Ensemble learning for large-scale crowd flow prediction

Karbovskii, V.; Lees, M.; Presbitero, A.; Kurilkin, A.; Voloshin, D.; Derevitskii, I.; Karsakov, A.; Sloot, P.M.A.

DOI
10.1016/j.engappai.2021.104469

Publication date
2021

Document Version
Final published version

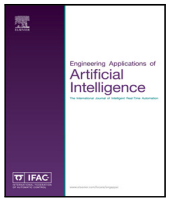
Published in
Engineering Applications of Artificial Intelligence

Engineering Applications of Artificial Intelligence 106 (2021) 104469

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



Ensemble learning for large-scale crowd flow prediction

Vladislav Karbovskii [a,b], Michael Lees [a,*], Alva Presbitero [c,d], Alexey Kurilkin [d], Daniil Voloshin [e], Ivan Derevitskii [d], Andrey Karsakov [d], Peter M.A. Sloot [a,d,f]

[a] University of Amsterdam, The Netherlands
[b] Deutsche Telekom, Russia
[c] Asian Institute of Management, Philippines
[d] ITMO University, Saint Petersburg, Russia
[e] Uptick Inc, Ireland
[f] Nanyang Technological University, Singapore

ARTICLE INFO

Keywords:
Crowd modelling
Artificial intelligence
Agent-based model
Machine learning
Ensemble learning
Kumbh Mela

ABSTRACT

The Kumbh Mela festival is the largest mass gathering in the world that is celebrated every three years. In 2016, it attracted over 70 million people to Ujjain, India. The Mahakal temple is the "heart" of the festival that attracts a huge number of pilgrims and needs to accommodate with massive crowds. These types of events pose significant safety challenges as large-scale mass gatherings are often associated with risks such as crowd crushes. There have been a number of serious incidents documented in recent history such as the Hajj crush at Mina, Mecca, Saudi Arabia (2006 and 2015), the Lame Horse crush during a fire at Perm, Russia (2009), the Love Parade disaster at Duisburg, Germany (2010), and the Kumbh Mela stampede at Allahabad, Uttar Pradesh, India (2013) to name a few. Safety assurance at events of such tremendous size is closely connected with crowd control and understanding the general behaviour of the crowd. One of the basic challenges in understanding crowd dynamics is being able to predict crowd flows at a particular location based on past/present flows from another location. There are several existing methods and models used to predict and manage crowd flow. In this paper, we introduce a novel method for short-term crowd flow prediction and show that it decreases the prediction error by 13% as compared to existing methods. The model is based on ensemble learning where we demonstrate that a combination of complementary methods with different a-priory assumptions can create better estimations. Utilizing a unique data set derived from CCTV camera recordings of pilgrims that we collected during the Kumbh Mela 2016 festival, we tested different methods from artificial intelligence and computational modelling, such as simple shift of time-series (time-shift), agent-based modelling, machine learning methods, and show how combinations of these different methods as an ensemble provide synergy to obtain better predictions. Our results demonstrate that agent-based modelling, when combined with other models, provides better predictive power especially in complex scenarios. These results point to something fundamental about the information contained within and generated by these methods. We anticipate that our research could be a starting point for further research of informational synergetic aspects of models and predictors.

1. Introduction

The Kumbh Mela is a mass Hindu pilgrimage, also known as the festival of chalice (Khanna et al., 2013) that happens every three years in one of four designated places in India, with each place chosen once for the festival every twelve years. Kumbh Mela is the biggest mass gathering in the world (Baranwal et al., 2015) and attracts millions of people every year: 100 million people in Allahabad in 2013 and 70 million in Ujjain in 2016. This accumulation of people at such a large scale is associated with certain risks that the government, and organizers need to carefully manage. An analysis of the last 40 years (Soomaroo and Murray, 2012) defines five main challenges for mass gathering safety that also apply to the Kumbh Mela: (a) overcrowding and crowd management, (b) fire safety (Sridhar et al., 2015), (c) medical preparedness and healthcare (Cariappa et al., 2015), (d) event access points and (e) emergency response (Greenough, 2013; Sridhar et al., 2015). Indeed, for such a large-scale event, global management and planning should also be included (Baranwal et al., 2015; Mehta et al., 2014).

* Corresponding author.
E-mail addresses: vladislav.k.work@gmail.com (V. Karbovskii), m.h.lees@uva.nl (M. Lees), lpresbitero@aim.edu (A. Presbitero), kurilkin@iac.spb.ru (A. Kurilkin), achoched@gmail.com (D. Voloshin), iderevitskiy@gmail.com (I. Derevitskii), kapc3d@gmail.com (A. Karsakov), p.m.a.sloot@uva.nl (P.M.A. Sloot).

https://doi.org/10.1016/j.engappai.2021.104469
Received 8 February 2021; Received in revised form 16 August 2021; Accepted 5 September 2021
Available online 20 September 2021

The term Kumbh Mela comes from Sanskrit (Kumbha — a pitcher, Mela — fair). During the pilgrimage, people take a dip in the river and take part in different activities such as visiting sacred places including temples. In Ujjain, the Mahakal Temple plays a key role. It is the heart of the festival and attracts a majority of the pilgrims that in turn leads to the accumulation of significant concentrations of the crowd inside a relatively small and confined space. These factors are our motivation for considering the Mahakal temple (Fig. 1) as the focus of our research.

Incidents related to ineffective preparation and poor crowd management are frequent and dangerous. Injuries can be prevalent and life threating e.g., tissue damage, asphyxia, heatstroke, insulation etc. (World Health Organization, 2015). Overcrowding especially becomes even more critical in religious festivals due to the sheer volume of participants as well as participants' average age and health state (Turris et al., 2014). Hence, simulations of real-world data can help to develop real world understanding of crowd dynamics. A crowd simulation of the Love Parade disaster in 2010 (Zhao et al., 2020) has been implemented based on the data from official reports. Bratsun in Bratsun et al. (2013) simulated the disaster in Lame Horse in 2009 based on synthetic data and official reports. Authors in Mordvintsev et al. (2014) simulated crowd evacuation during the flood in St. Petersburg. Helbing et al. (2007) performed observation and empirical study of crowd dynamics during Hajj in 2006.

Our aim is to introduce a novel method for short-term crowd flow prediction that improves prediction quality. In this research, we define crowd flow as the number of people moving from one location to another inside a confined space (in our case, a corridor) with reference to a "marker" per unit time. One could imagine this marker as a straight line that is virtually drawn perpendicular to the corridor, making it possible to count the number of people crossing this line per unit time. We emphasize that we only look into the movement of a crowd in a corridor where there could only be two general directions: pedestrian flows moving in opposite directions and sharing the same corridor space.

One popular method for analysing crowds is through the use of agent-based modelling (Bratsun et al., 2013; Turris et al., 2014). This method has major advantages such as: (a) modelling and simulation of individual agents with their own behaviour, (b) capability to take into account complex geometry and space, including walls and obstacles. However, this method is often complex and computationally demanding as compared to alternative approaches. It also requires extensive input and validation data in order to gain sufficient confidence in the prediction. There are a number of existing tools and applications of agent based modelling and a comprehensive review can be found in Musse and Thalmann (2001) and Duives et al. (2014).

Flow prediction is also possible using machine learning algorithms. Among the machine learning algorithms that we used for this research, only the recurrent neural network has been applied to crowd flow before Duives et al. (2019). However, the application of machine learning is more common in vehicular traffic flows (Smith and Demetsky, 1994; Vlahogianni et al., 2005; Zheng et al., 2006). Unlike agent-based modelling, machine learning approaches are based on a completely different philosophy, rather than providing a mechanistic description of the system, machine learning identifies patterns, and correlations in the system's input and output.

Since there seems to be no single ideal method for modelling crowd dynamics, we would assume that models that are derived from various methods based on different philosophies, would make different predictions for different situations. We emphasize that in the context of our work, *methods* refer to techniques that are applied, but not limited to, modelling crowd dynamics. *Models*, on the other hand, are what we refer to as "products" or outcomes when a method or a collection of methods used in predicting crowd dynamics are used together to better understand a phenomenon. Hence, perhaps different methods may work better under different conditions.

Automating the generation of crowd behaviours has been the focus of several recent works that made use of data-driven, approaches (Zhong et al., 2017, 2015, 2014). The core idea is to learn via examples from video data. The examples are then used in the simulation as behavioural rules to generate agents' movement. Approaches that aim at learning global motion patterns of crowds from video data have been proposed in [25–30]. Global motion patterns can be integrated with the crowd simulation model in order to generate realistic crowd behaviours. However, the knowledge learned by these methods are highly scenario-specific (e.g. examples and global motion patterns), such that once the scenario is changed, the learned knowledge may no longer be able to generate the desired crowd dynamics.

We hypothesize that by combining these methods, it may be possible to increase the quality of the prediction. This technique of combining different models is known as ensemble learning and it has been studied (Dietterich, 2000) in statistics and machine learning. The application of ensemble learning to understand crowds is a relatively unexplored area. In fact we are not aware of existing attempts other than our own published research (Kiselev et al., 2016) especially at combining seemingly disparate models like state-of-the-art machine learning methods, time shift model, and agent based modelling. Unlike our previous work, which combines different agent-based models, our current research combines models based on different philosophies and evaluates this approach on real data obtained from the Kumbh Mela festival.

The paper is structured as follows: data set collection, processing, and post-processing are presented in Section 2. The methods used in modelling, such as time-shift modelling, agent-based modelling, machine learning algorithms, and ensemble models, are presented in Section 3. Results are presented in Section 4. Finally, we give our summary and conclusion in Section 5.

2. Data set

2.1. Data collection

Data collection in the Mahakal Temple was divided in two stages:

Stage 1: construction of a 3D model of the temple by taking the measurements of all rooms and corridors. See Figs. 1 and 2.

Stage 2: crowd flow video data collection

In Stage 1, every necessary measurement (*i.e.* of walls, corridors, etc.) was conducted using a high-resolution laser rangefinder with an accuracy of 0.003 m. Stage 2 was performed during one of the most crowded days of the festival (May 5, 2016), also known as "Vaishakh Shukla". Video data was collected during peak time which involved 90 min, from 15:00 to 16:30. The data set includes 5 videos from CCTV cameras captured during the same period inside the Mahakal temple. Fig. 2 shows the main flow of the pilgrims. We emphasize that there are two flows of pilgrims merging (blue and yellow Sections) between Cameras 3 and 5, where the flow from the VIP entrance (yellow, Camera 4) merges the crowd flow from the general entrance (blue). Segments in Fig. 2 correspond to paths between cameras, for example segment 01–02 runs from Camera 01 to Camera 02 etc. Flows are unidirectional and pilgrims cannot leave the flow or merge except at one merge point, where pilgrims coming from Marbal Galiyara (Camera 03) as well as from the Ramp (Camera 04) merge towards Sabha Mandap ramp (see Fig. 2). This means there should be *conservation of mass*.

Below we summarize the details for each camera including the resolution, FPS, and a short description of the viewing angle. Fig. 3 shows example images from each of the cameras where we see a significant variance in quality and coverage of the cameras.

Camera 01 - "**Main Gate**", 1920x1080, 25FPS. Entrance in the temple. The video is of high quality and the angle and position of the camera are also good. However, there are nonlinear distortions in the angle of the camera.

Fig. 1. 3D view of Mahakal Temple and crowd near the entrance.

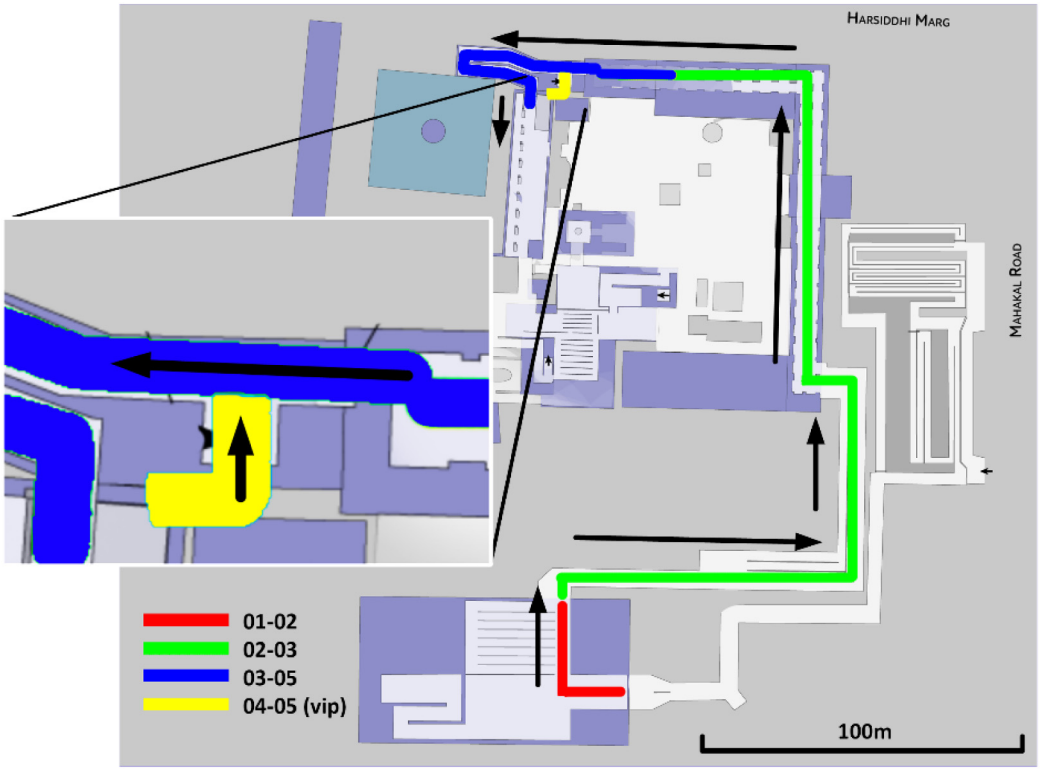


Fig. 2. Scheme of flows in the temple. Paths from one camera to another are represented by different colours. Red is from Camera 01 to Camera 02, green is from Camera 02 to 03, blue is from Camera 03 to 05, and yellow is from Camera 04 to Camera 05. The crowd moving from Camera 03 and 04 merges and proceeds towards where Camera 05 is located.

Camera 02 - "**Fc Barricade**", 1920x1080, 25FPS. Corridor next to the entrance. The video is also of high quality and the angle and position of the camera are also relatively good.

Camera 03 - "**Marbal Galiyara to Ramp**", 960x576, 20FPS. Gallery. Video is of average quality but has good position and angle.

Camera 04 - "**Ramp Turn**", 960x576, 25FPS. Gallery. Video quality is also average and has nearly the same position as Camera 04. However, the camera is directed towards the turn, where the VIP pilgrims merge with the main flow. Due to water sprays inside the temple, this video appears foggy.

Camera 05 - "**Sabha Mandap ramp**", 1920x1080, 20FPS. Room before sanctuary. Video quality is high but it has a low framerate. However, both position and angle are good, which makes this camera still useful.

2.2. Data processing

Data processing comprises of two stages. Stage 1 pertains to detecting individuals in the video, and Stage 2 predicts the trajectories of movement of each individual. The final result is the crowd flow.

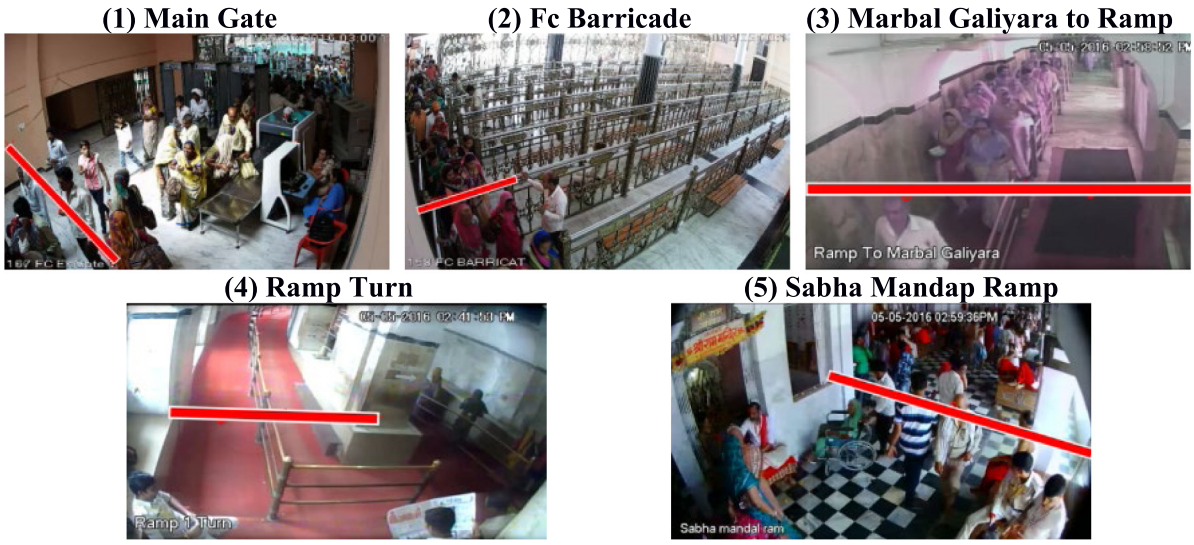


Fig. 3. Cameras, red lines are virtual detectors.

In order to extract pedestrian data or the number of pedestrians crossing a specific corridor in the temple from the videos, we position a "marker" denoted by a red (or white) virtual line shown in Fig. 4, and count the number of pedestrians crossing this marker.

Detecting individuals in the videos was challenging due to frequent overlapping of objects as well as the changes in the linear size of the objects or the people, and the level of lighting in the videos. In order to improve the quality of the image, we calibrated the cameras by filming the calibration target (Heikkila and Silven, 1997; Zhang, 2000). The calibration target is a standard "chess board" square pattern that has 140 squares (70 black and 70 white) and each square has a dimension of 5x5 cm.

In order to address this concern, we applied tracking-by-detection method (Breitenstein et al., 2011; Andriluka et al., 2008). This algorithm uses the current position of the object from the detector and creates a model of movement for the object. It then searches for the same object in the subsequent frames. The process of getting crowd flow from raw videos are summarized by the following steps.

1. ***Detecting*** the people in each video frame.
2. ***Tracking*** people between video frames, and assigning "tracks" that are defined as the paths the individuals take when moving from one location to another as indicated by the green lines in Fig. 4.
3. ***Counting*** the number of people "flowing" or moving through each camera with the use of the tracks mentioned in 2.

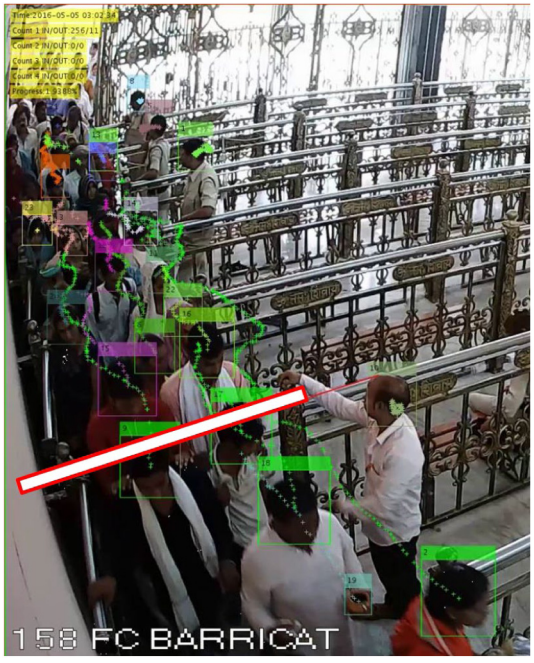


Fig. 4. Detection, tracking, and crossing example.

In Step 1, the choice of which object should be *detected* is crucial in determining the quality of measurement. In our case, people, who are part of and moving with the crowd, cannot always be observed in full as their faces may not always be directed towards the camera. This is sometimes due to the angle and direction at which the camera was installed. Therefore, we chose to use heads as the object for detection. We utilized two methods for head detection: ACF (aggregate channel features), and HOG (histogram of oriented gradients).

In Step 2 we use head detection to *track* the people between video frames. The movement of a particular individual can be tracked from frame to frame, and this is what we call a "*trajectory*".

Finally, for Step 3, in order to measure crowd flow, we set up a virtual barrier, such as the white line shown in Fig. 4, where each person's "trajectory" is *counted* when it crosses this marker.

For the two detection methods that we applied, we tuned the parameters with grid search (Tuning the hyper-parameters: Exhaustive Grid Search) method to find the best performance for our videos. In ACF (Dollar et al., 2012; Dollár et al., 2014), we utilized 4 training stages, a negative sample factor of 5, and 2048 maximum number of weak learners. However, the quality of the result is not enough to build tracks between frames. The main reason for this is that there is no enough contrast between the objects we try to detect (i.e., heads) and the background. HOG (Dalal and Triggs, 2005), where 5 and 20 cascade stages, 0.01–0.2 acceptable false alarm rates, and 0.995 minimum true positive rate are utilized, shows better results than ACF. However, it produces many false positives as it can also detect other objects in the background.

Since using HOG alone gives a significant number of false positives, we opted to couple HOG with a convolutional neural network (CNN). That is, we fit HOG in such a way that it gives us a significant amount of positive detections in general. This pertains to a maximum number of both true and false positives. We then employ a convolutional neural network, based on the method described in Gao et al. (2016), to classify

Table 1
Video processing results.

Method	Detection Mean Average Precision	Total RMAE (# of *people per 15 seconds*)
Ground truth	–	–
ACF	0.27	9.38
HOG	0.14	4.67
HOG & CNN	0.45	3.84

the detections and filter out false positives. To do this all detections were divided in two groups: training and test. For learning, the data was divided: 80% of the data was used for training, while 20% was used for testing. The training part of the data was manually processed to classify them as true positives and false positives in preparation for the learning of the classifier: first class — true positive, second class — false positive.

In order to assess the quality of the detection, we look at the mean average precision for the methods that we used. Mean average precision is a standard metric for object detection problems. High mean average precision implies that the method is able to detect substantially more relevant objects (heads) than irrelevant ones (non-heads). Ideally, we hope to develop a method with high mean average precision.

To summarize, the mean average precisions for each method are as follows: ACF:0.27, HOG:0.14 and HOG with neural network:0.45. Our results show that HOG coupled with the neural network exhibits the highest mean average precision as compared to other methods that were tested. We further compare the performance of each method by looking at predicted trajectories against data in Stage 2.

The second stage of data processing pertains to the prediction of the trajectories that the movement of individuals creates. In order to do this, we applied a Kalman filter with the following specification: the motion model used assumes that people travel at a constant velocity. The initial location of a person was set as the centroid of a box that bounds their head. The initial estimate for the uncertainty of the variance is set to 2.1. Deviation between the selected and actual model is 5.5, and the variance for inaccuracy is set to 100, parameters are tuned with grid search (Tuning the hyper-parameters: Exhaustive Grid Search) method. Fig. 5 shows the estimated flow for all methods compared against the flow observed from the cameras.

The final product of video processing is the crowd flow. We then look at the mean absolute error of the crowd flow as a metric to determine the performance of the models. The mean absolute errors between the real data (observation) and the methods used are 3.84 for HOG & CNN, 4.67 for HOG and 9.38 for ACF. Based on our results, HOG & CNN predicts better than the other methods, while ACF is the lowest performing method among the three.

To summarize, to accomplish Stage 1 (detection of individuals in the video) if the data processing, first we utilize a detector algorithm that returns rectangular frames where each frame is centred around a detected head from the videos. A tracking algorithm then builds trajectories connecting the rectangular frames of the same objects that were detected between video frames. This is followed by a simple algorithm that counts the number of crossings through a virtual line. The output data is represented by events that are plotted through time, where each event corresponds to the number of crossings detected through the virtual line. We see Fig. 5 that the method implemented for detection and tracking gives results that are reliable enough to be used for our flow prediction. The mean average precision and the total root mean absolute error (RMAE) between the predicted crowd flow against the observed data for all methods used in video processing are summarized in Table 1.

2.3. Data analysis

What we have so far is a series of events plotted over a continuous time frame. In order to make sense of this data, we do post-processing to generate a time-series of crowd flow for each camera as summarized by the following steps: (1) selection of discreetness and discretization, (2) normalization, and (3) filtering.

Discretization is necessary because the data that we have is represented in a continuous time and still needs to be transformed into a time series. To select the discreetness of the time series, two conditions should be considered: (a) stability of the time series, and (b) normality of the distribution. Selection of discretization was based on the mean absolute deviation and standard deviation of the resulting time series (see Appendix A). The next step is data normalization. This step can only be applied between specific cameras where people cannot leave or merge with the flow. To normalize the flow, say for Camera 2 with respect to Camera 1, we use a coefficient that is calculated through the following equation:

$$k_{n2} = \frac{\sum_{t=0}^{t=T}{}_{c1} f(t)}{\sum_{t=l}^{t=T+l}{}_{c2} f(t)} \tag{1}$$

where the summation of crowd flow $f(t)$ for Camera 1 ($c1$) from time $t = 0$ to $t = T$ is divided by the summation of crowd flow $f(t)$ for Camera 2 ($c2$) from the lag time $t = l$ to $t = T + l$.

Since the processed data contains noise and errors (due to measurement, analysis, and discretization) we still see abrupt changes in the time series. To reduce these abrupt changes, data filtering is applied. A widely known method that addresses this concern is the Kalman filter (the application of Kalman filter here is not related to the application of Kalman filter in Section 1.2). Fig. 6 shows an example of a time series for Camera 02 that has been processed using Kalman filter. The time series data for all cameras are presented in Appendix D.

We then have estimated basic metrics for all pairs of cameras in the temple. The Distance (D) between cameras was measured physically. The average time T_{avg} was estimated using cross correlation analysis. Cross correlation was utilized on the data sets for each pair of cameras, where the highest value in each pair's correlation is picked as the average time shift (see Appendix C for details). The average speed V_{avg} for every scenario was calculated given the segment's distance and time. In order to check time shift calculations, we manually selected and tracked two pilgrims across all cameras: the first pilgrim has a speed lower than average (T_1, V_1), while the second pilgrim has speed that is higher than average (T_2, V_2).

Table 1 shows all pairwise metrics calculated for the cameras. For instance, the label 01–02 refers to Cameras 01 and 02 respectively. Distance $\boldsymbol{D}$ corresponds to the distance between the two cameras measured in metres, $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ are the times it takes for pilgrim 1 and pilgrim 2 to travel distance $\boldsymbol{D}$ respectively, while $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ are their corresponding speeds. $\boldsymbol{T}_{\boldsymbol{avg}}$ and $\boldsymbol{V}_{\boldsymbol{avg}}$ refer to the average travel time and average speed of all the pilgrims traversing the entire distance. Turns refer to the number of corners, while complexity factors list down some specifics in the route. We pick combinations of Cameras 01–02, 02–03, and (3&4) - 05 as Scenarios 1, 2, and 3, respectively (see Table 2).

3. Modelling methods

Having processed the time series data, we now proceed with formulating the short-term crowd flow predictions for specific spatial points (e.g. positions of CCTV cameras or other detectors) as a time series prediction, where one point in the timeseries corresponds to the number of people passing the virtual line within the selected time window (we used 15 s, see Appendix A; e.g. one data point — '25 people per 15 s'). We then predict the output timeseries of crowd flow based on the input time series of crowd flow. To make short-term crowd

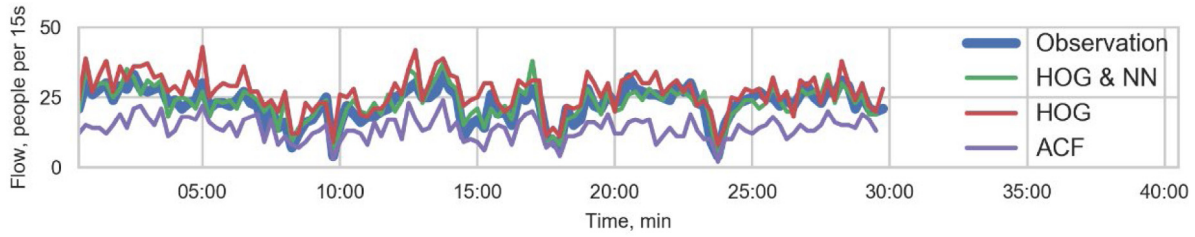


Fig. 5. Evaluation of processing methods against observed data.

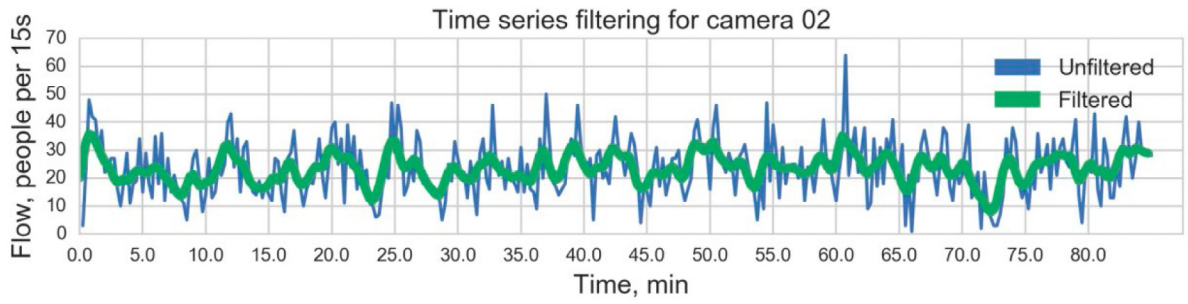


Fig. 6. Filtered and unfiltered data.

Table 2
Measurements and observations for all camera pairs.

	01 – 02	02 – 03	(03&04) – 05
D, m	18	259	58
T_{avg}, s	45	345	90
T_1, s	36	256	77
T_2, s	51	321	65
V_{avg}, m/s	0.4	0.75	0.64
V_1, m/s	0.5	1.01	0.75
V_2, m/s	0.35	0.8	0.89
Turns	1	5	6
Complexity factors	–	long path, police control	merging of flows, pilgrims pray
Scenario	**1**	**2**	**3**

flow predictions, we test a number of different approaches: (a) time shift, (b) machine learning algorithms, (c) agent-based modelling, and (d) their ensembles.

Each method has its own limitations and constraints. For instance, time shift is limited to cases when people cannot leave the crowd when moving between two specific points due to the assumption of conservation of mass. Machine learning methods do not explicitly make such assumptions. However, their performance is fully determined by the quality of training data available. Agent-based model, on the other hand, attempt to capture the system in a mechanistic and physical way — this includes a full specification of the space and assumptions made about the behaviour and mobility of the people. This, in principle, assuming the model is valid and correct, could offer advantages since it can take into account complex geometries (corridors, bottlenecks etc.), gaps in the data and better deal with discontinuities in the flow. Another advantage of agent-based models is their capability of predicting not only values of flow or density at specific points, but also estimating fields and being able to recover values at arbitrary points. The drawback is that agent-based models are often hard to validate, require large amounts of data and are computationally intensive. These approaches differ not only on the underlying mathematics, but are developed on fundamentally different philosophies, which means they are likely to make better or worse predictions in different situations.

Our approach is to try and combine these methods to profit from possible synergies in the methods. To do this, we combine these different approaches in an ensemble model. The result is a new approach to short-term prediction of crowd flow dynamics.

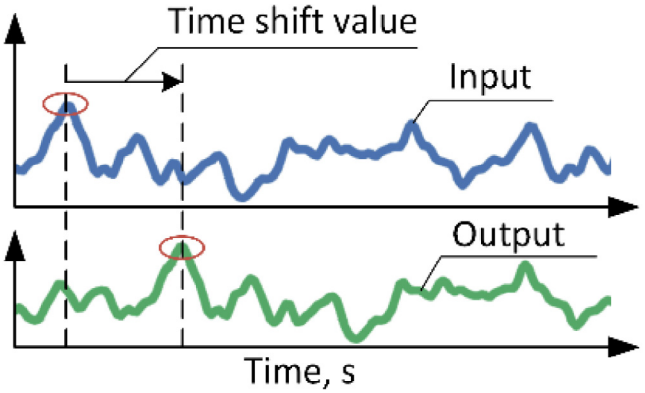


Fig. 7. Time shift. The predicted time series is a simple shift of the time series by some delta I. Hence, the predicted output y_{t+I} is simply the value at x_t.

3.1. Time shift

The simplest method we employ is a shifted flow, where we move or shift the time series forward by some delta. Despite its limitations due to the simplistic underlying assumption, this method can give low predictive error for simple cases over short and non-complex building geometries (e.g. a short straight corridor). The model *shifts* the time-series (or sums them in cases when there are two input timeseries like

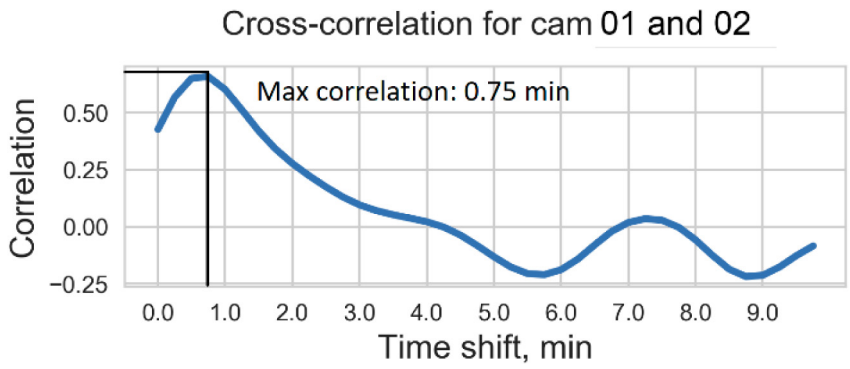


Fig. 8. Cross correlation between Camera 01 and 02 for different time shifts (minutes).

merging flows). The simple formalization is presented in Eq. (2), where $\hat{y}_{t+l}$ is the output value at time $t+l$, l is the shift value (lag), and x_t is the input at time t. The method is illustrated in Fig. 7.

$$\hat{y}_{t+l} = x_t \tag{2}$$

To calculate the time shift value of crowd flows between two adjacent cameras, we utilized cross-correlation (see Fig. 8). The details of the cross-correlation are shown in Appendix C. Fig. 9 shows an example result for Cameras 01 and 02 (Scenario 1). Here, the result of the time shift (and cross-correlation) gives maximum correlation for a time shift of 45 s (0.75 min in Fig. 9). Given the average speed of the individuals is 0.4 m/s and the distance between the two cameras is 18 m, this result seems consistent. In Fig. 9, we see the fit of a smoothed flow for Camera 1 shifted by 45 s to overlap with Camera 2. Results for all three scenarios are presented in Section 4.

3.2. Machine learning methods

Machine learning methods can also be used to construct a short-term forecast of the dynamics of the crowd at the control points (for CCTV cameras or other detectors).

In order to apply machine learning methods to timeseries, we have to consider that the order of the datapoint is important. For a machine learning model $f(x_t)$ that outputs the prediction $\hat{y}_{t+l}$ (see Eq. (3)) at timestep $t+l$, where t is time, and l is lag. t_{max} is the total time, the machine learning algorithm solves an optimization task by minimizing the *cost function* in Eq. (4). y_{t+l} is the so-called *ground truth* or the data being modelled and $\hat{y}_{t+l}$ is the model prediction, and $t_{max} - l$ is the number of objects in the training data set. The cost function for regression models represents the summation of the square of the error between the actual data and the predicted values divided by the total number of objects in the training data set. The same logic is applied here as in the time shift model.

$$\hat{y}_{t+l} = f(x_t) \tag{3}$$

$$minimize \frac{1}{t_{max} - l} \sum_{t=l}^{t_{max}} (\hat{y}_{t+l} - y_{t+l})^2 \tag{4}$$

In this paper, we apply five commonly used machine learning methods. These are *linear regression, gradient boosting regression, dense neural network, long short-term memory network, and support vector regression*.

Linear regression is a simple approach for studying and modelling linear relationship between a target (true answer) and one or multiple predictors (Wetherill and Seber, 1977). **Gradient boosting** is a method that builds a model by iteratively boosting (training with bootstrapping and aggregation) weak models (normally decision trees) into a stronger model (Friedman, 2002). Artificial neural networks are based on connected artificial neurons that transform input signals with so-called "activation functions" to produce output signals used in the predictions (Hinton, 1990). We take advantage of a fully connected layer, which learns from all combinations of features in previous layers, in a **Dense Neural Network**.

One of the appeals of recurrent neural network over traditional neutral network is that recurrent neural networks are able to connect the present task to the previous ones. This capability of recurrent neural networks is particularly useful in timeseries data because most likely a particular event in a timeseries depends on the previous events. One could imagine that in videos, the present frame might be explained by the previous video frames. However, as the gap between the frames become larger, connecting the two states from frames becomes challenging and prediction becomes more difficult. This is where **Long Short-Term Memory Network** comes in. It is a special type of recurrent neural network that is capable of learning long-term dependencies. Hence it is able to remember information for a long period of time. For a comprehensive discussion, see Hochreiter and Schmidhuber (1997). **Support Vector Regression** uses the same principles as the Support Vector Machine where the decision boundary is chosen based on the data points closest to the hyper plane or the support vectors (Smola and Schölkopf, 2004). Support vector regression is different from a simple linear regression in that instead of minimizing the error between the predictions and ground truth, it tries to fit the error within a certain threshold. That is, decision boundaries are established as margin of tolerance for the data points in order to decide for a better fitting model.

For this paper we used the Scikit Learn[1] (Pedregosa et al., 2011) (version 0.18.2) as a toolkit for the linear regression and support vector regression, XGBoost[2] (version 0.6) was used as implementation of gradient boosting, and Tensor Flow[3] for neural networks (multi-layer perceptron).

Time series data is always characterized by the correlation between observations that are located close to each other in the timeseries (autocorrelation). However, traditional cross-validation methods, including KFold, assume that samples are independent. Thus, it is important to ensure the predictions are evaluated on the "future" data. We therefore use KFold especially implemented on timeseries data. This means that we train our models on the "past" data, which we define as our training data set and validate our model on the "future" data or the test data set.

In Scikit Learn implementation, this cross-validation method that specifically deals with timeseries data is called TimeSeriesSplit (Pedregosa et al., 2011). Thus, cross-validation for timeseries was used to optimize hyperparameters (*i.e.* parameters of the model, that cannot be estimated from data) of machine learning algorithms.

Model optimization was implemented by using the grid search method, which employs an exhaustive search through a defined hyperparameter space and has an implementation in Scikit Learn. After

[1] http://scikit-learn.org.
[2] https://xgboost.readthedocs.io.
[3] https://tensorflow.org/.

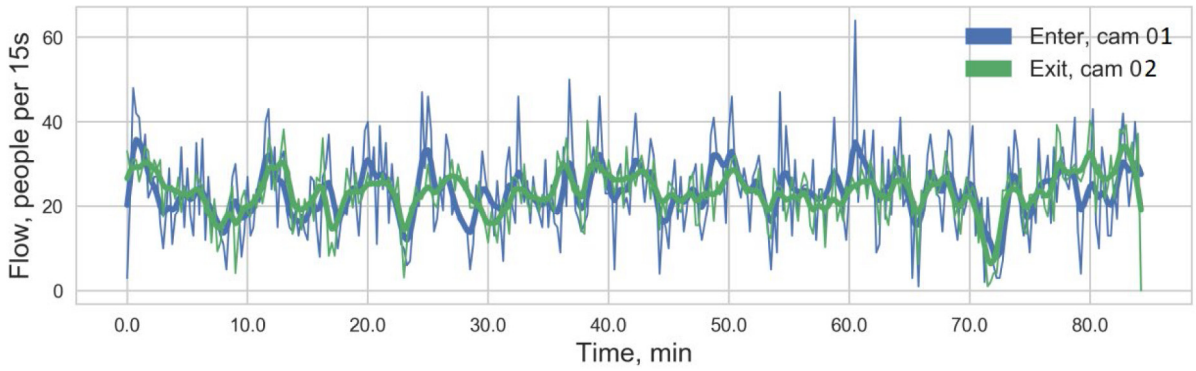


Fig. 9. Shifted flow for Camera 01, compared to Camera 02.

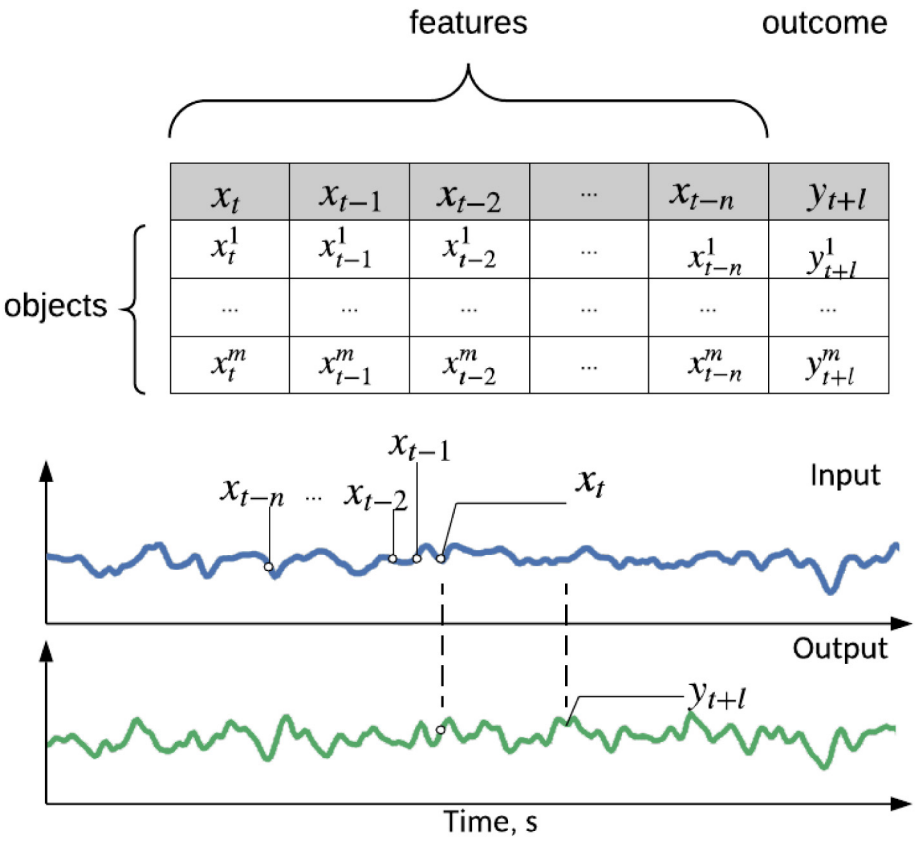


Fig. 10. Object-feature matrix. The features used to predict output y_{t+l} are x_t as well as data points up until $t-n$ time steps away in the input timeseries data.

testing, we used the following parameters for each of the methods: (a) Scikit Learn implementation of linear regression does not require additional parameters, (b) neural network with adaptive learning at learning rate 0.001, and one hidden layer with 10 neurons, (b) for XGBoost, we used 100 estimators and a learning rate of 0.001.[4]

The input for the machine learning model is an object-feature matrix, where the objects are the data points in the timeseries of the "past" data. The output of the machine learning model is the time series in the "future" data set. The set of features chosen for the model corresponds to the data points in the time series (x_t) and the lagging n data points (x_{t-n}) in the time series. A summary of the object-feature matrix and a visualization of how the output (y_t) is mapped back to the input timeseries is shown in Fig. 10. The motivation for this is that we assume that the outcome y_t may also be predicted by the data prior to x_t in the timeseries.

[4] Note that linear regression has no parameters.

3.3. Agent-based model

The agent-based crowd simulation is based on the PULSE framework (Karbovskii et al., 2018, 2015). The agent-based model models the set of individual agents and the environment. Agents are autonomous independent entities, and their state determines the state of the system. The environment has spatial and temporal parameters (Voloshin et al., 2015) that include the start and end of the simulation, as well as the simulation time step τ and different objects, such as points of interest (e.g. the sanctuary, in the case of the temple), obstacles (including walls and fences), points of inflow and outflow (entrances and exits). Every agent has three sub-models executed at each time step: (1) *Decision*: deciding or 'where to go', (2) *Navigation*: path planning and or 'how to go' (3) *Movement*: moving to the next waypoint while avoid collisions. Decision making in this case is quite simple as agents are designed to follow the prescribed path around the temple until the agent reaches the exit of the temple (see Fig. 11).

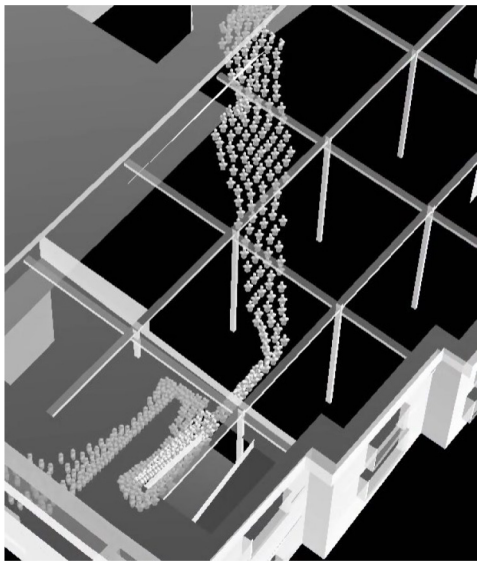


Fig. 11. Agent-based simulation visualization.

The *navigation* model receives input data about the target point of movement (from the prescribed path) and returns the desired direction of movement — typically in the direction of the path at some individual preferred speed. We use navigation fields (Patil et al., 2011) exactly for this purpose in our agent-based model. The navigation field is calculated for each point of interest. The field is represented by a regular grid, where each cell contains a vector with a direction of movement for agents in the area.

In order to move an agent in space and avoid collisions with the walls and other agents, an appropriate model should be used. To handle low and high crowd densities, we used a hybrid model based on forces and rules specified by HiDAC (High Density Autonomous Crowds) (Pelechano et al., 2007). This model is based on the popular Social Force model (Helbing and Molnar, 1995; Viswanathan et al., 2014), but one of its advantages is that it incorporates an addition rule mechanism, which helps to avoid oscillations at very high density. The position $\vec{r}_\alpha$ of each agent α on the time $t+1$ is calculated by Eq. (5):

$$\vec{r}_\alpha(t+1) = \vec{r}_\alpha(t) + \omega_\alpha(t) \cdot v_\alpha(t) \cdot \tau \cdot \hat{F}_a(t) + \vec{F}_\alpha^R(t) \tag{5}$$

where t — current time, $\vec{r}_\alpha(t)$ — current position of agent α, $\omega_\alpha(t)$ — stopping rule, helping to avoid undesirable oscillation behaviour, $v_\alpha(t)$ — current speed, τ — simulation step, $\hat{F}_a(t)$ — normalized desired direction of the agent, $\vec{F}_\alpha^R(t)$ — repulsion forces. $\hat{F}_a(t)$ is calculated by Eq. (6), and $\vec{F}_\alpha(t)$ by Eq. (7):

$$\hat{F}_a(t) = \frac{\vec{F}_\alpha(t)}{\left|\vec{F}_\alpha(t)\right|} \tag{6}$$

$$\vec{F}_\alpha(t) = \vec{F}_\alpha(t-1) + \vec{F}_\alpha^{At}(t)\,\omega^{At} + \sum_{\beta(\neq\alpha)} \vec{F}_{\alpha\beta}^{Ta}(t)\omega^{Ta} + \sum_B \vec{F}_{\alpha B}^{To}(t)\omega^{To} \tag{7}$$

where $\vec{F}_\alpha(t-1)$ — value of this force on previous step, it helps to avoid sharp changes of trajectory, $\vec{F}_\alpha^{At}(t)$ — attraction force to current goal, $\vec{F}_{\alpha\beta}^{Ta}(t)$ и $\vec{F}_{\alpha B}^{To}(t)$ — tangential forces, which help to avoid obstacles and other agents; ω^{At}, ω^{Ta}, ω^{To} — weights. A more detailed description of the HiDAC model is presented in Pelechano et al. (2007). As input data, the model takes one or multiple (in case of merging flows) time series that are used for agent generation to reproduce pilgrims flow.

3.4. Ensemble method

Each model, or method, has its own advantages and disadvantages. However, the general quality of the final result can be improved by combining multiple individual models. This is known as the ensemble method or ensemble learning.

Ensemble learning pertains to a machine learning paradigm that trains multiple learners to solve the same problem. The ensemble method combines multiple models to make better predictions than those that can be obtained from the individual component algorithms. The ensemble method is a widely-used approach in machine learning and statistics that has been shown to improve the predictive power of the final model (Dietterich, 2000; Opitz and Maclin, 1999).

There are fundamental reasons why applying ensemble learning could help us build better models. This is explained in detail in Dietterich (2000), we briefly list the main reasons below:

(1) statistical — ensemble models can get the average from different answers, taking weights into account, which reduces the risk of wrong answers

(2) computational — sub-models can suffer due to local optima, especially if they are based on local search. Ensembles can give better approximation by running a local search from many different points;

(3) representational — in the case when the answer is not in the hypothesis space of separate models, ensemble modelling helps by approximating multiple answers

There are multiple different ensemble methods, but the following families of ensemble methods are the most widely used (Pedregosa et al., 2019; DeFilippi, 2018; Hastie et al., 2009): *bagging*, *boosting*, and *stacking* methods.

Bagging typically starts by "bootstrapping" or taking a random sub-sample of data for each model in a way that all models are a little different from each other. A bootstrap sample is pooled together through subsampling from the training data set with replacement in such a way that the sample size is the same as that of the training data set. Hence, for a bootstrap sample, some examples from the training set may appear, but others may not. Each model then has different observations that are based on the bootstrap process.

Boosting, on the other hand, is an ensemble technique where the predictors are not randomly and independently sampled, but rather sequentially. This type of ensemble technique makes use of the fact that subsequent predictors learn from the shortcomings of prior predictors. As opposed to the bagging method where observations are chosen based on a bootstrap process, boosting chooses observations based on error.

Stacked generalization, or **stacking**, is an ensemble method that combines multiple models to one meta-model or ensemble model. Bagging and boosting are often applied to homogeneous models. However, we are dealing with heterogeneous models, or models that are of different nature. Therefore, we use the stacking ensemble method. The idea behind stacking is illustrated in Fig. 12.

There are no strict steps in preparing a stacking model, but the following summarized steps are explicitly or implicitly required (DeFilippi, 2018; Hastie et al., 2009): Preparing the ensemble model using stacking consists of the following steps.

Step 1: the data set is split into 40–40–20 parts, where the first data set (40%) is where the individual models are trained and validated using k-fold cross validation, second data set (40%) is used to train and validate the ensemble model, and the third data set is used to test the ensemble model (20%).

Step 2: individual models are trained and cross validated using data set 1.

Step 3: predictions using individual models serve as the input for the ensemble model.

Step 4: stacked models are trained and cross validated using data set 2.

Step 5: test data is prepared so that it conforms with the object-feature matrix of the ensemble model.

Step 6: predict using the ensemble model.

Step 7: ensemble model is validated using the test data set.

We discuss the steps in details below.

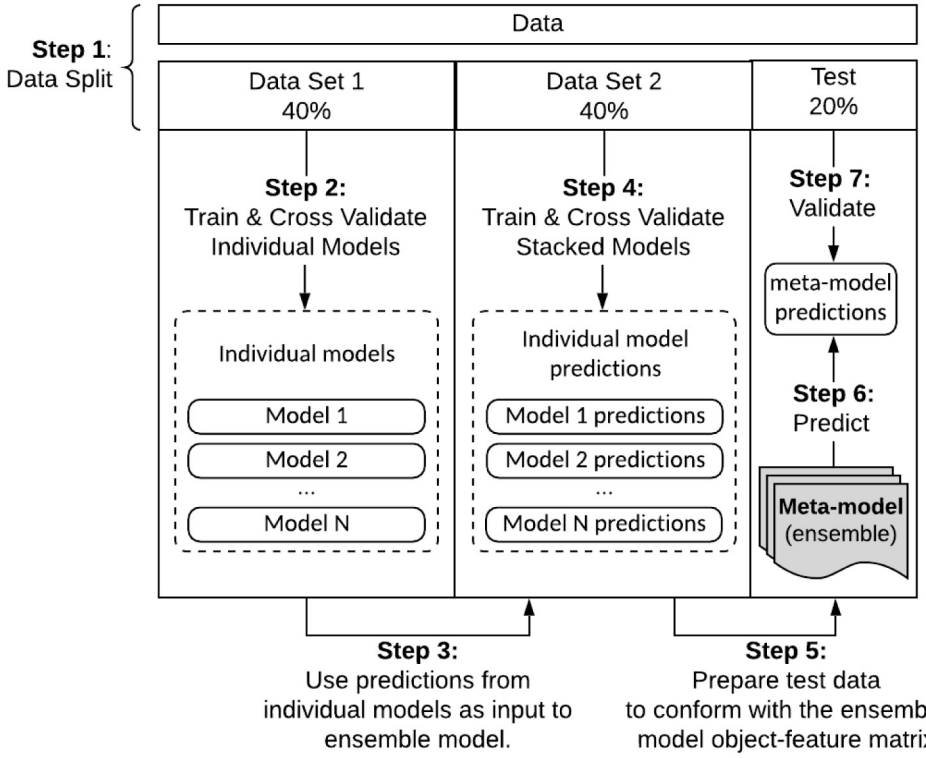


Fig. 12. Model Structure Diagram **of the Ensemble Model.** See main text for detailed discussion.

Splitting the data (*Step 1*) is required to avoid overfitting when preparing and validating the individual models and the ensemble model to achieve a generalizable model. More so, training and validating a model on the same data set may increase the risk of overfitting. Overfitting is a methodological mistake, and this happens when the model is prepared and validated on the same data. Hence, we make use of the first 40% of the data set (Data Set 1) in order to train the individual models as well as doing Kfold cross validation, the second 40% of the data set (Data Set 2) to train and validate the ensemble model, and the remaining 20% of the data set as training data to validate the final results of the ensemble model.

Since we are working with timeseries data, the manner at which the data is split should also be sequential. That is, all data points are ordered in time so that the training data points have earlier timestamps than the test data set.

The next step is (*Step 2*) individual model preparation where each of the models are trained (machine learning models) or calibrated (agent-based models) and cross-validated using data set 1. The predictions from the individual models are (*Step 3*) then used as input to the ensemble model. In other words, each feature in the object-feature matrix of the ensemble model corresponds to the predictions made using the individual models. For instance, the result from the agent-based model can be considered one feature (feature 1) for ensemble model, while the result from the linear regression is another feature (feature 2), and so on. The stacked model is then trained and validated (*Step 4*) using data set 2.

For the ensemble model, we simply used linear regression to predict the output. The order of features used in the ensemble model can either be important or not. This depends on the particular ensemble model used. In our case, we tried all machine learning models we have mentioned in Section 2.3 and found that the order in which the features were used has no effect on both the models that we used and the chosen ensemble model.

Step 5 involves preparing the test data set to conform with the object-feature matrix of the ensemble model. Specifically, each column in the *new* test data set corresponds to the predictions made using individual model on the *original* test data set. The ensemble model is (Step 6) then used to make predictions and (Step 7) validates on the *new* test data set.

4. Experiments: prediction of Mahakal temple crowd flow

In our experimental work, we focused on the following combinations of cameras: 01–02, 02–03, and 03&04–05. These pairs were chosen based on the quality of video data we have collected. The better the quality is, the more useable it is for further processing and interpretation. Moreover, we observed a low correlation between the time series for other pairs of cameras (see Appendix C).

Scenario 1: walking from Camera 01 to Camera 02

Scenario 2: walking from Camera 02 to Camera 03

Scenario 3: merging of flow from Cameras 03 and 04 and the subsequent movement of the merged flow to Camera 05

This raises an important question — *Would a combination of different methods give better predictions due to the synergetic effects for different cases?* In order to answer this, we performed a series of experiments as follows. Each individual model was applied independently to predict the flow between each pair of cameras. We then constructed the ensemble model using up to a maximum of 3 combinations of models and compared how they performed in comparison with the individual models they are made of. We assumed that different combinations of individual models in the ensemble exhibited synergy and that the combinations that made up the ensemble model can somehow influence the prediction in a positive way.

We used a dummy predictor that predicts the mean of the training set as base case scenario in order to compare the performance among our models. We evaluate the quality of our prediction based on coefficients of correlation and the error between the calculated result and real data. We repeated our experiments 1000 times for the agent-based model.

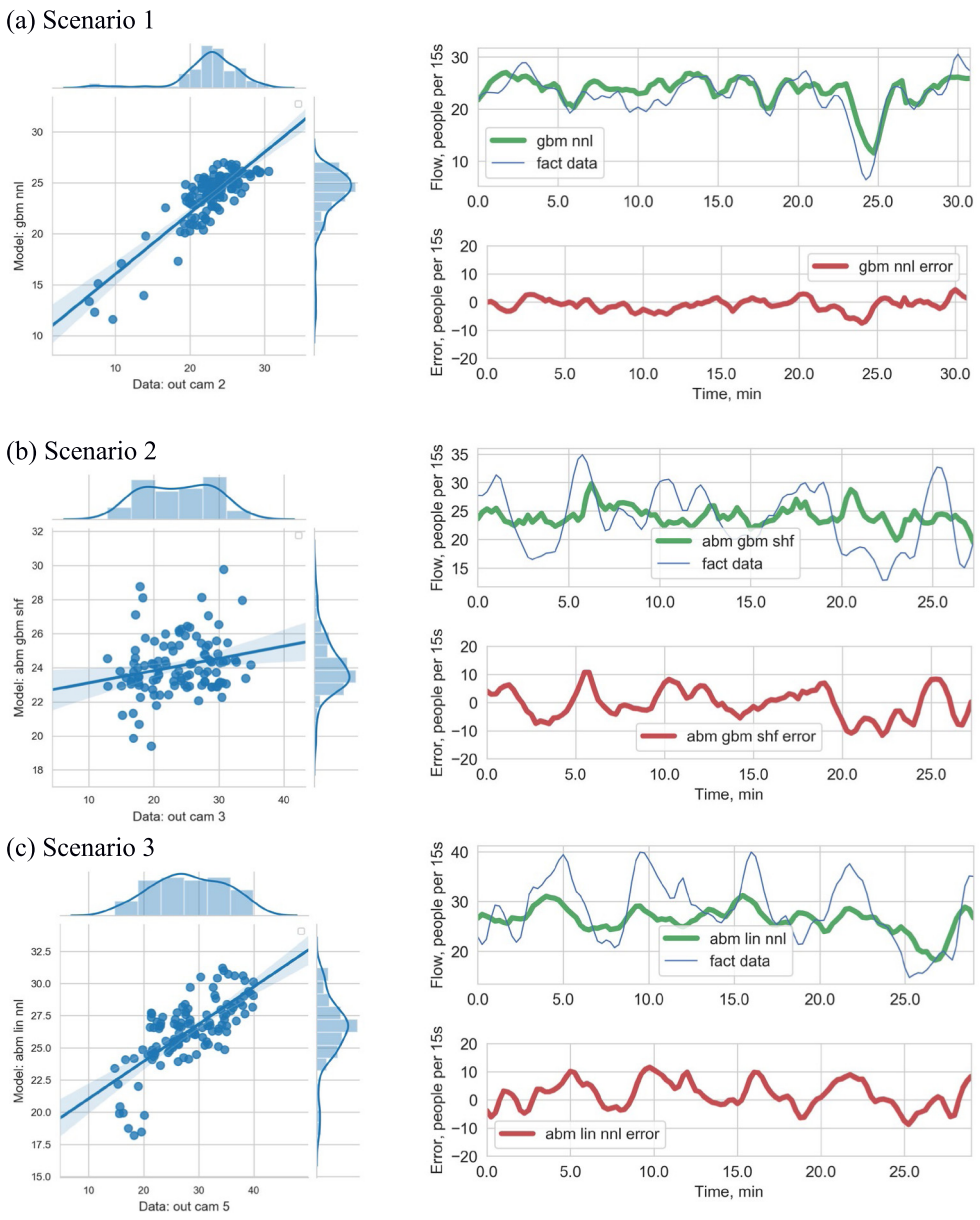


Fig. 13. Best predictions for: (a) Scenario 1; (b) Scenario 2; (c) Scenario 3. Here we show the Pearson correlation coefficient and the error. Aliases specified on the legend are as follows: 'abm' — agent based model, 'nnl' — long short term memory network, 'nnd' — dense neural network, 'gbm — gradient boost method, 'svm' — support vector machine, 'lin' — linear regression, 'shft' — shifted flow, 'fact' — observed data, 'dummy' — dummy regression.

The results of the best-performing models are presented in Fig. 13. Recall that Scenario 1 involves walking from Camera 01 to Camera 02. Scenario 2 is from Camera 02 to Camera 03, which is a relatively longer path compared to the other paths. Scenario 3 incorporates a merging of flows coming from Cameras 03 and 04 and the subsequent movement of the merged flow to Camera 05. Our results in Fig. 13 show that the ensemble of **gradient boost method** and **long short-term memory network** in Scenario 1 as well as the ensemble of **agent-based model, linear regression**, and **long short term memory network** in Scenario 3 exhibit high correlation. The lowest prediction error corresponds to Scenario 1. This is not surprising because Scenario 1 is a relatively simple corridor compared to the corridors in Scenarios 2 and 3. Scenarios 2 and 3 exhibit higher errors in terms of prediction. However, the results of the ensemble model in Scenario 3 follows closely the trend of the data as compared to that in Scenario 2.

Interestingly, the **agent-based model** is a part of the ensemble that best predicts complex scenarios. We assume this is because the agent-based model has the capability of incorporating an explicit model of the environment into its algorithm, so that it can better deal with more complex corridor, thereby making it possible to create better predictions in more complicated scenarios. On the other hand, this type of algorithm strongly depends on the complexity of the scenarios (see Table 1 for complexity factors), such as police control of crowd flow that is done manually (Scenario 2), and instances when pilgrims randomly stop to pray (Scenario 3). This type of behaviour was not included in the model.

More interesting results can be observed with the **ensemble models** (see Fig. 14). First off, by just looking at how the individual models perform, we show that gradient boost method exhibits the lowest error in Scenario 1, support vector regression in Scenario 2, and agent-based modelling in Scenario 3. Shift model, on the other hand performs the worst among all models especially in Scenarios 1 and 3. This is no

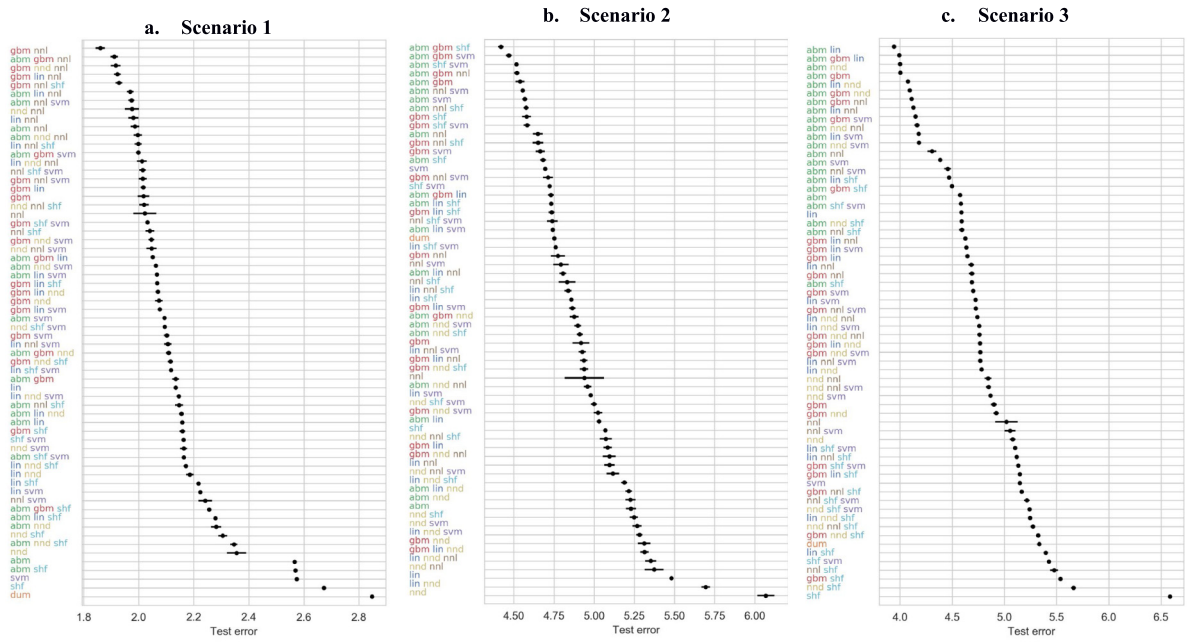


Fig. 14. Error Simulation results for scenarios: (a) Scenario 1; (b) Scenario 2; (c) Scenario 3. Aliases specified on the legend are as follows: 'abm' — agent based, 'nnl' — long short term memory network, 'nnd' — dense neural network, 'gbm' — gradient boost method, 'svm' — support vector machine, 'lin' — linear regression, 'shft' — shifted flow, 'fact' — observed data, 'dummy' — dummy regression.

surprise as the shift model naively assumes that the rate of movement of the crowd is uniform in all areas of the temple.

Interestingly, combining the top 2 best performing individual models in Scenarios 1 (gradient boost and neural network long-short memory) and 3 (agent based modelling and linear modelling) improved the predictive power of the ensemble model. It is also apparent that combinations of models that incorporate agent-based modelling have better predicative power than all the other combinations in Scenarios 2 and 3. Notice that these combinations of models with the agent-based model occupy the upper half of the plot in Scenarios 2 and 3. Our results suggest that agent-based modelling, especially when combined with other models, provides better predictive power especially in complex scenarios such as those of 2 and 3. Recall that Scenario 2 is a long corridor that incorporates several corners. Scenario 3, on the other hand, involves merging of flows. One more advantage of agent-based modelling over the machine learning methods is its versatility in incorporating physical details in the simulation environment. Obstacles, may, for instance be added to the complexities of the scenarios. See Karbovskii et al. (2019) for a detailed study on the impact of obstacles on crowd movement.

Finally, we observe that the worst-performing model is the dense neural network (fully connected). Even when it is combined with other models, it performs poorly even as a component of an ensemble in Scenario 2. However, its performance is better when combined with stronger models in Scenarios 1 and 2. In general, the predictive power is improved when better-performing models are coupled with weaker models via ensemble modelling than better-performing models alone. This is because ensemble models average out biases and reduce the variance, and at the same time reduce the likelihood of overfitting.

One should always be wary of overfitting. We look especially into this aspect by computing for the difference between the training metric and the test metric. Hence, higher values for overfit also imply greater extent of overfitting. See Fig. 15.

Our results show that gradient boosting appear to be consistently the most overfitted across all three scenarios. Interestingly, although agent-based modelling alone shows overfitting in all three scenarios, incorporating agent-based modelling in any of the combinations of models show less overfitting in Scenario 3.

The best combination for the ensemble seems to be scenario-dependent. That is, ensemble models performing under specific scenarios gave the best results. This implies that different combinations of models could better describe different scenarios. Although it seems clear at this point that ensemble models indeed provide better results in predicting crowd dynamics, understanding how and more importantly why certain combinations of models work as they are, calls for a rigorous and systematic investigation through setting up more experiments. Due to the limitation in our data, we can only provide speculations as to why certain combinations of models perform well like they do in specific scenarios.

In order to study the emerging synergy in more detail, we plotted the errors for the best-performing ensembles, their respective components, and the dummy regression against the ground truth. See Figs. 15 and 16. We show that applying ensemble learning increases the quality of prediction and reduces the error.

Two things play a major role when combining models: (1) complexity of the scenario and (2) quality of the models. On the one hand, agent-based models for flow prediction are limited to specific locations such as the entry and exit points of crowd movement. Although the models we used, such as the gradient boost method, long-short term memory network, dense neural network, support vector regression, linear regression, and time shift model give good results for simple scenarios and moderate results for complex scenarios, agent-based model, on the other hand, shows stable results for the complex scenarios but its error is relatively high on its own. Unlike other models, the agent-based model can in principle consider more complex factors, including police control, pilgrim behaviour such as praying etc. We emphasize that modelling these detailed processes is beyond the scope of the current research. Moreover, agent-based models can give spatial predictions, such as fields of flow and density, which opens up new avenues for analysis. More importantly, the use of multiple models could significantly increase the quality of prediction.

5. Discussion & conclusion

Major events and gatherings as massive as the Kumbh Mela festival pose risks to crowd safety that should be addressed efficiently. Ensuring

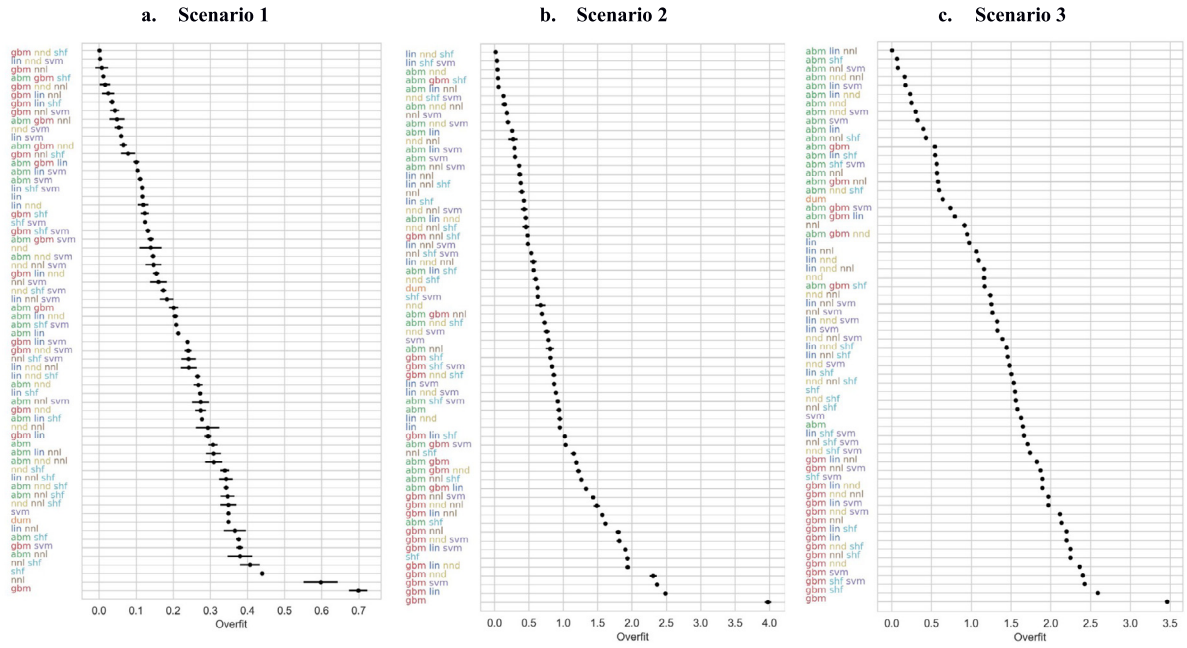


Fig. 15. Overfit Simulation results for scenarios: (a) Scenario 1; (b) Scenario 2; (c) Scenario 3. Aliases specified on the legend are as follows: 'abm' — agent based, 'nnl' — long short term memory network, 'nnd' — dense neural network, 'svm' — support vector machine, 'lin' — linear regression, 'shft' — shifted flow, 'fact' — observed data, 'dummy' — dummy regression.

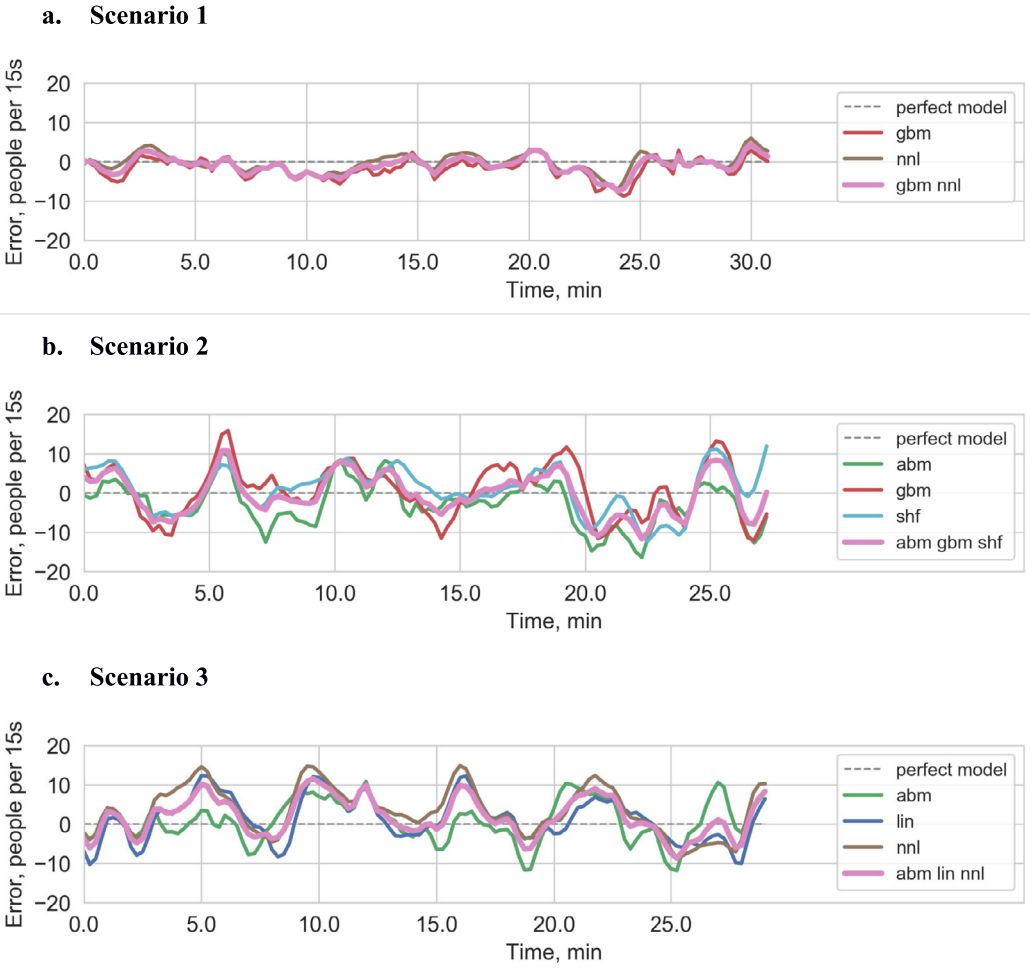


Fig. 16. Simulation errors for best ensembles and their components for: (a) Scenario 1; (b) Scenario 2; (c) Scenario 3.

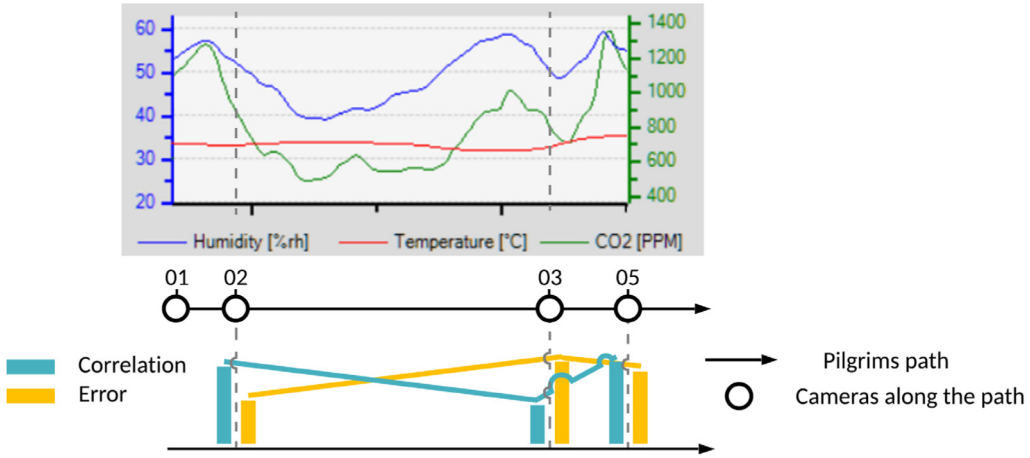


Fig. 17. Overlay of graphs for: (a) humidity, temperature and CO2 measurements inside the temple; (b) change of crowd flow prediction correlation and error between measure points 01–08 (stationary cameras). Grey vertical lines represent cameras' positions.

safety in these types of events would mean looking closely into crowd control and more importantly understanding how the crowd behaves. There are numerous existing methods and models that predict crowd behaviour, where each model has its own defined limitations.

We introduce a novel method for short-term crowd flow prediction and showed its advantages as well as disadvantages in comparison to separate methods. We show that methods hinged on disparate philosophies (agent-based modelling tries to understand how a system works, while machine learning algorithms learn to search general laws on empirical data) have different predictive power for different scenarios. Utilizing a unique data set that we collected during the Kumbh Mela festival, we use different models from a wide area of artificial intelligence to show how proper combinations of such a diverse set of models could provide synergy to obtain better prediction. We increase the quality of prediction by 13%, depending on the complexity of the scenario and emphasize that agent-based model combined with other models provided better predictive power in complex scenarios. We anticipate that our research would be a starting point for further investigation of model synergy and predicting crowd flow. We hope that our research would inspire other researchers to look into our model as well as encourage stakeholders to share data sets that could be useful to further develop the concepts we have presented in our work.

Additionally, different environmental conditions such as ambient factors including air temperature, humidity, availability of space, number of people in confined environments (Wells et al., 2016), can influence movement, interpersonal behaviour, and the decision making of people. Having acquired the results for the estimation of accuracy of crowd flow and density predictions, we decided to draft them in terms of other characteristics referring to space or the environment: humidity, temperature, and level of CO2. We summarize our findings in Fig. 17.

The chart shows that CO2 and humidity curves tend to have similar profiles. Their values peak and match between Cameras 01 and 02, which can be attributed to the congestion near the metal detectors. Between Cameras 02 and 03, which is a longer path, CO2 follows the shape of humidity. Here, error and correlation values change inversely at almost the same ratio and then almost doubles between the 03–05 interval. Given the measured prediction accuracy and ambient factors, we hypothesize that these might be correlated. However, since it is hard to justify a direct connection between the described variables, we assume that environmental factors may be linked to the prediction accuracy with a mediator variable. At this point, due to the limitation in data, it is difficult to conclude if indeed high levels in humidity and CO2 would cause disturbance in the crowd, significant enough to increase the error in our predictions.

We will look into how these environmental factors coupled with the dynamics of the crowd can affect prediction accuracy in our future work. Of additional interest is the development of models capable of reproducing the relationship between characteristics of environment and dynamics of the crowd.

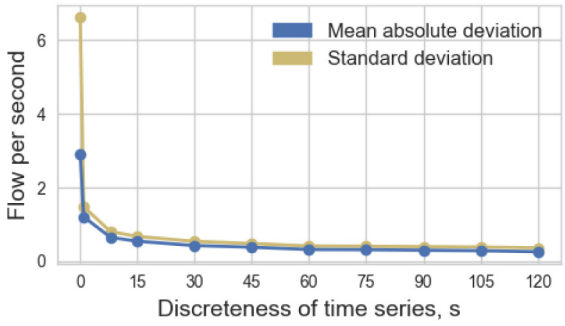


Fig. 18. Discreetness plot.

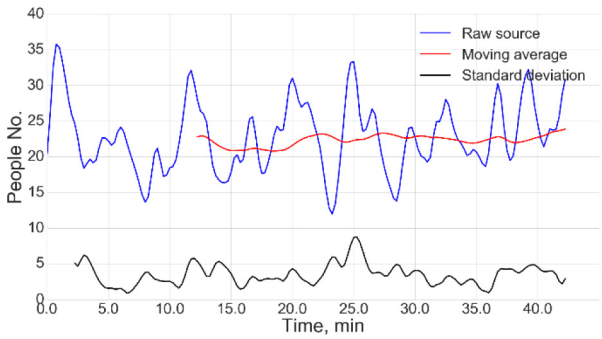


Fig. 19. Stationarity test.

Exploring the advantage of using more advanced agent-based models could be an interesting direction to look into. Since this after all is a spatial model, we could then closely look into field predictions and utilize flow fields and incorporate more detailed behaviours. Data assimilation techniques to correct field predictions can also be useful for various cases, e.g. optimization (Butakov et al., 2015). Finally, looking more in detail into informational synergy of the models coming from different philosophies, as well as multiple synthetic cases on synthetic data could help us and future researchers to better understand the predictability of the human crowd. This overall could lead to increasing the understanding of crowd behaviour and hopefully improve safety in mass gatherings.

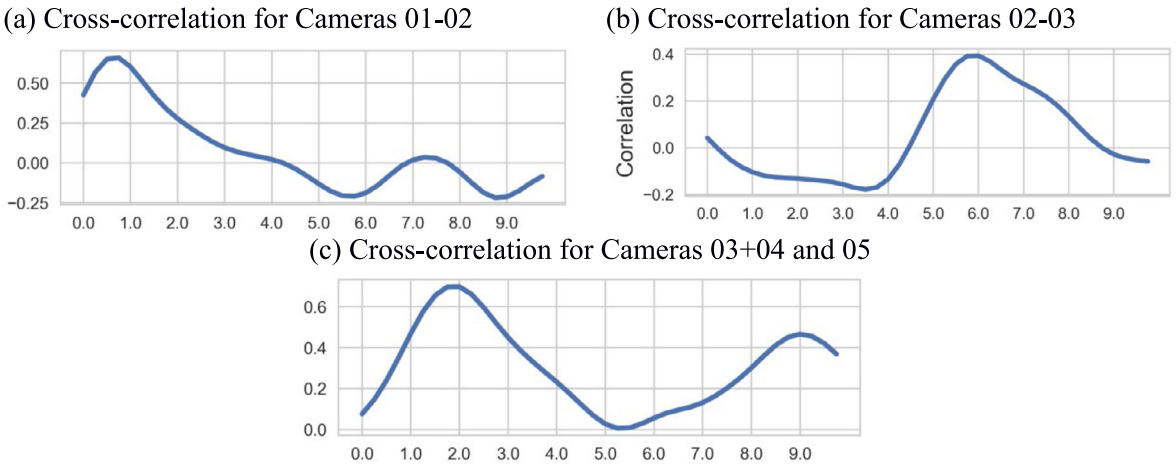


Fig. 20. Cross correlation functions for cameras pairs (full time series).

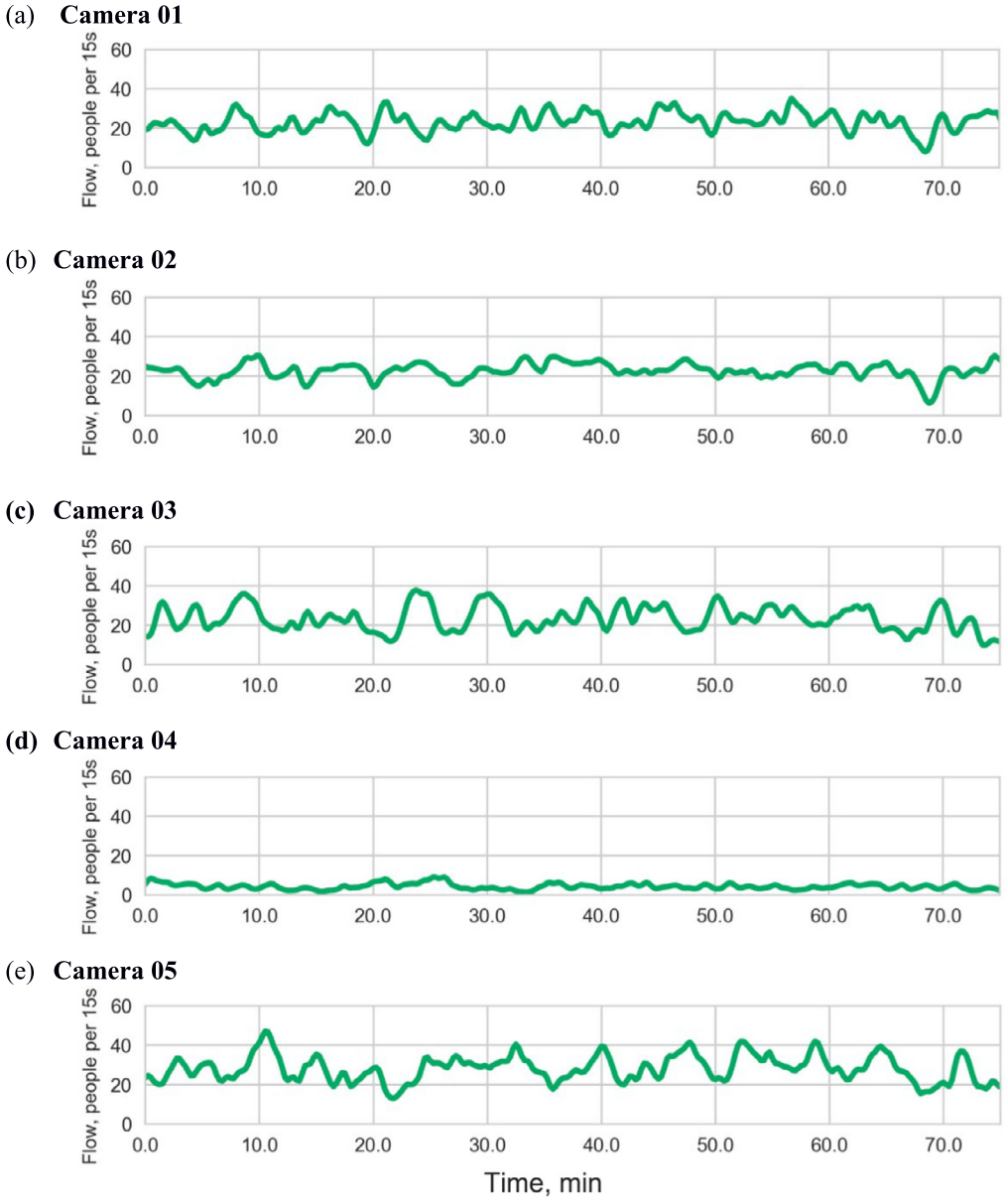


Fig. 21. Discrete normalized & filtered time series for all cameras.

CRediT authorship contribution statement

Vladislav Karbovskii: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – original draft, Data curation, Validation, Supervision. **Michael Lees:** Investigation, Conceptualization, Writing – original draft, Writing – review & editing, Funding acquisition. **Alva Presbitero:** Methodology, Writing – original draft, Writing – review & editing. **Alexey Kurilkin:** Software. **Daniil Voloshin:** Writing – original draft, Investigation. **Ivan Derevitskii:** Software. **Andrey Karsakov:** Visualization, Investigation. **Peter M.A. Sloot:** Project administration, Funding acquisition, Supervision, Investigation.

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.

Acknowledgements

This research is financially supported by the NWO Kumbh Mela project under grant agreement 629.002.202. This paper is also financially supported by Ministry of Education and Science of the Russian Federation, Agreement #14.584.21.0015 (11.11.2015). We would also like to thank the team of Ashish Verma and the local municipality of Ujjain, India.

Appendix A. Discreetness of time series

Our choice of discreetness is based on Jianhong and Xiaohong (2011), where mean absolute deviation and standard deviation for time series for different discreetness. The discreetness plot specifically for our data set is presented in Fig. 18. We choose the discreetness as 15 s, since it shows stability. To check the normality of distribution, we used the Shapiro–Wilk test as it gives favourable results with a score of at least 0.95 for each time series.

Appendix B. Stationarity test

Even though the flow of pilgrims changes within the day, this can be considered stationary in our data set, since the standard deviation and rolling mean are also stationary (Fig. 19). In addition, we performed Dickey–Fuller test to check the stationarity hypothesis. The test results to a p-value < 0.05, hence, we rejected the null hypothesis which assumes that the flow is non-stationary in our data. In fact, the process is quasi-stationary. This means that the process is stationary at a specific point of observation, but not stationary in general (i.e. pilgrim flow is not static in general, could change in the evening, at lunch, etc.).

Appendix C. Cross-correlation analysis

See Fig. 20.

Appendix D. Prepared time series for all cameras

See Fig. 21.

References

Andriluka, M., Roth, S., Schiele, B., 2008. People-tracking-by-detection and people-detection-by-tracking. In: 26th IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 1–8. http://dx.doi.org/10.1109/CVPR.2008.4587583.

Baranwal, A., Anand, A., Singh, R., Deka, M., Paul, A., Borgohain, S., Roy, N., 2015. Managing the earth's biggest mass gathering event and wash conditions: Maha kumbh mela (India). PLoS Curr. 7, http://dx.doi.org/10.1371/currents.dis.e8b3053f40e774e7e3fdbe1bb50a130d.

Bratsun, D., Dubova, I., Krylova, M., Lyushnin, A., 2013. Computational modeling of collective behavior of panicked crowd escaping multi-floor branched building. In: Springer Proceedings in Complexity. pp. 659–663. http://dx.doi.org/10.1007/978-3-319-00395-5_80.

Breitenstein, M.D., Reichlin, F., Leibe, B., Koller-Meier, E., Van Gool, L., 2011. Online multiperson tracking-by-detection from a single, uncalibrated camera. IEEE Trans. Pattern Anal. Mach. Intell. 33, 1820–1833. http://dx.doi.org/10.1109/TPAMI.2010.232.

Butakov, N., Nasonov, D., Knyazkov, K., Karbovskii, V., Chuprova, Y., 2015. The multi-agent simulation-based framework for optimization of detectors layout in public crowded places. In: Procedia Computer Science. pp. 522–531. http://dx.doi.org/10.1016/j.procs.2015.05.278.

Cariappa, M.P., Singh, B.P., Mahen, A., Bansal, A.S., 2015. Kumbh mela 2013: Healthcare for the millions. Med. J. Armed Forces India 71, 278–281. http://dx.doi.org/10.1016/j.mjafi.2014.08.001.

Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. In: Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2005, pp. 886–893. http://dx.doi.org/10.1109/CVPR.2005.177.

DeFilippi, R.R.F., 2018. Boosting, Bagging, and Stacking — Ensemble Methods with sklearn and mlens [WWW Document]. M. https://medium.com/@rrfd/boosting-bagging-and-stacking-ensemble-methods-with-sklearn-and-mlens-a455c0c982de. (Accessed 21 March 2019).

Dietterich, T.G., 2000. Ensemble methods in machine learning. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 1–15. http://dx.doi.org/10.1007/3-540-45014-9_1.

Dollár, P., Appel, R., Belongie, S., Perona, P., 2014. Fast feature pyramids for object detection. IEEE Trans. Pattern Anal. Mach. Intell. 36, 1532–1545.

Dollar, P., Wojek, C., Schiele, B., Perona, P., 2012. Pedestrian detection: An evaluation of the state of the art. IEEE Trans. Pattern Anal. Mach. Intell. 34, 743–761.

Duives, D.C., Daamen, W., Hoogendoorn, S.P., 2014. State-of-the-art crowd motion simulation models. Transp. Res. Part C Emerg. Technol. 37, 193–209. http://dx.doi.org/10.1016/j.trc.2013.02.005.

Duives, D.C., Wang, G., Kim, J., 2019. Forecasting pedestrian movements using recurrent neural networks: An application of crowd monitoring data. Sensors (Switzerland) 19, 382. http://dx.doi.org/10.3390/s19020382.

Tuning the hyper-parameters: Exhaustive Grid Search [WWW Document], URL https://scikit-learn.org/stable/modules/grid_search.html.

Friedman, J.H., 2002. Stochastic gradient boosting. Comput. Stat. Data Anal. 38, 367–378.

Gao, C., Li, P., Zhang, Y., Liu, J., Wang, L., 2016. People counting based on head detection combining adaboost and CNN in crowded surveillance environment. Neurocomputing 208, 108–116. http://dx.doi.org/10.1016/j.neucom.2016.01.097.

Greenough, P.G., 2013. The kumbh mela stampede: disaster preparedness must bridge jurisdictions. BMJ 346. http://dx.doi.org/10.1136/bmj.f3254.

Hastie, T., Tibshirani, R., Friedman, J., 2009. Ensemble learning. In: RN. pp. 1–20. http://dx.doi.org/10.1007/b94608_16.

Heikkila, J., Silven, O., 1997. Four-step camera calibration procedure with implicit image correction. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 1106–1112. http://dx.doi.org/10.1109/cvpr.1997.609468.

Helbing, D., Johansson, A., Al-Abideen, H.Z., 2007. Dynamics of crowd disasters: An empirical study. Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys. 75, 46109. http://dx.doi.org/10.1103/PhysRevE.75.046109.

Helbing, D., Molnar, P., 1995. Social force model for pedestrian dynamics. Phys. Rev. E 51, 4282.

Hinton, G.E., 1990. Connectionist learning procedures. In: Machine Learning. Elsevier, pp. 555–610.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9, 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Jianhong, Y., Xiaohong, C., 2011. Optimal measurement interval for pedestrian traffic flow modeling. J. Transp. Eng. 137, 934–943.

Karbovskii, V., Severiukhina, O., Derevitskii, I., Voloshin, D., Presbitero, A., Lees, M., 2019. The impact of different obstacles on crowd dynamics. J. Comput. Sci. 36, 100893. http://dx.doi.org/10.1016/j.jocs.2018.06.010.

Karbovskii, V., Voloshin, D., Karsakov, A., Bezgodov, A., Gershenson, C., 2018. Multimodel agent-based simulation environment for mass-gatherings and pedestrian dynamics. Futur. Gener. Comput. Syst. 79, 155–165. http://dx.doi.org/10.1016/j.future.2016.10.002.

Karbovskii, V., Voloshin, D., Karsakov, A., Bezgodov, A., Zagarskikh, A., 2015. Multi-scale agent-based simulation in Large City Areas: Emergency evacuation use case. Procedia Comput. Sci. 51, 2367–2376. http://dx.doi.org/10.1016/j.procs.2015.05.407.

Khanna, T., Macomber, J., Chaturvedi, S., 2013. Kumbh Mela: India's Pop-up Mega-City.

Kiselev, A.V., Karbovskii, V.A., Kovalchuk, S.V., 2016. Agent-based modelling using ensemble approach with spatial and temporal composition. In: Procedia Computer Science. pp. 530–541. http://dx.doi.org/10.1016/j.procs.2016.05.333.

Mehta, D., Yadav, D.S., Mehta, N.K., 2014. A literature review on management of mega event-maha kumbh (simhastha). Int. J. Res. Sci. Innov. 1, 45–49.

Mordvintsev, A.S., Krzhizhanovskaya, V.V., Lees, M.H., Sloot, P.M.A., 2014. Simulation of city evacuation coupled to flood dynamics. In: Pedestrian and Evacuation Dynamics 2012. Springer, pp. 485–499.

Musse, S.R., Thalmann, D., 2001. Hierarchical model for real time simulation of virtual human crowds. IEEE Trans. Vis. Comput. Graph 7, 152–164. http://dx.doi.org/10.1109/2945.928167.

Opitz, D., Maclin, R., 1999. Popular ensemble methods: An empirical study. J. Artif. Intell. Res. 11, 169–198. http://dx.doi.org/10.1613/jair.614.

Patil, S., Van Den Berg, J., Curtis, S., Lin, M.C., Manocha, D., 2011. Directing crowd simulations using navigation fields. IEEE Trans. Vis. Comput. Graph 17, 244–254. http://dx.doi.org/10.1109/TVCG.2010.33.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michael, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É, 2011. Scikit-learn: Machine learning in python. J. Mach. Learn. Res. 12, 2825–2830.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michael, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É, n.d, 2019. Ensemble methods — scikit-learn documentation [WWW Document]. URL https://scikit-learn.org/stable/modules/ensemble.html. (Accessed 19 March 2019).

Pelechano, N., Allbeck, J.M., Badler, N.I., 2007. Controlling individual agents in high-density crowd simulation, in: Symposium on Computer Animation 2007 - ACM SIGGRAPH / Eurographics Symposium Proceedings, SCA 2007, pp. 99–108.

Smith, B.L., Demetsky, M.J., 1994. Short-term traffic flow prediction models-a comparison of neural network and nonparametric regression approaches. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics. IEEE, 1706–1709. http://dx.doi.org/10.1109/ICSMC.1994.400094.

Smola, A.J., Schölkopf, B., 2004. A tutorial on support vector regression. Stat. Comput. 14, 199–222. http://dx.doi.org/10.1023/B:STCO.0000035301.49549.88.

Soomaroo, L., Murray, V., 2012. Disasters at mass gatherings: Lessons from history. PLoS Curr. 1–7. http://dx.doi.org/10.1371/currents.RRN1301.

Sridhar, S., Gautret, P., Brouqui, P., 2015. A comprehensive review of the Kumbh Mela: identifying risks for spread of infectious diseases. Clin. Microbiol. Infect. 21, 128–133.

Turris, S.A., Lund, A., Bowles, R.R., 2014. An analysis of mass casualty incidents in the setting of mass gatherings and special events. Disaster Med. Public Health Prep. 8, 143–149.

Viswanathan, V., Lee, C.E., Lees, M.H., Cheong, S.A., Sloot, P.M.A., 2014. Quantitative comparison between crowd models for evacuation planning and evaluation. Eur. Phys. J. B 87, 27. http://dx.doi.org/10.1140/epjb/e2014-40699-x.

Vlahogianni, E.I., Karlaftis, M.G., Golias, J.C., 2005. Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach. Transp. Res. Part C Emerg. Technol. 13, 211–234. http://dx.doi.org/10.1016/j.trc.2005.04.007.

Voloshin, D., Rybokonenko, D., Karbovskii, V., 2015. Towards a performance-realism compromise in the development of the pedestrian navigation model. Procedia Comput. Sci. 51, 2799–2803. http://dx.doi.org/10.1016/j.procs.2015.05.437.

Wells, N.M., Evans, G.W., Cheek, K.A., 2016. Environmental psychology. Environ. Heal. from Glob. To Local 203.

Wetherill, G.B., Seber, G.A.F., 1977. Linear regression analysis. J. R. Stat. Soc. Ser. A 140, 546. http://dx.doi.org/10.2307/2345290.

World Health Organization, 2015. Public health for mass gatherings: Key considerations. World Heal. Organ. 82–94.

Zhang, Z., 2000. A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. 22, 1330–1334. http://dx.doi.org/10.1109/34.888718.

Zhao, H., Thrash, T., Kapadia, M., Wolff, K., Holscher, C., Helbing, D., Schinazi, V.R., 2020. Assessing crowd management strategies for the 2010 love parade disaster using computer simulations and virtual reality. J. R. Soc. Interface 17, http://dx.doi.org/10.1098/rsif.2020.0116.

Zheng, W., Lee, D.-H., Shi, Q., 2006. Short-term freeway traffic flow prediction: Bayesian combined neural network approach. J. Transp. Eng. 132, 114–121. http://dx.doi.org/10.1061/(ASCE)0733-947X(2006)132:2(114).

Zhong, J., Cai, W., Lees, M., Luo, L., 2017. Automatic model construction for the behavior of human crowds. Appl. Soft Comput. J. 56, 368–378. http://dx.doi.org/10.1016/j.asoc.2017.03.020.

Zhong, J., Hu, N., Cai, W., Lees, M., Luo, L., 2015. Density-based evolutionary framework for crowd model calibration. J. Comput. Sci. 6, 11–22. http://dx.doi.org/10.1016/j.jocs.2014.09.002.

Zhong, J., Luo, L., Cai, W., Lees, M., 2014. Automatic rule identification for agent-based crowd models through gene expression programming, in: 13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2014, pp. 1125–1132.