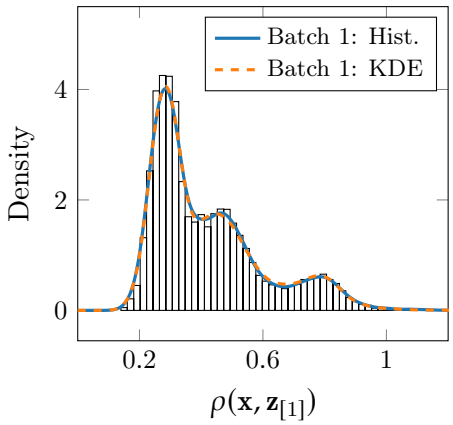


Figure 5.14: The distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values for all 20,000 simulations using Gaussian interactions.

Dataset $\mathbf{z}_{[i]}$	Tolerance level $\varepsilon_{[i]}$
1	0.19
2	0.87
3	0.47
4	0.84

Table 5.4: Tolerances values used to obtain 1% of simulations with the smallest $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values for $i \in [1, 2, 3, 4]$ when comparing to modification 2.

$\rho(\mathbf{x}, \mathbf{z}_{[1]})$ flattens out, in the same way as in the previous model. The relationship between σ and $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ is similar to that of r and $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ shown in Figure 5.9. Since we expect the behaviour when $\sigma \approx r$ to be similar, it is reassuring to see that $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ flattens out when $\sigma \approx 100$, showing that at these large σ values all agents are interacting.

Figure 5.14 shows the distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values we obtain from running two batches of the ABC rejection scheme, Algorithm 5.1, both of which with 20,000 simulations. This is similar to the distribution produced by the previous modification, Figure 5.10, but the first two peaks are much closer together, and the final peak is now much smaller. Since more simulations are achieving smaller $\rho(\mathbf{x}, \mathbf{z}_{[v]})$alues overall this modification is a better fit to our observational data.

In order to obtain the approximate posterior distribution for σ we use the same procedure that was used for r in the previous model. Looking at the distribution of σ values of the top 1% of simulations is equivalent to setting a tolerance level of $\varepsilon_{[1]} = 0.19$, tolerance levels for comparison with other dataset shown in Table 5.4. By changing type of interaction from a radius of interaction to a Gaussian interaction, we see that the peak in the approximate posterior distribution for the size of the interaction when comparing to Video 1 reduces from 75 to approximately 55, Figure 5.15. The spread of the distribution is also smaller. The posterior distributions for Video 2 and 3 see similar changes. The posterior distribution for σ when comparing to Video 4, $\pi(\sigma|\mathbf{z}_{[4]})$, is now slightly narrower, with a less density in the regions $\sigma < 10$ and $\sigma > 60$. This shift to smaller σ values is the opposite of our initial hypothesis. This could be explained by the neighbours which are further away now having a larger weight (in modification 2 it is non-zero) so agents start to interact even when far apart.

We can compare the posterior distributions, for r from comparing modification 1 and $\mathbf{z}_{[1]}$, $\pi(r|\mathbf{z}_{[1]})$, and for σ from comparing modification 2 with $\mathbf{z}_{[1]}$, $\pi(\sigma|\mathbf{z}_{[1]})$, by converting the parameter values into the equivalent area under the weight curve, A, for that parameter value. This quantity A can be thought of as the amount of influence the surroundings can have on an agent. The area under the weight curve for modification 1 where the weight

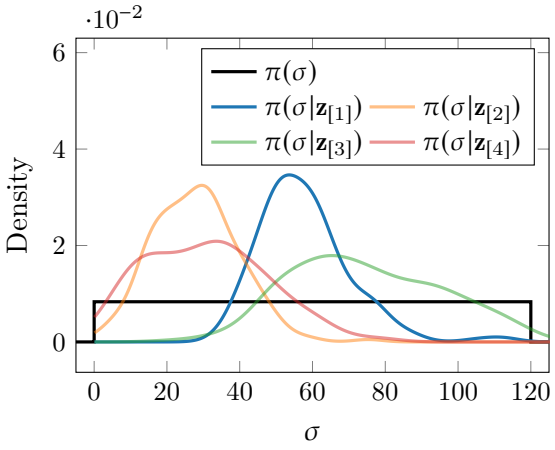


Figure 5.15: The prior and posterior distributions for each of the datasets $\mathbf{z}_{[i]}$ using modification 2.

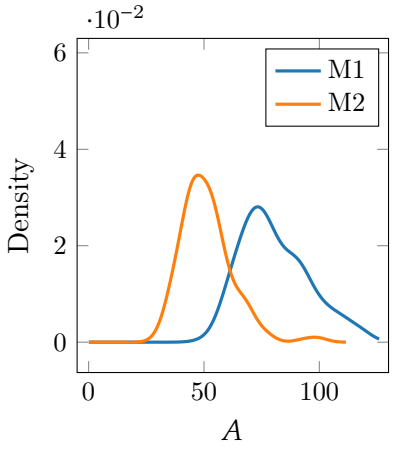


Figure 5.16: Distribution of A for modification 1 and modification 2.

function is given by Equation (5.11) is simply

$$\begin{aligned} A &= \int_0^\infty w_{ij}^{(t)} \, \mathrm{d}r_{ij}^{(t)} \\ &= \int_0^\infty H[r_{ij}^{(t)} - r] \, \mathrm{d}r_{ij}^{(t)} \\ &= r \end{aligned} \tag{5.13}$$

where $H[r_{ij}^{(t)} - r]$ is the heaviside function described in Section 5.9 and where $r_{ij}^{(t)}$ is the distance between agent i and its neighbour agent j at time t.

The area under the weight curve for modification 2 where the weight function as given by Equation (5.12) is

$$\begin{aligned} A &= \int_0^\infty w_{ij}^{(t)} \, \mathrm{d}r_{ij}^{(t)} \\ &= \int_0^\infty \exp\left\{ -\frac{\left(r_{ij}^{(t)}\right)^2}{\sigma^2} \right\} \mathrm{d}r_{ij}^{(t)} \\ &= \frac{\sqrt{\pi}}{2}\sigma. \end{aligned} \tag{5.14}$$

Figure 5.16 shows the distribution of A for both of these models. The peak for modification 1 is at a higher value of A that the peak in the distribution for modification 2. Therefore on average the agents simulated using the second modification to the Vicsek model require less influence from their neighbours than those simulated by the first modification to obtain the same behaviour.

5.11 Modification 3: Interactions Dependent On Neighbours Speed

Modification 3 has two parameters, σ and α, Section 4.2.3. The interaction weights for this modification are calculated using

$$w_{ij}^{(t)} = \exp\left\{-\frac{r_{ij}^2}{\left(v_j^{(t)}+1\right)^{\alpha}\sigma^2}\right\}. \tag{5.15}$$

as seen in Equation (4.7). The interaction weights now depend on both the distance between agents i and j at time t, $r_{ij}^{(t)}$, and also the speed of the neighbour at time t, $v_j^{(t)}$. Which as mentioned earlier allows us to model the idea of neighbours which are moving faster must have more information about danger or food than the individual. So the individual is more likely to follow them than neighbours which are moving slower.

To ensure that this model drops back to the previous modification when $\alpha = 0$ we look at the case where α has the continuous uniform prior distribution $\alpha \sim \mathcal{U}(0, 0.05)$. This can be case is shown in Figure 5.17 by the dashed green line (- -). Since we have limited knowledge about what the value of α should be we initially use a continuous uniform prior distribution $\mathcal{U}(0, 4)$. Figure 5.17 shows that allowing α to vary this much gives us much broader distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values than when using the prior $\alpha \sim \mathcal{U}(0, 0.05)$. Therefore, the larger values of α result in larger values of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ so the upper bound of the continuous uniform prior distribution can be reduced to aid with computation and to allow for a more thorough investigation into an appropriate parameter space. Hence for our model comparisons we use the continuous uniform prior $\alpha \sim \mathcal{U}(0, 1)$ as shown in Table 5.2. Figure 5.17 shows that when $\alpha \sim \mathcal{U}(0, 1)$ we obtain fewer simulations which result in $\rho(\mathbf{x}, \mathbf{z}_{[1]}) > 0.5$ and more simulations which result in $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values at the peak of the distribution, $\rho(\mathbf{x}, \mathbf{z}_{[1]}) \approx 0.3$. Therefore a value of α in this range gives better results than the previous model where $\alpha = 0$.

When simulating from this modification we use the same continuous uniform prior distribution for σ as was used in the previous modification, $\sigma \sim \mathcal{U}(0, 120)$, as shown in Table 5.2.

How the distance metric $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ changes as the individual parameters, σ and α, are varied can be seen in Figure 5.18. This shows that there is a range of (σ, α) values which result in low distance ($\rho(\mathbf{x}, \mathbf{z}_{[1]}) < 0.35$). In the region where α decreases from 0.9 to 0.3 and as σ increases from 60 to 80 the distance $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ is at its smallest. Since this region depends on both σ and α values the two parameters have a non-zero correlation. As seen in modification 2, Figure 5.14, when $\sigma < 15$ the value of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ increases due to agents not interacting with each other. Figure 5.19 shows the distribution of the resulting

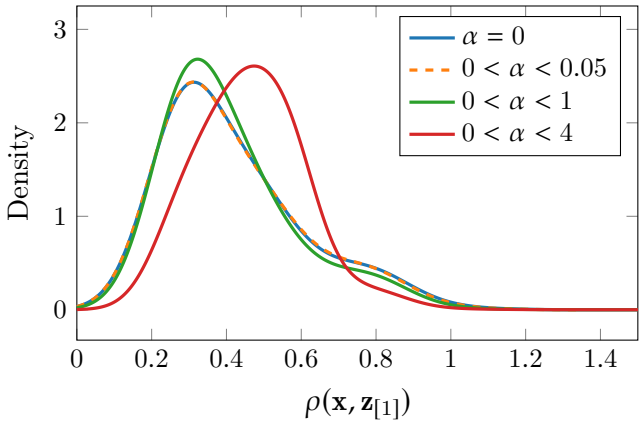


Figure 5.17: The distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values for M3 with α taking different ranges of values.

distance metric values, $\rho(\mathbf{x}, \mathbf{z}_{[1]})$, for both batches of the ABC scheme, Algorithm 5.1, when using this model. The $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ distributions for both batches are very similar, so despite increasing the number of dimensions of the parameter search the number of simulations is still sufficient. The $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ distribution no longer has three peaks, but instead has one broad peak and one smaller bump. The small bump in the distribution comes from the small σ values, $\sigma < 10$. Regardless of the α value chosen when σ is in the range 0–15 the resulting fit to the collected data is poor, $\rho(\mathbf{x}, \mathbf{z}_{[1]})> 0.8$. This was also observed in the 2D surface plot shown in Figure 5.18.

The joint posterior distribution for σ and α is shown in Figure 5.20. The posterior distribution for comparison with $\mathbf{z}_{[1]}$ is calculated using a tolerance level of $\varepsilon_{[1]} = 0.19$. Table 5.5 shows the tolerance levels for the other observation comparisons. The joint posterior distribution, $\pi(\sigma, \alpha|\mathbf{z}_{[1]})$, peaks when $\sigma \approx 50$ and $\alpha \approx 0.7$. The peak in σ is reflected in the marginal posterior distribution for σ but the peak in α does not appear in its marginal posterior distribution which reiterates the implication that there is correlation between the two model parameters.

The marginal posterior distribution for σ when comparing to $\mathbf{z}_{[1]}$ and $\mathbf{z}_{[3]}$ peak at approximately the same value, $\sigma \approx 50$. However, when comparing to $\mathbf{z}_{[3]}$ it is very broad distribution spanning the entire domain with only a slight increase in density at the peak of the distribution. The marginal posterior distributions when comparing to the other datasets both peak at a smaller value, $\sigma \approx 20$ for comparison with $\mathbf{z}_{[2]}$ and $\sigma \approx 10$ for comparison with $\mathbf{z}_{[4]}$.

The marginal approximate posterior distributions for α appear to split into two categories across our observational datasets. The comparison with $\mathbf{z}_{[1]}$ results in a distribution which cover all values between 0–1 but has a slight peak at $\alpha > 0.5$. Whereas the other three observational datasets result in distributions where they still cover all values 0–1 but the

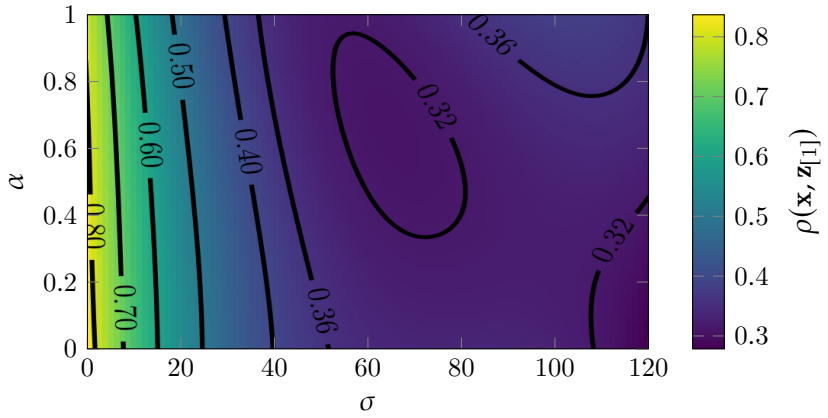


Figure 5.18: A 2D surface showing contours fitted to the $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values obtained from a single ABC scheme using modification 3.

peak is much more well defined and is at a smaller, $\alpha \approx 0.15$.

5.12 Modification 4: Interactions Dependent On All Speeds

The final modification has three parameters in the weighted interactions: σ, α and β. The interaction weights are calculated using

$$w_{ij}^{(t)} = \exp\left\{-\frac{r_{ij}^2\left(v_i^{(t)}+1\right)^{\beta}}{\left(v_j^{(t)}+1\right)^{\alpha}\sigma^2}\right\}, \tag{5.16}$$

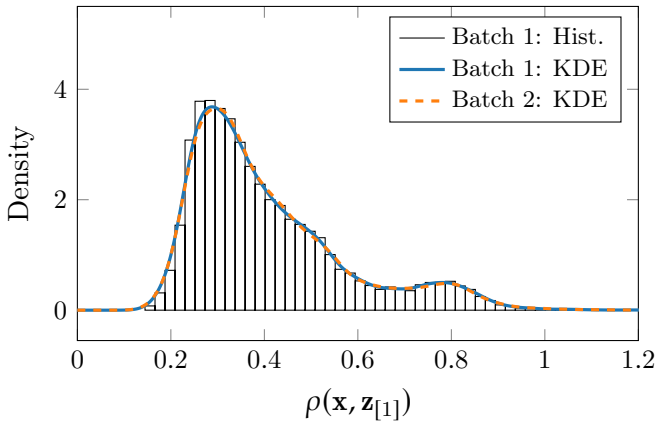


Figure 5.19: The distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values for two batches of 20,000 simulations using Gaussian interactions with variable speed using $\sigma \sim \mathcal{U}(0, 120)$ and $\alpha \sim \mathcal{U}(0, 1)$, model M4.

Dataset $\mathbf{z}_{[i]}$	Tolerance level $\varepsilon_{[i]}$
1	0.19
2	0.95
3	0.58
4	0.91

Table 5.5: Tolerances values used to obtain 1% of simulations with the smallest $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values for $i \in [1, 2, 3, 4]$ when comparing to modification 3.

where $r_{ij}^{(t)}$ is the distance between agent i and j at time t and $v_i^{(t)}$ and $v_j^{(t)}$ are their respective speeds at this time, as previously seen in Equation (4.8). This modification allows the weight of the interactions to depend on the agents own speed: if it is going fast then its interaction weights will be smaller, and conversely when an agent is going slow its interaction weights will be larger. As with the previous modification we assume that an agent which is moving faster than its neighbours has more information about its surroundings. Before we allowed faster agents to influence its neighbours more than slow agents, this modification is a continuation of that idea. Now faster agents can be influenced less by its neighbours as it has more information about its surroundings than they do.

As with introducing the α parameter in modification 3, Section 5.11, we had very limited

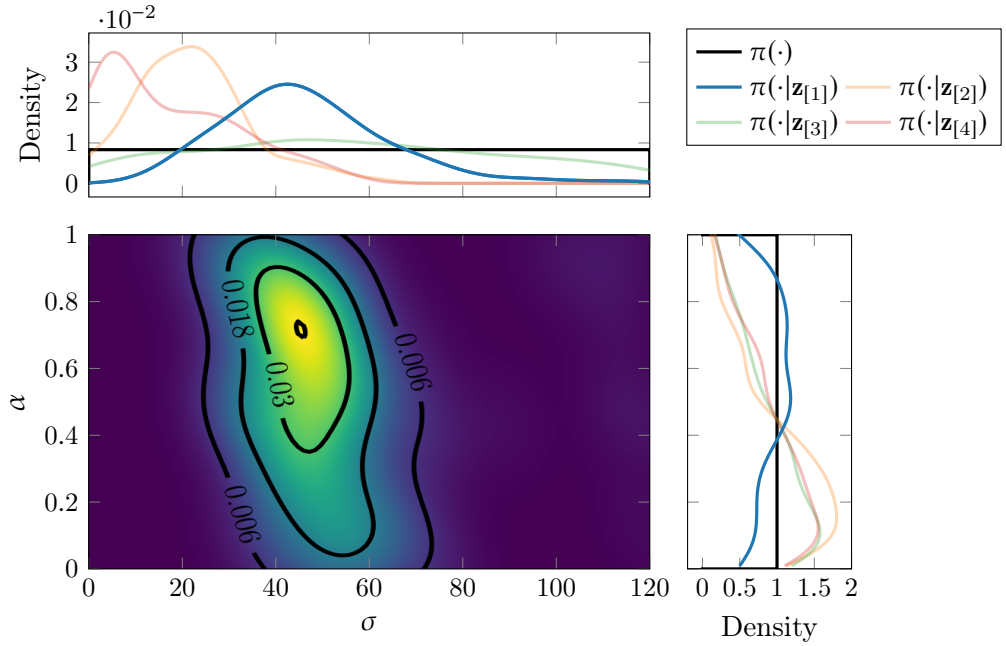


Figure 5.20: Approximate joint posterior distribution for comparison with $\mathbf{z}_{[1]}$ and marginal distributions of the parameters σ and α from the comparison with all four datasets when using modification 3.

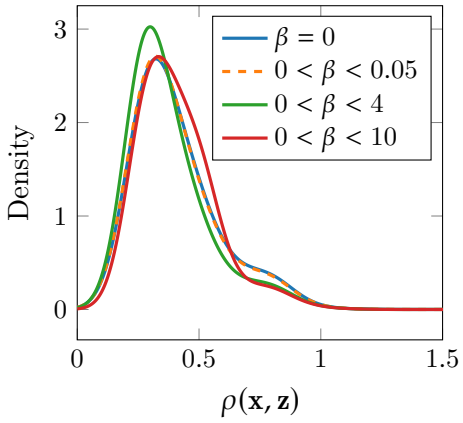


Figure 5.21: The distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values with β using different priors in M4.

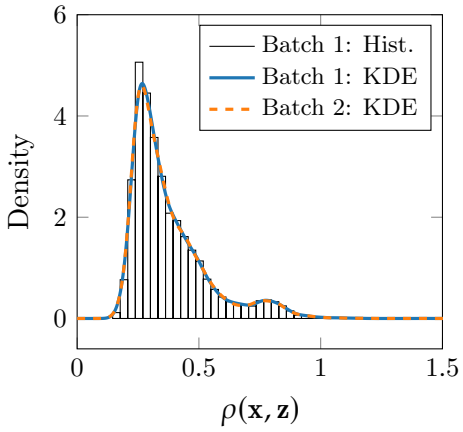
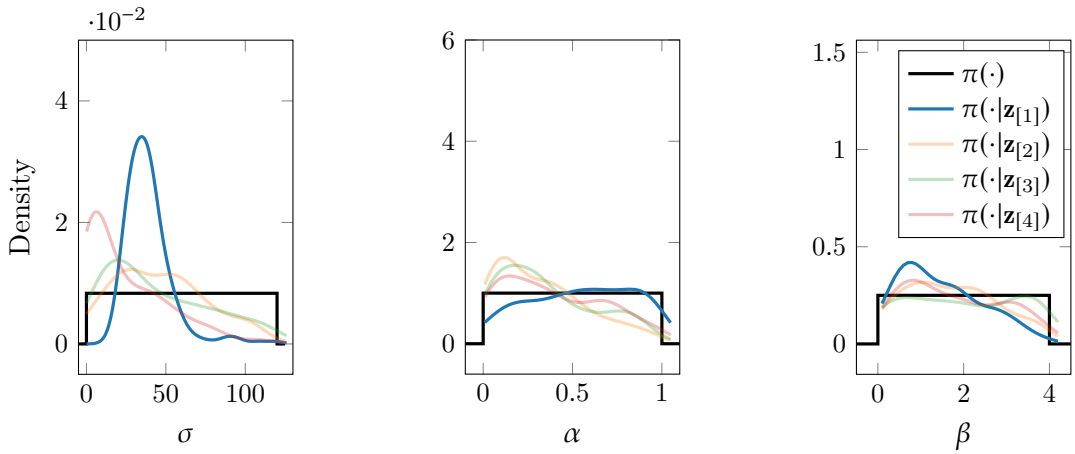


Figure 5.22: The distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values for both ABC schemes of M4.

prior knowledge of what the value for β should be so we gave it the prior distribution $\mathcal{U}(0, 4)$. Figure 5.21 shows that using this prior distribution shifts the distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ to the left giving us a better fit than that of the previous model. When we allow β to have a broader prior distribution ($\beta \sim \mathcal{U}(0, 10)$) we see that the overall performance of the model is worse so therefore to aid inference, a smaller continuous uniform prior distribution where $\beta < 4$ is sufficient. As before when adding α, Section 5.11, we check that the previous model can be realised when $\beta \approx 0$. Figure 5.21 shows that the curve for the previous modification where $\beta = 0$ and using this modification where $\beta < 0.05$ are equivalent.

The distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values when using the final Vicsek modification model, Figure 5.22, shows that the distribution still comprises of a large peak at $\rho(\mathbf{x}, \mathbf{z}_{[1]}) < 0.5$ and a small bump at $\rho(\mathbf{x}, \mathbf{z}_{[1]}) \approx 0.75$. The small bump in the $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ distribution was also observed in the distribution $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ when using the previous model. It peaks at approximately the same $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ value but it is now composed of less density. The decrease in density at larger $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values implies that the over all fit of the model is improving as we add more complexity. The two independent batches of the ABC scheme are approximately the same therefore even in a 3D parameter search space the number of simulations is sufficient.

Figure 5.23 shows the three approximate marginal posterior distributions for σ, α and β and their respective marginal prior distributions and Figure 5.24 shows the pairwise approximate joint marginal posterior distributions. The approximate posterior distributions we obtained by seeing the tolerance level to $\varepsilon_{[1]} = 0.19$ leaving 1% of the simulations. The tolerance levels for the 1% of simulations with the smallest $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values for all $i \in [1, 2, 3, 4]$ can be seen in Table 5.6. From these distributions we can see that of the



(a) Distributions for σ. (b) Distributions for α. (c) Distributions for β.

Figure 5.23: The prior distributions and approximate marginal posterior distributions for the parameters in modification 4.

three parameters we inferred more about the parameter σ than the other two. This is shown by the amount the distribution changed from prior to posterior.

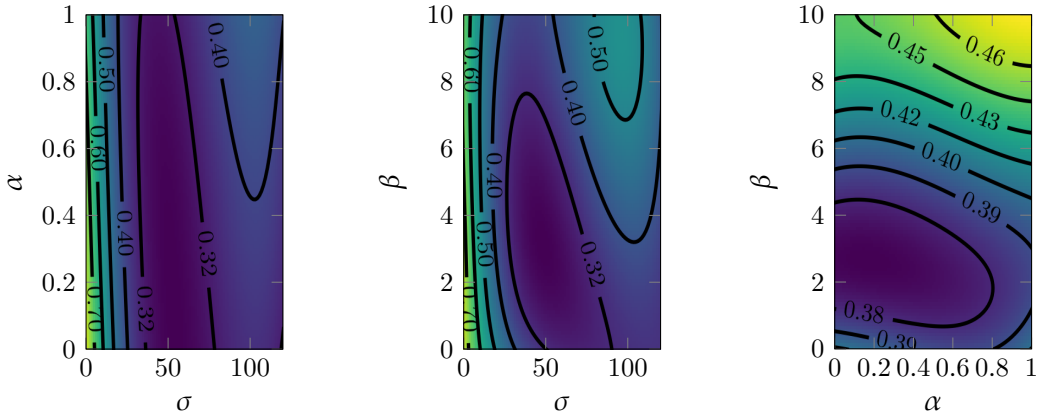
The posterior distributions for σ are now very much skewed towards the smaller values of σ. The comparison with Video 1 suggests that the underlying value of σ lies in the range 20–60. Whereas the comparison with the other three datasets, $\mathbf{z}_{[i]}$ where $i = 2, 3, 4$, suggest that smaller values may also be acceptable.

The approximate posterior for α shows very little change from the prior .When considering the posterior distributions for α we can see that one of the distributions is skewed towards larger α whereas the other three are skewed towards smaller α value. However, all four distributions have small density values, < 2, at their peak suggesting that more simulations or comparing more summary statistics might be necessary to determining the underlying value of α in the observations.

The peaks in the posterior distributions for β are more well defined than those in the posterior distributions for α. In the simulations we allowed β to vary from 0–4 from the

Dataset $\mathbf{z}_{[i]}$	Tolerance level $\varepsilon_{[i]}$
1	0.19
2	0.97
3	0.62
4	0.92

Table 5.6: Tolerances values used to obtain 1% of simulations with the smallest $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values for $i \in [1, 2, 3, 4]$ when comparing to modification 4.



(a) σ and α (b) σ and β (c) α and β

Figure 5.24: Approximate joint posterior distributions for comparison with $\mathbf{z}_{[1]}$ when using modification 4.

posterior distributions we can conclude that β is more likely to lie in the 0–2. All four of the posterior distributions for β have approximately the same shape with a peak around $\beta = 1$ and the density reducing as β increases. With a peak at $\beta > 0$ we can conclude that the inclusion of $v_i^{(t)}$ in the calculation of the interaction weights is important to the fit of the model. Physically this is the equivalent to animals moving at higher speeds interacting less with their surroundings.

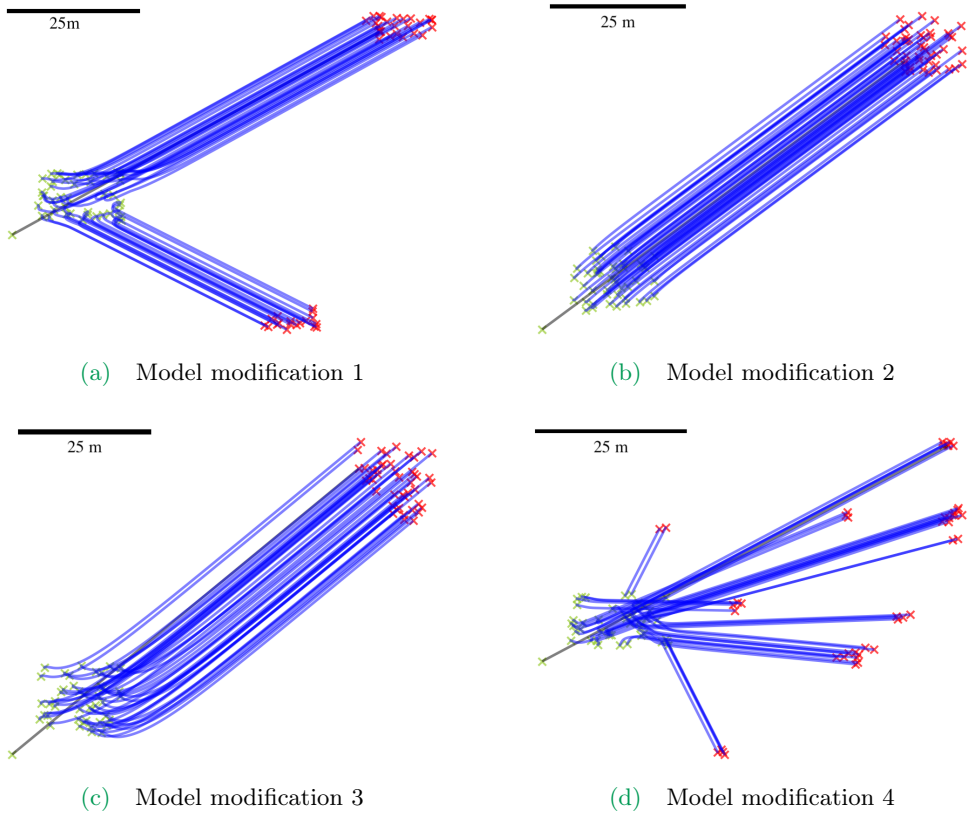
5.13 Trajectories from Simulation

We can view example trajectories of the simulated agents using the parameters from near the peak of the approximate posterior distributions. The selected examples shown in Figure 5.25 are for the four model modifications described in the previous sections. The parameters input in these simulations are show in Table 5.7.

The initial condition for each example simulation was set as described in Section 5.7.1. All example simulations had 45 agents, and the speed of agent N was fixed at the mean speed of the observed sheep at the end of our dataset from Video 1.

Model Modification	N	R	σ	α	β
M1	45	70	-	-	-
M2	45	-	70	-	-
M3	45	-	50	0.7	-
M4	45	-	50	0.3	3

Table 5.7: The parameter values used in the example trajectories.



(a) Model modification 1

(b) Model modification 2

(c) Model modification 3

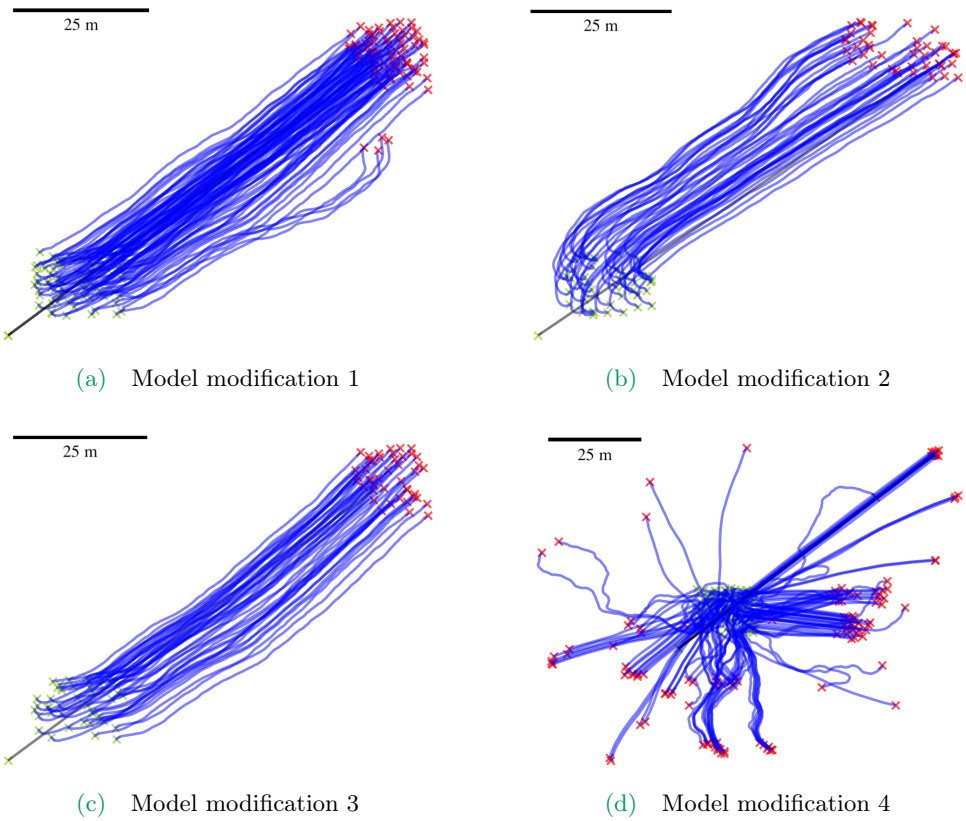
(d) Model modification 4

Figure 5.25: Example trajectory plots from the four Vicsek model modification used optimal parameter values.

5.13.1 Examples with no noise

As used so far in this chapter the following examples are simulated with no noise, $\xi = 0$ in Equation (4.2). This allows us to more easily observe the relative difference in behaviour for the four models and shows the type of trajectories they can produce. It is trajectories such as these that were used as input for the ABC method.

For each model modification a change in behaviour can be observed in the trajectory plots. Each additional level of complexity produces more varied trajectories. For example model 4 includes β a term that changes the behaviour of the agents depended on their velocity. This can be seen directly in the surprisingly different trajectories as compared to the other three model modifications. This is because as $\beta > 3$ the agents are influenced less by their surrounding so there is even less of a cohesive force keeping the flock together.



(a) Model modification 1

(b) Model modification 2

(c) Model modification 3

(d) Model modification 4

Figure 5.26: Example trajectory plots from the four Vicsek model modification used optimal parameter values including a small noise term.

5.13.2 Including a small amount of noise

The full modified models included a noise term that was previously suppressed throughout this chapter for ease of parameter inference. Here we include the noise term so that we can simulate sample trajectories from the full model reintroducing small differences between agents.

The trajectories in Figure 5.26 use the same parameters as in Figure 5.25 but now have $\xi = 0.1$ which was chosen by observation as as to only make a small effect and not overwhelm the model.

The overall behaviour is similar to the previous case with no noise, however variations in individual trajectories can be observed.

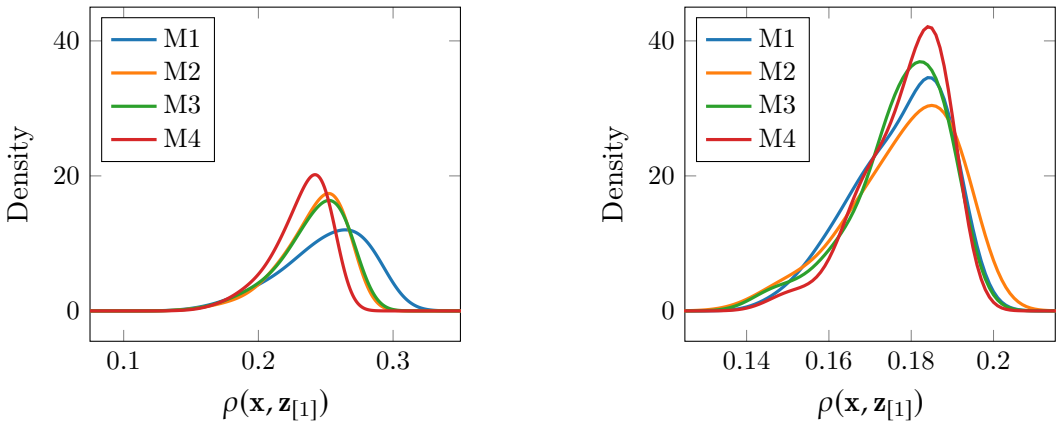
Model Modification	20%	1%
M1	0.292	0.191
M2	0.270	0.194
M3	0.272	0.191
M4	0.256	0.190

Table 5.8: The tolerances equivalent to looking at the best p%, $p \in \{1, 20\}$, of simulations for each model when comparing to Video 1.

5.14 Comparing The Model Variations

Looking at the best 20% of simulations when using the observational dataset from Video 1, $\mathbf{z}_{[1]}$, we obtain the tolerances shown in the fist column of Table 5.8. The tolerances show that on average as the model complexity goes up the closer the simulations get to the collected data. This can also be seen in Figure 5.27a, which shows the distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values for each model once the tolerance has been applied. The peaks of each of the distributions moves to smaller $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values and the distributions get narrower as the model goes from modification 1 to modification 4. This means that there are more parameter combinations that fit closer to the collected data for the higher dimensional models.

However, when looking at the best 1% of simulations for comparing to the observational dataset from Video 1, $\mathbf{z}_{[1]}$, using the tolerance levels in the second column in Table 5.8 there is now very little difference between the models. Figure 5.27b shows the distributions of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values across the four model modifications are approximately the same when the



(a) The best 20% of simulations.

(b) The best 1% of simulations.

Figure 5.27: The distribution of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values for the best p% of simulations for all 4 modifications.

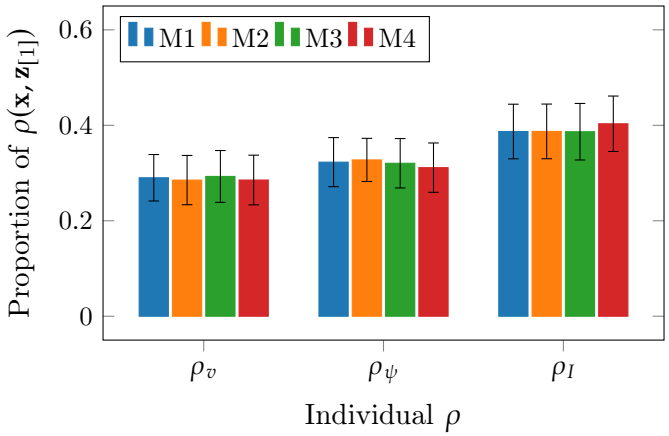


Figure 5.28: Proportion of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ from each ρ_j with standard deviation shown in the error bars.

tolerance level is set to $\varepsilon = 0.19$. The distribution for each model now has approximately the same peak at $\rho(\mathbf{x}, \mathbf{z}_{[1]}) \approx 0.18$ and spread where the range of values is $0.14 < \rho(\mathbf{x}, \mathbf{z}_{[1]}) < 0.19$. This implies that when using this family of models to recreate the behaviour seen in Video 1 there appears to be a cap on how close the models can recreate the observed data.

For qualitative comparisons we can use the best 20% of simulations for model comparisons as there is a discernible difference in the distributions of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ values when using the best 20% of simulations but not when using the best 1% version. However, since we used ABC to estimate our posterior distributions more complex methods of comparison such as Bayes factor should be used in future work.

We can breakdown $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ into its three constituent parts (ρ_v, ρ_ψ and ρ_I) and look at the proportion each ρ_j passes on to the combined distance metric, $\rho(\mathbf{x}, \mathbf{z}_{[1]})$. Figure 5.28 shows proportion of ρ_j in $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ for each of the four models. We see that all three distance metrics for each of the models account for approximately a third of the full $\rho(\mathbf{x}, \mathbf{z}_{[1]})$. This means that no one summary statistic is more influential than the others for comparison with all four models.

5.15 Conclusions

In summary using approximate Bayesian computation we were able to calculate the approximate joint posterior distribution for the parameters in each of the four Vicsek model modifications for each of the four observational datasets. These approximate posterior distributions characterise the behaviour seen in our observational datasets, $\mathbf{z}_{[i]}$ $i \in \{1, 2, 3, 4\}$.

As previously described in Chapter 4 modification 1 is the Vicsek model with variable

speed with a radius of interaction parameter r. When comparing to $\mathbf{z}_{[1]}$ and $\mathbf{z}_{[3]}$ the approximate posterior distributions had bell shaped curve in the range $50 < r < 120$ with well defined peaks at $r = 70$ and $r = 90$ respectively. For these two observations simulations using an interaction radii of less than 50 do not produce comparable behaviour. For the other two observational datasets, $\mathbf{z}_{[2]}$ and $\mathbf{z}_{[4]}$, the approximate posterior distributions are much flatter meaning that less information was inferred from the observational data. The approximate posterior distributions for these two observational datasets are also skewed toward smaller values, which means in order to best replace the behaviours seen in $\mathbf{z}_{[2]}$ and $\mathbf{z}_{[4]}$ smaller radii of interaction should be used when using modification 1.

When moving on to modification 2 where the radius of interaction is replaced with a Gaussian with standard deviation σ we see similar results. The approximate posterior distributions for $\mathbf{z}_{[1]}$ and $\mathbf{z}_{[3]}$ are skewed towards large values of σ, $40 < \sigma < 100$. Whereas the other two approximate posterior distributions for $\mathbf{z}_{[2]}$ and $\mathbf{z}_{[4]}$ are skewed the other way, $0 < \sigma < 60$. The approximate posterior distributions for these two observational datasets now have defined peaks meaning that more information could be inferred when using modification 2 compared to modification 1.

When using modification 3 the size of the Gaussian depends on the agents neighbours' speed to the power of α (when $\alpha = 0$ we return to modification 2). For this model we find that for $\mathbf{z}_{[1]}$ the approximate joint posterior distribution peaks at $\sigma \approx 50$ and $\alpha \approx 0.7$. Whereas the approximate posterior distributions for the other three observational datasets peak at smaller σ and α values.

Finally when simulating using modification 4 (which allows the size of the Gaussian to change with respect to both the agents speed and its neighbours' speed) with parameters σ, α and β we again see $\mathbf{z}_{[1]}$ producing a different approximate joint posterior distribution to the other observations. When looking at the approximate marginal posterior distributions we see that for $\mathbf{z}_{[1]}$ the peaks are $\sigma \approx 40$, $\alpha \approx 0.75$ and $\beta < 1$. When comparing to the other observational dataset the approximate marginal distributions for σ is flatter and peaks at smaller values $\sigma < 30$. The approximate marginal distributions for α are now skewed towards the smaller values with the peaks at $\alpha \approx 0.2$. The approximate marginal distributions for β for all four observational datasets are very similar.

The approximate posterior distributions when using modification 1 split the observational datasets into two categories. Two datasets produced approximate posteriors distributions skewed to the higher values of r and the other two produced approximate posterior distributions skewed the other way towards the smaller values of r. When the model gets more complex the differences between the four observations become less. This supports the idea that the more complex model, modification 4, can account for a much more varied behaviour. Additionally comparing the distributions of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ for the best 20% of sim-

ulations we are able see that as the model complexity increase the more the distribution is skewed towards smaller values of $\rho(\mathbf{x}, \mathbf{z}_{[1]})$ mean that more of the simulations have a closer fit to the observations.

In the next chapter we will be following a similar ABC rejection scheme to compute the approximate posterior distributions for the parameters in the combined sheep model when compared to observational datasets 9, 10 and 11.

6

Comparison Of The Combined Sheep Model To Observational Data

6.1 Introduction

In this chapter we will investigate steady-state behaviour rather than emergent behaviours which we have been focusing on so far in Part III. We will compare three observational datasets to the combined sheep model described in Section 4.6. The combined sheep model was not appropriate for comparison to the emergent behaviour as the transition from disordered to ordered happens too quickly and is chaotic. As before, an approximate Bayesian computation rejection scheme will be used to obtain a posterior distribution of the parameters in the combined sheep model for each observational dataset. We will then look at the trajectories of the simulated agents using the model with optimal parameter values to compare likeness with the observed trajectories of the collected data.

6.2 ABC Rejection Scheme

The ABC rejection algorithm follows the same procedure as in the previous chapter, Section 5.3, but since the combined sheep model produces different behaviour to that of the Vicsek modifications we have to define different summary statistics. From simulations of this model, the summary statistics used in the previous chapter, (the average speed, the global alignment and the integral lengthscale), no longer increase over time. The global alignment of the simulated agents stays consistently high for the full simulation and because of this looking at the integral lengthscale would not give us any extra information not already provided by the alignment and the average speed. This is because the integral lengthscale is a measure of the distance over which the velocities are correlated.

The parameter c in the combined sheep model, controls the size of the repulsion force

Algorithm 6.1: Using ABC with the combined sheep model and our observational data.

1. From the prior, sample parameters ϕ^*;
2. Simulate $\mathbf{x}$ from the combined sheep model using ϕ^*;
3. Compare $\mathbf{x}$ to $\mathbf{z}_{[i]}$, where $i \in [9, 10, 11]$;
4. Accept ϕ^* if $\rho(\mathbf{x}, \mathbf{z}_{[i]}) < \varepsilon_{[i]}$ where $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ is the average distance between observed and simulated summary statistics, Equation (6.7);

between agents, so as a result of that it also controls the distance between an agent and its nearest neighbour. Therefore we are interested in how the average distance to the nearest neighbour compares between simulated and observed data. We denote this distance of an observed dataset at time t to be $D^{(t)}([i])$.

The ABC rejection scheme algorithm stays very much the same as in the previous chapter but we now use the combined sheep model instead of the Vicsek model modifications and we use the datasets, $\mathbf{z}_{[i]}$ where $i \in [9, 10, 11]$, Algorithm 6.1.

6.3 Observational Data

The observational data we will be comparing the combined sheep model to is not the same observational data which we compared the Vicsek models to. This is because the behaviour observed from the combined sheep model results in the summary statistics described in the previous section being constant rather than increasing. This is due to the combined sheep model producing behaviour more readily seen in 'steady-state' flocking rather than the emergent collective behaviour which the modified Vicsek model was able to reproduce. The observational datasets we use in this chapter come from Videos 9, 10 and 11. The trajectory plots for these datasets can be seen in Appendix C in Figures C.9, C.10 and C.11. As used previously we denote the observational datasets as $\mathbf{z}_{[i]}$ where $i \in [9, 10, 11]$ is the number of the video it came from. These datasets were visually identified as having periods of time where alignment, speed and average distance to nearest neighbour all remain approximately constant. The remaining eleven videos either do not have periods of time where these quantities are all constant, there is no period of time where the alignment is both constant and high or these periods of time is too short (number of viable frames less than 100).

The observational datasets have differing lengths, Table 6.1. Our largest observational dataset comes from Video 9, $\mathbf{z}_{[9]}$, which comprises of 295 data points and is almost 3 times larger than the smallest dataset, $\mathbf{z}_{[11]}$. We normalise the time period so that time starts at $\tau = 0$ and ends at $\tau = 1$. This makes it easier to compare the videos to each other and also to the simulated data. Once time has been normalised we can look at each of the

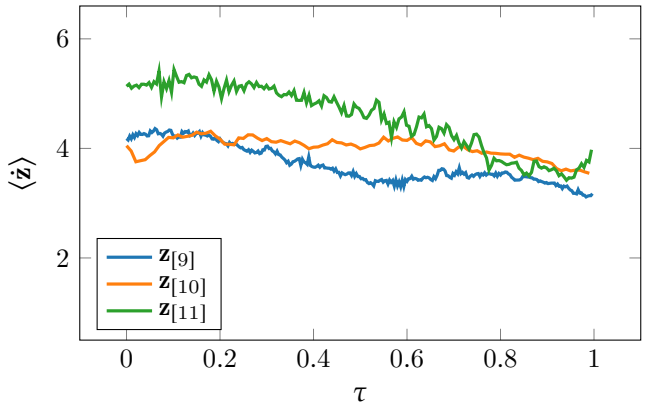


Figure 6.1: The average speed over rescaled time, τ, for all three observational datasets used in comparison with the combined sheep model.

summary statistics mentioned above over τ.

As seen in the previous chapter, Equation (5.4), the mean speed of the observed sheep and the simulated agents is calculated using

$$\begin{aligned} \langle |\dot{\mathbf{x}}^{(t)}| \rangle &= \frac{1}{N} \sum_{j=1}^{N} |\dot{\mathbf{x}}_j^{(t)}|, \\ \langle |\dot{\mathbf{z}}_{[i]}^{(t)}| \rangle &= \frac{1}{N} \sum_{j=1}^{N} |\dot{\mathbf{z}}_{[i],j}^{(t)}|, \end{aligned} \tag{6.1}$$

where $|\dot{\mathbf{x}}_j^{(t)}|$ is the speed of simulated agent j at time t, and $|\dot{\mathbf{z}}_{[i],j}^{(t)}|$ is the speed of the j^{th} observed sheep at time t in observational dataset i. For each observational datasets we can see how the mean speed varies over τ, Figure 6.1. The mean speed for $\mathbf{z}_{[9]}$ and $\mathbf{z}_{[10]}$, have an approximately constant speed at around 4, whereas the average speed for $\mathbf{z}_{[11]}$ decreases over time. This change in speed is not large enough to cause issues.

The global alignment of the agents at time t, as described in Chapter 3, for both simulated

	Observation length	
Dataset	Frames ($T_{[i]}$)	Seconds
$\mathbf{z}_{[9]}$	295	12.29
$\mathbf{z}_{[10]}$	187	7.79
$\mathbf{z}_{[11]}$	100	4.17

Table 6.1: The number of frames and time covered in each of the observational datasets using for comparison with the combined sheep model, $\mathbf{z}_{[i]}$ where $i \in [9, 10, 11]$.

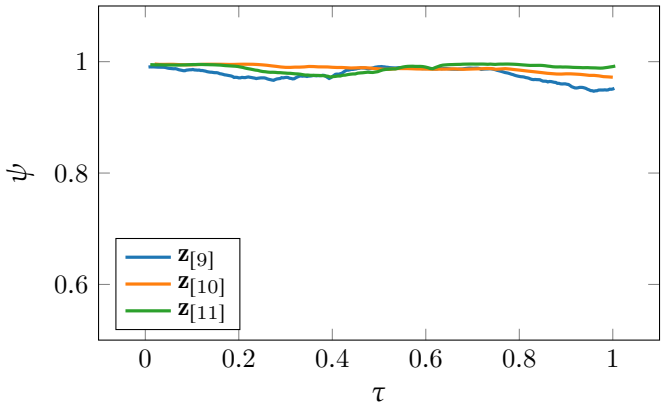


Figure 6.2: The global alignment over rescaled time, τ, for the observational datasets used in comparison with the combined sheep model.

agents and observed sheep, is defined as

$$\psi^{(t)}(\mathbf{x}^{(t)}) = \frac{1}{N}\left|\sum_{j=1}^{N}\dot{\mathbf{x}}_j^{(t)}\right|,$$
$$\psi^{(t)}(\mathbf{z}_{[i]}^{(t)}) = \frac{1}{N}\left|\sum_{j=1}^{N}\dot{\mathbf{z}}_{[i],j}^{(t)}\right|, \tag{6.2}$$

where $\dot{\mathbf{x}}_j^{(t)}$ is the velocity of agent j at time t, and similarly $\dot{\mathbf{z}}_{[i],j}^{(t)}$ is the velocity of sheep j at time t in observational dataset i, as previously seen in Equation (5.5). The global alignment over time τ, for each the observational datasets is shown in Figure 6.2. We can see that for all three datasets the alignment is constantly approximately 1.

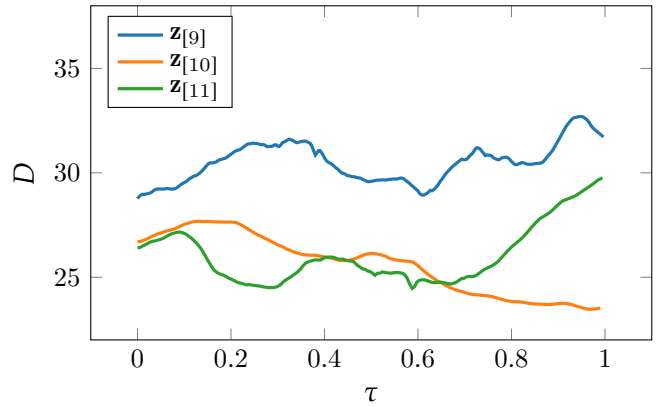


Figure 6.3: The average distance to the nearest neighbour over rescaled time, τ, for all three observational datasets used in comparison with the combined sheep model.

The third and final summary statistic is calculated using

$$D^{(t)}(\mathbf{x}^{(t)}) = \frac{1}{N}\sum_{j=1}^{N}\left|\mathbf{x}_j^{(t)} - \mathbf{x}_k^{(t)}\right|$$
$$D^{(t)}(\mathbf{z}_{[i]}^{(t)}) = \frac{1}{N}\sum_{j=1}^{N}\left|\mathbf{z}_{[i],j}^{(t)} - \mathbf{z}_{[i],k}^{(t)}\right| \tag{6.3}$$

where N is the number of agents in the simulation or sheep in observational dataset i and k is the nearest neighbour to agent/sheep j at time t. The time series of the average distance over time for the three observational datasets is shown in Figure 6.3. The average distance for Video 9, $\mathbf{z}_{[9]}$, has the greatest separation with $D^{(t)}(\mathbf{z}_{[9]}) \approx 30$ for all t. The average separation for the other two videos is closer to 25. The variation in this summary statistic is at maximum ±3 pixels, which when you compare to the size of the sheep, which is in the range of 11 to 33 pixel (Section 3.2), is small.

6.4 Determining t_j

Our observational data is from the period of time in the videos where the global alignment, velocity and distance to the nearest neighbour are all approximately constant but our simulations often have a period of time at the beginning of the simulation where this is not true. Therefore we trim our simulations to start as the first of these summary statistics starts to remain constant, t_s, Figure 6.4. This procedure leaves us with almost constant summary statistics over time. As in the previous chapter, Section 5.5 in order to compare the observations to the simulations we rescale time so that they start at $\tau = 0$ and end at $\tau = 1$ for both types of data.

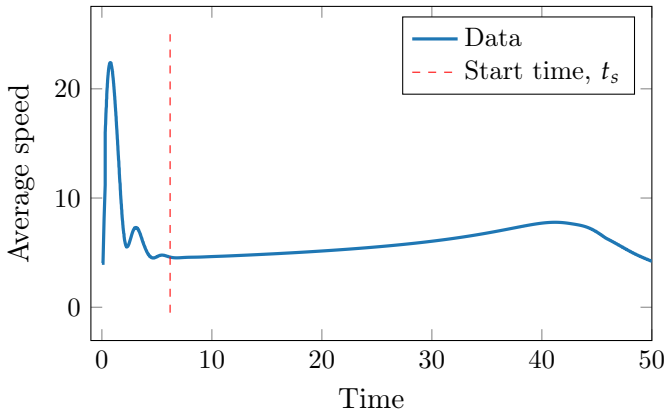


Figure 6.4: The average speed of simulated agents over time, showing the cut off point used to trim simulations. (Simulation of CSM shown in Figure 4.12 with parameters in Table 4.6)

The sampling rate of the observational data and simulated data do not match. The combined sheep model using variable times stepping. This means that in some extreme cases we may end up with simulations with hundreds more samples that out observed datasets. We use linear interpolation to obtain a the value of τ in the simulation for each time point where we have an observation. The time point $\tau = t_j$ is simply the time at which the j^{th} observation was made.

6.5 ABC With The Combined Sheep Model

Similarly to when we were using the Vicsek model modifications, the likelihood function of the combined sheep model is either difficult or not possible to formulate. However, as shown at the end of Chapter 4 simulation from the model is straightforward so therefore we can use ABC to approximate the posterior distribution of the parameters by comparing to our observational data. The ABC rejection scheme follows the procedure set out in Algorithm 6.1.

As before in Chapter 5 we have three summary statistics that will be used to decrease the dimensionality of our observed and simulated data. The summary statistics as described above are: the mean speed, the global alignment and the average distance to the nearest neighbour. For each summary statistic we calculate the distance between observed and simulated using the following equations.

- The distance between the simulated and observed mean speed over all time is defined as

$$\rho_v(\mathbf{x}, \mathbf{z}_{[i]}) = \frac{1}{T_{[i]}} \left| \sum_{j=1}^{T_{[i]}} \left\{ \frac{\langle |\dot{\mathbf{x}}^{(t_j)}| \rangle - \langle |\dot{\mathbf{z}}_{[i]}^{(t_j)}| \rangle}{\langle |\dot{\mathbf{z}}_{[i]}^{(t_j)}| \rangle} \right\} \right| \tag{6.4}$$

  where $T_{[i]}$ is the number of observations in observational dataset i, $\langle |\dot{\mathbf{x}}^{(t)}| \rangle$ and $\langle |\dot{\mathbf{z}}_{[i]}^{(t)}| \rangle$ is the mean speed of the agents in the simulated data and observed data at time t respectively and are calculated using Equation (6.1).

- The distance between simulated and observed global alignment over all time is calculated using

$$\rho_\psi(\mathbf{x}, \mathbf{z}_{[i]}) = \frac{1}{T_{[i]}} \left| \sum_{j=1}^{T_{[i]}} \left\{ \frac{\psi(\mathbf{x}^{(t_j)}) - \psi(\mathbf{z}_{[i]}^{(t_j)})}{\psi(\mathbf{z}_{[i]}^{(t_j)})} \right\} \right| \tag{6.5}$$

  where $T_{[i]}$ is the number of observations in observational data i, $\psi(\mathbf{x}^{(t)})$ and $\psi(\mathbf{z}_{[i]}^{(t)})$ is the global alignment of the simulation and observed data at time t respectively, computed using Equation (6.2).

- The distance between simulated and observed average distance to the nearest neighbour over time is calculated using

$$\rho_D(\mathbf{x}, \mathbf{z}_{[i]}) = \frac{1}{T_{[i]}} \left| \sum_{j=1}^{T_{[i]}} \left\{ \frac{D^{(t)}(\mathbf{x}^{(t_j)}) - D^{(t)}(\mathbf{z}_{[i]}^{(t_j)})}{D^{(t)}(\mathbf{z}_{[i]}^{(t_j)})} \right\} \right| \tag{6.6}$$

  where $D(\mathbf{x}^{(t_j)})$ and $D(\mathbf{z}_{[i]}^{(t_j)})$ is the average distance to the nearest neighbour of the simulation and observed data at time t respectively, Equation (6.3). As seen with the other two distance metrics, $T_{[i]}$ is the number of observations in observational data i.

These distances calculate the difference between the simulated and observed quantity which we then normalise by dividing by the observed quantity. As before in Chapter 5 by normalising in this fashion the three distance metrics are in the same order of magnitude despite the differences in the raw values.

Once all three of the distances between observed and simulated summary statistics are calculated we use the average distance of those calculated in Equations (6.4), (6.5) and (6.6) as the distance used in the ABC rejection scheme when comparing to the i^{th} dataset, $\mathbf{z}_{[i]}$,

$$\rho(\mathbf{x}, \mathbf{z}_{[i]}) = \frac{1}{3} \left(\rho_v(\mathbf{x}, \mathbf{z}_{[i]}) + \rho_\psi(\mathbf{x}, \mathbf{z}_{[i]}) + \rho_D(\mathbf{x}, \mathbf{z}_{[i]}) \right). \tag{6.7}$$

By using the arithmetic mean of the three distance metrics we ensure that all three of the summary statistic distances have to be close in order for the parameter combination $\boldsymbol{\phi}^*$ to be accepted.

6.6 Initialising The Simulations

The initialisation of the simulations is vital to being able to create simulations which are comparable to the observed data. We use the predator in the combined sheep model to mirror the stimulus the sheep in the data videos received from the quadbike, but limiting the speed it can go at so that it cannot catch up with the simulated sheep.

6.6.1 Initialising Agents

The initial (x, y)-locations of the agents in the simulations are randomly allocated by sampling from a uniform distribution. In this ABC scheme we use a $\mathcal{U}\left(\bar{\mathbf{z}}_{[i]}^{(t_0)}, \bar{\mathbf{z}}_{[i]}^{(t_0)} + \mathbf{140}\right)$ distribution, where $\bar{\mathbf{z}}_{[i]}^{(t_0)}$ is the average starting location of sheep in the observed data in

dataset i and $\mathbf{140} = (140, 140)$. The width of this distribution is chosen to minimises the amount of burn in time as the agents in this model arrange themselves to minimise the forces acting upon them.

We use the distribution $\mathcal{U}(-\pi, \pi)$ as the prior distribution for the initial directions. We can allocate the initial angles randomly like this since the simulation will start by ordering the agents into a regular grid like pattern in order to balance the forces acting upon the agents, so the initial directions are irrelevant to the overall behaviour of the agents.

The initial speeds of the agents are sampled from a normal distribution. The mean and standard deviation of this distribution are taken from the observed data at time 0. The normal distribution used as the prior for the initial speed when comparing to Video 9 is $\mathcal{N}(4.07, 0.48)$.

6.6.2 Initialising The Predator

The initial location of the predator in the combined sheep model simulations is taken to be the location of the quadbike in the observational dataset at time t_0. The direction of predator at the beginning of the simulation is initialised to point directly towards the prey agents. The initial direction of the predator does not affect the behaviour of the simulation as over the first few time steps the equation of motion will determine the direction the predator will travel. By initialising the direction in approximately the same direction as the direction determined by the equation of motion we minimise any burn in time.

The speed the predator moves at is initialised to be a separate realisation from the same normal distribution used for the prey agents, which for Video 9 is $\mathcal{N}(4.07, 0.48)$. Since we do not have an active predator in our data videos, we do not want the predator in our simulations to be able to catch up with the prey agents. Hence we cap the speed at which the predator can travel to be the mean speed of the prey agents.

6.6.3 The Size And Mass Of The Agents And Predator

The agents in the original Helbing model, and therefore also in our combined sheep model, are no longer point like particles, they have physical size and mass. Each agent is assumed to be circular with radius r_i. The individual radii for the prey agents and the predator are sampled from a $\mathcal{U}(0.5, 0.7)$ distribution. The mass of the agents is assumed to come from a $\mathcal{U}(30kg, 80kg)$ distribution which from conversations with farmer and scientific literature is appropriate [122]. The mass of the quadbike is considerably higher with a distribution of $\mathcal{U}(350kg, 550kg)$ [123].

Parameter	Value
$A_j,\ j \in [0,\ldots,N]$	10,000
$B_j,\ j \in [0,\ldots,N]$	10
K_1	120,000
K_2	240,000

Table 6.2: The parameter values which define the characteristics of the walls in the Helbing model [114] and the combined sheep model.

6.6.4 The Walls

The parameters which define the walls can be seen in Table 6.2. The values of K_1 and K_2 remain unchanged from the original Helbing paper. The values of A_j and B_j have been chosen to keep the minimum distance between an agent to the wall in the same order of magnitude in our observations. The walls in our simulations are defined to be in the same locations as the fences we detected in the drone videos, Section 2.9.3. When comparing to Video 9 this method only returns $K = 5$ fences, $K = 19$ when comparing to Video 10 and $K = 12$ when comparing to Video 11.

6.6.5 Remaining Parameters

Before describing the marginal prior distribution for the parameter of the combined sheep model. We start by recapping the equations of motion for the combined sheep model. As seen in Chapter 4, these are

$$\begin{aligned}\frac{\mathrm{d}^2\mathbf{x}_i^{(t)}}{\mathrm{d}t} &= -\mu_i\frac{\mathrm{d}\mathbf{x}_i^{(t)}}{\mathrm{d}t} + \frac{1}{N}\sum_{i=1,i\neq j}^{N} F\left(|\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\right) + G\left(|\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}\right) \\ &\quad + \frac{1}{m_i}\sum_{k=0}^{K}\mathbf{f}_{i,w_k}, \\ \frac{\mathrm{d}^2\mathbf{p}^{(t)}}{\mathrm{d}t} &= -\frac{\mathrm{d}\mathbf{p}^{(t)}}{\mathrm{d}t} + +\frac{q}{N}\sum_{i=1}^{N} H\left(|\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}|\right)\left(\mathbf{x}_i^{(t)} - \mathbf{p}^{(t)}\right) + \frac{1}{m_0}\sum_{k=0}^{K}\mathbf{f}_{i,w_k},\end{aligned}$$

where

$$F(r) = \frac{c}{r^2} - a, \quad G(r) = \frac{b}{r^2} \quad \text{and} \quad H(r) = \frac{1}{r^s + r}.$$

The equation calculating the force between an agent an a wall is

$$\mathbf{f}_{iw} = A_i\mathbf{n}_{iw}\exp\left\{\frac{d_{iw}}{B_i}\right\}. \tag{6.8}$$

Parameter	Lower bound	Upper bound
μ_i	0	5
a	0	10
b	5,00	12,000
c	20,000	120,000

Table 6.3: The upper and lower bound for the uniform prior distributions of parameters using in comparisons between the combined sheep model and observed data.

We can see that there are six remaining parameters of the combined sheep model. Two of these parameters determine how the predator reacts to the prey, q and s. Both of these parameters aid the predator to catch up with the prey, so since we do not require this in our simulations for comparison with the observed datasets we fix these parameters to be $q = 12,000$ and $s = 2.5$ as these parameters allowed the predator to change direction swiftly to react to the prey.

Therefore we are left with only four parameters. These are sampled from a Latin hypercube, Section A.7, with marginal prior distributions for each of the parameters shown in Table 6.3. Using the parameter value we found to work we set the lower bound of our initial prior distribution to be the value round down an order of magnitude and the upper bound to be the parameter value rounded up an order of magnitude. Then after running 1,000 simulations using these prior distributions we were able to modify these ranges to give the values shown in Table 6.3.

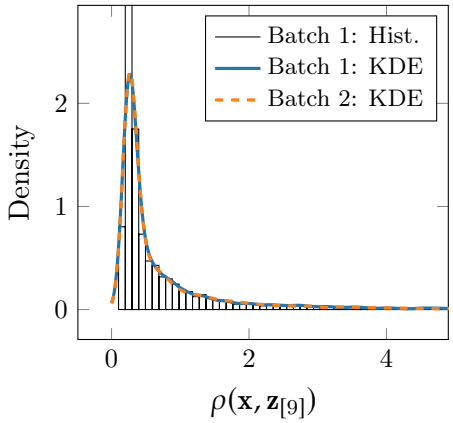


Figure 6.5: The distributions of $\rho(\mathbf{x}, \mathbf{z}_{[9]})$ values when comparing to $\mathbf{z}_{[9]}$.

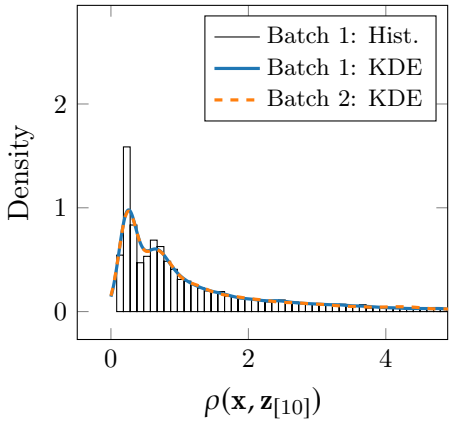


Figure 6.6: The distributions of $\rho(\mathbf{x}, \mathbf{z}_{[10]})$ values when comparing to $\mathbf{z}_{[10]}$.

6.7 Comparing Simulations To Observational Data

By comparing the simulations to the observational data, $\mathbf{z}_{[i]}$ where $i \in [9, 10, 11]$, we obtain a distribution of $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values. For each observational dataset we run two batches of the ABC algorithm each of which with 10,000 simulations. By running two batches of the ABC scheme, Algorithm 6.1, we are able to see whether we are running a sufficient number of simulations to get a good idea of what the true distribution of $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ is.

The curve shown in Figure 6.5 is the resulting distributions of $\rho(\mathbf{x}, \mathbf{z}_{[9]})$ values obtained by comparing the two batches of simulations to Video 9 where $\rho(\mathbf{x}, \mathbf{z}_{[9]}) < 5$. When comparing to this observational dataset the $\rho(\mathbf{x}, \mathbf{z}_{[9]})$ values lie in the range 0.08 – 48, but 95% of simulations result in a $\rho(\mathbf{x}, \mathbf{z}_{[9]})$ value less than 5. The second run when completed follows the same shape showing that we are running sufficient simulations.

For the comparison with the observational dataset from Video 10 we see a very similar shape, Figure 6.6. Just like the previous study, when comparing to Video 9, we see that the second run of the ABC rejection scheme produced a very similar distribution of $\rho(\mathbf{x}, \mathbf{z}_{[9]}) > 5$. In this comparison only 88% of the simulations lie in the range 0–5, hence the maximum density achieved by the $\rho(\mathbf{x}, \mathbf{z}_{[9]})$ distribution is smaller than before when comparing to Video 9.

When comparing with the observational dataset $\mathbf{z}_{[11]}$, we see the shape of the distribution $\rho(\mathbf{x}, \mathbf{z}_{[11]})$ is the same of the distributions for $\rho(\mathbf{x}, \mathbf{z}_{[9]})$ and $\rho(\mathbf{x}, \mathbf{z}_{[10]})$, Figure 6.7. This comparison results over 95% of the simulations with a $\rho(\mathbf{x}, \mathbf{z}_{[11]})$ value in the range 0–5.

We can see that all three distributions of $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ $i \in [9, 10, 11]$, have the same shape, the peak at 0.5 with a long upper tail.

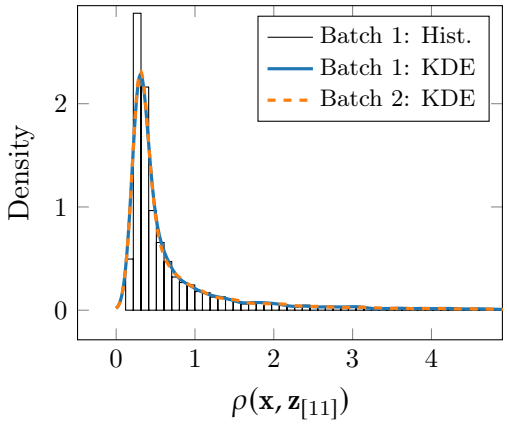
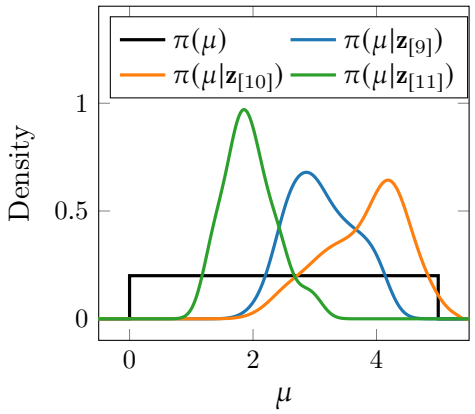


Figure 6.7: The distributions of $\rho(\mathbf{x}, \mathbf{z}_{[11]})$ values when comparing the combined sheep model simulations with observational data from Video 10.

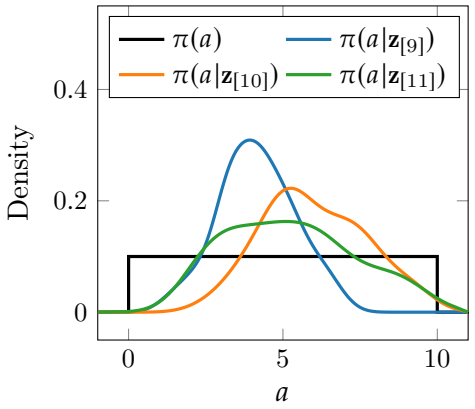
6.7.1 Posterior Distributions

Since our combined sheep model has four parameters, $\phi = (\mu, a, b, c)$, which we are want to infer values from the data we are unable to visualise the joint posterior distributions as it resides in 4-dimensional space. Hence, we have to look at each marginal posterior distribution individually. Figure 6.8 shows the individual marginal posterior distributions of the parameters, μ, a, b and c.

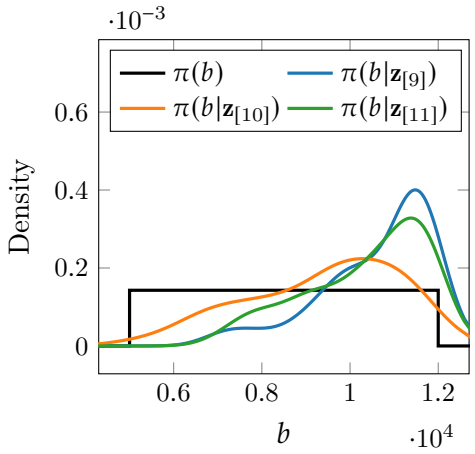
Starting with the posterior distributions for μ we can see that each observational dataset produced different distributions. This variance could be down to the value of μ being correlated to one of more of the other parameters. Starting with the comparison with $\mathbf{z}_{[9]}$ the marginal posterior distribution, $\pi(\mu|\mathbf{z}_{[9]})$, peaks at approximately $\mu = 3$, with the other distributions peaking at 4.5 and 1.5 for comparisons with $\mathbf{z}_{[10]}$ and $\mathbf{z}_{[11]}$ respectively.



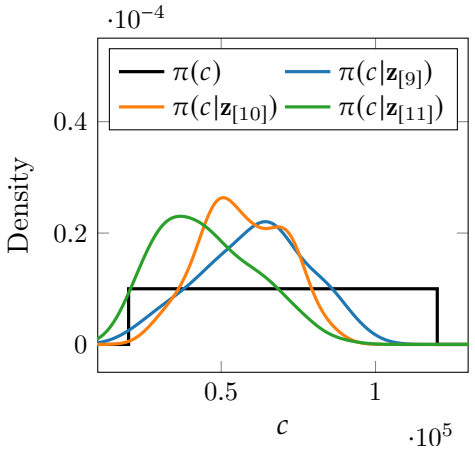
(a) Marginal prior/posterior distributions for μ.



(b) Marginal prior/posterior distributions for a.



(c) Marginal prior/posterior distributions for b.



(d) Marginal prior/posterior distributions for c.

Figure 6.8: The prior and posterior distributions for the parameters $\phi = (\mu, a, b, c)$ in the combined sheep model.

The parameter μ represents the coefficient of friction acting on the individuals, in the papers by Chen *et al.* and by Helbing *et al.* this parameter was fixed at $\mu = 1$ [45, 114] whereas our data comparisons show that the coefficient of friction of the sheep we observed is likely to be higher than this.

When looking at the posterior distributions for a we see the data observations agreeing closer than we did with μ. All three marginal posterior distributions peak in the middle of the original range of 0–10, for $\rho(\mathbf{x}, \mathbf{z}_{[9]})$ this peak is at 3.5 whereas for $\rho(\mathbf{x}, \mathbf{z}_{[10]})$ and $\rho(\mathbf{x}, \mathbf{z}_{[11]})$ it is at 5 but $\pi(11|\mathbf{z}_{[i]})$s much flatter than that of $\pi(10|\mathbf{z}_{[.]})$ In the Chen *et al.* acceleration model the parameter was set to $a = 1$ [45] when we compare to our observations this value of a seems unlikely.

The marginal posterior distributions for the parameter b has a nice well defined peak at $b \approx$ 11,000. When comparing to $\mathbf{z}_{[10]}$ the marginal posterior distribution for b has a broader peak spreading from 9,000 to 11,000. However, the b value with highest density is still at approximately the same value as the other two marginal posterior distributions. The parameter b determines the strength of the force repelling the agents from the predator, in our observations the quadbike is the predator. As a result of b controlling the strength of the repulsion force the size of b is critical in determining the speed at which the agents move at.

In the prior distribution for c, $\pi(c)$, the parameter varied from 20,000 up to 120,000. After the comparison with the datasets we see that the value of c is more likely to lie in that range 30,000 to 75,000. All three marginal posterior distributions are skewed towards smaller values of c giving a slight peak in density around 60,000 for $\pi(c|\mathbf{z}_{[9]})$ and $\pi(c|\mathbf{z}_{[10]})$ and around 35,000 for $\pi(c|\mathbf{z}_{[11]})$. The parameter c determines the size of the repulsion force between agents and as a result of that the average distance between an agent and its nearest neighbour. Since these distributions are board we would expect that this parameter must be correlated to one or more of the other parameter values.

	a	b	c
μ	0.26 (± 0.26)	0.68 (± 0.09)	0.47 (± 0.25)
a		0.31 (± 0.13)	0.83 (± 0.06)
b			0.45 (± 0.15)

Table 6.4: The average Pearson correlation coefficient (± one standard deviation) for the parameters μ, a, b and c across the three model comparisons.

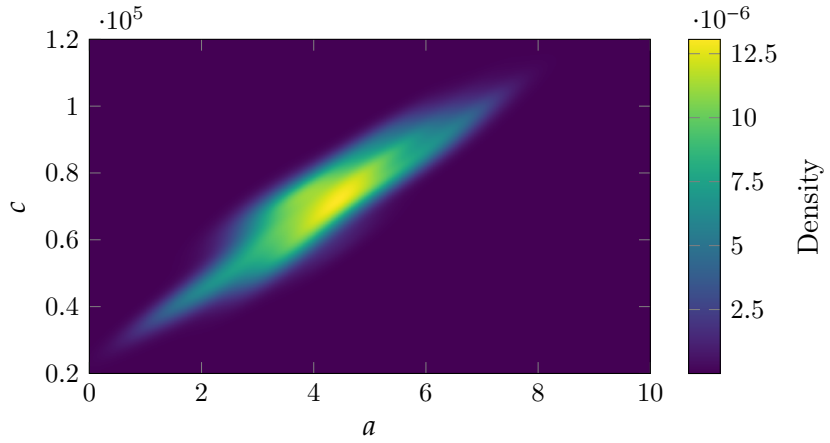


Figure 6.9: The joint marginal posterior distribution for parameters a and c when comparing with $\mathbf{z}_{[9]}$.

6.8 Correlation Between Parameters

In the previous section we noted that there could be some correlation between parameters since the marginal posterior distributions for a and c were so broad. Therefore we calculated the pairwise Pearson correlation coefficient, Section A.6, for all four parameters. The resulting average Pearson correlation coefficient values can be seen in Table 6.4.

The pair with the largest average Pearson correlation coefficient is a and c where $r_{ac} = 0.83$. Hence the parameters a and c are highly correlated. The joint marginal posterior distribution for these parameters, Figure 6.9, shows that the there is a peak in the distribution around the point $(a, c) \approx (4.5, 0.7 \times 10^5)$.

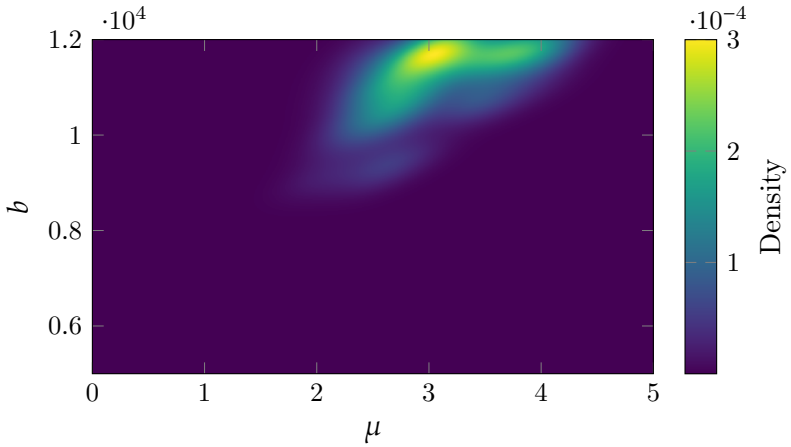


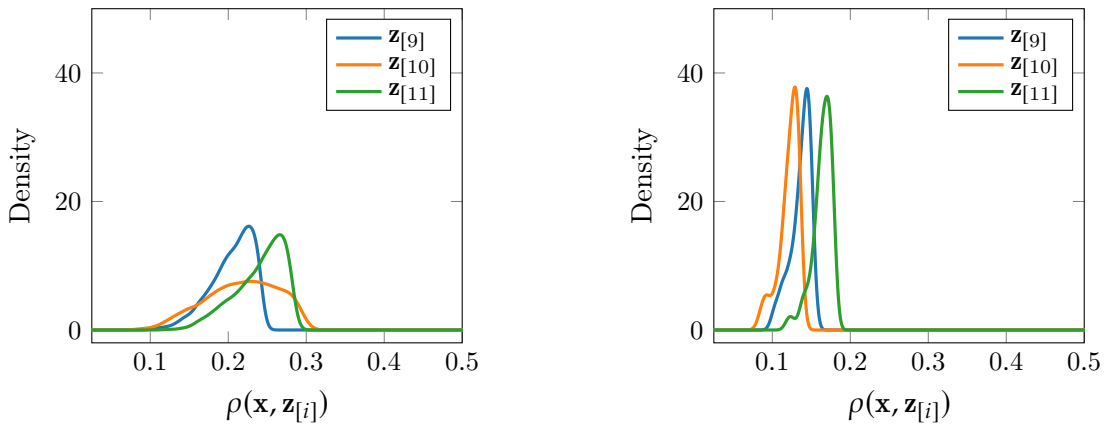
Figure 6.10: The joint marginal posterior distribution for parameters μ and b when comparing with $\mathbf{z}_{[9]}$.

The other pair of parameters that are slightly correlated is (μ, b) with a Pearson correlation coefficient of $r_{\mu b} = 0.68$. The joint marginal posterior distribution for this pair are shown in Figure 6.10. The joint marginal posterior distribution for μ and b has a small region of high density ($2 < \mu < 4$ and $1 \times 10^4 < b < 1.2 \times 10^4$) with a peak at $\mu \approx 3$ and $b \approx 1.15 \times 10^4$ which matches up with the individual marginal posterior distributions seen earlier, Figure 6.8a and 6.8c. From the joint marginal distribution we can now see that there is a point of inflection in the distribution before it peaks.

6.9 Comparing The Fit Across The Videos

To compare the fit of the combined sheep model to the observational data we can look at the $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values for the best simulations which result in the smallest 20% and smallest 1% of the $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values for $i \in [9, 10, 11]$, Figure 6.11.

When only 20% of the simulations are considered for all three of the comparisons the $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ distribution only contains values of $\rho(\mathbf{x}, \mathbf{z}_{[i]}) < 0.4$. The three distributions when comparing to the observational datasets, $\mathbf{z}_{[i]}$ where $i \in \{9, 10, 11\}$, produce similar $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ distributions, with values ranging from 0.1 to 0.3. The distributions for $\rho(\mathbf{x}, \mathbf{z}_{[9]})$ and $\rho(\mathbf{x}, \mathbf{z}_{[11]})$ produce distributions with a well defined peak, at 0.225 and 0.275 respectively whereas the distribution for $\rho(\mathbf{x}, \mathbf{z}_{[10]})$ is much flatter with no obvious peak.Considering the peaks are within 0.15 of each other and that the original distributions of $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ extended further than the range 0–5 we can say that these distributions are very similar.



(a) The resulting distribution of $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values using only the 20% of simulations with the smallest $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values.

(b) The resulting distribution of $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values using only the using only the 1% of simulations with the smallest $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values.

Figure 6.11: The distribution of $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ using only the 20% or 1% of simulations with the smallest $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values when compared to all three observational datasets.

When only using the 1% of simulations with the smallest $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values to create the distribution the all three distributions have a well defined peaks and are only separated by approximately 0.1. They are now at $\rho(\mathbf{x}, \mathbf{z}_{[9]}) \approx 0.15$, $\rho(\mathbf{x}, \mathbf{z}_{[10]}) \approx 0.125$ and for the final observational dataset comparison the peak is $\rho(\mathbf{x}, \mathbf{z}_{[11]}) \approx 0.175$. The $\rho(\mathbf{x}, \mathbf{z}_{[11]})$ distribution for the 1% of simulations with the smallest $\rho(\mathbf{x}, \mathbf{z}_{[11]})$ values may result in larger values due to the fact that observational dataset was the smallest, with only 100 observations, Table 6.1. We could argue then that a dataset with observation at only 100 time points is not long enough to be able to closely fit a mathematical model to.

6.10 Analysing $\rho(\mathbf{x}, \mathbf{z}_{[i]})$

As in the previous Chapter we can break down $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ into its constituent parts, ρ_v, ρ_ψ and ρ_D. However, unlike in the previous Chapter we do not see that the three part contribute to the overall value of $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ equally, Figure 6.12. When comparing the combined sheep model to our observational datasets, $\mathbf{z}_{[9]}$ $\mathbf{z}_{[10]}$ and $\mathbf{z}_{[11]}$, the distance between the observed alignment and the simulated alignment, ρ_ψ, becomes the most influential summary statistic distance. The value of ρ_ψ is approximately two thirds of $\rho(\mathbf{x}, \mathbf{z}_{[i]})$.

The summary statistic distance ρ_v averaged across the three data comparisons makes up the smallest proportion of $\rho(\mathbf{x}, \mathbf{z}_{[i]})$, approximately 1%. The third summary statistic, ρ_D, takes up the rest of the remaining third. Showing that the distance between agents is more influential to the fit of the model than the speed they travel at.

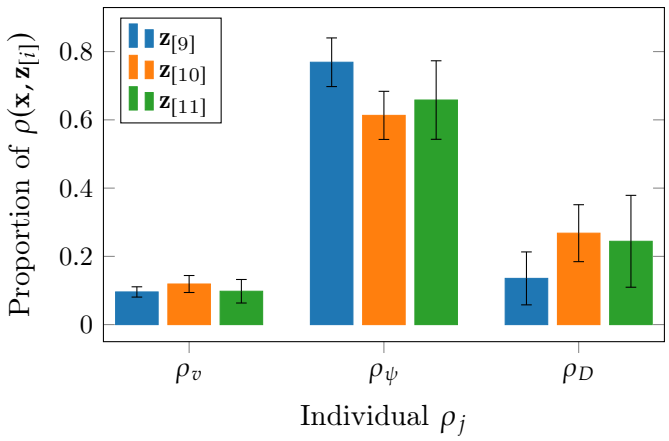


Figure 6.12: Proportion of $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ from each ρ_j with standard deviation shown in the error bars.

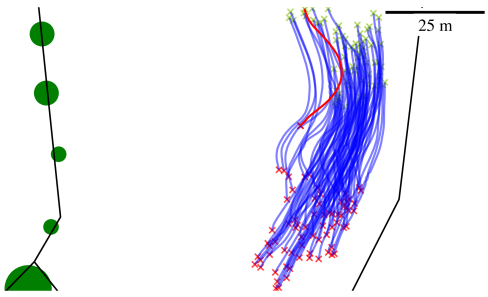
		Parameter Value			
Video i	$\rho(\mathbf{x}, \mathbf{z}_{[i]})$	μ	a	b	c
9	0.24	4.16	4.15	1.01×10^4	9.28×10^4
10	0.28	4.87	3.99	0.71×10^4	5.36×10^4
11	0.24	2.95	2.35	0.93×10^4	2.49×10^4

Table 6.5: The $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ value for the simulation chosen to be compared to the observed trajectories, with the parameter values which created it.

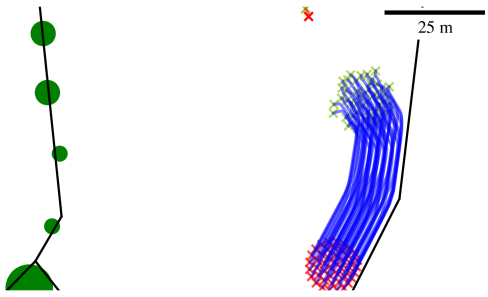
6.11 Comparing The Observed And Simulated Trajectories

Since the initialisation of the simulations was very dependent on the initial locations of the sheep in the observational datasets we can also see how well the model recreates our data by comparing the trajectories of the prey agents in a simulation to the observed trajectories in the observational datasets.

To decide which simulation's trajectories to compare to the observed trajectory we look at the joint marginal posterior plots, for comparison with Video 9 these are shown in Figures 6.9, and 6.10. For this video we see that there is a peak in the density of the posterior distribution at around $\mu \approx 3, a \approx 4.5, b \approx 1.15 \times 10^4$ and $c \approx 0.65 \times 10^5$. So therefore for comparison with Video 9 we select the simulation with parameters close to this peak value with minimal $\rho(\mathbf{x}, \mathbf{z}_{[9]})$ value. The $\rho(\mathbf{x}, \mathbf{z}_{[i]})$ values for the simulations selected for comparison with their respective parameter values (to 3 significant figures) are displayed in Table 6.5.

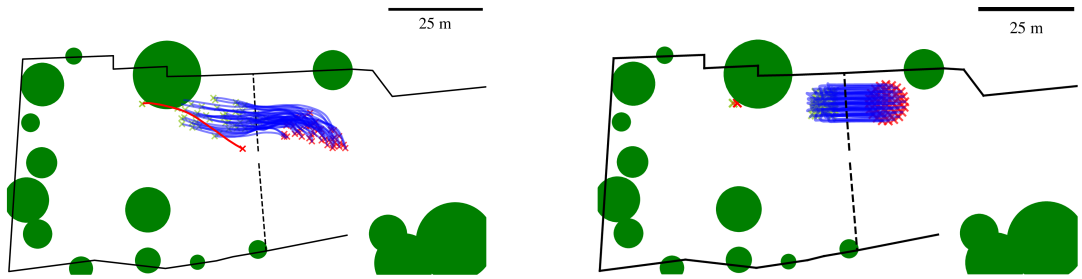


(a) Observed trajectories for Video 9. The initial positions of the agents in the simualtion are taken from the 80^{th} frame of the video. This is where the quadbike is visible in the observations.



(b) Simulated trajectories using optimal parameters taken from the approximate posterior distribution when compared to Video 9.

Figure 6.13: The observed and simulated trajectories for Video 9. The black lines show the fences, the green circles show the approximate size and location of any trees or bushes in the field. The sheep trajectories are shown by a solid blue line, with the direction of travel from the green cross to the red cross. The quadbike's trajectory is shown by a solid red line with the direction of travel similarly denoted as from the green to red cross.



(a) Observed trajectories for Video 10. The dashed line shows a fence where the sheep can go underneath, but the quadbike has to go around.

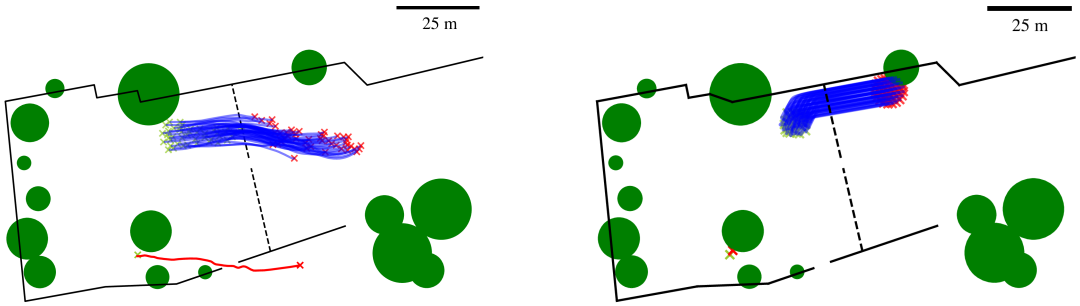
(b) Simulated trajectories using optimal parameters taken from the approximate posterior distribution when compared to Video 10.

Figure 6.14: The observed and simulated trajectories for Video 10. The black lines show the fences, the green circles show the approximate size and location of any trees or bushes in the field. The sheep trajectories are shown by a solid blue line, with the direction of travel from the green cross to the red cross. The quadbike's trajectory is shown by a solid red line with the direction of travel similarly denoted as from the green to red cross.

Figure 6.13 shows the observed trajectories from Video 9, and the simulated trajectories using the parameters shown in Table 6.5. Figure 6.14 shows the same but for Video 10, and Figure 6.15 for Video 11. We can see that for Videos 9 and 10, the simulated agents follow very similar path to the sheep in the observed trajectories. For Video 11, we see the simulated agents deviate from this by travelling further up the field than was seen in the observations. This could be down to the predator/quadbike being further down the field from the initial positions of the sheep and hence the agents would feel a push up the field towards the walls. This is different from the observed behaviour of the sheep who still move across the field. This could be down to the sheep knowing where they should do when being herded in this field.

In further analysis we could include a new summary statistic in the ABC algorithm to capture similarity in the paths. The new summary statistic would be the centre of mass of the flock at each point in time. This comparison of trajectories between observed and simulated was not possible when comparing the Vicsek model modification simulations to observations in Chapter 5 as the simulated trajectories were too distinct from the observed trajectories and comparison using the centre of mass would give meaningless results.

The trajectories from our three simulations have one clear difference to those from our observations. The simulated agents move as one collective group where no individual moves out of group formation after the initial burn in time. The end of this disorganised burn-in time can be seen in Figure 6.15b. Whereas in the observations we made the sheep mix and the distance between sheep regularly changes as they travel. If there was more noise in the model, either by adding a noise term to the direction of travel, or allowing the sizes of the agents to vary more, there could be a less rigid formation to the simulated flock.



(a) Observed trajectories for Video 11. Only the first 100 out of the 192 frames tracked were used for simulation comparison.

(b) Simulated trajectories using optimal parameters taken from the approximate posterior distribution when compared to Video 11.

Figure 6.15: The observed and simulated trajectories for Video 11. The black lines show the fences, the green circles show the approximate size and location of any trees or bushes in the field. The sheep trajectories are shown by a solid blue line, with the direction of travel from the green cross to the red cross. The quadbike's trajectory is shown by a solid red line with the direction of travel similarly denoted as from the green to red cross.

In Figure 6.13b we see that the simulated agents also get much closer to the walls than the sheep did to the fences in our observations. This could be rectified by tweaking the parameters in Table 6.2. However, overall the interaction between the walls and the simulated agents seem reasonable, as they never pass through a wall or get so close to encounter repulsion forces to relocate an agent from one side of the flock to the other.

6.12 Conclusions

Using an approximate Bayesian computation rejection scheme we were able to calculate the approximate joint posterior distributions for the parameters in the combined sheep model for each of the three observational datasets. These approximate posterior distributions characterise the behaviour seen in our observational datasets, $\mathbf{z}_{[i]}$ $i \in \{9, 10, 11\}$.

The combined sheep model has four parameters which we have calculated the approximate posterior distribution of, μ, a, b and c. The marginals of these distributions can be seen in Figure 6.8. When comparing to all three data videos the approximate marginal distribution for μ, $\pi(\mu|\mathbf{z}_{[i]})$ $i \in \{9, 10, 11\}$, we see that all three distributions have a well defined peak between $\mu = 1.5$ and $\mu = 4.5$. The marginal posterior distributions for μ show the most variation across the three videos. The approximate marginal distributions for a, b and c are similar across observational datasets, with the approximate marginal posterior distribution for a peaking at $3 < a < 5$, the approximate marginal posterior distribution for b peaking at $b \approx 1.15 \times 10^4$ and the approximate marginal distributions for c peaking at values between 0.25×10^5 and 0.75×10^5. Given that three of the four marginal posterior distributions are quite similar it follows that the parameter a is more sensitive to the

differences in the observed behaviours.

In the final section of this chapter we looked at how observed trajectories and the simulated trajectories of the combined sheep model using parameters shown in Table 6.5 compared visually. We found that overall the simulated sheep did have visually similar paths. However, there was more mixing among the sheep within the flock in the collected trajectories. So therefore in the future we would look into how behaviour changes if at each time step the parameters were re-sampled from the approximate posterior distribution, $\pi(\phi|\mathbf{z}_{[i]})$. We would also look into how adding a small amount of noise to the direction of travel would effect the rigid structure of the simulated flock.

IV

Conclusions And Appendix

7

Summary

Using computer vision techniques we devised an algorithm that was able to identify and individually track over forty sheep across 14 data videos showing sheep being herd across a field by a farmer. We extended computer vision methods from the literature to include novel processing steps which improved the precision of the determined agent locations, and improved prediction of trajectory paths. We introduced several models of collective animal motion incorporating different physical behaviours observed in real collective motion. We obtained feasible parameters for all our models of collective motion using parameter inference with our observational data.

In Part II we describe in detail how we performed our own data collection by travelling to a farm in North Yorkshire on a number of occasions to film sheep being herded using a camera mounted on the underside of a drone. Using our computer vision algorithm we were able to acquire a complete trajectory for each individual and from these we were able to examine quantities such as individual speeds and global flock alignment. Of the 14 videos that were suitable for data collection the longest continuous trajectories obtain were 593 frames in size, equivalent to approximately 25 seconds of collective animal motion for 45 sheep.

In Part III we introduced mathematical models for collective motion from the literature. We proposed four modified versions of the Vicsek model [38] for collective motion, making it appropriate for comparison with the observational data. We found that the Ginelli model for collective motion in herds of sheep [56] was unable to reproduce behaviour captured in our data videos and so was unsuitable for comparison with our collected data.

The final two models we discussed (Chen and Kolokolnikov [45] and Helbing et al. [114]) were also unsuitable for direct comparison with our collected data individually, as neither fully captured the behaviour observed. However, we proposed a new combined model where simulated agents interacted with both a predator and hard boundaries to produce features similar to those in the observational data.

Using periods of time showing emergent flocking behaviour, where quantities such as speed and alignment were increasing, we compared the observational data to simulation using

an ABC rejection scheme to determine the approximate joint posterior distribution of the parameters in a family of point-like agent-based models based on the Vicsek model.

Similarly using different time periods where flocking was in a "steady-state", that is where the alignment and speed are approximately constant, we compared the observational data to simulated data to produce an approximate joint posterior distribution for the parameters in our new combined sheep model, which takes into account the physical size of the sheep and surrounding obstacles such as fences and walls.

We think that the modified Vicsek models and the combined sheep model created here would be interesting not only to agriculturalists, but also to anyone who has to deal with large numbers of people. We may have created our models with sheep in mind, but that does not limit its uses. For example the combined sheep model could be used to determine the path people would use to leave a packed stadium after a concert, or where people will naturally gravitate towards in a shopping centre.

In addition to the direct collective motion applications we believe our computer vision algorithm on its own could be interesting to farmers or park rangers who often have to go out and count up their livestock or animals. A perfected version of our computer vision model has the potential to help these people so that they would only need to know how to fly a drone and it could survey their land, autonomously counting the number of the animal they were interested in, whether that be sheep or cattle or elephant or even endangered species.

The same perfected computer vision algorithm would have wider applications to anyone using, developing or interested in image segmentation software. For example on a much smaller scale our methods could be used to count biological entities such as cells as they move through a biological fluid. The methods can be easily extended to include an increasing number of agents allowing you to track cell multiplication over time which currently often has to be done by hand [124].

7.1 Future Work

To improve the computer vision algorithm to locate and track sheep in drone footage we could collect footage of other animals being herded in different environments, such as wildebeest herding in the Serengeti National Park. In this situation there is no longer a bright white object of interest against a green background, which we have currently not tested due to a lack of footage of such a style. Ensuring the algorithm copes with, or enabling the algorithm to cope with, footage like this would allow for more varied datasets to examined.

Additional datasets that show different flocking behaviours would enable a more in depth investigation in to the final Vicsek modification model. One dataset that would be particularly interesting would be one where the sheep start from stationary and then start to move at a slow speed. Doing parameter inference on this dataset for the final Vicsek modification would give us insight into the value of the parameters change when β, the parameter that determines how much an individual is influenced by its surroundings when moving at speed, plays a smaller role.

For both the Vicsek models and the CSM model datasets with more agents would be interesting to look at and analyse. We were limited to under 100 sheep in our observations due to the increased computing power needed to track larger flocks. It would be interesting to see if the inferred parameters remain the same when the flock is double or ten times the size. Also, for both models it would be interesting to see if different animals produce similar model parameters and if not, which parameters differ between species and by how much.

An additional dataset that would be more useful for inference with the combined sheep model would be one where the sheep change direction while being herded, but not directly influenced by the fences in the field. Similarly, it could be studied to see if sheep retain knowledge about the features of fields that they are used to. It would be interesting to see which parameters remain the same if the sheep are being herded in locations that they know compared to locations they have never seen before. As a casual observation while on the farm, we noticed that sometimes the behaviour of the sheep changed if the flock was herded for the second time in one day.

Other datasets that would be interesting would be ones where the sheep are in a much smaller enclosure. Does there need to be a change in the model when this is the case or is it just the parameters that change? Similarly when multiple predators are introduced or when two of more flocks are herded together, which parameters change under these situations?

A

Numerical Methods And Mathematical Definitions

A.1 Introduction

To aid with the flow of the thesis in this chapter we will describe a number of numerical time stepping algorithms that are used in future chapters when simulating from the mathematical models for collective motion. After describing each of these time stepping methods we also present its derivation.

This chapter also outlines a number of mathematical properties that are used when forming the mathematical models of collective motion or when analysing the output from these models.

A.2 Numerical Time Stepping Methods

There are many standard numerical time stepping methods used to approximate the numerical solution of ordinary differential equations. In this section we will go through just a small number of these, as these methods have been used in our model simulations in Chapters 4, 5 and 6. When implemented all methods have been coded from scratch, no in built ODE solvers have been used. Conceptually these methods allow you to start from an initial point and then take a short step forward in time to find the next solution point. We can then continue to take steps until a solution is found. These methods can be single-step which only refer to the previous point in time, like Euler's method in Section A.2.1, or can refer to many previous points such as four-step Adams-Bashforth, Section A.2.2. Multistep methods increase the accuracy of the solution from $O(h^2)$ to $O(h^{n+1})$, where h is the size of the step taken. There are even families of methods, such as the Runge-Kutta methods, which calculate intermediate points to increase accuracy of the solutions found.

A.2.1 Euler's Method

The Euler time stepping method is a first-order linear method for solving ordinary differential equations with a given initial value. It is the most basic explicit method for solving ordinary differential equations and the simplest Runge-Kutta method. The Euler method can be written as

$$y\left(t_{n+1}\right) = y\left(t_n\right) + hf\left(t_n, y\left(t_n\right)\right) \tag{A.1}$$

where h is the size of time step and $f(t_n, y(t_n))$ is the differential of y at time t_n. This method is accurate to order h^2. The Euler method is consistent, meaning that the local truncation error goes to zero quicker than the size of the time step, h, goes to zero. Unfortunately, the Euler method is unstable, meaning that the numerical solution can grow very large for ordinary differential equations where the exact solution does not. Therefore, the stability of Euler's method can determine whether or not it is appropriate to use for numerically integrating a specific ordinary differential equation. This can be seen in Section A.2.6.

Derivation Of Euler's Method

Euler's method can be derived in many ways. One possibility is to describe the problem geometrically. However, here we prove Euler's method by looking at the Taylor expansion of $y(t_{n+1})$ at t_n up to the terms of order h:

$$y\left(t_{n+1}\right) = y\left(t_n\right) + \left(t_{n+1} - t_n\right) y'\left(t_n\right) + O\left(h^2\right) \tag{A.2}$$

where h is defined to be the difference between the time at t_{n+1} and t_n. Hence since $f(t_n, y(t_n))$ is the differential of y at time t_n we have Euler's method

$$y\left(t_{n+1}\right) = y\left(t_n\right) + hf\left(t_n, y\left(t_n\right)\right).$$

A.2.2 Four-Step Adams-Bashforth

Four-step Adams-Bashforth is a linear multistep method used to calculate approximate numerical solutions of initial value problems of ordinary differential equations. Multistep methods use information from previous steps rather than discarding it, which methods such as n-step Adams-Bashforth and n^{th} order Runge-Kutta do (where n is greater than 1). Linear multistep methods refer to a linear combination of several previous points and derivative values. The Adams-Bashforth methods were designed by John Couch Adams to solve a differential equation modelling capillary action derived by Francis Bashforth. They published Bashforth's theory and Adams' numerical methods in 1883 [125]. The

linear multistep method we refer to in this section refers back four steps in time. Euler's method in the previous section, Section A.2.1, is a one-step method.

The Adams-Bashforth methods are explicit methods, that is to say that the methods can be used directly to compute $y(t_{n+s})$ without having to know the derivative at that point in time, $y'(t_{n+s}, y(t_{n+s}))$. To simplify the notation we set $y' = f(t, y(t))$ and $y_i = y(t_i)$ where $t_i = t_0 + ih$. The variable h is the time step size and i is an integer. The four-step Adams-Bashforth method is

$$y_{n+4} = y_{n+3} + \frac{h}{24}\left(55f\left(t_{n+3}, y_{n+3}\right) - 59f\left(t_{n+2}, y_{n+2}\right) + 37f\left(t_{n+1}, y_{n+1}\right) - 9f\left(t_n, y_n\right)\right). \tag{A.3}$$

This method needs four values, $y_{n+3}, y_{n+2}, y_{n+1}$ and y_n, to compute the next value, y_{n+4}. However, the initial value problem only provides the one value y_0. Therefore, we use the approximate value of y_1 calculated by Euler's method (which is also the one-step Adams-Bashforth), y_2 calculated but the two-step Adams-Bashforth, and y_3 calculated by the three-step Adams-Bashforth. The one to three-step methods are:

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{A.4}$$

$$y_{n+2} = y_{n+1} + \frac{h}{2}\left(3f\left(t_{n+1}, y_{n+1}\right) - f\left(t_n, y_n\right)\right) \tag{A.5}$$

$$y_{n+3} = y_{n+2} + \frac{h}{12}\left(23f\left(t_{n+2}, y_{n+2}\right) - 16f\left(t_{n+1}, y_{n+1}\right) + 5f\left(t_n, y_n\right)\right) \tag{A.6}$$

Derivation Of Four-Step Adams-Bashforth

The general form of an s-step Adams-Bashforth method is

$$y_{n+s} = y_{n+s-1} + h\sum_{k=1}^{s} \lambda_k y'(t_{n+s-k})$$

where the coefficients of the differential terms must follow $\sum_{k=1}^{s} \lambda_k = 1$. We can simplify the notation by allowing $y'_n = y'(t_n)$. Expanding the sum for $s = 4$ gives

$$y_{n+4} = y_{n+3} + h\left(\lambda_1 y'_{n+3} + \lambda_2 y'_{n+2} + \lambda_3 y'_{n+1} + \lambda_4 y'_n\right). \tag{A.7}$$

Each of the differential terms can be Taylor expanded at t_{n+3} up to order h^4

$$y'_{n+2} = y'_{n+3} - hy''_{n+3} + \frac{(-h)^2}{2!}y^{(3)}_{n+3} + \frac{(-h)^3}{3!}y^{(4)}_{n+3} + O\left(h^5\right), \tag{A.8}$$

$$y'_{n+1} = y'_{n+3} - 2hy''_{n+3} + \frac{(-2h)^2}{2!}y^{(3)}_{n+3} + \frac{(-2h)^3}{3!}y^{(4)}_{n+3} + O\left(h^5\right), \tag{A.9}$$

$$y'_{n} = y'_{n+3} - 3hy''_{n+3} + \frac{(-3h)^2}{2!}y^{(3)}_{n+3} + \frac{(-3h)^3}{3!}y^{(4)}_{n+3} + O\left(h^5\right). \tag{A.10}$$

$$\tag{A.11}$$

Substituting these into Equation (A.7) results in

$$\begin{aligned} y_{n+4} = y_{n+3} + h\lambda_1 y'_{n+3} &+ h\lambda_2\left(y'_{n+3} - hy''_{n+3} + \frac{h^2}{2}y^{(3)}_{n+3} - \frac{h^3}{6}y^{(4)}_{n+3}\right) \\ &+ h\lambda_3\left(y'_{n+3} - 2hy''_{n+3} + \frac{4h^2}{2}y^{(3)}_{n+3} - \frac{4h^3}{3}y^{(4)}_{n+3}\right) \\ &+ h\lambda_4\left(y'_{n+3} - 3hy''_{n+3} + \frac{9h^2}{2}y^{(3)}_{n+3} - \frac{9h^3}{2}y^{(4)}_{n+3}\right). \end{aligned}$$

Collecting like terms and simplifying gives

$$\begin{aligned} y_{n+4} = y_{n+3} + hy'_{n+3} &- h^2(\lambda_2 + 2\lambda_3 + 3\lambda_4)y''_{n+3} \\ &+ \frac{h^3}{2}(\lambda_2 + 4\lambda_3 + 9\lambda_4)y^{(3)}_{n+3} \\ &- \frac{h^3}{6}(\lambda_2 + 8\lambda_3 + 27\lambda_4)y^{(4)}_{n+3}. \end{aligned} \tag{A.12}$$

We can compare the coefficients of this with the Taylor expansion of y_{n+4} at t_{n+3} up to order h^4:

$$y_{n+4} = y_{n+3} + hy'_{n+3} + \frac{h^2}{2!}y''_{n+3} + \frac{h^3}{3!}y^{(3)}_{n+3} + \frac{h^4}{4!}y^{(4)}_{n+3} + O\left(h^5\right). \tag{A.13}$$

The comparison of coefficients in Equation (A.12) and the coefficients in Equation (A.13) leaves us with 3 simultaneous equations in λ_2, λ_3 and λ_4:

$$2\lambda_2 + 4\lambda_3 + 6\lambda_4 = -1, \tag{A.14}$$

$$3\lambda_2 + 12\lambda_3 + 27\lambda_4 = 1, \tag{A.15}$$

$$4\lambda_2 + 32\lambda_3 + 108\lambda_4 = -1. \tag{A.16}$$

We can solve these to obtain $\lambda_2 = -\frac{59}{24}, \lambda_3 = \frac{37}{24}$ and $\lambda_4 = -\frac{9}{24}$. To find the value of λ_1 recall that $\sum_{k=1}^{4}\lambda_k = 1$, hence we have that $\lambda_1 = \frac{55}{24}$.

Therefore, the four-step Adams-Bashforth method is

$$y_{n+4} = y_{n+3} + \frac{h}{24}\left(55f\left(t_{n+3}, y_{n+3}\right) - 59f\left(t_{n+2}, y_{n+2}\right) + 37f\left(t_{n+1}, y_{n+1}\right) - 9f\left(t_n, y_n\right)\right).$$

where $f(t_n, y_n) = y'(t_n)$.

A.2.3 Three-Step Adams-Moulton

The Adams-Moulton methods are similar to Adams-Bashforth methods in that they only depend on the previous value plus a linear combination of the derivative at different points in time. However, the Adams-Moulton methods are implicit methods which means that in order to compute $y(t_{n+s})$ we need to know the derivative at that point in time, $y'(t_{n+s}, y(t_{n+s}))$. Therefore the method needs to solve an algebraic equation for $y(t_{n+s})$. This change allows the s-step Adams-Moulton method to obtain order $s + 1$.

The one to three-step Adams-Moulton methods are [126]:

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}) \tag{A.17}$$

$$y_{n+2} = y_{n+1} + \frac{h}{2}\left(f\left(t_{n+2}, y_{n+2}\right) + f\left(t_{n+1}, y_{n+1}\right)\right) \tag{A.18}$$

$$y_{n+3} = y_{n+2} + \frac{h}{12}\left(5f\left(t_{n+3}, y_{n+3}\right) - 8f\left(t_{n+2}, y_{n+2}\right) - f\left(t_{n+1}, y_{n+1}\right)\right) \tag{A.19}$$

Though this method was solely due to John Adams, Forest Moulton's name is often associated with this methods because he realised they could be used together with the Adams-Bashforth methods to form a predictor-corrector pair [127]. This will be looked into further in the next section.

A.2.4 Adams-Bashforth With Variable Time Stepping

In order to allow the time step size to vary we use predictor-corrector [127]. It combines the explicit four-step Adams-Bashforth method with the implicit three-step Adams-Moulton method, with step size h. This method uses the value of y_{n+1} created by the four-step Adams-Bashforth as an estimate of the value at t_{n+1}, denoted $\tilde{y}_{n+1}$. It is used to estimate the derivative at t_{n+1},

$$\tilde{f}(t_{n+1}, y_{n+1}) = \frac{1}{h}\left(\tilde{y}_{n+1} + y_n\right).$$

The estimated derivative is then used in the three-step Adams-Moulton, which at t_{n+4} has the form

$$y_{n+4} = y_{n+3} + \frac{h}{24}\left(9\tilde{f}\left(t_{n+4}, y_{n+4}\right) + 19f\left(t_{n+3}, y_{n+3}\right) - 5f\left(t_{n+2}, y_{n+2}\right) + f\left(t_{n+1}, y_{n+1}\right)\right) \tag{A.20}$$

where $f(t_n, y_n) = y'(t_n)$. This gives the value of y_{n+1} that is used going forwards. The variable time step size comes from multiplying the original time step size, h, by q. When q is defined to ensure that the local truncation error is less than ϵ. The local truncation error for the four-step explicit Adams-Bashforth method is

$$\tilde{\tau}_{n+1}(h) = \frac{251}{720}y^(5)(\zeta_n)h^4$$

for $t_n < \zeta_n < t_{n+1}$, and the local truncation error for the three-step Adams-Moulton method is

$$\tau_{n+1}(h) = -\frac{19}{720}y^(5)(\eta_n)h^4$$

for $t_n < \eta_n < t_{n+1}$. If we presume all previous values, $y_0, y_1, y_2, \cdots, y_{n-1}, y_n$, are exact then it follows that

$$\tilde{\tau}_{n+1}(h) = \frac{y(t_{n+1}) - \tilde{y}_{n+1}}{h}, \qquad \tau_{n+1}(h) = \frac{y(t_{n+1}) - y_{n+1}}{h},$$

where $\tilde{y}_{n+1}$ and y_{n+1} are the values computed using the explicit and implicit method, respectively. We assume that $y_{n+1}(\zeta_n) \approx y_{n+1}(\eta_n)$, and then subtract the local truncation errors, to obtain

$$\frac{y_{n+1} - \tilde{y}_{n+1}}{h} \approx \frac{270}{720}h^s y^{s+1}(\eta_n).$$

This can be rearranged to give

$$y^{s+1}(\eta_n) \approx \frac{270}{720}\frac{y_{n+1} - \tilde{y}_{n+1}}{h^{s+1}}.$$

Substituting this into the local truncation error τ_{n+1} gives

$$|\tau_{n+1}(h)| = \frac{|y(t_{n+1}) - y_{n+1}|}{h} \approx \frac{19}{270}\frac{1}{h}|y(t_{n+1}) - y_{n+1}|.$$

In this method we multiple the step size h by q therefore, this multiplies $\tau_{n+1}(h)$ by q^s. In order for $|\tau_{n+1}(h)| < \varepsilon$ we must have that

$$\frac{19}{270}\frac{q^s}{h}|y(t_{n+1}) - y_{n+1}| < \varepsilon.$$

Which when rearranged to make q the subject of the formula is

$$q < \left(\frac{270}{19} \frac{\epsilon h}{|y_{n+1} - \tilde{y}_{n+1}|} \right)^{1/4}. \tag{A.21}$$

In practice we chose q to be the value calculated in Equation (A.21) rounded down to two decimal places and $\varepsilon = h^6$.

A.2.5 Fourth Order Runge-Kutta

Like the Adams-Bashforth methods the Runge-Kutta methods are a family of iterative methods, which include Euler's method, used for the approximate solution of ordinary differential equations with known initial value. The most widely known member of the Runge-Kutta family is the fourth-order Runge-Kutta method, commonly referred to as 'RK4'. As with the other time stepping methods we have seen we allow $f(t_n, y_n) = y'(t_n)$ and $y_n = y(t_n)$. With time step of size h we define

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{A.22}$$

where $t_{n+1} = t_n + h$, for $n = 0, 1, 2, \ldots$, using

$$k_1 = hf\left(t_n, y_n\right), \tag{A.23}$$

$$k_2 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right), \tag{A.24}$$

$$k_3 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right), \tag{A.25}$$

$$k_4 = hf\left(t_n + h, y_n + k_3\right). \tag{A.26}$$

Here we see that in contrast to the Adams-Bashforth methods which use previous values to calculate y_{n+1} the Runge-Kutta method uses intermediate values. For the fourth order method the next value is determined by the present value plus a weighted average of four increments, where the increments are based on an estimated slope specified by the function f.

The fourth-order Runge-Kutta, is a fourth-order method hence it has a local truncation error of $O(h^5)$.

Derivation Of Second Order Runge-Kutta

The derivation of the Runge-Kutta second order method follows the same ideas and methods as the derivation of the fourth order method but is much shorter. So to be more concise only the derivation of the second order method is presented here. The initial value problem is specified as

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0$$

where $t_n = t_0 + nh$, h is the size of the time step and the function f is known. The notation y' denotes the full derivative of y with respect to t.

We start by looking at the Taylor expansion of y_{n+1} at t_n up to order h^2:

$$y_{n+1} = y_n + hf(t_n, y_n) + \frac{h^2}{2!} f'(t_n, y_n)_n + \mathcal{O}\left(h^3\right). \tag{A.27}$$

Since f is a function of t and y we can convert the full derivative into partial derivatives using

$$f' = \frac{\mathrm{d}f}{\mathrm{d}t} = \frac{\partial f}{\partial t} + f\frac{\partial f}{\partial y}.$$

Hence Equation (A.27) becomes

$$y_{n+1} = y_n + hf + \frac{h^2}{2}\left[\frac{\partial f}{\partial t} + f\frac{\partial f}{\partial y}\right] + \mathcal{O}\left(h^3\right). \tag{A.28}$$

where the arguments of $f(t_n, y_n)$ have been suppressed for the ease of reading.

We will return to Equation (A.28) later in the derivation. The Runge-Kutta method attempts to find the unknown value of y_{n+1}, which can be written as

$$\begin{aligned} y_{n+1} &= y_n + \left[y_{n+1} - y_n\right], \\ &= y_n + \int_{t_n}^{t_{n+1}} f\left(\tau, y(\tau)\right)\, \mathrm{d}\tau. \end{aligned}$$

The integral is often difficult to calculate so it can be approximated using quadrature, this gives the modified expression

$$y_{n+1} = y_n + h\sum_{i=1}^{N} \omega_i f\left(t_n + hv_i, y(t_n + hv_i)\right), \tag{A.29}$$

where ω_i are the weights of the average and v_i are locations in time between t_n and t_{n+1}. Since we want to derive the second-order Runge-Kutta we set $N = 2$ and expand the sum

which gives us the equation

$$\begin{aligned} y_{n+1} = y_n &+ h\omega_1 f\left(t_n + hv_1, y\left(t_n + hv_1\right)\right) \\ &+ h\omega_2 f\left(t_n + hv_2, y\left(t_n + hv_2\right)\right). \end{aligned} \tag{A.30}$$

This can be written in terms of k_i

$$y_{n+1} = y_n + \omega_1 k_1 + \omega_2 k_2 \tag{A.31}$$

where

$$k_i = hf\left(t_n + hv_i, y\left(t_n + hv_i\right)\right).$$

Convention states that $v_1 = 0$ so that the first intermediate value calculated is at $t = t_n$. Using this we can obtain the expression for k_1:

$$k_1 = hf(t_n, y(t_n)). \tag{A.32}$$

A single step of Euler's method, Equation (A.1), allows us to write k_2 in terms of k_1,

$$\begin{aligned} k_2 &= f\left(t_n + hv_2, y\left(t_n + hv_2\right)\right), \\ &\approx f\left(t_n + hv_2, y_n + hv_2 y'\left(t_n\right)\right), \\ &\approx f\left(t_n + hv_2, y_n + hv_2 f\left(t_n, y_n\right)\right), \\ &\approx f\left(t_n + hv_2, y_n + v_2 k_1\right). \end{aligned} \tag{A.33}$$

In the fourth order scheme you would calculate k_3 and k_4 in the same way so that they would end up in the form $k_i = f\left(t_n + hv_i, y_n + \sum_{j=1}^{i-1} a_{ij}k_j\right)$. To determine the values of ω_i and v_2 we start by substituting k_1 and k_2 into Equation (A.31) which results in

$$y_{n+1} = y_n + \omega_1 hf(t_n, y_n) + \omega_2 hf(t_n + v_2 h, y_n + v_2 k_1). \tag{A.34}$$

Now consider the two-dimensional Taylor expansion,

$$f(t + h, y + g) = f(t, y) + h\frac{\partial f(t, y)}{\partial t} + g\frac{\partial f(t, y)}{\partial y} + \ldots$$

where for the fourth-order derivation higher order terms are also included. Substituting this expansion into our expanded quadrature equation, Equation (A.34), results in

$$y_{n+1} = y_n + \omega_1 hf(t_n, y_n) + \omega_2 h\left[f(t_n, y_n) + v_2 h\frac{\partial f}{\partial t} + v_2 hf\frac{\partial f}{\partial y}\right] + O\left(h^3\right).$$

We then set this equal to the Taylor expanded y_{n+1} that was found earlier

$$y_n + hf + \frac{h^2}{2}\left[\frac{\partial f}{\partial t} + f\frac{\partial f}{\partial y}\right] + O\left(h^3\right) = y_n + (\omega_1 + \omega_2)\, hf + \omega_2 v_2 h^2 \frac{\partial f}{\partial t} + \omega_2 v_2 h^2 f \frac{\partial f}{\partial y} + O\left(h^3\right).$$

which when we equate terms on both sides, we find that there are two statements that must hold if the Runge-Kutta method is to be consistent. These are

$$\omega_1 + \omega_2 = 1, \tag{A.35}$$

$$v_2\omega_2 = \frac{1}{2}. \tag{A.36}$$

When deriving the fourth order method, there are more of these equivalences to be made in order to find the statements of consistency. Since we have two equations for three unknowns, we have the choice of one of our parameters. The canonical choice for the second order Runge-Kutta method is for $v_2 = 1$, which then results in $\omega_1 = \omega_2 = \frac{1}{2}$. Therefore, the second order Runge-Kutta method is

$$y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2) + O\left(h^3\right) \tag{A.37}$$

where $k_1 = hf(t_n, y_n)$ and $k_2 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$.

The derivation for the fourth order Runge-Kutta follows the same ideas shown in this section but is significantly longer. The derivation is shown in full in the paper by Musa et al. [128] published in 2010.

A.2.6 Testing Numerical Method Stability

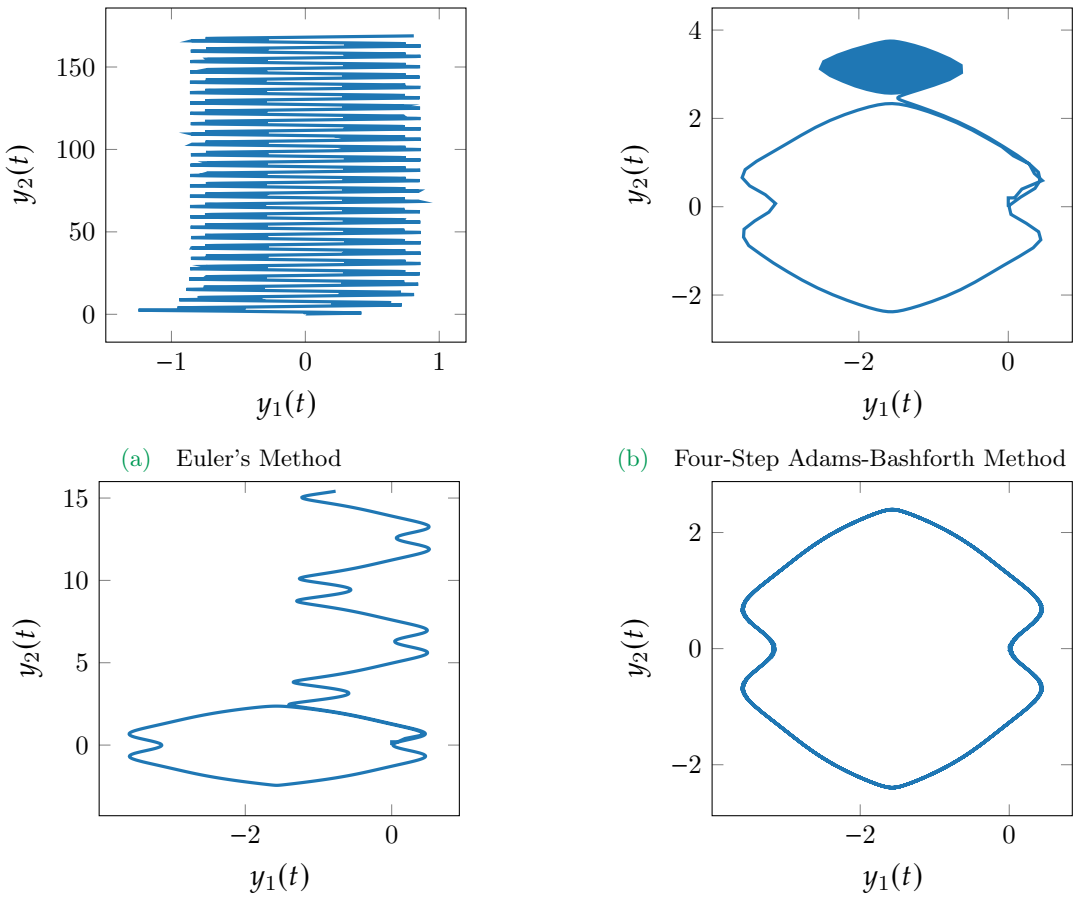
As mentioned at the end of Section A.2.1 the stability of the different methods can determine which method should be used for numerically integrating a specific ordinary differential equation. To show the stability of the four different time stepping methods we have described here we will look at how they perform for the following coupled ODEs

$$\dot{y}_1 = \sin(5\sin(y_2)), \tag{A.38}$$

$$\dot{y}_2 = \cos(y_1), \tag{A.39}$$

where both y_1 and y_2 are functions of t.

When setting the time step size to $h = 0.2$ and the precision value to $\varepsilon = h^{10}$ in the variable Adams-Bashforth method the differences between the four methods mentioned earlier in this chapter are emphasised. The result of numerically integrating the coupled



(a) Euler's Method

(b) Four-Step Adams-Bashforth Method

(c) Variable Stepsize Adams-Bashforth Method

(d) Runge-Kutta 4 Method

Figure A.1: Solving the same ODE with different time stepping methods to observe method stability.

ODEs for each of our four methods is shown in Figure A.1. From the figures we can see that for the coupled ODEs, Equations (A.38) & (A.39), only one of the four methods tried reaches a stable solution. Euler's method fared poorly for these coupled ODEs with the value of $y_2(t)$ quickly growing very large. The Adam-Bashforth methods both initially produced a solution close to the stable one, but then deviated away. Only the fourth order Runge-Kutta method was able to find the stable solution.

A.3 Random Distributions

Now that we have introduced the numerical time stepping methods which will be used while simulating from the mathematical models of collective motion we will define two continuous probability distributions. These distributions will be used in the definition of some of the mathematical models for collective motion as well as in our computer vision

algorithm for detecting and tracking sheep in the data videos.

A.3.1 Uniform Distribution

The continuous uniform distribution (also known as the rectangular distribution) is a family of probability distributions. The general form of the probability density function for this family is

$$f(x) = \begin{cases} \frac{1}{b-a}, & \text{for } a \leq x \leq b, \\ 0, & \text{otherwise.} \end{cases} \tag{A.40}$$

A.3.2 Normal Distribution

The normal distribution (also known as the Gaussian distribution) is a continuous probability distribution for a random variable. The general form of the probability density function is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}. \tag{A.41}$$

The parameter μ is the mean, median and mode of the distribution and σ is its standard deviation. When a random variable X is distributed normally with mean μ and variance σ^2 it is written

$$X \sim \mathcal{N}\left(\mu, \sigma^2\right) \tag{A.42}$$

The standard normal distribution sets the mean to 0 and the standard deviation to 1. This is denoted $\mathcal{N}(0, 1^2)$.

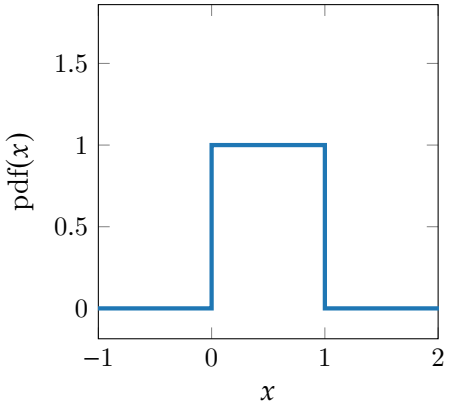


Figure A.2: The standard uniform probability density function.

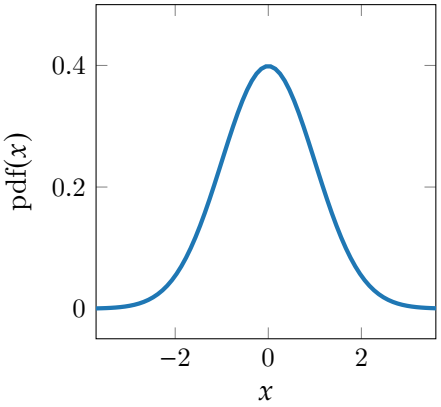


Figure A.3: The standard normal probability density function.

A.4 Convex Hull

In mathematics, the convex hull of a set of points X in the Euclidean plane, is the smallest convex set that contain X [113]. For example, when X is a bounded subset of the plane, the convex hull may be visualised as the shape enclosed by a rubber band stretched around X, Figure A.4.

The convex hull is the set of all convex combinations of its points. In a convex combination, each point x_i in X is assigned a weight α_i in such a way that they are all non-negative and sum to one. These weights are used to compute a weighted average of the points. For each choice of coefficients the resulting convex combination is a point in the convex hull. The whole hull is formed by selecting all possible coefficients and calculating each convex combination. This can be expressed in a single formula: the convex hull is the set

$$\text{Conv}(X) = \left\{ \sum_{i=1}^{|X|} \alpha_i x_i \;\middle|\; (\forall i : \alpha_i \geq 0) \wedge \sum_{i=1}^{|X|} \alpha_i = 1 \right\}.$$

The convex hull of a finite set of points in 2-dimensions, $X \in \mathbb{R}^2$, is a convex polygon. Each point in x_i in X which is not in the convex hull of the remaining points, $x_i \notin \text{Conv}(X \backslash \{x_i\})$ is called a vertex of $\text{Conv}(X)$. If all the points in X lie on a line, then the convex hull is the line segment joining the two outermost points.

We will be using the convex hull of the locations of simulated agents and observed sheep to find the approximate area covered by the flock. To do this we use the spatial SciPy package [129] in Python 2.7 [87] to calculate the convex hull of the locations, Section 3.6.2 and Section 4.3.

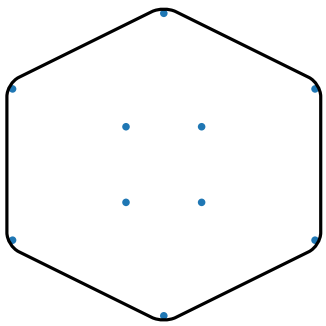


Figure A.4: Convex hull of the points •

A.5 Voronoi Diagram

A Voronoi diagram, also known as Voronoi tessellation, is a way of partitioning the plane into regions based on distance to points on the plane. The set of points, known as seed points, is specified before the partitioning and for each seed point there is a corresponding region consisting of all points closer to the seed point than to any of the other seed points. These regions are known as Voronoi cells.

Voronoi diagrams are named after Georgy Voronoy, a Russian mathematician who defined and studied the n-dimension case in 1908 [112].

Voronoi tessellations of regular lattices of points in 2-dimensions and 3-dimensions give rise to many familiar tessellations:

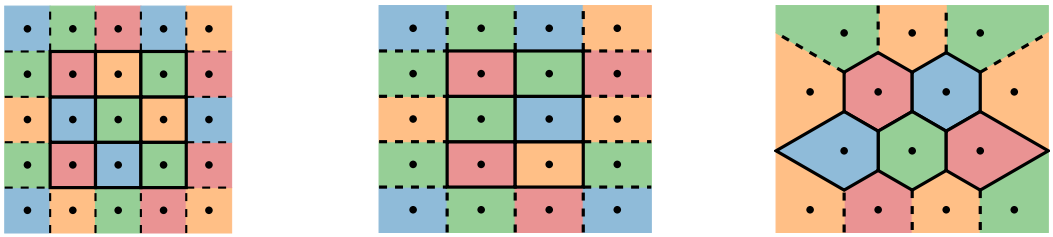
- A square lattice gives rise to the regular tessellation of squares.
- A rectangular lattice produces a tessellation of rectangles sorted into rows and columns.
- Other 2D lattices gives an irregular honeycomb tessellation.

A depiction of what these Voronoi tessellations look like can be seen in Figure A.5.

The formal definition of a Voronoi tessellation is as follows. Let X be a metric space where d is a distance function. Let K be a set of indices and $(P_k)_{k\in K}$ be a tuple of nonempty subsets in the space defined by X. Each of these P_k are the sites of the Voronoi diagram. The Voronoi cell, R_k, affiliated with the site P_k is the set of all points in X whose distance to P_k is not greater than their distance to other sites, P_j where $j \neq k$. So therefore if $d(x, P_i)$ denotes the distance between a point x and a seed point P_i, then the Voronoi cell is

$$R_k = \{x \in X | d(x, P_k) \leq d(x, P_j) \forall j \neq k\}.$$

The full Voronoi diagram is the tuple of these cells, $(R_k)_{k\in K}$.



(a) Square lattice (b) Rectangular lattice (c) Hexagonal lattice

Figure A.5: Examples of Voronoi diagrams for different types of 2-dimensional lattices.

In our work using Voronoi diagrams we use the spatial SciPy package [129] in Python 2.7 [87] to calculate the Voronoi tessellations of the locations of simulated agents, Section 4.3.

A.6 Pearson Correlation Coefficient

In statistics the Pearson correlation coefficient is a measure of the linear correlation between two variables X and Y [130]. According to the Cauchy-Schwarz inequality [131] it has a value between +1 and −1. A value of +1 refers to total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation, Figure A.7.

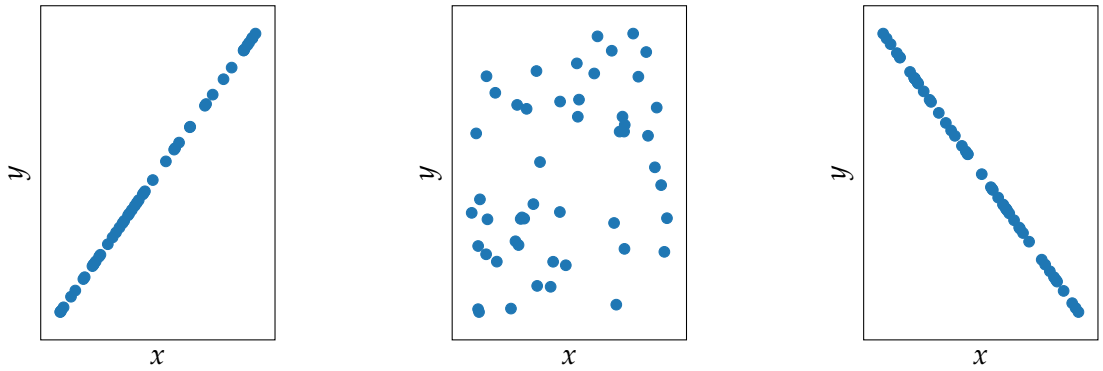
The Pearson correlation coefficient can be applied to both the population and a sample. When calculated for a population this quantity is denoted ρ_{XY} and is sometimes referred to as the population correlation coefficient. Given a pair of random variables (X, Y) the formula for calculating ρ_{XY} is

$$\rho_{XY} = \frac{cov(X, Y)}{\sigma_X \sigma_Y} \tag{A.43}$$

where σ_X, σ_Y are the standard deviations of the random variables X and Y respectively, and $cov(X, Y)$ is the covariance of the two random variables and is defined to be the expected value of their product minus the product of their expected values:

$$cov(X, Y) = E[XY] - E[X]E[Y].$$

When applied to a sample the Pearson correlation coefficient is often denoted r_{xy}. Given paired data in the form $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ where n is the number of pairs the quantity



(a) $r_{xy} = 1$ (b) $r_{xy} = 0$ (c) $r_{xy} = -1$

Figure A.6: Schematics showing pairwise data with Pearson correlation coefficient r_{xy} shown.

r_{xy} is defined as

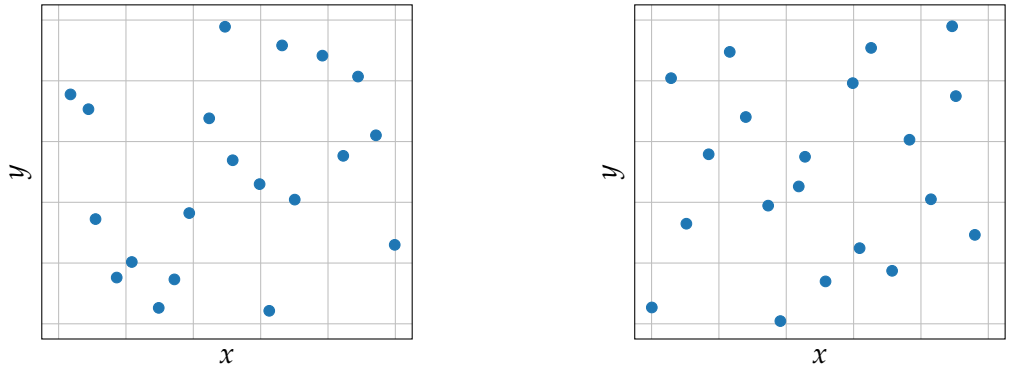
$$r_{xy} = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sqrt{\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right)}\sqrt{\left(\sum_{i=1}^{n} y_i^2 - n\bar{y}^2\right)}} \tag{A.44}$$

where x_i and y_i are the i^{th} points in the sample. The quantities $\bar{x}$ and $\bar{y}$ are the means of the two individual samples.

A.7 Latin Hypercube Sampling

Latin Hypercube sampling is a method of sampling numbers to cover the whole sample space. The method was proposed by McKay et al. [132] in 1979 and its ease of use has made it popular in computer experiments [133].

In 2-dimensions, the sampling constructs a Latin square by splitting the sample space into n_r rows and columns each with exactly one sampled point in it. Figure A.7a shows the case where there are $n = 20$ sampled points in the range $0 < x, y < 1$. When the sampled points are projected onto the axis they provide good coverage in that dimension. A Latin hypercube is the generalisation of this idea to any number of dimensions, where there is only one sample in each axis-aligned hyperplane containing it. However, the method proposed by McKay et al. [132] does not ensure that the full sample space is well covered as sampled points on the diagonal would produce a Latin square which is valid but does not fill the whole sample space.



(a) Latin hypercube (b) Maximum Latin hypercube

Figure A.7: Schematics showing two different Latin hypercube sampling methods, each with 20 sampled points. Whilst both methods cover every value of x and y the original sampling method (a) has a small cluster of points for small x and y leaving areas with a low density of points whereas in (b) clusters like this should not occur.

Other methods have been designed which account for this; Morris and Mitchell [134] proposed that in order to find the best Latin hypercube sample the distance between the points should be maximised. Figure A.7b shows $n = 20$ sampled points using the maximised Latin hypercube. Comparing to the standard Latin hypercube case, Figure A.7a, we can see that they have slightly better coverage over the sample space. However, this method is computationally more intensive and so we chose to use the original Latin hypercube sampling throughout the following chapters. We used the pyDOE [135] package in Python 2.7 [87] to compute our Latin hypercubes.

Now that most of the mathematical concepts used throughout this thesis have been laid out we will move on to Part II where we will explore the computer vision algorithm we devised to locate and track sheep in aerial drone footage and the information we can extract from the trajectories of the sheep.

B

The Mahalanobis Distance

An alternative method we investigated to convert the colour image to greyscale was to calculate the Mahalanobis distance between each pixel and the average background values. To calculate the Mahalanobis distance of the image we first need to know the covariance of the three colour channels, $S = \text{Cov}(R, G, B)$, so we know how much the three colour channels influence each other. We also need to know the average background values. We do this by cropping the image down to show only the background and then calculating the average of this subsection. The larger we can make this area the more reliable our background average will be. Once we have these two pieces of information we can calculate the Mahalanobis distance for each pixel:

$$MD_{i,j} = \sqrt{(\mathcal{F}_{i,j} - \mathbf{b})^T S^{-1} (\mathcal{F}_{i,j} - \mathbf{b})} \tag{B.1}$$

where $\mathbf{b}$ is a vector representing the background average and S is our covariance matrix.

When doing this calculation with an image from the drone we obtain an image that appears much noisier than the original, Figure B.1. This is because the shadows of the sheep will

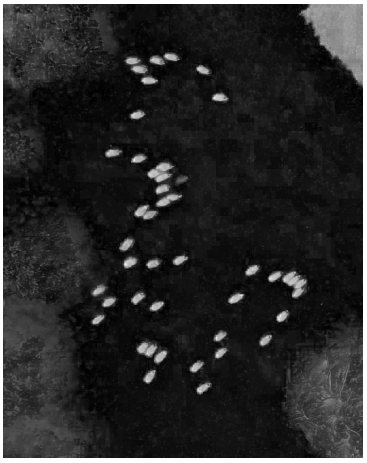


Figure B.1: The Mahalanobis distance from the average background values.

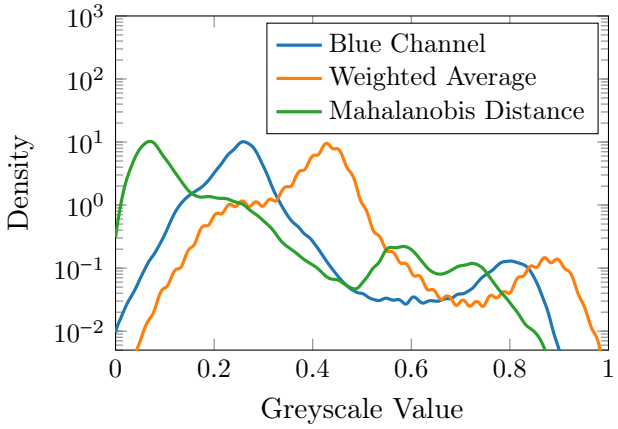


Figure B.2: The kernel density of 2 methods from Chapter 2 and the Mahalanobis distance.

have a greater distance to the average grass value than the grass not in shadow and hence will appear bright in the greyscale image. In other methods the appearance of the fence was subdued, but in this method the appearance of the fence was strengthened which is not ideal. Because of all of this added noise is it unlikely that even a human could correctly locate each of the sheep in the image, therefore this method is not an appropriate method for our problem.

Consider the kernel density curve for the Mahalanobis distance, Figure B.2. This density curve does not have two defined peaks reinforcing our ideas that this method should not be used for detecting the sheep.

C

Flocking videos

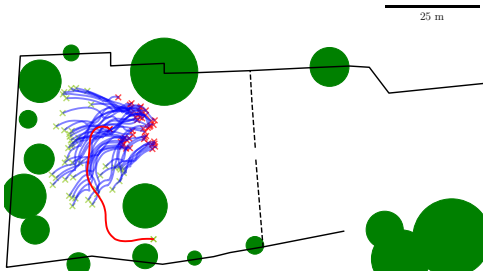


Figure C.1: Video 1.

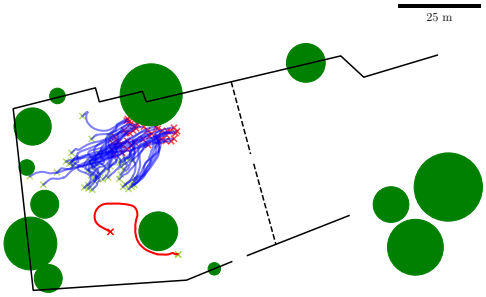


Figure C.2: Video 2.

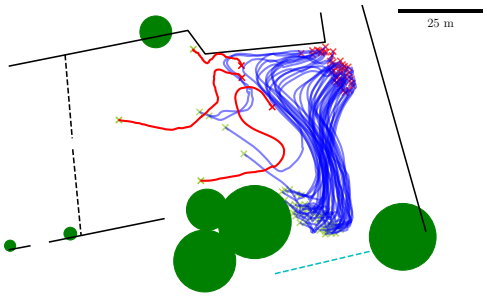


Figure C.3: Video 3.

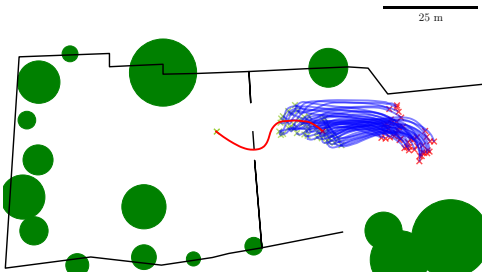


Figure C.4: Video 5.

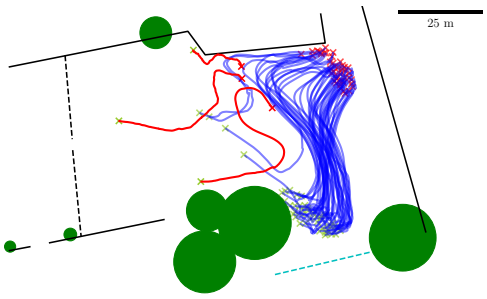


Figure C.5: Video 4.

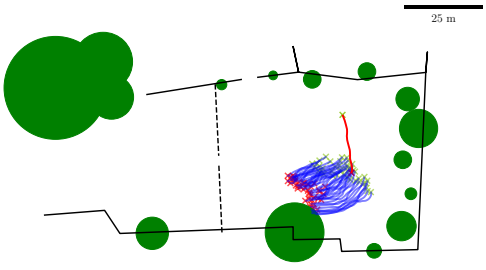


Figure C.6: Video 6.

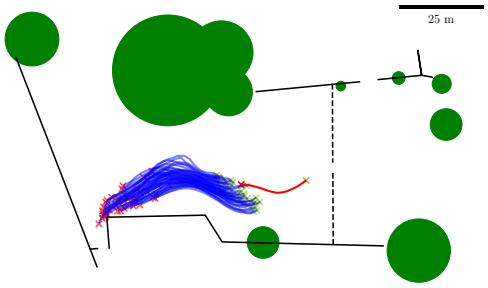


Figure C.7: Video 7.

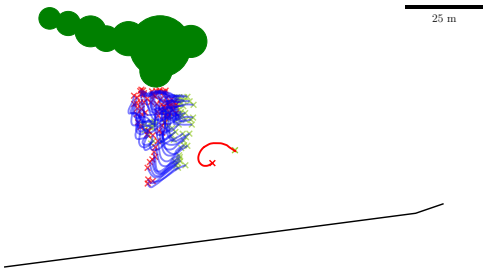


Figure C.8: Video 8.

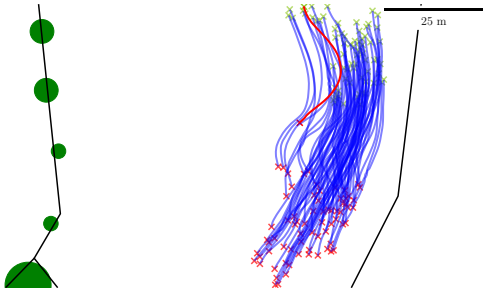


Figure C.9: Video 9.

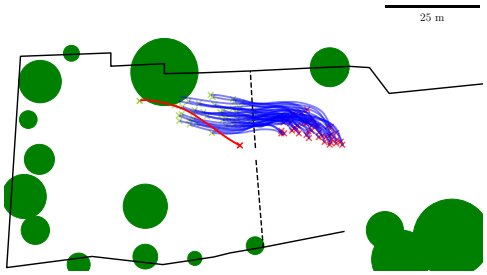


Figure C.10: Video 10.

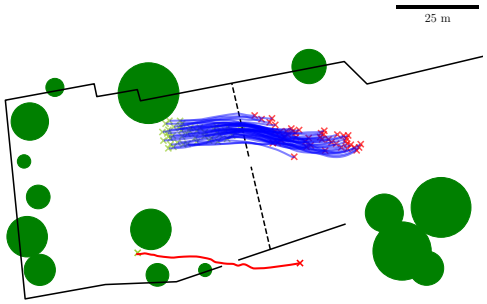


Figure C.11: Video 11.

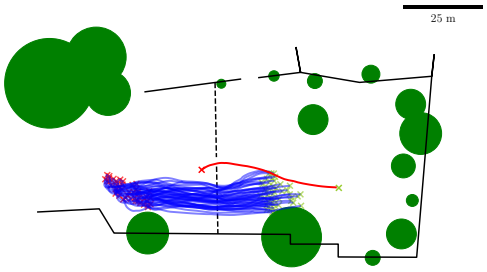


Figure C.12: Video 12.

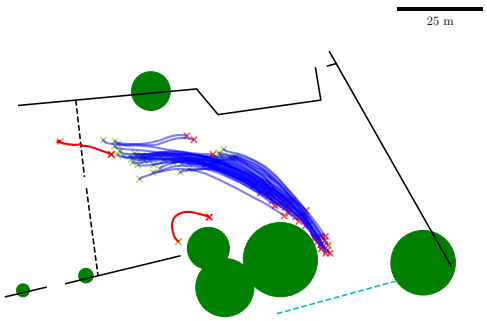


Figure C.13: Video 13.

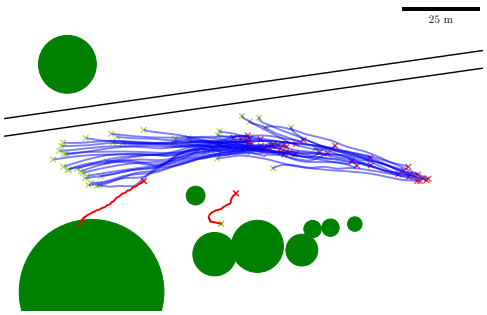


Figure C.14: Video 14.

Bibliography

[1] L. Spence, *An encyclopedia of occultism* (Citadel Press, 1993).

[2] H. Chrisholm, *The Encyclopædia britannica: a dictionary of arts, sciences, literature and general information*, vol. 29 (At the University press, 1911).

[3] O. A. Croze, R. N. Bearon, and M. A. Bees, Journal Of Fluid Mechanics **816**, 481–506 (2017).

[4] J. K. Parrish, S. V. Viscido, and D. Grünbaum, The Biological Bulletin **202**, 296 (2002).

[5] J. K. Parrish and L. Edelstein-Keshet, Science **284**, 99 (1999).

[6] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, E. Bonabeau, and G. Theraula, *Self-organization in biological systems*, vol. 7 (Princeton university press, 2003).

[7] M. Moussaid, E. G. Guillot, M. Moreau, J. Fehrenbach, O. Chabiron, S. Lemercier, J. Pettré, C. Appert-Rolland, P. Degond, and G. Theraulaz, PLOS Computational Biology **8**, e1002442 (2012).

[8] E. O. Wilson, *Sociobiology: The new synthesis* (Harvard University Press, 1975).

[9] N. Carter, S. Levin, A. Barlow, and V. Grimm, Ecological Modelling **312**, 347 (2015).

[10] H. Kruuk, Journal Of Zoology **184**, 1 (1978).

[11] A. Jolly, Science **153**, 501 (1966).

[12] A. Hough, The Telegraph (2011), Accessed: 01/06/2017, URL `http://www.telegraph.co.uk/news/earth/wildlife/8315106/Starling-flocks-how-they-form-into-incredible-wildlife-spectacles.html`.

[13] D. Rosengren, *Wildebeest migration in Serengeti national park, Tanzania* (May, 2013).

[14] J. Rotman, `http://tinyurl.com/mnvmnhc`, Accessed: 04/02/2015.

[15] M. Joseph, *Bavaria beer festival* (Sept, 14).

[16] D. L. Lack, *Ecological adaptations for breeding in birds* (Methuen, 1968).

[17] E. Shaw, American Scientist **66**, 166 (1978).

[18] G. Arnold, Applied Animal Ethology **3**, 263 (1977).

[19] N. C. Makris, P. Ratilal, S. Jagannathan, Z. Gong, M. Andrews, I. Bertsatos, O. R. Godø, R. W. Nero, and J. M. Jech, Science **323**, 1734 (2009).

[20] D. Gorbonos, R. Ianconescu, J. G. Puckett, R. Ni, N. T. Ouellette, and N. S. Gov, New Journal Of Physics **18**, 073042 (2016).

[21] E. Danchin, R. H. Wagner, J. K. Parrish, and L. Edelstein-Keshet, Science **287**, 804 (2000).

[22] M. L. Cody, Theoretical Population Biology **2**, 142 (1971).

[23] J. R. Krebs, M. H. MacRoberts, and J. Cullen, Ibis **114**, 507 (1972).

[24] W. D. Hamilton, Journal Of Theoretical Biology **31**, 295 (1971).

[25] I. Vine, Journal Of Theoretical Biology **30**, 405 (1971).

[26] P. J. Watt, S. F. Nottingham, and S. Young, Animal Behaviour **54**, 865 (1997).

[27] S. V. Viscido and D. S. Wethey, Animal Behaviour **63**, 735 (2002).

[28] J. Burger and M. Gochfeld, *The Common Tern: its breeding biology and social behavior* (iUniverse, 1991).

[29] T. Vicsek and A. Zafeiris, Physics Reports **517**, 71 (2012).

[30] A. Shklarsh, G. Ariel, E. Schneidman, and E. Ben-Jacob, PLOS Computational Biology **7** (2011).

[31] J. Garnier, G. Papanicolaou, and T.-W. Yang, ArXiv Preprint arXiv:1611.02194 (2016).

[32] L. Edelstein-Keshet, J. Watmough, and D. Grunbaum, Journal Of Mathematical Biology **36**, 515 (1998).

[33] J. M. Miller, A. Kolpas, J. P. J. Neto, and L. F. Rossi, Bulletin Of Mathematical Biology **74**, 536 (2012).

[34] C. M. Topaz, M. R. D'Orsogna, L. Edelstein-Keshet, and A. J. Bernoff, PLOS Computational Biology **8** (2012).

[35] C. A. Yates, R. Erban, C. Escudero, I. D. Couzin, J. Buhl, I. G. Kevrekidis, P. K. Maini, and D. J. Sumpter, Proceedings Of The National Academy Of Sciences **106**, 5464 (2009).

[36] D. Helbing, I. J. Farkas, P. Molnar, and T. Vicsek, Pedestrian And Evacuation Dynamics **21**, 21 (2002).

[37] A. Choudhary, D. Venkataraman, and S. S. Ray, EPL (Europhysics Letters) **112**, 24005 (2015).

[38] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, Physical Review Letters **75**, 1226 (1995).

[39] J. Conway, Scientific American **223**, 4 (1970).

[40] S. Wolfram, Advances In Applied Mathematics **7**, 123 (1986).

[41] S. Wolfram, *Cellular automata and complexity: collected papers* (Westview, 1994).

[42] I. Aoki, Bulletin Of The Japanese Society Of Scientific Fisheries **48**, 1081 (1982).

[43] C. W. Reynolds, Proceedings of the 14th annual conference on Computer graphics and interactive techniques pp. 25–34 (1987).

[44] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, Journal Of Theoretical Biology **218**, 1 (2002).

[45] Y. Chen and T. Kolokolnikov, Journal Of The Royal Society Interface **11** (2014).

[46] J. Toner and Y. Tu, Physical Review Letters **75**, 4326 (1995).

[47] J. Toner and Y. Tu, Physical Review E **58**, 4828 (1998).

[48] E. Bertin, M. Droz, and G. Grégoire, Physical Review E **74**, 022101 (2006).

[49] F. Cucker and S. Smale, IEEE Transactions On Automatic Control **52**, 852 (2007).

[50] S.-Y. Ha, E. Jeong, and M.-J. Kang, Nonlinearity **23**, 3139 (2010).

[51] C. A. Yates, R. E. Baker, R. Erban, and P. K. Maini, Canadian Applied Mathematics Quarterly **18**, 299 (2010).

[52] G. D. Kattas, X.-K. Xu, and M. Small, PLOS Computational Biology **8** (2012).

[53] N. Khurana and N. T. Ouellette, New Journal Of Physics **15**, 095015 (2013).

[54] A. W. Baggaley, Physical Review E **91**, 053019 (2015).

[55] D. Armbruster, S. Motsch, and A. Thatcher, Physica D: Nonlinear Phenomena **344**, 58 (2017).

[56] F. Ginelli, F. Peruani, M.-H. Pillot, H. Chaté, G. Theraulaz, and R. Bon, Proceedings Of The National Academy Of Sciences **112**, 12729 (2015).

[57] H. E. L. Moore, *Photograph of a daffodil* (April, 2020).

[58] N. Snavely, S. M. Seitz, and R. Szeliski, in *ACM Siggraph 2006 Papers* (2006), pp. 835–846.

[59] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, IEEE 11th International Conference On Computer Vision pp. 1–8 (2007).

[60] T. B. Moeslund, A. Hilton, and V. Krüger, Computer Vision And Image Understanding **104**, 90 (2006).

[61] S. Noiumkar and S. Tirakoat, International Conference on Informatics and Creative Multimedia pp. 310–313 (2013).

[62] R. Szeliski, *Computer vision: algorithms and applications* (Springer Science & Business Media, 2010).

[63] Hawkeye Innovations, `https://www.hawkeyeinnovations.com`, Accessed: 09/05/2020.

[64] M. Mori, K. F. MacDorman, and N. Kageki, IEEE Robotics Automation Magazine **19**, 98 (2012).

[65] K. F. MacDorman and D. Chattopadhyay, Cognition **146**, 190 (2016).

[66] M. A. Boden, *Mind as machine: A history of cognitive science* (Oxford University Press, 2008).

[67] E. Kabir, A. Downton, and R. Birch, Proceedings. 10th International Conference on Pattern Recognition **1**, 469 (1990).

[68] R. A. Lotufo, A. D. Morgan, and A. S. Johnson, IEEE Colloquium on Image Analysis for Transport Applications pp. 1–6 (1990).

[69] D. Roble, ACM SIGGRAPH Computer Graphics **33**, 58 (1999).

[70] D. Roble and N. B. Zafar, Computer **42**, 35 (2009).

[71] B. Singh Bal and G. Dureja, Sport Science Review **21** (2012).

[72] R. Cross, Sports Engineering **17**, 239 (2014).

[73] R. Lukeman, Y.-X. Li, and L. Edelstein-Keshet, Proceedings Of The National Academy Of Sciences **107**, 12576 (2010).

[74] D. Liu and J. Yu, 2009 Ninth International Conference on Hybrid Intelligent Systems **1**, 344 (2009).

[75] A. Bleau and L. J. Leon, Computer Vision And Image Understanding **77**, 317 (2000).

[76] M. Usaj, D. Torkar, M. Kanduser, and D. Miklavcic, Journal Of Microscopy **241**, 303 (2011).

[77] J. Byun, M. R. Verardo, B. Sumengen, G. P. Lewis, B. S. Manjunath, and S. K. Fisher, Molecular Vision **12**, 949 (2006).

[78] C. Rueden, B. DeZonia, and G. Harris, *Imagej2*, `https://imagej.net/software/imagej2/`.

[79] Labelbox Inc, *Labelbox*, `https://labelbox.com`.

[80] MatLab, *Matlab image processing toolbox*, `https://uk.mathworks.com/products/image.html`.

[81] Oxford Instruments, *Imaris*, `https://imaris.oxinst.com`.

[82] D. H. Ballard and C. M. Brown, *Computer Vision* (Prentice Hall Professional Technical Reference, 1982), 1st ed., ISBN 0131653164.

[83] R. E. Kalman, Transaction of the ASME - Journal of Basic Engineering pp. 35–45 (1960).

[84] S. Toulet, J. Gautrais, R. Bon, and F. Peruani, PLOS One **10**, e0140188 (2015).

[85] DJI, *Dji phantom 3 standard*, `www.dji.com/phantom-3-standard`, Accessed: 16/05/2018.

[86] Google, Map Data ©2009 Google, Accessed: 04/06/2018.

[87] Python Software Foundation, *Python 2.7.15 documentation*, Accessed: 25/05/2016, URL `https://docs.python.org/2.7/`.

[88] T. E. Oliphant, *Guide to NumPy* (CreateSpace Independent Publishing Platform, USA, 2015), 2nd ed., ISBN 151730007X, 9781517300074.

[89] G. Bradski, Dr. Dobb's Journal Of Software Tools (2000).

[90] H. E. L. Moore, *Photograph of Tynemouth priory* (April, 2018).

[91] MatLab, *rgb2gray documentation*, `uk.mathworks.com/help/matlab/ref/rgb2gray.html`.

[92] C. A. Poynton, *Digital video and HDTV: algorithms and interfaces* (Morgan Kaufmann, 2007), ISBN 1-55860-792-7.

[93] H. E. L. Moore, *Photograph of Mike "Beard Guy" Taylor playing keyboard at the Walk off the Earth concert at The Sage, Gateshead* (May, 2018).

[94] H. E. L. Moore, *Photograph of Luna the cat* (Nov, 2017).

[95] N. Ostu, IEEE Transactions On Systems, Man And Cybernetics **9**, 62 (1979).

[96] J. Shi and C. Tomasi, 9th IEEE Conference On Computer Vision And Pattern Recognition pp. 593–600 (1994).

[97] Skoda, *Fabia hatchback performance and engines*, `http://www.skoda.co.uk/models/fabia-hatch/performances-and-engines`.

[98] G. Gallego, C. Cuevas, R. Mohedano, and N. Garcia, IEEE Transactions On Signal Processing **61**, 4387 (2013).

[99] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, vol. 55 (Courier Corporation, 1965).

[100] P. Davidson, *Turbulence. an introduction for scientists and engineers oxford university press* (2004).

[101] G. W. Stagg, N. G. Parker, and C. F. Barenghi, Physical Review A **94**, 053632 (2016).

[102] E. Ising, Zeitschrift Für Physik A Hadrons And Nuclei **31**, 253 (1925).

[103] N. D. Mermin and H. Wagner, Physical Review Letters **17**, 1133 (1966).

[104] G. Chen, IEEE Transactions On Automatic Control **62**, 636 (2016).

[105] P. W. Miller and N. T. Ouellette, Physical Review E **89**, 042806 (2014).

[106] G. Baglietto, E. V. Albano, and J. Candia, Physica A: Statistical Mechanics And Its Applications **392**, 3240 (2013).

[107] A. Huth and C. Wissel, Ecological Modelling **75**, 135 (1994).

[108] M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, V. Lecomte, A. Orlandi, G. Parisi, A. Procaccini, et al., **105**, 1232 (2008).

[109] B.-M. Tian, H.-X. Yang, W. Li, W.-X. Wang, B.-H. Wang, and T. Zhou, Physical Review E **79**, 052102 (2009).

[110] K. W. Rio, G. C. Dachner, and W. H. Warren, Proceedings Of The Royal Society B: Biological Sciences **285**, 20180611 (2018).

[111] T. Pitcher and J. Parrish, *Functions of Shoaling Behaviour in Teleost* (London: Chapman and Hall, 1993), chap. 12, pp. 363–439.

[112] G. Voronoi, Journal Für Die Reine Und Angewandte Mathematik **134**, 198 (1908).

[113] D. E. Knuth, *Axioms and hulls*, vol. 606 (Springer-Verlag Berlin, 1992).

[114] D. Helbing, I. Farkas, and T. Vicsek, Nature **407**, 487 (2000).

[115] U. Weidmann, IVT Schriftenreihe **90** (1993).

[116] P. M. Lee, *Bayesian Statistics* (Wiley, 2012).

[117] K. Csilléry, M. G. Blum, O. E. Gaggiotti, and O. François, Trends In Ecology & Evolution **25**, 410 (2010).

[118] M. A. Beaumont, Annual Review Of Ecology, Evolution, And Systematics **41**, 379 (2010).

[119] A. Sottoriva and S. Tavaré, in *Proceedings of COMPSTAT'2010* (Springer, 2010), pp. 57–66.

[120] P. Joyce and P. Marjoram, Statistical Applications In Genetics And Molecular Biology **7** (2008).

[121] X. Didelot, R. G. Everitt, A. M. Johansen, D. J. Lawson, et al., Bayesian Analysis **6**, 49 (2011).

[122] M. Ptáček, M. Milerski, J. Schmidová, J. Ducháček, V. Tančin, M. Uhrinčať, J. Hakl, and L. Stádník, Small Ruminant Research **165**, 131 (2018).

[123] BikeSure, *Quad bike FAQ*, `https://www.bikesure.co.uk/quad-bike-faq/`.

[124] E. Johns, Private Communication (2018).

[125] F. Bashforth and J. C. Adams, *An attempt to test the theories of capillary action: by comparing the theoretical and measured forms of drops of fluid* (University Press, 1883).

[126] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations in Nonstiff problems*, vol. 8 (Springer Series in Computational Mathematics, 1993).

[127] F. R. Moulton, *New methods in exterior ballistics* (Chicago, 1926).

[128] H. Musa, I. Saidu, and M. Waziri, International Journal Of Computer Applications **9**, 51 (2010).

[129] E. Jones, T. Oliphant, P. Peterson, et al., *SciPy: Open source scientific tools for Python*, `http://www.scipy.org/` (2001–), [Online; accessed 01/02/2019].

[130] K. Pearson, Proceedings Of The Royal Society Of London **58**, 240 (1895).

[131] A.-L. Cauchy, Cours d'Analyse, 1er Partie: Analyse Algebrique pp. 373–377 (1821).

[132] M. D. McKay, R. J. Beckman, and W. J. Conover, Technometrics **21**, 239 (1979).

[133] K. Q. Ye, Journal Of The American Statistical Association **93**, 1430 (1998).

[134] M. D. Morris and T. J. Mitchell, Journal Of Statistical Planning And Inference **43**, 381 (1995).

[135] J.-M. Martinez, Y. Collette, M. Baudin, M. Christopoulou, M. Baudin, et al., *pyDOE: The experimental design package for python*, `https://pythonhosted.org/pyDOE/` (2009–), [Online; accessed 22/10/2019].