Real-time Pedestrian Modelling: Implementing the Ensemble Kalman Filter for an Agent-Based Model

Keiran Suchak — 200888140

July 9, 2019

Abstract

This is the abstract.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A better understanding of how people move around their environment is of great utility to both academics and policy-makers. Such knowledge can be made use of in the contexts of urban planning, event management and emergency response, particularly when considering urban environments. Furthermore, this may also be of use to those interested in the social issues of mobility, inclusivity and accessibility of opportunities.

When considering such concepts, 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. However, these methods exist largely in isolation of the real-world — that is to say that whilst the simulations aim to reflect the real-world, there is no method by which we can incorporate up-to-date observations into these models to stop their divergence from reality.

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; 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. 2003) or lane formation (Liu et al. 2014). The evaluation of these rules is often not deterministic and instead introduces some element of randomness; these stochastic elements aim 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 result in simulation runs diverging from the real system.

In constructing their models, agent-based modellers undertake a development process that involves model verification, validation and calibration. We can take these to mean the following:

- **Model verification**: the process of ensuring that the implementation is an accurate representation of the model (Xiang et al. 2005).

- **Model validation**: the process of ensuring that the chosen model is an accurate representation of the phenomenon that we wish to study (Crooks et al. 2008).

- **Model calibration**: the process of searching for model parameter values such that we can achieve validation (Thiele et al. 2014).

Beyond this, modellers also make efforts to ensure that the initial model conditions are realistic by setting them based on historical data.

The practices of validation, calibration and setting initial model states based on historical data are appropriate for offline evaluations such as testing designs of new buildings or experimenting with different individual behaviours; however, when aiming to simulate events in real-time, this simply delays the inevitable divergence of the model from the real system. Furthermore, model parameters may be transient and thus require to be updated as time passes and the dynamics evolve.

Given the apparently inevitable divergence of stochastic simulations from the real systems that they aim to model, one may alternatively turn to big data. Data is 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 typically sparse in either time or space (or both); this is to say that observations rarely provide complete coverage of the events. 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**: Run the model forward, estimating the state of the system, until new observations become available.

2. **Update**: Upon receipt of new observations, combine the model's estimate of the system state with the new data.

These steps are repeated iteratively in a cycle. It is important to note that just as the there is error associated with the model, we also acknowledge that there is observational error associated with the data. The aim of incorporating the observations into the model is to improve the model accuracy with respect to the true system state.

A large volume of work exists in which such techniques are applied to meteorological systems where the models used are based on differential equations. Significantly less work exists in which data assimilation methods are applied to agent-based models — in particular pedestrian models. This dissertation therefore aims to expand on the pre-existing work by implementing a data assimilation scheme known as the Ensemble Kalman Filter in conjunction with a relatively simple agent-based model of pedestrians crossing a two-dimensional station from one side to the other.

1.1 Aims and Objectives

With the above in mind, this dissertation aims to determine whether the Ensemble Kalman Filter method of data assimilation can be used to improve the accuracy with which an Agent-Based Model simulates pedestrian movements given synthetic real-time data. In order to achieve this, the following objectives are set out:

1. Develop a general Ensemble Kalman Filter computational class.

2. Apply the Ensemble Kalman Filter to the pedestrian model.

3. Compare the accuracy with which the Ensemble Kalman Filter implementation of the model simulates the pedestrian movement against the based model without data assimilation.

The remainder of this dissertation will provide an overview of some of the work that has been undertaken thus far on implementing data assimilation schemes with agent based models (Chapter 2), elaborate on how data assimilation works — particularly focussing on the aforementioned Ensemble Kalman Filter (Chapter 3) — and examine the effectiveness of this method in improving the accuracy of the agent-based model (Chapter 4).

Chapter 2

Literature Review

As touched upon in Chapter 1, the process of developing an agent-based model typically involves some form of model calibration. Model calibration is the procedure of fine-tuning the model that we are using such that it best fits the particular situation that we are seeking to model (Crooks & Heppenstall 2012). There are a large number of different manners in which we can calibrate agent-based models (Thiele et al. 2014). These approaches typically involve making use of real-world data to estimate the parameters and initial state of the model; this is, however, undertaken once prior to running the model.

In some situations, we aim to simulate events in real-time (or close to real-time). In such situations, we are often able to observe the evolution of the real-world system which we seek to model and consequently may wish to use this information to recalibrate the model. This would, however, require that we stop the simulation, undertake calibration, and restart the model. We therefore seek an approach that allows us to incorporate observations of the system whilst simulating the system — data assimilation.

This Chapter will therefore seek to provide a basic overview of data as-

similation, along with some coverage of the attempts that have been made to implement such techniques to agent-based models that simulate urban systems.

2.1 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. Such a process can be achieved through a Bayesian filtering approach (Bar-Shalom et al. 2004, Jazwinski 1970).

Under such a framework, the updating of the model state is undertaken on the basis of Bayes Rule (for which a derivation is provided in Appendix A):

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

Bayes Rule is made up of four components:

1. $P(A)$: The probability of A, known as the **Prior**.
2. $P(A|B)$: The probability of A given B, known as the **Posterior**.
3. $P(B|A)$: The probability of B given A, known as the **Likelihood**.
4. $P(B)$: The probability of B, known as the **Marginal Likelihood**.

When applying this notation to the problem at hand, the components become:

1. **Prior**, $P(\mathbf{x})$: The probability distribution representing the prior state of the model.

2. **Posterior**, $P(\mathbf{x}|\mathbf{d})$: The probability distribution representing the updated state of the model in light of the observed data, that is to say the probability of the model state given the data.

3. **Likelihood**, $P(\mathbf{d}|\mathbf{x})$: The probability distribution of the observed data given the model state.

4. **Marginal Likelihood**, $P(\mathbf{d})$: The probability distribution representing the observed data.

With the above notation, Bayes Rule becomes:

$$P\left(\mathbf{x}|\mathbf{d}\right) = \frac{P\left(\mathbf{d}|\mathbf{x}\right) P\left(\mathbf{x}\right)}{P\left(\mathbf{d}\right)} \tag{2.2}$$

The aim of a data assimilation scheme therefore becomes to provide an update to the state in the form of the posterior, $P\left(\mathbf{x}|\mathbf{d}\right)$, given new observations, $P\left(\mathbf{d}\right)$.

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 (and variations thereof), Particle Filter.

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

Of the work that currently exists wherein investigators attempt to apply data assimilation schemes to agent-based models, most make use of sequential schemes.

2.2 Application of Sequential Data Assimilation to Agent-Based Models

2.2.1 Particle Filter-based Approaches

One of earliest pieces of work undertaken on the application of data assimilation schemes to agent-based models of urban environments was by Wang & Hu (2013). In this work, they simulate a smart office environment with people in it — a scenario that is becoming increasingly common with the advent of the Internet of Things (Zanella et al. 2014). The aim of the work was to make use of real-time data in conjunction with the agent-based simulation to provide more accurate estimates of the occupancy of the environment. This was achieved using the Particle Filter method of data assimilation; the method was chosen as it did not require the system to be Gaussian. The particle filter method operates by holding an ensemble of realisations of the simulation, each of which are evolved forward over time between observations; when observations are received, the particle states are weighted and the new state is obtained by sampling from these weighted particles. The observations used were synthetic data generated by the agent-based model, aiming to emulate motion sensors which would provide a binary response of whether a person was present in a given location.

The work undertaken consisted of two experiments — firstly simulating single agent in the environment, then going on to simulate two agents in the same environment. In the case if the first experiment, the agent was simulated with two different routing behaviours; for the first routing behaviour, the agent move forward sequentially through a series of waypoints, whilst for the second routing behaviour, the agent moves through a series of waypoints before turning back to return to its initial position. In this experiment, it was

found that the simulation error decreased when the agent was detected at each of the sensors for both routing behaviours with error growing between detections; this is as expected — it confirms that the simulation becomes more accurate with the addition of further information regarding the system. In the case of the second routing behaviour, the simulation error also grew following the agent's turn to head back to its origin. In the second experiment, they aimed to simulate two agents in the same environment, with the two agents maintaining spatial separation. This simulation was run a number of times for different numbers of particles with a view to establishing a relationship between the number of particles and simulation accuracy. It was found that as the number of particles was increased (through 400, 800, 1200 and 1600), the simulation error decreased. It was also found, however, that the experiments with fewer particles (400 and 800) struggled to converge, with the smaller number of particles unable to provide sufficient coverage of the state space. It was therefore noted that as the number of agents was increased, the method was likely to struggle.

This final issue may be solved by an increase in the number of particles; however, this comes with an attached increase in the computational cost (both in terms of compute time and space). The implementation of the particle filter requires that a realisation of the model be kept for each particle, resulting in growing memory requirements as the number of particles are increased. Furthermore, each particle is required to evolve the model for each time-step, resulting in an increasing computational cost.

There are two subsequent pieces of research which have sought to build directly upon the above initial investigation; each of them make use of the same simulation model, adding different developments.

The first of these was undertaken by Rai & Hu (2013) and aimed to add

a further layer by estimating behavioural patterns in the system. This was achieved using a hidden Markov model which was trained on the historical data of the system. The information encoded in the hidden Markov model is incorporated into the particle filter at the sampling step; the hidden Markov model is first used to identify the types of behaviour being exhibited by the system and particles then sample from different types, subsequently engaging the simulation with said behaviour. The remaining steps of the filtering process remained the same. The aim of applying this approach was to further improve the accuracy of simulations through the identification of pedestrian behaviours. In order to do so, the behaviours were divided into the following categories:

- **Outside**: All agents wait outside of a conference room.
- **In Conference**: All agents are attending meeting in the conference room.
- **Few Entering**: A small number of agents entering the conference room.
- **High Entering**: A large number of agents entering the conference room.
- **Few Leaving**: A small number of agent leaving the conference room.
- **High Leaving**: A large number of agent leaving the conference room.

These categorisations, particularly the *entering* and *leaving* states, present a problem. When entering the conference room, agents are more likely to be categorised into three groups — early few entering, high entering and late few entering — this re-categorisation helps to describe the status of the agents that are not in the process of entering:

- **Early few entering**: A small number of agents entering the conference room early before a meeting, with the other agents outside the conference room.
- **High entering**: A large number of agents entering the conference room.
- **Late few entering**: A small number of agents entering the conference room late after the meeting has started, with the other agents inside the conference room.

The investigation then attempts to determine the accuracy with which the model is able to identify the correct behaviour for agents, with accuracy being defined as

$$\frac{1}{T}\sum_{k=1}^{T} S_t^k - S_t^{real} \tag{2.3}$$

where T is taken to be the total number of simulation steps and S is the behaviour pattern state. It is unclear, however, how this calculation is undertaken given that the behavioural states are categories and not numerical; furthermore, the meaning behind the state notation is explained — it appears that k is a time-step index, however it does not make sense to compare the behavioural state at each time state to a static "real" behavioural state and the latter would likely be a transient property. Some indication is given that a numerical encoding of the categories has been used for visualisation purposes, however this should not be used for arithmetic purposes, nor should any ordinality be inferred from it. Indeed, the results presented take the form of accuracy percentages, suggesting that a more conventional accuracy score has been used, such as

$$\frac{1}{T}\sum_{k=1}^{T} \mathbb{1}\left(\hat{S}_k = S_k^{real}\right) \tag{2.4}$$

where the indicator function, $\mathbb{1}(\ldots)$, returns 1 when the condition is fulfilled, else 0. The results suggest that the model developed accurately identifies the behavioural states except for states that occur infrequently; the model performs particularly well when identifying states in which agents are static, i.e. *outside* and *in conference*. Such supplementary information could improve the performance of data driven simulations, likely helping the process of data assimilation for parameter estimation. It is worth considering, however, that the addition of this further layer to the assimilation process would also result in a further increase in the computational cost.

The second of the investigations to develop on the initial work by Wang & Hu (2013) was undertaken by Wang & Hu (2015), this time making use of three different resampling schemes: standard resampling (as seen in the original investigation), component set resampling and mixed component set resampling. Given these resampling schemes, they undertake four experiments.

The first of these experiments seeks to address the use of component set resampling. This is achieved by testing the implementation for increasing numbers of agent with component set resampling, and comparing against the corresponding results when using standard resampling. It was found that, when using standard resampling, the number of particles in the ensemble that matched with observations decreased as the number of agents increased; the use of mixed component resampling was found to reduce the rate at which this occurred.

The second of the experiments aims to assess the effectiveness of the particle filter using the standard resampling scheme when simulating one agent. For this experiment, the agent was imparted with two different behaviours as in the original investigation. The effectiveness of the data assimilation

scheme was assess by measuring the average distance per particle between the simulated agent and the real agent. It was once again found that the simulation error fell with each observation. Furthermore, it was noted that there were two situations that caused an increase in the simulation error:

1. The agent turning back on itself in an area where no sensors are present.

2. The agent approaches a 3-way intersection, where the agent is offered discretely different option of direction in which to travel; the particles, therefore, struggle to converge on the true state.

The third experiment aims to asses the effectiveness of standard resampling when applying the scheme to a system containing two agents. This was achieved by comparing the average error per agent per particle for different numbers of particles (1200 particles, 1600 particles and 2000 particles). It was found that as the number of particles increased, the simulation error decreased.

The final experiment aimed to similarly assess the effectiveness mixed component set resampling for a system containing multiple agents, first starting with two agents. It was found that for each of the options for number of particles, the implementation of mixed component set resampling reduced the simulation error. Furthermore, when considering systems containing more agents (four or six), it was also found that the implementation of mixed component set resampling improved the simulation accuracy; however, as the number of agents in the system increased, this improvement reduced. This was attributed to situations when agents would crowd together, thus causing difficulties for the particles to distinguish different agents from binary sensors.

2.2.2 Kalman Filter-based Approaches

Other investigations have sought to apply different data assimilation schemes including the Ensemble Kalman Filter. As shall be explained in Section 3, the Ensemble Kalman Filter is an adaptation of the original Kalman Filter (Evensen 2003). This data assimilation technique was implemented in the investigation by Ward et al. (2016), which sought to expose agent based modelling practitioners to the technique in the context of modelling how many people are in a major city at a given time. This investigation consisted of two experiments. In the first experiment, the Ensemble Kalman Filter was implemented with a simple box model that estimated how many people were present in the box based on probabilities of people entering and exiting. In the second experiment, the Ensemble Kalman Filter was applied to an epidemic-like model in which the population was split into workers and shoppers, with works either being at home or at work, and shoppers either being susceptible to going shopping, shopping in town or having returned home after shopping.

The first experiment made use of synthetic data generated using the model with randomly drawn parameter values in order to produce ground truth data. Observations were then generated by adding normally distributed random noise to the ground truth. Running the filtering process with an ensemble of 100 realisations, data assimilation for state estimation was performed at each time-step, and it was found that on average, the error (with respect to the synthetic ground truth) of the model state was smaller after assimilation, as well as being smaller than the error in the observations. Furthermore, this approach also outperforms the theoretical steady state calculated for the system.

Beyond this, the first experiment also aimed to carry out parameter es-

timation by including the parameters, which are assumed to be unknown, in the state vector. In doing so, estimates of the parameters are produced at both the forecasting stage and the updating stage. The filtering process succeeds in reducing the error in the parameters with respect to the ground truth; despite this, the parameter estimates do not converge on their true values, underestimating both the arrival rate and departure rate. The ratio of the two parameters, however, is correctly estimated, suggesting that the data assimilation process has correctly estimated the governing dynamics.

The second experiment aimed to model pedestrians arriving and departing at Briggate in Leeds. Pedestrians are divided into shoppers and workers, with each group being governed by epidemic-like dynamics. Shoppers are either at home before shopping (susceptible), in town shopping (infected) or at home having returned from shopping (recovered); workers are either at home or at work. This approach seeks to more realistically represent pedestrian behaviour, designating different types of people and introducing more complex behaviours for agents deciding to enter the city. The data used was sourced from a footfall camera on Briggate which recorded hourly counts of the number of pedestrians arriving; as such, the primary target of the assimilation process was the cumulative count of the number of agents to have arrived in the city, combining both shoppers and workers. The Ensemble Kalman Filter was applied using different ensemble sizes (10, 100 and 1000), and it was found that as the ensemble size increased, the accuracy of the simulation improved; it was noted, furthermore that the improvement observed between ensemble sizes of 10 and 100 were much greater than between ensemble sizes of 100 and 1000. Once again, parameter estimation was undertaken; however, in this case a particle filter-like approach was used.

Whilst this investigation displays that the Ensemble Kalman Filter 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; indeed, the authors admit that the models used for the investigation were not developed in the standard object-oriented framework typically used in agent-based modelling. The model used in the first experiment is a binary state model, with each agent either being in the city or not in 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 & 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. In the case of each of the experiments, data assimilation is used to perform both state estimation and parameter estimation.

Beyond this, it should be noted that the inclusion of parameter estimation, whilst not uncommon, is not a standard approach and so some attention should be given to the impact of its inclusion in the procedure.

2.3 Summary

As has been seen, there exist a small number of investigations which attempt to implement sequential data assimilation schemes in conjunction with agent-based pedestrian models; in each case, the filtering process lead to improvements in the modelling accuracy. Much of the work that has been undertaken

has used the Particle Filter, and has made use of synthetic data; the ultimate goal of developing such a technology will inevitably be to use it with real-world data which is generated in real-time.

Each of the two data assimilation schemes that have been used have their own strengths and weaknesses. The Ensemble Kalman Filter used by Ward et al. (2016) are typically run with smaller ensemble sizes, but struggle in cases when probability distributions are non-normal. The Particle Filters used by Rai & Hu (2013), Wang & Hu (2013, 2015) resolve this issues, offering an exact solution at the cost of requiring larger ensemble sizes and consequently greater computation space and time.

This work therefore seeks to build upon the work by Ward et al. (2016) in implementing an Ensemble Kalman Filter in conjunction with a agent-based pedestrian model.

As an additional note, whilst this dissertation focuses on the application of data assimilation methods to agent-based models, there also exists a body of work that makes use of the same methods in conjunction with cellular automata (Li et al. 2017, 2012). This may be of interest to those concerned more broadly with the development of real-time social simulation.

Chapter 3

Method

As summarised in Chapter 2, there exist a number of different data assimilation schemes, many of which are used extensively in fields such as numerical weather prediction, navigation and tracking, and other forecasting problems. Such methods, however, have been sparsely used in the field of real-time urban simulation and as such this investigation attempts to build upon the existing work by implementing the Ensemble Kalman Filter in conjunction with a pedestrian agent-based model. This chapter therefore seeks to outline the method used in this investigation — the Ensemble Kalman Filter. As shall be explained, the Ensemble Kalman Filter is an approximation of the Kalman Filter, and as such some attention will first be given to the original Kalman Filter, followed by an explanation of the Ensemble Kalman Filter along with the innovations that it incorporates.

3.1 Kalman Filter

One of the earliest forms of Bayesian filtering is the sequential data asssimilation scheme known as the Kalman Filter, which forms the foundation of

this piece of work. As with other sequential data assimilation schemes, the Kalman Filter operates on a model with a given state and forecasting process, updating the state and 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 with a view to minimising the posterior mean squared error. Under the Bayesian framework outlined in Section 2.1, we refer to the model state vector and associated covariance before updating as the prior, $\mathbf{x}$ and $\mathbf{Q}$, and to the model state and associated covariance after updating as the posterior, $\hat{\mathbf{x}}$ and $\hat{\mathbf{Q}}$. Given new observations, $\mathbf{d}$, the posterior state vector is given by

$$\hat{\mathbf{x}} = \mathbf{x} + \mathbf{K}\left(\mathbf{d} - \mathbf{H}\mathbf{x}\right), \tag{3.1}$$

and the posterior covariance is given by

$$\hat{\mathbf{Q}} = \left(\mathbf{I} - \mathbf{K}\mathbf{H}\right)\mathbf{Q}, \tag{3.2}$$

where $\mathbf{K}$ is the Kalman gain matrix and $\mathbf{H}$ is the observation operator. The Kalman gain matrix is given by

$$\mathbf{K} = \mathbf{Q}\mathbf{H}^T\left(\mathbf{H}\mathbf{Q}\mathbf{H}^T + \mathbf{R}\right)^{-1} \tag{3.3}$$

where $\mathbf{R}$ is the observation covariance.

In the case when the errors are normally distributed, this filter provides an exact posterior estimate. 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 Ensemble Kalman Filter.

3.2 Ensemble Kalman Filter

In order to address some of these problems, the Ensemble Kalman Filter was developed (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 & Berliner 2007). The remainder of this section will seek to outline this framework in a manner similar to that documented by Mandel (2009). The state is represented as an ensemble of state vectors as follows:

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] = [\mathbf{x}_i], \quad \forall i \in (1, N), \tag{3.4}$$

where the state ensemble matrix, $\mathbf{X}$, consists of N state vectors, $\mathbf{x}_i$. The mean state vector, $\bar{\mathbf{x}}$, can be found by averaging over the ensemble:

$$\bar{\mathbf{x}} = \sum_{i=1}^{N} \mathbf{x}_i. \tag{3.5}$$

Similarly, the observations are represented as follows:

$$\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_N] = [\mathbf{d}_i], \quad \forall i \in (1, N), \tag{3.6}$$

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:

$$\mathbf{d}_i = \mathbf{d} + \epsilon_i, \quad \forall i \in (1, N). \tag{3.7}$$

The random vector is drawn from an unbiased normal distribution:

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

As with the model state, the mean data vector, $\bar{\mathbf{d}}$, can be found by averaging over the ensemble:

$$\bar{\mathbf{d}} = \sum_{i=1}^{N} \mathbf{d}_i. \tag{3.8}$$

Given that the noise added to the original data vector is unbiased, we should expect that the mean data vector converges to the original data vector, $\mathbf{d}$:

$$\lim_{N \to \infty} \bar{\mathbf{d}} = \mathbf{d}.$$

By implementing these adaptations, the Ensemble Kalman Filter 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 forecasted by applying the model operator, but instead can simply be generated as a sampling covariance. 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, $\hat{\mathbf{X}}$, begin undertaken on the basis of the following equation:

$$\hat{\mathbf{X}} = \mathbf{X} + \mathbf{K}\left(\mathbf{D} - \mathbf{H}\mathbf{X}\right), \tag{3.9}$$

where $\mathbf{H}$ is once again the observation operator. In this case, the Kalman gain matrix, $\mathbf{K}$ is given by

$$\mathbf{K} = \mathbf{C}\mathbf{H}^T\left(\mathbf{H}\mathbf{C}\mathbf{H}^T + \mathbf{R}\right)^{-1}. \tag{3.10}$$

in which the previous state covariance matrix, $\mathbf{Q}$, has been replace with the sample state covariance matrix, $\mathbf{C}$.

3.3 Model

The above data assimilation scheme will be implemented in conjunction with an pedestrian agent-based model known as `StationSim`[1], which aims to simulate pedestrians crossing from one side of the station to the other. StationSim has been developed under an object-oriented framework, and as such both the model and the agent are represented by classes. The state of each agent comprises of its positions in two-dimensional continuous space, whilst the state of the model comprises of the collection of all of the agents in the model. At each time-step, the model state is evolved by sequentially evolving the agents.

The model is initialised by passing a number of arguments to the constructor, including the number of agents in the population and the dimensions of the rectangular environment. Upon initialisation, the model generates a population of agents, each of which are randomly allocated the following:

- Entrance through which to enter the environment.
- Exit through which to leave the environment.
- Speed at which to traverse the environment.

[1] https://github.com/Urban-Analytics/dust/blob/master/Projects/ABM_DA/at_risk/StationSim_KM.py

As shown in Figure 3.1, entrances are located on the left-hand side of the rectangular environment, and exits are located on the right-hand side of the environment, with each agent seeking to traverse the environment from their respective entrance to their respective exit. Where the paths of agents intersect in time and space, crowding occurs — faster agents attempt to pass slower agents, at times getting stuck behind them.

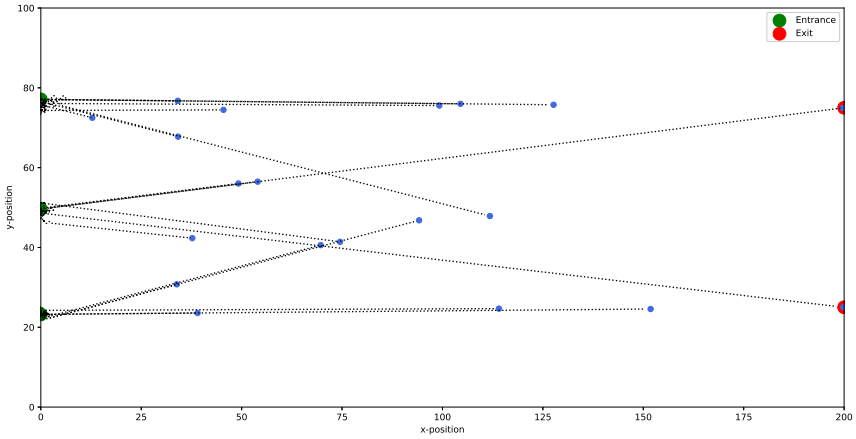


Figure 3.1: Sample model run.

3.4 Experimental Setup

With the above model in mind, a simple preliminary experiment was designed to demonstrate that the Ensemble Kalman Filter method of data assimilation can be applied to improve the model estimates of agent position of the agent based model. In order to achieve this, a class is developed in Python to represent the Ensemble Kalman Filter. The Python class representing the model is passed to the filter class as an argument, along with the filter ensemble size, the frequency with which the filter should assimilate data,

and parameters governing the observational noise. Upon construction, an instance of the filter class creates an instance of the model, referred to as the `base_model`. Subsequently, an ensemble of models is created by taking deep-copies of the `base_model`, thus ensuring that each of the ensemble members starts with model- and agent-attributes that match those in the `base_model`.

The instance of the filter class steps forward through time according to the predict-update cycle. At each predict step, each of the ensemble member models are stepped forward once, along with the base model; the frequency with which the filter undertakes the update step is governed by a parameter known as the `assimilation_period`. Upon reaching the update step of each cycle, the state of each of the ensemble member models is updated according to equations found in Section 3.2. The ground truth for these experiments is taken to be synthetic data generated by the `base_model`; observations are then generated by adding normally distributed random noise to the ground truth at each assimilation step.

- Population size: 20
- Ensemble size:
- Assimilation period:
- Observation error:

Chapter 4

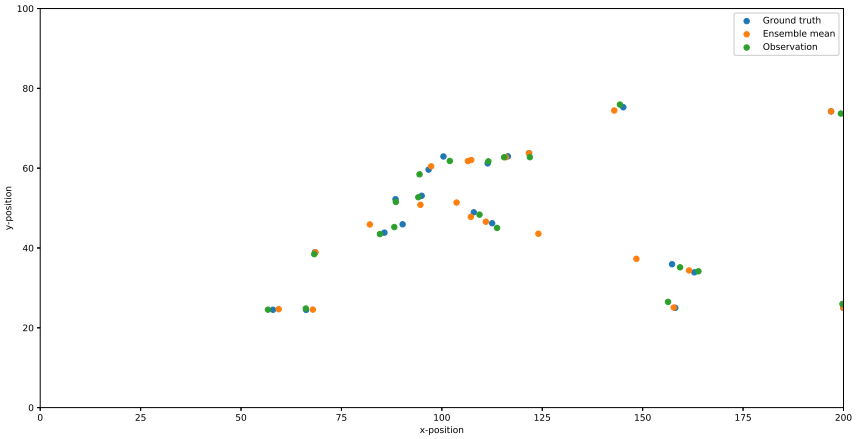
Results

Things to note about experiments:

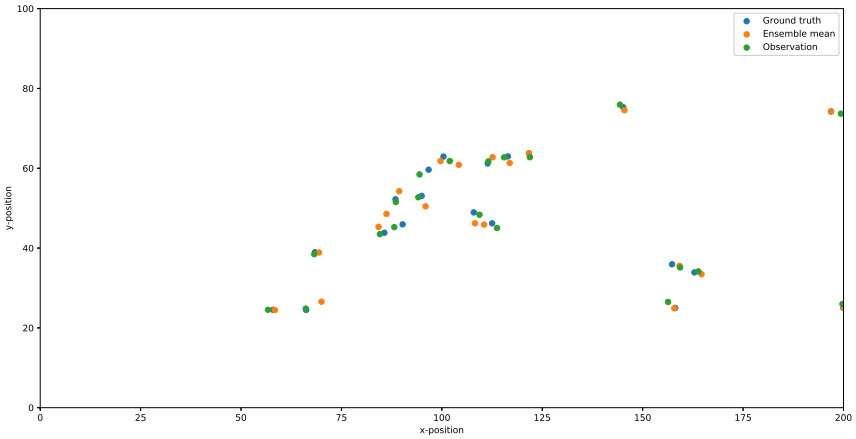
- Ensemble size
- Assimilation period
- Number of agents
- Parameters for generated noise

Things to write about:

- Show the impact of the update step. We can do this by showing the individual ensemble members, which are separated before the update process, converge on the particle when new data is provided.
- Show the variation in per-agent error over time in the case of with assimilation vs without assimilation.
- Compare average error before update and after update for each assimilation step.

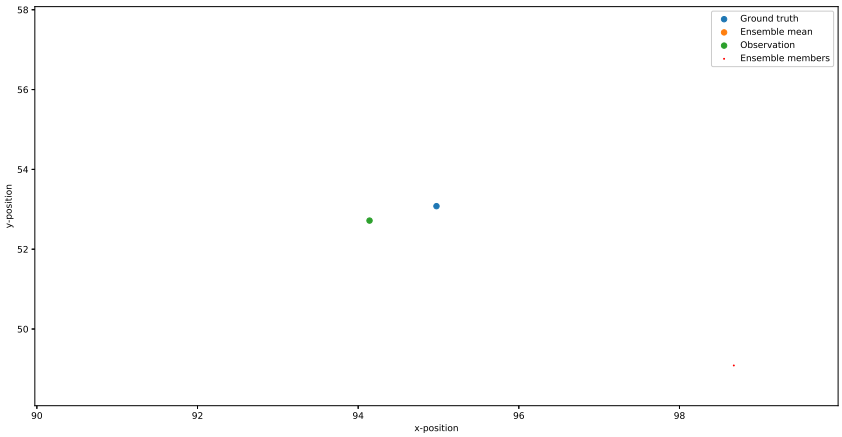


(a) Before update.

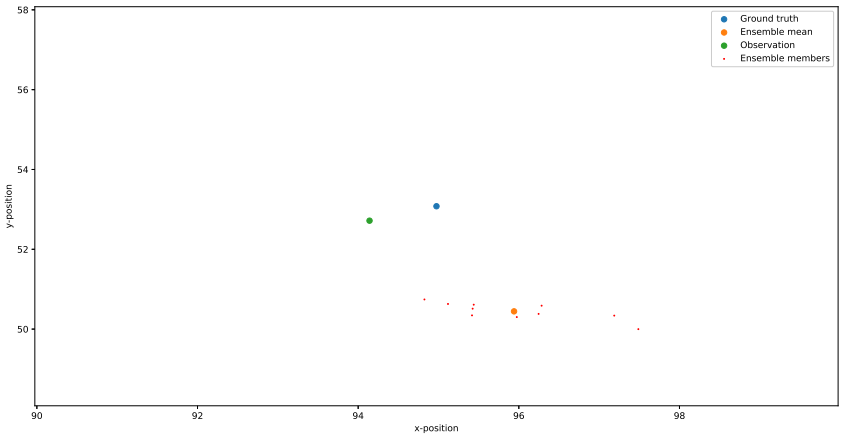


(b) After update.

Figure 4.1: Effect of Kalman Filter update on state of ABM.



(a) Before update.



(b) After update.

Figure 4.2: Effect of Kalman Filter update on state of a single agent in the ABM.

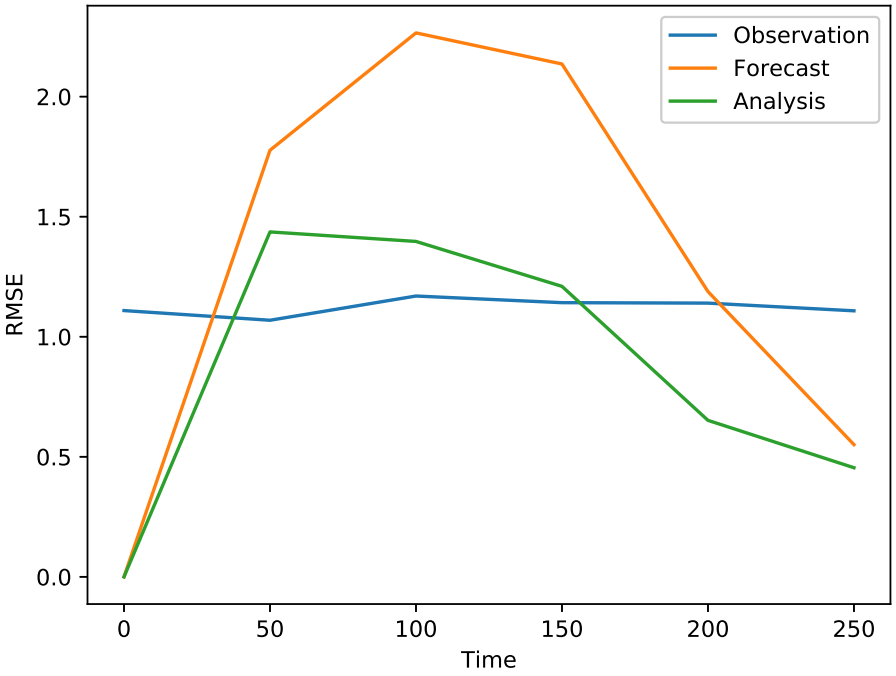


Figure 4.3: Comparison of RMSE.

Chapter 5

Conclusion

This is the conclusion.

- What does this investigation aim to do?
- How does it do it?
- Is is successful, and if so how successful?
- What are the limitations?

Problems to consider:

- What happens when agents leave the system — does the filter recognise this correctly?
- What happens when we are provided with aggregated information?
- What happens when we are provided with different levels of information for different agents?
- Can the filter tell agents apart?

- We have artificially told the filter information about agents' entrance and exits - what happens if it doesn't know these? Do we then need to do some assimilation for data assimilation?

5.1 Future Work

Ideas for transfer:

- Explore impact of different filter parameters on filter performance
- Explore impact of missing information; this can come in the form of less frequent observation, observations of a selection of the agents.
- Explore impact of aggregated observation
- Include derivation of multivariate Kalman filter
- Exploration of complexity of code (time and space)
- Multithreading to deal with computational cost - identifying cut-off point below which it is better to use serial computation. This doesn't seem necessary at this stage as code runs in reasonable time with given parameters, but something to consider for larger ensemble sizes, population sizes. There will likely be some trade-off when looking at assimilation period due to cost of message passing.

Different types of EnKF and the implications for ensemble size (Keller et al. 2018).

- Damping: counteract filter divergence
- Localisation: reduce the effect of spurious correlations

- Hybrid EnKF: Covariance matrix is made up of the weighted sum of the usual covariance matrix and a separate static covariance matrix that encodes prior underlying knowledge about the system
- Dual EnKF: Split the state vector into state and parameters. At assimilation: update parameters, recalculate forecast, update state
- Normal Score EnKF: Developed to handle non-Gaussian PDFs in EnKF. At assimilation: transform state, parameters and measurements into Z-scores, perform EnKF update based on transformed values, transform back from Z-scores
- Iterative EnKF

Extensions to the EnKF Katzfuss et al. (2016):

- Variance inflation: often have ensemble size much smaller than state dimension, which can mean that $\mathbf{K}$ can be a poor approximation of the kalman gain matrix; we can therefore get downwardly biased estimates of the posterior state covariance matrix — we can fix this with covariance inflation whereby we multiply the covariance matrix by a constant (greater than one). (Should check if this is an issue that we need to consider).
- Localisation: small ensemble sizes can result in spurious correlations between state components that are physically far apart — we can use localisation to avoid this.
- Parameter estimation:

Appendix A

Bayes Rule

The joint probability — $P(A, B)$ — is the probability of event A occurring and event B occurring. The conditional probability — $P(A|B)$ — is the probability of event A occurring given that event B occurs. The joint and conditional probabilities are linked by the following relationship:

$$P(A, B) = P(A)P(B|A). \tag{A.1}$$

Similarly, we have

$$P(B, A) = P(B)P(A|B). \tag{A.2}$$

Given that the joint probabilities of A, B and B, A are equal:

$$P(A, B) = P(B, A),$$

Equations A.1 and A.2 can be put together to give

$$P(A)P(B|A) = P(B)P(A|B). \tag{A.3}$$

From this, we can derive the typical form of Bayes Rule by dividing by $P(A)$:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}. \tag{A.4}$$

Bibliography

Bar-Shalom, Y., Li, X. R. & Kirubarajan, T. (2004), *Estimation with applications to tracking and navigation: theory algorithms and software*, John Wiley & Sons.

Batty, M., DeSyllas, J. & Duxbury, E. (2003), '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.

Burstedde, C., Klauck, K., Schadschneider, A. & 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.

Chen, M., Mao, S. & Liu, Y. (2014), 'Big data: A survey', *Mobile networks and applications* **19**(2), 171–209.

Crooks, A., Castle, C. & 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. & Heppenstall, A. J. (2012), Introduction to agent-based modelling, *in* 'Agent-based models of geographical systems', Springer, pp. 85–105.

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.

Helbing, D. & Molnar, P. (1995), 'Social force model for pedestrian dynamics', *Physical review E* **51**(5), 4282.

Jazwinski, A. H. (1970), 'Mathematics in science and engineering', *Stochastic processes and filtering theory* **64**.

Kalnay, E. (2003), *Atmospheric modeling, data assimilation and predictability*, Cambridge university press.

Katzfuss, M., Stroud, J. R. & Wikle, C. K. (2016), 'Understanding the ensemble kalman filter', *The American Statistician* **70**(4), 350–357.

Keller, J., Hendricks Franssen, H.-J. & Marquart, G. (2018), 'Comparing seven variants of the ensemble kalman filter: How many synthetic experiments are needed?', *Water Resources Research* **54**(9), 6299–6318.

Li, X., Lu, H., Zhou, Y., Hu, T., Liang, L., Liu, X., Hu, G. & Yu, L. (2017), 'Exploring the performance of spatio-temporal assimilation in an urban cellular automata model', *International Journal of Geographical Information Science* **31**(11), 2195–2215.

Li, X., Zhang, Y., Liu, X. & Chen, Y. (2012), 'Assimilating process context information of cellular automata into change detection for monitoring land use changes', *International Journal of Geographical Information Science* **26**(9), 1667–1687.

Liu, S., Lo, S., Ma, J. & 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.

Mandel, J. (2009), 'A brief tutorial on the ensemble kalman filter', *arXiv preprint arXiv:0901.3725* .

Rai, S. & 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)', Vol. 2, IEEE, pp. 171–178.

Stanislaw, H. (1986), 'Tests of computer simulation validity: what do they measure?', *Simulation & Games* **17**(2), 173–191.

Talagrand, O. (1997), 'Assimilation of observations, an introduction', *Journal of the Meteorological Society of Japan* **75**(1B), 191–209.

Thiele, J. C., Kurth, W. & 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.

Wang, M. & 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', ACM, pp. 379–384.

Wang, M. & 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. & Malleson, N. S. (2016), 'Dynamic calibration of agent-based models using data assimilation', *Royal Society open science* **3**(4), 150703.

Wikle, C. K. & Berliner, L. M. (2007), 'A bayesian tutorial for data assimilation', *Physica D: Nonlinear Phenomena* **230**(1-2), 1–16.

Xiang, X., Kennedy, R., Madey, G. & Cabaniss, S. (2005), Verification and validation of agent-based scientific simulation models, *in* 'Agent-directed simulation conference', Vol. 47, p. 55.

Zanella, A., Bui, N., Castellani, A., Vangelista, L. & Zorzi, M. (2014), 'Internet of things for smart cities', *IEEE Internet of Things journal* **1**(1), 22–32.