Simulation Modelling Practice and Theory 113 (2021) 102386

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Real-time agent-based crowd simulation with the Reversible Jump Unscented Kalman Filter[☆]

Robert Clay [a,*], Jonathan A. Ward [a], Patricia Ternes [a], Le-Minh Kieu [b], Nick Malleson [a]

[a] *University of Leeds, Leeds, LS2 9JT, UK*
[b] *University of Auckland, Auckland 1010, New Zealand*

ARTICLE INFO

Keywords:
Agent-based modelling
Data assimilation
Unscented Kalman filter
Crowd simulation
MCMC

ABSTRACT

Commonly-used data assimilation methods are being adapted for use with agent-based models with the aim of allowing optimisation in response to new data in *real-time*. However, existing methods face difficulties working with categorical parameters, which are common in agent-based models. This paper presents a new method, the RJUKF, that combines the Unscented Kalman Filter (UKF) data assimilation algorithm with elements of the Reversible Jump (RJ) Markov chain Monte Carlo method. The proposed method is able to conduct data assimilation on both continuous and categorical parameters simultaneously. Compared to similar techniques for mixed state estimation, the RJUKF has the advantage of being efficient enough for online (i.e. real-time) application. The new method is demonstrated on the simulation of a crowd of people traversing a train station and is able to estimate both their current position (a continuous, Gaussian variable) and their chosen destination (a categorical parameter). This method makes a valuable contribution towards the use of agent-based models as tools for the management of crowds in busy places such as public transport hubs, shopping centres, or high streets.

1. Introduction

Agent based models (ABMs) have become a popular method for simulating the dynamics of crowds [1,2] due to their ability to readily simulate individuals and their behaviour. In general, such models are calibrated to historical data and then used to make predictions. However, ABMs rarely incorporate any real-time data [3], so cannot reflect changes within the target system that occur in real time. This prohibits their use as real-time management tools, limiting them to conducting offline experiments based on historical data. If ABMs were able to update their current state based on up-to-date information from the real-world, as is common in fields such as meteorology [4], then they could become valuable tools for the management of crowds in busy places such as public transport hubs, shopping centres, or high streets.

To address this limitation, data assimilation techniques have been proposed as a way to incorporate new data into predictive ABMs. The Unscented Kalman Filter (UKF), in particular, is a method that could operate effectively without requiring a large number of model replicas to be run simultaneously (an ‘ensemble’) which is important as ABMs are typically extremely computationally

☆ This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 757455), through a UK Economic and Social Research Council (ESRC) Future Research Leaders grant (ES/L009900/1), and through an internship funded by the UK Leeds Institute for Data Analytics (LIDA). Computation was undertaken on ARC4, part of the High Performance Computing facilities at the University of Leeds, UK.

* Corresponding author.
E-mail addresses: gyrc@leeds.ac.uk (R. Clay), J.A.Ward@leeds.ac.uk (J.A. Ward), p.ternesdallagnollo@leeds.ac.uk (P. Ternes), minh.kieu@auckland.ac.nz (L.-M. Kieu), n.s.malleson@leeds.ac.uk (N. Malleson).

https://doi.org/10.1016/j.simpat.2021.102386
Received 12 March 2021; Received in revised form 8 July 2021; Accepted 29 July 2021
Available online 8 August 2021

expensive [5]. However, the vanilla implementation of the UKF has a number of limitations. In particular, the UKF is designed only for assimilating states with a Gaussian distribution – such as an agent's location in real space – not categorical or discrete variables. Most ABMs have mixed states of both categorical and continuous variables with comparable importance [6,7]. In models of crowds, for example, agents may have categorical variables to represent their gender [8], beliefs [9], destination [10], etc. These variables can define a totally different set of behaviours for each class of agent, so should not be disregarded as part of a data assimilation framework.

There is very limited research that deals data assimilation for ABMs, and even less that attempts to tackle the problem of combining continuous and categorical variables. This partly stems from the fact that advances in data assimilation have traditionally emerged from fields such as meteorology and hydrology, where models rarely incorporate categorical variables [4]. In short, there are two main problems that limit the veracity of current techniques. First, the model selection method, which is commonly used to deal with categorical variables, will struggle with the 'curse of dimensionality' that comes with categorical variables. For example, given an ABM with 10 agents each having a single categorical variable with 5 possible values, there are 5^{10} (nearly 10 million) different combinations of categories. Second, typical data assimilation frameworks require each individual model realisation to have the same parameter dimensions. Because each agent has its own local parameter values, adding or removing agents from a model has the effect of changing the number of model parameters. This complicates the implementation of data assimilation with ABMs, as it is common for the number of agents to vary between model runs.

This paper extends the state-of-the-art by proposing a method that allows an Unscented Kalman Filter (UKF) to conduct data assimilation on an agent-based model that consists of a mixed state of categorical and Gaussian variables. The method, which we call Reversible Jump UKF (RJUKF) is inspired by Reversible Jump Markov Chain Monte Carlo (RJMCMC) [11]. Compared to similar techniques for mixed state estimation, the RJUKF has the advantage of being efficient enough for online (i.e. real-time) application. This method is also able to handle variable dimension models making it suitable for application to ABMs.

The approach is demonstrated in the context of estimating the behaviour of a crowd of people as they traverse a train station – specifically Grand Central Terminal in New York City – using an agent-based model, *StationSim*. The model of the crowd will inevitably diverge from the 'true' behaviour of the crowd for two reasons: (i) the trajectories of the pedestrians vary stochastically as they interact with each other; and (ii) although we assume an individual's time and location of entry into the station are known (people are often recorded as they pass through barriers to enter a station), we cannot assume their *destination* is known, so this must be inferred from their evolving trajectories. The incorporation of up-to-date observations from the real world using an Unscented Kalman Filter can help to estimate the true *locations* of the pedestrians, but it cannot estimate the true *destinations* because these are categorical rather than continuous variables. For this, the RJUKF is needed. Experiments are presented to show that the RJUKF is able to estimate both location *and* destination, creating a much more robust real-time simulation of the station.

This paper is structured as follows: Section 2 reviews the relevant literature; Section 3 outlines the methods used, including a description of the agent-based model and the RJUKF; Section 4 outlines the experiments and results; and Section 5 draws conclusions and outlines potential future work.

2. Related work

The aim of Data Assimilation (DA) is to use current, real-world observations to update the internal state of a model. In this manner, "all the available information" [12] is used to create a combined representation of a system that is closer to its *true* state than either the observations or the model in isolation. Recent efforts have been made to develop methods that will allow ABMs to react to real-world events as they happen. Whilst promising, they exhibit a number of limitations such as: the need for manual calibration [13] (which is infeasible in most cases); implementations that contain only a few agents and/or limited interactions [14,15]; the use of agent-based models that are simple enough to be approximated by an aggregate mathematical model [3,16]; parameters that can be dynamically optimised but a model *state* that is not updated [17]; or the use of DA methods that do not scale to large model size [10,15]. Also, importantly, none of the previous approaches have attempted to assimilate data for categorical variables.

Of the DA methods commonly used in fields such as meteorology and hydrology, Kalman Filtering is probably the most widely known and has seen a broad application to discrete-time state-space estimation with the aim of stabilising noisy measurements into more consistent, smooth paths [4]. While the traditional filter is limited to linear models there are many extensions proposed to accommodate nonlinearity that typically fall into two categories: non-parametric techniques such as the Particle Filter (PF) [18]; and Ensemble Kalman Filters [19] that estimate model states through Monte Carlo techniques. The particle filter has seen the largest number of applications to ABMs [10,14,15,20,21] but requires potentially thousands of ensemble members ('particles') which can be prohibitively computationally expensive. Parametric techniques such as the Unscented Kalman Filter (UKF) and Extended Kalman Filter [5] instead estimate model states explicitly through the Kalman Gain equations [22]. The UKF, which is the technique adopted here, aims to use the bare minimum number of particles, dubbed 'sigma points', to greatly increase efficiency at the expense of a minimal loss in prediction accuracy. If the model state of n agents is assumed to be Gaussian, it can been shown that only $4n + 1$ sigma points [23] are needed to estimate the mean and covariance of agent positions.

Previous research has shown potential for the application of the Unscented Kalman Filter (UKF) to ABMs [24], particularly in dealing with problems such as partially observed data. The problem with the UKF is its assumption of Gaussian innovations [5]. The filter on its own is limited to a very specific set of agent attributes and requires further augmentation to apply real data to ABMs. Research is being done into the application of the Kalman Filtering to continuous non-Gaussian data [25] and discrete variables [26–28] but little is being done [29] on estimating a mixed state of both categorical and Gaussian variables with the

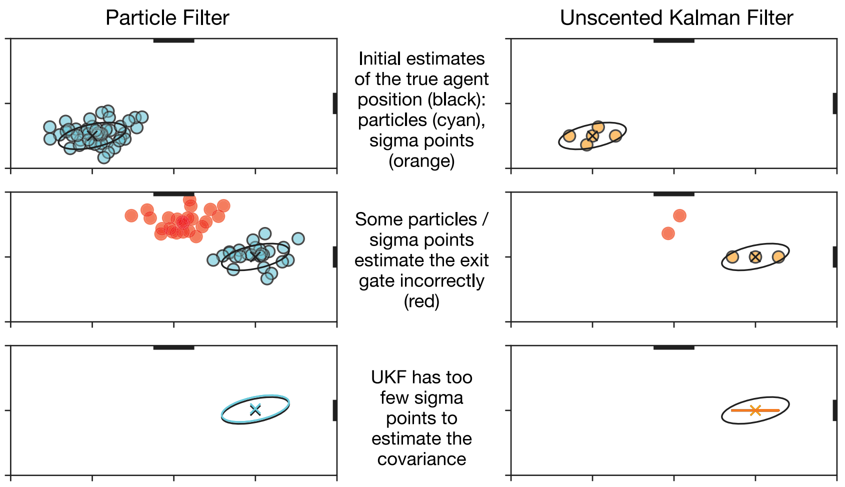

Particle Filter
Unscented Kalman Filter
Initial estimates of the true agent position (black): particles (cyan), sigma points (orange)
Some particles / sigma points estimate the exit gate incorrectly (red)
UKF has too few sigma points to estimate the covariance

Fig. 1. Demonstrating why the UKF cannot be used to infer exit gates using a single agent's 2D coordinates. (1) A sample of particle filter particles (cyan) and UKF sigma points (orange) are generated about the agent position (black). (2) Each member of these ensembles is assigned randomly to an exit gate and processed through the ABM stepping mechanism. Any that choose the incorrect top gate (red) are discarded. (3) The remaining sigma points/particles are then used to calculate estimates for the new agent position mean and covariance. It is clear that the particle filter keeps a sufficient sample of particles to accurately estimate the model state. The UKF does not have a sufficient number of sigma points and cannot correctly estimate the covariance.

UKF. This has been attempted for use in agent-based models using particle filters [e.g.10,14] in a brute force fashion, but the incorporation of discrete values into the UKF state space can quickly cause numeric instability [30]. Furthermore, the sigma points of the UKF are interdependent, so every sigma point is required to have the same values for its categorical parameters to provide a sufficient sample for estimating the model state. If any of these sigma points have a different combination of categorical parameters then immediate sample degeneracy occurs and the model estimate quickly diverges; see Fig. 1.

To estimate categorical states, model selection techniques [31] are typically used, whereby each model (e.g. a UKF in this case) represents a different combination of categorical parameters. Choosing the optimum set of categorical parameters is achieved by selecting the best fitting model. Traditionally, this is done with respect to some criterion based on model likelihood [31,32] and has been extended to data assimilation via multiple model inference [33–35]. However these techniques struggle with respect to in their applications to ABMs due to the need for high- and constant-dimension parameters (as discussed in Section 1).

Markov Chain Monte Carlo (MCMC) techniques [36–38] have been applied to this problem with success being seen in Reversible Jump MCMC (RJMCMC) [11]. RJMCMC performs well in sampling both the choice of model and its associated parameters of different dimensions simultaneously using a modified Bayes factor. Current implementations of this approach are executed offline due to the requirement of multiple iterations over the entire ABM. Incorporating any new information requires re-calibrating for all previous data. Attempts at sequential applications of RJMCMC have been made [38] but these techniques are not efficient enough for real time application due to the high dimensionality of ABMs. This paper proposes to combine RJMCMC with the UKF to produce an efficient algorithm that can apply multiple model inference to ABMs in real time.

3. Methods

3.1. Reversible Jump Unscented Kalman Filter (RJUKF)

The main innovation offered by this paper is the introduction of the Reversible Jump Unscented Kalman Filter (RJUKF). We propose a combination of Reversible Jump Markov Chain Monte Carlo (RJMCMC) [11] with the Unscented Kalman Filter (UKF) as a means of conducting data assimilation utilising an efficient algorithm, the UKF, to estimate both categorical and Gaussian agent attributes in real time.

For example, in the *StationSim* agent-based model that simulates agents crossing a train station (discussed in detail in Section 3.3), each agent's destination that it moves towards is stored as a categorical parameter and its current spatial location in the environment is a $2-d$ Gaussian variable. The UKF alone cannot search for the correct categorical parameter combination (i.e. the correct destinations for the individual agents), so the natural extension to this is to use multiple UKF filters at once, each assimilating the Gaussian positions of agents but with different combinations of categorical parameters (destinations).

In the current implementation, two UKFs are used. In theory more filters allows for better scanning of the categorical parameter space but there are two major issues with this. The RJUKF is designed to be as efficient as possible for real time application and using many filters can impact performance. To the authors knowledge there are no guidelines available for determining the optimum number of UKFs that give the best efficiency to efficacy ratio. There are also issues of robustness when comparing more than 2 filters. Selection is typically based on information criteria [39], although these may struggle to find the 'best fitting' filter [32]. Techniques such as Bayesian model averaging may be applicable here [40].

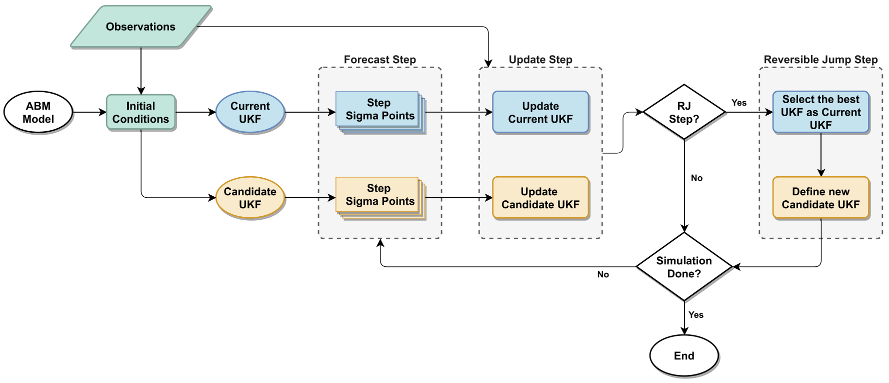


Fig. 2. Flowchart of data assimilation process using a Reversible Jump Unscented Kalman Filter.

Fig. 2 illustrates the process of running a RJUKF with an agent-based model. Consistent with typical data assimilation approaches, the method consists of *forecast* and *update* steps. The forecast step involves running a simulation forward (an agent-based model in this case) up to the point that some new observational data become available. In effect this creates a prior estimate of the next system state. The *update* step updates the forecast using new observations. The innovation here is to add a new 'Reversible Jump' step. Rather than using a single filter to conduct data assimilation, two filters are compared. The *current* filter contains model instances with previously established categorical parameter combinations from either initial conditions or a previous time step. The *candidate* filter contains model instances with Gaussian agent states that are identical to those in the current filter, but with a new set of categorical parameters (discussed in Section 3.1.1). These two filters are fitted to data and compared by some acceptance probability (discussed in Section 3.1.2), discarding the filter with the lower likelihood. In short, a standard data assimilation filter (the Unscented Kalman Filter in this case) is used to assimilate data for the model's Gaussian parameters and the RJ step provides a means of assimilating on the categorical parameters.

To summarise, the algorithm is outlined as follows:

(i) In the *initialisation step*, a 'current' UKF is generated using some prior estimate of the unknown categorical parameters. These estimates typically come from either initial conditions or some previous RJUKF step. A candidate filter is also generated using some new randomly drawn categorical parameters. Both filters contain the same ensemble of 'sigma points' which are individual instances of Gaussian parameters for the underlying agent-based model.
(ii) In the *forecast step*, the filters are iterated (their sigma points are stepped forwards in time) to gain two forecasts of the Gaussian agent states given their categorical parameter combinations.
(iii) In the *update step* (data assimilation step), when the filters are confronted with observations, both of these forecasts are assimilated to get final estimates of the agents' Gaussian states.
(iv) In the *reversible jump (RJ) step*, these estimates are then compared by generating some acceptance probability ψ. With this probability the best filter (candidate or current) is accepted as the new current filter and the forecast step starts again. Note that the RJ step does not necessarily need to take place after every update step (as discussed below).

We use the 'identical twin' experimental framework [14], which has been used regularly in similar ABM data assimilation work [3,10,14,15]. It involves first running a single instance of the agent-based model to generate pseudo-real crowd behaviour data, storing the locations of the agents as the model evolves, and then using these data to create 'real world' observations. As the simulation is stochastic and the initial parameters are not known, the task for data assimilation is to infer the current model state and parameters from the 'real world' observations.

Following this broad overview of the method, the following sections will discuss the approach in detail: the remainder of this section outlines the specific details of the RJUKF implementation; Section 3.2 provides an overview of the Unscented Kalman Filter; and Section 3.3 outlines the agent-based model used (*StationSim*) to test the proposed method.

3.1.1. *Drawing new categorical parameters*

When initialising a new candidate filter in the Reversible Jump (RJ) step (see Fig. 2), the categorical parameters for each agent in each of the filter's agent-based model instances ('sigma points') need to be estimated. This choice could be completely random, but the number of possible categorical parameters combinations is usually huge. For example, if the model has n agents and one categorical parameter with G values, the number of possible combinations is n^G, that is, the probability of selecting the correct combination of parameters is usually very low. Therefore, to improve the RJUKF efficiency a matrix of transition probabilities $P = p_{jg}$ is defined at each time step for each unknown categorical parameter, where p_{jg} is the probability the jth agent will draw the parameter value g. The point is to reduce the number of potential parameter combinations by estimating agent behaviour and removing poor choices of parameters. We explain how this is implemented for the *StationSim* model specifically below.

Drawing new combinations of parameters in the RJ step does not need to take place after every update step. If the parameters are relatively easy to estimate then it may be more efficient to perform multiple data assimilation updates before performing the reversible jump step and recalculating categorical parameters. However, the size of the reversible jump window, R, should be a multiple of the data assimilation window, f, to ensure that the reversible jump is made using assimilated estimates of the model state rather than pure prediction. For example, if $f = 5$ then the data assimilation update occurs after every 5 model iterations so R must be some multiple of 5 (e.g. if $R = 15$ then it occurs once in every 3 data assimilation updates, or once every 15 model iterations).

An additional restriction is that new categorical parameters combinations are drawn stepwise. Only some subset $s \leq n$ of the agents draw new categorical parameter combinations. Trying to redraw all parameters at once in larger models can result in poor candidate filters that are never accepted. Conversely if too few parameters are redrawn then the algorithm will struggle to find every correct value in time. This method of drawing new parameter combinations provides a good stopgap solution but broader scanning techniques [41,42] may be necessary for larger ABMs.

Each matrix of transition probabilities should be produced based on the characteristics of the underlying model. Here, the challenge in the *StationSim* model is to correctly determine where an agent will leave the environment; its 'destination' or 'exit gate'. There are 11 possible gates and each agent can choose any gate as its desired destination. To produce the matrix of transition probabilities $P = p_{jg}$ for the exit gate parameter, we use each agent's direction of movement, assigning a larger probability to gates that are within some cone of vision – defined by an arc segment some θ radians wide – in front of the agent. The lengths of each gate contained within this segment are then normalised to sum to 1 giving an empirical distribution of each of the agent's likely gates. For more complex (non-linear) paths, this prediction of an agent's exit gate can easily be extended to more elaborate methods.

3.1.2. Acceptance probability

In the Reversible Jump step, an acceptance probability is required to determine whether to keep the current filter or replace it with the new candidate. Traditionally, model selection techniques based on likelihood would be used such as Deviance Information Criterion [31] but this is known to be problematic for high/variable dimension cases [39]. Instead, a metric that is based on Bayes factors [11], is used to determine whether to keep the current or candidate model:

$$\psi_k = \min\left\{1, \frac{p(\zeta'_k|z_k)}{p(\zeta_k|z_k)} \times \frac{q(m_k|m'_k)}{q(m'_k|m_k)} \times \frac{r(u'_k)}{r(u_k)} \times |J|\right\}, \tag{1}$$

There are four terms in Eq. (1) to calculate. The first term uses the posterior densities $p(\zeta'_k|z_k)$ and $p(\zeta_k|z_k)$ where ζ_k are the current model parameters, i.e. the mean and covariance of agent Gaussian positions, and similarly ζ'_k are the candidate model parameters. The joint posterior distributions for these parameters can be hard to find in general but the UKF has the advantage that it is calculated in its update step. Assuming we calculate Gaussian posterior distributions for each filter with parameters μ_{xk}, Σ_{xk} and μ'_{xk}, Σ'_{xk} respectively, the ratio of densities can be written as

$$\frac{p(\zeta'_k|z_k)}{p(\zeta_k|z_k)} = \frac{|\Sigma_{xk}|^{\frac{1}{2}}}{|\Sigma'_{xk}|^{\frac{1}{2}}} \times \frac{\exp(-\frac{1}{2}(z_k - \mu'_{xk})^T \Sigma'^{-1}_{xk}(z_k - \mu'_{xk}))}{\exp(-\frac{1}{2}(z_k - \mu_{xk})^T \Sigma^{-1}_{xk}(z_k - \mu_{xk}))}. \tag{2}$$

We also require the transition densities $q(m_k|m'_k)$ and $q(m'_k|m_k)$, i.e. the probabilities of moving from the current model to the candidate and back respectively. These quantities determine the likelihood for the current and candidate filters choices of categorical parameters given observation data. The categorical model parameters are captured in the vector m_k. In *StationSim*, this is the vector of n exit gate choices defined as $g = (g_1, \dots, g_n)$. Using the transition probability matrix P, then these densities are

$$\frac{q(m_k|m'_k)}{q(m'_k|m_k)} = \prod_{j=1}^{n} \frac{p_{j,g_j}}{p_{j,g'_j}}. \tag{3}$$

Multiplying small probabilities will often lead to numeric stability issues in high dimensional cases. These densities can however be useful in the easy rejection of more unlikely candidate models. Using stepwise selection makes this significantly more stable as a large number of terms in this product will cancel out.

The final two terms in Eq. (1) are the proposal densities $r(u'_k)$ and $r(u_k)$ and Jacobian J given auxiliary variables u that allow for changes in model dimensions. Here we assume that all possible UKF filters have model parameters with the same dimensions. In this case, both of these terms are trivial and simplify to 1. This is a special case of RJMCMC known as the Random Scan Hastings [43] algorithm. For the case where agents do leave StationSim these quantities are defined in Appendix B.

With ψ_k calculated, a Bernoulli random variable $r = \text{Bern}(\psi_k)$ is drawn. If $r = 1$ the candidate filter is chosen over the current model and if $r = 0$ the current filter is kept.

3.2. Data assimilation with an unscented kalman filter (UKf)

The previous section presented the RJUKF method as a means of maintaining multiple filters to better estimate categorical variables. This section outlines the UKF method itself, which is used to estimate the Gaussian agent parameters (in this case their spatial location in a train station) by combining 'real world' observations with predictions from the agent-based model. The UKF is a well-known method, but one that has rarely been applied to agent-based modelling, likely due to the inability to handle non-Gaussian variables.

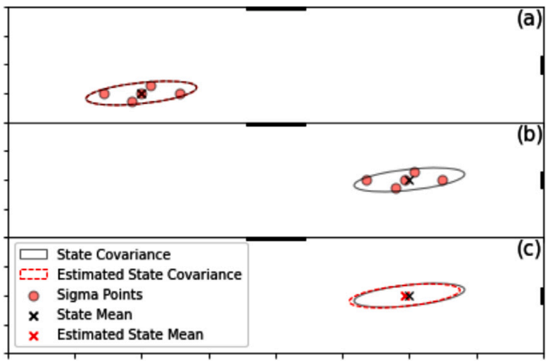


Fig. 3. An example of the UKF forecast step for a single agent position. Sigma points are generated about the current model state (a), then processed forwards in time (b) and used to estimate the new model state (c). Note that in the full application the UKF will be estimating the 2D positions of all pedestrians in the system, not just one.

In this paper, the UKF implemented by [24] is used. A simplified explanation of the method is included here and for full details see [5,44,45] and Appendix A. Consider that our model state can be fully described at time k with the state vector, x, and that it can be updated with a stepping function f:

$$x_{k+1} = f(x_k). \tag{4}$$

In effect, f iterates the model forward by one time step; such *step* functions are common in agent-based model implementations. Now consider that some observation data become available at time $k+n$ (i.e. some interval n after the last time we updated the model state) and we want to update x to better reflect the information contained in these observations of the real system. As discussed in Section 3.1, data assimilation proceeds via two steps, which are repeated each time new data become available [5,46]:

Forecast An ensemble of *sigma points* are computed from the mean and covariance of the posterior distribution for the current model state x_k. They are evolved using f to give a sample of forecasted states at time $k+n$. The sigma points are then used to estimate the true mean and covariance by applying the Unscented Transform [23]; see Fig. 3.

Update Following the arrival of observations at time t_{k+n}, standard Kalman filter practice [22] is used to update the model state x_{k+n} and its mean and covariance. This is taken as the new posterior distribution of the current model state, i.e. our best guess of the current state of the underlying system. This posterior estimate is fed back into the forecast step to calculate new sigma points.

In addition to deciding on the types of sigma points used and their weightings (as detailed in Appendix A), the identical twin experimental approach requires two additional parameters. The data assimilation window f determines how often data assimilation is performed on *StationSim*. For every time step, the StationSim model evolves forwards producing some observation data. The UKF will also evolve forwards predicting agent positions but not incorporating these observations. On every fth time step the UKF does incorporate observation data in order to produce assimilated position estimates. The window size can be varied such that a larger window induces larger error in the prediction of the UKF. The longer it goes without assimilating real data the more prediction will diverge from the truth. Observation noise is also required as, without noise, observations would exactly match the pseudo-true state which is infeasible in reality. Uncertainty is induced into these observations by adding Gaussian distributed noise with mean 0 and variance σ^2. Larger values of σ^2 imply noisier observations and worse 'sensors' that are more difficult for the UKF to assimilate. Values of these parameters must be calculated separately for each model used. These values must be chosen such that the UKF is able to assimilate position data with better accuracy than pure prediction or observation alone. Calibrated values for the StationSim model are derived later (see Section 4.1) as 5 and 0.5^2 for the assimilation window and observation noise respectively. Other parameter values are listed in Table 1.

3.3. StationSim

Grand Central Station Simulation (hereafter '*StationSim*') is an agent-based model that replicates simple pedestrian movement through the central atrium of the Grand Central Terminal (GCT), New York City. A raw data set inferred from a video sequence recorded at GCT is available online[1], and the *StationSim* environment is designed to match the GCT camera's field of view. The model environment is a rectangular subsection of the atrium's lower floor, with 53 m width and 50 m height, 11 entrance/exit gates around its perimeter and a large circular obstacle in its centre; see Fig. 4. This section will briefly outline the main components of the model; full model details are included in Appendix C.

[1] GCT data are available at http://www.ee.cuhk.edu.hk/~xgwang/grandcentral.html

Table 1
Main parameters used in the experiments.

Parameter	Symbol	Value
Number of agents	n	$[10, 20, 30]$
Number of gates	G	11
Number of repeat experiments	N	50
Observation noise.	σ^2	0.5^2
Data assimilation 'window' (time steps)	f	5
Tuning parameters	α, β, κ	$[0.3, 2, 0]$
Reversible Jump window (time steps)	R	$[5, 10, 20]$
New stepwise candidate gates (number of agents)	s	5
Agent vision angle (rads)	θ	$\pi/3$

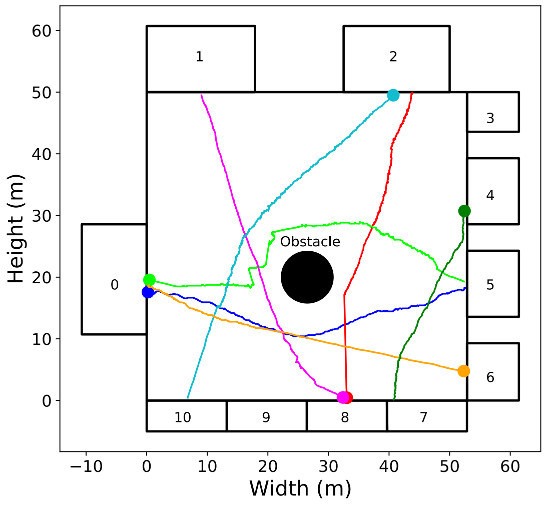


Fig. 4. An outline of model environment over the Grand Central Station atrium in New York, including the central information kiosk obstacle. The solid lines represent trajectories of seven real pedestrians inferred from a video recorded at GCT. The circles mark the beginning of each pedestrian trajectory.

Upon initialisation, n agents are created. We assume that their entrance gate and time of entry are known from the pseudo-truth data – people are often recorded entering a station as they pass through ticket barriers – but as we cannot know a person's destination the agents are randomly assigned an exit gate. Each agent has a desired speed which is also chosen randomly from a truncated Gaussian distribution. In each iteration, agents try to move linearly towards their desired exit gate. However, if the way ahead is blocked, either by another agent or by the environment, the agent will not be able to proceed towards the destination. When such 'collisions' occur, the agent makes a random binary choice to overtake the obstacle by trying to move left or right. If the drawn position is available, the agent moves sideways, otherwise, the agent stands still until the next iteration. This random choice causes crowding to emerge at different times and locations each time the model is executed. The simulation ends when all agents have reached their desired exit gate.

The model is a relatively simple description of pedestrian movement that does not attempt to reflect the realism present in state-of-the-art approaches [1,2,8,9,47]. However, it is able to represent the main behaviours observed through the GCT data. Fig. 4 illustrates some real pedestrian trajectories. According to the available data, approximately 70% of pedestrians have similar behaviour to that shown in Fig. 4; pedestrians enter through one of the gates and walk towards the exit gate avoiding obstacles when necessary. The remaining pedestrians have more complex trajectories with destinations that appear to change throughout the journey. With the RJUKF method we hope to correctly assign the exit gate in each reversible jump step and correctly predict even the more complex trajectories.

In addition to outlining the model environment and agent behaviour, GCT data are also used to calibrate the simulation parameters. The raw trajectories were highly fragmented with portions of missing observations [9], so a laborious process was performed to manually designate a unique trajectory to capture the full movement of an individual across the concourse. Following this, the average pedestrian speed was found to follow a Gaussian distribution with mean of 1.6 m/s and standard deviation of 0.6 m/s. The minimum observed speed was 0.05 m/s. Although it is difficult to extract patterns for the entrance/exit gates, considering that 121 combinations are possible, two characteristics were observed in the processed data: (i) no pedestrian leaves the station through the same gate that they enter; and (ii) only about 1% of pedestrians leave the station through gates that are on the same side as the gate through which they entered. Therefore, the initial choice for an agent's exit gate is drawn uniformly from the set of gates that are on different sides of the agent's the entrance gate. These estimates are then updated in the RJ step based on the agent's subsequent movements, as discussed in Section 3.1.1. Each simulation step is equivalent to one video frame, i.e. 0.04 seconds.

3.4. Error metrics

Two error metrics are required: one used when estimating exit gates and another for comparing agent positions. To calculate the exit gate error – i.e. whether a filters' *StationSim* instances have correct exit gate choices – we define an indicator function, γ_{jk}, that assigns 0 if an agent's current choice of gate is its correct gate and 1 if it is not. Given current exit gates M and true gates $\mathcal{M}$ for the jth agent at time k this is defined as:

$$\gamma_{jk} = \begin{cases} 1 & \text{if } M_{jk} = \mathcal{M}_{jk} \\ 0 & \text{if } M_{jk} \neq \mathcal{M}_{jk}. \end{cases} \tag{5}$$

Taking the sum of these values at each time point gives an indication of how the RJUKF converges towards the true gates. This value is bounded between 0 where all exit gates are correct and n where all gates are wrong.

$$\gamma_k = \sum_{j=1}^{N} \gamma_{jk}. \tag{6}$$

With respect to the positional error, consider that we repeat an experiment N times where the ith experiment has n_i agents and t_i time steps. For some agent $j \in 1,\dots,n_i$ at time $k \in 1,\dots,t_i$ we analyse the efficacy of the UKF using the Euclidean distance between each agents' 'true' (x_{jk}, y_{jk}) position – i.e. the position of their corresponding pedestrian in the pseudo-truth data – and UKF predicted $(\hat{x}_{jk}, \hat{y}_{jk})$ position. This provides an $n_i \times t_i$ matrix of distances:

$$d^i_{jk} = \sqrt{(x_{jk} - \hat{x}_{jk})^2 + (y_{jk} - \hat{y}_{jk})^2}, \tag{7}$$

with each row j representing the spread of agent errors at a single time point and each column k representing an agent's error over time. For the ith experiment, we calculate the error vector $\tilde{x}^i = (\tilde{x}^i_0, \tilde{x}^i_1, \dots, \tilde{x}^i_{n_i})$, where

$$\tilde{x}^i_j = \underset{k\in 1,\dots,t_i}{\text{median}}(d^i_{jk}) \tag{8}$$

i.e. the jth element of $\tilde{x}^i$ is the median error for the jth agent. We use medians here to avoid bias caused by taking the means of heavily right skewed agent error distributions.

For multiple runs we calculate the grand median error vector $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots,$ $\bar{x}_N)$ where the ith element represents the median of median agent errors for the ith model run:

$$\bar{x}_i = \underset{j\in 1,\dots,n_i}{\text{median}}(\tilde{x}^i_j). \tag{9}$$

We then use this as a sample to gain a measure of the UKF's general efficacy given certain parameters. We use both the raw sample and the sample mean for boxplots and choropleths respectively.

4. Experiments

Two experiments are conducted to illustrate the efficacy of the proposed method. The first establishes the feasibility of applying the UKF to an ABM. The final experiment demonstrates the application of the UKF with categorical variables; i.e. using the RJUKF method. In the first experiment we assume that the agents' destinations are known, so the filter only needs to correct for the noise introduced as agents collide and deviate from their otherwise straight paths. In the second experiment we relax this assumption and use the RJUKF method to estimate the agents' exit gates as well. All other parameters were calibrated as per Section 3.3.

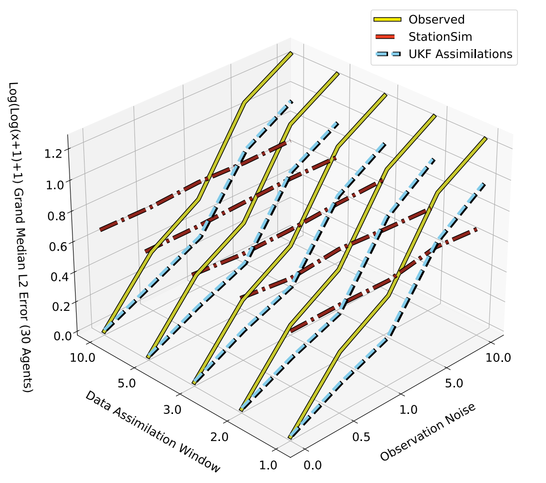
4.1. Experiment 1 – Filtering, observation and prediction

Before testing the full RJUKF method, this experiment determines the improvement that the Unscented Kalman Filter (UKF) offers by correcting deviations in the agents' (x, y) locations, assuming that the agents' destinations are known. It will contrast the filtering approach to an entirely data-driven approach (i.e. relying purely on noisy observations with no underlying model) and an entirely model-based approach (i.e. relying on *StationSim* predictions with no assimilation). It also explores the impacts of different data assimilation window sizes, f, i.e. the number of model iterations between data assimilation updates. Using these values, we establish a suitable benchmark under which the UKF performs well in estimating Gaussian parameters, before moving on to test the RJUKF as a means of estimating categorical parameters as well in Experiment 2.

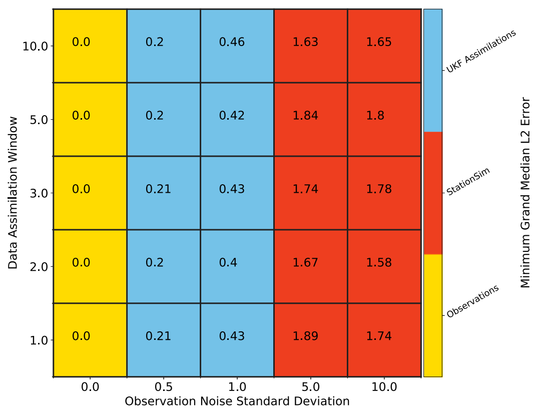
For each model run we calculate the grand median distance between each 'true' agent position and its estimate and take a further scalar median of 30 model repetitions[2] (see Section 3.4). Fig. 5 presents these scalars over varying noises and assimilation rates. We assume that the noise added to the pseudo-real observations (sensor noise) and the uncertainty associated with the individuals in *StationSim* (the process noise) are treated equally, so the UKF relies on both predictions and observations performing similarly well[3].

[2] $N = 30$ model repetitions is sufficient to capture the variability in the results within a reasonable computation time

[3] In practice, noise assumptions can be adjusted to improve performance, but under high dimensional scenarios such as this it can prove difficult to optimise. This provides a strong motivation for further adaptions to the UKF particularly adaptive filtering [48].



(a) The error associated with all three approaches (observation, prediction, or UKF).



Data Assimilation Window	0.0	0.5	1.0	5.0	10.0
10.0	0.0	0.2	0.46	1.63	1.65
5.0	0.0	0.2	0.42	1.84	1.8
3.0	0.0	0.21	0.43	1.74	1.78
2.0	0.0	0.2	0.4	1.67	1.58
1.0	0.0	0.21	0.43	1.89	1.74

(b) The error of the best performing approach.

Fig. 5. Errors of estimated agent positions against 'true' positions comparing: (1) observations in isolation; (2) *StationSim* predictions in isolation; and (3) UKF predictions (assimilation of *StationSim* predictions and observations) with different data assimilation window sizes and levels of observation noise.

Fig. 5(a) compares the error of the three approaches (pure observation, pure model, or UKF) for each noise/data assimilation window pair. Similarly, Fig. 5(b) shows the error of just the *best performing* metric. It is evident that when there is no observation noise then the observations in isolation give the best estimate of the pseudo-true system evolution (the yellow area in the left of Fig. 5(b)). Conversely, when observation noise is very high then the *StationSim* prediction provides the best (albeit relatively poor) estimate because it is not confounded by noisy observations (the red area to the right of Fig. 5(b)). The UKF gives a more accurate prediction than the model or the observations in isolation when the observation noise is not extreme (the blue area in the middle of Fig. 5(b)). For the remaining experiments we set the data assimilation window size and measurement noise to be 5 and 0.5 respectively. These values provide a reasonable compromise for future experiments under more uncertain conditions.

4.2. Experiment 2 – RJUKF

In the previous experiment, the UKF was tasked with estimating the true pedestrian positions. This will be important when collisions or other factors cause pedestrians to deviate from an approximately straight trajectory. However, the filter does not attempt to estimate the desired locations of the agents. These were hard-coded in Experiment 1 so all agents were moving towards the 'correct' destination; i.e. the same gate as the pedestrian that they are simulating. Relaxing this assumption would pose a substantial problem for the filter because there is little value in slightly adjusting the position of an agent if the agent is moving towards a different exit than that of the pedestrian it is simulating. Therefore in this experiment the assumption is relaxed so an agent's destination is not known and the RJUKF method is used to estimate the agents' destination as well as their position.

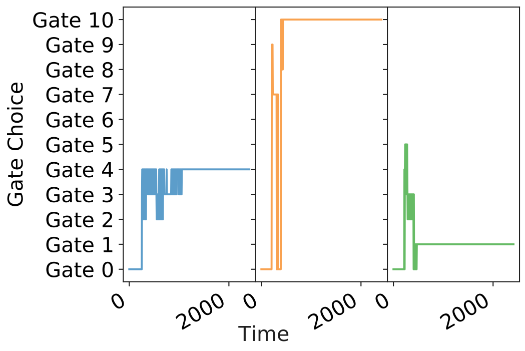


Fig. 6. The choice of exit gate for each agent over time.

Results are presented in two parts. First, diagnostics for a single run of the RJUKF are taken. This examines the general results from the algorithm as well as individual error metrics. Second, multiple run diagnostics are presented for an overall view on how well the RJUKF estimates exit gates on *StationSim*.

4.2.1. *Single run diagnostics*

A single model run for *StationSim* is performed with $n = 10$ agents, $f = 5$ data assimilation window and $R = 5$ jump window. Hence a *update* step takes place after every 5 iterations, and a reversible jump takes place after this (as per Fig. 2). This small number of agents limits the number of interactions between agents that occur so uncertainty in the model arises less from the collisions between agents and more from uncertainty about the agents' correct destination gates. A subset of 3 example trajectories in Fig. 6 shows how the predicted exit gate choice varies over time for each agent. Note that as the agent starts their journey there is no information to inform the gate prediction so the prediction defaults to gate 0, but this is quickly revised as the agent moves towards their destination. Estimates of agent positions typically fall into three categories depending on how quickly an agent's correct gate is found:

(i) Some agents find their correct gate quickly giving minimal error dependent on model noise (blue).
(ii) Some agents take longer to find their gate and the prediction shifts from one gate to another (orange). This is usually due to either a contention between numerous gates (e.g. when two adjacent gates have similar probabilities of being the correct one) or failure to draw the correct gate by chance.
(iii) Some paths take a long time to converge to the correct gate (green) and often snap quickly towards the correct exit gate when discovered. This usually happens with agents that move quickly through the model, giving limited chances to scan for the correct gate. Even when each agent draws a new gate ($s = 10$) at every jump window, candidate models with a good gate for one given agent can be rejected due to some poor choice of gate for another agent, skewing the acceptance probability. It is rare to see these poor choices of exit gate in this model. Due to the low levels on non-linear behaviour it generally becomes quickly obvious where an agent is going.

4.2.2. *Multiple run results*

We now describe the results of $N = 50$ *StationSim* runs. In this experiment the agent population, n, and reversible jump window, R, are varied. The larger population sizes in the experiments, where $n \in \{10, 20, 30\}$, means that collisions are likely so the UKF must estimate the destination gate as well as the precise agent position. To estimate the rate of convergence, we use the gate distance metric (Eq. (6)). The mean of the gate distance metric and 95% confidence interval over the 50 model runs for each population and jump window combination are then plotted in Fig. 7. Note that the gate distance metric does not normalise by the number of agents, so increasing the number of agents in the simulation will naturally increase the apparent error. More important than the absolute error per size of agent population is the difference in error by jump window.

The results show that the RJUKF works well for 10 agents at all jump windows. It is rare the RJUKF will predict every single gate correctly (0 error) as agents may often have multiple gates in their cone of vision. However, as the population increases it quickly becomes clear that the RJUKF finds fewer correct exit gates. This stems from a relatively low number of agents drawing new exit gates ($s = 5$) at each reversible jump step. Not enough agents are choosing new gates resulting in slower convergence. For large populations ($n = [20, 30]$) increasing the number of agents who draw new gates ($s > 10$) can resolve this. However for very large populations $n \gg 100$ often no step size is large enough and better methods of drawing new gates are required. Too many agents drawing new gates results in the candidate filter being unlikely to draw a 'good' combination of gates and never being accepted.

To observe the error in agent position, the grand median error metric (9) is calculated across all 50 model runs for each population and jump window pair; see Fig. 8(b). For jump window $R = 5$ there is near constant error over the populations. The RJUKF quickly finds the correct gates, tracking agents with error proportional to the observation noise. These errors align with those seen in other work [24]. For $R = 10$ the position error increases with population. The RJUKF scans more slowly and takes more time to find the

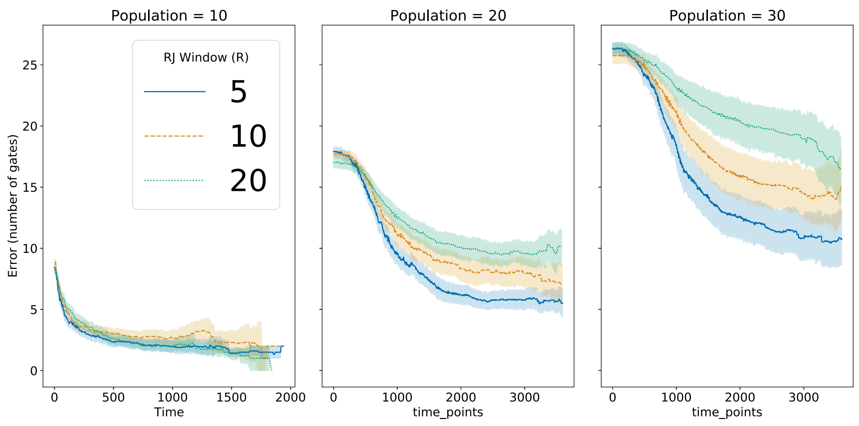
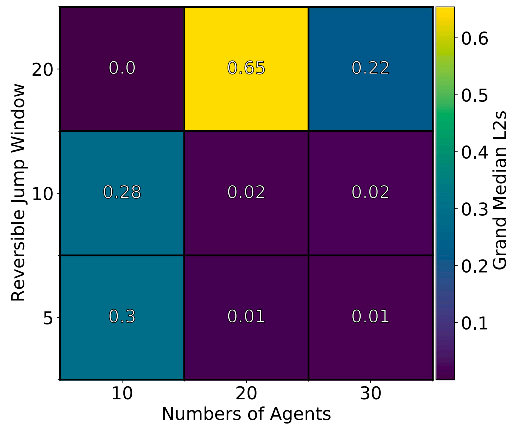
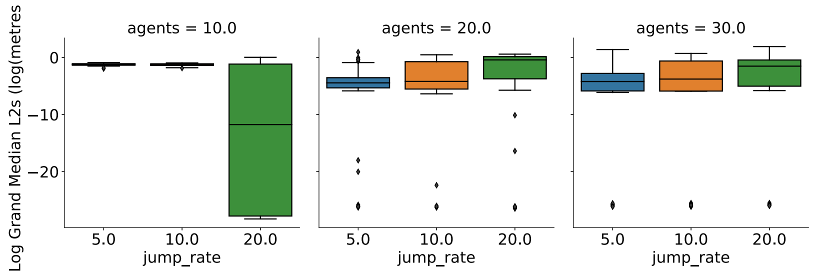


Fig. 7. Convergence to correct exit gates over varying populations and jump windows. For each population there are three lines for 5 (solid blue), 10 (dashed yellow), and 20 (dotted green) jump windows respectively.



(a) Choropleth



(b) Boxplot

Fig. 8. Error in the agent positions for varying populations and jump windows. Overall the error increases with higher populations and jump windows.

correct gate combinations for a larger number of agents. If an agent's correct gate is not found quickly, it diverges from the truth giving a large error in position.

For the highest jump window $R = 20$ the RJUKF performs the least well for all populations, although still better than would be the case were there no gate estimation. Regularly agents do not converge to their gate or do so very late giving large errors in position. It is clear from Fig. 8(b)(b) that there are strong outliers for all populations but the lowest population $n = 10$ is worst affected. There are a number of potential reasons for this. First, a lower population model is more sparse. Agents move through the model quickly without colliding and a wrong choice of gate leads to a larger error. This is compounded by the fact the RJUKF has less time to look for each agent's correct gate. For higher populations with regular collisions and crowding, agents move more slowly and find their correct exit gates before moving too far. Second, smaller populations are less robust to outliers. Using medians alleviates this problem somewhat, but only a small number of agents need to perform poorly to inflate the overall error.

As an aside, it should be noted that the results of this experiment cannot compared directly to those of experiment 1. This is because in experiment 1 the exit gates of the agents were known, whereas here they need to be inferred. However, the size of the errors at the beginning of the simulation illustrated in Fig. 7 provide an illustration of how poorly the UKF would perform if agents' exit gates were chosen randomly. The error reduces substantially as the UKF is able to correctly estimate the exit gates; this reduction would not take place without the gate inference.

5. Conclusions

This paper has demonstrated how the combination of an Unscented Kalman Filter (UKF) and elements of the Reversible Jump MCMC algorithm can be combined to create an efficient algorithm that can scan a set of filters (parameterised by categorical variables) and filter over continuous variables simultaneously. This offers an advantage over previous attempts to conduct data assimilation on agent-based models that have largely used particle filters [10,14,15,20] as the UKF can perform well with a considerably smaller ensemble size. Some previous work has demonstrated the successful of application of the UKF to crowd dynamics [49] but there is very little research using the UKF to estimate categorical variables due to the requirement of Gaussian innovations. Hence the work here presents a useful and novel contribution to the literature.

The RJUKF method was demonstrated on a simulation of a crowd of pedestrians, *StationSim*. The model was designed as a relatively simple description of pedestrian movement, and it is likely that more developed pedestrian models – such as [1,2,8,9,47] – will more faithfully represent the true underlying pedestrian dynamics. Although the behaviour of the agents in the model was relatively simple, the model environment was nevertheless designed to reflect Grand Central Terminal in New York City. The real pedestrian data were not used in the data assimilation experiments directly, but the underlying agent-based model was calibrated using real data. The results demonstrate that the RJUKF is able to successfully estimate the *exit gate* categorical agent parameter and, as such, more accurately estimate the true behaviour of the underlying system. Having demonstrated the utility of the RJUKF approach, immediate future work will begin to experiment with the method's applicability to a wider range of applications, including more advanced pedestrian models as well as entirely different systems for which the combination of agent-based modelling and data assimilation offers the most appropriate simulation approach.

There are, of course, some drawbacks. The bottleneck of this method is currently the stepwise fashion in which new exit gate combinations are drawn. This technique can struggle to find the global optimum in higher dimensional cases. As such, scanning techniques such as simulated annealing and Hamiltonian Monte Carlo [41,42,50] are required. Other data assimilation techniques such as the particle filter [21] provide faster convergence but struggle with efficiency in real-time due to the large number of particles necessary. There may be a compromise with hybrid techniques such as the unscented particle filter [51] that need fewer particles due to "smarter" choices of exit gate.

Furthermore, the calculation of agent gate distributions using a cone of vision can be naive. A generalised method for drawing new gates is needed particularly for non-linear paths and more complex topographies. There are a number of potential alternatives including time series and machine learning techniques [52,53]. It would also be desirable to obtain theoretical results relating to observability, i.e. whether or not the agents' positions are sufficient to infer their exit gates. However, ABMs are typically formulated as computer programs, rather than time dependent systems of equations, which present a significant challenge for the standard control theory paradigm [54]. With respect to the input data, in the current implementation it is assumed that pedestrians can be tracked as they move through the environment. Such data are relatively rare for real environments and they have numerous potential ethical and privacy implications. It would be better to use aggregate data (such as estimates of crowd density, or anonymous counts of people entering or leaving different parts of the environment) and as such methods to map from these simple data domains to more complex model domains [e.g.20], could be integrated. This will be essential as the methods mature and applications move towards real, 'live' simulations [55].

Despite these problems, this approach shows promise in overcoming some of the limitations in the application of DA and ABMs. Future work will: (1) aim to validate UKF/RJUKF results by applying the algorithm to real data (i.e. moving beyond the identical-twin experiment). This will require augmenting the UKF to estimate further attributes including agent speed and observation association and possibly improvements to the *StationSim* model, or the adoption of a more comprehensive crowd simulation, to reduce model discrepancy. (2) Implement extensions to the RJUKF methodology, such as an improvement to its model scanning ability – e.g. through the use of more than 2 UKF filters or more broadly through scanning MCMC techniques – or a generalisation to more complex discrete data with more complex distributions including count and aggregate data. (3) Implement further efficiency improvements including a variable dimension RJUKF that could be useful in scaling to larger ABMs. (4) Conduct a comparison of the RJUKF with other similar real-time scanning techniques particularly their efficiency to accuracy ratio.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Ukf definition

To define the UKF, the dynamic state space model is used. Let $x_k \in \mathbb{R}^a$ denote the true a-dimensional model state at the discrete observation time $t_k, k \in 0, 1, 2, \ldots$. This is the state of agent attributes which we are interested in knowing but may not directly observe. We assume that the model state is updated according to the difference equation,

$$x_{k+1} = f(x_k, q_k), \tag{10}$$

where the process noise q_k is a random variable with known probability distribution that captures the stochasticity of the model. The transition function f represents the agent-based model's stepping mechanism, which for *StationSim* moves each agent towards its desired exit whilst avoiding collisions with other agents. The b-dimensional observation vector $y_k \in \mathbb{R}^b$, which can be fully observed, is determined from the true state-vector via

$$y_{k+1} = h(x_{k+1}, r_k), \tag{11}$$

where r_k captures the sensor noise. Recall that in this application only the positions of the agents are observed. The measurement function h simply extracts agent positions from the single run of StationSim that was used to generate the pseudo-truth data.

Under the assumption that the true and observed model states are multivariate Gaussian distributed with means μ_{xk} and μ_{yk}, and covariances Σ_{xk} and Σ_{yk} respectively, i.e. $x_k \sim N_a(\mu_{xk}, \Sigma_{xk})$ and $y_k \sim N_b(\mu_{yk}, \Sigma_{yk})$, data assimilation proceeds via two steps [5,46]:

(i) In the *forecast* step, an ensemble of *sigma points* $\mathcal{X}_k$ are computed deterministically from the mean and covariance of the posterior distribution for the current true model state x_k. The sigma points are then evolved independently forward in time via Eq. (10) to give a sample of forecasted states $\mathcal{X}_{k+1} = f(\mathcal{X}_k, q_k)$. The sigma points are used to compute a forecast of the mean and covariance of the true state at time t_{k+1} by applying the Unscented Transform function [23]. Forecasts for the observation vector y_{k+1} are also computed in a similar way using (11) on the forecasted sigma points and again computing the unscented mean and covariance.

(ii) The *update* step, following the observations at time t_{k+1}, standard Kalman filter practice [22] is used to calculate a final assimilated estimate of the model state x_{k+1} and its parameters $\mu_{x(k+1)}$, $\Sigma_{x(k+1)}$. This is taken as the new posterior distribution of the current model state and fed back into the forecast step to calculate new sigma points.

These steps are then iterated until the final observation.

The UKF requires user choices for both the type of sigma points used and their weightings. For simplicity, we use the standard choice of Merwe's Scaled Sigma Points and weights [5]. This set of sigma points is constructed using the current model state mean and covariance and three tuning parameters α, β, and κ. The concept is similar to that of an n-dimensional confidence interval, using a central mean sigma point as well as $4n$ outer sigma points centred about the mean some distance away depending on the covariance structure. Given our high-dimensional state-space, we adopt the recommended [5] tuning parameter values $(\alpha, \beta, \kappa) = (0.3, 2, 0)$. Values for the process and sensor noise are chosen as additive Gaussian noise with 0 mean and a/b-dimensional covariances I_a and I_b respectively.

Appendix B. Variable dimension RJUKF

This work above assumes the StationSim model instances have fixed dimensions. No agents leave or enter the model so this assumption is reasonable. However future application to variable dimensional states is desired. To do this it is then required to calculate the proposal densities and Jacobian matrix for Eq. (1).

To calculate the desired quantities the agents in both filters are split into 3 parts. First, common agents X that occur in both filters. These are agents in both sets and can be still be compared directly. For these agents the process is the same as in the invariable dimension case. Second, are agents that only exist in one filter. These could be agents that have entered or left one filter but not the other. Agents in the first and second filter only are denoted by sets A and B respectively. These agents cannot be compared directly. To compare these agents two sets of auxiliary variables U and V must be added. In essence, these auxiliary variables act as fake agents to be added to both filters to allow for direct comparison. Variables U correspond to agents B missing from the first filter. Likewise, V correspond to the A agents missing from the second filter. This results in two augmented states each with $|X + A + B|$ agents to be compared as follows.

$$\begin{bmatrix} X \\ A \\ U \end{bmatrix} = \begin{bmatrix} X \\ V \\ B \end{bmatrix}$$

The question is then how to calculate the proposal densities for these U and V auxiliary agents. It is simple to use the actual densities of the agents A and B they correspond to. For example, the set of agents U represent a set fake agents that correspond 1 to 1 with B. It is natural then to use the posterior distribution of these B agents calculated in the candidate UKF as the proposal density of U. This will be a multivariate Gaussian distribution whose parameters $(\mu_b,\ \Sigma_b)$ are easy to calculate. Similarly, the proposal density of set V is given as the Gaussian distribution of A. Hence the final proposal densities are just a ratio of two multivariate Gaussian distributions for the positions of the A and B agents respectively. U and V are then multivariate Gaussian with means μ_B,

μ_A and variances Σ_A, Σ_B respectively. Per Eq. (1) the density for the candidate auxiliary variates V are in the numerator and U in the denominator.

$$\begin{aligned}\frac{r(u'_k)}{r(u_k)} &= \frac{r(V)}{r(u)}\\ &= \frac{|\Sigma_B|^{\frac{1}{2}}}{|\Sigma'_A|^{\frac{1}{2}}} \times \frac{\exp(-\frac{1}{2}(z_k-\mu'_A)^T\Sigma'^{-1}_A(z_k-\mu'_A))}{\exp(-\frac{1}{2}(z_k-\mu_B)^T\Sigma_B^{-1}(z_k-\mu_B))}.\end{aligned}$$

To calculate the Jacobian is also simple, since both augmented states are the same size with a 1 to 1 correspondence the Jacobian is simply the identity matrix as seen in the RSH case. This also has an easy to calculate determinant of 1.

For example, given the $2n$ dimensional state vector of positions for the current $\mu = (x_1, y_1, \dots, x_n, y_n)$ and candidate $\mu' = (x_1\}, y_1\}, \dots, x_n\}, y_n\})$ filters respectively. The correspondence is as follows:

$$\begin{aligned}x_1 &= x'_1\\ y_1 &= y'_1\\ &\vdots\\ x_n &= x'_n\\ y_n &= y'_n\end{aligned}$$

The Jacobian and its determinant are calculated as follows given the Kronecker delta $\delta_{i,j}$ for matrix row i and column j.

$$\delta_{i,j} = \begin{cases}1, & \text{if } i = j,\\ 0, & \text{if } i \neq j.\end{cases} \tag{12}$$

$$\begin{aligned}|J| &= \left|\frac{\partial\mu}{\partial\mu'}\right|\\ &= \left|\begin{pmatrix}\frac{\partial x_1}{\partial x'_1} & \dots & \frac{\partial x_1}{\partial y'_n}\\ \vdots & \ddots & \vdots\\ \frac{\partial y_n}{\partial x'_1} & \dots & \frac{\partial y_n}{\partial y'_n}\end{pmatrix}\right|\\ &= \left|\begin{pmatrix}1 & \dots & 0\\ \vdots & \ddots & \vdots\\ 0 & \dots & 1\end{pmatrix}\right|\\ &= |\delta_{ij}|\\ &= |I_{2n}|\\ &= 1\end{aligned}$$

Hence the proposal densities and Jacobian are calculated as desired.

Appendix C. Stationsim implementation

StationSim is an agent-based model where each agent has the objective of crossing a rectangular environment. To fulfil this objective, agents must follow three simple rules:

1. Try to follow an ideal path (linear) towards their chosen exit point.
2. If the ideal path is blocked (e.g. by an agent or other obstacle), try to move sideways to avoid the obstruction.
3. If both the ideal path and the lateral positions are blocked, stay still until one of them becomes vacant.

Mathematically these rules were implemented as follows. Consider that agent i is in the position $\vec{r}_{i,t}$ at time t. The ideal path towards the exit point during a time τ is given by the linear equation:

$$\vec{r}_{i,(t+\tau)} = \vec{r}_{i,t} + v_i\hat{d}_{i,t}\tau, \tag{13}$$

where for the ith agent v_i is the desired speed and $\hat{d}_{i,t}$ is a unit vector that indicates the destination (exit gate position). To verify if it is possible perform this movement, the collision time between agent i with all other agents and all edges of the station is computed. The time that agents i and j will take to collide is determined by

$$\Delta t_{ij} = \frac{-\Delta\vec{v}\cdot\Delta\vec{r} - \sqrt{d}}{\Delta\vec{v}\cdot\Delta\vec{v}}, \tag{14}$$

where $\Delta\vec{r}$ and $\Delta\vec{v}$ are the differences between the vector positions and velocities of two agents i and j in time t, and $\sigma = 1$ m is the sum of agents i and j's radii. If $\Delta t_{ij} > \tau$ for all $j \neq i$, then no collision between agents was detected in the time interval $t \to t + \tau$. The collision time between an agent i and a vertical wall is

$$\Delta t_i = \frac{w_x \pm \sigma_i - x_i}{v_{xi}}, \tag{15}$$

where w_x is the x-position of the vertical wall, $\sigma_i = 0.5$ m is the agent i radius, and x_i and v_{xi} are the horizontal position and velocity of the agent i. If $v_{xi} < 0$, then the wall is on the left of the agent and the positive sign in (15) is used. Similarly, if $v_{xi} > 0$ then the wall is on the right of the agent and the negative sign in (15) is used. If $v_{xi} = 0$ then the agent is either stationary or moving vertically and so does not collide with the wall. The time until collision between the agent i and a horizontal wall can be obtained through an analogous equation. Again, if $\Delta t_i > \tau$ for all walls, then no collision between the agent i and a wall was detected in the time interval $t \to t + \tau$.

Therefore, if Δt_{ij} and Δt_i are greater than τ there is no block in the ideal path in the time interval t and $t + \tau$, and the agent i will follow rule number 1. Otherwise the agent will try to move laterally (rule number 2). In this scenario, the agent i will try to move from position $\vec{r}_{i,t}$ to position

$$\vec{r}_{i,(t+\tau)} = \vec{r}_{i,t} + \hat{d}^{\perp}_{i,t}\mathcal{N}(\sigma_i, \sigma_i^2), \tag{16}$$

where $\hat{d}^{\perp}_{i,t}$ is a unit vector obtained from a 90° rotation (randomly clockwise or counter-clockwise) of the vector $\hat{d}_{i,t}$. The random size of the step is controlled by the $\mathcal{N}(\sigma_i, \sigma_i^2)$ term and ensures that the lateral step is consistent with the size of the agent i. Finally, if the position $\vec{r}_{i,(t+\tau)}$ is available, then the agent i can fulfil the rule number 2, otherwise, the agent will follow rule number 3. Putting it all together, the model state at time t is defined as $S_t = \{s_{1,t}, s_{2,t}, s_{3,t}, \dots, s_{n,t}\}$, where n is the number of agents and $s_{i,t} = \{\vec{r}_{i,t}, \hat{d}_{i,t}, v_i\}$ is the agent i state at time t.

Despite the agents following 3 simple rules, the possible outcomes are diverse and include the possibility of crowd formation. This is due to the outcome dependence with interactions between agents and consequent random decisions. Since interactions are so important for this model, we chose to follow a classic approach to identify collisions such that Eqs. Eqs. (14) and (15) were defined using *hard disc model* standard formulae described in [56].

For the case of using the StationSim model with the Grand Central Terminal data, the addition of a circular obstacle in the environment was necessary. The obstacle was implemented as a static hard disc with a 4 m radius and the collision time between the agent i and the obstacle was determined through the Eq. (14).

The StationSim was implemented in Python and the full code is open source and publicly available. The source code to run the StationSim model as well as the scripts and instructions to run the RJUKF experiments can be found in *[link unblinded after review]* The UKF and RJUKF algorithms are executed using a high memory node from Leeds' ARC4 HPC system[4]. Run times vary depending on the number of agents. There are currently no detailed run time statistics due to further efficiency improvements that are being made. The major run-time bottleneck is currently the repeated running of the python based StationSim model.

References

[1] D. Helbing, Agent-based modeling, in: Social Self-Organization, Springer, 2012, pp. 25–70.

[2] M.-L. Xu, H. Jiang, X.-G. Jin, Z. Deng, Crowd simulation and its applications: recent advances, J. Comput. Sci. Tech. 29 (5) (2014) 799–811, http://dx.doi.org/10.1007/s11390-014-1469-y, ISSN: 1000-9000, 1860-4749.

[3] J.A. Ward, A.J. Evans, N.S. Malleson, Dynamic calibration of agent-based models using data assimilation, R. Soc. Open Sci. 3 (4) (2016) http://dx.doi.org/10.1098/rsos.150703.

[4] E. Kalnay, Atmospheric Modeling, Data Assimilation and Predictability, Cambridge University Press, 2003.

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

[6] R.J. Rockett, A. Arnott, C. Lam, R. Sadsad, V. Timms, K.-A. Gray, J.-S. Eden, S. Chang, M. Gall, J. Draper, et al., Revealing COVID-19 transmission in Australia by SARS-CoV-2 genome sequencing and agent-based modeling, Nature Med. (2020) 1–7.

[7] S. Swarup, A. Marathe, M.V. Marathe, C.L. Barrett, 26. Simulation analytics for social and behavioral modeling, Social-Behav. Model. Complex Syst. (2019).

[8] C.N. van der Wal, D. Formolo, M.A. Robinson, M. Minkov, T. Bosse, Simulating crowd evacuation with socio-cultural, cognitive, and emotional elements, in: J. Mercik (Ed.), Transactions on Computational Collective Intelligence XXVII, vol. 10480, Springer International Publishing, Cham, 2017, pp. 139–177, http://dx.doi.org/10.1007/978-3-319-70647-4_11, ISBN: 978-3-319-70646-7 978-3-319-70647-4.

[9] B. Zhou, X. Wang, X. Tang, Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Providence, RI, 2012, pp. 2871–2878, http://dx.doi.org/10.1109/CVPR.2012.6248013, ISBN: 978-1-4673-1228-8 978-1-4673-1226-4 978-1-4673-1227-1.

[10] N. Malleson, K. Minors, L.-M. Kieu, J.A. Ward, A. West, A. Heppenstall, Simulating crowds in real time with agent-based modelling and a particle filter, J. Artif. Soc. Soc. Simul. (ISSN: 1460-7425) 23 (3) (2020) 3, http://dx.doi.org/10.18564/jasss.4266.

[11] P.J. Green, Reversible jump Markov chain Monte Carlo computation and Bayesian model determination, Biometrika 82 (4) (1995) 711–732.

[12] O. Talagrand, The use of adjoint equations in numerical modelling of the atmospheric circulation, in: A. Griewank, G.F. Corliss (Eds.), Automatic Differentiation of Algorithms: Theory, Implementation, and Application, SIAM, Philadelphia, PA, ISBN: 0-89871-284-X, 1991, pp. 169–180.

[13] N.B. Othman, E.F. Legara, V. Selvam, C. Monterola, A data-driven agent-based model of congestion and scaling dynamics of rapid transit systems, J. Comput. Sci. (ISSN: 1877-7503) 10 (2015) 338–350, http://dx.doi.org/10.1016/j.jocs.2015.03.006.

[14] M. Wang, X. Hu, Data assimilation in agent based simulation of smart environments using particle filters, Simul. Model. Pract. Theory (ISSN: 1569-190X) 56 (2015) 36–54, http://dx.doi.org/10.1016/j.simpat.2015.05.001.

[15] L.-M. Kieu, N. Malleson, A. Heppenstall, Dealing with uncertainty in agent-based models for short-term predictions, R. Soc. Open Sci. (ISSN: 2054-5703, 2054-5703) 7 (1) (2020) 191074, http://dx.doi.org/10.1098/rsos.191074.

[16] D.J.B. Lloyd, N. Santitissadeekorn, M.B. Short, Exploring data assimilation and forecasting issues for an urban crime model, European J. Appl. Math. (ISSN: 1469-4425) 27 (Special Issue 03) (2016) 451–478, http://dx.doi.org/10.1017/S0956792515000625.

[17] F. Oloo, K. Safi, J. Aryal, Predicting migratory corridors of white storks, ciconia ciconia, to enhance sustainable wind energy planning: a data-driven agent-based model, Sustainability (ISSN: 2071-1050) 10 (5) (2018) 1470, http://dx.doi.org/10.3390/su10051470.

[18] P. Del Moral, Nonlinear filtering: Interacting particle resolution, C. R. L'Acad. Sci. I. Math. 325 (6) (1997) 653–658.

[4] https://arcdocs.leeds.ac.uk/systems/arc4.html

[19] G. Evensen, The ensemble Kalman filter: Theoretical formulation and practical implementation, Ocean Dyn. 53 (4) (2003) 343–367.

[20] J. Lueck, J.H. Rife, S. Swarup, N. Uddin, Who goes there? using an agent-based simulation for tracking population movement, in: Winter Simulation Conference, Dec 8 - 11, 2019., National Harbor, MD, USA, 2019.

[21] P. Ternes, J.A. Ward, A.J. Heppenstall, V. Kumar, L.-M. Kieu, N. Malleson, Using data assimilation to reduce uncertainty in an agent-based pedestrian simulations in real time, Philos. Trans. A (2020) (in review).

[22] R.E. Kalman, A new approach to linear filtering and prediction problems, J. Basic Eng. (1960).

[23] J.K. Uhlmann, Dynamic Map Building and Localization: New Theoretical Foundations (Ph.D. thesis), University of Oxford Oxford, 1995.

[24] R. Clay, L.-M. Kieu, J.A. Ward, A. Heppenstall, N. Malleson, 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, Springer, 2020, pp. 68–79.

[25] L. Yin, Z. Deng, B. Huo, Y. Xia, C. Li, Robust derivative unscented Kalman filter under non-Gaussian noise, IEEE Access 6 (2018) 33129–33136.

[26] M. Raitoharju, S. Ali-Löytty, R. Piché, Binomial Gaussian mixture filter, EURASIP J. Adv. Signal Process. 2015 (1) (2015) 36.

[27] H. Tanizaki, Kalman Filter model with qualitative dependent variables, Rev. Econ. Stat. (1993) 747–752.

[28] P.X.-K. Song, Monte Carlo Kalman Filter and smoothing for multivariate discrete state space models, Canad. J. Statist. 28 (3) (2000) 641–652.

[29] C. Andrieu, M. Davy, A. Doucet, Efficient particle filtering for jump Markov systems. Application to time-varying autoregressions, IEEE Trans. Signal Process. 51 (7) (2003) 1762–1770.

[30] F. Deng, J. Chen, C. Chen, Adaptive unscented Kalman filter for parameter and state estimation of nonlinear high-speed objects, J. Syst. Eng. Electron. 24 (4) (2013) 655–665.

[31] A. Van Der Linde, DIC In variable selection, Stat. Neerl. 59 (1) (2005) 45–56.

[32] C.M. Pooley, G. Marion, Bayesian Model evidence as a practical alternative to deviance information criterion, R. Soc. Open Sci. 5 (3) (2018) 171519.

[33] K. Xiong, T. Liang, L. Yongjun, Multiple model Kalman filter for attitude determination of precision pointing spacecraft, Acta Astronaut. 68 (7–8) (2011) 843–852.

[34] J. Schulz, C. Hubmann, J. Löchner, D. Burschka, Multiple model unscented Kalman filtering in dynamic Bayesian networks for intention estimation and trajectory prediction, in: 2018 21st International Conference on Intelligent Transportation Systems, ITSC, 2018, pp. 1467–1474, http://dx.doi.org/10.1109/ITSC.2018.8569932, ISSN: 2153-0017.

[35] M. Edali, G. Yücel, Exploring the behavior space of agent-based simulation models using random forest metamodels and sequential sampling, Simul. Model. Pract. Theory 92 (2019) 62–81.

[36] R.J. Barker, W.A. Link, Bayesian Multimodel inference by RJMCMC: A gibbs sampling approach, Amer. Statist. 67 (3) (2013) 150–156.

[37] P. Craciun, M. Ortner, J. Zerubia, Integrating RJMCMC and Kalman filters for multiple object tracking, in: GRETSI–Traitement Du Signal Et Des Images, 2015.

[38] Z. Huibo, P. Quan, PF-UKF-RJMCMC Approaches for radar target-tracking, in: 2009 International Conference on Information Technology and Computer Science, vol. 2, IEEE, 2009, pp. 373–376.

[39] D.J. Spiegelhalter, N.G. Best, B.P. Carlin, A. Van der Linde, The deviance information criterion: 12 years on, J. R. Stat. Soc. Ser. B Stat. Methodol. (2014) 485–493.

[40] A. Theorell, K. Nöh, Reversible jump MCMC for multi-model inference in Metabolic Flux Analysis, Bioinformatics 36 (1) (2020) 232–240.

[41] R. Astroza, L.T. Nguyen, T. Nestorović, Finite element model updating using simulated annealing hybridized with unscented Kalman filter, Comput. Struct. 177 (2016) 176–191.

[42] M. Betancourt, M. Girolami, Hamiltonian Monte Carlo for hierarchical models, Current Trends Bayesian Methodol. Appl. 79 (30) (2015) 2–4.

[43] R.A. Levine, A note on Markov chain Monte Carlo sweep strategies, J. Stat. Comput. Simul. 75 (4) (2005) 253–262.

[44] S.J. Julier, J.K. Uhlmann, Unscented filtering and nonlinear estimation, Proc. IEEE 92 (3) (2004) 401–422.

[45] R. Van Der Merwe, et al., Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models (Ph.D. thesis), OGI School of Science & Engineering at OHSU, 2004.

[46] R. Van Der Merwe, E.A. Wan, The square-root unscented Kalman filter for state and parameter-estimation, in: 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221), 6, IEEE, 2001, pp. 3461–3464.

[47] B. Liu, H. Liu, H. Zhang, X. Qin, A social force evacuation model driven by video data, Simul. Model. Pract. Theory (ISSN: 1569190X) 84 (2018) 190–203, http://dx.doi.org/10.1016/j.simpat.2018.02.007.

[48] T. Berry, T. Sauer, Adaptive ensemble Kalman filtering of non-linear systems, Tellus A 65 (1) (2013) 20331.

[49] G. Chen, X. Meng, Y. Wang, Y. Zhang, P. Tian, H. Yang, Integrated WiFi/PDR/Smartphone using an unscented kalman filter algorithm for 3D indoor localization, Sensors 15 (9) (2015) 24595–24614.

[50] W.M. Farr, I. Mandel, D. Stevens, An efficient interpolation technique for jump proposals in reversible-jump Markov chain Monte Carlo calculations, R. Soc. Open Sci. 2 (6) (2015) 150030.

[51] R. Van Der Merwe, A. Doucet, N. De Freitas, E.A. Wan, The Unscented Particle Filter, 2000.

[52] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, S. Savarese, Social lstm: Human trajectory prediction in crowded spaces, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 961–971.

[53] Z. Yan, Traj-ARIMA: a spatial-time series model for network-constrained trajectory, in: Proceedings of the Third International Workshop on Computational Transportation Science, 2010, pp. 11–16.

[54] B. Herd, S. Miles, P. McBurney, M. Luck, Compositional transient reachability analysis for agent-based simulations, Stud. Inform. Univ. 10 (3) (2012) 87–118.

[55] S. Swarup, H.S. Mortveit, Live simulations, in: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, 2020, pp. 1721–1725.

[56] J.M. Haile, Molecular Dynamics Simulation: Elementary Methods, in: Monographs in Physical Chemistry Series, Wiley, ISBN: 9780471819660, 1992.