Developing Methods for Real-time Pedestrian Agent-Based Modelling

An Ensemble Kalman Filter Approach

Keiran Suchak

University of Leeds

School of Geography

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

November, 2022

Intellectual Property Statement

Acknowledgements

The process of completing a PhD is not something that is done alone. Whilst the work itself — the words on the page, the lines of code, the graphs and diagrams — may be the work of an individual, the journey of the PhD is one that is taken with a great deal of support and assistance. It is thanks to the support of my various families and communities that I have reached this stage.

The first of these that I would like to thank is my academic community. I would like to thank my supervisors — Nick and Jon — for all of the support that they have provided over the course of the PhD. The discussions that we have had have been invaluable in my academic development. I would also like to thank Minh for all of his advice and help, both whilst he was with us at Leeds and since he has moved to Auckland.

I would also like to thank my friends. To the friends that I have made over the course of the PhD — Natacha, Iain, Eugeni, Annabel, Sedar and Debbie — you have been like family to me over the last few years. To my friends from before the PhD — Léa, Tom, Katherine, Helena, Marco and Kush — thank you for always being there when I needed time away from the madness of the PhD.

And finally, I would like to thank my family. My journey up to this point has been a long one, and you have supported me through all of it. To my parents — from a young age, you inspired a curiosity, a desire to ask questions and to solve problems, and a determination to see things through; without these, this piece of work would not exist. Thank you.

Abstract

This research aims to progress the development of real-time pedestrian simulation methods. Simulating pedestrian systems in close to real-time affords us up-to-date knowledge of how a population is distributed around a city. This may allow local authorities to react to demand for services that varies in time and space by engaging in data-driven decision-making approaches.

One of the most popular approaches for simulating pedestrian motion is Agent-Based Modelling. Such models are typically calibrated using historical data to ensure that they reflect the real system. Ultimately, however, these models usually incorporate some degree of randomness to emulate the variability of human behaviour resulting in the model diverging from the real system. This restricts their application to predominantly offline simulation.

Another potential approach to exploring pedestrian systems is the use of Big Data. Data are being generated in increasing volumes at an increasing velocity, and data regarding pedestrian flows are no exception. Such data, however, may offer sparse coverage with regards to time, space or demographic portions of the population — an issue from which Agent-Based Modelling does not suffer. Consequently, we seek an approach that allows us to benefit from both simulation and data. At present, there is not a consensus on how best to incorporate new observations into an Agent-Based Model whilst it is running.

The challenge of incorporating observations into running models is frequently tackled in other fields such as numerical weather prediction where meteorologists make use of data assimilation techniques to introduce observations into their models. This thesis, therefore, focuses on adapting the Ensemble Kalman Filter data assimilation technique to work with Agent-Based Models of pedestrian motion. Given the differences between Agent-Based Models and the models to which data assimilation methods are typically applied, additional challenges arise. This thesis will document some of these challenges and propose solutions.

Contents

List of Figures

List of Tables

Abbreviations

ABM	Agent-Based Modelling
DA	Data Assimilation
KF	Kalman Filter
EnKF	Ensemble Kalman Filter
UKF	Unscented Kalman Filter
PF	Particle Filter
GCS	Grand Central Station
FPS	Frames per second
PPM	Pixels per metre

Chapter 1

Introduction

With growing numbers of connected sensors being installed in cities, local governments are gaining access to a wealth of data (Psyllidis et al., 2015; Batty, 2019; Boeing et al., 2021). This could allow them to make more intelligent and transparent choices with regards to service provision for their citizens by engaging in data-driven decision-making processes (van Veenstra and Kotterink, 2017; Maffei et al., 2020). One of the key insights that can be gained from city-based sensors is an understanding of how people move around their environment.

When exploring this topic, investigators often make use of modelling techniques. At their most fundamental, models represent our understanding of the system that we are studying — an understanding that may not be perfect (Stanislaw, 1986). There exist modelling techniques for the simulation of how pedestrians move around urban spaces (Vermuyten et al., 2016). Simulating pedestrian behaviour is often undertaken at the micro-scale, with such models typically aiming to model at the individual level or on a spatially fine-grained grid (Burstedde et al., 2001). One of the most prevalent simulation methods in this field is that of Agent-Based Modelling. Such methods consist of two key components: agents and environments. In an Agent-Based Model, we prescribe sets of rules by which individuals interact with each other and their local environments (Macal and North, 2005); as interactions take place on the micro-scale, we typically observe the emergence of structure at the macro-scale such as crowding (Batty et al., 2003a) or lane formation (Liu et al., 2014). The evaluation of these rules often consists of stochastic elements aiming to emulate the variability of human behaviour. The introduction of such randomness, in conjunction with an imperfect understanding of the phenomena at play, however, typically results in simulation runs diverging from the real system. Modellers may attempt to prevent (or at least delay) this divergence by undertaking a development process that involves model verification (Xiang et al., 2005), validation (Crooks et al., 2008) and calibration (Thiele et al., 2014), as well as setting initial conditions based on historical data. These practices are appropriate for offline evaluations such as testing designs of new buildings (Chen and Zhan, 2014) or experimenting with different individual behaviours (Badham et al., 2018); however, when aiming to simulate events in real-time, this simply delays the inevitable divergence of the model from the real system. We may be particularly interested in the simulation of systems in real-time (or close to real-time) for a number of reasons:

- **Urban resource allocation:** The presence of pedestrians in urban spaces ultimately leads to an increase in the local incidence of crime, littering and noise. Local authorities may wish to respond to these issues, and an understanding of the number of pedestrians present allow for a proportionate response to problems (Chen et al., 2017).

- **Crowd safety:** In spaces where pedestrians gather in high density, there is a potential for stampeding and trampling which may subsequently lead to severe injury and loss of life (Helbing et al., 2007; Still et al., 2020). An improved understanding of how pedestrians are distributed across an urban setting may better inform the choice of evacuation and safety protocols.

- **Social distancing compliance:** Recent investigations have sought to estimate the density of pedestrians with a view to checking public compliance with social distancing measures put in place in response to the COVID-19 pandemic (Yang et al., 2021).

In such cases, we seek to replicate the events that occur in the real-world system within our model; in some cases we may even seek to produce accurate forecasts such that we can take pre-emptive actions to address issues that may occur. Given the likely divergence of models from real-world systems and the growth of uncertainty noted above, making such forecasts in a reliable manner is extremely challenging.

Given the demand for real-time insights, investigators may alternatively turn to big data. Data are now being generated in higher volumes and at greater velocity than ever before (Chen et al., 2014); however, there also exist issues with observation data from such systems. Whilst models typically allow us to simulate a whole system, observations are often sparse in either time or space (or both) (Zheng et al., 2014); this is to say that observations rarely provide complete coverage of the events. Beyond this, observations also bring their own associated uncertainties (Li et al., 2016) which depend on the method of data collection. Although our models may offer a solution to the problem of sparsity, they often introduce greater degrees of uncertainty as noted above. We therefore seek a solution whereby we can integrate up-to-date observations into our models as the models continue to simulate the system.

One of the methods by which we can combine knowledge represented by our model with observations as they become available is through data assimilation techniques, which are most commonly used in the field of numerical weather prediction (Kalnay, 2003). Such techniques are typically made up of two steps:

1. **Predict:** The model is run forward, estimating the state of the system,
2. **Update:** The model's estimate of the system state is combined with new data in the form of observations.

These steps are repeated iteratively in a cycle. The aim of incorporating the observations into the model is to improve the model accuracy with respect to the true system state, thus allowing us to place greater trust in our simulations whilst acknowledging the uncertainty in both our predictions and observations.

A large volume of work exists in which such techniques are applied to meteorological systems where the models used are based on differential equations (Navon, 2009); other work has focussed on the use of data assimilation schemes where real-time forecasting is of great value such as Tsunami forecasting (Maeda et al., 2015). Significantly less work exists in which data assimilation methods are applied to agent-based models — in particular pedestrian models. The application of data assimilation schemes to such models brings with it a set of challenges that may not be encountered in its traditional fields of application. One problem that may be encountered in traditional data assimilation fields of application is the estimation of unobserved quantities (Ruiz et al., 2013), e.g. model parameters. Such unobserved quantities are often continuous variables — something that data assimilation schemes are well suited to estimating. When applying data assimilation methods to pedestrian models, we are also likely to want to use the assimilated observations to infer unobserved quantities; however, in Agent-Based Models of pedestrian systems, it is likely that some of these variables will not be continuous, but instead categorical. This presents additional challenges.

The majority of the pre-existing work has focussed on implementing Agent-Based Models in conjunction with a data assimilation method known as the Particle Filter (Wang and Hu, 2013; Rai and Hu, 2013; Wang and Hu, 2015; Lueck et al., 2019; Malleson et al., 2020; Ternes et al., 2021). Other work has sought to apply variants of a data assimilation method known as the Kalman Filter (Ward et al., 2016; Clay et al., 2020, 2021). Each of these methods has their strengths and weaknesses (which will be touched upon in Chapters 2 and 4).

This thesis will build upon this novel and growing field of work by applying a data assimilation method known as the Ensemble Kalman Filter (EnKF) to Agent-Based Models of pedestrian motion.

1.1 Aim and Objectives

This investigation aims to define an approach for the application of the EnKF data assimilation method to Agent-Based Models of pedestrian motion for the purposes of simulating systems in real-time. In order to achieve this, the following objectives out set out:

1. Review the literature around the simulation of pedestrian dynamics with a particular focus on simulation at close to real-time.
2. Describe the modelling approaches used to simulate pedestrian systems in this investigation.
3. Describe the data assimilation approaches used to update models in this investigation.
4. Apply the EnKF to improve estimates of pedestrian locations in simple Agent-Based Models to show that such approaches can be used with systems other than those that are described by systems of differential equations.
5. Apply the EnKF to improve estimates of pedestrian locations in more realistic Agent-Based Models to show that such approaches are also effective when to systems that more closely reflect reality.
6. Apply the EnKF to improve estimates of both observed and unobserved variables in an Agent-Based Model to show that such approaches are effective in simultaneously performing state and parameter estimation for pedestrian systems.

1.2 Thesis Outline

This thesis is made up of 8 chapters (including this chapter). The way in which these chapters relate to the objectives described above is outlined in Table 1.1.

Following on from this chapter, Chatpter 2 aims to introduce the relevant literature for the investigation, providing a foundation on which to build the work contained within this thesis. This begins with an introduction to the literature on pedestrian dynamics and their simulation. Based on this, the chapter goes on to introduce Agent-Based Models and their use in the simulation of pedestrian systems; this includes topics

Objective	Chapters
O1: Review the literature around the simulation of pedestrian dynamics with a particular focus on simulation at close to real-time.	2
O2: Describe the modelling approaches used to simulate pedestrian systems in this investigation.	3
O3: Describe the data assimilation approaches used to update models in this investigation.	4
O4: Apply the EnKF to improve estimates of pedestrian locations in simple Agent-Based Models to show that such approaches can be used with systems other than those that are described by systems of differential equations.	5
05: Apply the EnKF to improve estimates of pedestrian locations in more realistic Agent-Based Models to show that such approaches are also effective when to systems that more closely reflect reality.	6
O6: Apply the EnKF to improve estimates of both observed and unobserved variables in an Agent-Based Model to show that such approaches are effective in simultaneously performing state and parameter estimation for pedestrian systems.	7

Table 1.1: Table outlining the relationship between objectives and chapters.

related to their calibration and the ways in which real-world data are used to inform their design and implementation. Criticism is provided of these approaches, outlining their shortcomings when looking to specifically simulate systems in real-time (or close to real-time). This chapter concludes with a review of the existing work in which data assimilation methods have been used to alleviated these shortcomings, whilst also providing critique.

Having reviewed the literature in Chapter 2, the next two chapters focus on the methods used in this investigation. The first of these — Chapter 3 — aims to outline the Agent-Based Modelling aspect of the work within this investigation. This involves describing two models which will be used in conjunction with data assimilation schemes. The first of these models is a toy model which seeks to simulate the movement of pedestrians across a fictitious train station concourse; when outlining this model, its shortcomings are also included. The second model looks to provide an application case which more closely represents a real-world setting, and therefore seeks to simulate the movement of pedestrians around the concourse of Grand Central Station in New York. This description includes a documentation of the model, an outline of the calibration process and a sensitivity analysis. The purpose of using these models is twofold. The first of these is that the models exhibit crowding phenomena — something that is typically not captured by modelling approaches that are based on differential equations. Furthermore, under the conditions of crowding, we are likely to find that our models are more liable to diverge from the corresponding real-world system at an individual level, thus justifying the case for applying data assimilation at an individual level. Secondly, the use of these models allows us to apply the data assimilation approach to the inference of unobserved agent parameters — a problem that, as noted previously, it not common when using these methods in their typical fields of application.

The next chapter focussing on methods — Chapter 4 — aims to outline the data assimilation methods used in this investigation. This starts by defining the process of data assimilation within the context of Bayesian inference. The chapter then goes on to describe the EnKF (and the Kalman Filter on which it was originally based). This is contrasted with other data assimilation methods, such as those that have been used in similar investigations. The chapter concludes by outlining the challenges of applying such data assimilation methods in conjunction with Agent-Based Models.

The following three chapters present the experiments and results of this investigation. These are centred around the idea of showing that the EnKF can be used in conjunction with an Agent-Based Model of pedestrian motion to improve the accuracy with which we can simulate the trajectories taken by pedestrians across an environment.

The first of these — Chapter 5 — looks to apply the filter to a toy model which represents the concourse at a fictitious station (which has been detailed in Chapeter 3). This acts as a preliminary experiment to explore the initial challenges of applying the data assimilation method to such an Agent-Based Model.

The second results chapter — Chapter 6 — seeks to take this a step further by applying the EnKF an Agent-Based Model that more realistically represents a real-world scenario; the model in question, known as `StationSim_GCS`, models the motion of pedestrians around the concourse of Grand Central Station in New York, as was designed based on empirical data (as outlined in Chapter 3).

The final results chapter — Chapter 7 — tackles one of the key issues that is overlooked in previous results chapters. In Chapters 5 and 6, it is assumed that we have perfect knowledge of pedestrians' origins and destinations within the model environment from the outset — this is seldom the case when simulating real-world systems. As a consequence, Chapter 7 seeks to explore the impact of relaxing this assumption, proposes a set of solutions, and explores their effectiveness.

This thesis is then concluded by Chapter 8 which draws together the findings of the investigation. This includes a comparison with other relevant pieces of work and an outline of the limitation of the work contained herein. This concludes with the suggestion for some avenues for future work.

Chapter 2

Literature Review

Having outlined the context, motivation and aim of this investigation in the previous chapter, this chapter aims to review the literature relevant to such a piece of work. In the previous chapter, we noted that this investigation seeks to establish a new approach to using Agent-Based Modelling to simulate pedestrian systems in close to real-time. In order to undertake such an investigation, we must first consider the literature surrounding the modelling of pedestrian dynamics more generally. This chapter therefore begins with a review of the literature on the general modelling of pedestrian dynamics; this will highlight the issues with previous approaches. This is followed by a specific focus on the use of Agent-Based Models and their use in the context of pedestrian dynamics; this section will highlight the need for the use of real-world data when modelling pedestrian systems. The chapter will conclude with a section reviewing the recent work that has been undertaken with "online" approaches to using agent-based models in the field of pedestrian dynamics. This will highlight the shortcomings of previous attempts, with a view to justifying the need for this investigation.

2.1 Pedestrian Dynamics

As highlighted in Chapter 1, the investigation of pedestrian movement is of great interest for many reasons; this interest may come from a number of different domains — from urban planners and government officials to retailers and advertising agencies (Kitazawa and Batty, 2004). With increasing numbers of people living in cities, this interest is only likely to grow.

When considering the dynamics of pedestrian systems, we are able to explore them by looking at quantitative qualities and qualitative phenomena. Each of these shall be described in this section, along with an overview of the modelling techniques typically used when considering pedestrian systems.

2.1.1 Quantitative Measures

In pedestrian dynamics, there are often three quantitative measures that are of interest: pedestrian flow rate, pedestrian velocity and pedestrian density (Gupta and Pundir, 2015).

For pedestrian velocity, v, we can consider the classical definition of velocity:

$$\mathbf{v} = \frac{d\mathbf{x}}{dt}, \tag{2.1}$$

where $\mathbf{x}$ is the pedestrian's position and t is time, i.e. a pedestrian's velocity is the rate of change in its positions with respect to time. If we are only interested in a pedestrian's scalar velocity, we can take the magnitude of the vector, $v = \|\mathbf{v}\|$. If we consider the case where the pedestrian's is defined by its location in a 2-dimensional x-y space, then we can define the scalar velocity as:

$$v = \|\mathbf{v}\| = \sqrt{\mathbf{v}_x^2 + \mathbf{v}_y^2}, \tag{2.2}$$

where $\mathbf{v}_x$ is the x-component of the pedestrian's velocity and $\mathbf{v}_y$ is the y-component of the pedestrian's velocity.

For pedestrian density, we can also consider the classical physical definition of density, ρ:

$$\rho = \frac{m}{V},$$

where m is mass and V is volume. This classical definition is typically applied to the description of 3-dimensional matter; in our case, we are interested in the description of pedestrians in 2-dimensional space and the definition is therefore adapted to:

$$\rho = \frac{N}{A}, \tag{2.3}$$

where A is the space being considered and N is the number of pedestrians present in the space. Whilst this approach is commonly used, it only provides an average density over a space and does not provide a spatial distribution which we are likely more interested in. Furthermore, the result of the calculation is dependent on the geometry of the reference area used. An alternative approach is the use of Voronoi cells (Steffen and Seyfried, 2010). These cells represent the physical area "belonging" to each pedestrian. A single pedestrian's cell is defined as containing all coordinates that are closer to them than to other pedestrians. Each cell, c_i, has a respective area, A_i, from which we can derive a density, ρ_i:

$$\rho_i = \frac{1}{A_i}. \tag{2.4}$$

This results in a collection of cells which partition the spatial environment, each of which pertain to a single agent in the system. Each of these cells have an associated density which can be graphed to visualise the distribution of density by tiling the space. Such a tiling is almost always irregular, but the respective density distribution on a regular grid can be found by considering the weighted average of each of the Voronoi cells which intersect with a given grid cell.

For pedestrian flow rate, J, we consider the scenario in which we are observing a single point in space and counting the number of pedestrians passing by. Based on such a situation, we define the flow rate, J, as the number of pedestrians passing the point over a given time:

$$J = \frac{dN}{dt}, \tag{2.5}$$

where N is the number of pedestrians and t is the time.

Following the analogy with classical physical definitions for these quantities, we may consider the relationship between the three quantities often observed in fluid dynamics (Schadschneider and Seyfried, 2011):

$$J = \rho v. \tag{2.6}$$

Consequently, we may only need to measure two of the three quantities and may be able to derive the third from this relationship. These relationships are often explored graphically in the form of fundamental diagrams (Seyfried et al., 2005), and have been used to explore the way in which pedestrian behaviour varies across cultures (Chattaraj et al., 2009).

2.1.2 Qualitative Phenomena

When considering pedestrian systems, there are a number of phenomena that occur persistently across a number of settings, both at a microscopic and macroscopic level.

When considering microscopic phenomena, Helbing et al. (2001) observed the following set of behaviours occurring across different settings:

- Pedestrians dislike taking detours and wish to take the most direct route to their destination.
- Each pedestrian typically has their own desired walking speed, and the speeds amongst a population are usually normally distributed.
- Pedestrians have a desire to maintain a certain distance between themselves and others (as well as objects within the environment).
- Pedestrians often act “automatically” and have a tendency not to change their behaviours.

Whilst their may be some variation in the extent to which each of these phenomena occur, it is important that they are incorporated on some level into modelling approaches that seek to faithfully represent crowd dynamics.

When considering the dynamics of pedestrian systems, there often emerge a number of different macroscopic phenomena based on the individual-level behaviours of pedestrians between each other and between pedestrians and their environment. Three of the most common phenomena observed are lane formation, queuing, and density waves (Schadschneider et al., 2009). These shall be discussed briefly in this section.

Lane Formation

When considering pedestrians systems set in narrow corridors or pedestrian crossings, the direction of travel of the individuals becomes restricted in one of the two-dimensions with the pedestrians seeking to cross from one side of the environment to the other. If all of the pedestrians are starting on one side and aiming to cross to the other side, we may consider the flow to be unidirectional. If, however, the population is split into two groups and are initially located on opposite side, seeking to cross the environment, we observe what is known as a bidirectional or a counterflow. When observing situations involving bidirectional flows we often see the formation of lanes (Kretz et al., 2006). If we consider a simple setting in which we have a long corridor spanning in the x-direction and narrow in the y-direction with some portion of a pedestrian population starting at each end with the goal of getting to the opposite side, we expect that ultimately we will observe the formation of number of lanes such that at least one lane facilitates the movement of pedestrians in the positive x-direction and at least one lane facilitates the movement of pedestrians in the negative x-direction. Such phenomena may occur naturally or may be aided by environmental features such as dividing barriers and signage (e.g. Green Park underground station in London).

Queuing

Another phenomenon that occurs commonly in pedestrian systems is that of queuing. Queuing can be framed as a demand-supply problem (Martinez-Gil et al., 2017), using the terms defined in Section 2.1.1:

> Consider a scenario in which we have a unidirectional flow of pedestrians in the x-direction. As described by Helbing et al. (2001), pedestrians

> have a set of demands — for the speed at which they walk, v, and for a degree of personal space around them (which translates into a specific local density, ρ). These demands govern the speed and density of a population of pedestrians. If the stream of pedestrians encounters a narrowing of the path that is sufficiently constricting that it violates their demands for personal space, the flow rate will fall. With the imbalance between the rate at which pedestrians can pass through the constriction and the rate at which pedestrians arrive at the constriction, a queue forms.

This scenario describes the demand-supply problem as one based on cross-sectional space, but the same process can apply for supply and demand of other resources such as ticket counters at train stations. Furthermore, the example above describes a scenario based on a single resource-supplier, but this can also be extended to multiple resource-suppliers; such a scenario may result in multiple queues, and present arriving pedestrians with multiple choices (Wagoum et al., 2017).

Density Waves

The final phenomenon that we shall consider is that of density waves. Density waves are common in vehicular traffic systems, and it is not uncommon to observe them in pedestrian systems which emulate the conditions of vehicles driving on the road, i.e. unidirectional flow with restriction in movement tangential to the flow of the crowd (although they may also occur in other settings such as stationary crowds). We can think of density waves in pedestrian systems as periodic oscillations of density in space and time (Schadschneider et al., 2009), $\rho(\mathbf{x}, t)$.

2.1.3 Modelling Approaches

When investigating pedestrian systems, many researchers seek to define models of the dynamics at play. When building these models, researchers need to make decisions regarding a number of factors (Schadschneider et al., 2009; Zsifkovits and Pham, 2017):

- **Microscopic or macroscopic?** Do we wish to to model the system at the individual or at the group level?
- **Discrete or continuous?** Do we treat the state variables of space and time as discrete integer values or continuous real values?

- **Deterministic or stochastic?** Is the future state of the system determined precisely by the current state of the system, or is there some random variability involved?

Two of the most common way in which to model pedestrian systems are through mechanical models and through cellular automata models (Martinez-Gil et al., 2017). Beyond these methods, researchers also used Agent-Based Models. This investigation will focus on the use of Agent-Based Models, and as such a section dedicated to them will follow this one; this section will proceed by outlining mechanical models and cellular automata.

Mechanical models typically involve the use of mathematical equations to describe the evolution of a system. Some of these seek to use equations to describe the time-evolution of the macroscopic state of the system, such as fluid dynamic models (Helbing, 1992); others seek to use equations to describe the forces acting on pedestrians at an individual level (Helbing and Molnar, 1995). Such approaches are typically deterministic, and often treat the state variables as continuous (by virtue of being governed by equations using real-valued variables).

Cellular Automata model the system at a microscopic level based on discrete space and time. They consider the environment as a grid of cells with each cell containing a number of pedestrians, and evolving the number of pedestrians in each cell from one time-step to another based on a set of rules (Schadschneider, 2002). The rule by which the number of pedestrians in a cell is evolved is often stochastic (although it may be deterministic), and aims to reflect the movement of pedestrians from one discrete space to another. A related technique which also simulates a system at a microscopic level is that of Agent-Based Modelling, which shall be explained in the next section.

2.2 Agent-Based Modelling

In the previous section, we touched upon the topic of Cellular Automata models which seek to model the evolution of pedestrian systems at a microscopic level. Related to Cellular Automata, we have Agent-Based Models. In the former, we consider space to be our unit of analysis, modelling how discrete units of space vary over time; in the latter, we consider individual pedestrians as our units of analysis, modelling how they move through space over time.

When considering social systems, we can think of them as being complex systems, comprised of many interacting components. When looking to study such systems, we can often think of modelling them from one of two perspectives: top-down and bottom-up. In a top-down approach, we seek to model aggregate-level system characteristics, which might be achieved by approaches such as the equation-based models mentioned in Section 2.1. Such approaches, however, do not provide us with any information regarding the status of individual components within the system. Furthermore, they are liable to overlook individual-level heterogeneity.

The alternative approach — bottom-up — aims to model individual components, defining rules for their interactions. This allows us to derive the macroscopic behaviours that we could observe from a top-down approach. Furthermore, it allows us to incorporate heterogeneity in terms of variation in individual-level characteristics and behaviours and explore their impact on the emergence of macroscopic states. One of the most popular approaches to bottom-up modelling is Agent-Based Modelling in which we characterise a system as a collection of units which interact with each other and their environment based on a set of behaviours (which are often prescribed as rules). Here, we take environment to include not just spatial environments but also other factors which also mediate interactions between agents such as interaction networks. This approach has been applied to a wide range of fields including exploring molecular and cellular interactions in Pharmacology (Sun et al., 2008; Cosgrove et al., 2015), the analysis of movement and habitat selection in Ecology (Railsback et al., 1999; Railsback and Harvey, 2002), the exploration of criminological theory (Birks and Davies, 2017), the simulation of transport networks (Bazghandi, 2012; Bazzan and Klügl, 2014) and the exploration of market dynamics (Ghoulmie et al., 2005; Eppstein et al., 2011; Ge, 2017). With the broad array of disciplines to which Agent-Based Modelling is applied comes an array of ways to encode the environment in which the agents exist. When considering ecological applications, our agents exist in a spatial environment which may be represented as either continuous or discrete space; when using an Agent-Based Model to explore the impact of pandemic interventions, we may wish to incorporate a network structure between our agents (Vermeulen et al., 2021).

Such an approach which allows us to simulate a system at an individual level lends itself naturally to the modelling of pedestrians, and the simulation of pedestrian systems using Agent-Based Models is a well established field (Burstedde et al., 2001; Hoogendoorn and Bovy, 2003; Helbing et al., 2005). In applying the technique of agent-based modelling to the study of pedestrian dynamics, we characterise individual people as agents in the model, and define the rules of how they may interact with each other and the environment around them (Batty et al., 2003a; Liu et al., 2014). Such an approach allows us to explore not only the impact of different spatial environments, but also how the dynamics of the system may change as a result of different agent characteristics and behaviours — something that may have been much more difficult if using a top-down approach. This allows us to apply agent-based modelling to the exploration of scenarios such as crowding dynamics at carnivals (Batty et al., 2003a,b), the testing of evacuation policies (D'Orazio et al., 2014; Chen and Zhan, 2014) and usability of new building designs (Andrews et al., 2011).

In order to ensure that these models accurately reflect the real-world scenarios that they are simulating, modellers undertake the aforementioned processes of verification, validation and calibration. As shall be highlighted in Section 2.2.1 a number of different approaches to calibration exist, each with their own strengths and weaknesses. These methods typically make use of historical data regarding the system being studied in order to find appropriate initial model states and parameters.

2.2.1 Offline Agent-Based Modelling

Traditionally, Agent-Based Models are used in the manner outlined above whereby the model is calibrated before use. This calibration is typically undertaken once before simulating the system in question, and the model remains static from that point onwards. Such an approach is referred to as "offline". The alternative is an "online" approach, which is typically used in scenarios in which researchers would like to simulate systems in close to real-time and have their models respond to observed changes in the system (such as when using models in smart cities applications).

The distinction regarding when to use online or offline approaches depends upon both which approach is feasible and which is desirable.

From a feasibility perspective, we should note that online approaches place requirements on both computational infrastructure and the data used. With regards to data, online approaches require the provision of low-latency observations of the system being modelled. With regards to to infrastructure, online approaches are likely to incur a greater computational cost beyond what is typically found when applying offline approaches. This places technical restrictions upon the scenarios in which online approaches may be applied.

Even in the situation in which we have the necessary observations and the infrastructure to handle the computational cost, we may find that an online approach is not appropriate and that an offline approach is more desirable. This may include situations in which researchers wish to explore the impact of different architectural designs on the behaviours of the individuals in a system (Wu and Chen, 2019). In such scenarios, researchers are not aiming to make forecasts (as would likely be the case with online approaches), but instead to use models as a test-bed to explore the impact of different policies or individual-level behaviours.

The remainder of this section will focus on the types of calibration methods used in offline approaches. Literature on online approaches shall be addressed in Section 2.3.

Calibrating Agent-Based Models

The process of calibrating an agent-based model involves identifying the values that should be assigned to model parameters in order to achieve the desired model behaviour (Crooks et al., 2008; Crooks and Heppenstall, 2012); the desired model behaviour is typically defined as behaviour which matches that observed in previous observations of the system in question by some measure. Such parameters govern the ways in which agents interact with each other and with their surrounding environment — it is, therefore, crucial that appropriate values are identified. In order to achieve this, there are a wide range of approaches at our disposal (Hazelbag et al., 2020) which often require that we evaluate model outputs against observations of the system by way of measuring the difference between the two.

One of the most primitive methods of model calibration is known as full factorial design, whereby a comprehensive parameter search is undertaken by running the model for every combination of parameter values (Thiele et al., 2014), evaluating the outputs of the model against the desired behaviour for each run. This method is appropriate when model runs are relatively inexpensive with regards to compute time, and when parameters take integer values; in cases where parameters take on non-integer values, care must be taken with regards to the grid resolution of the parameters that are tested as this may significantly change results.

As an alternative to the exhaustive search proposed in Full Factorial Design, Classical Sampling Methods involve sampling from the set of potential parameter combinations. In its most simple form, this would take the form of randomly sampling from uniform distributions for each parameter value; this is also relatively inefficient though (Thiele et al., 2014). Alternative sampling approaches have therefore been proposed such as Latin Hypercube Sampling (McKay et al., 1979), whereby the potential range of parameter values is divided up into equally probable intervals and samples drawn from each interval.

Optimisation Methods are a common technique for fitting models. They involve defining a cost function relating some aspect of model behaviour with the corresponding aspect of the observed phenomena; we then seek to minimise the difference (i.e. the cost). These approaches can therefore be viewed as minimisation problems over the parameter space in which we often seek the global minimum, i.e the parameter values which minimise the difference between the model behaviour and the observed system behaviour. Many techniques exist to achieve this goal, including simulated annealing (from statistical physics) (Kirkpatrick et al., 1983) and evolutionary algorithms (Duboz et al., 2010).

A final approach to calibration may be through Bayesian Methods. These methods seek to use Bayes theorem to identify appropriate parameter values (Jabot et al., 2013). This is achieved by making use of our prior understanding of the statistical distributions from which each of the parameter values are drawn; running a model a large number of times with parameter values drawn from these distributions, comparing our resulting model outputs with our observations with a view to inferring the true statistical distributions of the parameters. Due to the large number of times that models must be run, these methods can be computationally expensive; with the advent of modern computing, bringing with it increasing levels of computational power, the methods are gaining popularity with researchers (Beaumont, 2010; van der Vaart et al., 2015).

More generally, we may have to contend with some issues. In some cases, it is not possible to identify unique values for parameters, and instead we are left with parameter sets, i.e. multiple combinations of parameters that can be used to reproduce the desired system behaviour; this can result from over-parameterisation and/or interactions between parameters (Gan et al., 2014). This process of model calibration may also be further impeded by either a lack of data or data with high uncertainty (Thiele et al., 2014).

As highlighted in Section 1, however, these approaches are typically undertaken prior to models being run. As a consequence, models employing these approaches may attempt to make use of up-to-date observations in the calibration process prior to running, but cannot make use of subsequent observations that arise as the model runs, and as such are not appropriate for simulating systems in real-time.

2.2.2 Challenges of Agent-Based Modelling

Before proceeding, it should be noted that Agent-Based Modelling is not without its shortcomings (apart from the issue of incorporating real-time observations described above). In recent years, a number of articles have described at length the challenges faced by those using Agent-Based Models (Crooks et al., 2008; Schulze et al., 2017; An et al., 2021). This section, therefore, will briefly touch on a small selection of issues that are faced which are particularly relevant to the work in this investigation. This will include a discussion of challenges around the modelling of human behaviour, the problems arising from the computational cost of running such models, the issues of model verification and validation and the inadequate use of data in the model development process.

Modelling Human Behaviours

One of the most common reasons that Agent-Based Models are used to simulate systems of humans is that they allow researchers to incorporate more complex and heterogeneous behaviours (Macal, 2016). One way in which researchers often aim to incorporate some aspect of human behaviour is though the use of computational randomness; this is not, however, to represent human behaviour as random, but instead to represent the degrees of inconsistency and noisiness associated with the behaviour (Kennedy, 2012).

Deciding how human behaviours are represented and prescribed within a model is a challenge. In order to address this problem, a set of frameworks have been devised to better represent human behaviour (Crooks and Heppenstall, 2012). One of the most prominent of these is Belief-Desire-Intention (BDI) (Rao et al., 1995), in which an individual's beliefs, desires and intentions are encapsulated within the state of the corresponding agent; beliefs represent the individual's information about the environment in which they exist, desires represent the individual's goals, and intentions represent the set of actions that the individual selects in order to achieve their goals (Bandini et al., 2009). By encapsulating an individual's internal state and thought process in this way, researchers can design corresponding agents that have consistent beliefs and respond to stimuli in a timely and orderly manner (Kinny et al., 1996).

One of the criticisms of this framework is its assumption that the individuals who we seek to model are rational actors — an assumption which does not always hold. In order to address this issue the PECS framework (Physical Conditions, Emotional State, Cognitive Capabilities and Social Status) was devised in which researchers can include emotional and social factors into the characterisation of an individual within their model (Schmidt, 2005).

The development of such frameworks may seek to clarify and better formalise the way in which we encode human behaviour within our computational agents. Regardless of the choice of framework, however, models of human systems will typically fail to perfectly reflect the system as, ultimately, we are unlikely to be able to account for all types of behaviour in all settings (whether this arises from the finite description of behaviour in our model, or from limitations of the data with which we calibrate the model) (Kennedy, 2012; An et al., 2021).

Verification and Validation

The issues of verification and validation of Agent-Based Models are ones that have been raised repeatedly (An et al., 2021). It has been suggested that this may arise due to some confusion around their definitions (Crooks et al., 2008). When considering Agent-Based Models, we may consider the following definitions:

- **Verification:** The process of ensuring the logical implementation of the model is correct (Xiang et al., 2005).
- **Validation:** The process of ensuring that the model represents the system and phenomena of interest (Crooks and Heppenstall, 2012).

When undertaking model verification, we seek to ensure that the model that has been implemented is consistent with the researcher's conceptual model. If constructing the model through the writing of computer code, verification process may be achieved by undertaking a test-driven development process (Collier and Ozik, 2013). Such testing involves ensuring that some aspect of the model produces a prescribed set of behaviours. Testing may be undertaken at a range of different levels including Unit, Agent and

Integration (Nguyen et al., 2009). Reproducibility of behaviours (and by extension model verification by testing) in Agent-Based Models can be something of a challenge, particularly when the models incorporate some degree of stochasticity (Houhamdi, 2011).

When undertaking model validation, we seek to ensure that the model is capable of reproducing patterns and phenomena that we observe in the real system. This introduces a challenge that is intrinsically linked to one of the strengths of the approach; Agent-Based Modelling allows us to explore the behaviour of system at both the microscopic level and the macroscopic level, but at what level should we check that a model reproduces the behaviours of the real system? One suggested approach is that of pattern-oriented modelling (An et al., 2020), whereby the output model behaviour can be compared against multiple patterns at multiple scales.

Whilst some approaches are proposed for dealing with each of these challenges, there are not yet agreed-upon solutions and as such they remain open problems. Without the application of these processes, we may question the extent to which a model is credible and robust.

Computational Cost

One of the most notable disadvantages of Agent-Based Models in comparison to methods such as Equation-Based Models is the computational cost. In the former approach, we seek to evolve the state of each individual unit in our model over time — a process that may be either deterministic (e.g. depending on some deterministic rule or equation) or stochastic (e.g. a rule or behaviour depending on some random element). In the case of the latter approach, we may have a set of initial conditions, and propagate them forward in time based on a set of differential equations. The dimensionality of the latter is likely to be much smaller than the dimensionality of the former, and as such we typically expect individual-level simulation methods such as Agent-Based Modelling to be much more computationally expensive (Bonabeau, 2002).

In addition, we should note how we typically run Agent-Based Models. In Section 2.2, we noted that ensembles of Agent-Based Models are often run with a view to considering the average behaviour. This adds an additional layer of time complexity, further increasing the computational cost of running such models. There have been recent developments which have improved computational capacity such as novel parallelisation approaches (An et al., 2021) and the use of GPU platforms (Lysenko and D'Souza, 2008), but this remains a challenge.

Inadequate Use of Data-Driven Approaches

Whilst there exist a number of ways to calibrate Agent-Based Models based on empirical data (as seen in Section 2.2.1), it has been noted in recent times that a large number of Agent-Based Models still do not adequately use data to ensure that they produce system behaviours that reflect the real-world system (Kavak et al., 2018). Such issues may include the use of agent-behavioural rules based exclusively on theory, arbitrary manual selection of parameters or qualitative validation (Zhang et al., 2016). The practices may be reflective of a theory- or knowledge-based modelling approach (Keller and Hu, 2016). Whilst this may not be as much of a problem for models which aim to be more abstract (perhaps with a view to providing test scenarios for theory development), this likely poses a substantial problem in cases where models seek to inform decisions. There is, therefore, a growing movement towards data-driven modelling practices. Such practices involve using data to build the models themselves, not just calibrate them — an idea that is common in fields like machine learning (Keller and Hu, 2019). This has included the use of genetic algorithms to perform optimisation and sensitivity analysis (Oloo and Wallentin, 2017), the use of rich spatial data sources in conjunction with machine learning methods to improve the quality of crime pattern prediction (Rosés et al., 2021) and improving the fidelity with which an Agent-Based Model can emulate phenomena observed in video data by optimising for both microscopic and macroscopic patterns (Zhong et al., 2015).

2.3 Online Agent-Based Modelling

As touched upon in Section 2.2.1 the process of developing an Agent-Based Model typically involves some form of model calibration and initialisation. Whilst there are a large number of different manners in which we can calibrate such models these are typically undertaken once prior to running the model. In some situations, we aim to simulate events in real-time (or close to real-time). This would, however, require that we stop the simulation, undertake the calibration and initialisation steps, and restart the model, hoping that this might reflect our updated knowledge regarding the state of the system and the parameters governing the dynamics within it. This is often impractical. We therefore seek an approach that allows us to incorporate observations of the system whilst simulating the system.

One approach to dynamically incorporating observations into an Agent-Based Model of a system is known as Dynamic Calibration.approach to dynamically incorporating observations into an Agent-Based Model of a system is known as Dynamic Calibration. Whilst the approaches to calibration noted in Section 2.2.1 are applied once before running, Dynamic Calibration approaches seek to update the model parameters in question in response to new observations. This may be achieved using approaches such as Genetic Algorithms to optimise model parameters as streamed observations are received (Oloo and Wallentin, 2017).

Whilst Dynamic Calibration may improve on the offline static approaches outlined in Section 2.2.1, it only seeks to obtain correct estimates for the model parameters which characterise agent behaviour. When simulating systems in close to real-time, it is also important that we gain improved estimates of a system's state (Malleson et al., 2020). Improving estimates of a system's state is a problem often tackled with data assimilation methods, and therefore the following section will focus on the application of data assimilation methods to Agent-Based Models.

2.3.1 Data Assimilation

As touched upon in Chapter 1, data assimilation is an approach by which we can combine the knowledge that we hold regarding a system in the form of a model with knowledge that we have gained in the form of observations. The rationale behind this is that the models that we use typically represent our incomplete knowledge of a system, and as such these models are "incorrect" in some manner. As a consequence, the models are expected to make predictions of the system that that diverge from what is happening in the system. We therefore make use of observations by using data assimilation to correct for the errors produced by the models.

We may consider, as an example, a scenario where we are modelling the motion of a pedestrian walking along a straight pavement section in the x-direction. In such a model, we may characterise the pedestrian's motion based on the equation:

$$x_i = x_{i-1} + v\Delta t,$$

where x_i is the pedestrian's position at the ith time-step, x_{i-1} is the pedestrian's position at the previous time-step, v is the pedestrian's velocity (a parameter of the model), and Δt is the length of time that passes between time-steps. Given a fixed value for the velocity, v, this model predicts the pedestrian proceeding along the pavement in a linear fashion. It may be, however, that the pedestrian's velocity is not fixed and varies over time and space (perhaps in response to obstacles on the pavement or other pedestrians). This will result in the positions that our model predicts being incorrect. If we are provided with observations of the pedestrian's position at some point, we may use these observations to update the modelled state. As a specific example, if the model predicts that pedestrian will have moved 10 metres along the pavement by the 15th time-step and we receive an observation indicating that they have only moved 5 metres, we may update the modelled state to reflect this. There are a variety of approaches by which we can perform this update; they are often categorised as either sequential data assimilation approaches or variational data assimilation approaches — these terms shall be elaborated on in Chapter 4.

Of the work that currently exists wherein investigators attempt to apply data assimilation schemes to pedestrian agent-based models, most make use of sequential schemes, particularly Kalman Filter-based approaches (Kalman, 1960; Evensen, 2003; Julier and Uhlmann, 1997) and the Particle Filter (Arulampalam et al., 2002). Of these approaches, there is a noticeable similarity between the Ensemble Kalman Filer and the Particle Filter: both methods maintain ensembles of the same model, but differ in how they update the models upon receipt of new observations. It is also worth noting that there are some differences in terminology between the two: in the context of the Ensemble Kalman filter an individual model is labelled as an ensemble member whereas in the context of the Particle Filter it would be labelled as a particle. The internal workings of data assimilation schemes will be discussed in full in Chapter 4.

This section will go on to explore the work that has been done with both types of approaches and highlight potential shortcomings.

Kalman Filter-based Approaches

Kalman Filter-based approaches have sought to use two different variants of the Kalman Filter: the Ensemble Kalman Filter (EnKF) and the Unscented Kalman Filter (UKF).

Previous work focusing on implementing the EnKF in conjunction with an urban agent-based model has modelled pedestrians arriving and departing the city centre of Leeds, UK (Ward et al., 2016). Whilst this investigation displays that the EnKF can be implemented in conjunction with an agent-based model to successfully improve the model's prediction accuracy, it suffers from a number of shortcomings. First and foremost, it should be noted that the models used for the investigation are very simple in comparison to the majority of agent-based models, with one of the models used being a binary state model, with each agent either being in the city or not in the city. Such a model may be of use in gaining a picture of how the number of people in a city varies over time, but does not provide any additional information with regards to their spatial distribution within the city. The inter-agent interaction governing their transition between the two states is global — this is to say that each agent's decisions are based on the state of every other agent without considering more intricate mechanisms of attraction and repulsion between agents (Helbing and Molnar, 1995) such as spatially local ones. Whilst the second experiment seeks to include a richer set of behaviours by further segmenting the agents, it still fails to include any spatial aspect, with agents again able to interact in a homogeneous fashion.

One of the limitations of the EnKF is that is assumes that the probability distributions, i.e. the distributions representing the state variables and observations, are Gaussian; under this assumption, the filter provides a exact estimate of the posterior distribution. This assumption, however, does not always hold. One of the solutions to this issue is the use of the Particle Filter, which is capable of providing as exact estimate of the posterior distribution without relying on the Gaussian assumption. Work in which the Particle Filter has been used will be touched upon in the next section.

The EnKF has also been applied to Agent-Based Models that aim to simulate more realistically the individual-level spatial interactions of pedestrian by Togashi et al. (2020). In this investigation, the researchers considered two scenarios: one in which pedestrians travelled unidirectionally along a corridor, and another considering rotary motion within a circular environment. This investigation, however, makes no effort to use the filter and model in an "online" manner (i.e. it does not aim to assimilate data in real-time), instead simply using the EnKF to perform parameter estimation and model initialisation. Whilst this is a promising step towards a data-driven approach, it does not endeavour to handle real-time streamed data sources. Furthermore, when considering the accuracy of their simulations, the investigators seek to compare simulation results with observations, suggesting that they have not considered the errors associated with observations; observations are derived from the true system state by means of measurement apparatus which produce some degree of noise.

Beyond the works outlined above, the EnKF has also been applied to epidemiological Agent-Based Models (Cocucci et al., 2021). In this investigation the filter was used to assimilate aggregated infection numbers into an individual-level model. As part of this work, some novel disaggregation approaches are introduced. One of the issues with this research, however, is its treatment of spatial interactions — the investigators attempt to encode a spatial element into the interactions between agents by assigning them neighbourhoods and using a matrix to define the level of connectivity between neighbourhoods which influences the chances of agents interacting across different neighbourhoods. Whilst this may perform well on an aggregate level (the level with which this investigation is primarily interested), it makes no guarantees about the accuracy of simulation at an individual level.

Aside from the EnKF, another variant of the Kalman Filter — the Unscented Kalman Filter — has been applied to Agent-Based Models of pedestrian systems in two scenarios.

In the first of these scenarios, the Unscented Kalman Filter is applied to the toy model outlined in Section 3.1 (Clay et al., 2020), and the filter is used to assimilate period individual-level location data. The aims of the investigation are to explore the effectiveness of the filter in improving simulation accuracy, and to explore whether the filter is able to infer trajectories for all pedestrians if only provided with observations regarding a portion of the population. This investigation showed the Unscented Kalman Filter to be successful in both endeavours; however, the model used is rather simplistic (as shall be discussed in Chapters 3 and 5).

In the second of these scenarios, therefore, the Unscented Kalman Filter is applied to the model described in Section 3.2, and again is used to assimilate periodic individual-level location data regarding pedestrian trajectories as they move across the environment (Clay et al., 2021). In this case, the primary aim is to demonstrate the effectiveness of the filter in conjunction with a more realistic model of pedestrian motion. Furthermore, the investigation seeks to introduce a new method — the Reversible Jump Unscented Kalman Filter. This approach is designed to consider scenarios in which we have agent parameters that are unknown at the outset of the simulation and are not observed over the course of the simulation. This additional aim of this investigation is, therefore, to infer both pedestrian locations (which are modelled and observed) and pedestrian destinations (which are unknown at the outset and are not observed). Once again, the Unscented Kalman Filter was shown to be effective on both fronts.

In both cases, it should be noted that the Unscented Kalman Filter (being a variant of the Kalman Filter) also suffers from the assumption of Gaussian error distributions just as with the EnKF. Furthermore, the filter requires the calculation of a matrix square root — an operation which can be expensive. Therefore, whilst the filter was shown to be efficient in these scenarios, issues of computational cost may be encountered for systems with much higher dimensionality.

Particle Filter-based Approaches

Some work making use of the Particle Filter has focused on simulating a smart office environment (Wang and Hu, 2013, 2015) with a view to using real-time data in conjunction with an agent-based simulation to provide more accurate estimates of the occupancy of the environment. These investigations model the office environment as a two-dimensional continuous space segmented by corridors, monitored by binary sensors which provide observations regarding the presence of agents at given locations in the environment. Whilst this investigation was also successful in showing that its chosen data assimilation method could be used to improve the prediction accuracy of the agent-based model, it was only run for small agent populations. This raises questions regarding the scalability of such a method when compared to the EnKF (which typically requires smaller ensembles of models to successfully perform assimilation when conditions such as the Gaussian assumption are met (Hoteit et al., 2012; Zhou et al., 2006)).

Other work making use of the Particle Filter has focused on simulating the motion of pedestrians across a train station (Malleson et al., 2020). This investigation similarly modelled agents moving in two-dimensional continuous space, but with a simpler spatial set-up; agents traverse the rectangular space from entrances on the left-hand side to exits on the right-hand side in a largely linear fashion, with observations being provided in the form of synthetic data regarding each agent's coordinates in the environment. Once again, this investigation shows the method to be successful in improving the effectiveness with which the system can be modelled. The investigation, however, suffers from a number of shortcomings: the model used is simplistic, and the data used are synthetic. It is, therefore, not clear whether this provides sufficient evidence regarding whether such a method could be successfully implemented with a real-world set-up. Furthermore, this investigation assumes greater knowledge than is reasonable: when pedestrian models are run which allow agents a choice of destinations, we seldom know which destination an agent intends to head towards at the outset — something which is not considered in this work. This can have serious ramifications for the decisions (and subsequent trajectories) of agents in scenarios where potential destinations are at discretely different locations in the environment.

This highlights a shortcoming of the Particle Filter: considering each agent's destination as a model parameter at the outset, the Particle Filter is unable to reliably update such parameters, and thus relies on the correct parameter values being provided when it is initialised. This is a result of how the Particle Filter functions. When assimilating data, the Particle Filter produces a new collection of particles (i.e. ensemble members) by sampling from the old collection of particles based on weightings derived from each particle's relation to the observations. As a consequence, when assimilating data, the Particle Filter can only produce states which already exist in its previous collection of particles. In cases where none of the particles contain correct information regarding an agent's target destination, the Particle Filter then has no chance of correctly estimating the correct destination through sampling. This issue may extend to other parameters which are not either not observed or not updated by the model process.

Some follow-up work based on the investigations focussing on implementing the Particle Filter in conjunction with a model of a smart office environment (Wang and Hu, 2013, 2015) has attempted to deal with this issue, using a Hidden Markov Model to identify discrete agent behaviour patterns (Rai and Hu, 2013). This, however, does not alleviate the questions that have arisen regarding scalability; in fact, the inclusion of an additional process in the data assimilation method may result in a further increase in computational complexity.

The most recent work focussing on the use of Particle Filter has also sought to address this issue. In this case, the Particle Filter is applied to the model outlined in Section 3.2, and they consider the case where the destinations and speeds of the pedestrians are unknown at the outset of the simulation, and observations of the pedestrians' locations are assimilated periodically (Ternes et al., 2021). Furthermore, this investigations seeks to apply the filter by using real-world data whereas the preceding investigation used synthetic data (Malleson et al., 2020). In this investigation, the researchers encounter the problem whereby particles with correct unobserved parameter values are sampled out and subsequently the ensemble of particles is unable to correctly simulate the system dynamics. They propose a solution whereby the resampling process (explained in Section 4.4.1) only updates the locational aspect pertaining to each agent in the particle. This was found to perform about as well as the scenario in which the unobserved parameters where unknown with no data assimilation being performed. Once again, this highlights the issue that the Particle Filter may sample out a desirable particle and may not have any way to reintroduce it. Furthermore, the investigation used ensembles containing 5000 particles, indicating that the computational cost of achieving these results was high (although information is not provided for the impact of varying ensemble sizes).

In contrast, the EnKF may be used to infer information such as parameters as it functions by perturbing the model state based on the uncertainty in the prior model state and observations (as explained in Section 4.2). Furthermore, in cases where the Gaussian assumption hold, the Ensemble Kalman Filter typically requires smaller ensembles and should therefore be able to achieve similar improvements when compared to the Particle Filter, but at a lower computational cost.

Beyond the investigations mentioned thus far, there are also some other cases where the Particle Filter has been applied to Agent-Based Models — in particular, traffic models. These efforts, however, also suffer from issues.

In the work by Marinică et al. (2013), the investigators apply a Particle Filter to an Agent-Based Model of traffic flow, using noisy observations; however, given the computational costs of running a traffic model with the additional costs of using a Particle Filter, they choose to introduce a degree of abstraction by modelling the system based on "platoons" instead of individual agents. This approach reduces the size of the state and results in a more efficient model. Beyond this, however, it also moves away from the Agent-Based description of a system which provides us with the ability to observe individual-level interactions and heterogeneity.

In the work by Feng et al. (2015), the investigators also apply a Particle Filter to an Agent-Based Model of traffic flow. In this investigations, the observations of the system are treated as the true system state, ignoring any sensory error. Furthermore, the investigations uses error metrics based on quantities such as local vehicle density and local average speed at given locations within the environment. With such a measure of success, the investigators could just as easily used a hydrodynamic model of traffic which would likely have achieved the same results with a reduced computational cost.

Finally, in work by Kieu et al. (2020), the investigators make use of a Particle Filter with an Agent-Based Model of a fleet of buses collecting passengers along their route. Just as with some other previous investigations, this piece of work makes use of synthetic psuedo-truth data and observations to present a test case for implementing the data assimilation method with an Agent-Based Model (Malleson et al., 2020). The investigation clearly demonstrates the value of parameter calibration and data assimilation in improving the accuracy with which the model simulates and predicts the system state at close to real-time. The model used is relatively simple in comparison to some other models typically used in the field. One of the more notable simplifications is the way in which the model considers traffic, with changes in traffic density being applied uniformly across the transport network. This overlooks the potential spatial heterogeneities that may occur, how this may impact individual buses on the route and in turn how the buses may impact traffic in their local environment.

2.4 Concluding Remarks

This chapter has sought to review the literature relevant to this investigation. This has consisted of a review of pedestrian dynamics, covering the quantitative measures used to describe pedestrian systems, qualitative phenomena that may be observed in such systems and some examples of typical modelling approaches. We then went on to discuss Agent-Based Models — what they are, how they may be used in practice (particularly focussing on their calibration), and some of their challenges and shortcomings.

Finally, we discussed the literature around the novel field of real-time pedestrian simulation in which investigators have started implementing data assimilation methods in conjunction with Agent-Based Models of pedestrian systems. Many of the attempts so far have focussed on using Kalman Filter-based approaches and the Particle Filter, both of which will be discussed in Chapter 4. Both approaches have their strengths and weaknesses. Whilst early work focussed on unrealistic scenarios in which assumptions were made about unobservable parameters such as pedestrian destinations within a system, more recent investigations have examined this problem. Whilst some investigations have also attempted to make use of real-world observations with data assimilation methods, many of them have considered them as representative of the ground truth of the system of interest without acknowledging the errors introduced by the observation process.

This work, therefore, aims to expand upon the previous attempts to apply the EnKF to Agent-Based Models by using it in conjunction with a pedestrian model known as StationSim (outlined in Section 3.2). The ultimate aim of this is to show that the filter is not only capable of improving the estimations of observed variables like an pedestrian's location, but that it is capable of inferring unobserved parameters and that this can be achieved at a relatively low computational cost.

Chapter 3

Models

In the previous chapter, we reviewed the literature relevant to the investigation at hand in which we seek to apply the Ensemble Kalman Filter data assimilation method to and Agent-Based Model of pedestrian motion. In order to achieve this goal, we require both a data assimilation implementation and an Agent-Based Model to which to apply the method. In this chapter we seek to outline the models used to demonstrate the filter's effectiveness. For this investigation, we make use of two models which are documented in this chapter.

The first of these is a toy model which is described in Section 3.1. This model portrays an idealised scenario in which we consider hypothetical quadrilateral train station in which pedestrians enter through one of three entrances on one side and aim to exit the environment through one of two exits on the opposite side. This toy model acts as a preliminary test case in which to initially implement the Ensemble Kalman Filter. The second of these is a more realistic model named `StationSim_GCS` which portrays pedestrians moving around the concourse of Grand Central Station in New York. This model is described in Section 3.2. In addition to a basic model description, a basic calibration is undertaken as well as a sensitivity analysis.

3.1 Toy Model

The first of the models used in this investigation — `StationSim` — is referred to as the Toy Model[1]. The Toy Model is a deliberately simplified Agent-Based Model of pedestrian motion. It simulates the motion of pedestrians across a hypothetical train station environment, and does not necessarily accurately represent any real-world system; the aim of this model was predominantly to provide a test case on which to undertake preliminary tests with different data assimilation methods; as a consequence it has already featured in several pieces of work (Malleson et al., 2020; Clay et al., 2020).

The model consists of a rectangular environment with three entrance gates on the left edge of the environment and two exit gates on the right edge of the environment; a model run consists of a finite population of agents entering the environment through the entrance gates and traversing the environment to get to the exit gates, with the simulation ending when the last agent has completed its journey (As shown in Figure 3.1).

[1]Code for this model can be found in `Projects/ABM_DA/stationsim/stationsim_gcs.py` in the archive of the dust repository

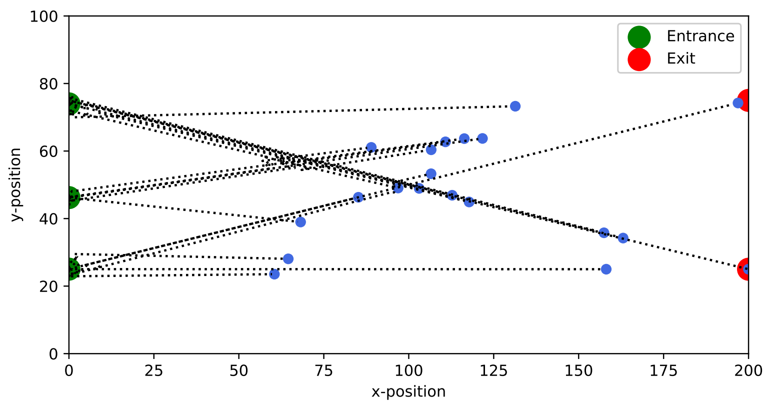


Figure 3.1: A sample run of `Station_Sim` showing pedestrian agents traversing the environment from left to right.

Upon initialisation, each of the agents are allocated:

- One of the three entrance gates through which to enter;
- One of the two exit gates through which to exit;
- A time at which to enter to environment through their allocated gate;
- A target speed at which they would like to traverse the environment.

The rate at which agents can enter/exit through a gate is dependent on the gate's size, i.e. the flow rate, J, is dependent on the gate's cross-sectional width. Target speeds are drawn from a Gaussian distribution, resulting in some variation in agent target speeds with some agents aiming to travel faster than others. With this set up, we may observe phenomena such as *crowding*.

Given that there is a variation in agent speeds, it often occurs that a faster agent is behind a slower agent and catches up to them; when this occurs, the 'collision' behaviour is used (Malleson et al., 2020) in which the faster agent attempts to sidestep the slower agent blocking their path. This is achieved by randomly choosing to sidestep left or right (i.e. tangential to the direction of travel). In some cases, this sidestep will not be sufficient to open up a new path for the faster agent as they may be blocked by another slower agent, resulting in them being stuck. As a consequence we may observe *crowding*. Such crowding phenomena are of interest as they demonstrate that the model being used is capable of exhibiting complex behaviour that may not easily demonstrable in mathematical models (Malleson et al., 2020).

The description of this model is intentionally brief, as it is only being used as a basic initial test case for the implementation of the data assimilation method. A greater degree of attention will be paid to the description of the `StationSim_GCS` model, which will include a fuller description of the behaviours, the calibration process, some elements of validation and verification, and a brief sensitivity analysis.

3.2 StationSim_GCS

Following the outline of `StaionSim` in the previous section which is used in the experiments in Chapter 5, this section seeks to detail the `StationSim_GCS` model which is used in Chapters 6 and 7[1].

This section consists of three parts. The first of these offers a description of the model; this contains details of the purpose of the model, the types of entities that make up the model and the underlying mechanisms by which it works. This is followed by a section in which the model parameters are calibrated based on real-world data. Finally, the section concludes with a sensitivity analysis which seeks to explore the how the model reacts to perturbations in parameter values.

3.2.1 Model Description

`StationSim_GCS` is an updated version of the StationSim model. The original StationSim aimed to simulate the motion of pedestrians across a hypothetical rectangular station with 3 entrances on one side and 2 exits on the opposite side as shown in Figure 3.1. The new StationSim GCS also aims to simulation the motion of pedestrians across a station; however, in this case the model is based on the real-world example of Grand Central Station in New York, focusing specifically on the concourse area highlighted in Figure 3.2. The environment consists of 11 gates which act simultaneously as both entrances and exits. Each pedestrian in the simulation is assigned an entrance and an exit and, upon entering the environment, seeks to move as directly as possible towards their assigned exit without colliding with other pedestrians. Where collisions

[1]Code for this model can be found in `Projects/ABM_DA/stationsim/stationsim_gcs.py` in the archive of the dust repository

are more likely to occur (e.g. close to entrances/exits and around solid obstacles), we typically observe crowding as population densities increase.

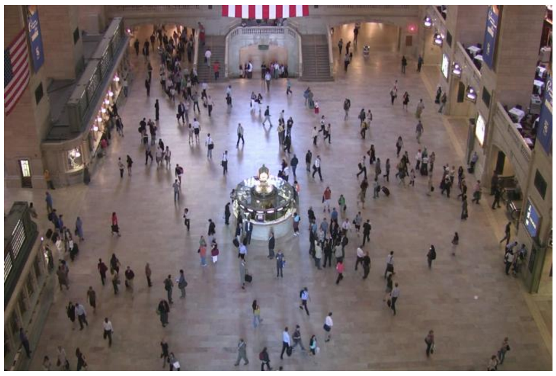


Figure 3.2: The main concourse at Grand Central Station in New York (Zhou et al., 2012).

The StationSim GCS model is made up of 4 different types of entities:

1. Agents,
2. The environment,
3. Gates around the edge of the environment, and
4. Obstacles in the environment.

These entities aim to simulate the scenario outlined above. The agents in this model represent pedestrians; these are portrayed as two-dimensional circular entities with finite radius. The variables pertaining to these agents can be found in Table 3.1. The environment in this model represents the concourse of Grand Central Station in New York; this is portrayed as two-dimensional continuous space bounded by rectangular walls within which agents may move. The model is designed such that the left-hand side of the environment represents the South side of the concourse, the right-hand side the North side, the top side the West side and the bottom side the East side. The variables pertaining to the environment entity can be found in Table 3.2.

Along the edge of these boundaries are located gates: one gate is located on the South side, five gates are located on the North side, two gates are located on the West side and two gates are located on the East side. The gates are points along the boundary of the environment at which agents may either enter or exit. These have specific fixed

Variable Name	Description
`location`	Agent's *x*-*y* coordinates in 2-dimensional continuous space; bounded by the height and width of the environment
`status`	Agent's status; 0 indicates agent has not started, 1 indicates agent is active, 2 indicates agent has finished
`size`	Radius of agent's circular body
`speed`	Agent's speed; indicative of the distance covered by an agent in a single time-step
`unique_id`	Unique numerical identifier for a specific agent in a model
`gate_in`	Number of the gate through which of the gates the agent enters the environment ($0 \leq$ `gate_in` ≤ 10)
`gate_out`	Number of the gate through which of the gates the agent exits the environment ($0 \leq$ `gate_out` ≤ 10)
`loc_desire`	*x*-*y* coordinate of the agent's target destination; defined by taking the *x*-*y* coordinates of `gate_out` and adding some uniformly distributed random noise

Table 3.1: Table of state variables pertaining to agent entities.

Variable Name	Description
`height`	Environment's height
`width`	Environment's width
`gates_in`	Number of gates through which agents can enter the environment
`gates_out`	Number of gates through which agents can exit the environment

Table 3.2: Table of state variables pertaining to the environment entity.

Variable Name	Description
`location`	Gate's x-y coordinates; restricted to one of ten distinct locations on the boundary of the environment

Table 3.3: Table of state variables pertaining to gate entities.

Variable Name	Description
`location`	Obstacle's x-y coordinates
`size`	Radius of obstacle's circular body
`speed`	Speed of agent characterising the obstacle; fixed value of 0

Table 3.4: Table of state variables pertaining to the obstacle entity.

x-y coordinates. Upon initialisation, each agent is provided with a start gate and end gate, from which it draws its initial location and target destination; in defining its target destination, the agent introduces some random noise to the x-y coordinate in order to emulate the non-zero width of the gate. The variables pertaining to the gate entities can be found in Table 3.3.

The environment also contains a single obstacle which represents information booth with a clock above it. As shown in Figure 3.2, this lies in the centre of the concourse. From a model architecture perspective, this obstacle is treated as a stationary agent; other agents therefore treat it as they would any other agent and make efforts to avoid colliding with it. The variables pertaining to the obstacle entity can be found in Table 3.4.

Of the variables detailed for each of the entities in Tables 3.1, 3.2, 3.3 and 3.4, the majority are set to fixed values upon model initialisation. The variables that change as the model runs are the `location` and `status` variables pertaining to the agent entities.

The overall model process is relatively simple; it is outlined in Figure 3.3. This process consists of 4 components:

1. Set up the station.
2. Initialise agents.
3. Step the model forward.

4. Check whether the model should be deactivated.

The process of setting up the station involves defining the state of the non-agent entities in the model, i.e. the model boundaries, the gates and the central clock obstacle, as outlined in Tables 3.2, 3.3 and 3.4 respectively.

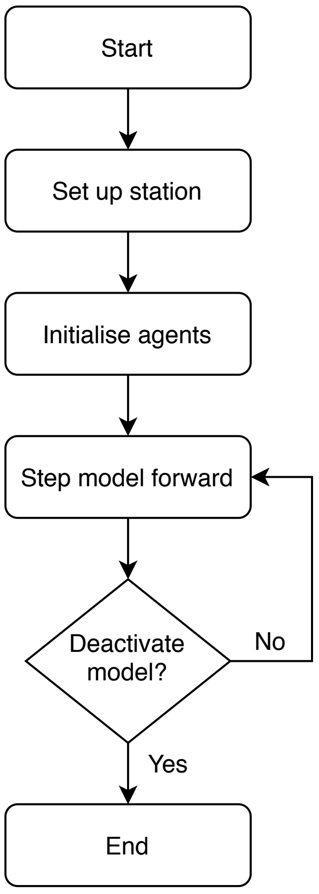


Figure 3.3: Flow diagram showing StationSim_GCS model process.

The process of initialising the agents in the model involves allocating their entrance and exit gates (along with their target destination), their speed and the time at which they will be activated (i.e. when they will attempt to enter the system).

The processes of stepping the model forward and checking for model deactivation are run iteratively whilst the model is still deemed active. After having stepped the model forward, it is checked whether the model should remain active; if so, the model stepping process is repeated, else the model run is completed.

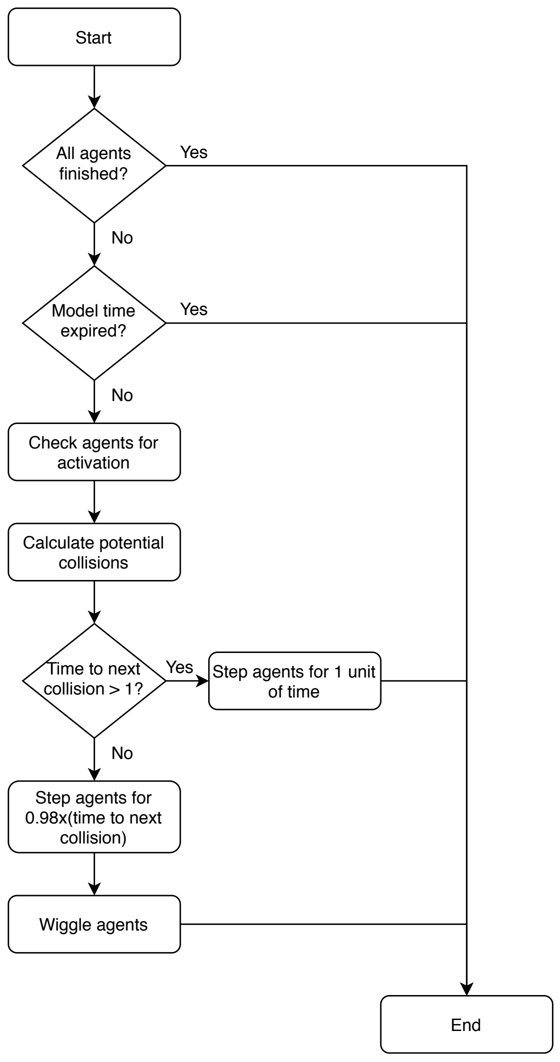

Start
All agents finished?
Yes
No
Model time expired?
Yes
No
Check agents for activation
Calculate potential collisions
Time to next collision > 1?
Yes
Step agents for 1 unit of time
No
Step agents for 0.98x(time to next collision)
Wiggle agents
End

Figure 3.4: Flow diagram showing StationSim_GCS model stepping process.

The process of stepping the model forward is displayed in Figure 3.4, which can be summarised as follows:

1. Checking whether inactive agents should be activated.
2. Identifying all of the potential collisions that may occur between agents and between agents and parts of the environment (i.e. the model boundaries and the clock obstacle).
3. Moving each of the active agents for a specific time interval based on the when the soonest collision will occur.
4. Where necessary, instruct agents to engage a sidestepping mechanic to avoid obstacles.

Each agent is stepped sequentially as part of the model stepping process; the agent stepping process (displayed in Figure 3.5) involves moving from its present location towards its target destination in a linear fashion. The movement vector is the product of the agents velocity vector and the time-step, which is dependent on the time interval to the next expected collision. As a consequence, although the agents are stepped sequentially and no collisions or interactions are missed.

It should be noted that whilst the stepping process may allow for agents to be moved over non-integer time intervals, each of the model time-steps corresponds to an integer time interval. This means that a model time-step may comprise of multiple non-integer time-steps taken by the individual agents. As an example, when stepping agents forward through time, the model may detect that a collision is imminent 60% of the way through a time-step. The model will then step all of the agents through to just short of this collision, apply the side-stepping mechanic for the agents that about to collide and then attempt to step the agents for the remainder of the time-step; if it does not detect any other imminent collisions, it will step them through the remaining 40% of the time-step. The model will then record the states of its agents. This ensures that all outputs produced based on the models have time-steps that are of equal length.

Flow diagrams describing other model subprocesses in greater detail can be found in Appendix B; this includes descriptions of how agent locations and speeds are allocated (Figures B.2 and B.3), how agents check for activation and deactivation (Figures B.4 and B.5), and how agent-agent and agent-wall collisions are handled (Figures B.8 and B.9).

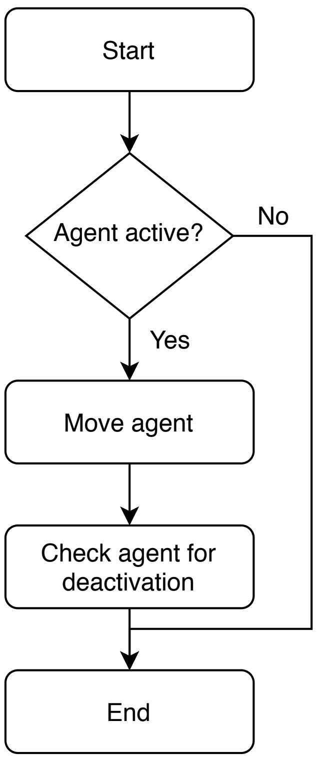


Figure 3.5: Flow diagram showing `StationSim_GCS` agent stepping process.

Of the design concepts outlined by Grimm et al. (2010), the following are considered relevant to this model:

1. Emergence
2. Adaptation
3. Prediction
4. Sensing
5. Interaction
6. Stochasticity
7. Observation

As outlined in Section 3.2.1, in each time-step agents engage in collision avoidance (both with each other and with walls and obstacles). This is achieved by *predicting* the paths that agents would take if they were to continue moving towards their target destinations; this is made possible through the use of a k-d tree which emulates a form of *sensing* whereby each agent is aware of the position of other agents who are likely to collide with them. In cases where collisions would occur, the agent paths are *adapted*. Such adaptations can be considered a form of *interaction* between an agent and other agents (as well as stationary objects in the environment).

Stochasticity is incorporated in the model in number of different ways. Upon initialisation, agents are randomly allocated an entrance gate and an exit gate; entrance gates are sampled from a uniform discrete distribution, and exit gates are sampled from a uniform discrete distribution excluding the gate through which the agent is entering. In cases where pedestrians are predicted to collide, part of the avoidance process involves the addition of normally distributed random noise to the agent's movement vector. Finally the time at which each agent enters the model is sampled from an exponential distribution.

Whilst the model is running, *observation* is undertaken by collecting information on the positions of each agent at each time-step for comparison with pseudo-truth data. In scenarios where pedestrian density is sufficiently high, we observe the *emergence* of crowding behaviour.

3.2.2 Model Calibration

Having described the way in which the model runs in the previous section, this section aims to outline the process undertaken to calibrate the model. The calibration process focusses on the estimation of two features: parameters pertaining to agent speeds and the agent birth rate parameter[1].

Although a number of calibration processes have been outlined in Section 2.2.1, this investigation will make use of a much simpler approach based exclusively on empirical data analysis. The motivation for this is that the aim of this investigation is not to produce a well-calibrated model that simulates the motion of pedestrians around the concourse at Grand Central Station, it is to demonstrate the effectiveness of the Ensemble Kalman Filter in improving the accuracy with which we can simulate the system. In order to demonstrate this, it is almost expected that the model not be perfect.

Data

The calibration of `StationSim_GCS` is undertaken using real-world data captured by cameras at Grand Central Station, which was produced by a study by Zhou et al. (2012). This data consisted of a series of trajectories, each of which consisted of a series of x-y-t coordinates, i.e. locations within the environment with associated timestamps.

The raw data suffered from two issues, each of which were addressed by Ternes et al. (2021). The first of these is that the trajectories were capture by a camera that was located at an angle on the east side of the concourse, which distorted the image. This was addressed by applying some perspective correction. Furthermore, many of these trajectories were partial in nature, which is to say that they do not capture the entire trip of a pedestrian across the environment but instead a subsection of this trip. Consequently, some pre-processing was applied to these partial trajectories to produce data which contained trajectories that were as close to complete as possible.

It should be noted that whilst this calibration makes use of real-world data, the data assimilation experiments in Chapters 5, 6 and 7 make use of pseudo-truth data in order to calculate errors and evaluate their effectiveness. This is a practice that has been undertaken in a number of recent investigations in the field including those by Malleson et al. (2020) and Clay et al. (2020, 2021). When running experiments in each of the aforementioned results chapters, a known "ground truth" is required against which to

[1]The calibration processes are outlined in python notebooks in `Projects/ABM_DA/experiments/stationsim_gcs_calibration/notebooks/` in the dust repository archive; speed calibration is found in `01_speed_estimation.ipynb` and birth rate calibration is found in `02_activation_rate_estimation_average.ipynb`

compare the outputs of the data assimilation implementations. In reality, the ground truth of a system is not knowable — the best we can hope for is a set of observations or estimates of the system which reflect the state with high fidelity. As a consequence, in order to provide a ground-truth, a set of pseudo-truth states are generated by running an instance of the calibrated model. In addition, where observations are required in the results chapters, these are derived from the pseudo-truth states by adding noise — a process which will be outlined in Chapter 5.

Estimating Speed Parameters

Our dataset contains the x-y locations for pedestrians in a series of frames. For each pedestrian, there exist a series of coordinates that collectively form a trajectory which describes the path that the pedestrian takes around the station concourse. Armed with knowledge regarding the time interval between frames, we can calculate the speed of individual pedestrians.

As part of this calibration process, we shall consider three separate approaches. The first of these seeks to calculate the average speed of pedestrians based on the distance and time between them entering the environment and exiting the environment.

For each pedestrian, we have a series of x-coordinates and a series of y-coordinates:

$$p_i = \{[x_i^0, \dots, x_i^{n_i}], [y_i^0 \dots, y_i^{n_i}]\} \quad \forall i \in [0, N], \tag{3.1}$$

where p_i represents the observations of the ith pedestrian, x_i^j is the ith pedestrian's x-coordinate in the jth frame in which the pedestrian appears, y_i^j is the ith pedestrian's x-coordinate in the jth frame in which the pedestrian appears, N is the number of pedestrians in the system and n_i is the number of observations of pedestrian i. Based on this, we can calculate the Euclidean distance travelled by the ith pedestrian, i.e. the total distance between a pedestrian's initial position and its final position:

$$d_i = \sqrt{(x_i^{n_i} - x_i^0)^2 + (y_i^{n_i} - y_i^0)^2}. \tag{3.2}$$

If we assume that each pedestrian appears in a sequential set of frames without any gaps, then we can define the total time between the initial observation and the final observation of the ith pedestrian in the system as:

$$t_i = n_i, \tag{3.3}$$

i.e. the time spent by a pedestrian in the system is equal to the number of frames over which it is observed.

We note, at this stage, that the provided data represents distances in pixels and times in frames; these can be rescaled based on the following ratios:

- Frames per second (`fps`) = 25.
- Pixels per metre (`ppm`) = 14.

We can then calculate the average speed of the ith pedestrian, $\bar{v}_i$:

$$\bar{v}_i = r \times \frac{d_i}{t_i}, \tag{3.4}$$

where r is the rescaling factor:

$$r = \frac{\texttt{fps}}{\texttt{ppm}}. \tag{3.5}$$

Based on this, we can produce a histogram showing the distribution of average speeds for all pedestrians as shown in Figure 3.6.

With a mean speed of 1.60 and a standard deviation of 0.66 this result is largely in agreement with average pedestrian speeds observed in the literature (Finnis and Walton, 2006), particularly when considering speeds observed in similar contexts (Young, 1999). There is, however, an issue with this approach: it measures the distance travelled by the pedestrian as a straight-line between the initial location and final location (Li et al., 2012). This assumes that each pedestrian travelled in a straight-line for the entirety of their traversal of the station concourse; this may be the case for many pedestrians but is unlikely to be true for all members of the population. In reality, we are likely to have some pedestrians who have crossed the station from initial location to final location in a straight line, some who crossed in a curve/arc, and some who may have chosen highly non-linear paths (perhaps stopping and changing direction). In each of the two latter scenarios, the distances travelled by the pedestrian would be increased.

An alternative approach to exploring the speeds of pedestrians is to consider the instantaneous velocity of a pedestrian from one frame to the next. We can do this by considering the marginal distance travelled between frames; let us consider the distance travelled between the jth frame and the $(j+1)$th frame. We can define the marginal distance between pedestrian's jth position and pedestrian's $(j+1)$th position as:

$$d_i^j = \sqrt{(x_i^{j+1} - x_i^j)^2 + (y_i^{j+1} - y_i^j)^2}, \tag{3.6}$$

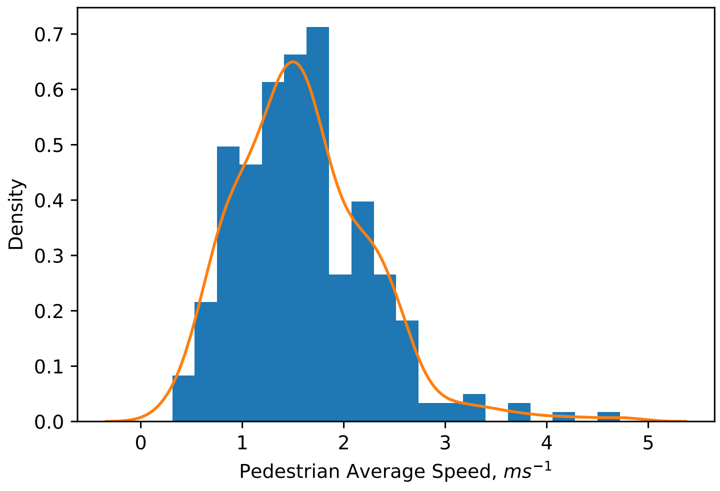


Figure 3.6: Distribution of average speeds

and the marginal time between jth observation and $(j+1)$th observation of ith pedestrian in system:

$$t_i = 1, \tag{3.7}$$

i.e. 1 frame.

Again, these distances and times are calculated in pixels and frames respectively. Consequently they need to be rescaled to metres and seconds. We can then calculate the instantaneous speed of the ith pedestrian in the jth observation, v_i^j:

$$v_i^j = r \times d_i^j. \tag{3.8}$$

We can then construct a list of these speeds for each pedestrian:

$$v_i = [v_i^0, \ldots, v_i^{n_i-1}], \tag{3.9}$$

noting that for the ith pedestrian who was observed in n_i frames, the final element of the list will be $v_i^{n_i-1}$ as this will be based on the distance between the pedestrian's location in the $n_i - 1$th frame and in the n_ith frame.

Based on this, we can produce a histogram showing the distribution of all instantaneous speeds for all pedestrians as shown in Figure 3.7. This shows that whilst a substantial proportion of the speeds are in a sensible range (i.e. $< 2ms^{-1}$), there is also a substantial proportion of the speeds that are $> 2ms^{-1}$; this is reflected by the

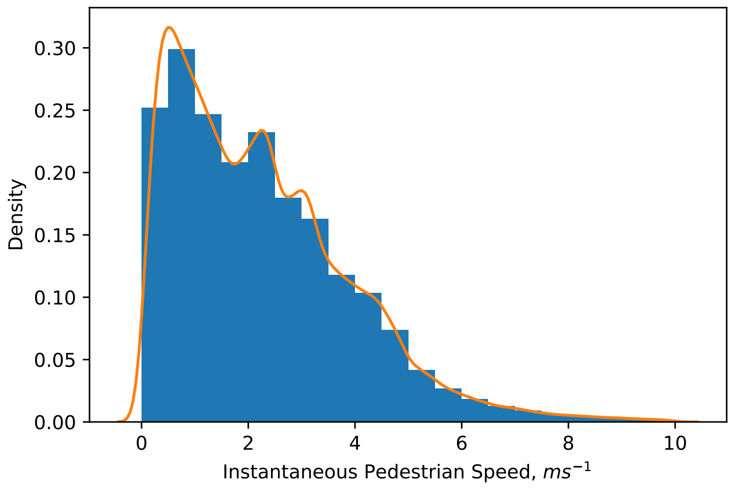


Figure 3.7: Distribution of instantaneous speeds

median speed of $1.99ms^{-1}$. This may be a result of the dataset that has been used for this analysis — this dataset contains modifications to the original dataset whereby segments of pedestrian trails are joined together in order to form pedestrian trails that are as complete as possible, i.e. as many pedestrians as possible have a full trajectory from an entrance gate to an exit gate. This involves interpolating a pedestrian's path between trail segments, which may require that the pedestrian would have move much faster than typically expected. This may, therefore, artificially introduce some pedestrian movements that are faster than expected.

An final approach to finding a pedestrian's average speed is to consider the list of speeds produced for each pedestrian in the previous section, and find the mean of that list:

$$\bar{v}_i = \frac{1}{n_i - 1} \sum_{j=0}^{n_i - 1} v_i^j \tag{3.10}$$

Based on this, we can produce a histogram showing the distribution of average speeds for all pedestrians as shown in Figure 3.8. As with the previous approach, this shows a distribution of pedestrian walking speeds that are typically faster than would be expected (with a mean of $2.63ms^{-1}$). Once again, this may be a result of the trajectory modification/connection process that the data have undergone. It may, therefore, be worthwhile looking at the original dataset which has not undergone any modifications

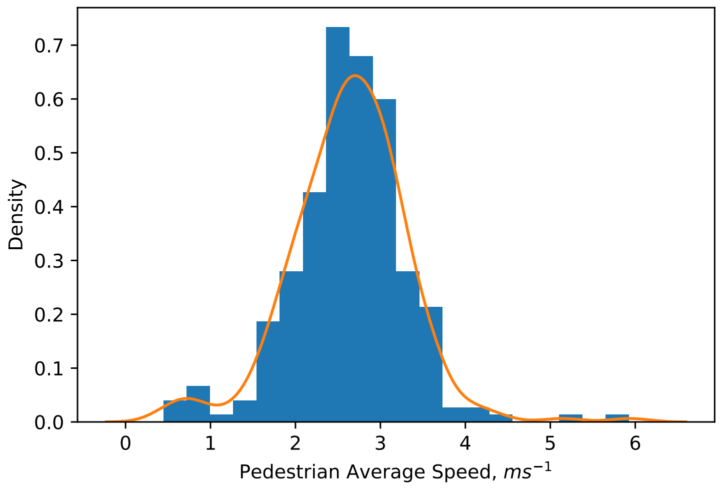


Figure 3.8: Distribution of average instantaneous speeds

to join up segments of pedestrian trajectories.

Ultimately, the parameter values estimated by the first approached, i.e.

$$v_\mu = 1.60ms^{-1}, \tag{3.11}$$

$$v_\sigma = 0.66ms^{-1}, \tag{3.12}$$

are used for this investigation as they provide best agreement with parameter values found in the literature.

Estimating Birth Rate

Having arrived as estimates for values for parameters pertaining to speed in the previous section, we now move on to estimating the birth rate of agents, i.e. the number of agents that are activated in a time-step. This is undertaken via two approaches: a visual approach and a numerical approach. The visual approach involves plotting graphs of how many pedestrians are active in the system according to the observed data and according to model runs for different birth rates. The numerical approach involves considering three error metrics:

1. Maximum number of active pedestrians in the system at any given time.
2. Time at which the maximum number of active pedestrians occurs.

3. Time at which all pedestrians have completed their journeys across the environment.

In each case, errors are calculated by comparing the value produced from the observed data with the value produced from the model runs. The variation of the number of pedestrians in the system over time can be seen in Figure 3.9. Model data is produced by running the model with the speed parameter values found in the previous section in conjunction with birth rates between 1.0 and 2.0 increasing in value by 0.1 each time; the model is run 20 times for each birth rate. The visual results of this procedure are presented in Figures A.1 — A.11. Each figure displays how the number of active pedestrians in the system varies over time for a specific birth rate.

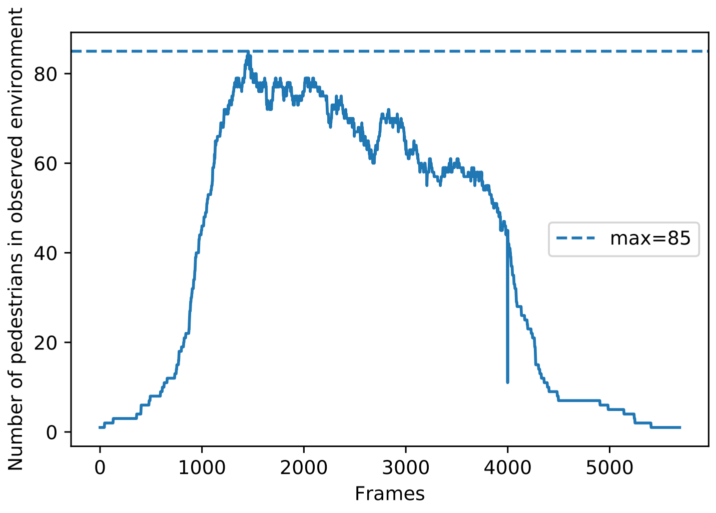


Figure 3.9: Variation of number of pedestrians in the system over time (in frames)

When considering the plot of the observed data in Figure 3.9, we can see that the maximum number of pedestrians in the system at any given time is 85. This occurs in the 1454th frame after which the number of pedestrians in the system decays quickly. By the 5687th frame, all pedestrians have left the environment.

When conducting visual inspections of the figures in Appendix A, we can see that with lower birth rates, the model takes longer to reach its maximum number of pedestrians in the system, and that the maximum number of pedestrians is lower. As the birth rate is increased, the maximum number of pedestrians in the system at any given time increases, and the time at which this occurs falls. Based on visual inspections, we can conclude that the figures produced by birth rates of 1.6 and 1.7 best fit the observed data.

When considering the numerical metrics, it was observed that the measure concerned with the time taken for all pedestrians to leave the system was of little use; looking at Figures A.1 — A.11, we can see that for the majority of birth rates there is little variation in the total time for all pedestrians to complete their journeys. Furthermore, it was found that, within the range of values considered, considering the time at which the maximum number of pedestrians were in the system was also of little use; looking at the figures in question, we can see that the instance of the maximum number of agents in the environment occurs later than the maximum number of pedestrians in the environment in the observed data for all birth rates. Whilst the difference in this time falls as the birth rate is increased, it is concluded that a much higher birth rate would be required for this metric to be considered.

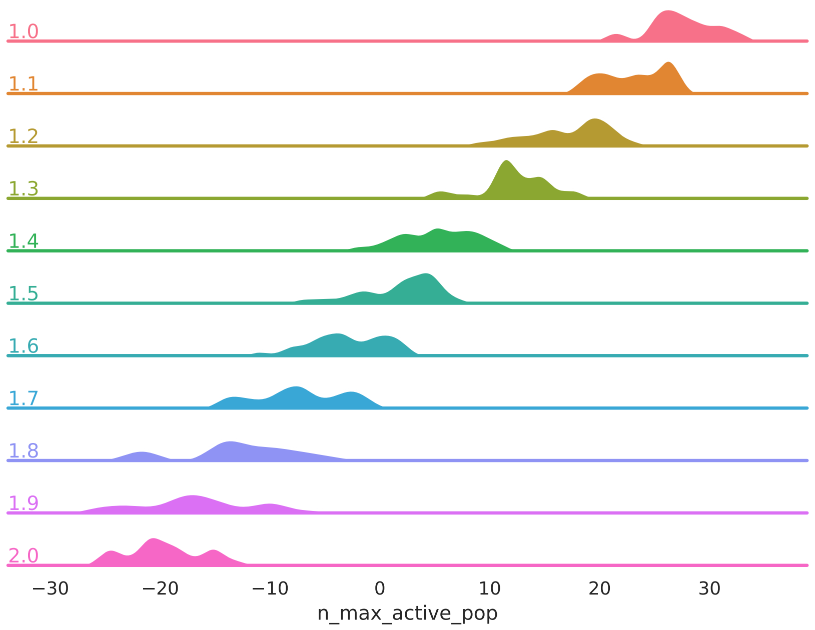


Figure 3.10: Ridge plot showing distribution of error in the maximum number of pedestrians in the system for different activation rates.

We may use Figure 3.10 to consider the error in the maximum number of pedestrians in the system. This figure consists of a ridge plot showing how the error in the maximum number of pedestrians in the system varies with agent activation rate. We can see that for the lower activation rate of 1.0, the error is comparatively high and falls as the activation rate increases. At a certain point, the error reaches zero and becomes increasingly negative as the activation rate continues to increase. The point at which the error is approximately zero is 1.6. We therefore conclude that the activation rate to be used with this model shall be 1.6.

3.2.3 Model Validation and Verification

As noted in Section 2.2.2, the processes of validation and verification are often overlooked when developing an Agent-Based Model. Whilst the aim of this investigation is not to produce a model that simulates the pedestrian system with as much fidelity as possible, we undertake some verification and validation to ensure that `StationSim` functions as expected. This section therefore aims to outline the verification and validation undertaken.

Recall from Section 2.2.2 that we define verification as the process of ensuring that the logical implementation of the model is correct (Xiang et al., 2005), and validation as the process of ensuring that the model represents the system and generates the phenomena of interest (Crooks and Heppenstall, 2012).

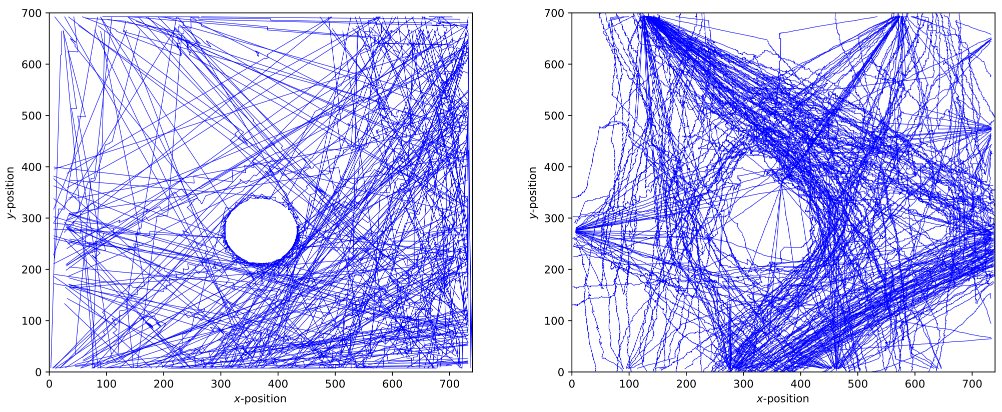
Model Validation

Given that this investigation does not aim to produce a high-fidelity model of the pedestrian system, the validation undertaken here will be relatively rudimentary and will focus on the visual comparisons of outputs from the model and the observations. Comparisons will be drawn on two fronts. The first of these will focus on the paths taken by pedestrians in the observed data and in the model. In order for a model to be considered valid on this front, we require that it is able to replicate the behaviours found in the observations; this is to say that the paths taken by agents in the model should resemble the paths taken by pedestrians in the observations, and the individual trajectories produced by the model should not be markedly different from those observed in the observations. This seeks to demonstrate that the pedestrian behaviours have been correctly characterised within the model. These second of these will focus on the time spent in the system by pedestrians in the observed data and in the model. In order for a model to be considered valid on this front, we require that the agents spend approximately the correct length of time within the environment. This seeks to demonstrate that the parameter values for the pedestrian agents in the previous section have been correctly chosen.

In Figure 3.11, we see a comparison of the paths taken by pedestrians in the observed data and in the model. In drawing this comparison, we seek to ensure that the behaviours of the pedestrians within the model reflects the behaviours observed within the data. In both Figures 3.11a and 3.11b, pedestrians take routes between their respective points of origin and destination. These origins and destinations are located around the boundary of the environment. In Figure 3.11a, we see that the trails generated by the pedestrian agents in the model are largely linear, tracing straight lines between origins and destinations. In some cases, we see that agents deviate from their straight paths, attempting to avoid obstacles (other agents and the central desk). This is, to some extent, reflected by the paths taken by the pedestrians in the observed data in Figure 3.11b.

There are, however, some noticeable changes in direction found in the paths taken by the pedestrians in the observed data, which are not reflected by the paths taken by the agents in the model. Whilst some of these deviations are reflective of pedestrians changing their direction of travel across concourse, some of them are artefacts of the way in which the observations have been preprocessed. The initial data regarding the paths taken across the concourse consist of partial trajectories identified in the computer vision analysis of the footage of the concourse. The preprocessing procedure applied to these partial trajectories sought to use them to construct full trajectories. This was achieved by linking them based on their the spatial and temporal distance between them, interpolating the points in-between. This interpolation results in some noticeable changes in direction and speed which are not found in the model outputs. Furthermore, this introduces some paths in the observed data which appear to pass through the central obstacle — a behaviour which is not physically possible and as such one which we should not expect to find in the model outputs. We may therefore conclude that whilst the model does not perfectly recreate the traces found in the observed data, this may be partly due to shortcomings in the data.

Beyond this, it should also be noted that the validation process also identifies some of the model's shortcomings which can be improved upon. One of these is the variation in trail density across the environment. When Figure 3.11b, we can see that a large proportion of pedestrian trajectories occur between a specific subset of the environment entrance/exit gates (specifically between the gates on the left-hand side of the top boundary and the centre of the right-hand side boundary, and between the gates on in the centre of the bottom boundary and centre of the right-hand side boundary). These distributions are not replicated in the model outputs found in Figure 3.11a. In the model, entrance and exit gates are allocated based on uniform distributions; this difference may be indicative of distributions that are not uniform.



(a) Model (b) Data

Figure 3.11: Comparison of trails from model and observations

When considering Figure 3.12, we see the time spent by pedestrians in the system, both in the model and in the observed data. In comparing the outputs of the model and the observed data, we seek here to confirm that the rate at which pedestrians enter the environment and speed at which they cross it in the model are reflective of the real-world system. In Figure 3.12, we see that the most common amount of time which pedestrians spend in the system in both the model and the observed data is approximately 800 frames, i.e. approximately 32 seconds. In comparing the distributions of the times from the model and the data, we find that the peak of the data distribution is higher than that of the model distribution, indicating that more pedestrians in the observed data are in the system for a shorter duration. Just as in the case of the comparison of traces above, this may be due to the data preprocessing approach. In particular, we note that when the preprocessing approach links two partial traces, it may introduce points to link the end of the first trace to the beginning of the second trace. This linking process is constrained by not only the end location of the first trail and start location of second

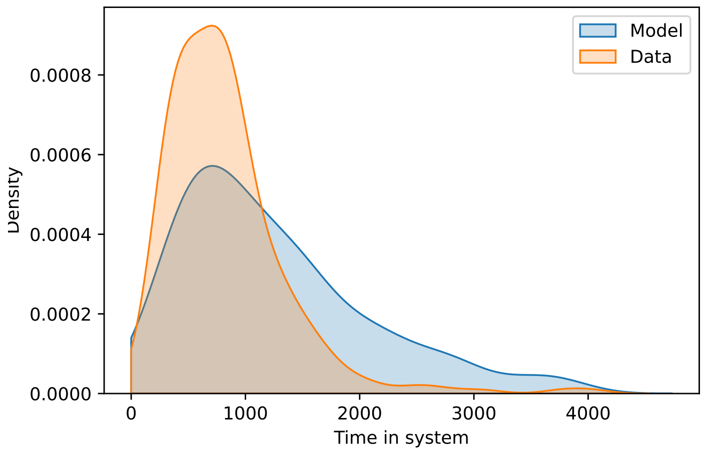


Figure 3.12: Comparison of time spent in system between agents in mode and observed pedestrians.

trace, but the respective times at which the pedestrians appear at these locations. In order to link the two traces, a new intermediate trace is drawn in-between which may depict the pedestrian moving much faster. As a result of this artificially increased speed, the time spent in the system by the pedestrian may be reduced. This may result in the increased peak we observed in the data distribution in Figure 3.12.

Model Verification

In Section 2.2.2, we note the difficulties encountered when trying to verify the implementation of Agent-Based Models, and that one approach to verification may be to apply a test-driven model development process (Collier and Ozik, 2013). Consequently, a set of tests were developed[1].

A unit testing approach was undertaken, which aimed to ensure that atomic elements of the codebase functioned as expected. To achieve this, some tests required the creation of dummy data. As noted in Section 3.2.1, some of the elements of the model make use of computational randomness; this presented a challenge when attempting to ensure that model methods function as expected in a consistent manner. In order to address this issue, some tests sought to check the results of methods in a manner that

[1]Tests can be found in `Projects/ABM_DA/tests/` in the dust repository archive; specific model tests can be found in `test_stationsim_gcs_model.py`

was less quantitative and more qualitative.

As an example, the model includes a method which assigns an origin location to an agent when provided with an origin gate; the origin location is generated in a fashion that makes use of computational randomness. The respective test did not check that the generated location was exactly correct, but instead checked that it was a valid location in proximity to the gate in question.

When running the tests, it was found that they provided a 59% coverage of `StationSim_GCS`, i.e. the running of all of the test cases resulted in 59% of the lines in the model codebase being called. This means that over 40% of the code for the model is not called by any of the tests. Whilst this proportion may appear high, it should be noted that a portion of the codebase is dedicated to plotting utilities which are not essential to the logic of the model; omission of plotting utilities results in an increased coverage of 74%.

One of the shortcomings of the approach implemented is that it focuses predominantly on unit testing, and does not undertake integration testing whereby tests focus on ensuring that software components interact with each other as expected. This investigation focuses on the use of Agent-Based Models in conjunction with Data Assimilation methods and it is, therefore, crucial that the model interacts as expected with the code developed for these methods. A more thorough verification approach would check not only that the logic of the model was encoded correctly, but also that the model was able interface with other parts of the codebase.

3.2.4 Sensitivity Analysis

In this section, we shall undertake a basic sensitivity analysis of the model. Sensitivity analysis is an approach to model analysis which seeks to explore how the model behaviour varies with respect to changes in model parameters (Saltelli, 2002). It can be undertaken with a view to different end-goals.

Sensitivity analysis can be used for a range of different reasons such as:

- **Confirming that the parameter values arrived at through calibration are 'optimal'.** Model calibration undertaken via an optimisation seeks to minimise some cost function/error, and in this situation, the introduction of perturbations to these parameter values should result in an increase in this cost function/error — in this situation, sensitivity analysis would consist of introducing perturbations to the parameter values and exploring how the cost function varies.

- **Exploring variable importance.** We can compare the extent to which model behaviour changes in response to changes in different parameters values, ranking variables based on which induce the greatest changes in model behaviour. In order to explore this, a cost function/error metric would again be required to quantify the behaviour of the model.

- **Exploring model behaviour.** We can explore how the model behaviour changes in response to changes in different parameter values; this could be something more visual.

We may wish to divide approaches to sensitivity analysis into two categories: local approaches and global approaches. Local approaches seek to quantify the impact of small perturbations in model parameters (Saltelli, 1999). A common approach to this is to use a one-factor-at-a-time method (OAT) whereby small perturbations are introduced to one of the model parameters in question whilst keeping all other model parameters the same, allowing us to understand the impact of changes to a model parameter in isolation (Thiele et al., 2014). This may, however, overlook interactions between model parameters that may lead to non-linear responses in model behaviour. Global approaches focus on much larger ranges of parameter values, seeking to explore how the model behaviour varies in response to changes in the model parameters over all parameter space (Saltelli et al., 2000). Furthermore, this may also explore the response of the system to simultaneous changes in multiple parameter values.

In this specific case, we wish to explore the sensitivity of the model's behaviour with respect to the following parameters:

- Mean agent speed,
- Standard deviation of agent speed,
- Agent activation rate.

Many of the other model parameters such as environment dimensions and gate locations are clearly measurable and fixed; these parameters were chosen for sensitivity analysis as they are uncertain and may vary. This will consist of a global sensitivity analysis[1].

[1]Code for this sensitivity analysis can be found in the `3_sensitivity_analysis_global.ipynb` in

This will allow us to explore the way in which model behaviour responds to variations in parameters, gauging variable importance.

Just as when undertaking the activation rate section of the model calibration, we shall consider errors in the maximum number of pedestrians in the system, the time at which this maximum occurs and the time take for all pedestrians to complete their journeys across the environment.

The aim of this sensitivity analysis is to explore the way in which the overall model behaviour responds to changes in parameter values, and ascertain the relative importances of the parameters in question. The basis of this analysis will be the repeated running of the model with a selection of parameter values. To gain the combinations of parameter values, samples will be drawn from uniform distributions between the respective minimum and maximum parameter values defined in Table 3.5. A sample size of 200 runs is chosen given the trade-off between the increasing computational cost of running many iterations of the model and the increasing reliability of any findings.

Variable	Calibrated Value	Minimum	Maximum
Activation rate	1.6	1.0	2.0
Mean speed	0.897	0.5	1.5
Std of speed	0.372	0.2	1.0

Table 3.5: Table of parameter values used for global sensitivity analysis.

Based on the collection of model runs, a set of regression models are constructed. With these models, we seek to fit a regression model for which we consider the following input variables:

- Mean agent speed,
- Standard deviation of agent speed,
- Agent activation rate,

and the following output variables:

the `Projects/ABM_DA/experiments/stationsim_gcs_calibration/notebooks/` in the dust repository archive

- Error with respect to maximum number of pedestrians in environment in environment at any given time,
- Error with respect to time taken for all pedestrians to complete their journey,
- Error with respect to the time at which the maximum number of pedestrians in the system at a given time occurs.

For this investigation, we shall be using ordinary least squares linear regression to fit a model to our data; whilst more advanced regression modelling approaches exist which would likely fit the data better, we seek a model which is easily explainable.

Maximum Number of Pedestrians

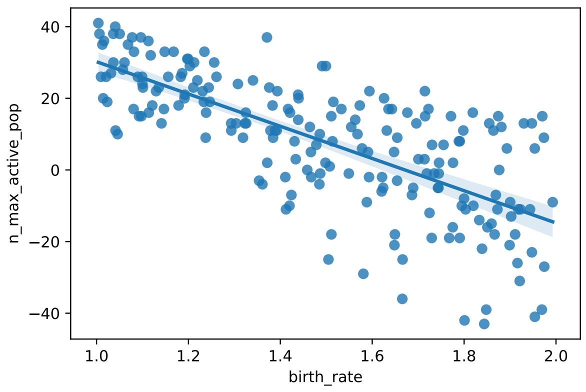
For the first of these models, we fit a linear regression in which the response variable is the error in the maximum number of active pedestrians:

$$Y = \beta_0 + \beta_{ar} X_{ar} + \beta_{ms} X_{ms} + \beta_{ss} X_{ss}, \tag{3.13}$$

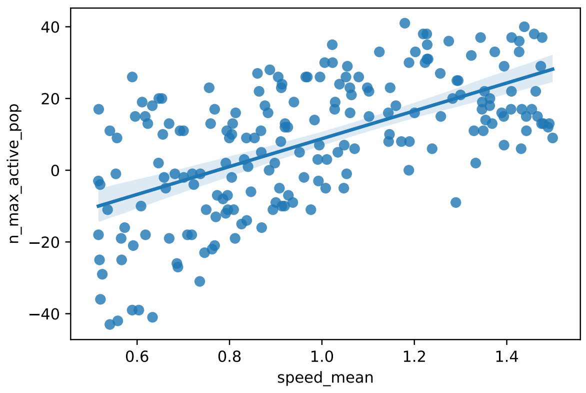
where terms with a ar subscript pertain to the activation rate, terms with a ms subscript pertain to the mean speed and terms with a ss subscript pertain to the standard deviation of the speed.

Upon fitting such a model, we obtain coefficient values pertaining to each of the model parameters as detailed in Table 3.6. When fitting the linear regression model, each of the three parameters were found to be significant (as indicated by the p-values in the table). Considering the results in Table 3.6, we see that all three of the parameters were found to be significant in influencing the error in the maximum number of pedestrians in the system. The way in which the error varies in response to each of the three parameters was therefore plotted in Figure 3.13.

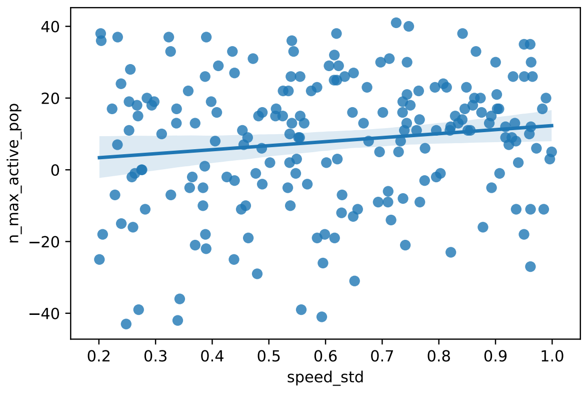
The coefficient pertaining to the activation rate was found to be negative, indicating a negative correlation with the error, i.e. as the activation rate increases, the error falls. The coefficient pertaining to the mean of the speed was found to be positive, indicating that as the mean speed increases the error increases. Finally, the coefficient pertaining to the standard deviation of the speed was found to be positive, indicating that as the standard deviation increases the error increases. Each of these three phenomena can be observed in Figures 3.13a, 3.13b and 3.13c respectively.



(a) Variation in activation rate.



(b) Variation in mean speed.



(c) Variation in standard deviation of speed.

Figure 3.13: Variation of error in the maximum number of active pedestrians with respect to different parameters

Parameter	Coefficient Value	Standard Error	p-value
Constant	30.6	2.63	0.000
Activation Rate	-45.5	1.34	0.000
Mean Speed	39.3	1.37	0.000
Std Speed	11.2	1.70	0.000

Table 3.6: Table of coefficient values; response variable: error in maximum number of pedestrians in the system.

Time of Maximum Number of Pedestrians

For the next of these models, we fit a linear regression in which the response variable is the error in the time of the maximum number of active pedestrians:

$$Y = \beta_0 + \beta_{ar} X_{ar} + \beta_{ms} X_{ms} + \beta_{ss} X_{ss}, \tag{3.14}$$

where once again terms with a ar subscript pertain to the activation rate, terms with a ms subscript pertain to the mean speed and terms with a ss subscript pertain to the standard deviation of the speed.

Upon fitting this model, we obtain coefficient values pertaining to each of the model parameters as detailed in Table 3.7. Considering the results in Table 3.7, we see that only the activation rate is significant in influencing the error in the time of the maximum number of pedestrians in the system. The way in which the error varies in response to the activation was therefore plotted in Figure 3.14.

Parameter	Coefficient Value	Standard Error	p-value
Constant	-4399.5	384.7	0.000
Activation Rate	1432.2	195.6	0.000
Mean Speed	-24.3	200.2	0.903
Std Speed	161.5	249.6	0.518

Table 3.7: Table of coefficient values; response variable: error in time of maximum number of pedestrians in the system.

The coefficient pertaining to the activation rate was found to be positive, indicating that as the activation rate increases the error in the time at which the maximum number

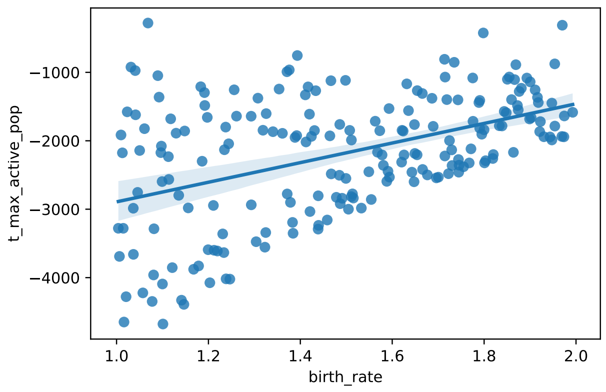


Figure 3.14: Variation of error in the time at which the maximum number of active pedestrians occurs with respect to activation rate.

of pedestrians is observed increases. When looking at Figure 3.14, however, we can see that the error for all activation rates used is negative. A positive error may be observed for activations in excess of 2.0. This indicates that a higher agent activation may be desirable in the goal of simulating the time at which the maximum active population occurs; however, as noted in the previous section, an increased activation rate may incur an greater error in the maximum active population.

Completion Time

Finally, for the third model, we fit a linear regression in which the response variable is the error in the time at which all pedestrians have completed their journeys across the environment:

$$Y = \beta_0 + \beta_{ar}X_{ar} + \beta_{ms}X_{ms} + \beta_{ss}X_{ss}. \tag{3.15}$$

Upon fitting this model, we obtain coefficient values pertaining to each of the model parameters as detailed in Table 3.8. Considering the results in Table 3.8, we see that only the activation rate is significant in influencing the error in the time at which all pedestrians have completed their journeys. The way in which the error varies in response to the activation was therefore plotted in Figure 3.15.

The coefficient pertaining to the activation rate was found to be positive, indicating that as the activation rate increases the error in the time at which all pedestrians have completed their journeys increases. Just as with the previous model, when looking at

Parameter	Coefficient Value	Standard Error	p-value
Constant	-7048.6	457.8	0.000
Activation Rate	2915.1	232.7	0.000
Mean Speed	285.3	238.2	0.233
Std Speed	139.7	296.9	0.638

Table 3.8: Table of coefficient values; response variable: error in time taken for all pedestrians to complete journeys.

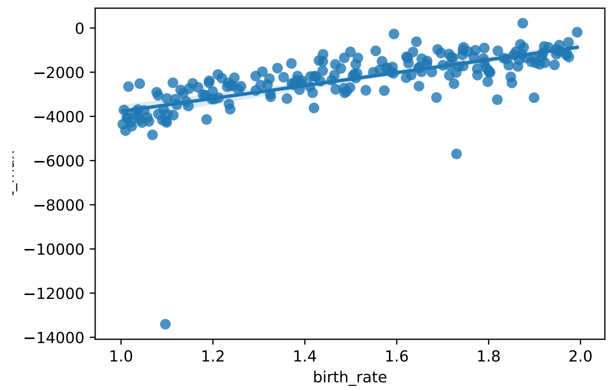


Figure 3.15: Variation of error in length of time take for the model to complete with respect to activation rate.

Figure 3.15, we can see that for all activation rates used is negative. Just as in the previous section, this indicates that a higher agent activation may be desirable in the goal of simulating the completion time; however, as noted above, the previous section, a increased activation rate may incur an greater error in the maximum active population.

3.3 Concluding Remarks

This section has sought to outline the models used in this investigation.

The first of these (referred to as the Toy Model) was described in Section 3.1. It is an abstract model developed for testing purposes, and as such shall be used in Chapter 5 to perform preliminary experiments to show that the Ensemble Kalman Filter can be applied to an Agent-Based Model. The second model — `StationSim_GCS` — was described in Section 3.2 (which also included a basic calibration process and sensitivity analysis for the model). This model aims build upon the Toy Model, simulating the motion of pedestrians around the main concourse at Grand Central Station in New York. It shall be used in Chapters 6 and 7.

The latter model was calibrated through the analysis of observed data of pedestrians crossing the main concourse at Grand Central Station, which sought to identify appropriate values for the rate at which pedestrian agents should join the system, i.e. their activation rate, the mean speed at which they should be travel, and the standard deviation of their speeds. The speed parameters were used to inform a normal distribution from which agent speeds could be sampled in the model. The analysis found the mean pedestrian speed to be $1.60ms^{-1}$ and the standard deviation to be $0.66ms^{-1}$. The analysis concluded that an appropriate activation rate would be 1.6.

Following this, a global sensitivity analysis was undertaken, seeking to explore how the model behaviour changed in response to each of these parameters, and which had the greatest influence. It was found that model behaviour was largely unchanged by the speed parameters, but that agent activation rate had a much larger impact. It was found that as activation rate increased, the error in the maximum active population decreased (changing from positive to negative), the error in the time of the maximum active population increased (starting negative and growing closer to 0), and the error in the system completion time increased (starting negative and growing closer to 0). Whilst it was found that increasing the activation rate would result in improvements in the simulation of the time at which the maximum active population occurred and the time at which the system would complete its run, this would have resulted in a growth in the error in the maximum active population. Given that having a large number of active agents would result in crowding — a phenomenon that we desire (see Section 3.1) — we have chosen to make no further changes to the activation rate.

Having described the modelling aspects of the methods used for this investigation in this chapter, the next chapter shall proceed to outline the other side of the methods: data assimilation.

Chapter 4

Data Assimilation

In Chapter 2, previous investigations into the use of data assimilation methods with Agent-Based Models were reviewed; although it touched on the strengths and weaknesses of different methods, the chapter did not provide significant detail regarding the way in which different data assimilation methods work. Furthermore, it only touched on the specific data assimilation methods that had be previously used with Agent-Based Models, and did not provided any coverage of the broader field. This chapter, therefore, aims to provide a greater level of detail on the previously mentioned methods, as well as providing some coverage of other methods that exist in the field

Firstly, the process of data assimilation will be explained in more detail, framing it as a Bayesian Updating problem. The main data assimilation methods used in this investigation will then be explained; this investigation uses a method known as the Ensemble Kalman Filter (EnKF) which is based on the Kalman Filter — the chapter will first cover the Kalman Filter before moving on to the EnKF. This will be followed by a review of some of the other data assimilation methods available. The chapter will conclude by detailing some of the anticipated challenges of applying data assimilation methods in conjunction with Agent-Based Models.

4.1 Bayesian Inference and Data Assimilation

The process of data assimilation involves making use of observations along with prior knowledge (which, in our case, is encoded in a model) to produce increasingly accurate estimates of variables of interest. More formally stated, we may phrase our problem:

> Having initially defined our initial knowledge of the system based on our model, what is our updated knowledge *given* the provided observations?

This states the problem as one of updating our knowledge of the system conditioned on observations. Such a process can be achieved through a Bayesian filtering approach (Jazwinski, 1970; Bar-Shalom et al., 2004).

In applying this process to a system that we are observing, we may wish to represent knowledge as the random variables, A and B. If we consider the true system state to be represented by the random variable, A, we may represent our initial knowledge of the system state as the probability distribution, $P(A)$. The use of a probability distribution reflects the uncertainty in our knowledge regarding the system state. When producing a set of observations of the system, we may represent them as the probability distribution

$P(B|A)$, i.e. a probability distribution representing the data that we have observed given the true system state. We can then represent our conditional understanding, i.e. our updated knowledge of the system when provided with observations, as the probability distribution $P(A|B)$. These terms can be related using Bayes Theorem. Bayes Theorem can be derived rather simply. We first assume that $P(A)$ and $P(B)$ are non-zero. We define the probability of A occurring given B, i.e. $P(A|B)$, as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \tag{4.1}$$

where $P(A \cap B)$ is the probability of both A and B occurring, and $P(B)$ is the probability of B occurring. Similarly, we can define the probability of B occurring given A as:

$$P(B|A) = \frac{P(B \cap A)}{P(A)} = \frac{P(A \cap B)}{P(A)}, \tag{4.2}$$

where $P(B \cap A)$ is the probability of both B and A occurring, and $P(A)$ is the probability of A occurring; we recall that $P(A \cap B) = P(B \cap A)$. Using Equations 4.1 and 4.2, we can write:

$$\frac{P(A|B)}{P(B|A)} = \frac{\frac{P(A \cap B)}{P(B)}}{\frac{P(A \cap B)}{P(A)}}, \tag{4.3}$$

which can be simplified to the traditional form of Bayes Theorem (Gelman et al., 1995):

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{4.4}$$

When using Bayes Theorem, we typically refer to the 4 constituent parts in the following way:

- Prior — $P(A)$: This distribution represents our initial understanding of the true system state.
- Data/likelihood — $P(B|A)$: This distribution represents the distribution of data given the true system state.
- Posterior — $P(A|B)$: This distribution represents our updated understanding of the true system state
- Marginal — $P(B)$: This distribution is the marginal distribution of our data and is typically a normalising constant (Wikle and Berliner, 2007) given by:

$$P(B) = \int P(B|A)P(A)dA$$

With these definitions in mind, it should be clear that we can apply Bayes Theorem to perform inference (indeed, Bayes Theorem has been applied in a number of fields (Ellison, 2004; Von Toussaint, 2011; Etz and Vandekerckhove, 2018)). Here, we refer to inference as the process of updating and improving some knowledge about a system. This inference can seek to improve our understanding about the true system state, or about parameters governing the behaviour of the system.

This inference process is applied in Meteorology under the banner of data assimilation. Wikle and Berliner (2007) define data assimilation as:

> . . . an approach for fusing data (observations) with prior knowledge (e.g. mathematical representations of physical laws; model outputs) to obtain an estimate of the distribution of the true state of a process.

There exist a number of different schemes for tackling this problem which are often divided into two groups Talagrand (1997):

1. **Sequential**: Upon the arrival of a new observation, the model state is updated at the time of the new observation; includes Kalman Filter (Kalman, 1960; Kalman and Bucy, 1961) (and variations thereof (Evensen, 2003, 2009)), Particle Filter (Arulampalam et al., 2002).

2. **Variational**: Upon the arrival of a new observation, the model solutions are updated at all times simultaneously; includes 3D-VAR, 4D-VAR.

In both cases, data assimilation schemes are applied to systems in a iterating cycle of prediction and updating. The prediction process involves evolving the system state forwards in time according to the model used to describe the system; the updating process involves applying the chosen data assimilation technique to produce a better estimate of the system state based on the most recent results of prediction and a set of observations. Depending on how frequently observations are provided and the time-step of the model, the prediction process may be applied multiple times between updating steps.

The next two sections will go on to outline two sequential approaches: the Kalman Filter and the EnKF —the latter of which is applied in this investigation. This will be followed by an overview of some other sequential approaches and some variational approaches.

In much of the remainder of this thesis, terminology will be drawn from both Meteorology and Bayesian Statistics. In particular, there are two terminological equivalences to consider. The first of these is the equivalence between the meteorological term “forecast” and the Bayesian term “prior”; these two may be used interchangeably as pertaining to states from a model before any form of updating; this is to say, the forecast is the result of the prediction process. The second of these is the equivalence between the meteorological term “analysis” and the Bayesian term “posterior”; these two may be used interchangeably as pertaining to states from a model after updating has taken place.

4.2 Kalman Filter

One of the earliest forms of Bayesian filtering is the sequential data assimilation scheme known as the Kalman Filter (Kalman, 1960; Kalman and Bucy, 1961), which forms the foundation of this piece of work. The Kalman Filter has been used in a range of fields including aircraft tracking (Pearson and Stear, 1974), signal processing (Auger et al., 2013) and high-energy particle physics experiments (Aggleton et al., 2017). As with other sequential data assimilation schemes, the Kalman Filter operates on a model with a given set of state variables and forecasting process (i.e. a process by which to step the model state forward in time). These state variables are considered random variables — that is to say they represent the quantities of interest as some sort of distribution. The analysis process implemented by the Kalman Filter seeks to update both the model state and the associated covariance matrix upon receipt of new observations. This is undertaken as part of the predict-update cycle. The prediction process is undertaken by applying the modelling operator to both the model state and model state covariance. The update process is undertaken based on the uncertainty in the model forecasts and the observation uncertainty; as seen below, this essentially consists of taking a weighted average of the model state and the observations, with the weightings being governed by the comparative degrees of uncertainty in each of the two sources of information. The aim of such an approach is to minimise the posterior variance.

In a scenario where we have two state variables of interest, our state vector may be:

$$\mathbf{x} = [x_1, x_2]^T.$$

These state variables represent the mean of the distributions representing each quantity.

The error in these quantities are represented by a covariance matrix. The associated covariance matrix would then be:

$$\mathbf{Q} = \begin{pmatrix} \sigma_{x_1}^2 & \sigma_{x_1 x_2} \\ \sigma_{x_2 x_1} & \sigma_{x_2}^2 \end{pmatrix}.$$

The diagonal matrix entries, $\sigma_{x_1}^2$ and $\sigma_{x_2}^2$ describe the variance in the state variables, x_1 and x_2, respectively. The off-diagonal matrix entries, $\sigma_{x_1 x_2}$ and $\sigma_{x_2 x_1}$, describe the variance of the state variables with respect to each other; for example, $\sigma_{x_1 x_2}$ describes the variances in x_1 with respect to x_2. We therefore expect that the covariance matrix is symmetric and as such find that:

$$\sigma_{x_1 x_2} = \sigma_{x_2 x_1}.$$

Under the Bayesian framework outlined in Section 4.1, we refer to the model state vector and associated covariance before updating as the prior, $\mathbf{x}$ and $\mathbf{Q}$ respectively, and to the model state and associated covariance after updating as the posterior, $\hat{\mathbf{x}}$ and $\hat{\mathbf{Q}}$. Here, we note that the prior state, $\mathbf{x}$, is a vector containing the state variables, and $\mathbf{Q}$ is the covariance matrix relating to the state variables.

Given new observations, $\mathbf{d}$, the posterior state vector, $\hat{\mathbf{x}}$ and posterior state covariance matrix, $\hat{\mathbf{Q}}$, are given by:

$$\hat{\mathbf{x}} = \mathbf{x} + \mathbf{K}\left(\mathbf{d} - \mathbf{H}\mathbf{x}\right), \tag{4.5}$$

$$\hat{\mathbf{Q}} = \left(\mathbf{I} - \mathbf{K}\mathbf{H}\right)\mathbf{Q}, \tag{4.6}$$

where $\mathbf{K}$ is called the Kalman gain matrix, $\mathbf{H}$ is the observation operator and $\mathbf{I}$ is the identity matrix which has 1 along the diagonal and 0 elsewhere:

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}.$$

The role of the observation operator, $\mathbf{H}$ is to transform the state vectors between the form in which we store state variables and the form in which they are represented in observations. In the case where we track the x-y location of an object in our model and we collect data regarding the x-y location of the object, we might have the following scenario:

$$\begin{aligned}\mathbf{x} &= [x, y]^T,\\ \mathbf{d} &= [d_x, d_y]^T,\\ \mathbf{H} &= \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix},\end{aligned}$$

where the observation operator is just the identity matrix.

In another scenario in which we are modelling the x-y location of an object as well as the object's velocity in the x-y plane in our model and we only collect data regarding the x-y location of the object, we would instead have the following setup:

$$\begin{aligned}\mathbf{x} &= [x, y, \dot{x}, \dot{y}]^T,\\ \mathbf{d} &= [d_x, d_y]^T,\\ \mathbf{H} &= \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{pmatrix}.\end{aligned}$$

The Kalman gain matrix in Equations 4.5 and 4.6 is given by:

$$\mathbf{K} = \mathbf{QH}^T\left(\mathbf{HQH}^T + \mathbf{R}\right)^{-1} \tag{4.7}$$

where $\mathbf{R}$ is the observation covariance, which is defined by the uncertainty in the observations. When considering Equation 4.5, we see that the Kalman gain matrix defines the weights applied to the elements of the difference between the observations of the system state, $\mathbf{d}$, and the modelled system state, $\mathbf{x}$, when updating the state. In cases where the uncertainty in the observations is large compared to the uncertainty in the modelled state, i.e. the elements of $\mathbf{R}$ are larger than the elements of $\mathbf{Q}$, the weights contained within the gain matrix are small and therefore updates to the model state are small. In the alternative case where the uncertainty in the observations is small in comparison to the uncertainty in the modelled state, i.e. the elements of $\mathbf{R}$ are smaller than the elements of $\mathbf{Q}$, the weights contained within the gain matrix, $\mathbf{K}$, are large and therefore changes to the model state are large.

The Kalman Filter assumes that errors (such as those associated with the state variables and with the observations) are normally distributed, and under these conditions the filter provides an exact posterior estimate; over the course of this investigation, the terms 'normal' and 'Gaussian' will be used interchangeably to refer to distributions.

However, this approach suffers from two issues. The first of these is that it assumes that the operators that are applied (namely the model transition operator and the observation operator) are linear; this is often not the case with more complex systems, particularly when the system elements interact with each other (as is typically the case in agent-based models). Furthermore, as the dimensionality of the model increases, the cost of propagating forward the model state covariance may increase to the point where it is intractable.

A number of approaches have been developed which attempt to solve these problems, one of which is the EnKF which will be discussed in the next section.

4.3 Ensemble Kalman Filter

A number of approaches have been developed which attempt to solve the above problems, one of which is the Ensemble Kalman Filter (Evensen, 2003, 2009), which acts as an approximation of the Kalman Filter. This approximation is achieved through a Monte Carlo approach of using an ensemble of sample state vectors to represent the state distribution; this development mirrors the recent incorporation of Monte Carlo methods in the field of Bayesian statistics (Wikle and Berliner, 2007). This section will seek to outline the EnKF. The state is represented as an ensemble of state vectors, $\mathbf{X}$ as follows:

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] = [\mathbf{x}_i], \quad \forall i \in \{1, 2, \ldots, N\}, \tag{4.8}$$

where the state ensemble matrix, $\mathbf{X}$, consists of N state vectors, $\mathbf{x}_i$. Considering the scenario where were we are modelling the x-y location of an object with an ensemble of 3 state vectors, we would have a state ensemble of:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3] = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}$$

The mean state vector, $\bar{\mathbf{x}}$, can be found by averaging over the ensemble:

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i. \tag{4.9}$$

Similarly, the observations are represented as follows:

$$\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_N] = [\mathbf{d}_i], \quad \forall i \in (1, N), \tag{4.10}$$

with each member of the data ensemble matrix, $\mathbf{D}$, being the sum of the original observation $\mathbf{d}$, and a random vector, ϵ_i:

$$\mathbf{d}_i = \mathbf{d} + \epsilon_i, \quad \forall i \in (1, N). \tag{4.11}$$

The random vector is drawn from an unbiased normal distribution:

$$\epsilon \sim \mathcal{N}(0, \mathbf{R}), \tag{4.12}$$

where R is the data covariance matrix, just as in the case of the Kalman Filter. As with the model state, the mean data vector, $\bar{\mathbf{d}}$, can be found by averaging over the ensemble:

$$\bar{\mathbf{d}} = \sum_{i=1}^{N} \mathbf{d}_i. \tag{4.13}$$

By implementing these adaptations, the EnKF aims to address the issues faced by the original Kalman Filter with respect to forecasting the state covariance matrix; more specifically, as a result of this approach the state covariance matrix no longer needs to be forecast by applying the model operator, but instead can simply be generated as a sampling covariance (Mandel et al., 2011). Consequently, concerns over the computational cost of forecasting the covariance matrix and over the requirement that the forecasting process be undertaken by applying a linear operator to the covariance matrix are greatly reduced.

Given the above framework, the data assimilation is once again made up of the predict-update cycle, with the updating of the state ensemble, $\hat{\mathbf{X}}$, being undertaken on the basis of the following equation:

$$\hat{\mathbf{X}} = \mathbf{X} + \mathbf{K}\left(\mathbf{D} - \mathbf{H}\mathbf{X}\right), \tag{4.14}$$

where $\mathbf{H}$ is the observation operator. Note here that we do not update our state covariance matrix, instead generating a sample covariance matrix based on the state ensemble, $\mathbf{C}$:

$$\mathbf{C} = \frac{1}{N-1} \sum_{i=1}^{N} \left(\mathbf{x}_i - \bar{\mathbf{x}}\right)\left(\mathbf{x}_i - \bar{\mathbf{x}}\right)^T.$$

The Kalman gain matrix, $\mathbf{K}$ is given by

$$\mathbf{K} = \mathbf{C}\mathbf{H}^T\Big(\mathbf{H}\mathbf{C}\mathbf{H}^T + \mathbf{R}\Big)^{-1}. \tag{4.15}$$

in which $\mathbf{C}$ is the sample state covariance (used instead of the state covariance matrix $\mathbf{Q}$), and $\mathbf{R}$ is the observation covariance. We can consider $(\mathbf{D} - \mathbf{HX})$ in Equation 4.14 to be the proposed perturbation to the ensemble states, and the Kalman gain matrix, $\mathbf{K}$, to be the weight given to this perturbation (just as in Section 4.2). When the uncertainty in the observations is low in comparison to the uncertainty in the model state, the gain increases, and therefore the model state receives a larger perturbation from the provided data; conversely, when the uncertainty in the observations is high in comparison to the uncertainty in the model state, the gain decreases, and therefore the model state receives a smaller perturbation from the provided data. This, therefore, allows us to combine two separate sources of information regarding the same phenomenon — in this case our model and the observations — whilst respecting the levels of uncertainty associated with each of them.

Whilst the derivation of the EnKF assumes linearity of the system in question, it has been found to perform relatively well when applied to non-linear systems (Katzfuss et al., 2016). Working with non-linear systems is of particular importance when applying a data assimilation method to a complex system such as one modelled by an Agent-Based Model. One of the ways in which we can better handle non-linearity is through the use of the Particle Filter (PF) which will be discussed in next section, alongside a number of other data assimilation methods. The Particle Filter additionally handles scenarios in which the distributions are non-Gaussian.

4.4 Other Data Assimilation Methods

In the previous section, we have detailed the way in which the Ensemble Kalman Filter works. Whilst this investigation focuses on the application of the Ensemble Kalman Filter, there exist a broad range of other methods. Here, we discuss some of these other methods. As noted in Section 4.1, we can divide data assimilation methods into sequential methods (i.e. methods which update the present model state upon receipt of new observations) and variational methods (i.e. methods which update the both the present model states and all previous model states simultaneously). This section will, therefore, first go on to discuss sequential methods before discussing variational methods. The former will touch upon other variants of the Kalman Filter, as well as the Particle Filter (which has seen popularity in application to Agent-Based Models). The latter will touch upon methods such as 4DVAR which are commonly used in numerical weather prediction.

4.4.1 Sequential Methods

This section will outline a further selection of sequential data assimilation methods. Two of these — the Extended Kalman Filter and the Unscented Kalman Filter — are based on the Kalman Filter described in Section 4.2. Beyond these, we will also go on to outline the Particle Filter which has also been applied to Agent-Based Models.

Extended Kalman Filter

The first of these sequential data assimilation methods — the Extended Kalman Filter — is a variation on the original Kalman Filter. This filter was designed to overcome the issue faced by the original Kalman Filter when working with non-linear systems (Smith et al., 1962). To describe this method, let us first consider a description of the system. If we consider situation in which we have a vector of state variables, $\mathbf{x}$, which is updated from time-step k to time-step $k+1$ by the model process operator $\mathbf{f}$, we can write this as:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) + w_k,$$

where w_k is normally distributed noise associated with the process being modelled. Similarly, we should consider the process of producing observations, $\mathbf{y}$ at time-step k based on the observation operator $\mathbf{h}$, writing it as:

$$\mathbf{y_k} = \mathbf{h}(\mathbf{x}_k) + v_k,$$

where v_k is the normally distributed noise associated with the observation process. It is assumed that whilst they may be non-linear, the operators $\mathbf{f}$ and $\mathbf{h}$ are smooth and therefore differentiable. Based on this, the modelling and observation operators are then linearised using the Taylor Series expansion about the state at a given time-step. As such, the prediction and updating steps are modified (Ribeiro, 2004). The prediction step consists of using the model process operator to step forward the model state and the associated model state covariance matrix:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}_k(\mathbf{x}_k), & (4.16) \\ \mathbf{P}_{k+1} &= \mathbf{F}_k \mathbf{P}_k \mathbf{F}_k^T, & (4.17) \end{aligned}$$

where $\mathbf{x}_k$ is the state at time-step k, $\mathbf{f}_k$ is the model process operator at time-step k, $\mathbf{P}_k$ is the state covariance matrix at time-step k and $\mathbf{F}_k$ is the Jacobian of the model process operator at time-step k. The update step consists of updating both the model state and state covariance matrix with respect to the provided observations:

$$\begin{aligned} \hat{\mathbf{x}}_{k+1} &= \mathbf{x}_{k+1} + \mathbf{K}_{k+1}\left[\mathbf{y}_{k+1} - \mathbf{h}_{k+1}(\mathbf{x}_{k+1})\right], && (4.18) \\ \hat{\mathbf{P}}_{k+1} &= \left[\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1}\right]\mathbf{P}_{k+1}, && (4.19) \end{aligned}$$

where $\hat{\mathbf{x}}_{k+1}$ is the posterior state at time-step $k+1$, $\hat{\mathbf{P}}_{k+1}$ is the posterior state covariance matrix at time-step $k+1$, $\mathbf{K}_{k+1}$ is the Kalman gain matrix at time-step $k+1$, $\mathbf{y}_{k+1}$ is the observed state at time-step $k+1$, $\mathbf{h}_{k+1}$ is the observation operator at time-step $k+1$ and $\mathbf{H}_{k+1}$ is the Jacobian of the observation operator at time-step $k+1$. The Kalman gain matrix at time-step $k+1$, $\mathbf{K}_{k+1}$ is given by:

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}\mathbf{H}_{k+1}^T\left[\mathbf{H}_{k+1}\mathbf{P}_{k+1}\mathbf{H}_{k+1}^T + \mathbf{R}_{k+1}\right]^{-1}, \qquad (4.20)$$

where $\mathbf{R}_{k+1}$ is the observation covariance matrix at time-step $k+1$.

Whilst these adjustments help the filter to deal with non-linear systems, they only involve the use of first-order terms and exclude higher order terms; as such, in order for the process to be effective it is required that the model process is not dominated by any higher order terms (Terejanu et al., 2008); the assumption underlying this is that the systems are linear local to the point about which we are estimating and in cases where this linearity is not observed, the filter may become unstable. Furthermore, the Extended Kalman Filter still requires the propagation of the state covariance matrix which is potentially computationally expensive (as noted in the case of the Kalman Filter in Section 4.2).

Unscented Kalman Filter

Another of the sequential data assimilation methods — the Unscented Kalman Filter — is also based on the Kalman Filter outlined in Section 4.2. Just as with the Extended Kalman Filter above, the Unscented Kalman Filter seeks to deal with non-linear systems without the use of linearisation (and consequently overcoming the stability issues encountered when applying the Extended Kalman Filter). This is achieved by applying the unscented transformation (Julier and Uhlmann, 1997). Given a model state, $\mathbf{x}$, with covariance matrix, $\mathbf{P}$, this involves creating a set of sample points (referred to as *sigma points*), with the size of the set being $2n+1$ where n is the dimension of the state space (Terejanu, 2011). These points are chosen deterministically in order to ensure that their mean is exactly equal to the model state and their covariance is exactly equal to the covariance matrix; this assumes that, just as with the original Kalman Filter, the error distributions are Gaussian (Wan et al., 2001). The sigma points, χ_i (where $0 \leq i \leq n$), are defined as:

$$\chi_0 = \bar{\mathbf{x}}, \tag{4.21}$$

$$\chi_i = \bar{\mathbf{x}} + \left(\sqrt{(n+\kappa)\mathbf{P}}\right)_i, \tag{4.22}$$

$$\chi_{i+n} = \bar{\mathbf{x}} - \left(\sqrt{(n+\kappa)\mathbf{P}}\right)_i, \tag{4.23}$$

where $1 \leq i \leq n$ and κ is a scaling variable. Each of the sigma points are assigned weights, W_i, defined as:

$$W_0 = \frac{\kappa}{n+\kappa}, \tag{4.24}$$

$$W_i = \frac{1}{2(n+\kappa)}, \tag{4.25}$$

$$W_{i+n} = \frac{1}{2(n+k)}, \tag{4.26}$$

for $1 \leq i \leq n$.

For the prediction process, each of these points is then evolved by propagating them forward in time based on the model process operator outlined in the case of the Extended Kalman Filter, and the estimated model state mean, $\hat{\bar{\mathbf{x}}}$, and covariance matrix, $\hat{\mathbf{P}}$, are calculated. For the update process, the model state mean and covariance are updated according to the process outlined in the Kalman Filter.

This improves upon the Extended Kalman Filter approach in a number of ways. By virtue of not linearising the operators, the Unscented Kalman Filter is able to capture non-linearity without discarding higher order terms (Jwo and Lai, 2008). Furthermore, it removes the need to calculate the Jacobian of the operators, which results in a much less computationally expensive algorithm (Julier et al., 1995). Just as with previous algorithms, however, the Unscented Kalman Filter assumes Gaussian error distributions (Särkkä, 2013) — something that may not be the case.

Particle Filter

Just as with the previous two data assimilation methods, the Particle Filter seeks to improve upon the Kalman Filter by better handling non-linear systems; however, in addition, the Particle Filter seeks to overcome the other shortcoming of Kalman Filter-based methods — the assumption that errors are normally distributed.

The Particle Filter achieves this by undertaking an approach similar to that used in the EnKF, i.e. maintaining an ensemble. In this case, however, the ensemble members are referred to as *particles*. Each of these particles are assigned a corresponding weight. We may represent these particles and weights as the sets (Orlande et al., 2011):

$$\left\{\mathbf{x}_k^i; i \in \{1, \ldots, N\}\right\} \tag{4.27}$$

$$\left\{w_k^i; i \in \{1, \ldots, N\}\right\} \tag{4.28}$$

where $\mathbf{x}_k^i$ is the ith model state at time-step k, w_k^i is the ith weight applied to the corresponding model state at time-step k and N is the number of particles. The weights, w_k^i, at any given time-step are normalised such that they sum to 1:

$$\sum_{i=1}^{N} w_k^i = 1 \tag{4.29}$$

Just as with other data assimilation methods outlined thus far, the Particle Filter involves the processes of prediction and updating. As in the EnKF, the prediction step involves predicting forward in time each of the ensemble-member models, i.e. the particles, producing a set of samples from the prior distribution. In the case of the Particle Filter, however, the update process is different. At the very least, the update process may consist of reweighting the particles, and in many cases may also include reampling the particles (Doucet et al., 2000). The process of reweighting involves redefining the weights pertaining to each particle by comparing the particle states with the observed states; these weights are normalised according to Equation 4.29. The case where only reweighting is undertaken is typically referred to as Sequential Importance Sampling.

This method may, however, suffer some issues. The most common of these is referred to as particle degeneracy whereby a single particle dominates the distribution; this may be evidenced by the weight pertaining to a single particle being close to 1 with other weights being close to 0 (Elfring et al., 2021). The most common solution is to employ resampling. Having assigned new weights to each of the particles, we generate a new population of particles by sampling from a population of particles made up of the prior particles, $\mathbf{x}_k^i$, weighted by their given weights, w_k^i.

The aim of this approach is that, given a large enough collection of particles, i.e. as $N \to \infty$, the samples exactly represent the posterior distribution (Arulampalam et al., 2002). Whilst the Particle Filter may offer an approach that is capable of providing an approximation of the posterior distribution, this is often limited by computational cost; in cases where the conditions for the Kalman Filter are fulfilled, it is typically better to use the EnKF as it requires fewer ensemble members/particles to achieve the same result (Wikle and Berliner, 2007; Orlande et al., 2011).

4.4.2 Variational Methods

As alluded to earlier in the Chapter, sequential and variational approaches reflect two very different ways to do data assimilation. Whilst sequential data assimilation methods seek to best estimate the state of the system at the time of observation provision, variational approaches seek to identify the trajectory of the model that best fits the provide observations (Carrassi et al., 2018). This typically involves updating not only the system state estimate at the time of data assimilation, but also updating the system state estimates at each previous time-step. In order to achieve this, we aim to minimise a cost function, $J(\mathbf{x})$, where $\mathbf{x}$ is a state vector. This cost function takes different forms depending on the variational approach. One of the most common approaches is 4DVAR, for which the cost function is (Carrassi et al., 2018):

$$\begin{aligned} J(\underline{\mathbf{x}}) = \; & \frac{1}{2} \|\mathbf{x}(t_0) - \mathbf{x}_b\|^2_{\mathbf{B}_0^{-1}} + \frac{1}{2} \sum_{t=0}^{T} \|\mathbf{y}_t - \mathbf{H}(\mathbf{x}(t))\|^2_{\mathbf{R}_t^{-1}} \\ + \; & \frac{1}{2} \sum_{t=1}^{T} \|\mathbf{x}(t) - \mathbf{M}_{t-1,t}(\mathbf{x}(t-1))\|^2_{\mathbf{Q}_t^{-1}} , \end{aligned} \tag{4.30}$$

where t is the time-step in the range $\{0, \ldots, T\}$, $\mathbf{x}(t)$ is the model system state at time t, $\underline{\mathbf{x}}$ is the collection of system states, $\{\mathbf{x}(t_0), \ldots, \mathbf{x}(T)\}$, $\mathbf{x}_b$ is the prior, otherwise referred to as the *background*, $\mathbf{y}_t$ is the set of observed system states at time t, $\mathbf{H}()$ is the observation operator (as used in previous data assimilation methods), $\mathbf{M}_{ij}()$ is the model process operator which transitions the state from $t = i$ to $t = j$, $\mathbf{B}$ is the background covariance matrix, and $\|\mathbf{a}\|^2_{\mathbf{A}^{-1}} \equiv \mathbf{a}^T \mathbf{A}^{-1} \mathbf{a}$ (often referred to as the L-2 norm).

We can think of this cost function as consisting of three terms (Bannister, 2017):

$$J = J_b + J_o + J_Q, \tag{4.31}$$

with the first term pertaining to the background error, the second to the observation error and the third to the model error. With this is cost function (and indeed any other cost function used in variational data assimilation), we seek to find the model trajectory that minimises it (Ide et al., 1997).

Variational approaches such as 4DVAR may have a number of advantages, such as the ability to assimilate asynchronous observations (Kalnay et al., 2007). The method also has some disadvantages, however; one such disadvantage is that whilst it may provide an estimate of the model state, it does not provide an associated covariance by default — something that may be desirable — and the calculation of such a posterior covariance may be particularly challenging (Carrassi et al., 2018).

4.5 Challenges of Data Assimilation with an Agent-Based Model

Having outlined the various data assimilation methods above, this section seeks to describe some of the challenge we encounter in our attempt to apply them — the EnKF in particular — to Agent-Based Models of pedestrian motion. In particular, we shall highlight the challenges arising from the way in which systems are represented, the challenges arising from the non-linearity often observed in complex systems and the challenges arising from the way in which observations are generated for social complex systems and how this might relate to the data assimilation process.

System Representation

When considering the literature around data assimilation, it is noticeable that it is dominated by meteorological applications. There are, however, cases in which algorithms have been applied to scenarios in which we wish to accurately track the trajectory of an object such as aircraft or autonomous vehicles (McDougall and Moore, 2017); in fact, when the Kalman Filter was first conceived, it was applied to the Apollo space flight project (Grewal and Andrews, 2010). Superficially, there may appear to be some direct analogues between the tracking problems encountered in the aerospace industry and the pedestrian tracking problems that we are tackling in this investigation — in both cases, we aim to produce the best estimate of a location given the output of a model and some observed quantity. There is, however, a fundamental difference: the way in which the systems are modelled. In the former case, we typically model the system based on a set of differential equations (such as those describing the laws of physics), whereas the modelling of social systems such as pedestrian motion are typically undertaken via rules-based methods such as Agent-Based Modelling. In some investigations, it has been noted that way in which Agent-Based Models are defined — in terms of rules instead of sets of differential equations — presents a challenge when applying data assimilation methods (Wang and Hu, 2015). This is true of approaches such as the Kalman Filter and Extended Kalman Filter which require analytical forms for the operators; however, approaches that approximate distributions using samples such as the EnKF and Particle Filter do not suffer from this problem (Wang, 2014). This makes them more amenable to being applied to the micro-level simulation of complex systems such as those considered in this investigation.

Non-linearity

Beyond the challenges presented by the way in which the systems are being represented, i.e. equation-based vs. rule-based, there are challenges presented by the nature of the phenomena being modelled. When considering complex systems, they are typically characterised by numbers of units which interact with each other and these interactions are often considered to be non-linear (Amaral and Ottino, 2004; Motter et al., 2006). In the case of the pedestrian modelling such as the models outlined in Chapter 3, non-linearities may be introduced through a number of manners. One the one hand, we have the non-linearities introduced by the collision avoidance employed by pedestrians (although when scaled up to large populations this may be much more impactful). On the other hand, we have the discrete changes introduced by variations in destination which result in pedestrians travelling in entirely different directions. Such non-linearity may be a challenge for some forms of data assimilation such as the Kalman Filter. Additionally, these non-linearities may also result in non-Gaussian error distributions, presenting further challenges for Kalman Filter-based methods.

Observations

When producing observations of social complex systems, we rarely have complete coverage, i.e. there will be gaps in our data either spatially or temporally. This reflects one of the key motivations of our use of data assimilation touched upon in Chapter 1 where we note the issue of data sparsity alongside the issue of growing model errors. The result is that, when considering pedestrian systems, we may not have trace data for a non-zero proportion of our population and may wish to infer the traces for the remainder of the population whilst also improving our trace estimates for the proportion for which we have observations. When considering systems which are described by systems of differential equations, we may be able to assume some degree of continuity between spatially separated points and as such, when applying data assimilation with spatially sparse data, may be able to interpolate values in-between. In the case of complex systems such as pedestrian systems, however, we are considering pedestrians as our units, not points in space; further our pedestrian units may act in a non-linear fashion based on interactions with other pedestrians and choice of destination. Consequently, there may be some challenges in applying data assimilation to accurately simulate a system with partial coverage (Clay et al., 2020).

The application of such trace data assumes that there is an obvious equivalence between the pedestrian units in the simulation and the observed quantities; this might reflect a scenario in which we have GPS trackers on individual pedestrians. Another way in which individual level observations may be generated is through the use of computer vision technologies such as that used to inform the design of `StationSim_GCS` (as noted in Section 3.2 whereby the calibration of the model made use of data from Zhou et al. (2012)). In this case, we may have complete coverage of all of the pedestrians in the system, however, drawing an equivalence between observed pedestrians and agents inside an Agent-Based Model may be non-trivial; as noted in Section 3.2.3, the approach used to identify pedestrian traces is not perfect, and generates partial trajectories which have to be joined up.

Alternatively, we might consider the scenario in which we do not have trace-like observations (either complete or incomplete), but instead have aggregated data — a scenario which may be more likely as it may more effectively protect the privacy of pedestrians in the system. In such a case, we may have intermittent observations of the number of pedestrians congregated in different geographical spaces at a given time (as is the case with the data generated by Leeds City Council (Whipp et al., 2021)). When presented with such data, we are faced with a similar challenge as with the previous type of data but on a potentially greater scale. The challenge here lies in the issue of disaggregating the observations and drawing equivalence between observed pedestrians and agents in the model.

One further observation-related issue that we may wish to consider is that of unobserved variables. When initialising a model, we typically base our initial model state on some form of empirical data. In the case of a pedestrian Agent-Based Model, this may involve defining the starting location of the individual pedestrian agents. When drawing equivalence between observations and model states, it should be trivial to initialise an agent with a given starting location upon first observing a pedestrian entering a system. This does not, however, account for the set of unobserved pedestrian attributes. This may include factors such as a pedestrians speed or their destination within the environment. Some of these may be inferred directly by observing the pedestrians (this may be the case with an attribute such as speed). This does not, however, account for all attributes.

4.6 Concluding Remarks

This chapter has sought to outline the process of data assimilation, framing it as a Bayesian updating problem. In particular, it has focussed on The Ensemble Kalman Filter and the Kalman Filter on which it is based. In addition, some attention has been paid to other data assimilation methods that exist, considering both sequential and variational approaches.

Having described a range of data assimilation methods, we have outlined some of the challenges that may be faced when applying them to Agent-Based Models in Section 4.5, highlighting which types of filters are particularly susceptible where appropriate. When considering issues around model representation, it was noted that ensemble-based data assimilation methods were likely to be the most appropriate to be applied to complex systems represented by rules-based models. When considering the issues around nonlinearity, it was noted that some Kalman Filter-based approaches, particularly the Kalman Filter itself, may not be appropriate given their difficulties in handling nonlinear systems. Whilst this may suggest that the most appropriate data assimilation method might be the Particle Filter, we should also consider the computational cost associated with this algorithm. As noted in Section 4.4, in many cases the EnKF may achieved results that are almost as good as the Particle filter with much smaller ensemble sizes/ It is on these grounds that the EnKF was chosen as the data assimilation method to be used in this investigation.

Having outlined the data assimilation approach to be used in this investigation, the following three chapters outline a set of experiments and associated results. The first of these — Chapter 5 — shows that the EnKF can be applied to an Agent-Based Model of pedestrian motion (albeit the Toy Model described in Section 3.1), and can effectively improve the accuracy with which we simulate the trajectories of pedestrians across a space. The second of these — Chapter 6 — takes this a step further by applying the data assimilation method to a more realistic model of pedestrian motion in `StationSim_GCS` which is described in Chapter 3.2. In doing so, we expect to demonstrate the effectiveness of the filter in handling a more realistic form of non-linearity. Finally, Chapter 7 apples the filter to estimate not only the trajectories of the pedestrians across the environment, but also to consider the scenario in which we have to deal with unobserved pedestrian attributes; in particular, this will deal with unknown pedestrian destinations, and look to estimate based on intermittent observations of pedestrian locations.

Chapter 5

Data Assimilation for Location Estimation: Toy Model

In Chapter 4, the Ensemble Kalman Filter (EnKF) was outlined. The aim of this chapter is to test the EnKF and show that the application of the EnKF can help to improve the accuracy with which an ABM simulates a system. To achieve this, it focuses on the implementation of the EnKF to the first of the models presented in Chapter 3, referred to as the Toy Model. This model is a very simple Agent-Based Model (ABM) which simulates the motion of pedestrians across a hypothetical train station. Whilst the model is not a realistic reflection of any real-world scenarios is it, instead, intended as a preliminary demonstration of the capabilities of the EnKF to improve the simulation accuracy of an ABM.

Evaluation of simulation accuracy — a measure that will be defined in Section 5.1.1 — is achieved through the use of synthetic data generated from the Toy Model itself. Whilst the EnKF maintains an ensemble of models, a separate instance of the model is run in order to act as a representation of the real system, i.e. to act as a ground truth, against which to compare. Data which are assimilated by the EnKF are generated from snapshots of this ground truth instance of the model by adding noise; this seeks to emulate the noise incorporated into observations by sensors.

The first of these focuses on outlining the experiments to be run with the EnKF[1] and the Toy Model. In particular, this focuses on detailing how each experiment will be set up, what parameters are provided and how each experiment is evaluated. The second section focuses on the results of these experiments, drawing some conclusions regarding the effectiveness of the EnKF under different parameter regimes and noting the limitations of this set of experiments[2].

5.1 Experimental Design

Under normal circumstances, an ABM seeks to simulate a scenario observed in the real-world, and consequently is formed of components pertaining to the relevant aspects of the real-world system; this can include the relative size and shape of the environment, the position of any obstacles in said environment and the behaviours of the individu-

[1]Code for the implementation of the EnKF used in this thesis can be found in `Projects/ABM_DA/stationsim/ensemble_kalman_filter.py` in the archive of the dust repository

[2]The experiments run for this chapter can be found in the notebooks found in `Projects/ABM_DA/stationsim/experiments/enkf_experiments/results_1/noteooks/` in the archive of the dust repository

als characterised in the model. Under such conditions, we are able to evaluate the performance of the model by comparing the phenomena observed in the model to the phenomena observed in the real-system.

This can be undertaken in a number of ways, considering metrics at an individual level, a fully aggregate level or somewhere in-between. An individual-level metric might involve measuring the distance between the location of individual agents in the model and the location respective individual pedestrians in the ground truth system; a fully aggregated metric might involve measuring the difference between the number of agents in the system environment in the model and the number of pedestrians in the system; a metric that lies between these two scales might involve comparing the density of agents in the model environment with the density of pedestrians in the ground truth system. In this case, however, the model is not based on a specific scenario in the real-world, and so there are no observations against which to compare the outputs of the model. A different approach, therefore, needs to be undertaken.

5.1.1 Developing a Model Baseline

The proposed solution to this problem is to consider an instance of the model as a ground truth state against which model outputs can be compared. This is an approach that has emerged in recent investigations that have focused on implementing data assimilation schemes in conjunction with ABMs (Malleson et al., 2020; Clay et al., 2020, 2021). Just as the EnKF steps each of the ensemble-member models forward through time, the model representing the ground truth state of the system can be stepped forward through time. This allows for the comparison of the mean state of the models contained by the EnKF with the ground truth state at each step in model time (as well as comparison of individual ensemble-member models with the ground truth state if desired). Such comparison can be undertaken at both an individual-agent-level or at greater levels of aggregation.

The first step in undertaking experiments to demonstrate the effectiveness of the EnKF in improving the simulation accuracy of the Toy Model is to evaluate the performance of the model without the EnKF. This provides us with a baseline against which to compare the performance of the model with the EnKF. For this purpose, the following accuracy metric is proposed: distance between agent positions in the model and agent positions in the ground state, averaged over all agents. The distance between the ith agent in the model and the respective agent in the ground truth system is given by d_i:

$$d_i = |\hat{\mathbf{x}}_i - \mathbf{x}_i|, \tag{5.1}$$

where $\hat{\mathbf{x}}_i$ is the x-y position of the ith agent estimated by the model and $\mathbf{x}_i$ is the x-y position of the ith agent in the ground state system. We can then calculate the average distance, $\bar{d}$ as

$$\bar{d} = \frac{1}{N}\sum_{i=0}^{N}|\hat{\mathbf{x}}_i - \mathbf{x}_i| = \frac{1}{N}\sum_{i=0}^{N} d_i, \tag{5.2}$$

where N is the total number of agents in the system. Given this metric, we can measure the error present in the model at each time-step.

When employing Agent-Based Models to simulate social systems, it is common to incorporate some computational stochasticity to emulate the variability of individual social behaviours. As such, two different instances of a model are unlikely to replicate exactly the same results (unless the simulations have the same random seed). In this model, stochasticity is incorporated through a number of sources:

- Pedestrian choice of entrance gate;
- Pedestrian choice of exit gate;
- Pedestrian target speeds;
- Pedestrian collisions.

A pedestrian's entrance gate is drawn at random from a uniform distribution containing all of the entrance gates and its exit gate is drawn from at random from a uniform distribution containing all of the exit gates. A pedestrian's target speed is chosen at random from a range of speeds between the minimum speed and maximum speeds, with the maximum being drawn from a normal distribution. Finally, when pedestrians interact, the try to avoid each other by making a random choice of whether to go to the left or the right of the agent or obstacle obstructing them; each of the options has an equal probability of occurring.

Given the stochasticity often involved in ABMs, individual model runs typically diverge from what we observe in the true system (as outlined in Chapter 2). When modelling a system using an ABM, it is, therefore, common to run a number of copies of the model, i.e. an ensemble, and to then take the average of the ensemble as the

Parameter	Value
Population size	100
Number of entrances	3
Number of exits	2
Environment height	100
Environment width	200

Table 5.1: Table of model parameters used for estimating the baseline level of error.

output model state. This Monte Carlo-like approach allows us to gain a general idea about the behaviour of the model. With this in mind, the process of developing baseline against which to compare will be as follows:

1. Run a single instance of the toy model with parameters outlined in Table 5.1 — this will represent the ground truth state;
2. Run an ensemble of instances of the toy model, each with the same model parameters outlined in Table 5.1, with each instance being initialised with the same starting conditions including agent-level parameters;
3. Calculate the average model state over the ensemble for each time-step;
4. Calculate the distance between agents in the average model state and the ground truth state for each time-step based on Equation 5.1;
5. Calculate the average distance per agent for each time-step based on Equation 5.2;
6. Plot the variation of average distance over time.

5.1.2 Initial Implementation of the Ensemble Kalman Filter

Having outlined the procedure by which to establish a baseline regarding model performance in the absence of data assimilation in the previous section, this section seeks to outline the process by which we first confirm that the addition of the EnKF results in an improvement in model performance. This is achieved by running the EnKF in conjunction with the Toy Model for a single set of model and filter parameters (outlined in Tables 5.2 and 5.3 respectively). The results of this can then be compared against a baseline (as outlined in Section 5.1.1) with the same set of model parameters from Table 5.2. A comparison of the results involves comparing how the model error (i.e. the average distance between the model agent positions and pseudo-truth agent positions) evolves over model time in each of the two cases. This allows us to evaluate whether the inclusion of the data assimilation scheme has a positive impact with regards to the accuracy with which the model simulates the system.

Errors are calculated for both the baseline and the EnKF at model time-steps at which data are assimilated by the filter; in the case of the filter, the posterior model state is used, i.e. the agent positions estimated by the model after having incorporated the observational data. The errors calculated with the prior model state can also be compared against the posterior model state at each data assimilation time-step to show the impact of assimilating data in each case.

There is an expectation that at the beginning of the model time (i.e. $t = 0$), both the baseline and the filter will have approximately no error; this is because in both cases, the models are exact copies of the ground truth model. Consequently, at $t = 0$, all agents are at the same entrance gates in the ground truth model, the baseline model and the filter model. A similar scenario should be encountered when the model has finished running. The model finishes running under the condition that all agents have reached their pre-allocated destinations; when this occurs, all agents will once again be at the same exit gates in the ground truth model, the baseline model and the filter model.

The expectation is, therefore, that there will be some differences between the three models in how the agents traverse the station concourse from their respective entrances to the their respective exits. The baseline model will receive no updates with regards to the ground truth between starting and finishing. The filter model, on the other hand, will receive information on a periodic basis with the period provided in Table 5.3. This information will be provided in the form of observations of individual agent x-y locations for each individual agent. An observation of the ith agent's location, $\mathbf{y}_i$, is produced by adding unbiased normally distributed random noise to the location of the agent taken from the ground-truth model:

$$\mathbf{y}_i = \mathbf{x}_i + \mathcal{N}(0, \sigma^2) \tag{5.3}$$

Parameter	Value
Population size	100
Number of entrances	3
Number of exits	2
Environment height	100
Environment width	200

Table 5.2: Table of model parameters.

Parameter	Value
Ensemble size	20
Assimilation period	20
Observation error standard deviation	1.0

Table 5.3: Table of filter parameters.

5.1.3 Exploring the Impact of Filter Parameters

Having established that the inclusion of the EnKF in the modelling process improves the accuracy with which the model simulates the system in Section 5.1.2, this section seeks to explore the extent to which different filter parameters impact the performance of the filter. The filter parameters to be considered are:

- Ensemble size: the number of model copies maintained by the EnKF.
- Observational uncertainty: the standard deviation of the random noise used to produce the observations from the ground-truth.
- Assimilation period: The number of model time-steps between each consecutive incorporation of data into the model by the EnKF.

When considering the scale of the observation uncertainty used here, it is comparable to the scale of the side-steps taken by the agents to avoid other pedestrians; in both cases, the deviations introduced are drawn from unbiased normal distributions with standard deviation of 1.0. When comparing this to the scale of the environment in Table 5.2, we can see that the majority of the deviations introduced by both the side-stepping mechanism and by the observation noise are on the order of magnitude of 1% of the environment height.

In order to explore this parameter space, the model will be run in conjunction with the filter for a range of values for each parameter outline in Table 5.4. This table details the range of values used for the ensemble size, assimilation period and standard deviation of the observation error used in the filter. The values used for the ensemble size and assimilation period (i.e. $[2, 5, 10, 20, 50]$) have been chosen to allow us to explore how the efficacy of the filter is impacted by the scale of these parameters. The values for the standard deviation of the observation error, on the other hand, vary linearly in steps of 0.5 from 0.5 to 2.5 (inclusive) to allow us to explore how the efficacy of the filter is impacted by linear changes in the standard deviation.

Given the parameter values outlined in this table, there are 125 combinations. For each combination, the system is run 10 times to produce a reliable average value. Each of the 10 instances of the filter being run with a specific parameter combination are independent, and as such do not share the same ground truths; this aims to ensure that the results that are produced are not specific to a single ground truth, but are instead generalisable. Exploring the parameter space in this manner allows for results to be viewed in two manners:

1. Exploring the variation of error with a single filter parameter whilst holding the other two filter parameters constant. In the case of exploring the impact of the variation of ensemble size, this would involve exploring the variation of error with ensemble size for constant values of observational uncertainty and assimilation period. The output of this analysis is a series of box plots for each value of ensemble size showing the distribution of error over the filter instances. This is similar to an approach undertaken by Malleson et al. (2020) who explore the impact of assimilation period.

2. Exploring the variation of error with two filter parameters whilst hold one filter parameter constant. This would allow for the exploration of the trade-off between two parameters. In the case of exploring the impact of the variation of ensemble size and observational uncertainty, this would involve exploring the variation of error with ensemble size and observational uncertainty for a constant value of assimilation period. The output of this analysis is a heatmap in which the x-y location would indicate the combination of ensemble size and observa-

Parameter	Value
Ensemble size	$[2, 5, 10, 20, 50]$
Assimilation period	$[2, 5, 10, 20, 50]$
Observation error standard deviation	$[0.5, 1.0, 1.5, 2.0, 2.5]$

Table 5.4: Table of filter parameter ranges

tional uncertainty, and the colour would indicate the error averaged over all filter instances for that parameter combination. This is also similar to an approach undertaken by Malleson et al. (2020) who explored the trade-off between the number of particles used in a Particle Filter and the model population size.

5.2 Results

This section aims to outline the results arising from the experiments outlined above. These experiments consist of developing a baseline against which to compare subsequent results, showing that an initial implementation of the Ensemble Kalman Filter is effective in improving the accuracy with which the model simulates the system and exploring the impact of different model and filter parameters on filter efficacy.

5.2.1 Developing a Model Baseline

This section seeks to outline the results of the process of establishing a model baseline as described in Section 5.1.1. This is achieved by setting up a single instance of the toy model which acts as a ground truth against which to compare. An ensemble of copies of the ground truth model are then created. The production of the ensemble involves copying all facets of the model and its agents. Consequently, each of the agents in the ensemble-member models has the same entrance gate, exit gate and speeds as the corresponding agents in the ground truth model.

The output of the ensemble of models consists of the average position of each agent in the system taken over the ensemble. In this case, the ensemble consists of 100 models. The result of this process is a time-series of the average error per agent, where agent error is taken as the distance between the agent position given by the average of the ensemble and the position of the agent in the ground truth system (as defined in Section 5.1.1). This time-series is shown in Figure 5.2.

When considering this figure, there are a number of points to consider. At the outset of the model (i.e at $t = 0$), the average error per agent is 0. This is a consequence of each of the ensemble member models being a copy of the ground state model — at $t = 0$, the location of each agent in each of the ensemble member models is the same as it is in the ground truth model.

This is somewhat mirrored by a very low average error per agent at the end of the model. This is, again, a consequence of each of the ensemble member models being a copy of the ground truth model — each agent is assigned the same exit gate (and consequently the same destination location) in the ensemble member models as they are in the ground truth model. The error at the end of the model run is not, however, 0 as it is at the outset of the model. This is a result of the way in which agent deactivations are handled in the model — agents are not deactivated when they reach the exact location of their exit gate but instead are deactivated when they are within a certain radius of this target destination.

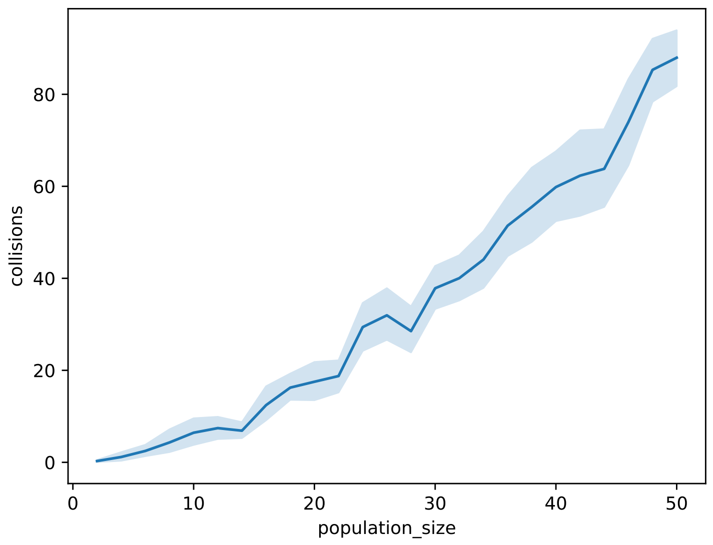
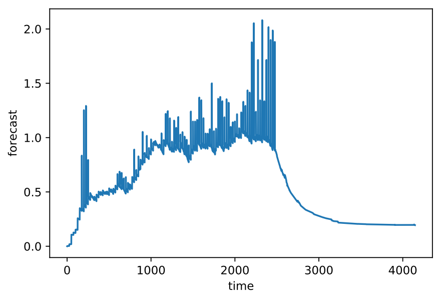
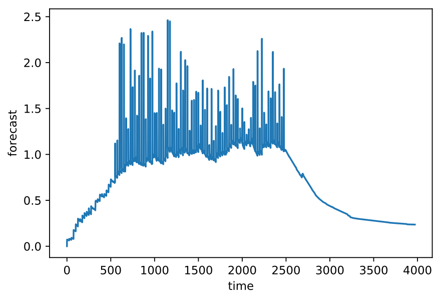


Figure 5.1: Variation in number of inter-agent collisions with population size.

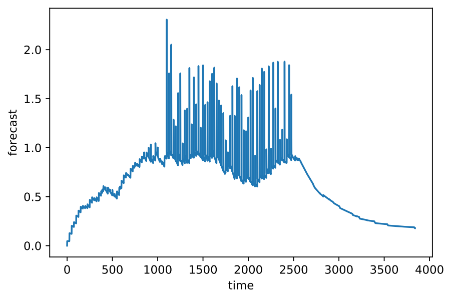
Between these two points in time, there is an increase in the average error per agent. At the beginning of the modelling process, each of the agent reside at their respective



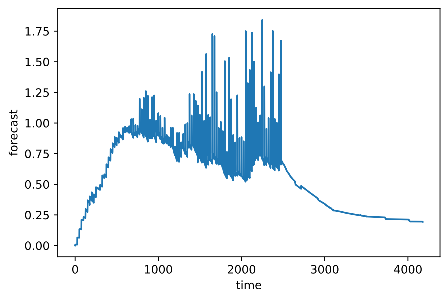
(a) Ensemble size = 10



(b) Ensemble size = 20



(c) Ensemble size = 50



(d) Ensemble size = 100

Figure 5.2: Toy model benchmarking — variation in average error per agent with model time.

entrance gates, and as model time steps forward, agents begin to enter the system through their gates. Given that each of the ensemble member models are copies of the ground truth model, and that the motion of the agents in this model is largely linear, we might expect that the average error per agent would remain low over the course of the model run. This is not, however, the case. This is a result of inter-agent interactions, i.e. collisions between agent. Such collisions occur when an agent proceeds towards its target destination and are obstructed by another agent in-front of them. When such a situation arises, the moving agent attempts to side-step to avoid a collision; this side-step is modelled as a random process, introducing stochasticity into the model. When these random side-steps occur, models diverge from each other and from the ground truth model, and consequently the average error per agent increases. The number of collisions that occur in a model run are a function of the size of the population running through the model environment. Figure 5.1 shows that as the number of pedestrian agents in the model increases, the number of collisions that occur over the course of the model run increases.

As model time is stepped forward, more agent enter the system, and therefore there are more collisions. Over the course of the model run, there is some variability in the model error which is reflected in the oscillations in the error shown in Figure 5.2. These oscillations reflect sudden increases and decreases in the error. Each increase pertains to a side-step which introduces an error relative to the ground truth, with decreases reflecting the agents returning to their original path towards their destination. In Figures 5.2a, 5.2b, 5.2c and 5.2d, we see how the evolution of average error per agent varies with the ensemble size, i.e. the number of models in the ensemble. In comparing these figures, we see that whilst there is some are some differences in how the error evolves, much of the behaviour remains unchanged by the changes in ensemble size: the initial error is still 0, the final error is still a low non-zero value and the is an increase in average error per agent in-between with a lot of variability.

It should be noted that each of the subfigures in Figure 5.2 pertain to experiments which make use of their own respective ground truth, i.e. each of the experiments in the subfigures has a different ground truth. As a result, whilst the overarching error trajectories from the start until the end are similar, we find that there are some differences. Furthermore, given the non-zero size of the side-steps taken by the agents, the errors have a lower-bound as the experiments proceed; consequently, we find that an increase in the ensemble size results in little improvement in the error.

Figure 5.3 is provided to illustrate the nature of the side-step behaviour. In this figure, we plot the trajectories of 3 agents traversing the environment (in a simulation of 100 agents). The figure shows the difference between an agent that is able to traverse the environment from its entrance to its exit without interacting with other agents (Agent 1) and agents that have interacted with other agents during their journey across the environment (Agents 0 and 2). In the latter case, we observed the agents side-stepping perpendicular to their direction of travel.

5.2.2 Initial Implementation of the Ensemble Kalman Filter

This sections seeks to outline the results of the initial implementation of the EnKF. Having established a baseline level of performance for the model in simulating the synthetic system, we can now compare this against the performance of the model in

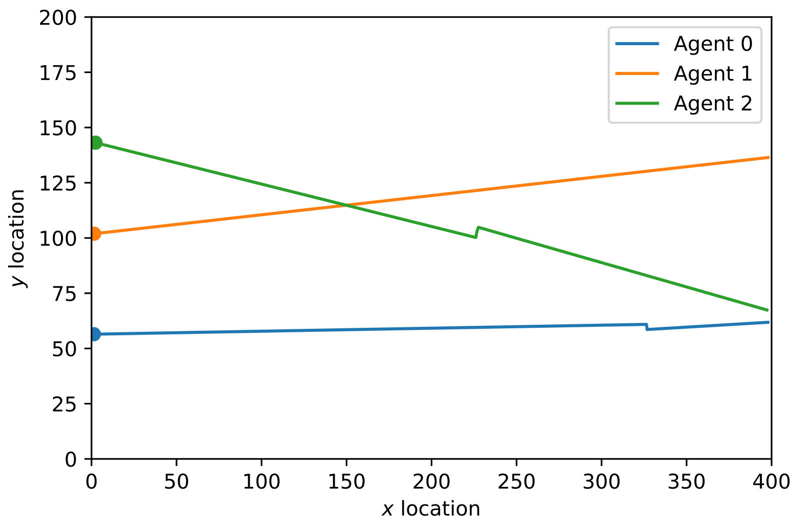


Figure 5.3: Trajectories of agents passing traversing the environment. Contains a sample of 3 agents from a simulation containing 100 agents. Points indicating starting location for each agent.

combination with the filter. This is approached with a single set of model and filter parameters (Tables 5.2 and 5.3). Given that the previous results section shows that the baseline performance of the model does not, to a large extent, depend on the size of the ensemble, the same number of models is used in baseline ensemble as in the filter ensemble.

The results for this experiment are summarised in Figures 5.4. In Figure 5.4, we are able to compare the variation in observation error, forecast error, analysis error and the error in the benchmark over model time. In this figure, it can be observed that the average observation error remains approximately constant over all model time. This is reflective of the standard deviation of the normally distributed random noise that was added to the ground truth in order to produce the observations. Beyond this, it can also be seen that the in each of the model errors (forecast, analysis and benchmark), the error starts at 0 and is has a very small non-zero value at the end of the model run. This arises for the same reasons outlined in Section 5.2.1 — at the beginning of the model run, all models in both the filter ensemble and benchmark ensemble are copies of the ground truth model, and given that each agent is assigned the same exit gate in

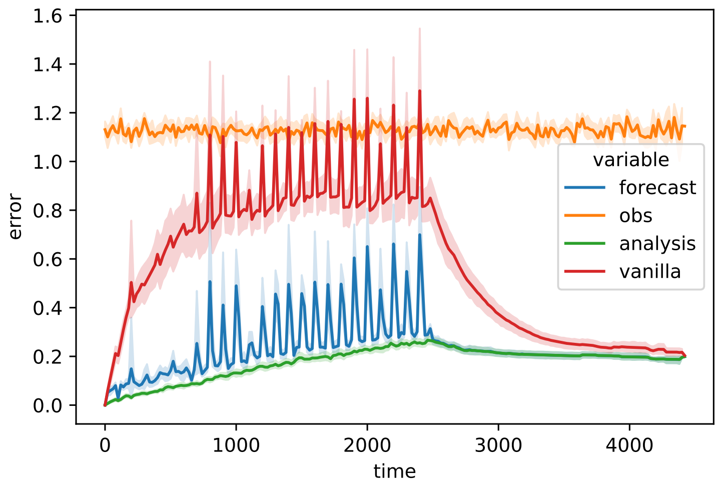


Figure 5.4: Comparison of how model errors vary over time. `vanilla` reflects the error in the mean state of the benchmarking ensemble. `obs` reflects the error in the observations used at each assimilation time-step. `forecast` reflects the error in the mean of state of the filter ensemble before data assimilation. `analysis` reflects the error in the mean state of the filter ensemble after data assimilation.

each of the models, they each arrive at the same target destination at the end of the model.

We can see that some of the detail found in Figure 5.2 is lost in Figure 5.4. This results from the data being sampled at different frequencies. In the benchmarking experiment, the figure shows data for each model time-step. In this experiment, the figure shows data only at the assimilation time-steps.

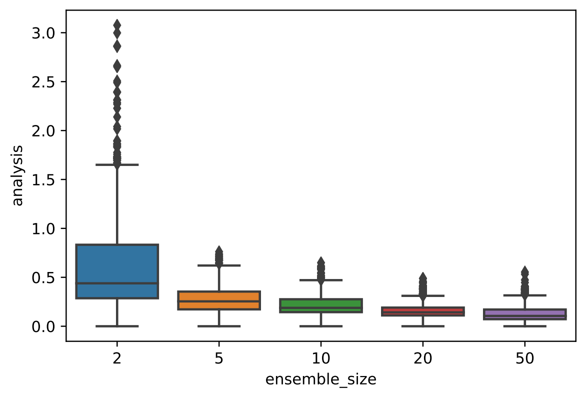
As model time progresses, the forecast error, analysis error and benchmark error each grow. Given that the average error per agent is calculated by dividing by the total population size, this growth is due, in part, to the increasing number of agents that are active and that are therefore contributing to the average error (agents that are not yet active contribute near-zero error). The growth in error is also a consequence of more agents entering the system, agents interacting with each other and causing agents in the ensemble member models to choose different paths to their counterparts in the ground truth model. This growth in error is much larger in the baseline than in the forecast or analysis. At each assimilation time-step, observation data is assimilated into the models contained within the filter, correcting divergences from the ground truth model state. In spite of the introduction of observation data, the ensemble member models still have a tendency to diverge from the ground truth state; this is reflected by the growths in forecast error, with the forecast error consistently exceeding the analysis error. In each case, however, the periodic introduction of observation data improves the accuracy of the ensemble of models in the filter.

5.2.3 Exploring the Impact of Filter Parameters

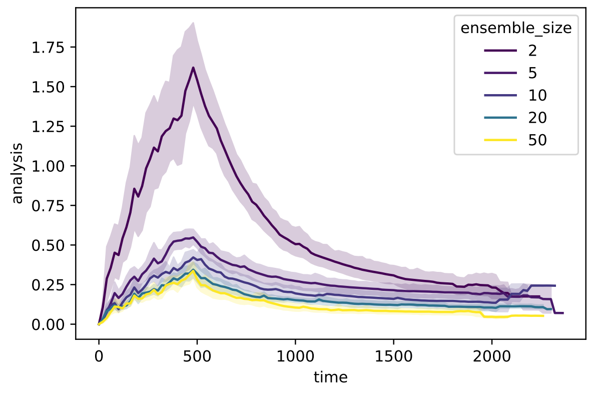
Whilst the previous experiment focussed on the comparison of forecast error, observation error and analysis error with a baseline error, this experiment seeks to focus exclusively on the analysis error of the filter, and how it is impacted by variations in different filter parameters. The filter parameters of interest in this case are the ensemble size, the assimilation period and the standard deviation of the error attached to the observations. This is achieved through two approaches: a univariate approach in which we consider how the analysis error varies in response to changes in each of the parameters individually, and a bivariate approach in which we consider how the analysis error varies in response to changes in pairs of parameters.

The first of these — the univariate approach — is undertaken by considering the results in Figures 5.5, 5.6 and 5.7. Each of these figures comprise of two subfigures. In each of the two subfigures, the data plotted reflect are the analysis errors from all of the 125 parameter combinations (each run 10 times), but focus on highlighting how the analysis error changes in response to one of the three filter parameters being explore. In Figure 5.5, the subfigures show the analysis errors vary with respect to variations in the ensemble size, and as such contain data from all of the realisations for the different combinations segmented by ensemble size. Similarly Figures 5.6 and 5.7 show how the analysis errors vary with respect to variations in the assimilation period and observation noise standard deviation respectively. The first subfigure in each case is a collection of box plots which summarise the analysis error for all time-steps in each of the realisations. As an example, consider Figure 5.5a — the first of these box plots shows the variation in analysis error for all time steps in all realisations for each parameter combination which has an ensemble size of 2. The second subfigure in each case is a collection of line plots which summarise how the analysis error analysis error varies over time for all of the realisations. As a similar example, if we consider

Figure 5.5b, the line labelled 2 shows how the analysis error varies over time for all of the realisations in which the ensemble size was 2. The shaded area around each of the lines represents the 95% confidence interval.



(a) Box-plot of how analysis error varies with ensemble size.



(b) Line-plot of how analysis error varies with ensemble size.

Figure 5.5: Plots of how analysis error varies with ensemble size.

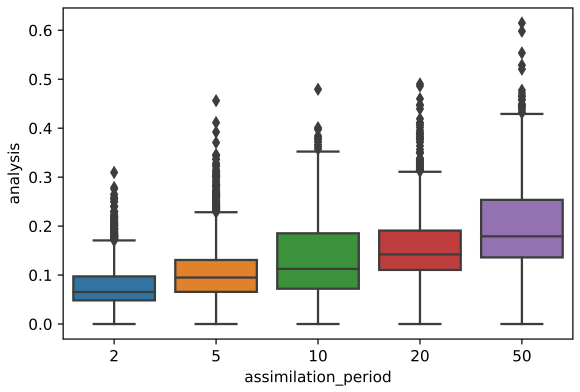
In Figure 5.5, we see two plots pertaining to the variation of analysis error in response to changes in the ensemble size of the filter. In this case, the assimilation period has been fixed to 20, the population size has been fixed to 20 and the observation error standard deviation has been fixed to 1.0. The ensemble size has been varied between 2 and 50 as per the values outlined in Table 5.4. The box-plot in Figure 5.5a shows how the analysis error varies with the ensemble size, with each box-plot pertaining to an ensemble size capturing all data-points for the specific set of parameters (ensemble size, assimilation period and observation error standard deviation). This shows that as the ensemble size of the filter increases, the analysis errors decrease, whilst also becoming less varied.

In order to ascertain whether the differences between the distribution of errors for different ensemble sizes are significant some statistical testing is employed. First of all, the collection of analysis errors for each of the ensemble sizes is tested for normality using a Shapiro-Wilk test. For each of the ensemble sizes, the p-values are sufficiently small that we can reject the null hypothesis of normality at the 5% confidence level, concluding that there is sufficient evidence to suggest that the analysis errors are not normally distributed in each case. This leads us to perform a Kruskal-Wallis test, which indicates whether there is a significant difference in the median values of each of the analysis error distributions. The p-value for this test is once again sufficiently low that we may reject the null hypothesis that the median value of all distributions is the same, indicating that there is a significant difference between the median values of the analysis error distributions.

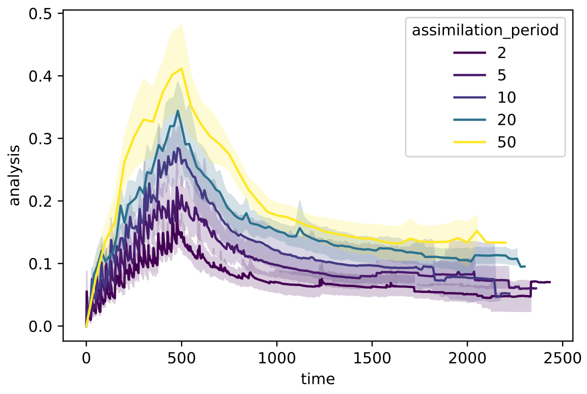
These differences are reflected in the line-plot in Figure 5.5b, which shows how the analysis error varies over time for each of the ensemble sizes; the figure shows a large difference between the analysis errors observed for an ensemble size of 2 and the analysis errors observed for other ensemble sizes.

Similarly in Figure 5.6, we see plots pertaining to the variation of analysis error in response to changes in the assimilation period. For these runs, the ensemble size has been fixes to 20, the population has been fixed to 20 and the observation error standard deviation has been fixed to 1.0. The assimilation period has been varied between 2 and 50 as per the values outlined in Table 5.4. The box-plot in Figure 5.6a shows how the analysis error varies with assimilation period, with each box-plot pertaining to an assimilation period capturing all of the data-points for the specific set of parameters. This shows that as the assimilation period of the filter increases, the analysis errors increase.

Again, in order to ascertain whether the differences between the distributions of analysis errors for difference assimilation periods are different, some statistical testing is employed. The analysis errors for each of the assimilation periods are tested for nor-



(a) Box-plot of how analysis error varies with assimilation period.

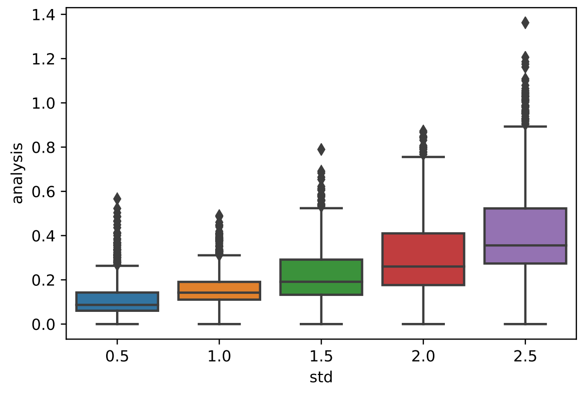


(b) Line-plot of how analysis error varies with assimilation period.

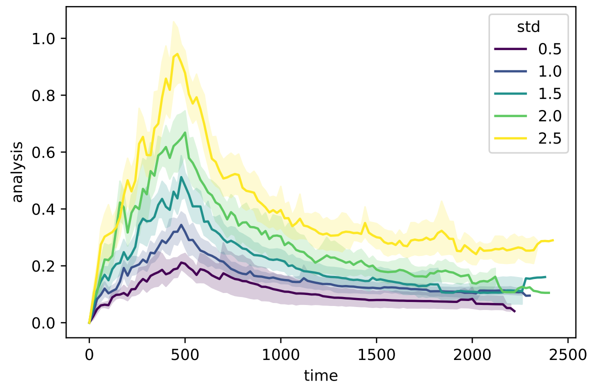
Figure 5.6: Plots of how analysis error varies with assimilation period.

mality using the Shapiro-Wilk test. As with the tests run for different ensemble sizes, the *p*-values returned by the tests on each of the assimilation periods are sufficiently small that we are able to reject the null hypothesis of normality a 5% confidence level, indicating that each of the distributions are non-normal. This leads us to perform a Kruskal-Wallis test to explore whether there is a significant difference in the median values of the analysis error distributions for the different assimilation periods. The *p*-value for this test is once again sufficiently low that we may reject the null hypothesis that the median value of all distributions is the same, indicating that there is a significant difference between the median values of the analysis error distributions.

These differences are reflected in the line-plot in Figure 5.6b, which shows how the analysis error varies over time for each of the assimilation periods. The trend observed above regarding the increase in analysis error is a direct consequence of the increasing assimilation period — as the assimilation increases, the frequency with which observations are integrated into the filter is reduced, allowing the models within the filter to diverge from the ground truth state to a greater extent.



(a) Box-plot of how analysis error varies with observation error standard deviation.



(b) Line-plot of how analysis error varies with observation error standard deviation.

Figure 5.7: Plots of how analysis error varies with observation error standard deviation.

In Figure 5.7, we see plots pertaining to the variation of analysis error in response to changes in the standard deviation of the observational error. For these runs, each of the ensemble size, assimilation period and population size have been fixed to 20, and the standard deviation of the observation error has been varied linearly between 0.5 and 2.5 as per the values outlined in Table 5.4. The box-plots in Figure 5.7a show how the analysis error varies with standard deviation, with each box-plot pertaining to standard deviation and capturing all of the data-points for the specific set of parameters. This shows that as the standard deviation of the observation error increases, the analysis error increases.

Once again, we perform statistical tests to ascertain whether the differences between the analysis error distributions for each standard deviation are significant. This begins with testing each of the distributions for normality — in each case the p-values returned by the Shapiro-Wilk test are sufficiently low that we may reject the null hypothesis of normality at a 5% confidence level. We subsequently perform a Kruskal-Wallis test and find that the p-value is sufficiently low that we can deduce that there is a significant difference between the median values of the analysis error distributions.

These differences are reflected in the line-plot in Figure 5.7b, which shows how the analysis error varies over time for each of the standard deviations. The trend of increasing analysis error with increasing observation error standard deviation is partly a consequence of the increasing standard deviation and partly of the way in which the EnKF works. An increased standard deviation in the observation error indicates a higher level of uncertainty in the observation, and consequently the noise added to the ground truth state may be larger, pulling the agents in the model states towards incorrect locations when observational data is introduced. Simultaneously, the EnKF accounts for the increase in uncertainty in observation data by reducing the gain, meaning that the observation data has less influence on the posterior model state; this may, however, mean that posterior model state does not see a marked improvement over the forecast.

Whilst the above analysis has focused on how the variation of a single filter parameter influences the distribution of analysis error, the following analysis with focus on how the analysis error varies in response to changes in two parameters simultaneously. In order to achieve this, a certain level of data reduction is required: instead of considering the distribution of all of the analysis errors for different parameter values, we shall instead consider the mean analysis error for each combination of parameters, i.e. for each combination of ensemble size, assimilation period and observation error standard deviation. This information is then encapsulated in the contour maps seen in Figures 5.8, 5.9 and 5.10. These figures denote average analysis error by colour. Contour lines on these plots indicate lines of equal analysis error. This allows us to explore how trade-offs between two parameters may influence the performance of the EnKF.

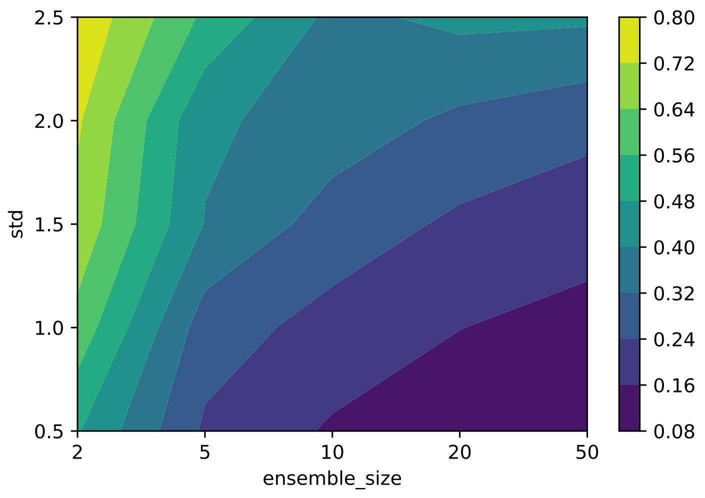


Figure 5.8: Contour map of variation in analysis error with ensemble size and observation error standard deviation.

Considering Figure 5.8, the contour maps shows the variation of average analysis error with filter ensemble size and observation error standard deviation. As has been observed in the univariate analysis above, an increase in ensemble size results in a reduction in the analysis error, i.e. as we move from left to right in the contour map, we move from regions of high analysis error to areas of low analysis error. Similarly, we can observe that an increase in the standard deviation of observation errors results in an increase in the analysis error, i.e. as we move bottom to top in the contour map, we move from regions of low analysis error to regions of high analysis error. This indicates that the best scenario would be when we are able to run a filter with a large ensemble size and low uncertainty observations, whilst the worst scenario would be running a filter with a small ensemble size and high uncertainty observations. Beyond this, we may use the contours to draw equivalences between different combinations of ensemble size and standard deviation, e.g. we might expect to observe a similar level of analysis error between filter runs which use an ensemble size of 10 and a standard deviation of 0.5 and filter runs which use an ensemble size of 50 and a standard deviation of 1.0. This, perhaps, emphasises the importance of high-quality observations.

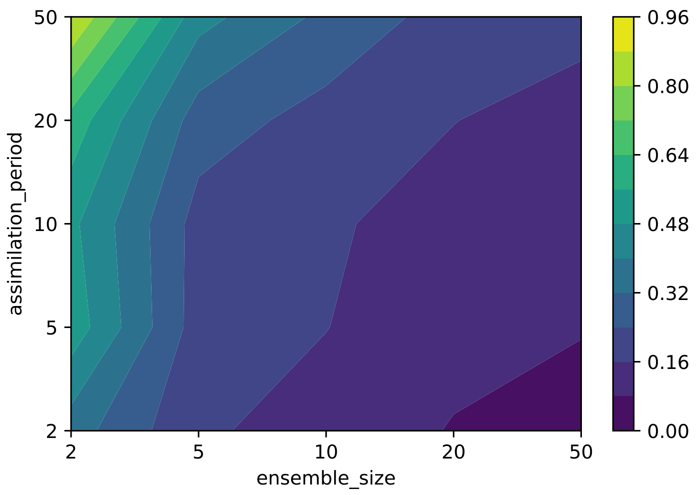


Figure 5.9: Contour map of variation in analysis error with ensemble size and assimilation period.

Considering Figure 5.9, the contour map shows the variation of average analysis error with filter ensemble size and filter assimilation period. Once again, we observe an reduction in analysis error with increasing ensemble size and an increase in analysis error with increasing assimilation period. This suggests that the best scenario would be when we can run a filter with a large ensemble size and receive observations with a high frequency, i.e a low assimilation period. Using the contours to draw equivalences between different combinations of ensemble size and assimilation period, we can see that running a filter with an ensemble size of just over 5 with very frequency observations (i.e. an assimilation period of 2) may return similar analysis errors to a filter with a larger ensemble size of 20 in conjunction with less frequent observations at an assimilation period of 20.

Finally, we may consider Figure 5.10 which shows the variation of average analysis error with filter assimilation period and observation error standard deviation. Here we observe an increase in analysis error with increasing assimilation period and an

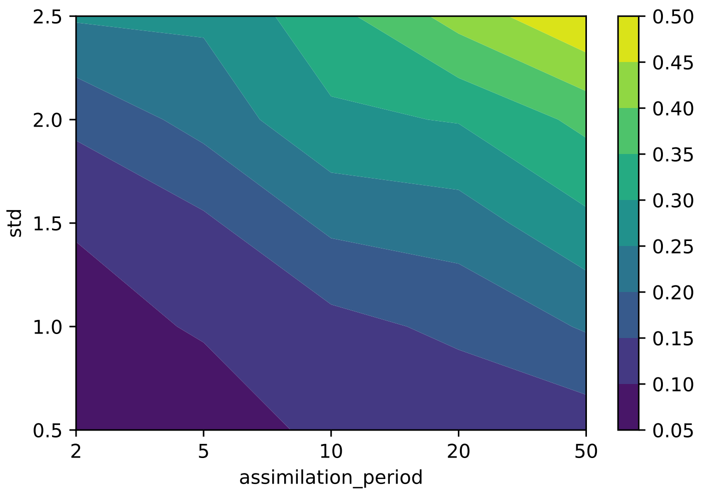


Figure 5.10: Contour map of variation in analysis error with assimilation period and observation error standard deviation.

increase in analysis error with observation error standard deviation. When provided control over these two parameters, the best scenario would be to assimilate low uncertainty observations on a frequent basis, i.e. a low assimilation period with a low observation error standard deviation, whilst the worst scenario would be to assimilate high uncertainty observations with low frequency. In the latter scenario, the ensemble member models would go long spans of time without any updates from observational data and so would be liable to diverge progressively more from the ground truth state. When considering this trade-off, we may consider that infrequently assimilating observations with a standard deviation of 1.0 at an assimilation period of 50 may return similar analysis errors to a case where slightly higher uncertainty observations with a standard deviation of 1.5 are assimilated much more frequently with an assimilation period of 10.

5.3 Concluding Remarks

This chapter has focused on the application of the EnKF to the Toy Model described in Section 3.1 to demonstrate that such an application can be effective in improving the accuracy with which an ABM can simulate a system. This has involved a series of experiments being run.

The first of these experiments sought to establish a baseline level of accuracy with which an ensemble of models can simulate the pedestrian system without any data being assimilated to update the models. This experiment found that at both the outset of a simulation run and at the end, the average error per agent across the ensemble of models was very low; this was a result of the fact that each of the models in the ensemble was a copy of the model which represented the ground truth. Consequently, each of the agents in the ensemble of models had the same origin and destination as their corresponding agent in the ground truth model. Between these two times, there was a growth in error as agents enter the system, interacting with each other. Furthermore, it was found that this patten was not greatly impacted by the ensemble size.

The next of the experiments demonstrated the initial implementation of the EnKF in conjunction with the Toy Model. This showed that the introduction of the EnKF successfully improved the accuracy with which the ensemble of models could simulate the system. Both the forecast and analysis errors observed in this experiment were an improvement on the benchmarking ensemble error, as well as the observation error.

The final experiment in chapter focused on the impact of different filter parameters on filter performance; in particular, it focused on the impact of filter ensemble size, the number of time-steps between consecutive observations being assimilated into the ensemble, i.e. the assimilation period, and the standard deviation of the noise attached to the observations. This was split into an analysis of how each of the parameters contribute to filter error in isolation, and an analysis of the trade-off between pairs of parameters. In the first part, it was found that:

- As ensemble size increases, average analysis error decreases;
- As assimilation period increases, average analysis error increases;
- As standard deviation of observation noise increases, analysis error increases.

In the second part, it was found that the decline in filter performance associated with increasing assimilation period and standard deviation of observation noise could be offset by increases in the size of the filter ensemble. An advantage of the way in which this analysis was undertaken is that the repeated realisations pertaining to each combination of filter parameters were independent and as such it was ensured that the results were not specific to an individual realisation but instead were more general.

Having established that the Ensemble Kalman Filter can be applied to an Agent-Based Model of pedestrian motion in this chapter, the next chapters seeks to expand upon this by applying the filter to the `StationSim_GCS` model — a model which is founded on observations of pedestrians crossing the main concourse at Grand Central Station in New York.

Chapter 6

Data Assimilation for Location Estimation:
`StationSim_GCS`

In the previous chapter, preliminary results were shown which indicated that the Ensemble Kalman Filter (EnKF) could be implemented in conjunction with an Agent-Based Model of pedestrian motion, and was effective in improving the accuracy with which the model simulated a system. The data assimilation scheme was tested for a range of different filter parameter values, and it was found that improvements in filter performance resulted from increases in the ensemble size, reductions in the standard deviation of the observation error and reductions in the number of time-steps between successive attempts to assimilate observational data into the system.

Whilst the data assimilation scheme was shown to be effective in improving the accuracy with which the model simulated the system, it should be noted that the model itself was limited in a number of ways:

- **Linear Movement:** The manner in which agents moved through the system was largely linear; we would typically expect to observe some pedestrians making deviations in their trajectories.
- **Agent Interactions:** Agents moved along trajectories which were, more often than not, almost parallel to other agents, limiting the types of pedestrian interactions that might occur; we would expect to observe pedestrians interacting through parallel, anti-parallel and perpendicular motion.
- **Separation of Entrances and Exits:** When considering the model used in the previous chapter, it should be noted that each of the gates on the left-hand side of the environment were used exclusively as entrances by the pedestrian agents, and the gates on right-hand side of the environment were used exclusively as exits; such a discrete segregation of function may occur in some scenarios, but is not representative of all pedestrian systems, with many pedestrian systems including co-located entrance/exit points.
- **Interactions with Environmental Obstacles:** The model used in the previous chapter did not contain any form of obstacles for the agents to interact with; whilst not an integral part of an Agent-Based Model of pedestrian motion, it is not unusual to observe stationary obstacles in urban environments in which pedestrians gather.

This is not representative of the majority of pedestrians systems that we would like to model. This chapter, therefore, seeks to apply the EnKF to a more realistic Agent-

Based Model of pedestrian motion. The EnKF is applied to the `StationSim_GCS` model which seeks to model the motion of pedestrian in the concourse of Grand Central Station in New York (this model is documented in Chapter 3). The aim of this chapter is to show that the implementation of the Ensemble Kalman Filter to a more realistic model of pedestrian motion can still be effective in improving the accuracy with which the model simulates the system. This is achieved through a series of experiment outlined in Section 6.1. As in the previous chapter, this starts with a benchmarking experiment which establishes the baseline level of error to be expected when using an ensemble of `StationSim_GCS` models alone to simulate the system. This is then followed by experiments which seek to show that, just as with the Toy Model in Chapter 5, the EnKF is effective in improving the accuracy with which we can simulate a pedestrian system — in this case, the motion of pedestrians around the concourse at Grand Central Station in New York. This is achieved by running multiple instances of the EnKF in conjunction with the Grand Central Station Model multiple times for the same set of model and filter parameters, and observing how the error across the ensemble of filters varies over time in comparison to the corresponding benchmarking models.

6.1 Experimental Design

In this section, the experiments run using `StationSim_GCS` in conjunction with the EnKF are outlined[1]. These experiments are visually outlined in Figure 6.1.

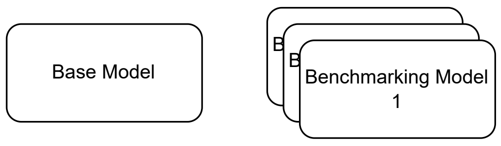
The initial experiment seeks to establish a benchmark against which to compare subsequent implementations of the EnKF. This is achieved by running an ensemble of models, each initialised as duplicates of a base model which is used to generate pseudo-truth values for the system state. This is shown in Figure 6.1a.

The second experiment seeks to explore how the accuracy with which individual models within an ensemble Kalman Filter varies across the ensemble. This is achieved by running a single EnKF which maintains a benchmarking ensemble of models which provides a baseline against which to compare results, along with and ensemble of models which are periodically updated by the EnKF assimilation process. These member models are updated using synthetic observation data generated from the pseudo-truth

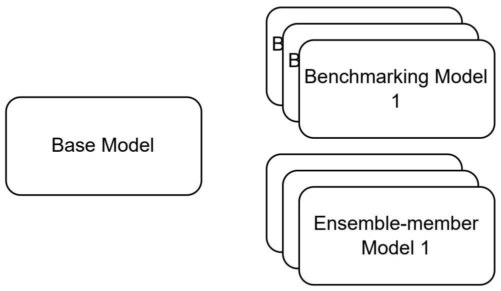
[1]The experiments run for this chapter can be found in the notebooks found in `Projects/ABM_DA/experiments/enkf_experiments/results_2/notebooks/` in the dust repository archive

states from the base model. In such a situation, we are able to compare the average error per agent in each of the ensemble member models at each assimilation time-step. This is shown in Figure 6.1b.

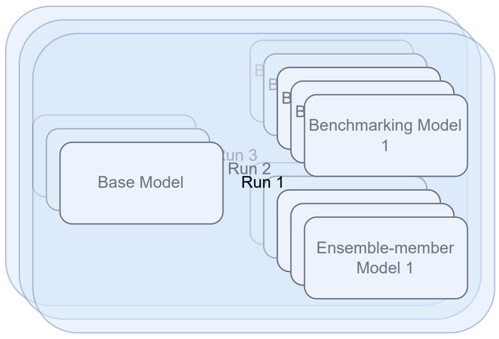
The final experiment in this chapter looks to take this exploration a step further, seeking to capture the variation in error at an ensemble level. This involves running a collection of EnKF for the same set of model and filter parameters, and in each case gathering data regarding the variation in the error in the ensemble mean state over time, comparing this with the variation in the corresponding collection of benchmark errors. This is shown in Figure 6.1c.



(a) Experiment 1: Benchmarking



(b) Experiment 2: Exploring Ensemble-Member Models



(c) Experiment 3: Implementing the Ensemble Kalman Filter

Figure 6.1: Graphical outline of experiments.

In these figures, we see that each filter has a base model associated with it which is used to represent the ground truth of the system in question (as alluded to in the previous Chapter). From a computational perspective, each of these filters can be thought as "containing" the respective base model[1]. In practicality, the ground truth of a system is not known and therefore a filter would be expected to function without a base model. The purpose of the base model for these experiments is simply to provide a state against which to compare the performance of filters.

6.1.1 Developing a Model Baseline

The initial experiment to be performed is to develop a model baseline, establishing the effectiveness of `StationSim_GCS` in modelling a system in the absence of any information whilst running. This is approached in a similar fashion to that outlined for the Toy Model in Section 5.1.1.

In order to evaluate the performance of a model, we once again consider the distance between the position of an agent estimated by the benchmarking ensemble of models and the position of the corresponding agent in the base model, d_i:

$$d_i = \begin{cases} |\hat{\mathbf{x}}_i - \mathbf{x}_i| & \text{if } i\text{th agent is active;} \\ 0 & \text{otherwise,} \end{cases} \tag{6.1}$$

where $\hat{\mathbf{x}}_i$ is the x-y position of the ith agent estimated by the benchmarking ensemble of models and $\mathbf{x}_i$ is the x-y position of the ith agent in the ground state system, i.e. the base model. This expression for the distance between an agent's position in the base model state and in the filter ensemble mean state is conditioned on whether the agent is active or not — in the case of inactive agents, the distance is assigned as 0. As noted in Chapter 3, an agent is deemed to be active from the time that it enters the environment until the time that it exits the environment, and is otherwise inactive; this is to say that an agent is inactive until the time that it leaves its entrance gate and after it has reached its exit gate.

An agent can be deemed to be active based on one of two sets of information:

1. Information regarding whether it is active in the base model, i.e. in the pseudo-ground truth.

2. Information regarding whether it is active in the estimating ensemble.

[1]Given that the filter has been coded in an object-oriented manner to produce an `EnsembleKalmanFilter` class, the base model to which a filter pertains is considered a class attribute. This can be observed in `Projects/ABM_DA/stationsim/ensemble_kalman_filter.py`.

For the purpose of this set of investigations, whether or not an agent is active will be gauged based on information from the estimating ensemble. The justification for this choice is that in scenarios in which we did not have access to the ground truth, we would not have access to a base model. It is, therefore, more reasonable to based our assessment of an agent's status on information that would be available in the form of the ensemble of models. In this investigation, an agent is considered active if its most common (i.e. modal) status across the ensemble is active.

As an example, consider an ensemble of 5 models, with each model containing 1 agent. Recall from Chapter 3 that an agent status of 0 represents an agent that has not yet been activated, an agent status of 1 represents an agent that is active in the system, and an agent status of 2 represents an agent that has completed its journey across the environment and has been deactivated. If we were to have the following vector of agent statuses, **s**, across the ensemble of models:

$$\mathbf{s} = [0, 1, 0, 1, 1],$$

we would see that the agent was yet to be activated in $\frac{2}{5}$ of the ensemble member models, and was active in $\frac{3}{5}$ of the ensemble member models. We would, therefore, conclude that most common status was 1, and that the agent was active. If, however, we were to have the following vector of agent statuses:

$$\mathbf{s} = [2, 2, 1, 2, 2],$$

we would see that the agent has completed its journey and been deactivated in $\frac{4}{5}$ of the ensemble member models, and was still active in $\frac{1}{5}$ of the ensemble member models. We would, therefore, conclude that the most common status was 2, and that the agent was not active.

When considering active agents, we can define the calculation of distance as:

$$d_i = \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2}, \tag{6.2}$$

where $\hat{x}_i$ is the x-position of the ith agent estimated by the model, $\hat{y}_i$ is the y-position of the ith agent estimated by the model, x_i is the x-position of the ith agent in the base model and x_i is the x-position of the ith agent in the base model.

We can, therefore, calculate the distance, d_i, for each agent. To gain an idea of the error across the whole system, we calculate the average distance over all active agents in the system, $\bar{d}$:

$$\bar{d} = \frac{1}{N} \sum_{i=1}^{N} d_i, \tag{6.3}$$

where N is the number of *active* agents. This average distance, $\bar{d}$, can then be used to measure the error in an ensemble of models given the ensemble mean state for a given time-step and the base model state at the same time-step.

Given this measure of error, we can, therefore, explore how the accuracy with which the `StationSim_GCS` model varies over time using the following steps:

1. Create an instance of the model to be considered the base model which provides pseudo-truth states of the pedestrian system.

2. Create an ensemble of 100 models, each of which is a copy of the base model; this means that each of the duplicates in the ensemble contain the same information regarding which exits each of the agents will enter and exit through, as well as at what time they will be activated within the model. These models, however, are liable to diverge from the base model due to the collisions that occur between pedestrian agents.

3. Iterate each of the base model and the ensemble of models forward for each time-step. At each time-step, calculate average model state for each agent in the system population, and calculate the average error per agent between this average model state and the pseudo-truth state generated from the base model for this time-step.

In following these steps, we create a base model which is used to generate a ground truth and an ensemble of models from which we can obtain the average behaviour behaviour by averaging across the ensemble. The accuracy with which this average behaviour simulates the ground truth generated by the base model is assessed by considering the error between the base model and the average of the ensemble. This is applied to ensembles with increasing population sizes (as per Table 6.1) to explore how this behaviour varies with population size.

Having followed these steps to collect data regarding how the error of an ensemble of models varies over time without any observations being assimilated, we can then plot this as a time-series. We expect that, whilst the initial error in the benchmarking ensemble will be low by virtue of the ensemble-member models being copies of the

Parameter	Value
Population size	$[10, 20, 50, 100]$
Ensemble size	100
Number of entrances	11
Number of exits	11
Environment height	700
Environment width	740

Table 6.1: Table of model parameters used for estimating the baseline level of error.

base model, this error will grow rapidly as agents enter the environment. As a growing number of agents are present in the system, the chance of inter-agent interactions occurring increases, and as such so does the chance that the ensemble mean state diverge from the base model state. With no observations being assimilated into the ensemble of models, these divergences go uncorrected. Consequently, we expect to observe an increase in the benchmarking error. As the ensemble of models near completion, i.e. a state where all of the agents in the ensemble member models have finished their journeys and are deactivated, we expect to see the average error per agent to fall, nearing 0. As with the expectation that the initial average error per agent is low, this would be a consequence of each of the ensemble member models being a copy of the base model, and therefore each of the agents in these member models having the same target destination as the corresponding agent in the base model.

6.1.2 Exploring Ensemble Member Models

After having established a benchmark for the accuracy with which the `StationSim_GCS` model can simulate the trajectories of pedestrians moving around the concourse of Grand Central Station in New York, the next experiment aims to explore the variation amongst the models within the ensemble of an EnKF whilst observations are being assimilated into the ensemble. In order to achieve this, distances are calculated not between the ensemble mean state and the base model state, but between each of the ensemble member model states and the base model state. If we consider d_{ij} to represent the distance error of the jth model's state for the ith agent compared to the ith agent in the base model, this can be calculated as follows:

$$d_{ij} = \begin{cases} |\hat{\mathbf{x}}_{ij} - \mathbf{x}_i| & \text{if } i\text{th agent is active;} \\ 0 & \text{otherwise,} \end{cases} \tag{6.4}$$

where $\hat{\mathbf{x}}_{ij}$ is the x-y position of the ith agent in the jth model and $\mathbf{x}_i$ is the x-y position of the ith agent in the ground state system, i.e. the base model; whether or not an agent is active is dictated by the modal agent activity across the ensemble of models. Given this expression, we can calculate the mean error per agent for each ensemble member model, $\bar{d}_j$, at a given time-step:

$$\bar{d}_j = \frac{1}{N} \sum_{i=1}^{N} d_{ij}, \tag{6.5}$$

where N is the number of active agents based on the modal agent activities across the ensemble.

Based on this, we can construct a vector containing all of the mean errors per agent for each of the ensemble member models:

$$\begin{aligned} \bar{\mathbf{d}} &= \left[\bar{d}_1, \ldots, \bar{d}_M\right] & (6.6) \\ &= \left[\bar{d}_j\right], \forall j \in (1, M), & (6.7) \end{aligned}$$

where M is the ensemble size. A vector of average errors per agent for each ensemble member models can then be calculated for each time-step.

Based on this error calculation process, we can explore the variation of error across the ensemble using the following steps:

1. Create an EnKF with a population size of 20 pedestrians, containing a base model and an ensemble of 20 duplicates (given that marginal improvements in performance of increasing from 20 to 50 ensemble-member models in the previous chapter was not substantial, and to reduce computational cost) of the base model. The base model provides pseudo-truth states of the pedestrian system. As with the ensemble of models in the benchmarking experiment, the ensemble in this case inherits information regarding which gates each of the agents enter and exit through, and at what time they are activated.

2. Iterate each of the base model and ensemble of models forward for each time-step. If the time-step is an assimilation time-step, i.e. a time-step in which synthetic

data are produced from the base model, the ensemble of models are updated using the update procedure outlined in Chatper 4. Assimilation time-steps occur with a fixed period of 20 time-steps between data assimilation updates — between data assimilation updates, filter time-steps consist only of using the model to predict the system state at the next time-step. At each assimilation time-step, errors are calculated between the following pairs:

- The base model state and each of the ensemble member models after updating with observations.
- The base model state and the mean state of the ensemble of models after updating with observations.
- The base model state and the mean state of the benchmarking ensemble (i.e. the baseline error).

Having collected data regarding how each of these three types of error vary over time, we can then plot the as time-series and compare them. It is expected that over the course of the ensemble run, the error in the ensemble mean state, the benchmarking ensemble mean state and each of the ensemble member models will be low at the beginning and end of the ensemble run, with increases in error in-between; in each case, agents in ensemble member models will enter the system and exit the system at the same locations as in the base model as each of the ensemble member models are copies of the base model, and error grow in-between as larger proportions of the agent population enter the system, interacting with each and causing their respective models states to diverge from the base model.

Whist it is expected that each of these errors will follow this pattern, it is expected that the growth in error between ensemble start and finish will be greater in the benchmarking ensemble mean state than in the filter ensemble mean state. This is because, whilst each of the member models of each of the ensembles are copies of the base model and so have their agents starting and finishing at the same locations, the benchmarking ensemble models will not benefit from the assimilation of observations, and consequently when the locations of agents in these models diverge from those in the base model, they will continue uncorrected whilst the corresponding agents in the filter ensemble models will be perturbed by the observations derived from the base model.

It is expected that whilst the error time-series for each of the ensemble member models will follow a similar pattern to the benchmarking ensemble mean and the filter ensemble mean, there will be some variation across the models, with some models incurring consistently larger errors than the filter ensemble mean and others incurring smaller errors than the filter ensemble mean. We expect, however, that the error for the filter ensemble mean will always lie within the range of the errors for the ensemble member models.

6.1.3 Implementing the Ensemble Kalman Filter

The experiments described in the previous section will show that, as expected, the error in the ensemble mean state accurately reflects the variation in error over time, and that it does not differ greatly from the error in each of the ensemble member models. On this basis, this experiment focusses on comparing the variation in the ensemble mean state error over time with the error in the benchmarking ensemble as well as with the error in the observations of the pedestrians' locations on the station concourse; this considers both the ensemble mean state before and after assimilating observations, i.e. the forecast error and the analysis error. The experiment will make use of the parameters outlined in Table 6.2.

Once again, we calculate errors as the distance between the 'estimated' agent location and the agent's location in the pseudo-truth base model:

$$d_i = \begin{cases} |\hat{\mathbf{x}}_i - \mathbf{x}_i| & \text{if } i\text{th agent is active;} \\ 0 & \text{otherwise,} \end{cases} \tag{6.8}$$

where $\hat{\mathbf{x}}_i$ is the estimated x-y position of the ith agent and $\mathbf{x}_i$ is the x-y position of the ith agent in the base model. Note that in this case, an 'estimate' of an agent's location can be either the location given by the forecast ensemble mean state, the analysis ensemble mean state, the location given by the benchmark ensemble mean state or the observed location — in each case the distance is calculated in the same manner. As previously, this distance depends on whether an agent is active — in the case of agents for which the modal activity is active across the filter ensemble, the distance is given by the expression in Equation 6.2, otherwise the distance is given as 0.

Consequently, an average error in each of the four state estimates across all pedestrians can also be calculated in the same manner:

$$\bar{d} = \frac{1}{N} \sum_{i=1}^{N} d_i, \tag{6.9}$$

Parameter	Value
Population size	20
Ensemble size	100
Assimilation period	20
Observation noise standard deviation	1.0

Table 6.2: Table of filter parameters used for the EnKF.

where N is the number of active pedestrians in the system (with active agents again being defined as those for which the modal activity is active across the ensemble).

Based on this error calculation process, we can explore the variation of error across the ensemble using the following steps:

1. Create an EnKF containing a base model and an ensemble of 20 duplicates of the base model. The base model provides pseudo-truth states of the pedestrian system. As with the ensemble of models in the benchmarking experiment, the ensemble in this case inherits information regarding which gates each of the agents enter and exit through, and at what time they are activated.

2. Iterate each of the base model and ensemble of models forward for each time-step. If the time-step is an assimilation time-step, i.e. a time-step in which synthetic data are produced from the base model, the ensemble of models are updated using the update procedure outlined in Chatper 4. Assimilation time-steps occur with a fixed period of 20 time-steps between data assimilation updates — between data assimilation updates, filter time-steps consist only of using the model to predict the system state at the next time-step. At each assimilation time-step, errors are calculated between the following pairs:

   - The base model state and the ensemble mean state before updating with observations (i.e. the prior error).
   - The base model state and the ensemble mean state after updating with observations (i.e. the posterior error).
   - The base model state and the observations provided to the ensemble for updating (i.e. the observation error).

- The base model state and the mean state of the benchmarking ensemble (i.e. the baseline error).

This process is repeated 20 times for the same model and filter parameter values. Having collected the aforementioned information over the course of the ensemble run, we can then plot them as time-series and compare how they evolve. Given the repetitions of the experiment process, the information can be summarised in the time-series plot using the median error at each assimilation time-step (across the repetitions) along with confidence intervals.

As previously, we wish to explore the variations in errors in two manners: how each type of error varies over time and how the different types of error compare with each other. Having run the previous experiment (and the benchmarking experiment), we expect that error in the benchmarking ensemble will evolve as observed previously. The filter ensemble mean error per agent considered in the previous experiment pertains to the analysis error considered in this experiment, and as such, we expect that the analysis error will evolve as observed in the previous experiment. Just as in the previous experiment, the analysis error is expected to be lower than the benchmark error due to the assimilation of data into the ensemble of models.

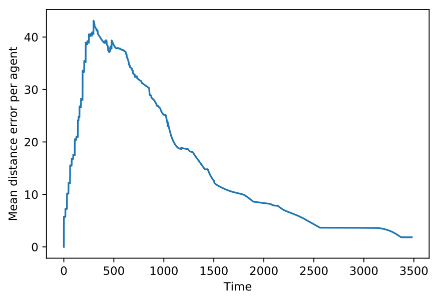
The forecast error in this experiment represents the filter ensemble mean error per agent prior to the ensemble member model states being updated with synthetic data from the pseudo ground truth base model state; consequently, it is expected to evolve in a similar manner to the analysis error, although it expected to be higher; the updating of ensemble member model states based on synthetic observations is expected to be mostly corrective — i.e. it should reduce the average error per agent at each assimilation time-step with respect to the base model state.

The average observation error per agent is expected to be close to constant over the course of the filter run. This is to due the way in which observations are produced — they are derived from the base model state by adding unbiased normally distributed random noise with a constant standard deviation, and therefore the error should match this standard deviation. However, due to the finite population over which these errors are averaged, this error will experience some variations. Furthermore, as the filter nears completion, the number of *active* members of the population will decrease, and as such the number of agents over which the error is averaged will fall meaning that the average observation error per agent may experience larger fluctuations.

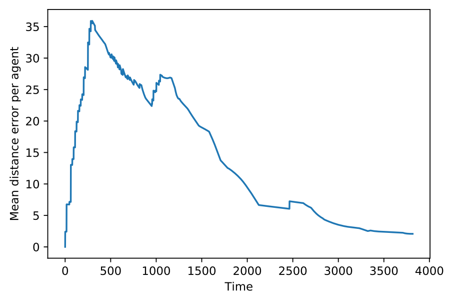
6.2 Results

6.2.1 Developing a Model Baseline

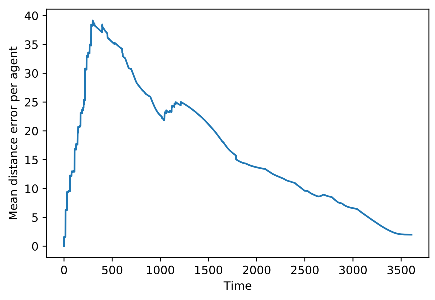
Having run the experiment outlined in Section 6.1.1, the results have been plotted in Figure 6.2. This figure contains a series of subfigures showing the variation in error in a benchmarking model for a series of different population sizes; Figure 6.2a shows the evolution of average error per agent for a population of 10 agents, Figure 6.2b for 20 agents, Figure 6.2c for 50 agents and Figure 6.2d for 100 agents.



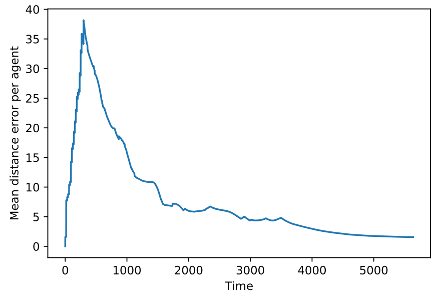
(a) 10 agents



(b) 20 agents



(c) 50 agents

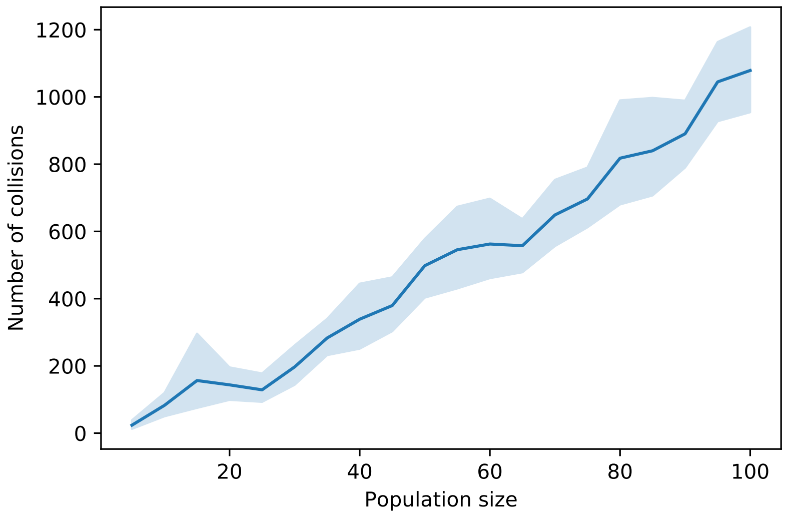


(d) 100 agents

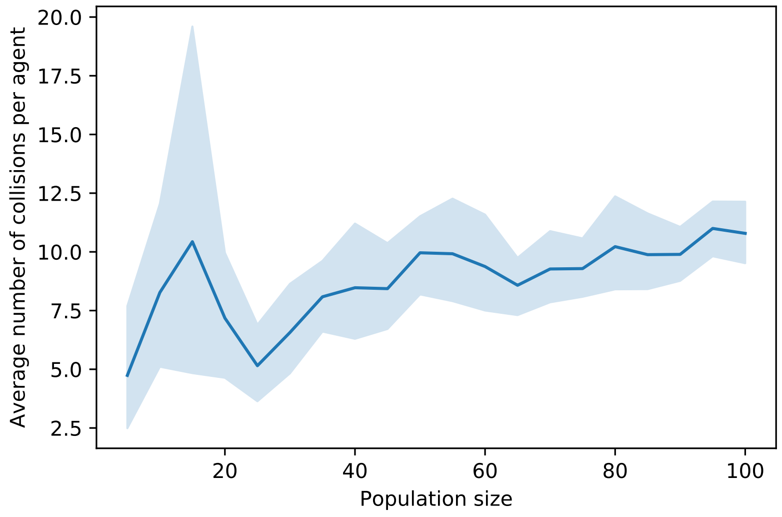
Figure 6.2: Variation in average error per agent with model time for different population sizes.

Each of the time-series follows a similar trajectory, with the initial error per agent being very low (near 0), but rising sharply as agents start to enter the environment through their respective entrance gates. For each of the population sizes, this error peaks at approximately 40 within the first 500 time-steps before declining and approaching 0 as the ensemble of benchmarking models approach completion. This common peak in error is likely a consequence of the activation rate parameter of the model — this parameter controls the rate at which agents enter the environment. This means that there is an upper limit to how many agents may attempt to enter the environment in any given time-step. In order to contextualise the scale of the average errors seen for each population size, we should consider two sets of measurements against which to compare. The first of these is the scale of the environment — the height and width of the environment can be found in Table 6.1. In comparison to the scale of the environment, the errors found in this chapter are much large than those found in the previous chapter; in this chapter, the peak of the benchmark errors is approximately $40/700 \approx 5.7\%$ of the scale of the environment, whilst in previous chapter the peak of the benchmark errors was approximately $1/100 \approx 1\%$. When considering the scale of the sidestepping motion in this model — 7 in comparison to 1 in the previous chapter — we observe that the peak error has grown disproportionately. Within the model, error is largely attributed to two factors — variations in the location along a gate at which an agent enters and agent-agent interactions. At lower agent population sizes, the latter is likely to contribute less towards the error with fewer interactions occurring (as illustrated in Figure 6.3a). We may attribute the increased peak of the average error to the introduction of a greater degree of uncertainty in origin and destination in comparison to those in the previous chapter, and to the more diverse types of interactions (including interactions between agents and environmental obstacles) which may result in an increase in the level of the average error.

When considering the number of agent-agent collisions that occur based on the population size in Figure 6.3a, we observe that the number of collisions increases as the number of agents in the system increases. We may, therefore, expect that the error contributed by inter-agent collisions would increase as the population size increases. When calculating errors, however, we perform calculations on a per-agent basis. It is, therefore, more useful to consider the average number of collisions per agent instead; when we consider Figure 6.3b, we see that increases in population size lead to only a small increase in the average number of collisions per agent as the population size increases. This is, once again, likely a consequence of the activation rate which means that only a certain number of pedestrians are present in the environment at any given time. This means that, when consider the average error per agent, the error contribution of agent-agent collisions does not increase much as the population size increases.



(a) Number of collisions.



(b) Number of collisions normalised by population size.

Figure 6.3: Variation in number of collisions over course of filter run with population size.

Whilst the trajectories followed by each of these time-series may be similar, the time at which each of the ensembles reaches completion changes. As the population size increases, the time taken to reach completion also increases. This is also a consequence of the constant activation rate which means that the rate at which pedestrians are introduced into the environment is fixed.

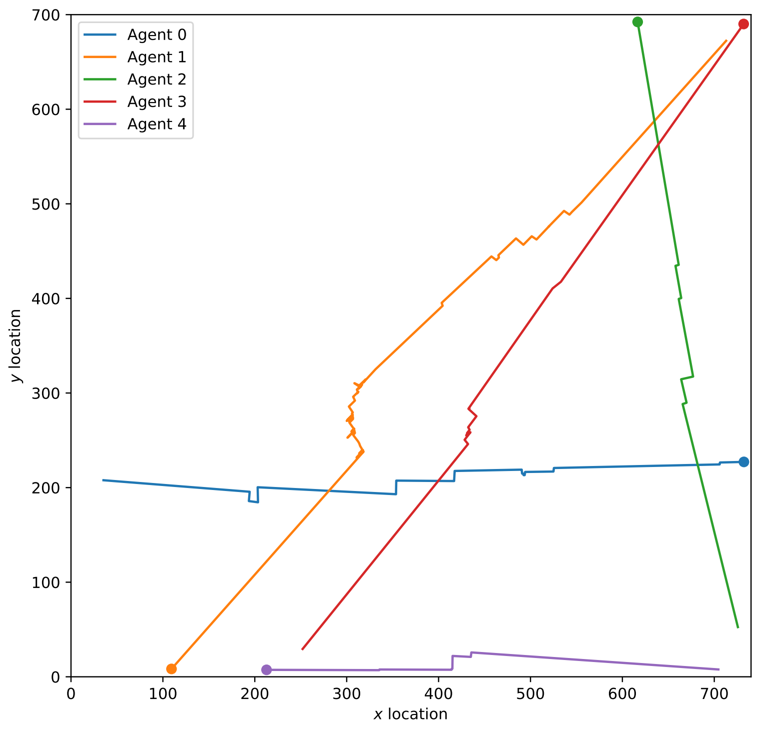


Figure 6.4: Trajectories of agents passing traversing the environment. Contains a sample of 5 agents from a simulation containing 100 agents. Points indicating starting location for each agent.

Considering Figure 6.4, we can see the way in which agents engage in the side-stepping behaviour in this new environment. As in the previous chapter, we see the pedestrians engaging in the side-stepping behaviour to avoid other agents (e.g. the trajectory of Agent 2). Comparing this to Figure 5.3 in the previous chapter, we find the inclusion of a new behaviour: side-stepping to avoid the central information desk obstacle (see Agent 1). This results in a much larger deviation from the direct path between an agent's entrance and exit points.

Having established a benchmark for the average error per agent resulting from running an ensemble of models without the assistance of data assimilation, the next experiment goes on to explore how error varies within an ensemble of models when data is assimilated, and compares this against a benchmarking ensemble of the same size.

6.2.2 Exploring Ensemble Member Models

Having run the experiment outlined in Section 6.1.2, the results have been plotted in Figures 6.6 and 6.7. This experiment consists of running a filter consisting of an ensemble of models and a benchmarking ensemble of models, allowing us to compare the performance of the filter ensemble against the benchmarking ensemble, as well as comparing the distribution of errors between the different filter ensemble models.

Prior to exploring the results displayed in each of these figures, we can discuss the results of the benchmarking ensemble, and how this compares to the variation in analysis error across the ensemble. As in the case of the results of the previous experiment, it is seen in Figure 6.5 that the benchmarking error is large, particularly when considered in comparison to the average error per agent calculated from the filter ensemble mean state. Similarly to the previous experiment, the benchmarking error is high at beginning of the experiment, declining over the course of the filter run. When comparing the benchmarking error in this experiment to the benchmarking error in the previous experiment, however, it can be seen that the benchmarking error in the experiment oscillates much more as it declines towards 0 at the end of the filter run. This is a result of the comparatively smaller ensemble size used for the benchmarking ensemble in this experiment — in this experiment, an ensemble of 20 models was used for the benchmarking ensemble whereas an ensemble of 100 models was used for the benchmarking ensemble in the previous experiment. Furthermore, as in the previous chapter, we may observe the impact of different frequencies with which data has been sampled; in the previous experiment data is sampled at every time-step and as such we have a relatively smooth line, whereas in this experiment the data are sampled at every assimilation time-step and so we find noticeable jumps between sequential data points. When comparing the average error in the benchmarking ensemble against the average error in the filter ensemble mean state, the filter ensemble mean shows a much

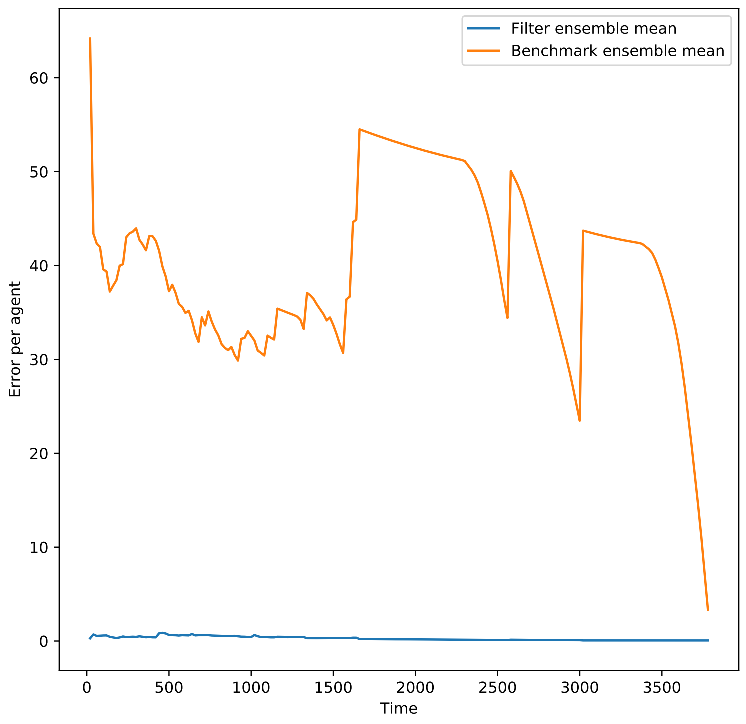


Figure 6.5: Line plot of the average error per agent based on the mean state of the benchmarking ensemble and the mean state of the filter ensemble.

lower error; consequently, the benchmarking error will be omitted in subsequent figures for this experiment.

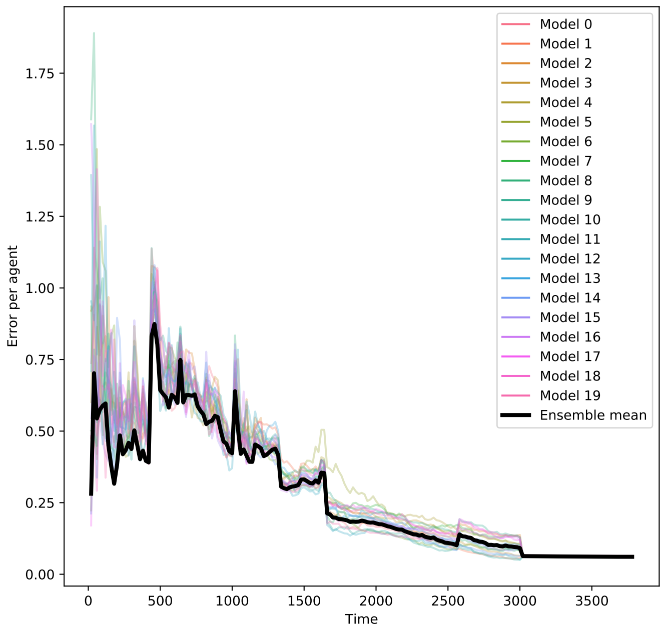


Figure 6.6: Line plot of average error per agent based on each ensemble member model.

After having explored the way in which the average error of the benchmarking ensemble compares against the average error per agent in the filter ensemble mean state, we can explore how the average error per agent varies across the filter ensemble member models and how this compares to the ensemble mean state; this is shown in Figures 6.6 and 6.7.

In Figure 6.6, the average error per agent is plotted for each individual ensemble member model as well the average error per agent for the ensemble mean state (plotted in bold black). In this figure, we can see that the variations in error in the individual models largely mirror those seen in the ensemble mean state error. The error in the ensemble mean state, however, appears to typically be lower than the errors in the majority of the individual models. This is further supported by Figure 6.7 which also compares the error in the ensemble mean state against the error in the individual

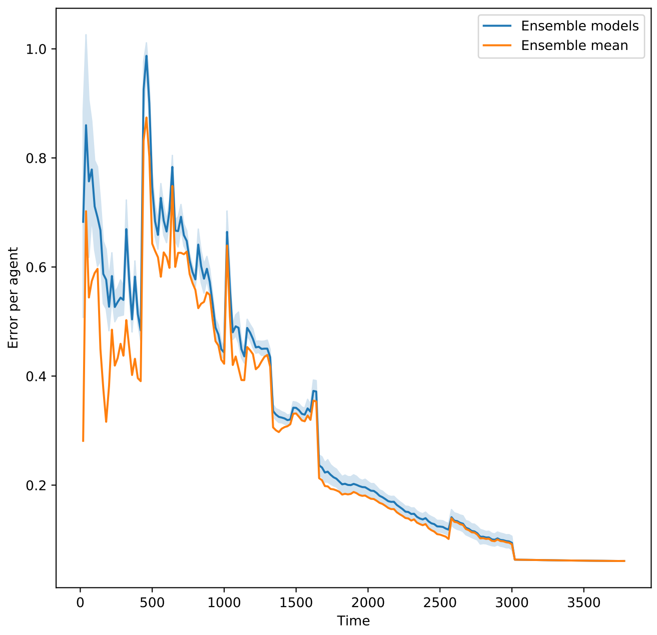


Figure 6.7: Line plot of average error per agent based on each ensemble member model, with confidence intervals.

models; in this case, the individual models are summarised as the mean of the model errors and the 95% confidence interval around it. Whilst we may have expected that these two sets of errors would match each other, this is not the case, with the error in the ensemble mean state consistently being lower than the mean error of the ensemble member models.

In order to explain why this is the case, we may consider an example scenario outlined in Figure 6.8. In this simple example, we consider a scenario in which we have a single agent being modelled by an ensemble of 3 models. The ground truth location of the agent (which would be provided by the base model) is given to be $(25, 25)$. Each of the ensemble member models have their own respective estimations of the agent's location:

- Location in model A: $(24, 26)$
- Location in model B: $(23, 25)$
- Location in model C: $(24, 24)$

From these estimates of the agent location, we can calculate the mean estimate of the ensemble: $(23\frac{2}{3}, 25)$. When comparing the mean estimate of the ensemble and the ground truth, we can calculate the error in the ensemble mean:

$$\sqrt{(x-\hat{x})^2 + (y-\hat{y})^2} = 1\frac{1}{3}.$$

In a similar manner, we can calculate the error in the estimates from each of the ensemble member models:

$$\begin{aligned} A &= \sqrt{(x-\hat{x})^2 + (y-\hat{y})^2} &&= \sqrt{2} \\ B &= \sqrt{(x-\hat{x})^2 + (y-\hat{y})^2} &&= 2 \\ C &= \sqrt{(x-\hat{x})^2 + (y-\hat{y})^2} &&= \sqrt{2} \end{aligned}$$

Based on these individual errors, we can calculate the mean of the errors in the estimates from ensemble member models, μ:

$$\begin{aligned} \mu &= \frac{A+B+C}{3} \\ &= \frac{\sqrt{2}+2\sqrt{2})}{3} \\ &= \frac{2}{3}(\sqrt{2}+1) \end{aligned}$$

When comparing the error in the ensemble mean state and the mean of the ensemble member model errors, we can see that error in the ensemble mean state is lower than the mean of the errors in the ensemble member model estimates of the agent's location (just as in Figure 6.7).

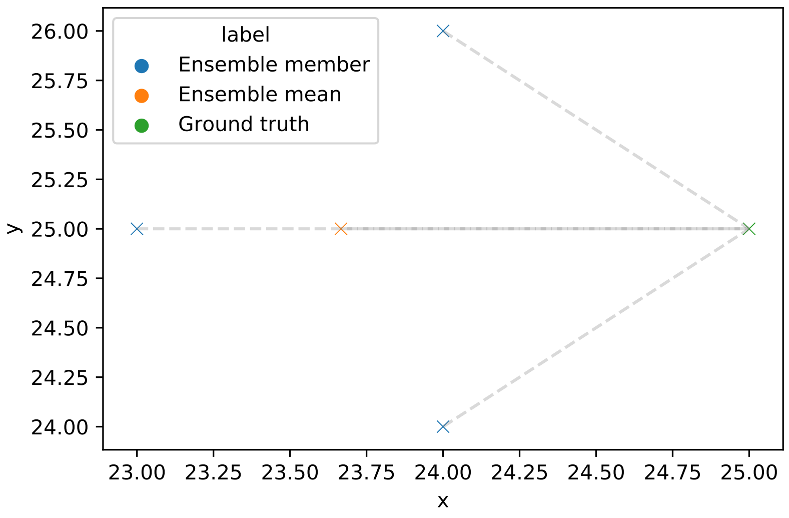


Figure 6.8: Working example — calculating error based on the ensemble mean vs. based on taking the mean of the individual member models.

6.2.3 Implementing the Ensemble Kalman Filter

Having established a model baseline level of error and undertaken some exploration of the variation in error across the ensemble-member models within an EnKF, this section seeks to implement the EnKF and explore some of the challenges associated with this task. Before tackling these challenges, some attention is paid to the what is happening to the individual agents, and their representations across the ensemble. In Figure 6.9, we see the ensemble-member model representations of two individual agents (agent A and agent B); this figure provides us with both the prior and posterior distributions of the locations for each agent. The prior distributions for agents A and B are shown by sections A and C respectively, and the posterior distributions are shown by sections B and D respectively. When comparing the prior and posterior for both agents, we see that the introduction of observations through data assimilation has resulted in the reduction in the spread of the ensemble-member model representations of the agents, i.e. the uncertainty in the model estimate of the positions has been reduced. This pattern is observed across the other agents in the system.

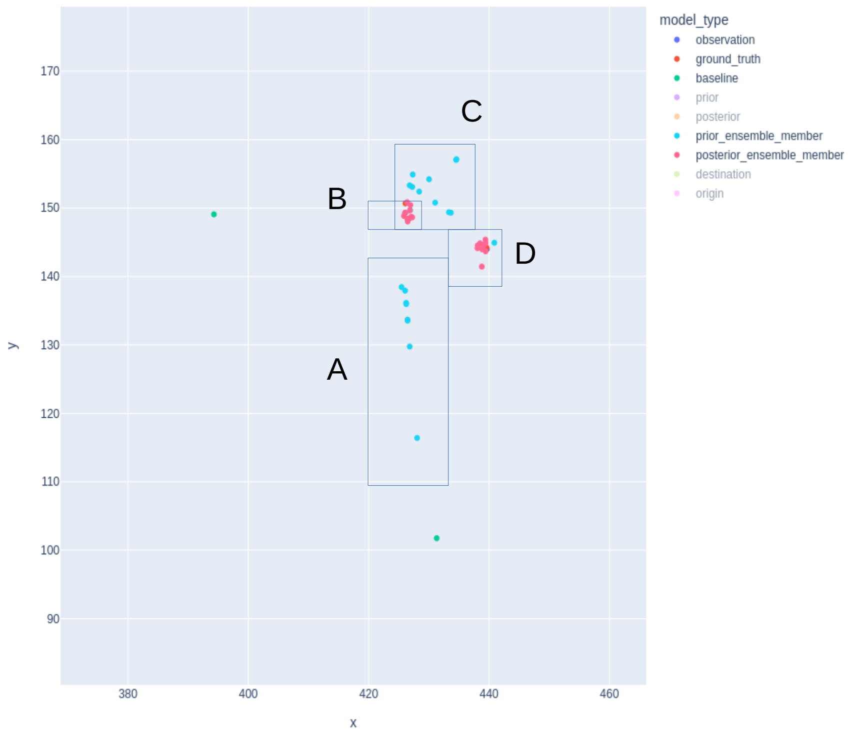


Figure 6.9: Comparison of prior and posterior positions of two agents (A and B). Section A: the ensemble-member model representations of the prior position of agent A in blue. Section B: the ensemble-member model representations of the posterior position of agent B in pink. Section C: the ensemble-member model representations of the prior position of agent B in blue. Section D: the ensemble-member model representations of the posterior position of agent B in blue.

Having established the ability of the EnKF to reduced the uncertainty in our estimates of pedestrians' positions in the environment, we see to tackle a number of challenges.

Managing Outliers

When running a large number of filters with the same filter and model parameters, there is some variation in the results; this variation pertains to both the way in which the average error per agent varies over time, and how long a filter takes to reach completion.

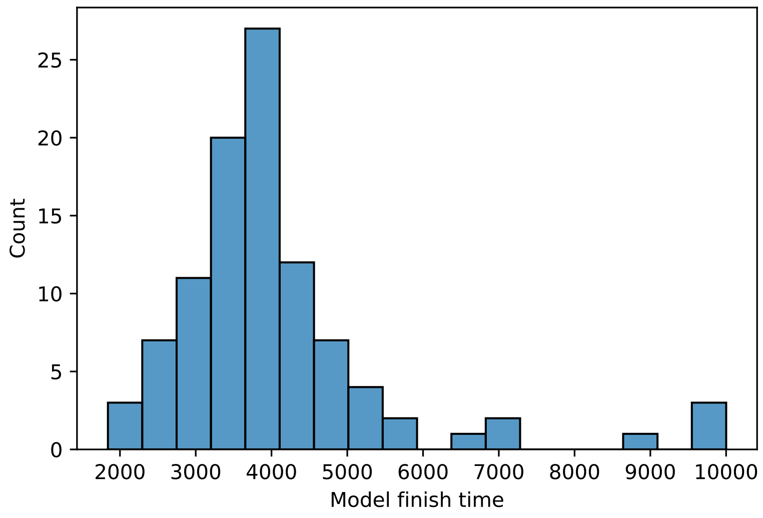


Figure 6.10: Histogram of filter finish times.

When visualising results for these collections of filter runs in aggregate form, care must be taken — filters which take longer to reach completion are often those that exhibit higher levels of average error per agent and consequently we may observe that, towards latter time-steps, the aggregated picture of average error per agent shows the error rising as a result of the filters in which the error is low are reaching completion, and consequently no longer contributing to the aggregated results. This leaves us with a scenario in which average error per agent appears to be rising as time proceeds. This can be seen when considering Figures 6.10 and 6.11. Figure 6.10 shows a histogram of the finishing times of the filters in this experiment; this shows that whilst the majority of the filters reach completion within the first 4500 time-steps, some take longer, with some not completing until approximately 10000 time-steps. This is corroborated by Figure 6.11 which shows an empirical cumulative distribution function plot of the filter finish times. This highlights that 90% of the filters reach completion within 6000 time-steps.

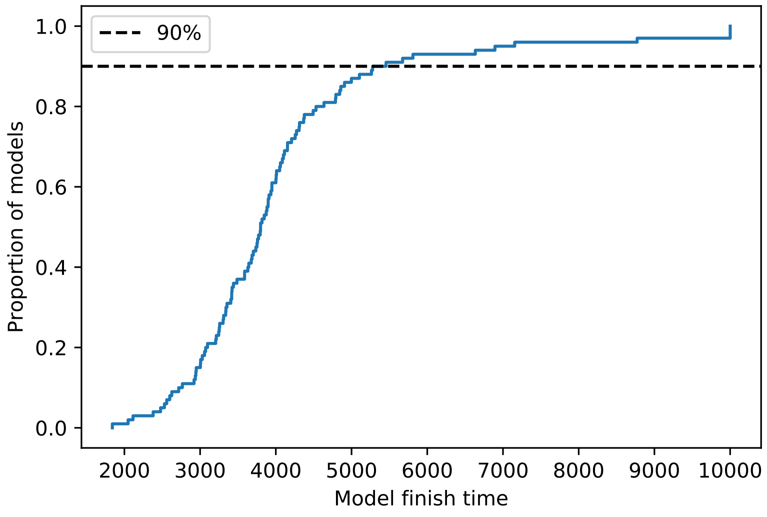


Figure 6.11: Empirical cumulative distribution function (eCDF) plot of filter finish times; dotted line represents a cumulative level of 90%.

In this experiment, this issue is handled by only considering time-steps in which more than 10% of the filters are still running. The impact of this measure is observable when comparing the top row and bottom row of Figure 6.12. The top row shows the average analysis error per agent when considering data from all time-steps. The bottom row shows the average analysis error agent when considering only data from time-steps in which more than 10% of the filters are still running.

When considering Figure 6.13, we can see the impact of the minority of filters which have much larger average error per agent; Figure 6.13 shows a boxplot of the average analysis error per agent for all time-steps in the truncated version of the data outlined above. This figure also contains a dashed line representing the mean of the data. This figures shows that the majority of the data points lie below 1.5. The mean value, however, lies around 2.0 — values in this range and above are considered outliers in the figure given their position beyond the whiskers of the boxplot. This suggests that the mean may not be an appropriate summary statistic to plot given that is is influenced by outliers. An alternative summary statistic may be the median.

The difference between the use of the mean and median as an estimator for the truncated data may be observed when considering the bottom row Figure 6.12. This

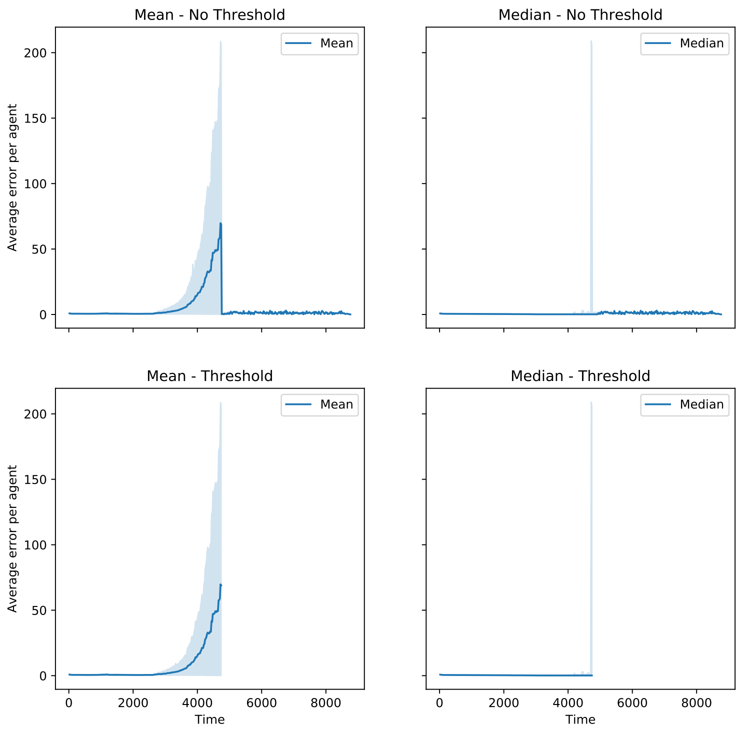


Figure 6.12: Comparison of impact of measures for handling outliers; the left-hand column uses a mean estimator for the line whilst the right-hand column uses a median estimator; the top row contains data for all run times whilst the bottom row restricts the data to run times for which greater than 10% of the filters are still running.

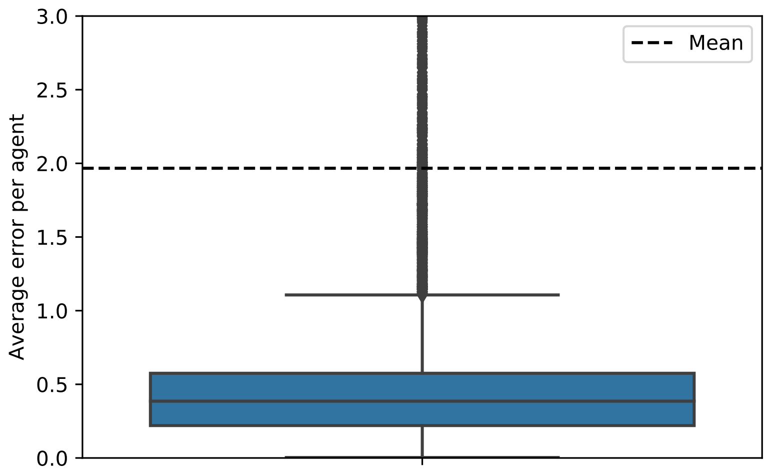


Figure 6.13: Box plot of average analysis errors for all filters for all time-steps in the truncated data; dotted line represents the mean of the data

highlights the extent to which the mean estimator is skewed by poorly performing filters, particularly as many of the better-performing filers reach completion.

The conclusion here is that it is appropriate to truncate our results as outlined above, and to use the median as a summary statistic instead of the mean. The result of applying these changes can be seen in Figure 6.14.

Filter Performance

Having established the issues that may arise when handling outputs from this experiment, we may implement the changes to the analysis process outline in the previous section, and explore the results. When exploring these results regarding the average error per agent, there are three comparisons to be drawn: a comparison between the benchmarking ensembles and the analysis of the filter ensembles, a comparison between the forecast of the filter ensembles and the analysis of the filter ensembles, and a comparison of the observations and the analysis of the filter ensembles. This section goes on to explore each of these comparison. In each case, the comparison is aided by the use of three figures: a line plot of the average error per agent over time where the line represents the median of the collection of filters at each time-step and the shaded area

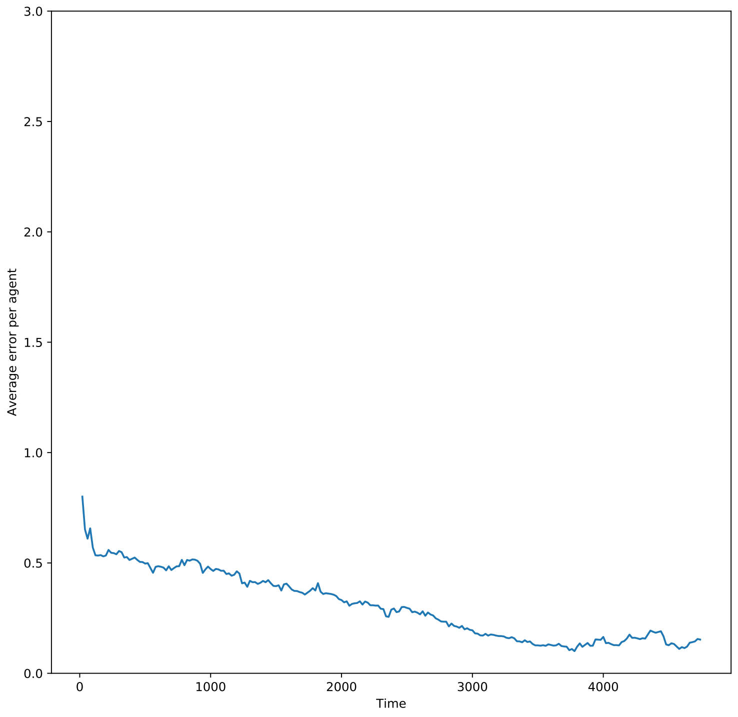
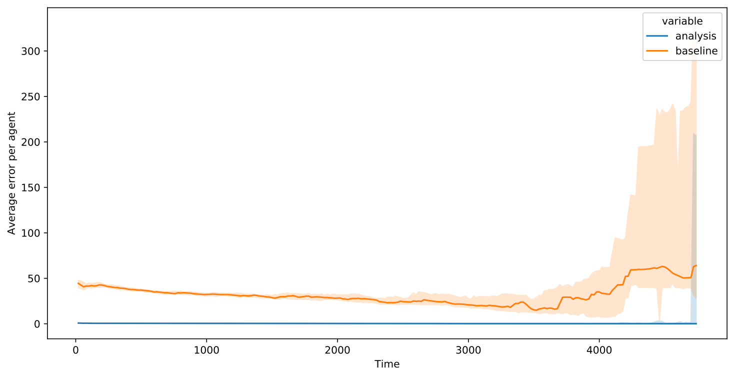
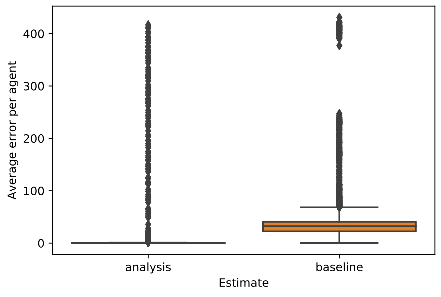


Figure 6.14: Line plot of average analysis error per agent for 100 filters; solid line represents the median of active filters at each time-step and shaded region represents the 95% confidence interval around this median.

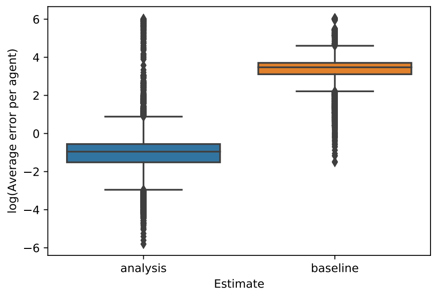
represents the 95% confidence interval around the line, a boxplot showing the distribution of these errors when aggregated over time, and a boxplot showing the log of these values to highlight the differences in scale.



(a) Line plot of average error per agent over time.



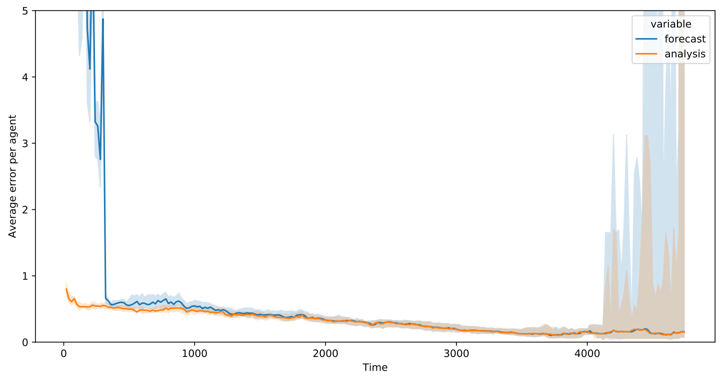
(b) Box plot of average error per agent.



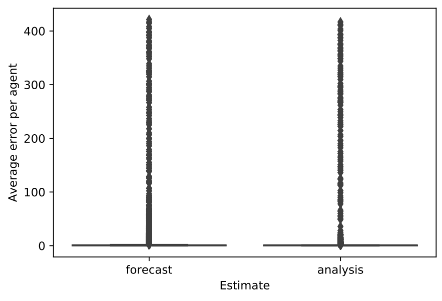
(c) Box plot of log of average error per agent.

Figure 6.15: Comparison of average error per agent between analysis and benchmarking filters

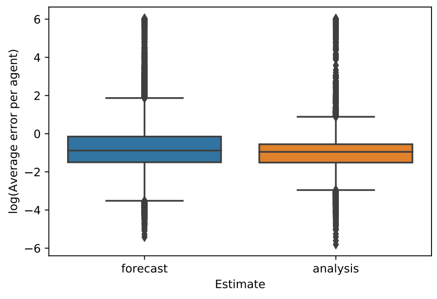
When considering Figure 6.15, we can see the comparison of the average error per agent observed in the collection of benchmarking model ensembles and observed in the analysis state of the filter model ensembles. Figure 6.15a displays the comparison over time as a line plot; in this figure, we can see that the average error per agent in the filter analysis states is much lower over the course of the time shown. This is similarly shown in the boxplot seen in Figure 6.15b; gaining information from this boxplot, however, is somewhat difficult as the whiskers and box in the case of both the analysis error and benchmarking error are dwarfed by the presence of outliers that lie beyond the upper whiskers of each boxplot. This is indicative of the large range in the errors observed in each case. In order to handle this issue, Figure 6.15c contains a similar boxplot, but with the log of the average error per agent in each case. The majority of the data pertaining to the analysis error lies below 0, indicating that the average error per agent for the analysis is often below 1. In the case of the benchmarking data, however, the majority of the data lie above 0, indicating that in most cases, the average error per agent for the benchmarking ensembles is much higher.



(a) Line plot of average error per agent over time.



(b) Box plot of average error per agent.

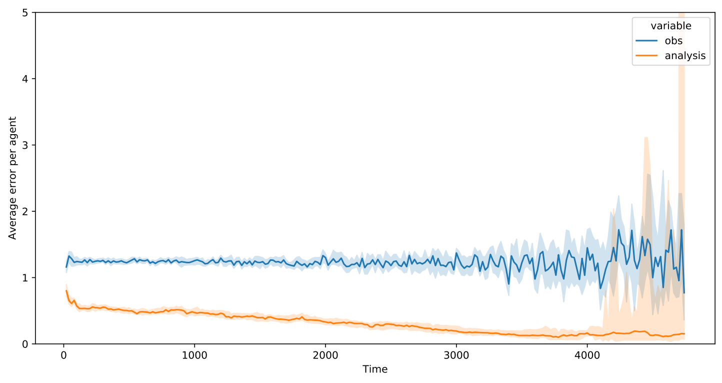


(c) Box plot of log of average error per agent.

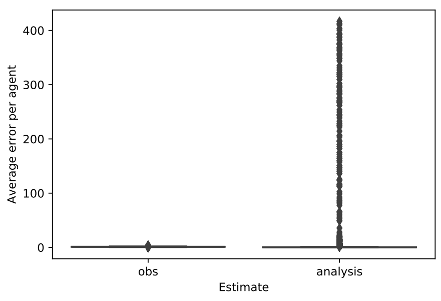
Figure 6.16: Comparison of average error per agent between analysis and forecast.

When considering Figure 6.16, we can see the comparison of the average error per agent observed in the forecast and analysis states of the filter model ensembles, i.e. the error before and after assimilating data at each time-step. Figure 6.16a shows the comparison of these two error over time as a line plot. In this figure we can see that the error in the analysis state is typically an improvement on the error in the forecast state, with the improvements being most noticeable at the beginning of the set of time-steps and at the end of the time-steps. The difference at the beginning is due to the reasoning outlined in Section 6.2.1 — the entrance of multiple agents at the beginning of the filter run time and the entry of agents at points on the gates that do not match the entry point of corresponding agents in the base model lead to an initial growth in error. The error in the analysis state does not suffer from this growth as it is updated by the provided observations. This improvement is not as marked as time proceeds past 1000. This is reflected in Figure 6.16c which shows a boxplot of the log of the analysis and forecast errors aggregated over time. In each case we find that the majority of the data lie below 0, once again indicating that the average error per agent in each case often lies below 1.

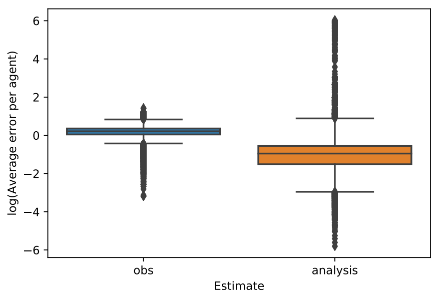
When considering Figure 6.17, we can compare the variation of the average error per agent of the collection of observations of the agents and the analysis states of the filter model ensembles. The way in which the two vary over time can be seen in Figure 6.17a. This figure shows that the average observation error per agent over the collection of filters is largely constant over the set of time-steps. As we near the end of the filter runs, there in increase in the variance of this error, with the confidence intervals broadening. This is a consequence of filters reaching completion, and this the summary statistics being drawn over a decreasing number of filters. A similar phenomenon is observed when considering the line for the analysis error. In comparison to the observation error, the analysis error is typically lower for the majority of the time-steps for which the filters are running. This is not always the case, however, as highlighted in Figure 6.17b. This figure contains a boxplot of the errors for each dataset, aggregating over time. In each case, the errors appear to be low; however, the boxes and whiskers for each dataset are dwarfed by a set a outliers pertaining to the analysis error. To make the comparison of the two datasets clearer, they are plotted on a log scale in Figure 6.17c. This figure shows that whilst the analysis error is typically lower than the analysis error, the dataset pertaining to the analysis error contains a number of outliers. The data pertaining to the observation error, on the other hand, contains relatively few outliers. This is a consequence of how the observations are produced; given that observations are produced by adding normally distributed random noise to the base model state in



(a) Line plot of average error per agent over time.



(b) Box plot of average error per agent.



(c) Box plot of log of average error per agent.

Figure 6.17: Comparison of average error per agent between analysis and observations.

each case, we expect that the observation error should not vary very much.

6.3 Concluding Remarks

This chapter has focussed on the application of the EnKF to a more realistic agent-based model — `StationSim_GCS`. In applying the filter to this model, a series of experiments were undertaken.

The first of these aimed to develop a benchmark for the performance of an ensemble of models in the absence of any data assimilation, measuring performance by comparing against a pseudo-ground-truth state drawn from a single instance of the model referred to as the "base model". This experiment found that in the absence of any data being provided, the ensemble of models did not perform very well, quickly diverging from the ground truth state. This phenomenon of divergence was seen to be largely independent of the population size of the model. The extent of the divergence was seen to be particularly bad when considering the early stages of a model run in which many agents are entering the system simultaneously resulting in an increased number of agent-agent interactions. This, partnered with the variation in entry locations along a gate amongst ensemble member models for a given agent meant that the benchmarking ensemble performed poorly from the outset of the model run. Furthermore, the number of interactions occurring in the system was not seen to vary much when scaling for population size.

The next experiment compared the performance of a similar benchmarking ensemble (which received no data) with the performance of an EnKF which received observations on as periodic basis. As in the case of the previous experiment, this experiment measured performance by considering the error between the ground truth base model and each of the benchmarking ensemble and the analysis state of the filter ensemble. This showed that the ensemble mean fared much better in simulating the state of the base model than the mean of the benchmarking ensemble by virtue of receiving periodic updates in the form of observations. The performance of the EnKF was further explored by considering how error varied within the ensemble of models over time. It was found that, whilst there was some variation between the ensemble-member models, they typically followed a similar trajectory over time as the error in the ensemble mean. The error in the ensemble mean was found to typically be lower than the mean of the errors in the ensemble member models. In previous investigations, little work has been done regarding the extent to which the ensemble mean error is representative of what is happening within the ensemble.

The final experiment was based on running a collection of experiments for a given set of model and filter parameters, and aggregating the results. This allowed us to explore the average behaviour of the filter. When running this experiment, we found that there is some variation amongst the different filters in terms of the range of values exhibited for average error per agent, and the times at which filters reach completion. In some cases, these variations produced outliers, potentially distorting the analysis process. This raised the issue of how to handle such variation. The following analysis decisions were, therefore, made:

- When producing line plots to summarise the evolution of error over time, the median of the errors across the filters at each time-step was used instead of the mean;
- Data pertaining to time-steps in which fewer than 10% of the filters were running were removed from the analysis.

The first of these decisions — the use of the median when considering errors over multiple filter runs — is one that has been mirrored by recent investigations in the literature (Malleson et al., 2020; Clay et al., 2020, 2021). The second decision, however, is not one that has been touched upon explicitly in other investigations and, as shown in this chapter, can have noticeable effects with regards to the outcome of experiments.

Based on these decisions, a comparison was drawn between the analysis error exhibited by the collection of filters and the benchmarking error, the forecast error and the observation error. As was the case in the previous experiment, the filter analysis states were found to outperform their corresponding benchmarking ensembles in simulating the states of their corresponding base models. This was also found to be the case when comparing against the forecast errors and the observation errors.

Up until now, we have assumed that we have knowledge of the gates to which each agent is heading and so have made each of the ensemble member models (as well as the benchmarking ensemble member models) copies of the base model. When simulating real-world systems, this is not realistic. In the next chapter, we shall remove this assumption, exploring how the benchmarking ensemble and filter ensemble perform, and proposing some solutions to the problems that arise from this scenario.

Chapter 7

Data Assimilation for Exit Estimation

In the previous chapter, we explored the use of the Ensemble Kalman Filter (EnKF) when applied to an Agent-Based Model of pedestrian motion around the concourse at Grand Central Station in New York. This showed that the filter was effective in improving the accuracy with which an ensemble of models could simulate the trajectories of pedestrians moving across the environment. This set of experiments (as well as those undertaken in Chapter 5) suffer from a key limitation: they make the assumption that we know the final destination of each pedestrian at the outset of the modelling process. In real-world applications, this assumption is not realistic — whilst we may know the location at which a pedestrian enters the system, there are a number of other attributes which we are likely not to know. Some of these unknown attributes, such as a pedestrian's scalar speed, may be observed as the pedestrian crosses the environment; others, such as the pedestrian's final destination, may not be directly observed. A more realistic simulation of the scenario should, therefore, include two further elements beyond those included in the previous chapter: a way of encoding some degree of uncertainty in these unobserved parameters at the outset, and some mechanism by which we can derive these unobserved attributes over the course of the filter run. With this in mind, this chapter will focus on ways in which uncertainty in agents' final destinations in the environment are represented in the EnKF and mechanisms by which we can correctly estimate agents' final destinations within the EnKF as we assimilate observations of the agents' locations. This will be approached by first describing the proposed solutions to each of these problems in Section 7.1; this will be followed by an outline of the set of experiments to be employed with these solutions; finally, the results of these experiments will be presented.

7.1 Proposed Solutions

As outlined above, this chapter focuses on two problems. The first of these is the development of a method within the data assimilation framework to infer the value of agent parameters that are neither known at the outset of a simulation run, nor directly observed over the course of the run. Although such parameters are unknown at the outset of a simulation run, the initialisation of models requires that a value be assigned to them. The second problem is, therefore, one of developing a method for the initial allocation of these parameters such that it reflects a lack of knowledge, and exploring the ramifications of the decisions made in developing such a method. Whilst there may be a number of different latent variables pertaining to pedestrian agents, this chapter will focus the inference of pedestrian destinations within the system. Beyond the initial problem of inferring latent variables, this also presents us with the challenge of estimating categorical variables — a problem to which data assimilation schemes are not typically applied.

Consequently, this section focuses on outlining the proposed solutions to two issues:

- How do we encode uncertainty regarding the final destination of agents within the EnKF?
- How do we correctly estimate the final destination of agents within the EnKF given unknown values for this parameter at the outset of a simulation run, and given periodically assimilated observations of the agents' locations over the simulation time?

In the following subsections, we propose solutions to each of these problems. When considering the problem of encoding uncertainty in agents' final destination, we propose that different encodings are represented through different distributions across the ensemble of models; this is expanded upon in Section 7.1.1. When considering the problem of correctly estimating agents' final destinations with the EnKF, an approach known as state augmentation is used (Katzfuss et al., 2016), whereby the unknown parameter values are included in the state vector pertaining to each model; this is expanded upon in Section 7.1.2.

7.1.1 Destination Uncertainty

When considering an agent at the outset of a simulation run, we may assume that its initial location, i.e. the gate through which it enters the environment, is known. There is some uncertainty in an agent's initial location, which is encoded in the way in which a model allocates agents' initial locations; upon initialisation, agents are allocated a gate through which to enter the environment, and subsequently are allocated a starting location along the gate, with location along the gate being drawn from a uniform random distribution.

At the outset of a simulation run, we may not, however, assume quite as much knowledge about an agent's final destination. Upon initialisation, each agent in each ensemble-member model is allocated a destination gate. Previously (in Chapters 5 and 6), when initialising a simulation, we had allocated the correct target destination to each agent in each model of the ensemble which represented perfect knowledge of the final destination of each agent. In this Chapter, however, we relax this assumption with a view to exploring the implications of different initial distributions for final destinations for agents. We therefore need to make a decision regarding how we initially allocate destinations to agents within the model; the way in which destinations are allocated to agents will have implications regarding the uncertainty in our knowledge of the destinations, which in turn will have implications regarding how the EnKF treats the prior state when updating. This Section outlines three different distributions that may be considered:

- Uniform across ensemble
- Random across ensemble
- Adjacent gates

These distributions are described below.

Uniform Across Ensemble

Under this regime, the agents across the ensemble pertaining to a specific pedestrian are all allocated the same destination gate, i.e. the first agents in each of the models are allocated the same destination gate. The destination gate allocated across the ensemble to the agents representing a specific pedestrian is drawn from a uniform distribution of the gates which excludes the gates on the side of the environment through the pedestrian entered.

As an example, consider a pedestrian who entered the environment through gate 2 on the top side of the environment; the initial destination for the pedestrian would then be drawn from a uniform random distribution across the gates on each of the boundaries excluding the top boundary, i.e. gate 0 from the left boundary, gates 3-6 on the right boundary and gates 7-10 on the bottom boundary. If gate 7 were drawn from the distribution, this would be allocated to the agents in each of the ensemble-member models pertaining to the pedestrian in question. This is expanded to entrance gates from each side of the environment in Table 7.1.

Whilst such a distribution may correctly incorporate a complete lack of knowledge regarding each agent's target destination, we expect that it would suffer from a key shortcoming. Given that the allocation of destination gates to the agents pertaining to a specific pedestrian is uniform, i.e. the value is the same across the ensemble. The issues with this are both conceptual and functional. From a conceptual standpoint, such a distribution of values across the ensemble would indicate that there is no uncertainty in the destination gate — this is not an accurate representation of our knowledge of an agent's target destination, which is likely highly uncertain at the outset of a simulation run. From a functional standpoint, this will impact how the EnKF updates the target destination estimate in the state vector. As outlined in Chapter 4, the ensemble updates values in the state vector based on the uncertainties in the ensemble and the observations; given a scenario in which there was no uncertainty in a state vector value, we would expect that the posterior state vector would not differ from the prior state vector in any substantial manner, indicating that little weight was placed on the observations being assimilated.

Entrance Side	Possible Entrance Gates	Possible Exit Sides	Possible Exit Gates
L	0	R, T, B	1 - 10
T	1 - 2	L, R, B	0, 3 - 10
R	3 - 6	L, T, B	0 - 2, 7 - 10
B	7 - 10	L, T, R	0 - 6

Table 7.1: Table of possible entrance and exit gates given a specified entrance side. In this table, L refers to the left side of the environment boundary, T to the top side, R to the right side and B to the bottom side. All ranges are inclusive of both upper and lower bounds.

Random Across Ensemble

If we once again assume a complete lack of knowledge regarding a pedestrian's target destination, we may consider another way of allocating destination gates to agents across the ensemble-member models. In this approach, we once again consider a random uniform distribution of destination gates across the gates around the environment excluding the gates on the side through which a pedestrian enters (as detailed in Table 7.1).

Let us consider, as an example, a scenario in which we are simulating a system containing 1 pedestrian with a EnKF with and ensemble size of 5, i.e. there are 5 ensemble-member models each containing 1 agent. When we initialise the simulation, we may claim to have knowledge of the gate through which the pedestrian enters the system, but not the gate through which they will exit the system. Let us imagine that, in this case, the pedestrian enters the environment through gate 1 on the top side of the environment. This would mean that the agent in each of the 5 ensemble-member models would be created such that they would also enter through gate 1; at this stage, however, they could be allocated any of the exit gates detailed in the second row of Table 7.1. In order to allocate an exit gate to the agent representing the pedestrian in each of the ensemble-member models, we would draw from the random uniform distribution across the possible exit gates outlined in the table. This might result in an vector of exit gates for the agent across the ensemble of

$$[0, 4, 7, 10, 3].$$

Such an approach would improve upon the Uniform Across Ensemble approach outlined above in both a conceptual and functional manner. From a conceptual standpoint, this more accurately represents our initial knowledge regarding the pedestrian's final destination — as outlined in Chapter 3, our model assumes that pedestrians will not exit through gates on the side of the environment through which they have entered, but beyond this we have no prior knowledge regarding which of the remaining gates a pedestrian will exit. From a functional standpoint, this encoding of uncertainty in a pedestrian's target destination would allow the EnKF to update the estimate of the target destination over the course of the simulation. It is likely, however, that a larger ensemble size than that provided in the example above would be beneficial in practical applications as such a small ensemble size would not be able to adequately capture the full range of different discrete values the target destination could take on.

Adjacent Gates

Above, we have considered scenarios in which we have no prior knowledge regarding an agent's target destination at the outset of the simulation. There may, however, be cases in which we have some prior knowledge regarding a pedestrian's target destination, but this knowledge may not be as perfect as represented in Chapter 6, i.e. there is some non-zero degree of uncertainty in our knowledge regarding the pedestrian's target destination. We may gain such information through an analysis of trace data (such as that used to inform the calibration of the Grand Central Station model), which may result in some knowledge regarding the most likely gate for a pedestrian to exit through given its entrance gate. In such a scenario, we can once again choose to draw destination gates for agents from a distribution; in this case, however, we choose a distribution other than the uniform distribution over all gates from boundaries other than that through which the agent enters.

In this case, we choose to use a uniform distribution over the correct gate and a number of adjacent gates symmetrically distributed around the correct gate; the filter takes as a parameter the number of gates to consider in each direction in the uniform distribution. The aim of this approach is to use this knowledge of the correct gate as a proxy for some sort of knowledge regarding the most likely destination for an agent (along with some uncertainty in this knowledge) which may be obtained through some sort of empirical analysis. As an example, consider that the correct destination for a pedestrian is gate 5. In this case, if we wish to consider 1 adjacent gate on each side, agent destinations would be drawn from a uniform distribution over 4, 5 and 6. Similarly, if we wish to consider 2 adjacent gates on each side, agent destinations would be drawn from the uniform distribution over 3, 4, 5, 6 and 7.

Given that there are a finite number of gates, we must consider what happens when the distribution from which the gates are drawn spans beyond the limits of range of gates, i.e. what happens if the correct gate is gate 10 and we wish to draw from the distribution of 2 adjacent gates on either side? In such a scenario, we would not be able to allocate to gates 11 or 12 as the gates are numbered 0-10 and, therefore, gates 11 and 12 do not exist. To resolve this situation, we use modular arithmetic. Practically, this involves defining the upper and lower bounds of a uniform distribution from which we may drawn offsets to be applied to the correct exit gate. These bounds are defined by the number of adjacent gates on either side that we would like to consider; if we would like to consider 2 gates on either side then we would drawn an offset from the uniform distribution:

$$\mathcal{U}\{-2, 2\}.$$

For each agent representing the pedestrian, we would draw an offset from this distribution. We would then add it to the correct gate, and then take the modulus with respect to the total number of gates.

Consider a case in which we have a system containing 1 pedestrian being simulated by an EnKF with an ensemble size of 5, and with the number of gate to consider on either side of the correct gate defined as 2. In this case, suppose that the correct destination gate for the pedestrian is gate 10. Upon initialisation the agent representing this pedestrian in each of the ensemble-member models is allocated a destination gate as outlined in the procedure above. This would mean first defining the upper and lower limits of the distribution of offsets:

$$\mathcal{U}\{-2, 2\}.$$

Suppose that the offsets pertaining to the 5 agents across the ensemble were

$$[-1, 1, 2, 0, 0].$$

We would then add these offsets to the correct destination gate to obtain

$$[9, 11, 12, 10, 10].$$

Finally, we would take the modulus of each prospective destination gate with respect to the total number of gates (i.e. 11), resulting in

$$[9, 0, 1, 10, 10],$$

which are the destination gates to be allocated to the agents across the ensemble representing the pedestrian.

7.1.2 Destination Estimation

Having outlined ways of encoding the initial uncertainty in a pedestrian's target destination in the previous section, this section aims to outline methods that can be applied to the EnKF to update these estimates with the aim of accurately identifying the correct destination for each agent. One candidate to achieve this goal is the use of the EnKF to update models' states and their parameters simultaneously. The most common approach to this is known as state augmentation (Katzfuss et al., 2016). When applying the EnKF with state augmentation, both the model state (i.e. the agent locations) and the model parameters (i.e. the agent destinations) are included in the state vector to be updated by the data assimilation process. If we consider a scenario in which we are using the EnKF to estimated the observed states, $\mathbf{x}$, and the unobserved model attributes, θ, we must make some adjustments to matrices used in the filter; this section will briefly outline the adjustments made, but a more detailed version can be found in Zhang et al. (2017). A state vector pertaining to the ith model in the ensemble might look like

$$\mathbf{X}_i = [\mathbf{x}_i, \theta_i]$$

where $\mathbf{x}_i$ is the vector of observed states pertaining to the ith model and θ_i is the vector of unobserved model attributes pertaining to the ith model. With such a model state vector, we must then update our observation operator, $\mathbf{H}$:

$$\mathbf{H} = [\mathbf{H}_x, 0]$$

where $\mathbf{H}_x$ is the observation operator used in previous experiments as described in Chapter 4/

Based on this state augmentation approach, this section will now go on to outline two ways in which an pedestrian's target destination can be parameterised. The first of these approaches is to simply include the number of the gate to which the pedestrian is heading in the state vector, and have the Ensemble Kalman Filter update the gate number. This shall be referred to as Gate Number Estimation. When undertaking this approach, we note that gate numbers, whilst numeric, are in fact discrete categories which presents some challenges as the Ensemble Kalman Filter is not typically applied to such problems; we shall elaborate upon these challenges in the following section. The second approach is to define the pedestrian's target destination based on the angle between the destination location on the environment boundary and the centre of the environment. This shall be referred to as Destination Angle Estimation. This treats the destination estimation problem as something closer to the continuous variable estimation problems for which the EnKF is appropriate; we also encounter some challenges with this approach which shall be outlined in the following sections.

Gate Number Estimation

This section outlines the first proposed approach to estimating pedestrians' target destinations. This involves adding an extra element to the state vector for each agent which pertains to the number of the gate through the agent will exit. In the previous chapter, when considering the state vector for a model containing N agents, we would expect it to have the following form:

$$[x_1, y_1, x_2, y_2, \ldots, x_N, y_N],$$

where x_i refers to the ith agent's x-location and y_i refers to the ith agent's y-location. In this case, however, we are including an additional entry for pertaining to the gate number for each agent. In this case, we would expect the state vector to have the following form:

$$[x_1, y_1, g_1, \ldots, x_N, y_N, g_N], \tag{7.1}$$

where x_i refers to the ith agent's x-location, y_i refers to the ith agent's y-location, and g_i refers to the ith agent's gate number. For this investigation, however, the choice was made to formulate augmented state vectors in the following manner:

$$[x_1, \ldots, x_N, y_1, \ldots, y_N, g_1, \ldots, g_N]. \tag{7.2}$$

This choice was made so as to increase the ease with which an appropriate observation operator could be constructed in an automated manner. If we consider an EnKF with ensemble size 3 and population size 2, we would have a state ensemble that looked like

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ y_{11} & y_{12} & y_{13} \\ g_{11} & g_{12} & g_{13} \\ x_{21} & x_{22} & x_{23} \\ y_{21} & y_{22} & y_{23} \\ g_{21} & g_{22} & g_{23}, \end{bmatrix} \tag{7.3}$$

where x_{ij} represents the x-location of the ith agent in the jth ensemble-member model, y_{ij} represents the y-location of the ith agent in the jth ensemble-member model and g_{ij} represents the gate number of the ith agent in the jth ensemble-member model. The EnKF would be assimilating observations of the form:

$$[x_1, y_1, \ldots, x_N, y_N]. \tag{7.4}$$

In such a situation, we would use an observation operator of the following form

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{7.5}$$

In comparison, when using the state vector for in Equation 7.2, the state ensemble would take the form of

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \end{bmatrix}, \tag{7.6}$$

and consequently the observation operator would take the form of

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{7.7}$$

Considering a system of N pedestrians, Equation 7.7 generalises to the row-concatenation of two matrices: an identity matrix of size $2N$ and a matrix of 0s of shape $2N \times N$:

$$H_N = [\mathbf{I}_{2N}, \mathbf{0}_{2N \times N}] \tag{7.8}$$

Both of these matrices are trivial to produce computationally at the outset of the simulation run, given the population size N, and are easily joinable. The result is an observation operator of shape $2N \times 3N$.

Using these forms for the state ensemble and observation operator, we can then use the EnKF to update both the locations of agents and their target destinations. If we return to the state update equation for the EnKF in Chapter 4, we may recall that the filter produces a posterior ensemble of model states, $\hat{\mathbf{X}}$ based on the following equation:

$$\hat{\mathbf{X}} = \mathbf{X} + \mathbf{K}\left(\mathbf{D} - \mathbf{H}\mathbf{X}\right), \tag{7.9}$$

where $\mathbf{X}$ is the prior ensemble of model states, $\mathbf{K}$ is the Kalman Gain Matrix, $\mathbf{D}$ is the ensemble of observations and $\mathbf{H}$ is the observation operator outlined above. The observation operator that we have specified is designed to reflect a scenario in which we are able to observe the location aspects of the model states (i.e. the x and y positions), but are unable to observe the destination aspects (i.e. the g values). As we see above, the application of the observation operator to the ensemble of model states reduces the ensemble to just the observable parts of the state. Consequently, we are able to calculate the difference between the observations and the observable sections of the model states, $(\mathbf{D} - \mathbf{HX})$.

As detailed in Chapter 4, this difference is applied as a weighted perturbation to the prior model state where the weighting is defined by the Kalman Gain Matrix, $\mathbf{K}$. Recalling the definition from Chapter 4, the gain matrix is given as:

$$\mathbf{K} = \mathbf{CH}^T \left(\mathbf{HCH}^T + \mathbf{R}\right)^{-1}, \tag{7.10}$$

where $\mathbf{C}$ is the state covariance matrix and $\mathbf{R}$ is the observation covariance matrix. The state covariance matrix contains the covariances of each of the elements in the model state — both observable and unobservable. The observation covariance matrix contains the covariances of each of the elements in the observations, i.e. only of the observed quantities. Based on the example state ensemble above, we would have the following covariance matrix:

$$\mathbf{C} = \begin{bmatrix} \sigma_{x_1x_1} & \sigma_{x_1x_2} & \sigma_{x_1y_1} & \sigma_{x_1y_2} & \sigma_{x_1g_1} & \sigma_{x_1g_2} \\ \sigma_{x_2x_1} & \sigma_{x_2x_2} & \sigma_{x_2y_1} & \sigma_{x_2y_2} & \sigma_{x_2g_1} & \sigma_{x_2g_2} \\ \sigma_{y_1x_1} & \sigma_{y_1x_2} & \sigma_{y_1y_1} & \sigma_{y_1y_2} & \sigma_{y_1g_1} & \sigma_{y_1g_2} \\ \sigma_{y_2x_1} & \sigma_{y_2x_2} & \sigma_{y_2y_1} & \sigma_{y_2y_2} & \sigma_{y_2g_1} & \sigma_{y_2g_2} \\ \sigma_{g_1x_1} & \sigma_{g_1x_2} & \sigma_{g_1y_1} & \sigma_{g_1y_2} & \sigma_{g_1g_1} & \sigma_{g_1g_2} \\ \sigma_{g_2x_1} & \sigma_{g_2x_2} & \sigma_{g_2y_1} & \sigma_{g_2y_2} & \sigma_{g_2g_1} & \sigma_{g_2g_2} \end{bmatrix}, \tag{7.11}$$

where σ_{ab} is the covariance between state elements a and b. Given that we are only observing the location elements, the observation covariance matrix would be given by:

$$\mathbf{R} = \begin{bmatrix} r_{x_1x_1} & r_{x_1x_2} & r_{x_1y_1} & r_{x_1y_2} \\ r_{x_2x_1} & r_{x_2x_2} & r_{x_2y_1} & r_{x_2y_2} \\ r_{y_1x_1} & r_{y_1x_2} & r_{y_1y_1} & r_{y_1y_2} \\ r_{y_2x_1} & r_{y_2x_2} & r_{y_2y_1} & r_{y_2y_2} \end{bmatrix}, \tag{7.12}$$

where r_{ab} is the covariance between observation elements a and b.

Based on the above matrices, we can calculate $\mathbf{CH}^T$ to be the matrix:

$$\mathbf{CH}^T = \begin{bmatrix} \sigma_{x_1x_1} & \sigma_{x_1x_2} & \sigma_{x_1y_1} & \sigma_{x_1y_2} \\ \sigma_{x_2x_1} & \sigma_{x_2x_2} & \sigma_{x_2y_1} & \sigma_{x_2y_2} \\ \sigma_{y_1x_1} & \sigma_{y_1x_2} & \sigma_{y_1y_1} & \sigma_{y_1y_2} \\ \sigma_{y_2x_1} & \sigma_{y_2x_2} & \sigma_{y_2y_1} & \sigma_{y_2y_2} \\ \sigma_{g_1x_1} & \sigma_{g_1x_2} & \sigma_{g_1y_1} & \sigma_{g_1y_2} \\ \sigma_{g_2x_1} & \sigma_{g_2x_2} & \sigma_{g_2y_1} & \sigma_{g_2y_2} \end{bmatrix}. \tag{7.13}$$

From this, we can calculate $\mathbf{HCH}^T$ to be the matrix:

$$\mathbf{CH}^T = \begin{bmatrix} \sigma_{x_1x_1} & \sigma_{x_1x_2} & \sigma_{x_1y_1} & \sigma_{x_1y_2} \\ \sigma_{x_2x_1} & \sigma_{x_2x_2} & \sigma_{x_2y_1} & \sigma_{x_2y_2} \\ \sigma_{y_1x_1} & \sigma_{y_1x_2} & \sigma_{y_1y_1} & \sigma_{y_1y_2} \\ \sigma_{y_2x_1} & \sigma_{y_2x_2} & \sigma_{y_2y_1} & \sigma_{y_2y_2} \end{bmatrix}. \tag{7.14}$$

We may therefore conclude that the gain matrix, $\mathbf{K}$ is given by:

$$\mathbf{K} = \begin{bmatrix} \sigma_{x_1x_1} & \sigma_{x_1x_2} & \sigma_{x_1y_1} & \sigma_{x_1y_2} \\ \sigma_{x_2x_1} & \sigma_{x_2x_2} & \sigma_{x_2y_1} & \sigma_{x_2y_2} \\ \sigma_{y_1x_1} & \sigma_{y_1x_2} & \sigma_{y_1y_1} & \sigma_{y_1y_2} \\ \sigma_{y_2x_1} & \sigma_{y_2x_2} & \sigma_{y_2y_1} & \sigma_{y_2y_2} \\ \sigma_{g_1x_1} & \sigma_{g_1x_2} & \sigma_{g_1y_1} & \sigma_{g_1y_2} \\ \sigma_{g_2x_1} & \sigma_{g_2x_2} & \sigma_{g_2y_1} & \sigma_{g_2y_2} \end{bmatrix} \begin{bmatrix} \sigma_{x_1x_1} + r_{x_1x_1} & \sigma_{x_1x_2} + r_{x_1x_2} & \sigma_{x_1y_1} + r_{x_1y_1} & \sigma_{x_1y_2} + r_{x_1y_2} \\ \sigma_{x_2x_1} + r_{x_2x_1} & \sigma_{x_2x_2} + r_{x_2x_2} & \sigma_{x_2y_1} + r_{x_2y_1} & \sigma_{x_2y_2} + r_{x_2y_2} \\ \sigma_{y_1x_1} + r_{y_1x_1} & \sigma_{y_1x_2} + r_{y_1x_2} & \sigma_{y_1y_1} + r_{y_1y_1} & \sigma_{y_1y_2} + r_{y_1y_2} \\ \sigma_{y_2x_1} + r_{y_2x_1} & \sigma_{y_2x_2} + r_{y_2x_2} & \sigma_{y_2y_1} + r_{y_2y_1} & \sigma_{y_2y_2} + r_{y_2y_2} \end{bmatrix}^{-1}. \tag{7.15}$$

In essence, this provides a set of weights to be applied to the perturbation. This differs from the gain matrix that would be used in used in Chapter 6 in that it also includes weights to be applied to the entries in the state pertaining to the unobserved destination variables. The weight applied to these unobserved variables will depend on the state covariance between the unobserved variables and the other variables, e.g. $\sigma_{g_1x_1}$ which is the covariance between the unobserved gate variable of the first agent and the x-location of the first agent which is observable.

One of the issues that may arise in such an application is that of differing orders of magnitude of variables in the state vector; x-locations can vary between 0 and 740 and y-locations can vary between 0 and 700 whilst gate numbers can vary between 0 and 10. This is important because the EnKF bases the weighting applied to the observations, in part, on the covariance matrix of the state ensemble. Given these scales, it is likely that elements of the state ensemble covariance matrix pertaining to the gate numbers will be much smaller than those pertaining to the locational elements of the matrix; this is not necessarily because the gate numbers do not vary with respect to the locational elements but more so because the scale of variation in gate numbers is much smaller.

One option for resolving this issue is to rescale all of the variables in the state ensemble prior to data assimilation (Katzfuss et al., 2016) (also, reversing the scaling process after data assimilation). For the purpose of this investigation, variables are scaled linearly against the range of possible values for the variable such that the upper bound is scaled to 1, the lower bound is scaled to -1 and the midpoint is scaled to 0. This is formalised as:

$$a_r = \frac{a - m}{s} \tag{7.16}$$

where a_r is the rescaled value of variable a, m is the midpoint of the upper and lower limits of a's values and s is a scaling factor. The midpoint, m, can be calculated as

$$m = \frac{a_{max} + a_{min}}{2},$$

where a_{max} is the maximum possible value of variable a and a_{min} is the minimum possible value of a. The scaling factor, s is calculated as

$$s = a_{max} - m.$$

This scaling process can be reversed by applying the inverse operation:

$$a = m + s \times a_r \tag{7.17}$$

Aside from the issue of variable scaling, one further problem to consider is that of how the EnKF estimates a discrete variable. Typically, the EnKF is applied to problems in which the estimated quantities are continuous (Kalnay, 2003), however, in this case, we are attempting to estimate a discrete category for each agent in the form of exit gate. Whilst we may provide agents with discrete integer gate numbers at the outset of the simulation run, the EnKF is likely to arrive at non-integer values when updating these quantities based on assimilated data. In order to deal with this problem, a relatively simple solution is proposed — Given the ensemble mean state vector, we can apply the following steps:

- **Step 1:** For each estimated gate number, round it to the nearest integer.
- **Step 2**: For each integer gate number, apply the modulus with respect to the total number of gates in the environment.

As an example, consider an ensemble mean state vector for a system with 3 pedestrians in it. In this case, the elements of the mean state vector might look something like the following after having been updated with assimilated data:

$$[1.25, 10.8, 5.5]$$

Clearly, none of these estimated values can be considered valid gate numbers as they are not integers and so the above steps are applied to each entry. In the case of the entry relating to the first pedestrian (1.25), the rounding process rounds the estimated gate down to 1 and the modulus step makes no difference resulting in an estimated gate number of 1. In the case of the second entry (10.8), the rounding process rounds the estimated gate up to 11 and the application of the modulus with respect to the number of gates (11) results in an estimated gate number of 0. In the case of the third entry (5.5), the rounding process round the estimated gate up to 6 despite the estimate being halfway between 5 and 6 which may result in some incorrect estimations; the likelihood of a gate estimate being exactly halfway between integer numbers is, however, extremely unlikely. As in the case of the first entry, the modulus has no impact on this entry. The result after applying this process is the following vector of estimated gate numbers:

$$[1, 0, 6]$$

This approach is also applied to the state vectors representing the individual ensemble-member models. An alternative approach to this would be to apply the rounding process to the gate estimates in each of the ensemble-member models, and then to calculate the average state by taking the modal gate for each of the agents across the ensemble.

Having estimated a destination gate for each agent, a target destination, i.e. an x-y location along the edge of the environment, can be allocated using a method built into the `Agent` class; this method provides a location by sampling from a uniform random distribution along the length of the provided gate.

Destination Angle Estimation

Having outlined the initial approach to gate estimation using the Ensemble Kalman Filter in the previous section, this section focuses on a different approach. One of the issues with the previous approach is that it framed the problem as one that was purely categorical, i.e. that of identifying the correct gate number. Whilst this is an important facet of accurately simulating a system given unknown destinations for the pedestrian population, it leaves the ultimate point destination up to a random distribution. The filter is required to primarily identify the correct exit gate for each agent, but furthermore it is required to identify the point along the length of the gate which is acting as the target destination. When choosing the point destination using the uniform random distribution spanning the width of a gate, we acknowledge that there is a continuous element of the problem. This section aims to frame the problem as a continuous one, looking to estimate points around the edge of the environment towards which a pedestrian is headed as described by the angle made between this point and the centre of the environment.

As was the case in the previous approach, this requires that we make modification to the EnKF. Once again, we change the way that we represent an ensemble-member model as a state vector from

$$[x_1, y_1, \ldots, x_N, y_N,] ,$$

given a population of N pedestrians, to

$$[x_1, \ldots, x_N, y_1, \ldots, y_N, \theta_1, \ldots, \theta_N] ,$$

where θ_i represents the angle made between the centre of the environment and the pedestrian's target destination on the edge of the environment. Angles are measured in radians relative to the x-axis extending from the environment centre, i.e. $(370, 350)$, to the point where it intersects on the right boundary, i.e. $(740, 350)$. Angles range between $+\pi$ and $-\pi$, with a positive angle indicating an anti-clockwise rotation from the reference axis and a negative angle indicating a clockwise rotation from the reference axis. As an example, the central point on the top boundary, i.e. $(370, 700)$, would be pointed to by an angle of $\pi/2$, whilst the central point on the bottom boundary, i.e. $(370, 0)$, would be pointed to by an angle of $-\pi/2$.

Just as in the previous approach, we change our observation operator from $\mathbf{I}_{2N}$, i.e. the identity matrix of size $2N$ where N is the population size, to $[\mathbf{I}_{2N}, \mathbf{0}_{2N\times N}]$.

As when using the destination gate number to determine an agent's target destination in the previous section, when using an angle, we again choose to rescale the variables in the state vector before performing data assimilation (as we as reversing the process after having adjusted the state vector in response to the assimilated observations). The elements pertaining to the x- and y-locations of agents in the environment are, again, scaled as outlined in Equation 7.16 with respect to the maximum and minimum x and y values.

Elements of state vectors pertaining to angles around the environment are scaled with respect to the maximum and minimum angular values — π and $-\pi$ respectively.

The result is that an angle of π would be rescaled to 1 and an angle of $-\pi$ would be rescaled to -1. Considering an example of an angle somewhere between, an angle of $\theta = \pi/2$ would be rescaled to 0.5.

When considering the previous approach, one of the issues that arose was that we needed to allocated a target destination in 2-dimensional real space having estimated a categorical integer value; whilst the present approach seeks to remedy this problem, it is not without its own challenges. Using the EnKF to estimate the continuous value of the angle around the boundary of the system environment can result in an angle ranging between $-\pi$ an π, however, not every angular value in this interval is considered a valid target destination.

The posterior estimates of agents' target destination angles can fall into one of two categories: angles which point to locations along the environment boundary which lie within a gate and those which point to locations which lie between two gates. The former case is that which we desire and describes a valid target destination; an example of such an angle would be $\theta = \pi$ which points to a location on the left edge of the environment which lies within gate 0. The latter case, however, does not describe a valid target destination and therefore requires some adjustments; an example of such an angle might be $\theta = \pi/2$ which points to a location which lies between gates 1 and 2.

In the case of angles which point to locations between two gates, we identify the gate edge which is closest in angle to the angle that has been estimated, and allocate that edge location as the target destination for the agent. If the difference angle between the angle and the two adjacent angles is the same (which is very unlikely), we randomly choose one of the two adjacent angled edge locations. Considering the case of $\theta = \pi/2$ above, this angle points to a location which lies closer to the edge of gate 2, and as such would be rounded to the nearest edge of gate 2.

7.2 Experimental Design

In the previous section, we outlined proposed solutions to two technical problems: how do we encode initial uncertainty regarding a pedestrian's target destination when they first enter the system, and how do we modify the Ensemble Kalman Filter to allow it to provide updates for both the pedestrian's simulated location in the system and their target destination in response to assimilated data. In this section, we will outline how we plan to put these solutions into practice by describing the experiments that shall be undertaken to test the efficacy of the EnKF in such a setting. This comprises of two parts. The first of these outlines the set of benchmarking experiments to be undertaken, which aim to demonstrate the shortcomings of the model and the filter when a pedestrian's target destination is not known and the representation of the target destination within the filter is not updated in response to assimilated observation. The second of these outlines the set of experiments undertaken using the two approaches to estimating a pedestrian's destination outlined above — gate number estimation and destination angle estimation.

7.2.1 Benchmarking

In this section, we outline the experiments undertaken to define a benchmark against which to compare filter performance when updating both the locational state of the model and the destinations of the agents within the model. This benchmarking takes two forms. First of all, we consider the case where pedestrians' target destinations are unknown and no data assimilation is undertaken; in essence, this involves running an ensemble of models without any data assimilation. This mirrors the benchmarking that was undertaken previously in Chapters 5 and 6; however, in previous benchmarking attempts, knowledge of pedestrians' final destinations was assumed whilst in this case we remove that knowledge. Following on from this, we consider a similar scenario in which pedestrians' target destinations are unknown, but this time we use the EnKF to perform data assimilation with periodic observations of the pedestrians' locations; this data assimilation, however, is only used to update the locational state of the ensemble-member models and not the pedestrians' target destinations. In each case, we consider the impact of the different ways in which we can represent unknown destinations for the pedestrians within the EnKF as outlined in Section 7.1.1. This will provide us with a benchmark against which to compare when we make use of observations to infer the latent pedestrian destinations.

The first benchmarking experiment focuses on the case in which an ensemble of models is run to simulate the motion of pedestrians across the model environment without any observations being assimilated. Just as in Chapter 6, this is done by instantiating an ensemble of models as copies of a base model which is used to define the ground truth state at all times. In this case, however, the ensemble of copies differ from the ground truth model in that the agent representations of the pedestrians in the system do not necessarily share the same target destination as those defined in the base model.

This is achieved by undertaking the different randomisation approaches outlined in Section 7.1.1; the aim of this is to emulate a more realistic scenario in which we have either limited or no knowledge regarding the target destination of pedestrians when they enter the system. Just as when performing benchmarking in previous chapters, the efficacy of the ensemble of models with respect to how well they simulate the system in question is assessed by looking at the average distance error per agent. The distance error for pedestrian i in a population of N agents, d_i, are defined as the distance between the ith pedestrians' location in the base model, $\mathbf{x}_i$, and the mean of the ensemble for active agents, $\hat{\mathbf{x}}_i$:

$$d_i = \begin{cases} |\hat{\mathbf{x}}_i - \mathbf{x}_i| & \text{if } i\text{th agent is active;} \\ 0 & \text{otherwise,} \end{cases} \tag{7.18}$$

and this error is averaged over the active agent population as outlined in Chapter 6:

$$\bar{d} = \frac{1}{N} \sum_{i=1}^{N} d_i, \tag{7.19}$$

where N is the number of active agents. Here, we assess agent activity based on the modal status across the ensemble; for a full mathematical description of how this works, see Section 6.1.1).

The first benchmarking experiment is run using a population size of 5 and an ensemble size of 100 (as outlined in Table 7.2). The small population size here is chosen to ensure that the uncertainty generated in the model is predominantly due to uncertainty in the initial allocation of pedestrian target destination as opposed to agent-agent interactions (as was observed in previous chapters).

Parameter	Value
Population size	5
Ensemble size	100
Assimilation period	100
Observation noise standard deviation	5

Table 7.2: Table of parameter value used for experiments in this chapter.

This experiment is expected to generate a collection of three time-series datasets of average error per agent over time, with each dataset pertaining to one of the three randomisation approaches. These will be plotted and compared. It is expected that given the randomisation of the pedestrian target destinations and the lack of data assimilation, the ensemble of models will perform relatively poorly in each case. The cases in which destinations are randomised based on the adjacent gate randomisation approach are expected to perform better than the other two approaches as this approach incorporates some substantive knowledge of the true pedestrian destination with some finite noise attached.

Having run the first benchmarking experiment, the second benchmarking experiment seeks to apply data assimilation, updating the ensemble of models with periodic observations. In this case, the state vector consists of the locations of the agents in each model, and as such only the locational aspects of the model are updated when observations are assimilated — not the agents' destinations. Just as in the previous benchmarking experiment, this involves instantiating an ensemble of models as copies of a base model, again applying different destination randomisations. The EnKF is instantiated with a population size of 5 and an ensemble size of 100, with observations with noise of standard deviation 5 being assimilated every 100 simulation time-steps (as defined in Table 7.2). We also use the same measure of efficacy as in the previous benchmarking experiment: the average distance error per agent.

This set of experiments will generate a collection of three time-series datasets of average error per agent over time, with each dataset pertaining to one of the three randomisation approaches. Each of these datasets will contain information for the average error per agent before and after data are assimilated, i.e. the prior and posterior error. Just as in the previous experiment, the case in which gates are randomised using the adjacent approach is expected to perform better than cases using the other approaches (both with respect to prior and posterior error) given the encoding of knowledge regarding each pedestrians' target destination. It is expected that in the case in which gates are allocated uniformly by agent across the ensemble will perform particularly poorly both in terms of prior and posterior error. This is expected due to the uniformity of destinations across the ensemble — the path taken by the agents pertaining to a specific pedestrian are likely to be very similar across the ensemble of models and consequently the uncertainty in agent position in the ensemble of models is likely to low in comparison to the uncertainty in the observation. As a result, the weight applied to the observation by the EnKF will not be sufficient to adjust the path of the agents. In the case in which gates are allocated randomly across the ensemble, it is expected that, whilst the prior state of the filter may not perform particularly well, the posterior will perform much better. This is because, given the destination allocation, the uncertainty in the model estimate of the agent locations will be large in comparison to the observation uncertainty and subsequently the filter will place a greater weight on the observation when updating the filter state.

7.2.2 Estimating Pedestrian Destinations

In the previous section, we outlined the experiments that would be undertaken in order to establish a benchmark against which to compare filter performance. This consisted of two benchmarks: the first pertained to the worst case scenario in which pedestrians' destinations are unknown and no observations are provided to update the system via data assimilation, whereas the second pertained to the scenario in which pedestrians' destinations are still unknown but observations are provided to update the locations of agents within the ensemble of models via the EnKF. In this section, we seek to outline a set of experiments to demonstrate that the EnKF can be used to update not only agents' locations, but also to determine their destinations in the case where such a parameter is unknown at the outset of the simulation.

Estimation of agents' destinations is undertaken through state augmentation within the EnKF whereby destinations are incorporated into each of the state vectors for the ensemble-member models as outlined in Section 7.1.2. The two approaches outlined in this section focus including either the target destination gate number or the angle between the target destination and the centre of the environment in the state vector.

Having completed the benchmarking experiments, it expected that the gate allocation approach whereby destinations are uniform across the ensemble for each agent will perform sufficiently poorly that it can excluded from subsequent experiments. As a result, this set of experiments considers the impact of varying the approach to destination estimation (which can be either gate number estimation or angle estimation) and initial destination allocation (which can be either random across the ensemble or uniform across a finite number of gates adjacent to the true gate). Based on this, this set of experiments consists of considering four different scenarios. In each scenario, the

EnKF is instantiated with a population of size 5, an ensemble of size 100 and agent destinations are randomised based on one of the two allocation approaches outlined. The filter is then run forward through simulation time, assimilating observations of pedestrians' locations which are provided periodically every 100 time-steps, and which are generated by taking the pseudo-truth state from the base model and adding normally distributed random noise with standard deviation 5 (as per the parameter values in Table 7.2).

The experiments outlined in the previous section sought to demonstrate the poor performance of an ensemble of models in the absence of knowledge regarding pedestrians' destination. In this experiment we wish to demonstrate that updating the destinations via state augmentation within the EnKF improves the accuracy with which our ensemble simulates the system. In this respect we still wish to use the average distance error per agent as our measure of filter effectiveness; this can be demonstrated by plotting the time-series dataset of the average error per agent. From this, it is expected that (just as in the case of the second benchmarking experiment) the posterior error will be lower than the prior error over the course of the simulation time for each of the combinations of initial destination allocations and destination estimation approaches.

Beyond this, however, we also wish to assess whether the filter is capable of correctly estimating the destination for pedestrians. Whilst inspection of the average distance error per agent may give us an indication of whether destinations have been correctly estimated, it may not necessarily be conclusive; we may, therefore, wish to employ further measures to assess this. One of the easiest ways to verify that a destination has been correctly estimated is by the following two plots for each pedestrian in the population:

- A line plot of the path taken by the pedestrian; for this, we plot a lines showing the positions representing the prior ensemble state mean, the posterior state mean, the true path taken in the base model and points representing the observations derived from the base model. Furthermore, we include the agent's origin and destination within the environment.

- A histogram showing both the initial distribution of destinations at the outset of the simulation run and final distribution of destinations at the end of the simulation run for the given pedestrian agent. Furthermore, we include a line indicating the true destination from the base model. In cases where gate number estimation is employed, histograms show the distribution of estimated gate numbers and the true destination is a single value for the gate number. In cases where angle estimation is employed, histograms show the distribution of angles around the environment and the true destination is a range of angles between the edges of the true gate.

7.3 Results

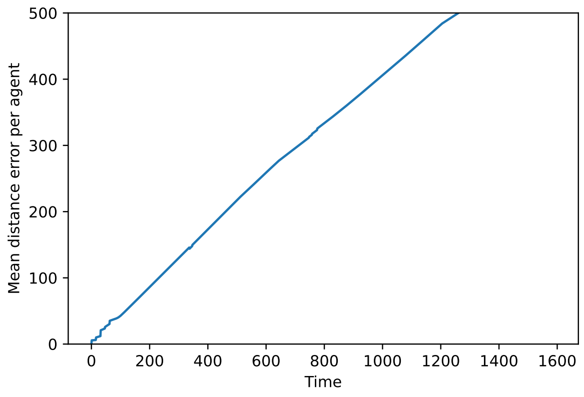
Having set out the experiments to be undertaken in Section 7.2, this section seeks to present the results of these experiments[1]. This consists of two section. The first of these — Section 7.3.1 — presents the results of the two different benchmarking experiments which show the consequences of running an ensemble of models without knowledge of pedestrians' destinations, both with and without location data being assimilated into the ensemble. The second section — Section 7.2.2 — presents the results of applying state augmentation within the EnKF to update agents' destinations as well as their locations when assimilating data; this considers multiple ways in which agent destinations can be encoded when applying state augmentation.

7.3.1 Benchmarking

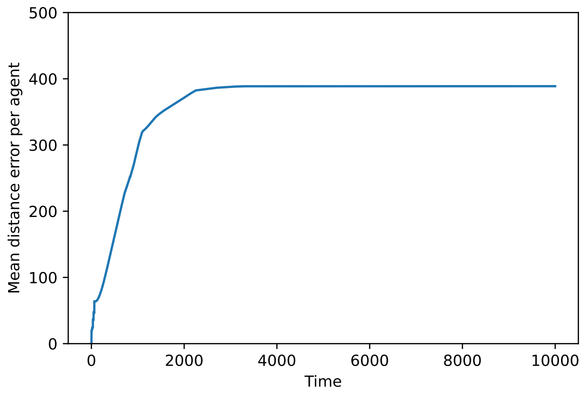
This section outlines the results of the benchmarking experiments undertaken in this chapter. The first of these focuses on exploring the impact of different ways in which we encode our lack of knowledge regarding pedestrians' destinations in a scenario where we simulate the system without assimilating any data. This is achieved by exploring the way in which the average distance error per agent varies over simulated time.

This variation is shown in Figure 7.1, which consists of three sub-figures. Each of the three sub-figures shows the variation of average distance error per agent given a specific approach to allocating agent destinations within the ensemble of models: Figure 7.1a shows the results of allocating destinations uniformly across the ensemble for each agent; Figure 7.1b shows the result of allocating destination randomly across the ensemble for each agent; Figure 7.1c shows the result of allocating destinations randomly drawn from a uniform distribution of destinations adjacent to the true destination for each agent.

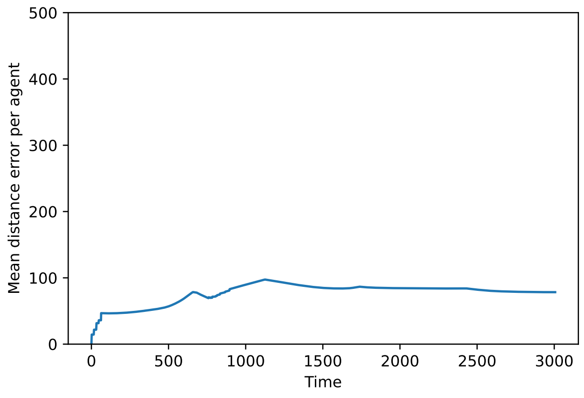
[1]The experiments run for this chapter can be found in the notebooks found in `Projects/ABM_DA/experiments/enkf_experiments/results_3/notebooks/` in the dust repository archive



(a) Uniform Across Ensemble

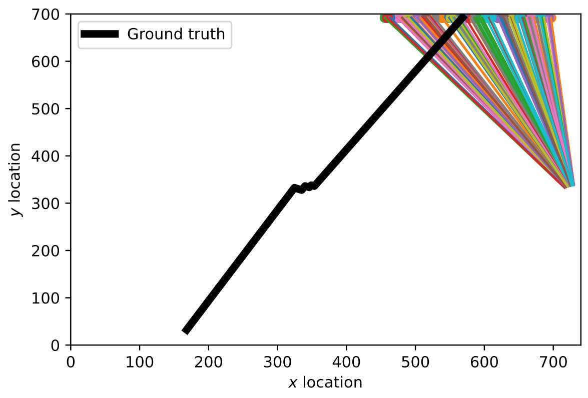


(b) Random Across Ensemble

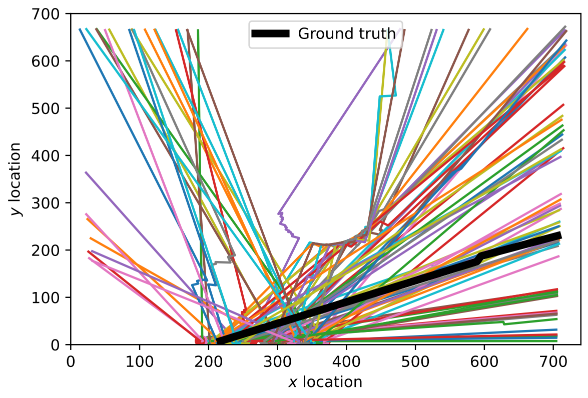


(c) Adjacent

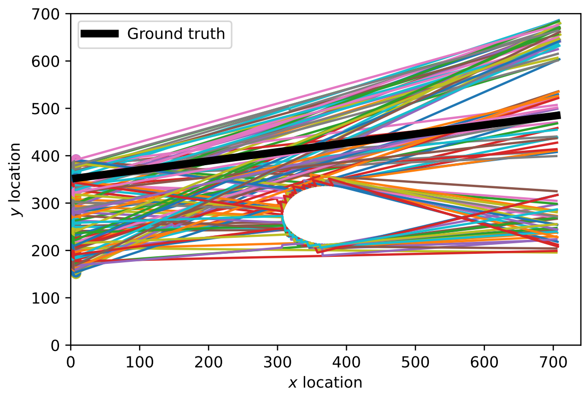
Figure 7.1: Variation in average error per agent over simulation time with no data assimilation.



(a) Uniform Across Ensemble



(b) Random Across Ensemble



(c) Adjacent

Figure 7.2: Sample trajectories of a single agent represented in each of the benchmarking ensemble models. Trajectory of corresponding agent in base model provided for comparison.

In Figure 7.1a, we see the variation in average distance error per agent when destinations are allocated uniformly across the ensemble for each pedestrian. Considering this figure, we see that the average error starts very small. This is reflective of how the ensemble of models is created; each of the ensemble-member models is a copy of the base model and, as such, the starting locations of the agents in each of the models is the same as in the base model. Whilst the initial average error may be small, it quickly grows and, unlike in Chapters 5 and 6, does not decrease over the course of the simulation run. This is primarily because no data are being assimilated into the ensemble of models and therefore the model states are not being corrected by observations of the true system state. Beyond this, however, the random allocation of destinations for each of the pedestrians in the system also means that agents in the ensemble-member models do not deactivate as a result of reaching the correct destinations but instead due to reaching other gates. As a consequence, each of the ensemble-member models may deactivate when their agents have completed their journeys across the environment, but this does not mean that the agents have reached the same destinations as their counterparts in the base model. This is reflected in Figure 7.2a.

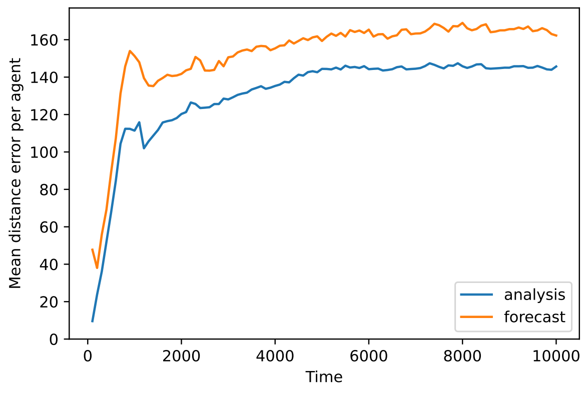
When considering Figure 7.1b, we observe a similar pattern. The initial average distance error per agent is small at the outset of the simulation run as a result the ensemble-member models being a copy of the base model and consequently the agents in each of the models sharing starting locations with the corresponding agents in the base model. The average distance error per agent quickly grows as the system evolves, with the ensemble mean state diverging from that of the base model. Given that the destinations for each of the pedestrians are allocated randomly across the ensemble, there is a chance that the destination of an agent in one of the ensemble-member models may match that of the corresponding agent in the base model; the probability of this occurring, however, is relatively low. In the case that an agent starts on the left side of the environment (the side with the fewest gates), the probability that each of the gates from the other boundaries are allocated is 1/10; in the case that an agent starts on either the right or bottom sides of the environment (the sides with the most gates), the probability that each of the gates from the other boundaries are allocated is 1/7. In addition, given that the destinations are randomised across the ensemble, the representations of a given pedestrian in each of the ensemble-member models is likely to be different to each other. As a result, the paths of the agents in each of these models are likely to be markedly different from each other. This is reflected in Figure 7.2b.

When considering Figure 7.1c, we observe a different pattern. The initial average distance error per agent is small at the outset of the simulation run; just as in the previous two cases, this is a result of the ensemble-member models being copies of the base model. Whilst we do observe an increase in the average error as the system evolves, this growth is not as great as in the other two scenarios. This is a result of the increased level of information regarding pedestrian destinations incorporated into the ensemble by the gate allocation approach; the reasoning behind this was outlined in the corresponding part of Section 7.1.1. Whilst the final average error per agent seen in this figure is larger than that we observed in the experiments in Chapters 5 and 6, it is less than the width of the majority of the gates, suggesting that on average the ensemble has estimated that the pedestrians will end up in approximately the correct location. This is reflected in Figure 7.2c.

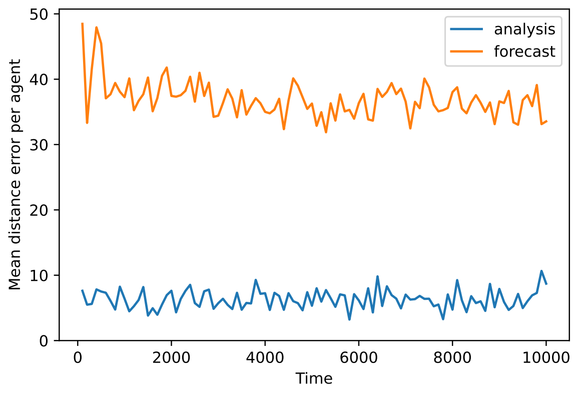
Having defined initial benchmarks for simulating a system with an ensemble without any data assimilation, we now turn to the results of simulating a system with an ensemble with data assimilation where the EnKF is used to update only the locations of agents within the ensemble-member models. This is explored using Figures 7.3, 7.5 and 7.6.

Figure 7.3 shows the variation in average distance error per agent over time, just as was shown in Figure 7.1 for the previous benchmarking experiment. In this case, however, we are able to show both the prior and posterior error as we are assimilating data to update the agent locations within the ensemble-member models. Each of the three sub-figures shows the variation of average distance error per agent given a specific approach to allocating agent destinations within the ensemble of models: Figure 7.3a shows the results of allocating destinations uniformly across the ensemble for each agent; Figure 7.3b shows the result of allocating destination randomly across the ensemble for each agent; Figure 7.3c shows the result of allocating destinations randomly drawn from a uniform distribution of destinations adjacent to the true destination for each agent.

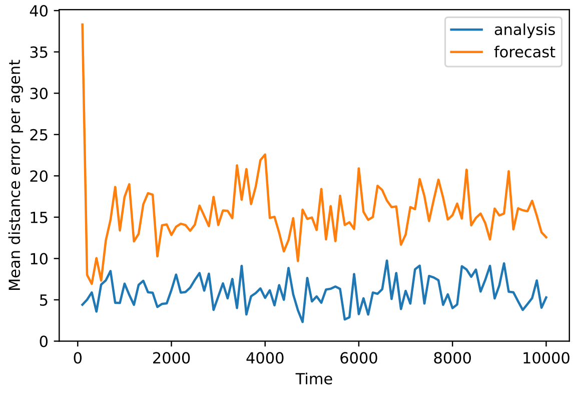
In each of the three cases, we observe that the posterior average errors are lower than those observed in the respective cases in Figure 7.1. This is, unsurprisingly, a result of the ensemble of models being updated with observations of pedestrians' locations derived from the base model. In each case, the posterior errors are lower over the course of the simulation than the respective prior errors, again, reflecting the benefits



(a) Uniform Across Ensemble

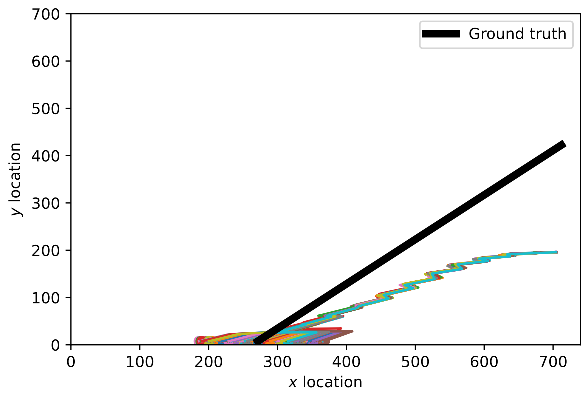


(b) Random Across Ensemble

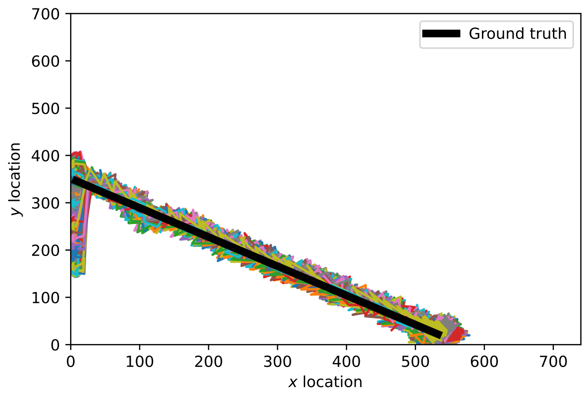


(c) Adjacent

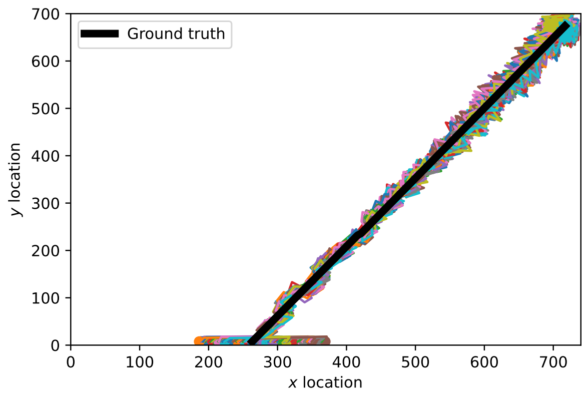
Figure 7.3: Variation in average error per agent over simulation time when assimilating data to update location.



(a) Uniform Across Ensemble



(b) Random Across Ensemble



(c) Adjacent

Figure 7.4: Sample trajectories of a single agent represented in each of the ensemble-member models. Trajectory of corresponding agent in base model provided for comparison.

of assimilating observations to update the model states.

These improvements are not, however, consistent across the different allocation approaches. When comparing Figures 7.3a and 7.3b, we notice that in the case of the former the assimilation of data results in a posterior error which consistently lies somewhere between 80 and 100, whilst in the case of the latter the assimilation of data results in a posterior error that is consistently below 10 — a marked improvement. The improvement seen here is a consequence of the destination allocation method used in each case.

In the former case, destinations are allocated uniformly across the ensemble to agents representing a given pedestrian. As a result, the paths taken by the agents across the environment between assimilation time-steps do not differ greatly, and so the uncertainty in the prior positions represented by the ensemble is typically small in comparison to the uncertainty in the observations. The EnKF therefore places less weight on the observations when updating the state, and consequently the state does not improve by much. This is reflected in Figure 7.4a.

In the latter case, however, destinations are allocated randomly across the ensemble to agents representing a given pedestrian. As a result, the paths taken by the agents across the environment between assimilation time-steps are markedly different; this means that the uncertainty in the prior positions represented by the ensemble is large in comparison to the uncertainty in the observations. The EnKF therefore places much more weight on the observations when updating the state. This is reflected in Figure 7.4b.

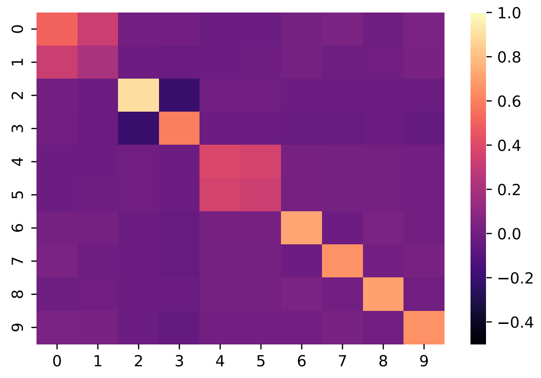
This difference in weighting applied to the observation can be observed in Figure 7.5. This figure shows a heatmap for the gain matrix at the end of the simulation run for each of the destination allocation methods. Recall from the Chapter 4 that the gain matrix indicates the weight applied to the observations in updating the elements of the state ensemble matrix. For example, the entry in the first column pertains to the weight applied to updating the x-location of the first agent in the first model with the x-location of the first observation. We expect that, in each case, elements off of the diagonal should be 0 as it is unlikely that observations of a pedestrian would contribute to the updating of the agents representing another pedestrian; this is confirmed by inspecting each of Figures 7.5a, 7.5b and 7.5c. We can judge the degree to which updates from observations are weighted by considering the values along the diagonal of the gain matrix. In the case of Figures 7.5b and 7.5c (i.e. cases where destinations are allocated randomly across the ensemble and where destinations are allocated from the true destination and the adjacent gates respectively), we observe that the values along the diagonal are high, typically between 0.8 and 1.0. In the case of Figure 7.5c, we see that there is some variation in value but this variation is not large.

In comparison, when we look at Figure 7.5a, we see that the values along the diagonal are lower than in the other heatmaps. This indicates that less weight is applied to the update from the observations than in the other two cases; indeed, some elements along the diagonal have values close to 0, indicating that no weight is applied to observations and therefore the specific element of the state to which that element pertains is unchanged by the observations. Furthermore, we notice some non-zero values off of the diagonal.

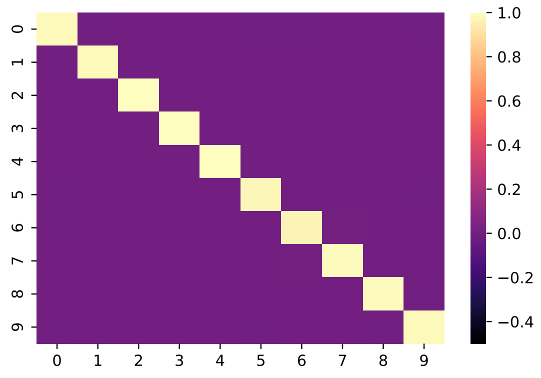
We can finally examine these scenarios by considering the impact of data assimilation on the position of one of the pedestrians being simulated. This is achieved by considering Figure 7.6. This figure shows the distribution of x- and y-locations across the ensemble for a single pedestrian in the population, comparing the prior and posterior distributions and also indicating the observed location.

In Figure 7.6a, we see that the distribution of x- and y-locations has a very low uncertainty (relative to those seen in Figures 7.6b and 7.6c) indicated by the relatively small spread of the distributions. Furthermore, the locations given by the state ensemble from the models often differ substantially from the observed locations (and subsequently from the observations). Due to the low levels of uncertainty in the ensemble state, we find that the posterior often does not differ much from the prior distribution; in Figure 7.6a, this is particularly noticeable in the case of the x-location. The more noticeable difference between the prior and posterior in the y-location arises due to the greater degree of uncertainty found in the prior y-location, allowing the state to be updated in response to the introduction of observations.

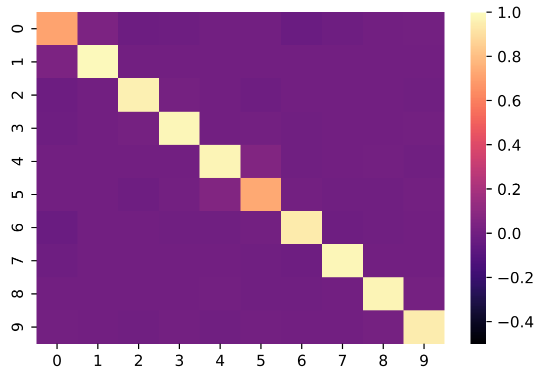
In Figure 7.6b, we see that the range of the prior distributions of x- and y-locations are much greater than those seen in Figure 7.6a. Just as in the previous scenario, the lack of information regarding the destination of the pedestrian results in the prior estimate being incorrect, and consequently not lying near the observed locations of the true locations. Given the much larger degree of uncertainty in the x- and y-locations, the EnKF places a much larger weight on the impact of the observation,



(a) Uniform Across Ensemble

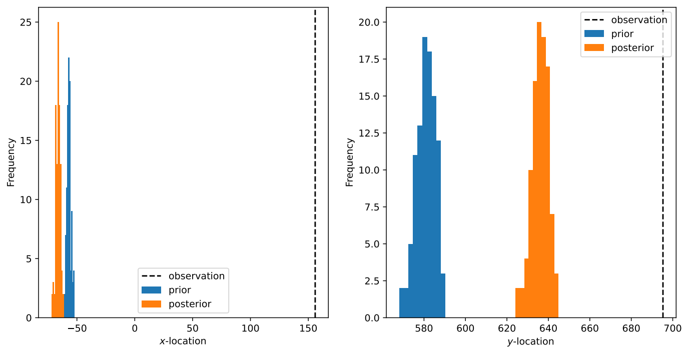


(b) Random Across Ensemble

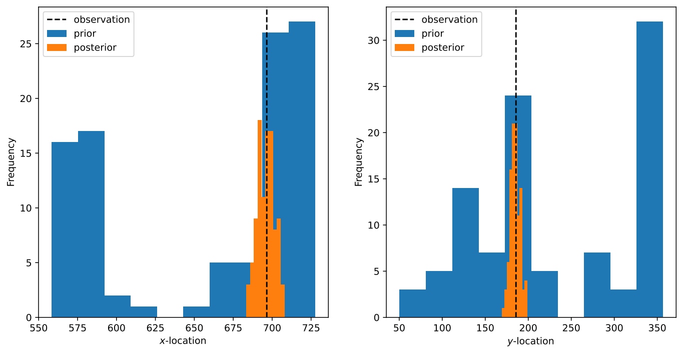


(c) Adjacent

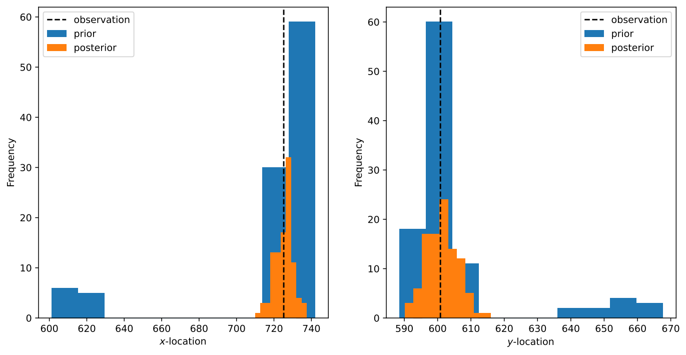
Figure 7.5: Heatmap of gain matrix for final data assimilation of simulation run.



(a) Uniform Across Ensemble



(b) Random Across Ensemble



(c) Adjacent

Figure 7.6: Distributions of x- and y-location for first pedestrian before and after the indicated observation was assimilated.

and consequently this results in a marked difference between the prior and posterior distributions; this is noticeable both with regard to the update in location and the degree of uncertainty in the posterior compared to the prior distribution.

In Figure 7.6c, we see that the range of the prior distributions of x- and y-locations are much smaller that those seen in Figure 7.6b, but larger than those seen in Figure 7.6a. In this case, this reduced uncertainty in the estimated locations of the pedestrian is a result of the increase degree of knowledge that we have incorporated in the models of the pedestrian's target destination. The added benefit of this increased level of knowledge is that the prior distribution is typically closer to the true location of the pedestrian found in the base model as well. Given the relatively low level of uncertainty in the estimated locations, the impact of data assimilation is smaller than that observed in Figure 7.6b; however, in light of the increased level of knowledge regarding pedestrian destinations, the small update provided by the assimilated data is sufficient. Furthermore, the update of the estimated pedestrian location results in a reduction in the uncertainty around the location.

As a result of these two experiments, we have established a benchmark for the degree of error that we might expect when running an ensemble of models with poor knowledge regarding pedestrians' destinations around the environment. However, we also see that even in the absence of knowledge regarding pedestrians' destinations, applying data assimilation to the system improves the accuracy with which we can simulate the system. In such a situation, we can conclude that a random allocation of destinations to agents across the ensemble is beneficial over a uniform distribution as the former introduces a larger degree of uncertainty in the estimates provided by the ensemble of models and the EnKF therefore relies more heavily on the observations provided. If, however, we have any prior knowledge regarding the destinations to which pedestrians are headed (even if this information has some level of uncertainty attached to it), this is even more beneficial.

In the next section, we shall build upon these benchmarks, applying the Ensemble Kalman Filter with state augmentation such that when data are assimilated, we update not only the estimated locations of the pedestrians but also the destinations. For this set of experiments, we shall discard the first destination allocation method — Uniform Across Ensemble — as we have seen that a more appropriate manner in which to encode our lack of knowledge regarding pedestrian destinations is the second approach, i.e. Random Across Ensemble.

7.3.2 Estimating Pedestrian Destinations

Having established benchmarks in the previous section, this section seeks to outline the results of running a state-augmented EnKF to undertake data assimilation to update not only the estimated locations of pedestrians but also estimates of their destinations as periodic observations become available. As outlined in Section 7.2.2, this is achieved by running the EnKF with the same parameter values as used in the benchmarking experiments with the addition that the destinations of pedestrians are included in the updating process using state augmentation (as detailed in Section 7.1.2). This section, therefore, details the results of running such experiments using two ways in which the state vector can be augmented and two ways in which we can incorporate uncertainty in the initial allocation of destinations; in the former case, we consider including the gate number in the state vector, and including a target location described by an angle in the state vector; in the latter case, we consider the Random Across Ensemble and Adjacent approaches for the initial allocation of destinations as used in the benchmarking experiments. As in the previous benchmarking experiments this is primarily achieved by considering the variation in average error per agent over the course of a simulation run for each scenario. In addition, however, we wish to explore whether the filter has been able to correctly identify the destination to which a pedestrian is heading; this is achieved by considering the distribution of destination estimates across the ensemble, as well as examining the path taken by agents across the environment and how this is updated in response to changing estimates of destinations.

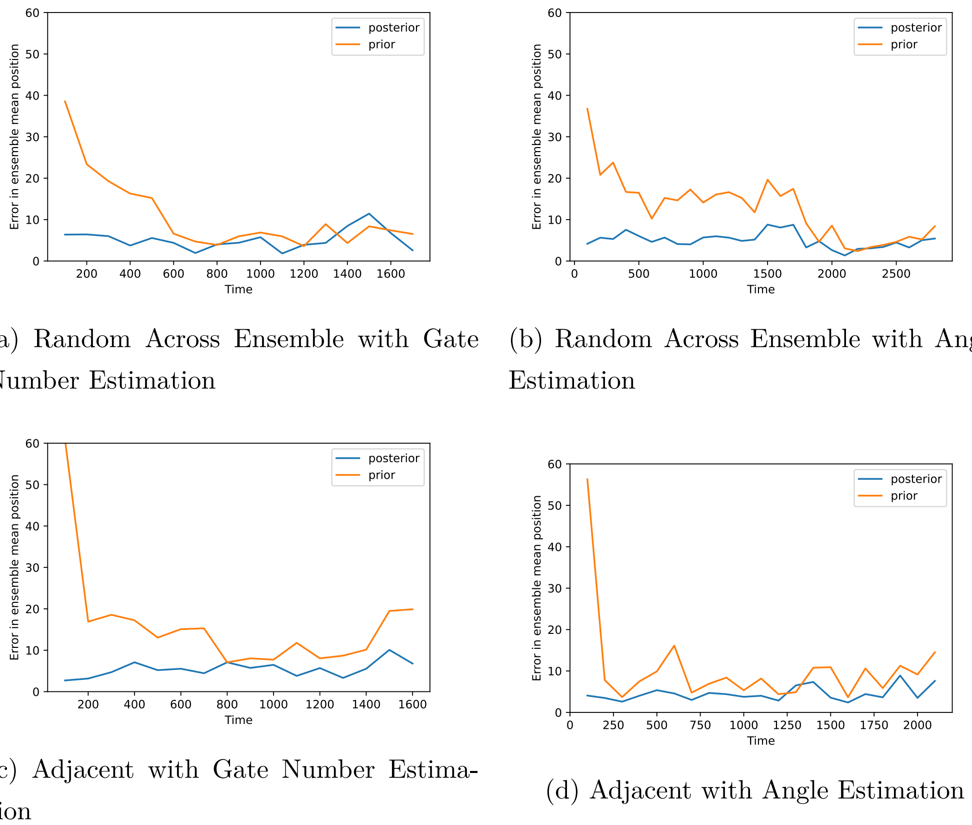
Looking at Figure 7.7, we can see the variation in average error per agent over the course of a simulation run. This comprises of four sub-figures.

The first of these — Figure 7.7a — shows the variation in prior and posterior error in the scenario when the destination is estimated by estimating the number of the gate to which pedestrians are headed and initial allocation of destinations is undertaken by randomly allocating destinations across the ensemble. In the case of both the prior and posterior errors, we see that over the course of the simulation run, they are lower than the corresponding errors seen in the final benchmarking experiment; indeed, by halfway through the simulation run, the prior error is almost as low as the posterior error seen in the corresponding benchmarking experiment in which data were assimilated to update just the locations (see Figure 7.3b). This is, perhaps, indicative that despite having started with random destinations allocated across the ensemble, the EnKF has updated these destination and consequently the agents in the ensemble-member models have acquired "correct" destinations (with respect to the destinations of the respective agents in the base model) — something that shall be further explored when considering Figure 7.8.

We next consider Figure 7.7b. In this figure, we see the variation of average error per agent over the course of the simulation run when initial destination allocation is undertaken by randomly allocating destinations across the ensemble, and over the course of the simulation destinations are updated by estimating the target angle around the environment towards which each agent is headed. As in the previous case, we see that both prior and posterior errors represent an improvement with regards to average error per agent in comparison to the corresponding benchmarking experiment (Figure 7.3b).

Looking at Figure 7.7c, we see a similar pattern — the variation in both the prior and posterior errors when applying the same form of state augmentation in conjunction with the Adjacent approach to allocation initial destinations are relatively low. The improvements seen here in comparison to the corresponding benchmarking experiment (the results for which can be see in Figure 7.3c) are not as marked as those seen in Figure 7.7a. This is, however, to be expected. In the previous case, the higher degree of uncertainty in pedestrian destinations meant that whilst the posterior ensemble state of the benchmark performed relatively well, time-steps between data assimilation steps saw the ensemble of models diverging from the true system state represented by the base model as the pedestrian destinations were not being updated and were likely very incorrect when compared to the destination of the agents in the base model. In this case, however, the corresponding benchmarking experiment was already able to incorporate some prior knowledge regarding the destinations of the pedestrians and, whilst this was not updated in the corresponding benchmarking experiment, this was sufficient for the benchmarking ensemble with data assimilation to provide a reasonable approximation of the path taken by the pedestrians.

This is corroborated by Figure 7.7d, in which we consider the scenario in which initial destinations were allocated using the Adjacent approach and destinations were updated over the course of the simulation by estimating the target angle around the edge of the environment. In this case, as in the previous case, both the prior and posterior average errors per agent represent improvements in comparison to the scenario depicted in Figure 7.3c, though these improvements are limited by the improvements offered by the use of the EnKF to perform simple location updating in conjunction with the Adjacent destination allocation method seen in Section 7.3.1.



(a) Random Across Ensemble with Gate Number Estimation

(b) Random Across Ensemble with Angle Estimation

(c) Adjacent with Gate Number Estimation

(d) Adjacent with Angle Estimation

Figure 7.7: Variation in average error per agent over simulation time. Subplots show the impact of different initial destination randomisation and destination estimation approaches within the Ensemble Kalman Filter.

When considering each of the sub-figures in Figure 7.7, we can draw two comparisons:

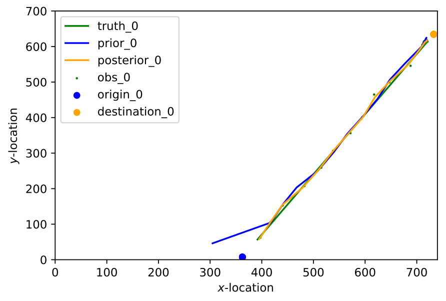
- The impact of the method of allocating initial destinations to agents, i.e. random across ensemble or adjacent to true destination;
- The impact of the type of state augmentation, i.e. estimating gate number or

estimating target angle around the environment.

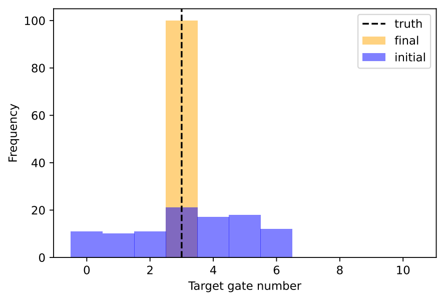
When considering both methods for allocating initial destinations to agents, i.e. scenarios in which initial destinations are allocated randomly across the ensemble in Figures 7.7a and 7.7b and scenarios in which they are allocated adjacent to the true destinations in Figures 7.7c and 7.7d, we see that both prior and posterior average errors present improvements in comparison to the corresponding scenarios when benchmarking. It is noticeable, however, that these improvements are much more noticeable when considering scenarios in which we allocate initial destinations randomly across the ensemble. This is a consequence of the relative poor performance seen in Section 7.3.1 when applying the Ensemble Kalman Filter to update the location of agents in the ensemble-member models. When we have an increased degree of information regarding pedestrians' target destination, the updating of agent locations is much more effective, and therefore the improvement seen when including destination estimation in the EnKF is not as noticeable.

When considering the type of state augmentation used, i.e. scenarios in which we estimate target gate numbers in Figures 7.7a and 7.7c and scenarios in which we estimate target angles in Figures 7.7b and 7.7d, we see that both appear to offer improvements with regards to the accuracy with which they simulate the paths of pedestrians as they cross the concourse environment; over the course of simulation runs, the average error per agent in all cases show improvements when compared to the corresponding scenarios outline in the benchmarking experiments. When comparing the results in Figure 7.7, it may appear that both approaches perform well. It is noticeable, however, that upon repeated simulation runs, we find instances when the angle estimation approaches appears unable to correctly estimate the correct destinations for pedestrians. This results in poor performance when considering the prior average error, but rarely impacts the posterior performance; despite not correctly identifying the correct destination, the inclusion of observations results in accurate estimates of the pedestrians' locations at assimilation time-steps.

Having considered the variation in average error per agent in each of the experimental scenarios outlined in Section 7.3.2, we have established that the application of the EnKF with state augmentation to update destinations is effective in improving the accuracy with which we can simulate pedestrians traversing the concourse environment. This, however, does not necessarily demonstrate that the method is effective in identifying the correct destination for pedestrians over the course of a simulation run. In order to show this, we consider Figures 7.8—7.11. Each figure considers a single member of the pedestrian population in each of the four scenarios explored above. In each case, the relevant figure is divided into two sub-figures. The first of these shows the origin and true destination of the pedestrian, the true path taken by the pedestrian (as determined by the respective agent in the base model), the observations of the true state used for data assimilation and the prior and posterior mean estimates from the ensemble of models in conjunction with the EnKF. The second of these shows the initial distribution of destinations across the ensemble for the given pedestrian, the final estimated distribution of destinations across the ensemble and the true destination; in cases where the gate number is being estimated, the true destination constitutes a single value, whereas in cases where the angle is being estimated the true destination constitutes a range of angles.



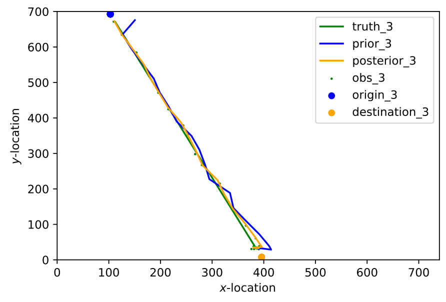
(a) Path taken by agent across environment



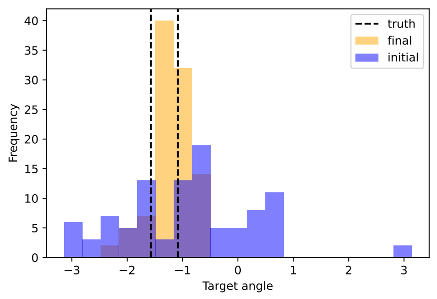
(b) Gates estimated by Ensemble Kalman Filter

Figure 7.8: Gates Estimation with initial destination allocation randomised across ensemble

In Figure 7.8, we see the results pertaining to the scenario in which initial destinations have been allocated randomly across the ensemble, and the state of the EnKF has been augmented to include the gate numbers of each of the pedestrians. When considering the paths shown in Figure 7.8a, we can see that the initial allocation of destinations was incorrect; this is indicated by the first step of the prior line veering off in a direction that would not direct the agents towards the destination (Gate 3) given their starting location (Gate 9). Over the course of the simulation, however, we see that both the prior and posterior lines largely match the line representing the true path of the pedestrian. This indicates that the EnKF has correctly estimated the destination of the pedestrian. This is backed up by Figure 7.8b in which we see the initial and final distributions of destinations gates across the ensemble for the given pedestrian. Given that the initial allocation of destinations was random across the ensemble, we can see that the initial distribution is largely uniform across the selection of valid destination gates; gates 7, 8, 9, and 10 are excluded from the selection of valid gates as the pedestrian departed from the top boundary of the environment (see Table 7.1). Over the course of the simulation run, the filter correctly identifies gate 3 as the pedestrian's destination; this is demonstrated by 3 being the modal value in the final distribution, agreeing with the line representing the true gate number.



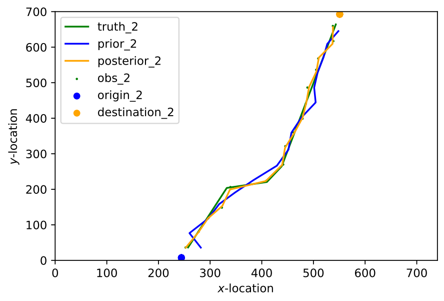
(a) Path taken by agent across environment



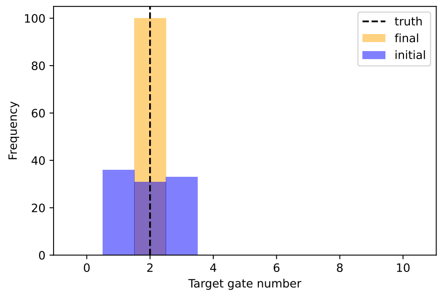
(b) Gates estimated by Ensemble Kalman Filter

Figure 7.9: Angle Estimation with initial destination allocation randomised across ensemble

In Figure 7.9, we see the results of the experiment in which initial destinations are allocated randomly across the ensemble and the filter state is augmented through the addition of a target angle for each pedestrian in the population. Just as in the previous case, the initially high degree of uncertainty in the pedestrian's target destination results in the first step in the prior path heading in a direction that does not align with the true path as seen in Figure 7.9a. As observations are assimilated, however, the path of the agent is corrected and aligns with the true path of the pedestrian to an increasing degree. Just as in the previous case, this is indicative of the filter gaining a correct estimate of the destination towards which the pedestrian is headed within the environment. This is confirmed by inspecting Figure 7.9b. As a consequence of the filter estimating an angle, the true value is a range of values lying between the two dotted lines in this case. The initial distribution of gates being randomly allocated across the ensemble is shown in the figure, with angles pertaining to gates 1 and 2 on the top boundary being excluded. By the time that the pedestrian has completed its journey across the environment, the filter has correctly estimated the target destination, with the majority of the angles across the ensemble lying in the range representing gate 8.



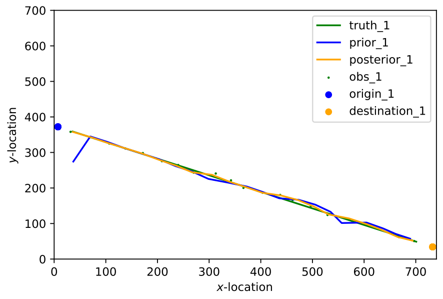
(a) Path taken by agent across environment



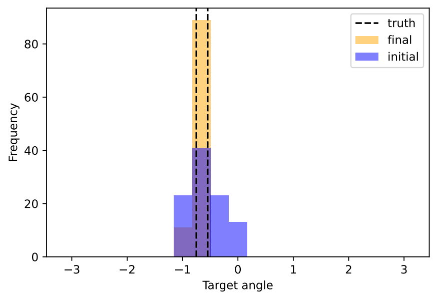
(b) Gates estimated by Ensemble Kalman Filter

Figure 7.10: Gates Estimation with initial destination allocation randomised across gates adjacent to the true gate.

In Figure 7.10, we see the results of the experiment for the scenario in which gates are allocated randomly across the interval between the two gates either side of the true destination gate, and the Ensemble Kalman Filter estimates the destination based on the augmentation of the state to include gate numbers for each of the pedestrians in the population. When considering Figure 7.10a, we see the true pedestrian path, the observations of this path, and the path modelled by the ensemble of models in conjunction with the EnKF. Given the increased level of knowledge regarding the destination of the pedestrian encoded in the initial distribution of destinations across the ensemble, we see that the initial step of the prior path already aligns closely with the true path taken by the pedestrian in the truth-generating base model. Consequently, the prior and posterior paths continue to closely track the true path taken by the pedestrian; this includes tracking the path of the pedestrian as it moves around the central clock obstacle. This increased level of knowledge is reflected in Figure 7.10b, in which we can see the initial distribution of target gate numbers across the ensemble is spread between gates 1, 2 and 3; the true destination is gate 2. Over the course of the simulation run, the EnKF correctly identifies that the pedestrian's destination is gate 2, with this being the value arrived at for the agent in each of the ensemble member models.



(a) Path taken by agent across environment



(b) Gates estimated by Ensemble Kalman Filter

Figure 7.11: Angle Estimation with initial destination allocation randomised across gates adjacent to the true gate.

In Figure 7.11, we see the results pertaining to the scenario in which the initial distribution of destinations is undertaken based on the Adjacent approach outlined in Section 7.1.1, and the EnKF's state vectors are augmented to include target angles for each pedestrian in the population. Although the Adjacent destination allocation approach has been used in this case, there is some degree of deviation between the first step in the prior path and the true path as seen in Figure 7.11a. This is, however, quickly corrected through data assimilation, with the corresponding step in the posterior path closely aligning with the true path. This continues over the course of the simulation run, with the paths converging on the true path as the target destination identified by the Ensemble Kalman Filter converges on the correct range of angles. In Figure 7.9b, we see that the initial distribution of destination angles is spread relatively uniformly across the angles pertaining to the true gate and each adjacent gate on either side. Over the course of the simulation run, the majority of ensemble-member models converge on angles that point to locations along the lower boundary that lie within the correct destination gate — gate 6.

7.4 Concluding Remarks

Just as with the previous chapter this chapter has focussed on the application of the EnKF to the `StationSim_GCS` model of pedestrian motion around the concourse of

Grand Central Station in New York. However, where Chapter 6 focussed solely on the exercise of using the EnKF to update the locations of pedestrians in the environment (having been provided with perfect information regarding the pedestrian destinations), this chapter places greater responsibility on the filter. In this chapter, it is assumed that we do not have perfect knowledge of pedestrian latent variables — in this case pedestrians' destinations — and tasks the EnKF with inferring the unobserved variables whilst still updating the estimates of the observed variables, i.e. pedestrian locations, when provided with observations.

The relaxation of the assumption of perfect knowledge regarding pedestrian latent variables presents some new challenges that have not been faced in previous experiments, as does the tasking of the EnKF with the estimation of such variables. These raise the questions of how we go about encoding prior knowledge regarding pedestrian destinations, and how we modify the EnKF to produce estimates of these variables. Section 7.1 seeks to present prospective solutions to both of these questions.

Experiments in this chapter were undertaken on two fronts. The first of these focused on benchmarking, showing the impact of a lack of knowledge regarding latent variables, i.e. pedestrian destinations, both with and without data assimilation; the second these focused on implementing the proposed solutions outlined in Section 7.1. When undertaking benchmarking, it was found that when a complete lack of knowledge was encoded in the ensemble, the model ensemble performed very poorly, resulting in increasingly large average errors between the ensemble mean and the base model; this was reflected when applying the Uniform Across Ensemble approach and the Random Across Ensemble approach. When considering the scenario in which we encoded some prior knowledge (i.e. the Adjacent approach), it was found that the ensemble of models performed much better; this was, however, still worse than the scenario in which we have perfect knowledge of pedestrian destinations seen in the previous chapter. Subsequently, the EnKF was run with each of the three approaches to representing a lack of knowledge regarding the pedestrian destinations, but this time allowing the filter to update the estimated locations of the pedestrians within the ensemble of models. The improvements seen when using the Random Across Ensemble approach were much greater than those seen when using the Uniform Across Ensemble approach. This was attributed to the greater degree of variation between ensemble member models observed when using the Random Across Ensemble approach which subsequently allowed the EnKF to make greater alterations to the ensemble state. When considering the scenario in which the Adjacent approach was used, the prior and posterior errors reflected that the ensemble of models was relatively effective in simulating the system of pedestrians as a result of the increased degree of knowledge regarding pedestrian destinations.

Having established a benchmark of the performance with which the Ensemble Kalman Filter could simulate the system when presented with uncertainty regarding pedestrian destinations, the final experiment aimed to demonstrate that the filter could effectively simulate the system and identify correct estimates of the destinations when allowed to update the destinations in some way. Having established in the previous experiment that the Uniform Across Ensemble approach to destination allocation did not lend itself to state updates, it was discarded as an approach, and so the final experiment aimed to explore the impact of the two remaining approaches (Random Across Ensemble and Adjacent) in conjunction with the two approaches to state augmentation (Gate Number estimation and Angle estimation). This resulted in the consideration of four cases. In each of the four cases, the filter was found to be effective at both updating estimates of pedestrian locations and estimating target destinations for the pedestrians within the system. This was established by, once again considering the average error per agent, as well as considering the changes made by the filter to the distribution of destinations across the ensemble for the agents representing a given pedestrian. It was, however, found that the gate number estimation approach performed more reliably with respect to correctly identifying destinations for the pedestrians.

Whilst this chapter has largely focused on the use of the EnKF to estimate pedestrian destinations, this has ramifications for the inference of latent attributes in Agent-Based Models more generally, particularly those that are categorical. When using Agent-Based Models, there may be features of agents that are neither known with certainty from the outset of the simulation, nor observed over the course of the simulation run. In scenarios where such systems are run in an "off-line" manner, this may not be a significant problem as we are more likely to consider average behaviours; when considering "on-line" scenarios, however, this may have a much larger impact.

Having demonstrated the efficacy with which the EnKF can simulate the motion of pedestrians across an environment and estimate unobserved parameters such as destination gates in this chapter, the next chapter goes on to conclude this thesis. This will summarise the work undertaken here, draw comparisons with other relevant pieces of work identified in the Literature Review, and suggest a set of future investigations to deal with limitations of this investigation and further expand on this novel research field.

Chapter 8

Conclusions

In Chapter 1, a set of research objectives were outlined. The objectives were as follows:

1. Review the literature around the simulation of pedestrian dynamics with a particular focus on simulation at close to real-time.
2. Describe the modelling approaches used to simulate pedestrian systems in this investigation.
3. Describe the data assimilation approaches used to update models in this investigation.
4. Apply the Ensemble Kalman Filter to improve estimates of pedestrian locations in simple Agent-Based Models to show that such approaches can be used with systems other than those that are described by systems of differential equations.
5. Apply the Ensemble Kalman Filter to improve estimates of pedestrian locations in more realistic Agent-Based Models to show that such approaches are also effective when to systems that more closely reflect reality.
6. Apply the Ensemble Kalman Filter to improve estimates of both observed and unobserved variables in an Agent-Based Model to show that such approaches are effective in simultaneously performing state and parameter estimation for pedestrian systems.

The overarching theme of these objectives was the development of a method for the integration of real-time observations into an Agent-Based Model of pedestrian motion such that the model could be run in an 'online' manner. This investigation sits within a relatively new field of research, with few other prior works seeking to address the problems herein; as such, the results of this investigation represent a novel piece of work. Over the course of this investigation, the Ensemble Kalman Filter (EnKF) method of data assimilation was found to be effective in integrating real-time observations into an Agent-Based Model of pedestrian motion, both in the case of a toy model (Chapter 5) and a model which is grounded in a real-world setting (Chapter 6 and 7).

This chapter, acting as a conclusion to this investigation, shall summarise the results of the various efforts that have been undertaken to achieve these objectives. This will consist of a section summarising the results of the chapters, an evaluation of these results and some recommendations for future work which should follow this investigation.

8.1 Summary of Results

This section aims to summarise the results of this investigation, and in doing so addresses Objectives 4, 5 and 6.

The first of these objectives — Objective 4 — was addressed in Chapter 5. In this chapter, we sought to show that the EnKF was effective in improving estimates of pedestrian locations over time in a simple Agent-Based Model. This involved undertaking a series of experiments. The first of these aimed to establish a benchmark for performance when an ensemble of models was used to simulate pedestrians crossing the environment without the assimilation of data. This was carried out by considering the average error per pedestrian across the agents in the average model state across the ensemble. This experiment showed that, whilst the average error per pedestrian was low at the beginning and end of the simulation run, an increase in error was observed between these times. This resulted from the randomness of agent-agent interactions whereby agents attempting to proceed from their origin to their destination who were obstructed by other agents would attempt to bypass them by sidestepping perpendicular to their direction of travel; the direction of the sidestep is a binomial trial-like choice, with each direction carrying equal probability. The number of these sidesteps (characterised as collisions in Chapter 5) was seen to grown disproportionately as the population size grew. Following this, the EnKF was introduced. For the same ensemble size and population size, the filter was found to improve the accuracy with which the ensemble of models simulated the underlying system. Finally, the impact of different filter parameters was considered. The parameters in question were the ensemble size, the assimilation period and the standard deviation associated with the error in the observations. As was observed in previous investigations using the EnKF, it was found that:

- An increase in the ensemble size results in a reduction in the error;
- An increase in the assimilation period results in an increased in the error;
- An increase in the observation error standard deviation results in an increase in the error.

Following this, the EnKF was applied to the `StationSim_GCS` model. This model sought to expand upon the toy model used in the previous chapter which was somewhat limited in the types of agent interactions that it incorporated, the way in which it separated entrances and exits and the exclusion of any environmental obstacles. Just as with the previous model, a set of benchmarks were established to show how an ensemble of models performed without any data assimilation. The filter was then applied to this model. In applying the filter to the model, two additional challenges were considered, focussing on how outlier simulations impacted summary statistics. Outliers were seen to occur with respect to both error magnitude and time taken for simulations to finish running. As a consequence, the approach to processing results was updated to account for these problems, choosing to take the median error instead of the mean error (handling error outliers) and finishing simulations when 90% of models have completed (handling time outliers). When applying the filter to `StationSim_GCS` with these adjustments, it was found that it was effective in improving the accuracy with which the ensemble of models simulated the system, outperforming the benchmarking ensemble of models.

The final results chapter went on to highlight a key problem with the previous two results chapters — namely that we had assumed perfect knowledge of latent pedestrian attributes — and sought to provide solutions to this problem. In reality, we seldom have perfect knowledge regarding such attributes (such as pedestrian destination). We therefore sought to use the EnKF to infer such latent variables. This introduced two challenges.

The first of these raised the question of how we go about encoding prior knowledge regarding such unobserved quantities in our ensemble of models. It was proposed that we could encode the uncertainty in our knowledge by using different schemes to allocated initial destinations to agents across the ensemble of models, either encoding a complete lack of knowledge by choosing destinations using a uniform random distribution across all gates in the environment, or encoding an limited knowledge of pedestrians destinations by using a random uniform distribution across the true destination gate and a limited number of adjacent gates.

The second of these raise the question of how we go about using the filter to infer unobserved quantities, particularly given that the quantities are discrete/categorical variables in the case of pedestrian destinations. It was proposed that we could use the state augmentation approach, whereby unobserved parameters are included in the state vector. To achieve this, two approaches for encoding destinations were proposed.

The first of these opted to simply add the number of the gate to which each pedestrian was headed to the state vector. The second opted to coerce the problem from one of a problem of inferring categorical variables to one of inferring continuous variables — something that is more common in typical data assimilation fields of application; this was achieved by considering the angle between the centre of the environment and the point on the edge of the environment towards which a pedestrian was headed.

In each of the cases, the EnKF was found to be effective in both improving the location estimates of pedestrians within the environment and in inferring the pedestrian destinations, showing the suitability of the method of the inference of latent parameters in Agent-Based Models.

8.2 Evaluation

Over the course of this investigation, the EnKF has been shown to be effective in improving the accuracy with which an ensemble of models simulate a pedestrian system, both when considering the fictitious toy model and when considering the mode realistic `StationSim_GCS`. This is largely in agreement with the limited number of investigations which have sought to apply other data assimilation schemes to similar scenarios. Of these other investigations, some have sought to apply to Particle Filter (Malleson et al., 2020; Ternes et al., 2021) and others have sought to apply the Unscented Kalman Filter (Clay et al., 2020, 2021).

One of the issues noted when reviewing the literature in Chapter 2 and outlining the data assimilation methods in Chapter 4 was the ways in which the number of ensemble-member models/particles varied across the Particle Filter and EnKF. Typically, it has been observed that when dealing with systems in which the transformations are linear and in which errors are Gaussian-distributed, the EnKF requires smaller ensemble sizes to achieve the same degree of error. Whilst this investigation has not explicitly explored whether these conditions have been met, it is noticeable that, when comparing the results of Chapter 5 with the results produced with the Particle Filter by Malleson et al. (2020), the number of particles used in the Particle Filter investigation is much larger than the ensemble sizes used in this investigation; whilst the EnKF used in Chapter 5 uses ensemble sizes of < 100 models, the Particle Filter investigation uses up to $10,000$ particles to produce results with comparable population sizes. This disparity in ensemble sizes was also noticeable when comparing the investigation by Ternes et al.

(2021) which sought to apply the Particle Filter to `StationSim_GCS` whilst also handling latent pedestrian parameters as in Chapter 7, with the Particle Filter investigation using $5,000$ particles and our EnKF using 100 ensemble-member models. In the case of the Particle Filter investigation, however, population sizes were much larger, and the filter was tasked with handling uncertainty in both the pedestrian speeds and destinations whilst this investigations only considered a single latent variable for each pedestrian — its destination.

One of the shortcomings of this investigation is the data used. Whilst real-world observations have been used in the model calibration process outlined in Chapter 3, the data used in the assimilation process have been synthetic, i.e. a base model has been used to generate a ground truth, and synthetic observations have been produced by adding Gaussian noise to these model states. The pedestrian trajectories across the environment in the model are largely linear except when pedestrians employ avoidance (i.e. sidestepping); this is not always the case in the real-world. As Ternes et al. (2021) notes, when working with real-world observations, we are likely to find some trajectories which are distinctly non-linear.

A further shortcoming is the assumption of perfect coverage in observations at each assimilation time-step. In this investigation, the observations generated have been at an individual-level, and have covered all of the pedestrians being modelled; this may not always be the case. A more common scenario is one in which we have have intermittent observations of the locations of a portion of the population (e.g. mobile phone data) or aggregated observations (e.g. pedestrian count data generate by cameras). The former case has not been considered here, but has been considered in investigations using the related Unscented Kalman Filter (Clay et al., 2020) which found the filter to be successful in handling the lack of information.

When exploring the use of different summary statistics for calculating errors in Chapter 6, it was noted that for that purpose, taking the *median* instead of the *mean* was a more appropriate summary statistic as it helped to alleviate issues with ensemble-member models which acted as outliers in the calculation of the average error. In a similar vein, it may have been of value to explore the use of different summary statistics when reducing an ensemble of models down to a single representative state. In this investigation, we have defaulted to taking the mean of the ensemble-member models. It is possible, however, that this could produced non-viable trajectories; when

when applying the EnKF to `stationsim_gcs`, this may produced trajectories that pass through the obstacle in the centre of the environment. It may, therefore, be more appropriate to use a different summary statistic such as the median to ensure that the representative average state that is produced is, in fact, a viable state which is found in at least one of the ensemble-member models.

A further issue regarding the summarisation of ensemble-member models into a single representative state may also occur in the process of inferring the pedestrians' destinations in Chapter 7. In this scenario we were attempting to infer locations around a (topologically) circular environment. The approaches used, however, may not sufficiently handle the circular nature of the environment. We may consider a scenario where we have a single pedestrians positioned near the centre of the environment which is represented by an EnKF containing two models which seeks to infer the pedestrian's destination by estimating the angle of the destination. One of the ensemble-member models may estimate that the target destination is at an angle of $\theta = \frac{-3\pi}{4}$, whilst the other may estimate the angle to be $\theta = \frac{3\pi}{4}$. In such a scenario the ensemble average would lead the pedestrian to a destination located at $\theta = 0$ where it is more likely that an appropriate average destination would be $\theta = \pi$. One way of facilitating these types of calculations might be the use of circular statistics (Jammalamadaka and Sengupta, 2001).

Finally, it should be noted that this investigation has largely dealt with the two sources of uncertainty, i.e. uncertainty resulting from agent-agent interactions (Chapter 6) and uncertainty resulting from lack of knowledge regarding agent destinations (Chapter 7), in isolation of each other. This is evidenced by the relatively small population sizes employed in Chapter 7.

With these points in mind, the following section seeks to outline a programme of suggested future work which may seek to expand on the present investigation.

8.3 Recommendations for Future Work

Having offered some critique of this investigation in the previous section, this section seeks to outline a ideas for future work which might alleviate some of these issues, and may expand upon what has been presented herein.

In the previous section, we mention the limitation of not using real-world data in the data assimilation process, and only in the calibration process. One of the primary recommendations for future work would, therefore, be to expand this suite of work to make use of real-world trace data when performing assimilation. The introduction of real-world data into the process may be undertaken incrementally, starting with basic examples and expanding to situations which exhibit greater degrees of complexity as in this investigation. As noted by Ternes et al. (2021), this brings with it substantial challenges when we encounter pedestrians who take paths between their origin and destination which are particularly non-linear. Such a set of experiments will likely draw into question the validity of the model being used — does it truly capture the behaviour of the pedestrians being simulated? In order to address this, we may wish to undertake a more data-driven model development process.

In such a scenario, we must also address the question of how we go about assessing filter performance; in this investigation, error was calculated by considering the ensemble mean and the ground truth generated from a base model, but when dealing with real-world traces we would not have a base model against which to compare. One solution may be to consider the observations as the ground truth and generate data to assimilate into the ensemble by adding noise to them (just as has been done in this investigation).

Another way in which this investigation could be expanded upon is through a more in-depth exploration of the different ways in which we encode our prior knowledge regarding latent variables. In Chapter 7, we considered a set of approaches which either assumed no knowledge of the pedestrians' destinations (Random Across Ensemble) or perfect knowledge with some limited random noise attached (Adjacent). This could be expanded upon one two fronts. Given that this investigation focused on the inference of categorical latent variables, one extension of the investigation could be task the filter with estimating continuous unobserved variables too, as was undertaken by Ternes et al. (2021); it is expected, however, that the inference of a continuous variable being a problem to which data assimilation schemes are more commonly applied would mean that this would not be particularly challenging. The next extension may seek to explore the way in which a filter's ability to correctly infer latent variables depends on the way in which the uncertainty in the prior knowledge is represented. In this investigation two approaches for encoding prior uncertainty were presented — one in which no prior knowledge was assumed, and one in which near-perfect knowledge was assumed. Neither of these approaches have considered a way of encoding knowledge which was derived from empirical data analysis, which is likely to result in a scenario which lies somewhere between the two approaches used in this investigation. Such a representation of uncertainty in knowledge could be arrived at by analysing the pedestrian trace data used in the calibration phase to devise an origin-destination matrix which would determine the random distribution used to draw pedestrian destinations base on the gate through which they enter the system.

A further way in which this investigation could be extended is through the use of other approaches to estimating latent variables. Whilst there may exist other approaches to using the EnKF to infer latent variables beyond state augmentation (Katzfuss et al., 2016), we may also wish explore approaches which help to infer these variables outside of the filter. If considering the specific scenario presented Chapter 7 in which we wish to infer the pedestrians' destinations, we could consider a very simple approach whereby we fit a straight line to the observations of each pedestrian, project the line forwards (following the procession of time over which the observations were collected), find the point at which this extended line meets the boundary of the environment and identify the nearest gate to this point. This approach, however, would likely not extend well to inference of other latent variables when apply an Agent-Based Model in conjunction with a data assimilation scheme. A more general approach which would likely be more conducive to the inference of categorical latent variables would be to use a Hidden Markov Model (Rabiner and Juang, 1986) — something that was attempted by Rai and Hu (2013). Hidden Markov Models seek to learn about the dynamics of an unobservable system using a set of observable states which are in some way influenced or controlled by the underlying system. In the case of the destination inference problem studied in this investigation, this would involve inferring the destination of the *unobserved* pedestrian destinations based on the *observed* pedestrian locations over time.

A final way in which this investigation could be extended is through direct comparison with other data assimilation methods. So far, each of the EnKF, Unscented Kalman Filter (Clay et al., 2020, 2021) and the Particle Filter (Malleson et al., 2020; Ternes et al., 2021) have been applied to similar pedestrian Agent-Based models. Their comparative strengths and weaknesses are well documented; however, the situations in which these are meaningful has not been explored in the setting of Agent-Based Models of pedestrian systems. As an example, we may consider the assumption of Gaussian error distributions in Kalman Filter-based approaches — when this is satisfied such filters should be optimal and as such should rival the performance of a Particle Filter with the requirement of less computational cost. In order to test this, we would need to assess the extent to which error distributions in our ensemble of Agent-Based Models are Gaussian, and if they are not also explore the ramifications for the trade-off between filter performance and ensemble size.

8.4 Concluding Remarks

This chapter has sought to provide an overview of the results of the investigation presented in this thesis, noting the limitations and offering some ideas for future work. Over the course of the investigation, we have presented a novel approach to running Agent-Based Models of pedestrian systems in an "on-line" manner. Running such models at close to real-time has previously been a significant challenge due to the lack of established methodology for incorporating streamed data into models. This investigation offers a potential avenue by which to solve this problem in the form of the EnKF data assimilation scheme. The application of data assimilation methods to Agent-Based Models of pedestrian systems is still a very you and expanding field of research; this investigation has, therefore, sought to address some of the open problems which are faced by those in the field. Whilst this investigation has been largely successful, there still remain challenges that need to be addressed. It is hoped that, with the resolution of such problems, "on-line" Agent-Based Models which receive streamed data from pedestrian systems may be operationalised such that they may be used by practitioners and policy makers to offer services in a more data-driven manner.

Appendix A

Supplementary Calibration Figures

This appendix contains all of the extra figures relating to the calibration of `StationSim_GCS`. This complements the analysis undertaken in Chapter 3 (specifically Section 3.2.2).

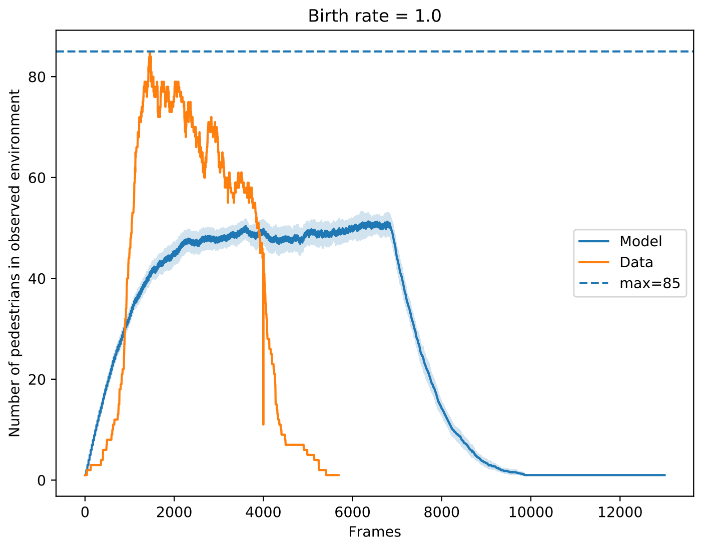


Figure A.1: Variation of number of agents in the system over time for $\lambda = 1.0$

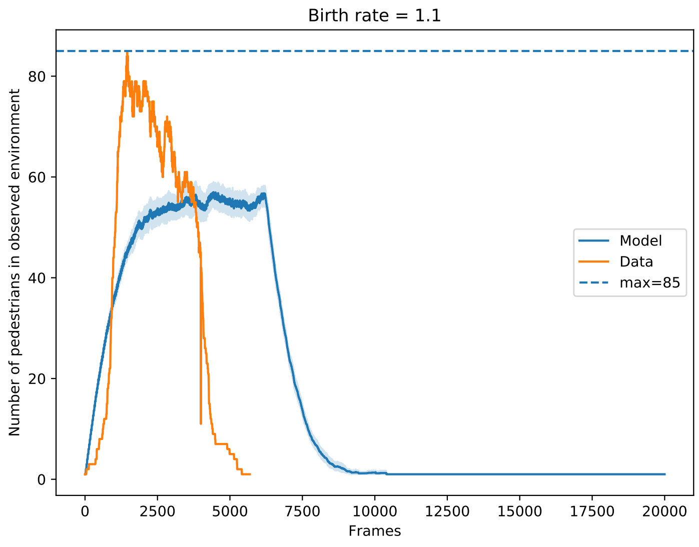


Figure A.2: Variation of number of agents in the system over time for $\lambda = 1.1$

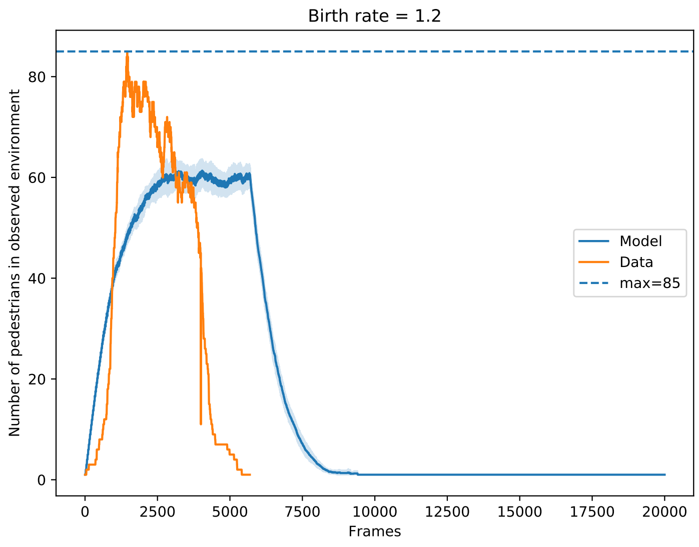


Figure A.3: Variation of number of agents in the system over time for $\lambda = 1.2$

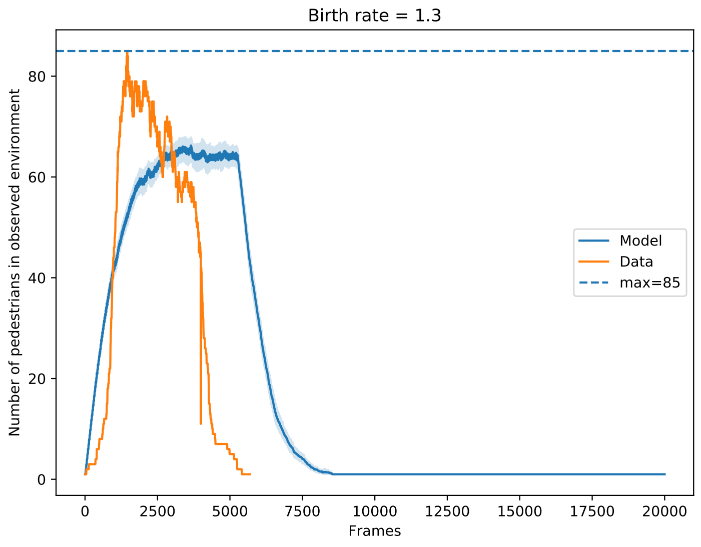


Figure A.4: Variation of number of agents in the system over time for $\lambda = 1.3$

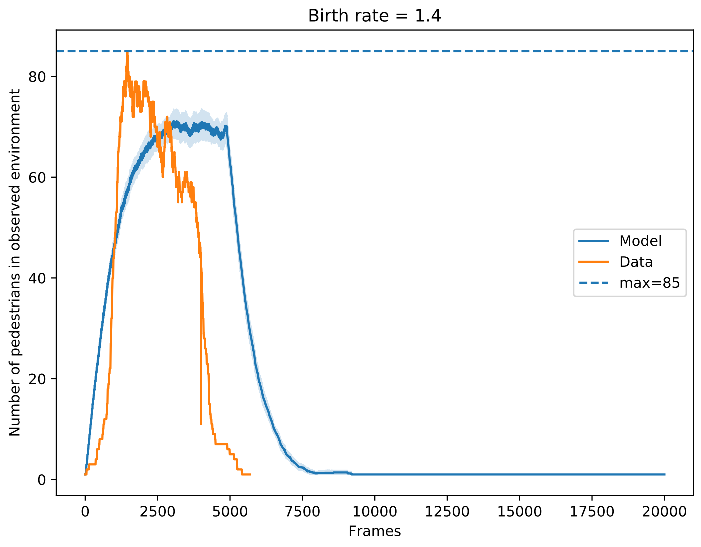


Figure A.5: Variation of number of agents in the system over time for $\lambda = 1.4$

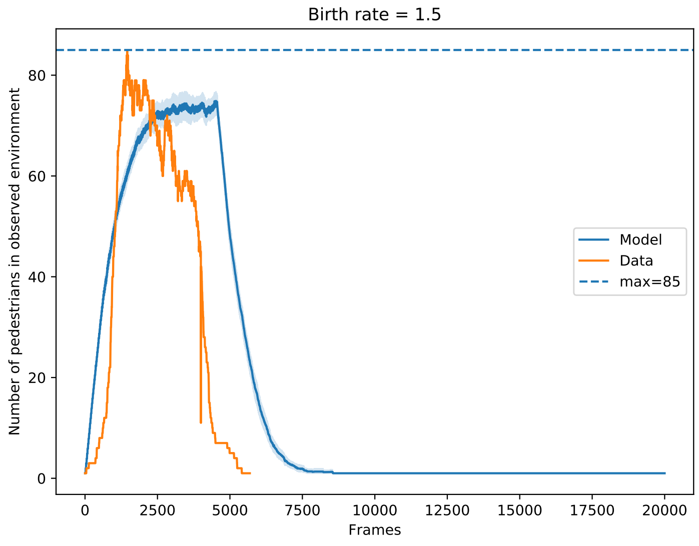


Figure A.6: Variation of number of agents in the system over time for $\lambda = 1.5$

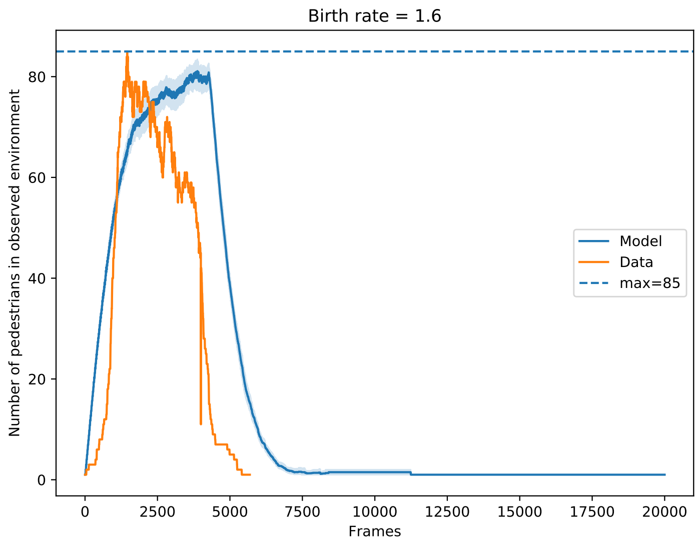


Figure A.7: Variation of number of agents in the system over time for $\lambda = 1.6$

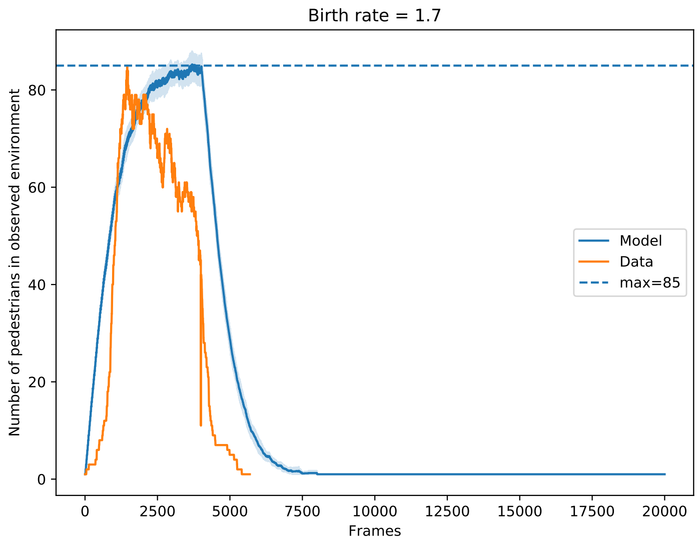


Figure A.8: Variation of number of agents in the system over time for $\lambda = 1.7$

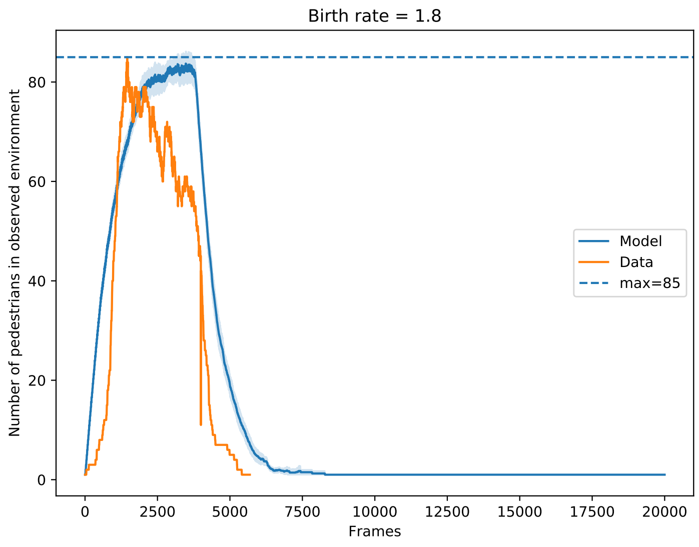


Figure A.9: Variation of number of agents in the system over time for $\lambda = 1.8$

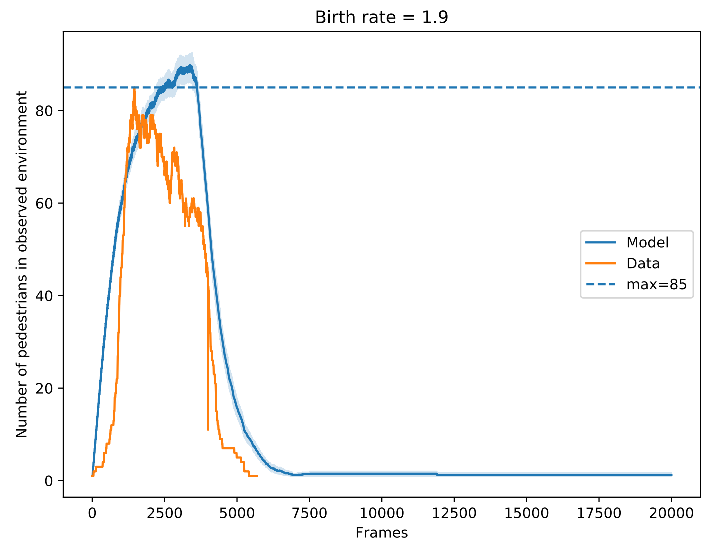


Figure A.10: Variation of number of agents in the system over time for $\lambda = 1.9$

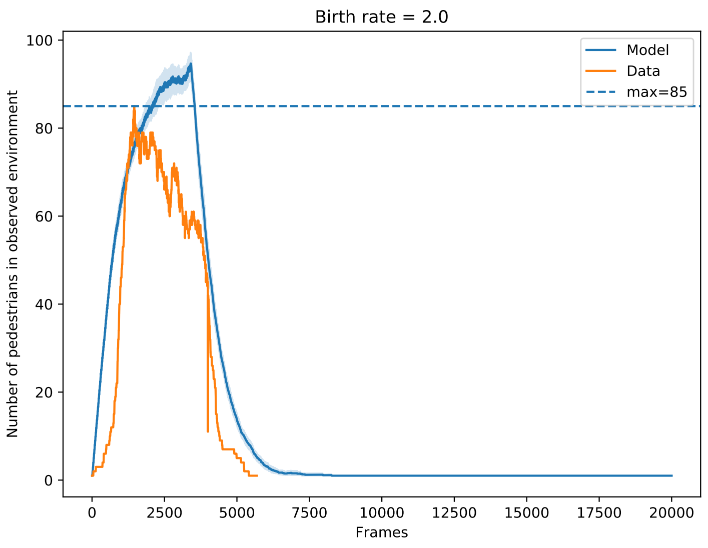


Figure A.11: Variation of number of agents in the system over time for $\lambda = 2.0$

Appendix B

Model Subprocesses

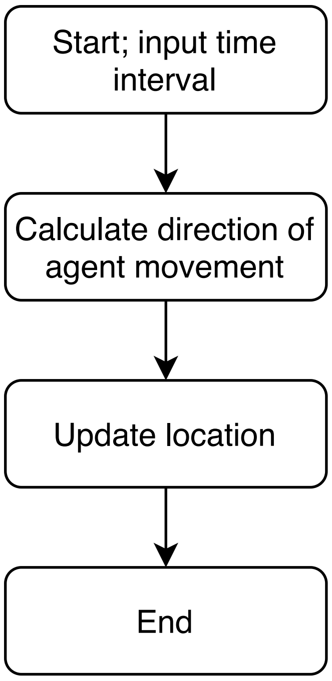
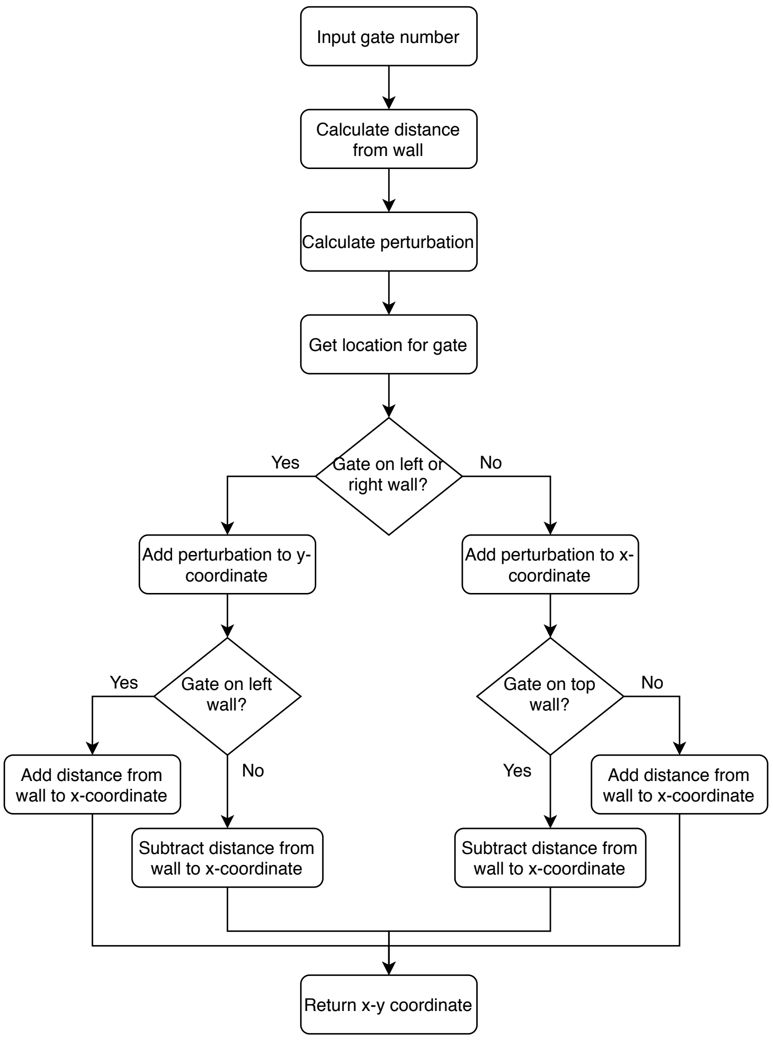

Start; input time interval
Calculate direction of agent movement
Update location
End

Figure B.1: Flow diagram showing `StationSim_GCS` agent movement process.



Input gate number
Calculate distance from wall
Calculate perturbation
Get location for gate
Gate on left or right wall?
Yes
No
Add perturbation to y-coordinate
Add perturbation to x-coordinate
Gate on left wall?
Yes
No
Gate on top wall?
No
Yes
Add distance from wall to x-coordinate
Subtract distance from wall to x-coordinate
Subtract distance from wall to x-coordinate
Add distance from wall to x-coordinate
Return x-y coordinate

Figure B.2: Flow diagram showing `StationSim_GCS` agent location allocation process.

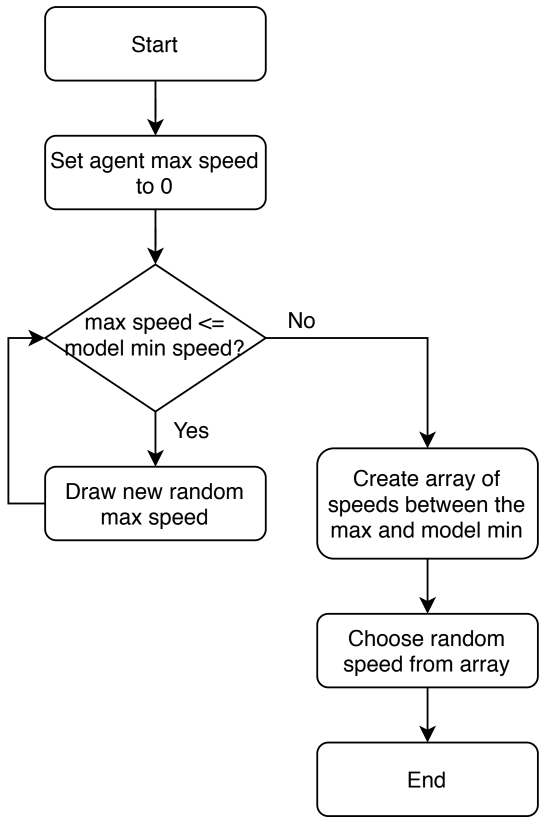
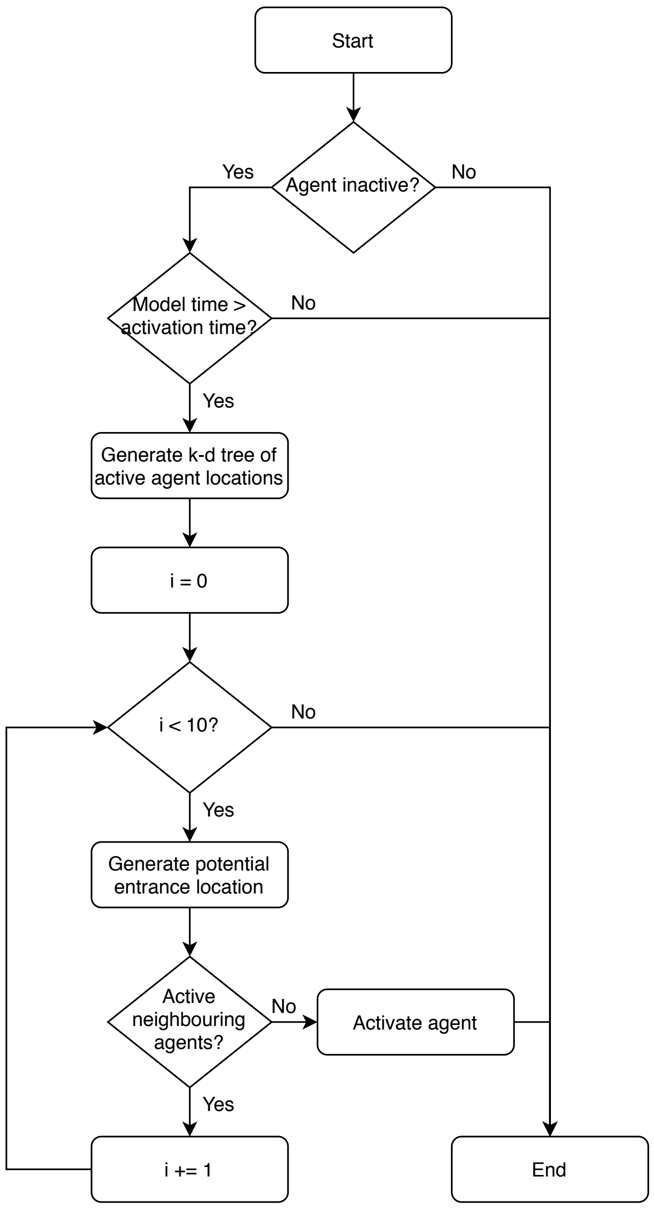

Start
Set agent max speed to 0
max speed <= model min speed?
No
Yes
Draw new random max speed
Create array of speeds between the max and model min
Choose random speed from array
End

Figure B.3: Flow diagram showing `StationSim_GCS` agent speed allocation process.



Start
Agent inactive?
Yes
No
Model time > activation time?
No
Yes
Generate k-d tree of active agent locations
i = 0
i < 10?
No
Yes
Generate potential entrance location
Active neighbouring agents?
No
Activate agent
Yes
i += 1
End

Figure B.4: Flow diagram showing `StationSim_GCS` agent activation checking process.

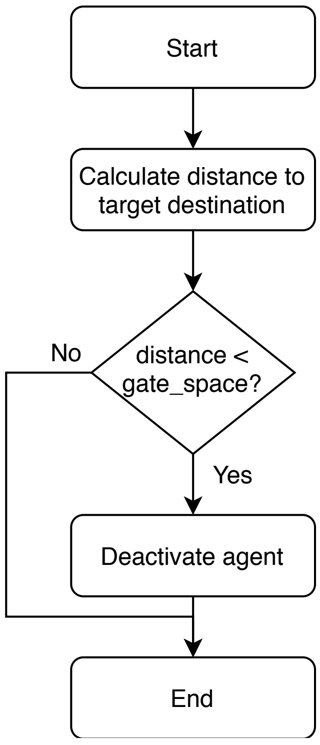


Figure B.5: Flow diagram showing `StationSim_GCS` agent deactivation checking process.

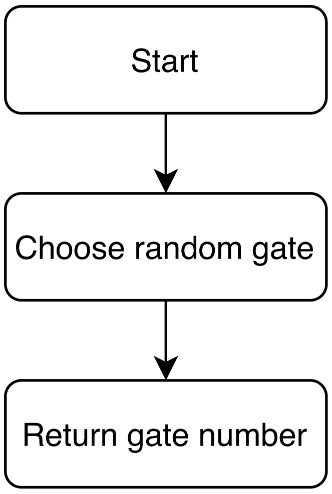


Figure B.6: Flow diagram showing `StationSim_GCS` agent entrance-choice process.

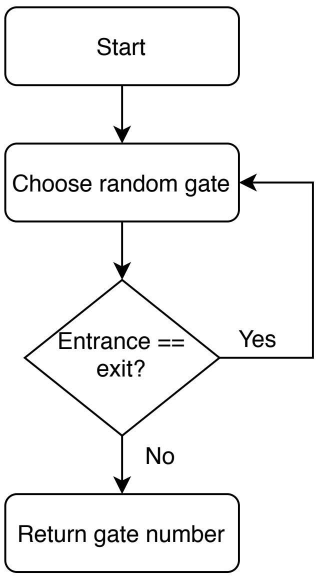


Figure B.7: Flow diagram showing `StationSim_GCS` agent exit-choice process.

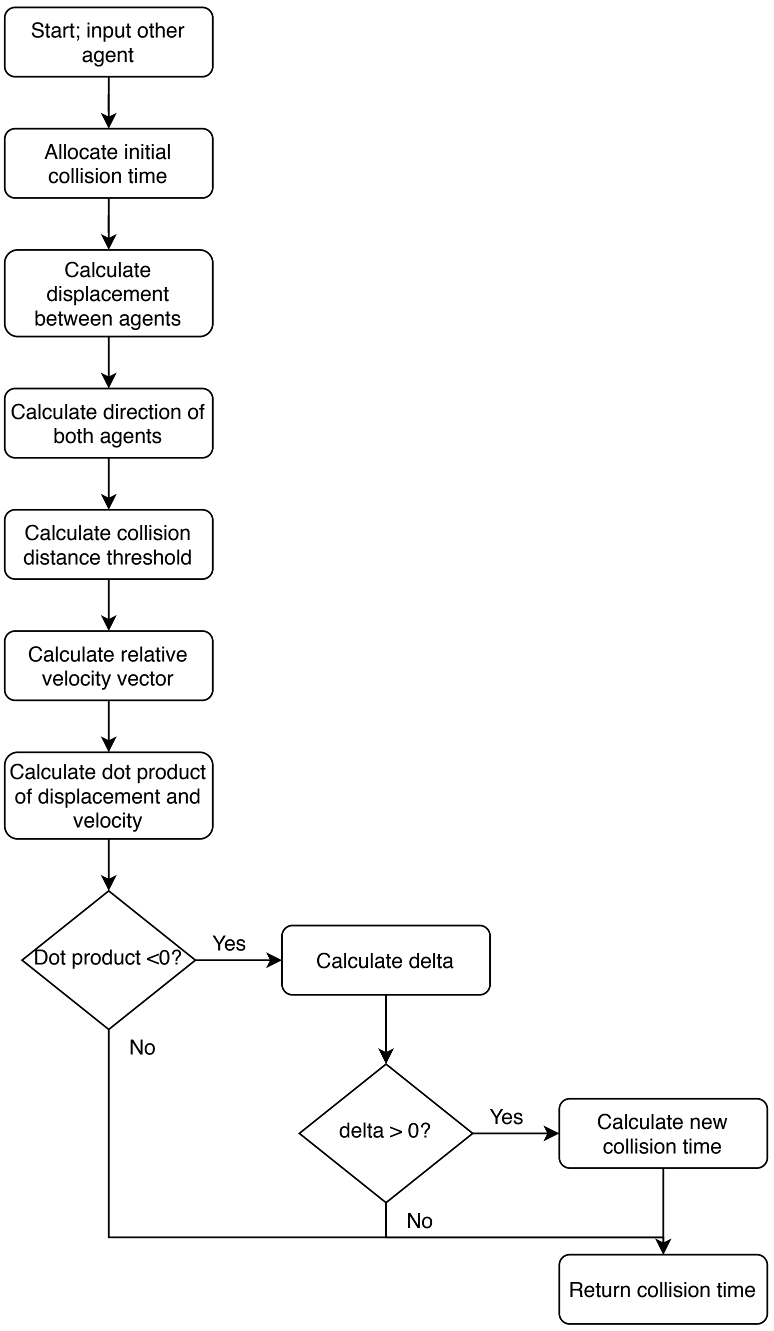

Start; input other agent
Allocate initial collision time
Calculate displacement between agents
Calculate direction of both agents
Calculate collision distance threshold
Calculate relative velocity vector
Calculate dot product of displacement and velocity
Dot product <0?
Yes
No
Calculate delta
delta > 0?
Yes
No
Calculate new collision time
Return collision time

Figure B.8: Flow diagram showing `StationSim_GCS` agent-agent collision process.

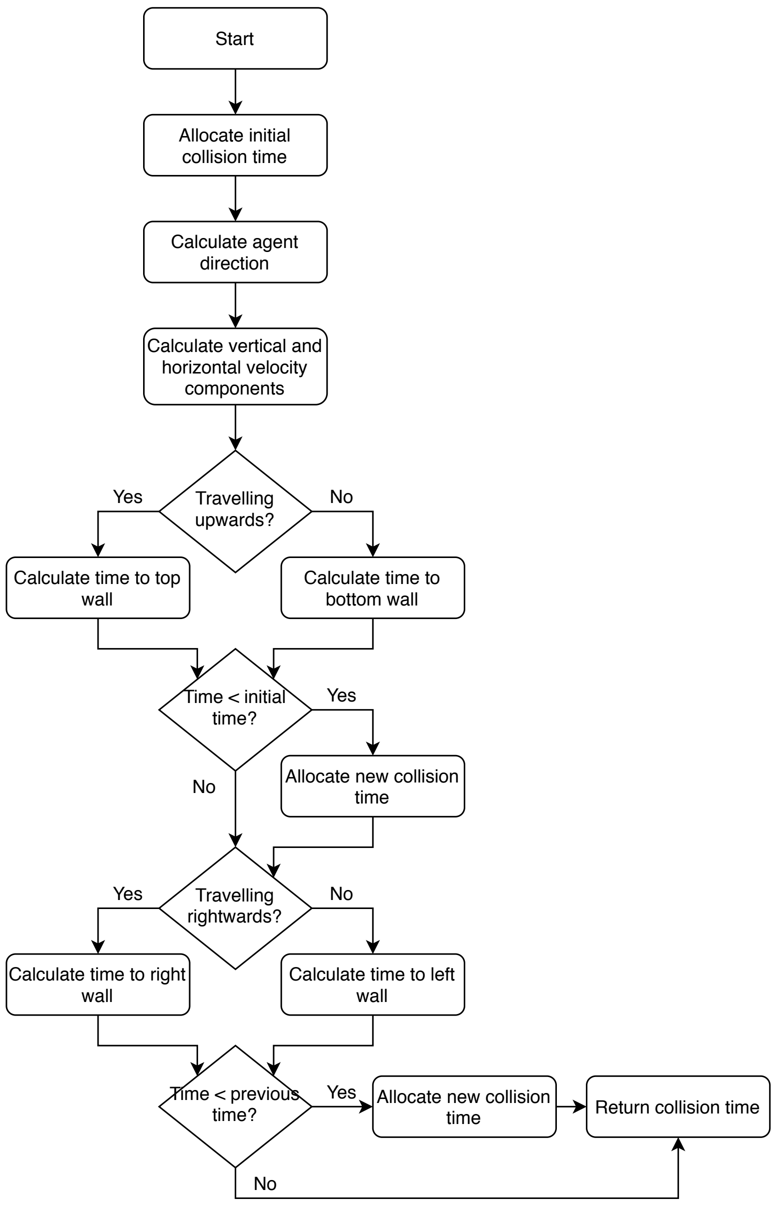

Start
Allocate initial collision time
Calculate agent direction
Calculate vertical and horizontal velocity components
Travelling upwards?
Yes
No
Calculate time to top wall
Calculate time to bottom wall
Time < initial time?
Yes
No
Allocate new collision time
Travelling rightwards?
Yes
No
Calculate time to right wall
Calculate time to left wall
Time < previous time?
Yes
No
Allocate new collision time
Return collision time

Figure B.9: Flow diagram showing `StationSim_GCS` agent-wall collision process.

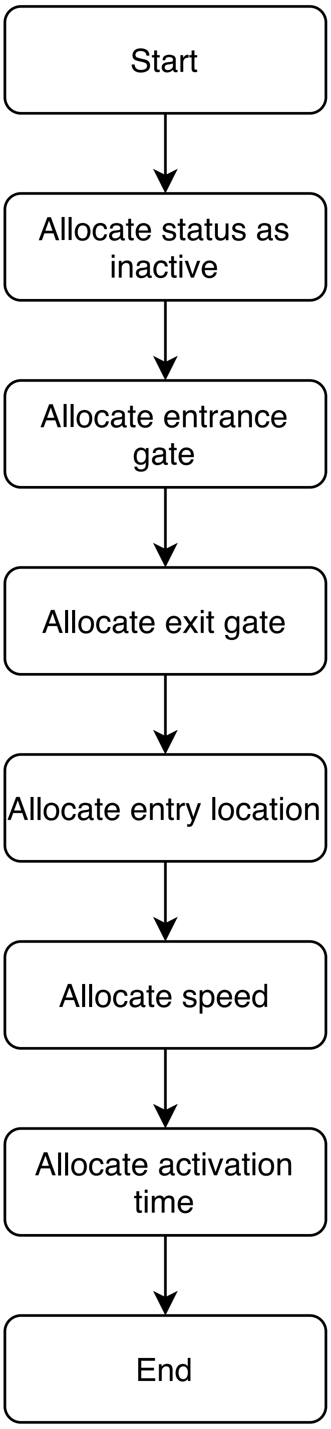


Figure B.10: Flow diagram showing `StationSim_GCS` agent initialisation process.

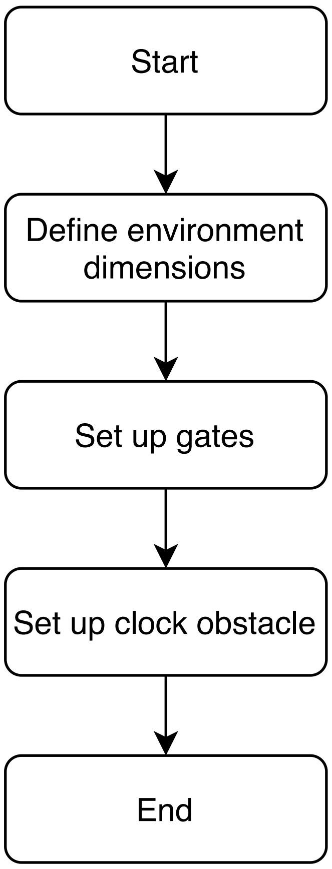


Figure B.11: Flow diagram showing `StationSim_GCS` station set-up process.

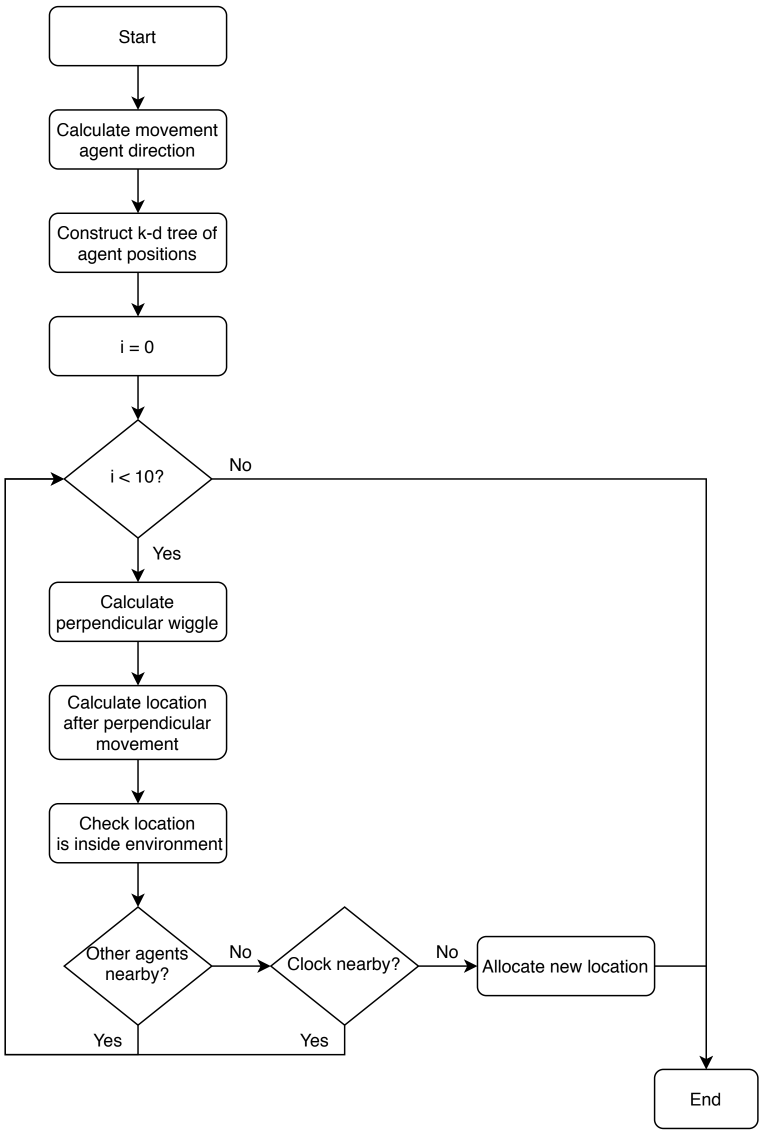

Start
Calculate movement agent direction
Construct k-d tree of agent positions
i = 0
i < 10?
No
Yes
Calculate perpendicular wiggle
Calculate location after perpendicular movement
Check location is inside environment
Other agents nearby?
No
Yes
Clock nearby?
No
Yes
Allocate new location
End

Figure B.12: Flow diagram showing `StationSim_GCS` wiggle process.

Bibliography

Aggleton, R., Ardila-Perez, L., Ball, F., Balzer, M., Boudoul, G., Brooke, J., Caselle, M., Calligaris, L., Cieri, D., Clement, E., et al. (2017). An fpga based track finder for the l1 trigger of the cms experiment at the high luminosity lhc. *Journal of Instrumentation*, 12(12):P12019.

Amaral, L. A. and Ottino, J. M. (2004). Complex networks. *The European physical journal B*, 38(2):147–162.

An, L., Grimm, V., Sullivan, A., Turner II, B., Malleson, N., Heppenstall, A., Vincenot, C., Robinson, D., Ye, X., Liu, J., et al. (2021). Challenges, tasks, and opportunities in modeling agent-based complex systems. *Ecological Modelling*, 457:109685.

An, L., Grimm, V., and Turner II, B. L. (2020). Meeting grand challenges in agent-based models. *Journal of Artificial Societies and Social Simulation*, 23(1).

Andrews, C. J., Yi, D., Krogmann, U., Senick, J. A., and Wener, R. E. (2011). Designing buildings for real occupants: An agent-based approach. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(6):1077–1091.

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188.

Auger, F., Hilairet, M., Guerrero, J. M., Monmasson, E., Orlowska-Kowalska, T., and Katsura, S. (2013). Industrial applications of the kalman filter: A review. *IEEE Transactions on Industrial Electronics*, 60(12):5458–5471.

Badham, J., Chattoe-Brown, E., Gilbert, N., Chalabi, Z., Kee, F., and Hunter, R. F.

(2018). Developing agent-based models of complex health behaviour. *Health & place*, 54:170–177.

Bandini, S., Manzoni, S., and Vizzari, G. (2009). Agent based modeling and simulation: an informatics perspective. *Journal of Artificial Societies and Social Simulation*, 12(4):4.

Bannister, R. (2017). A review of operational methods of variational and ensemble-variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 143(703):607–633.

Bar-Shalom, Y., Li, X. R., and Kirubarajan, T. (2004). *Estimation with applications to tracking and navigation: theory algorithms and software.* John Wiley & Sons.

Batty, M. (2019). Urban analytics defined.

Batty, M., DeSyllas, J., and Duxbury, E. (2003a). The discrete dynamics of small-scale spatial events: agent-based models of mobility in carnivals and street parades. *International Journal of Geographical Information Science*, 17(7):673–697.

Batty, M., Desyllas, J., and Duxbury, E. (2003b). Safety in numbers? modelling crowds and designing control for the notting hill carnival. *Urban Studies*, 40(8):1573–1590.

Bazghandi, A. (2012). Techniques, advantages and problems of agent based modeling for traffic simulation. *International Journal of Computer Science Issues (IJCSI)*, 9(1):115.

Bazzan, A. L. and Klügl, F. (2014). A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review*, 29(3):375.

Beaumont, M. A. (2010). Approximate bayesian computation in evolution and ecology. *Annual review of ecology, evolution, and systematics*, 41:379–406.

Birks, D. and Davies, T. (2017). Street network structure and crime risk: An agent-based investigation of the encounter and enclosure hypotheses. *Criminology*, 55(4):900–937.

Boeing, G., Batty, M., Jiang, S., and Schweitzer, L. (2021). Urban analytics: History, trajectory, and critique. *Trajectory, and Critique (May 14, 2021).*

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the national academy of sciences*, 99(suppl 3):7280–7287.

Burstedde, C., Klauck, K., Schadschneider, A., and Zittartz, J. (2001). Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295(3-4):507–525.

Carrassi, A., Bocquet, M., Bertino, L., and Evensen, G. (2018). Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535.

Chattaraj, U., Seyfried, A., and Chakroborty, P. (2009). Comparison of pedestrian fundamental diagram across cultures. *Advances in complex systems*, 12(03):393–405.

Chen, H., Cheng, T., and Wise, S. (2017). Developing an online cooperative police patrol routing strategy. *Computers, Environment and Urban Systems*, 62:19–29.

Chen, M., Mao, S., and Liu, Y. (2014). Big data: A survey. *Mobile networks and applications*, 19(2):171–209.

Chen, X. and Zhan, F. B. (2014). Agent-based modeling and simulation of urban evacuation: relative effectiveness of simultaneous and staged evacuation strategies. In *Agent-Based Modeling and Simulation*, pages 78–96. Springer.

Clay, R., Kieu, L.-M., Ward, J. A., Heppenstall, A., and Malleson, N. (2020). Towards real-time crowd simulation under uncertainty using an agent-based model and an unscented kalman filter. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 68–79. Springer.

Clay, R., Ward, J. A., Ternes, P., Kieu, L.-M., and Malleson, N. (2021). Real-time agent-based crowd simulation with the reversible jump unscented kalman filter. *Simulation Modelling Practice and Theory*, page 102386.

Cocucci, T. J., Pulido, M., Aparicio, J., Ruiz, J., Simoy, I., and Rosa, S. (2021). Inference in epidemiological agent-based models using ensemble-based data assimilation. *arXiv preprint arXiv:2111.00073*.

Collier, N. and Ozik, J. (2013). Test-driven agent-based simulation development. In *2013 winter simulations conference (WSC)*, pages 1551–1559. IEEE.

Cosgrove, J., Butler, J., Alden, K., Read, M., Kumar, V., Cucurull-Sanchez, L., Timmis, J., and Coles, M. (2015). Agent-based modeling in systems pharmacology. *CPT: pharmacometrics & systems pharmacology*, 4(11):615–629.

Crooks, A., Castle, C., and Batty, M. (2008). Key challenges in agent-based modelling for geo-spatial simulation. *Computers, Environment and Urban Systems*, 32(6):417–430.

Crooks, A. T. and Heppenstall, A. J. (2012). Introduction to agent-based modelling. In *Agent-based models of geographical systems*, pages 85–105. Springer.

Doucet, A., Godsill, S., and Andrieu, C. (2000). On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208.

Duboz, R., Versmisse, D., Travers, M., Ramat, E., and Shin, Y.-J. (2010). Application of an evolutionary algorithm to the inverse parameter estimation of an individual-based model. *Ecological modelling*, 221(5):840–849.

D'Orazio, M., Spalazzi, L., Quagliarini, E., and Bernardini, G. (2014). Agent-based model for earthquake pedestrians' evacuation in urban outdoor scenarios: Behavioural patterns definition and evacuation paths choice. *Safety science*, 62:450–465.

Elfring, J., Torta, E., and van de Molengraft, R. (2021). Particle filters: A hands-on tutorial. *Sensors*, 21(2):438.

Ellison, A. M. (2004). Bayesian inference in ecology. *Ecology letters*, 7(6):509–520.

Eppstein, M. J., Grover, D. K., Marshall, J. S., and Rizzo, D. M. (2011). An agent-based model to study market penetration of plug-in hybrid electric vehicles. *Energy Policy*, 39(6):3789–3802.

Etz, A. and Vandekerckhove, J. (2018). Introduction to bayesian inference for psychology. *Psychonomic Bulletin & Review*, 25(1):5–34.

Evensen, G. (2003). The ensemble kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367.

Evensen, G. (2009). The ensemble kalman filter for combined state and parameter estimation. *IEEE Control Systems Magazine*, 29(3):83–104.

Feng, X.-w., Yan, X.-f., and Hu, X.-l. (2015). Dynamic data driven particle filter for agent-based traffic state estimation. In *International Conference on Cloud Computing and Security*, pages 321–331. Springer.

Finnis, K. and Walton, D. (2006). Field observations of factors influencing walking speeds. *Ergonomics*.

Gan, Y., Duan, Q., Gong, W., Tong, C., Sun, Y., Chu, W., Ye, A., Miao, C., and Di, Z. (2014). A comprehensive evaluation of various sensitivity analysis methods: A case study with a hydrological model. *Environmental Modelling & Software*, 51:269–285.

Ge, J. (2017). Endogenous rise and collapse of housing price: An agent-based model of the housing market. *Computers, Environment and Urban Systems*, 62:182–198.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995). *Bayesian data analysis*. Chapman and Hall/CRC.

Ghoulmie, F., Cont, R., and Nadal, J.-P. (2005). Heterogeneity and feedback in an agent-based market model. *Journal of Physics: condensed matter*, 17(14):S1259.

Grewal, M. S. and Andrews, A. P. (2010). Applications of kalman filtering in aerospace 1960 to the present [historical perspectives]. *IEEE Control Systems Magazine*, 30(3):69–78.

Grimm, V., Berger, U., DeAngelis, D. L., Polhill, J. G., Giske, J., and Railsback, S. F. (2010). The odd protocol: a review and first update. *Ecological modelling*, 221(23):2760–2768.

Gupta, A. and Pundir, N. (2015). Pedestrian flow characteristics studies: A review. *Transport Reviews*, 35(4):445–465.

Hazelbag, C. M., Dushoff, J., Dominic, E. M., Mthombothi, Z. E., and Delva, W. (2020). Calibration of individual-based models to epidemiological data: A systematic review. *PLoS computational biology*, 16(5):e1007893.

Helbing, D. (1992). A fluid-dynamic model for the movement of pedestrians. *Complex systems*, 6(5):391–415.

Helbing, D., Buzna, L., Johansson, A., and Werner, T. (2005). Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. *Transportation science*, 39(1):1–24.

Helbing, D., Johansson, A., and Al-Abideen, H. Z. (2007). Dynamics of crowd disasters: An empirical study. *Physical review E*, 75(4):046109.

Helbing, D. and Molnar, P. (1995). Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282.

Helbing, D., Molnár, P., Farkas, I. J., and Bolay, K. (2001). Self-organizing pedestrian movement. *Environment and planning B: planning and design*, 28(3):361–383.

Hoogendoorn, S. and Bovy, P. (2003). Simulation of pedestrian flows by optimal control and differential games. *Optimal control applications and methods*, 24(3):153–172.

Hoteit, I., Luo, X., and Pham, D.-T. (2012). Particle kalman filtering: A nonlinear bayesian framework for ensemble kalman filters. *Monthly weather review*, 140(2):528–542.

Houhamdi, Z. (2011). Multi-agent system testing: A survey. *International Journal of Advanced Computer*, 2.

Ide, K., Courtier, P., Ghil, M., and Lorenc, A. C. (1997). Unified notation for data assimilation: Operational, sequential and variational (gtspecial issueltdata assimilation in meteology and oceanography: Theory and practice). *Journal of the Meteorological Society of Japan. Ser. II*, 75(1B):181–189.

Jabot, F., Faure, T., and Dumoulin, N. (2013). Easy abc: performing efficient approximate bayesian computation sampling schemes using r. *Methods in Ecology and Evolution*, 4(7):684–687.

Jammalamadaka, S. R. and Sengupta, A. (2001). *Topics in circular statistics*, volume 5. world scientific.

Jazwinski, A. H. (1970). Mathematics in science and engineering. *Stochastic processes and filtering theory*, 64.

Julier, S. J. and Uhlmann, J. K. (1997). New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–193. International Society for Optics and Photonics.

Julier, S. J., Uhlmann, J. K., and Durrant-Whyte, H. F. (1995). A new approach for filtering nonlinear systems. In *Proceedings of 1995 American Control Conference-ACC'95*, volume 3, pages 1628–1632. IEEE.

Jwo, D.-J. and Lai, C.-N. (2008). Unscented kalman filter with nonlinear dynamic process modeling for gps navigation. *GPS solutions*, 12(4):249–260.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

Kalman, R. E. and Bucy, R. S. (1961). New results in linear filtering and prediction theory. *Journal of basic engineering*, 83(1):95–108.

Kalnay, E. (2003). *Atmospheric modeling, data assimilation and predictability*. Cambridge university press.

Kalnay, E., Li, H., Miyoshi, T., Yang, S.-C., and Ballabrera-Poy, J. (2007). 4-d-var or ensemble kalman filter? *Tellus A: Dynamic Meteorology and Oceanography*, 59(5):758–773.

Katzfuss, M., Stroud, J. R., and Wikle, C. K. (2016). Understanding the ensemble kalman filter. *The American Statistician*, 70(4):350–357.

Kavak, H., Padilla, J. J., Lynch, C. J., and Diallo, S. Y. (2018). Big data, agents, and machine learning: towards a data-driven agent-based modeling approach. In *Proceedings of the Annual Simulation Symposium*, pages 1–12.

Keller, N. and Hu, X. (2016). Data driven simulation modeling for mobile agent-based systems. In *2016 Symposium on Theory of Modeling and Simulation (TMS-DEVS)*, pages 1–8. IEEE.

Keller, N. and Hu, X. (2019). *Towards data-driven simulation modeling for mobile agent-based systems*. PhD thesis.

Kennedy, W. G. (2012). Modelling human behaviour in agent-based models. In *Agent-based models of geographical systems*, pages 167–179. Springer.

Kieu, L.-M., Malleson, N., and Heppenstall, A. (2020). Dealing with uncertainty in agent-based models for short-term predictions. *Royal Society open science*, 7(1):191074.

Kinny, D., Georgeff, M., and Rao, A. (1996). A methodology and modelling technique for systems of bdi agents. In *European workshop on modelling autonomous agents in a multi-agent world*, pages 56–71. Springer.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

Kitazawa, K. and Batty, M. (2004). Pedestrian behaviour modelling.

Kretz, T., Grünebohm, A., Kaufman, M., Mazur, F., and Schreckenberg, M. (2006). Experimental study of pedestrian counterflow in a corridor. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(10):P10001.

Li, S., Dragicevic, S., Castro, F. A., Sester, M., Winter, S., Coltekin, A., Pettit, C., Jiang, B., Haworth, J., Stein, A., et al. (2016). Geospatial big data handling theory and methods: A review and research challenges. *ISPRS journal of Photogrammetry and Remote Sensing*, 115:119–133.

Li, S., Sayed, T., Zaki, M. H., Mori, G., Stefanus, F., Khanloo, B., and Saunier, N. (2012). Automated collection of pedestrian data through computer vision techniques. *Transportation research record*, 2299(1):121–127.

Liu, S., Lo, S., Ma, J., and Wang, W. (2014). An agent-based microscopic pedestrian flow simulation model for pedestrian traffic problems. *IEEE Transactions on Intelligent Transportation Systems*, 15(3):992–1001.

Lueck, J., Rife, J. H., Swarup, S., and Uddin, N. (2019). Who goes there? using an agent-based simulation for tracking population movement. In *2019 Winter Simulation Conference (WSC)*, pages 227–238. IEEE.

Lysenko, M. and D'Souza, R. M. (2008). A framework for megascale agent based model simulations on graphics processing units. *Journal of Artificial Societies and Social Simulation*, 11(4):10.

Macal, C. M. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, 10(2):144–156.

Macal, C. M. and North, M. J. (2005). Tutorial on agent-based modeling and simulation. In *Proceedings of the Winter Simulation Conference, 2005.*, pages 14–pp. IEEE.

Maeda, T., Obara, K., Shinohara, M., Kanazawa, T., and Uehira, K. (2015). Successive estimation of a tsunami wavefield without earthquake source data: A data assimilation approach toward real-time tsunami forecasting. *Geophysical Research Letters*, 42(19):7923–7932.

Maffei, S., Leoni, F., and Villari, B. (2020). Data-driven anticipatory governance. emerging scenarios in data for policy practices. *Policy Design and Practice*, 3(2):123–134.

Malleson, N., Minors, K., Kieu, L.-M., Ward, J. A., West, A., and Heppenstall, A. (2020). Simulating crowds in real time with agent-based modelling and a particle filter. *Journal of Artificial Societies and Social Simulation*, 23(3):3.

Mandel, J., Cobb, L., and Beezley, J. D. (2011). On the convergence of the ensemble kalman filter. *Applications of Mathematics*, 56(6):533–541.

Marinică, N. E., Sarlette, A., and Boel, R. K. (2013). Distributed particle filter for urban traffic networks using a platoon-based model. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1918–1929.

Martinez-Gil, F., Lozano, M., García-Fernández, I., and Fernández, F. (2017). Modeling, evaluation, and scale on artificial pedestrians: a literature review. *ACM Computing Surveys (CSUR)*, 50(5):1–35.

McDougall, D. and Moore, R. O. (2017). Optimal strategies for the control of autonomous vehicles in data assimilation. *Physica D: Nonlinear Phenomena*, 351:42–52.

McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245.

Motter, A. E., Matías, M. A., Kurths, J., and Ott, E. (2006). Dynamics on complex networks and applications.

Navon, I. M. (2009). Data assimilation for numerical weather prediction: a review. *Data assimilation for atmospheric, oceanic and hydrologic applications*, pages 21–65.

Nguyen, C. D., Perini, A., Bernon, C., Pavón, J., and Thangarajah, J. (2009). Testing in multi-agent systems. In *International Workshop on Agent-Oriented Software Engineering*, pages 180–190. Springer.

Oloo, F. and Wallentin, G. (2017). An adaptive agent-based model of homing pigeons: A genetic algorithm approach. *ISPRS International Journal of Geo-Information*, 6(1):27.

Orlande, H., Colaço, M., Dulikravich, G., Vianna, F., Silva, W., Fonseca, H., and Fudym, O. (2011). Kalman and particle filters. *METTI V-Thermal Measurements and Inverse Techniques.*

Pearson, J. B. and Stear, E. B. (1974). Kalman filter applications in airborne radar tracking. *IEEE Transactions on Aerospace and Electronic systems*, (3):319–329.

Psyllidis, A., Bozzon, A., Bocconi, S., and Bolivar, C. T. (2015). A platform for urban analytics and semantic data integration in city planning. In *International conference on computer-aided architectural design futures*, pages 21–36. Springer.

Rabiner, L. and Juang, B. (1986). An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16.

Rai, S. and Hu, X. (2013). Behavior pattern detection for data assimilation in agent-based simulation of smart environments. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 2, pages 171–178. IEEE.

Railsback, S. F. and Harvey, B. C. (2002). Analysis of habitat-selection rules using an individual-based model. *Ecology*, pages 1817–1830.

Railsback, S. F., Lamberson, R. H., Harvey, B. C., and Duffy, W. E. (1999). Movement rules for individual-based models of stream fish. *Ecological Modelling*, 123(2-3):73–89.

Rao, A. S., Georgeff, M. P., et al. (1995). Bdi agents: from theory to practice. In *Icmas*, volume 95, pages 312–319.

Ribeiro, M. I. (2004). Kalman and extended kalman filters: Concept, derivation and properties. *Institute for Systems and Robotics*, 43:46.

Rosés, R., Kadar, C., and Malleson, N. (2021). A data-driven agent-based simulation to predict crime patterns in an urban environment. *Computers, Environment and Urban Systems*, 89:101660.

Ruiz, J. J., Pulido, M., and Miyoshi, T. (2013). Estimating model parameters with ensemble-based data assimilation: A review. *Journal of the Meteorological Society of Japan. Ser. II*, 91(2):79–99.

Saltelli, A. (1999). Sensitivity analysis: Could better methods be used? *Journal of Geophysical Research: Atmospheres*, 104(D3):3789–3793.

Saltelli, A. (2002). Sensitivity analysis for importance assessment. *Risk analysis*, 22(3):579–590.

Saltelli, A., Tarantola, S., and Campolongo, F. (2000). Sensitivity analysis as an ingredient of modeling. *Statistical Science*, pages 377–395.

Särkkä, S. (2013). *Bayesian filtering and smoothing*. Number 3. Cambridge University Press.

Schadschneider, A. (2002). Cellular automaton approach to pedestrian dynamics-theory. *Pedestrian and Evacuation Dynamics*, pages 75–85.

Schadschneider, A., Klüpfel, H., Kretz, T., Rogsch, C., and Seyfried, A. (2009). Fundamentals of pedestrian and evacuation dynamics. In *Multi-Agent Systems for Traffic and Transportation Engineering*, pages 124–154. IGI Global.

Schadschneider, A. and Seyfried, A. (2011). Empirical results for pedestrian dynamics and their implications for modeling. *Networks & heterogeneous media*, 6(3):545.

Schmidt, B. (2005). Human factors in complex systems: The modelling of human behaviour. *Simulation in wider Europe, 19th European Conferance on Modelling and Simulation*, pages 5–14.

Schulze, J., Müller, B., Groeneveld, J., and Grimm, V. (2017). Agent-based modelling of social-ecological systems: achievements, challenges, and a way forward. *Journal of Artificial Societies and Social Simulation*, 20(2).

Seyfried, A., Steffen, B., Klingsch, W., and Boltes, M. (2005). The fundamental diagram of pedestrian movement revisited. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(10):P10002.

Smith, G. L., Schmidt, S. F., and McGee, L. A. (1962). *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle*. National Aeronautics and Space Administration.

Stanislaw, H. (1986). Tests of computer simulation validity: what do they measure? *Simulation & Games*, 17(2):173–191.

Steffen, B. and Seyfried, A. (2010). Methods for measuring pedestrian density, flow, speed and direction with minimal scatter. *Physica A: Statistical mechanics and its applications*, 389(9):1902–1910.

Still, K., Papalexi, M., Fan, Y., and Bamford, D. (2020). Place crowd safety, crowd science? case studies and application. *Journal of Place Management and Development*.

Sun, T., McMinn, P., Holcombe, M., Smallwood, R., and MacNeil, S. (2008). Agent based modelling helps in understanding the rules by which fibroblasts support keratinocyte colony formation. *PloS one*, 3(5):e2129.

Talagrand, O. (1997). Assimilation of observations, an introduction. *Journal of the Meteorological Society of Japan*, 75(1B):191–209.

Terejanu, G. A. (2011). Unscented kalman filter tutorial. *University at Buffalo, Buffalo*.

Terejanu, G. A. et al. (2008). Extended kalman filter tutorial. *University at Buffalo*.

Ternes, P., Ward, J. A., Heppenstall, A., Kumar, V., Kieu, L.-M., and Malleson, N. (2021). Data assimilation and agent-based modelling: towards the incorporation of categorical agent parameters. *Open Research Europe*, 1(131):131.

Thiele, J. C., Kurth, W., and Grimm, V. (2014). Facilitating parameter estimation and sensitivity analysis of agent-based models: A cookbook using netlogo and r. *Journal of Artificial Societies and Social Simulation*, 17(3):11.

Togashi, F., Misaka, T., Löhner, R., and Obayashi, S. (2020). Application of ensemble kalman filter to pedestrian flow. *Collective Dynamics*, 5:467–470.

van der Vaart, E., Beaumont, M. A., Johnston, A. S., and Sibly, R. M. (2015). Calibration and evaluation of individual-based models using approximate bayesian computation. *Ecological Modelling*, 312:182–190.

van Veenstra, A. F. and Kotterink, B. (2017). Data-driven policy making: The policy lab approach. In *International conference on electronic participation*, pages 100–111. Springer.

Vermeulen, B., Müller, M., and Pyka, A. (2021). Social network metric-based interventions? experiments with an agent-based model of the covid-19 pandemic in a metropolitan region. *Journal of Artificial Societies and Social Simulation*, 24(3).

Vermuyten, H., Beliën, J., De Boeck, L., Reniers, G., and Wauters, T. (2016). A review of optimisation models for pedestrian evacuation and design problems. *Safety science*, 87:167–178.

Von Toussaint, U. (2011). Bayesian inference in physics. *Reviews of Modern Physics*, 83(3):943.

Wagoum, A. K., Tordeux, A., and Liao, W. (2017). Understanding human queuing behaviour at exits: an empirical study. *Royal Society open science*, 4(1):160896.

Wan, E. A., Van Der Merwe, R., and Haykin, S. (2001). The unscented kalman filter. *Kalman filtering and neural networks*, 5(2007):221–280.

Wang, M. (2014). *Data assimilation for agent-based simulation of smart environment.* PhD thesis.

Wang, M. and Hu, X. (2013). Data assimilation in agent based simulation of smart environment. In *Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pages 379–384. ACM.

Wang, M. and Hu, X. (2015). Data assimilation in agent based simulation of smart environments using particle filters. *Simulation Modelling Practice and Theory*, 56:36–54.

Ward, J. A., Evans, A. J., and Malleson, N. S. (2016). Dynamic calibration of agent-based models using data assimilation. *Royal Society open science*, 3(4):150703.

Whipp, A., Malleson, N., Ward, J., and Heppenstall, A. (2021). Estimates of the ambient population: Assessing the utility of conventional and novel data sources. *ISPRS International Journal of Geo-Information*, 10(3):131.

Wikle, C. K. and Berliner, L. M. (2007). A bayesian tutorial for data assimilation. *Physica D: Nonlinear Phenomena*, 230(1-2):1–16.

Wu, C.-L. and Chen, Y. (2019). Effects of passenger characteristics and terminal layout on airport retail revenue: an agent-based simulation approach. *Transportation Planning and Technology*, 42(2):167–186.

Xiang, X., Kennedy, R., Madey, G., and Cabaniss, S. (2005). Verification and validation of agent-based scientific simulation models. In *Agent-directed simulation conference*, volume 47, page 55.

Yang, D., Yurtsever, E., Renganathan, V., Redmill, K. A., and Özgüner, Ü. (2021). A vision-based social distancing and critical density detection system for covid-19. *Sensors*, 21(13):4608.

Young, S. B. (1999). Evaluation of pedestrian walking speeds in airport terminals. *Transportation Research Record*, 1674(1):20–26.

Zhang, H., Hendricks Franssen, H.-J., Han, X., Vrugt, J. A., and Vereecken, H. (2017). State and parameter estimation of two land surface models using the ensemble kalman filter and the particle filter. *Hydrology and Earth System Sciences*, 21(9):4927–4958.

Zhang, H., Vorobeychik, Y., Letchford, J., and Lakkaraju, K. (2016). Data-driven agent-based modeling, with application to rooftop solar adoption. *Autonomous Agents and Multi-Agent Systems*, 30(6):1023–1049.

Zheng, Y., Capra, L., Wolfson, O., and Yang, H. (2014). Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):1–55.

Zhong, J., Cai, W., Luo, L., and Yin, H. (2015). Learning behavior patterns from video: A data-driven framework for agent-based crowd modeling. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 801–809.

Zhou, B., Wang, X., and Tang, X. (2012). Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2871–2878. IEEE.

Zhou, Y., McLaughlin, D., and Entekhabi, D. (2006). Assessing the performance of the ensemble kalman filter for land surface data assimilation. *Monthly weather review*, 134(8):2128–2142.

Zsifkovits, M. and Pham, T. (2017). Modelling and parameterizing pedestrian behaviour in public places: A review. *International Journal of Simulation Modelling*, 16(4):630–643.