Data Assimilation in Discrete Event Simulations: A Rollback Based Sequential Monte Carlo Approach

Xie, Xu; Verbraeck, Alexander; Gu, Feng

DOI
10.23919/TMS.2016.7918817

Publication date
2016

Document Version
Final published version

Published in
Proceedings of the Symposium on Theory of Modeling Simulation (TMS-DEVS)

Data Assimilation in Discrete Event Simulations – A Rollback based Sequential Monte Carlo Approach

Xu Xie
Delft University of Technology
Delft, The Netherlands
x.xie@tudelft.nl

Alexander Verbraeck
Delft University of Technology
Delft, The Netherlands
a.verbraeck@tudelft.nl

Feng Gu
City University of New York
New York, United States
Feng.Gu@csi.cuny.edu

ABSTRACT
Data assimilation is an analysis technique which aims to incorporate measured observations into a dynamic system model in order to produce accurate estimates of the current state variables of the system. Although data assimilation is conventionally applied in continuous system models, it is also a desired ability for its discrete event counterpart. However, data assimilation has not been well studied in discrete event simulations yet. This paper researches data assimilation problems in discrete event simulations, and proposes a rollback based implementation of the Sequential Monte Carlo (SMC) method – the rollback based SMC method. To evaluate the accuracy of the proposed method, an identical-twin experiment in a discrete event traffic case is carried out and the results are presented and analyzed.

Author Keywords
Data Assimilation, Discrete event simulations, Sequential Monte Carlo methods, Rollback.

ACM Classification Keywords
I.6.8 SIMULATION AND MODELING: Discrete event.

1. INTRODUCTION

Computer simulations have long been used for studying and predicting the behavior of complex systems [11]. However, accurate analysis and prediction of the behavior of complex systems are difficult, because even complex models are still lacking the ability to accurately describe such systems [6], therefore, even elaborate complex models of systems produce simulations that diverge from or fail to predict the real behavior of those systems. This situation is accentuated in cases where real-time dynamic conditions exist [6].

The availability of real-time observations from real systems has increased along with the advances in sensor technology. The increased availability of data allows for a new simulation paradigm – *dynamic data driven simulation*, where a simulation is continually influenced by the real time data streams for better analysis and prediction of a system under study [11]. The core technique in dynamic data driven simulation is *data assimilation*, in which observations are incorporated into a dynamic system model to produce accurate estimates of current states of the system [16].

Data assimilation has been applied with success in many applications, such as weather forecasting [12], chemical data assimilation [5], ocean data assimilation [3], etc. But in these applications, systems are continuous and are conventionally modeled as (*partial*) *differential equations*, and these differential equations are again computed using *numerical methods*, and thus approximated by *difference equations* [18]. Besides continuous systems and models, a large number of discrete event systems and models exist in practice, such as manufacturing systems, queuing networks, etc. In discrete event systems, entities are usually represented with discrete state variables which evolve at discrete moments over continuous time, and change their values due to the occurrence of particular events, and the system's evolution depends on the interactions of such events and their arrival times [10, 18]. Data assimilation is also a desired ability in discrete event simulations, especially in real-time applications of simulation models, where the model provides predictions based on the last known state of the system it represents. We take a traffic signal control example to explain the necessities of assimilating data in discrete event simulations. In [4, 14, 15], the authors have presented how traffic signal control systems can be modeled and simulated using discrete event methods; and in reality, we often see the phenomenon that vehicles accumulate at crossings in one direction, while in the orthogonal direction, roads are almost empty with green lights on. Observations of such a situation are very easy to collect by sensors (e.g., inductive loops). If we could assimilate these observations into the discrete event traffic signal control model and dynamically adjust durations of phases of traffic lights, a better performance (e.g., traffic flow) would be achieved.

However, data assimilation in discrete event simulations is not well researched yet, and due to the highly nonlinear, non-Gaussian properties, most data assimilation algorithms cannot be applied in discrete event simulations. Sequential Monte Carlo (SMC) methods seem to be a set of promising methods which might be applicable in discrete event simulations, since they are able to approximate arbitrary probability densities and have little or no assumption about the properties of the system model [2].

Applying SMC methods in discrete event simulations requires a variety of capabilities of the simulation environment, such as pausing or stopping the simulation, retrieving state from or setting a new state to the simulation, etc. Therefore, in this paper, we propose an implementation of SMC methods in object oriented discrete event simulations – the *rollback based SMC method*, which is based on the concept

SpringSim-TMS/DEVS 2016 April 3-6 Pasadena, CA, USA

of *rollback*. In this paper, rollback has its natural meaning which means restoring the simulation to a previous state. In the proposed method, particles are generated by recursively rolling back the simulation. When there is no data, only one simulation is kept running; while when data is available, the simulation is first rolled back to a particle in the last data assimilation, and then run to the current observation time. At this moment, a particle is generated by retrieving the model state. This procedure is repeated until the specified number of particles are generated. After resampling, the particle with the highest probability is assigned to the model, while other particles are kept for the next assimilation. The simulation assigned with the most probable particle is run again until the next observation arrives.

An identical-twin experiment in a traffic case which is implemented in a discrete event traffic simulation software is carried out to evaluate the accuracy of the proposed method. The results show that the simulation with the rollback based SMC method can accurately estimate the location of the slower vehicle on the road.

The remainder of the paper is organized as follows. Section 2 overviews the SMC methods, and clarifies the research gaps. Section 3 presents the rollback based SMC method, and provides the implementation details. Experiments and results are given in section 4. Conclusions are drawn in section 5.

2. RELATED WORK

2.1 Sequential Monte Carlo (SMC) Methods

To define the data assimilation problem, two models are required: one is the *system model* which describes the evolution of the state with time, and the other is the *measurement model* which relates the noisy observations to the state. The two models are conventionally expressed as difference equations [1, 7, 13]:

$$s_t = f_t(s_{t-1}) + \nu_{t-1}$$
$$m_t = g_t(s_t) + \varepsilon_t$$

where f_t is a possibly nonlinear function of the state s_{t-1}, ν_{t-1} is the process noise. g_t is possibly a nonlinear function which maps the state to the measurement, ε_t is the measurement noise.

The nonlinearity in the model and non-Gaussian property of noise make conventional data assimilation methods inapplicable, such as Kalman filters [1], Extended Kalman filters [9]. However, Sequential Monte Carlo (SMC) methods gain popularity since they are able to approximate arbitrary probability densities and have little or no assumption about the properties of the system model [2]. SMC methods, also called *particle filters*, is a technique for implementing a recursive Bayesian filter by Monte Carlo (MC) simulations [1]. The key idea is to represent the required posterior distribution by a set of random samples (also called *particles*) with associated weights and to compute estimates based on these samples and weights. As the number of samples becomes very large, this MC characterization becomes an equivalent representation to the usual functional description of the posterior distribution [1, 7].

We are interested in obtaining the posterior distribution $p(s_{0:t}|m_{1:t})$, where $s_{0:t} = \{s_i, i = 0, \ldots, t\}$ is the set of all states up to time t, and $m_{1:t} = \{m_i, i = 1, \ldots, t\}$ is the set of all available observations up to the same time. The SMC methods approximate $p(s_{0:t}|m_{1:t})$ by a *random measure*

$$\chi_t = \{s^i_{0:t}, w^i_t\}^N_{i=1} \tag{1}$$

where $\{s^i_{0:t}, i = 1, \ldots, N\}$ is a set of *support points* with associated *weights* $\{w^i_t, i = 1, \ldots, N\}$. The weights are normalized such that $\sum^N_{i=1} w^i_t = 1$. Then we have

$$p(s_{0:t}|m_{1:t}) \approx \sum_{i=1}^{N} w^i_t \delta(s_{0:t} - s^i_{0:t}) \tag{2}$$

where $\delta(\cdot)$ is the Dirac delta function. Usually, direct sampling from $p(s_{0:t}|m_{1:t})$ is intractable, therefore the sequential importance sampling (SIS) algorithm is developed, in which samples are drawn from an easily sampled distribution which is called *importance density*, and weights are updated using

$$w^i_t \propto \frac{p(m_t|s^i_t)p(s^i_t|s^i_{t-1})}{\pi(s^i_t|s^i_{0:t-1}, m_{1:t})} w^i_{t-1} \tag{3}$$

where $\pi(s^i_t|s^i_{0:t-1}, m_{1:t})$ is the importance density. Readers can refer to [1] and [7] for more details on the derivation of the SIS algorithm. The SIS algorithm can be implemented by performing the following two steps for every t:

1. draw particles $s^i_t \sim \pi(s_t|s^i_{0:t-1}, m_{1:t}), i = 1, \ldots, N$, and append them to $s^i_{0:t-1}$ to form $s^i_{0:t}$;

2. compute the weights w^i_t according to equation (3), and normalize the weights.

The importance density plays a very important role in the performance of the SMC methods. In general, the closer the importance density to that distribution, the better the approximation [7]. If $\pi(s_t|s^i_{0:t-1}, m_{1:t}) = \pi(s_t|s^i_{t-1}, m_t)$, then the importance density becomes only dependent on s_{t-1} and m_t. This is particularly useful in the common case when only a filtered estimate of $p(s_t|m_{1:t})$ is required at each time step. In such scenarios, only s^i_t need be stored, therefore, one can discard the path $s^i_{0:t-1}$ and history of observations $m_{1:t-1}$. The *prior importance density* which is given by $p(s_t|s^i_{t-1})$ [7] is a such density, and it implies particle weight updates by

$$w^i_t \propto p(m_t|s^i_t) w^i_{t-1} \tag{4}$$

The simplified SIS algorithm is shown in Algorithm 1.

Algorithm 1: The SIS algorithm

Input: random measure at $t-1$: $\chi_{t-1} = \{s^i_{t-1}, w^i_{t-1}\}^N_{i=1}$, new observation m_t
Output: random measure at t: $\chi_t = \{s^i_t, w^i_t\}^N_{i=1}$
for $i = 1 : N$ **do**
  draw particles $s^i_t \sim p(s_t|s^i_{t-1}, m_t)$;
  assign the particle a weight, w^i_t, according to (4);
end

A major problem with the SIS algorithm is that the discrete random measure degenerates quickly [1, 7]. In other words, all the particles except for a very few are assigned negligible weights. Degeneracy can be reduced by *resampling*, in which particles are replicated in proportion to their weights [7]. A complete implementation of the SMC methods based on the SIS algorithm and resampling is presented in Algorithm 2.

Algorithm 2: The SMC methods

Input: random measure at $t-1$: $\chi_{t-1} = \{s_{t-1}^i, w_{t-1}^i\}_{i=1}^N$, new observation m_t
Output: random measure at t: $\chi_t = \{s_t^i, w_t^i\}_{i=1}^N$
% the sampling step
for $i = 1 : N$ **do**
  draw particles $s_t^{'i} \sim p(s_t|s_{t-1}^i, m_t)$;
  assign $s_t^{'i}$ a weight: $w_t^{'i} = p(m_t|s_t^{'i})w_{t-1}^i$;
end
normalize the weights: $w_t^{''i} = \frac{w_t^{'i}}{\sum_{j=1}^N w_t^{'j}}$
% the resampling step
$c_0 = 0$;
for $i = 1 : N$ **do**
  $c_i = c_{i-1} + w_t^{''i}$
end
for $i = 1 : N$ **do**
  generate a random number $r \sim U[0, 1]$;
  if $c_{j-1} < r \leq c_j$ **then**
    $s_t^i = s_t^{'j}$;
    $w_t^i = \frac{1}{N}$;
  end
end

2.2 Data Assimilation in Discrete Event Simulations

In discrete event simulations, the behavior of discrete event simulations is highly non-linear, non-Gaussian. The functions to describe state evolution in discrete event simulations are usually rule-based. These functions are essentially step functions and can therefore not be linearized, because state changes happen instantaneously at the event. The high non-linearity of state transition functions in discrete event simulations hampers the application of data assimilation algorithms which are based on linear assumption (such as Kalman filters [1]) or local linearization (such as Extended Kalman filters [9]).

SMC methods seem to be a set of promising methods which are applicable in discrete event simulations, since they are able to approximate arbitrary probability densities and have little or no assumption about the properties of the system model [2].

In order to apply SMC methods in discrete event simulations, several functionalities should be added to the discrete event simulation environment. During data assimilation, we need to repeatedly pause or stop the simulation, retrieve state from or set new state to the simulation, etc. Therefore, in this paper, we propose a novel, efficient SMC method – the *rollback based SMC method*, which is implemented in an object oriented discrete event simulations. In the proposed method, particles are generated by recursively rolling back the simulation. When there is no data, only one simulation is kept running; while when data is available, the simulation is first rolled back to a particle in the last data assimilation, and then run to the current observation time. At this moment, a particle is generated by retrieving the model state. This procedure is repeated until the specified number of particles are generated. After resampling, the particle with the highest probability is assigned to the model, while other particles are kept for the next assimilation. The simulation assigned with the most probable particle is run again until the next observation arrives. The proposed method is elaborated in section 3.

3. ROLLBACK BASED SMC METHOD

The main idea of the rollback based SMC method is shown in Figure 1. In our proposed method, N particles are recursively generated by the *rollback based sampling step*. The phrase *roll back* originally means reversing or undoing something. In the simulation community, rollback is widely used in optimistic synchronization algorithm in parallel discrete event simulations which means undoing the computations of previously processed events [8]. In parallel discrete event simulations, rollback is always associated with unsending messages because logical processes are "optimistically" executed in parallel [8]. In our method, discrete event simulations are not parallelized, therefore, rollback has its natural meaning which means restoring the simulation to a previous state.

In a set of particles $\{s_t^i, i = 1, \ldots, N\}$, we always assume that the first particle s_t^1 has the highest probability. Assume the initial state is distributed as $p(s_0)$. In the initialization, N particles $\{s_0^i, i = 1, \ldots, N\}$ are drawn from $p(s_0)$, but only s_0^1 is assigned to the model and the simulation starts to run, i.e., only one simulation with the most probable state is running. Assume at time t, measurement m_t is collected, and the particles at the last data assimilation time $t-1$ are $\{s_{t-1}^i, i = 1, \ldots, N\}$. The rollback based sampling step generates N particles $\{s_t^{'i}, i = 1, \ldots, N\}$ as follows:

1. the first particle $s_t^{'1}$ is already embedded in the simulation, therefore, it can be generated by retrieving the current state of the model;
2. the other $N-1$ particles $\{s_t^{'i}, i = 2, \ldots, N\}$ are generated recursively as follows:
   (a) roll back the current simulation to s_{t-1}^i;
   (b) run the simulation to time t, and $s_t^{'i}$ is generated by retrieving the model state.

Except sampling, the resampling step in the proposed method is the same with that in the standard SMC methods, therefore, it will not be repeated here. After resampling, the particle with the highest probability is assigned to the model, while other particles are kept for the next assimilation. The simulation assigned with the most probable particle is run again until the next observation arrives.

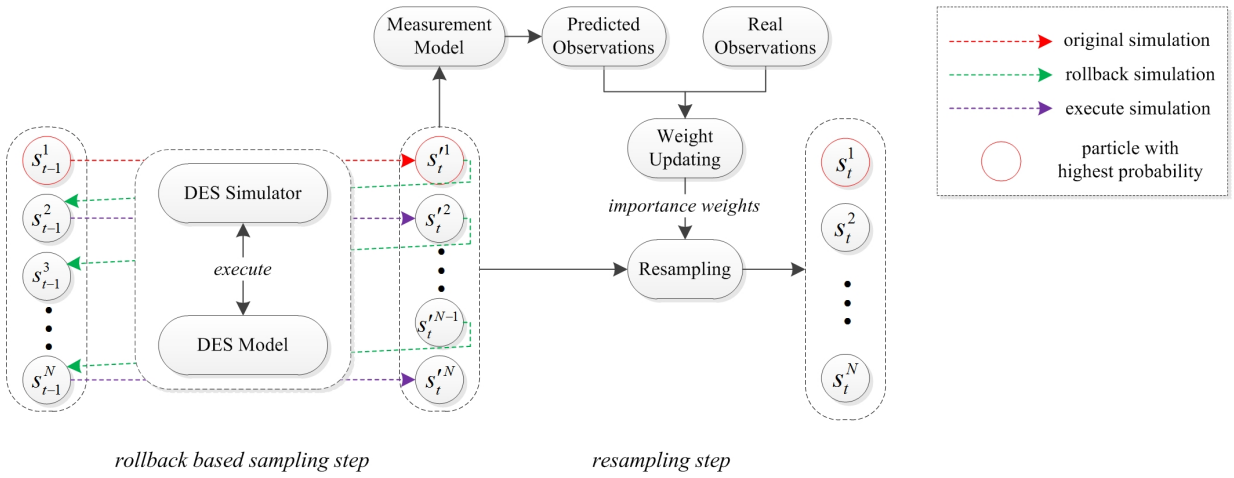


Figure 1: SMC methods based on simulation rollback.

In section 4, a discrete event traffic case is studied to prove the effectiveness of the proposed algorithm. The traffic model is built in OpenTrafficSim[1], which is a Java based, open source, discrete event simulation software to support research and development of multi scale and multi modal traffic models.

In order to do rollback in OpenTrafficSim, the vehicle class `LaneBasedIndividualCar` is extended to enable the discrete event model to save and restore its state; the discrete event simulator class `DESSimulator` is revised to be capable of setting its time backward. Besides, after the model state and simulator time are rolled back, the model reschedules events on its simulator to reconstruct the future event list. As shown in Figure 2, state saving & restoring and event rescheduling are enabled by adding following methods:

- `saveCurrentState`: saves current model state;
- `restoreState`: restores model state to a specified state;
- `getCurrentState`: retrieve current model state;
- `reSchedule`: reschedule events.

The only change in the discrete event simulator is one added function `setSimulatorTimeBackTo`, which enables the simulator time to be set back and maintains the future event list accordingly.

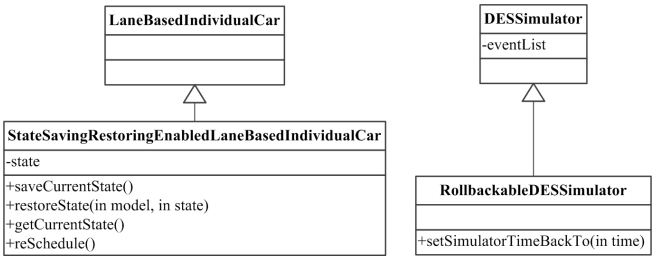


Figure 2: Class diagram.

[1]More information about OpenTrafficSim can be found in **`http://www.opentrafficsim.org/`**.

4. EXPERIMENTS AND RESULTS

4.1 Scenario Description

A two-lane circular road is shown in Figure 3a. The total length of the road is 2000 meters, on which $N_v = 80$ vehicles are driving in counterclockwise direction. Along the road, N_s sensors are evenly installed (in Figure 3a, $N_s = 8$), and each sensor has 100-meter detection range on each lane. Each sensor can report two types of information every ΔT seconds: 1) the number of vehicles in its detection range; 2) the average speed of all vehicles in the detection range (if there is no vehicle in the range, the speed is defined as the maximum allowed speed of the road). Among all vehicles on the road, there is one slower vehicle ($v_{max} = 25km/h$, which is $1/4$ of the normal vehicles' maximum speed). Our objective is to locate the slower vehicle by incorporating measurements from sensors into the discrete event microscopic traffic simulation which is implemented in OpenTrafficSim.

4.2 Problem Formulation

System Model

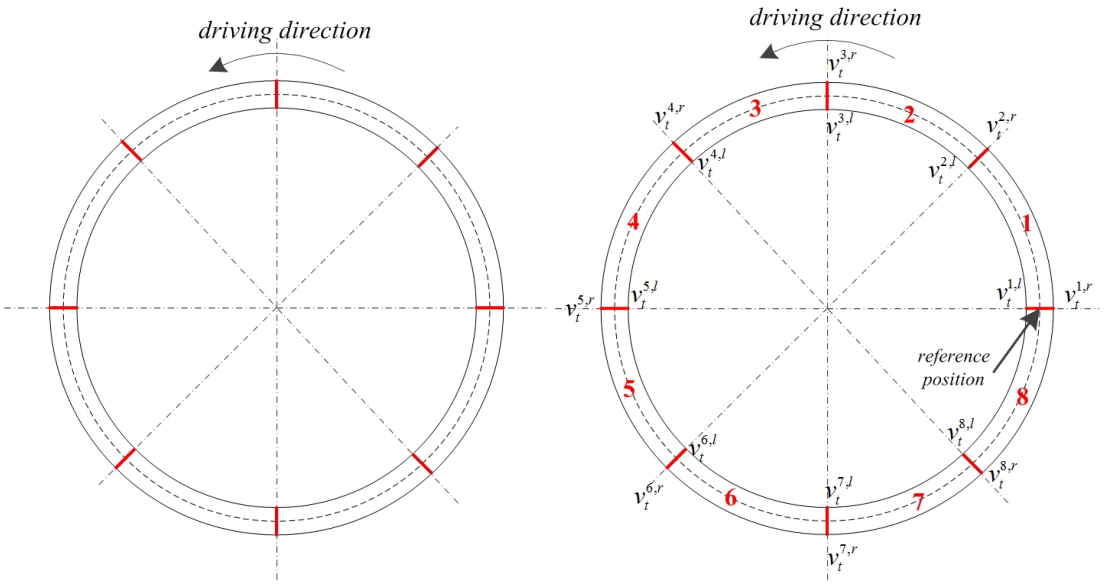
The system state at time t is defined by a collection of all vehicles' state

$$S_t = \{l_t^i, v_t^i, a_t^i, v_{t,max}^i\}_{i=1}^{N_v}$$

where l_t^i, v_t^i, a_t^i, $v_{t,max}^i$ are the i-th vehicle's *location*, *velocity*, *acceleration* and *maximum speed* (used for recognizing the slower vehicle) at time t, and N_v is the number of vehicles on the road. The system state evolves as simulation proceeds

$$S_t = CircularRoadTrafficSim(S_{t-1})$$

where $CircularRoadTrafficSim$ is the discrete event microscopic traffic simulation implemented in OpenTrafficSim, where driving behaviors, such as lane-changing, accelerating or decelerating, are modeled by MOBIL and IDM++ [17]. The time step ΔT is the period between two consecutive measurements, rather than the simulation step or time between two consecutive events. System noise ν is added by randomly selecting a normal vehicle around the slower vehicle (ΔL meters before and after) and exchanging their maximum speed



(a) The two-lane circular road, sensors are evenly installed at positions marked in red.

(b) Estimating the slow vehicle position by average speed increases between two consecutive sensors.

Figure 3: Two-lane circular road.

(in our experiment, $\Delta L = 50m$). The system model with noise is thus formulated as

$$S_t = CircularRoadTrafficSim(S_{t-1}) + \nu_{t-1}$$

Measurement Model

Measurements from sensors at time t are defined by

$$M_t = \{n_t^{j,l}, n_t^{j,r}, \bar{v}_t^{j,l}, \bar{v}_t^{j,r}\}_{j=1}^{N_s}$$

where $n_t^{j,l}$, $n_t^{j,r}$ are number of vehicles on the left and right lane respectively within the detection range of the j-th sensor at time t; $\bar{v}_t^{j,l}$, $\bar{v}_t^{j,r}$ are the average speeds of vehicles defined similarly, and N_s is the number of sensors. The measurement model MM

$$M_t = MM(S_t)$$

maps the system state S_t to measurement M_t by

$$n_t^{j,l} = \sum_{i=1}^{N_v} \delta_i^{j,l}, \quad n_t^{j,r} = \sum_{i=1}^{N_v} \delta_i^{j,r}$$

$$\bar{v}_t^{j,l} = \begin{cases} \frac{1}{n_t^{j,l}} \sum_{i=1}^{N_v} \delta_i^{j,l} v_t^i & n_t^{j,l} > 0 \\ V_m & n_t^{j,l} = 0 \end{cases}$$

$$\bar{v}_t^{j,r} = \begin{cases} \frac{1}{n_t^{j,r}} \sum_{i=1}^{N_v} \delta_i^{j,r} v_t^i & n_t^{j,r} > 0 \\ V_m & n_t^{j,r} = 0 \end{cases}$$

where V_m is the maximum allowed speed of the road. $\delta_i^{j,l} = 1$ if vehicle i is on the left lane of the j-th sensor's detection range, otherwise $\delta_i^{j,l} = 0$. Similar definition applies to $\delta_i^{j,r}$.

Error in measurements, ε, is assumed unbiased and modeled as a Gaussian noise $\mathcal{N}(0, \sigma^2)$ (in our experiment, we set $\sigma = 1.0$). The measurement model with noise is formulated as

$$M_t = MM(S_t) + \varepsilon_t$$

4.3 Weight Computation

Weight computation is a very important step in SMC methods since it provides the measure to keep the optimal particles for future steps. In this paper, the weight of the i-th particle is computed by taking three types of information into consideration:

- number of vehicles within each sensor's detection range. We define $d_i = max\{|n_t^{j,l} - n_t^{'j,l}|, |n_t^{j,r} - n_t^{'j,r}|\}_{j=1}^{N_s}$, where $n_t^{'j,l}$, $n_t^{'j,r}$ are numbers of vehicles on the left and right lane respectively within the detection range of j-th sensor in real traffic, while $n_t^{j,l}$ and $n_t^{j,r}$ are the corresponding numbers in the i-th particle. The contribution of this information to the weight is defined as

$$p_1^i = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d_i^2}{2}}$$

- average speed of vehicles in each sensor's detection range. Similar to the way in dealing with the number of vehicles, we assume v_i as the maximum average speed difference between the real traffic and the particle, and define its contribution as

$$p_2^i = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{v_i^2}{2}}$$

- the difference of slower vehicle locations between the real traffic and the particle. Because we have no information about the slower vehicle location in the real traffic, we need to estimate the area the slower vehicle is probably in. A

slower vehicle can block vehicles behind it, hence, there will be a sharp increase of average speed around the slow vehicle. We calculate the speed increase between two consecutive sensors by

$$I_j^l = \begin{cases} \bar{v}_t^{j+1,l} - \bar{v}_t^{j,l} & j = 1, \ldots, N_s - 1 \\ \bar{v}_t^{1,l} - \bar{v}_t^{N_s,l} & j = N_s \end{cases}$$

where $\bar{v}_t^{j,l}$ is the average speed of vehicles on the left lane within the detection range of the j-th sensor. Assume j^{*l} is the index of sensor which satisfies $I_{j^{*l}}^l = \max_{j=1,\ldots,N_s} I_j^l$, therefore, the slower vehicle should be in the area j^{*l} shown in Figure 3b ($N_s = 8$). But in order to be more robust, we extend the area by adding two adjacent areas: area j_b^{*l} and area j_f^{*l}, which is behind and in the front of area j^{*l}, respectively. Suppose in the particle, the slower vehicle is in area k, we define

$$p_l = \begin{cases} 0.9 & k = j^{*l} \\ 0.05 & k = j_b^{*l} \text{ or } j_f^{*l} \end{cases}$$

The same method applies to the data of the right lane, then we can get j^{*r}, j_b^{*r}, j_f^{*r} and p_r. The contribution to the weight is therefore defined by

$$p_3^i = (p_l + p_r)/2$$

The three contributions are linearly combined as $p^i = \alpha p_1^i + \beta p_2^i + \gamma p_3^i$, such that $\alpha + \beta + \gamma = 1$. In our experiment, we choose $\alpha = \gamma = 0.45, \beta = 0.1$. Then the weight of the i-th particle is accordingly updated by $w_t^{'i} = w_{t-1}^i p^i$, and is finally normalized by $w_t^i = w_t^{'i} / \sum_{j=1}^N w_t^{'j}$.

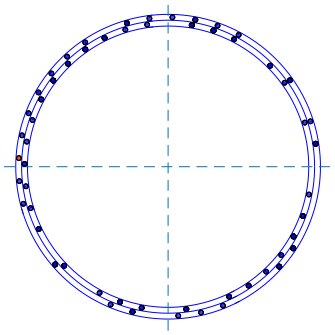
4.4 Experiment Setup

The identical-twin experiment is adopted in this paper to evaluate the effectiveness of the proposed method. In the identical-twin experiment, a simulation which contains one slower vehicle is first run and the corresponding data is recorded. These simulation results are considered as "real", therefore, the observation data obtained here are regarded as coming from the "real" system. Consequently, we estimate the location of the slower vehicle from the observation data using the proposed method and then check whether the estimation is closed to the "real" location.

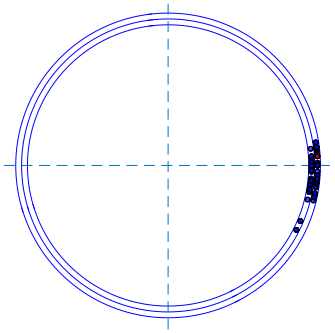
The run length of of the simulation is 480 seconds, and observations are fed every $\Delta T = 10$ seconds. The number of particles is chosen to be 100.

4.5 Experiment Results

The location of the slower vehicle in real traffic at $t = 0$ is shown in red in Figure 4a, while in each particle, slower vehicle is randomly chosen and located, which is shown in blue. As simulation proceeds, more and more observations are fed into the simulation, and Figure 4b shows the estimation of the slower vehicle's location at $t = 450$. We can see that the area where the slower vehicle is possibly in has shrunk to a very small area as more and more observations are assimilated.



(a) Location of the slower vehicle at $t = 0$.



(b) Location of the slower vehicle at $t = 450$.

Figure 4: Location of the slower vehicle; real location is shown in red, while locations in particles are shown in blue.

At time t when data assimilation is carried out, the location of the slower vehicle is estimated by

$$\hat{l}_t = \frac{1}{N_p} \sum_{i=1}^{N_p} l_t^i$$

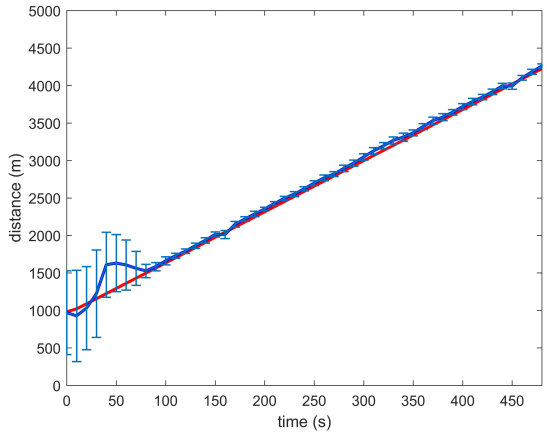
where N_p is the number of particles; l_t^i is the location of the slower vehicle in the i-th particle. Figure 5a shows the real trajectory and the estimated trajectory. The error bar narrows as more and more data is fed into the simulation , which means we are more certain about the estimated location of the slower vehicle.

We define estimation error in the i-th particle as

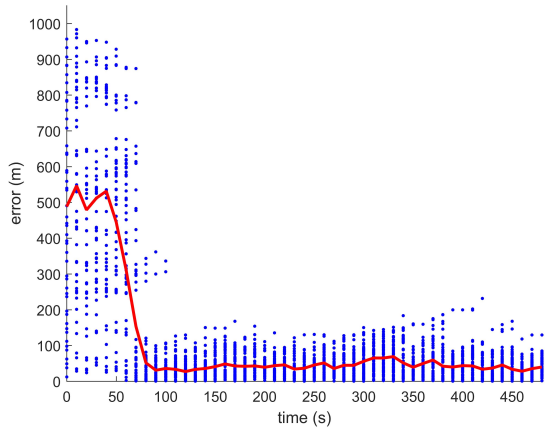
$$e_t^i = |l_t^i - l_t|$$

where l_t is the real location of the slower vehicle at time t. The mean error, $\bar{e}_t$, is defined by

$$\bar{e}_t = \frac{1}{N_p} \sum_{i=1}^{N_p} |l_t^i - l_t| = \frac{1}{N_s} e_t^i$$



(a) Trajectory of the slower vehicle; real trajectory is shown in red, while estimated trajectory and standard deviation are shown in blue.



(b) Estimation error; mean error is shown in red, while errors in particles are shown as blue dots.

Figure 5: Trajectory of the slower vehicle and its estimation error.

Figure 5b shows the evolution of estimation errors as simulation proceeds. The result shows that the estimation error converges quickly to a low level as observations from the real system are assimilated over time, which proves that the proposed method can accurately locate the slower vehicle.

5. CONCLUSION

Applying SMC methods in discrete event simulations requires a variety of capabilities of the simulation environment, such as pausing or stopping the simulation, retrieving state from or setting a new state to the simulation, etc. In this paper, we propose an rollback based implementation of SMC methods in object oriented discrete event simulations, which facilitates the application of SMC methods in discrete event simulations. An identical-twin experiment in a discrete event traffic case is carried out to evaluate the accuracy of the proposed method. The results show that the simulation with the rollback based SMC method can accurately estimate the slower vehicle position on the road, and thus proves the effectiveness of the proposed method.

Acknowledgment

This research is mainly financed by the China Scholarship Council (Grant NO. 201306110027), and two National Natural Science Foundations of China (Grant NO. 61374185 and 61403402, respectively).

REFERENCES

1. Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing 50*, 2 (2002), 174–188.
2. Bai, F., Gu, F., Hu, X., , and Guo, S. Particle routing in distributed particle filters for large-scale spatial temporal systems. *IEEE Transactions on Parallel and Distributed Systems* (2015), to appear.
3. Carton, J. A., and Giese, B. S. A reanalysis of ocean climate using simple ocean data assimilation (SODA). *Monthly Weather Review 136* (2008), 2999–3017.
4. Chi, S.-D., Lee, J.-O., and Kim, Y.-K. Discrete event modeling and simulation for traffic flow analysis. In *IEEE International Conference on Systems, Man and Cybernetics*, vol. 1 (1995), 783–788.
5. Constantinescu, E. M., Sandu, A., Chai, T., and Carmichael, G. R. Ensemble-based chemical data assimilation. I: General approach. *Quarterly Journal of the Royal Meteorological Society 133*, 626 (2007), 1229–1243.
6. Darema, F. Dynamic data driven applications systems: A new paradigm for application simulations and measurements. In *Computational Science - ICCS 2004*, M. Bubak, G. Albada, P. M. Sloot, and J. Dongarra, Eds., vol. 3038 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, 662–669.
7. Djurić, P. M., Kotecha, J. H., Zhang, J., Huang, Y., Ghirmai, T., Bugallo, M. F., and Miguez, J. Particle filtering. *IEEE Signal Processing Magazine 20*, 5 (2003), 19–38.
8. Fujimoto, R. M. *Parallel and Distributed Simulation Systems*. Wiley New York, 2000.
9. Gillijns, S., Mendoza, O., Chandrasekar, J., De Moor, B. L. R., Bernstein, D., and Ridley, A. What is the ensemble Kalman filter and how well does it work? In *American Control Conference* (2006), 4448–4453.
10. Ho, Y.-C. Introduction to special issue on dynamics of discrete event systems. *Proceedings of the IEEE 77*, 1 (1989), 3–6.
11. Hu, X. Dynamic data driven simulation. *SCS M&S Magazine II*, 1 (2011), 16–22.
12. Huang, X.-Y., Xiao, Q., Barker, D. M., Zhang, X., Michalakes, J., Huang, W., Henderson, T., Bray, J., Chen, Y., Ma, Z., Dudhia, J., Guo, Y., Zhang, X., Won, D.-J., Lin, H.-C., and Kuo, Y.-H. Four-dimensional variational

data assimilation for WRF: Formulation and preliminary results. *Monthly Weather Review 137*, 1 (2009), 299–314.

13. Ide, K., Courtier, P., Ghil, M., and Lorenc, A. C. Unified notation for data assimilation : Operational, sequential and variational. *Journal of the Meteorological Society of Japan, Special Issue on "Data Assimilation in Meteology and Oceanography: Theory and Practice" 75*, 1B (1997), 181–189.
14. Kang, D., Kong, J., and Choi, B. K. DEVS modeling of urban traffic systems (WIP). In *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium* (2012), 16:1–16:6.
15. Lee, J., and Chi, S. Using symbolic DEVS simulation to generate optimal traffic signal timings. *Simulation: Transactions of the Society for Modeling and Simulation International 81*, 2 (2005), 153–170.
16. Nichols, N. K. Data assimilation: aims and basic concepts. In *Data Assimilation for the Earth System*, R. Swinbank, V. Shutyaev, and W. Lahoz, Eds., vol. 26 of *NATO Science Series*. Springer Netherlands, 2003, 9–20.
17. Treiber, M., and Kesting, A. *Traffic Flow Dynamics: Data, Models and Simulation*. Springer-Verlag Berlin Heidelberg, 2013.
18. Wainer, G. A. *Discrete-Event Modeling and Simulation: A Practitioner's Approach*. CRC Press, 2009.