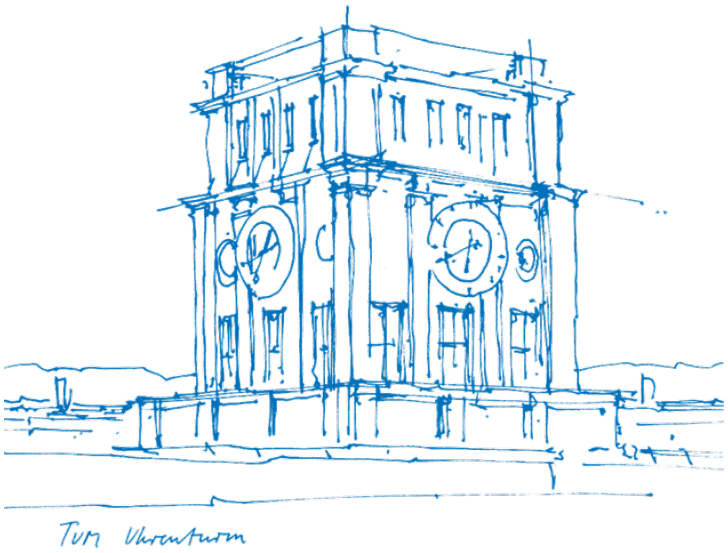
Chair of Scientific Computing
TUM School of Computation, Information and Technology
Technical University of Munich

Master's thesis in Computational Science and Engineering

Parameterized Koopman operator models for crowd dynamics

Koichiro Hida



Master's thesis in Computational Science and Engineering

Parameterized Koopman operator models for crowd dynamics

Koichiro Hida

Chair of Scientific Computing
TUM School of Computation, Information and Technology
Technical University of Munich

Master's thesis in Computational Science and Engineering

Parameterized Koopman operator models for crowd dynamics

Koichiro Hida

Thesis for the attainment of the academic degree

Master of Science (M.Sc.)

at the TUM School of Computation, Information and Technology of the Technical University of Munich.

Examiner:
Prof. Dr. Felix Dietrich

Submitted:
Munich, 30.04.2024

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Munich, 30.04.2024 Koichiro Hida

Abstract

There has been growing interest in crowd dynamics modeling, driven primarily by its relevance in various fields, including urban planning, safety measures, and event management. The central objective within this field is to understand how people behave collectively due to the interactions, movements, and responses to environmental factors. The recent progress in machine learning also impacts crowd modeling because it can utilize vast amounts of data from sensors, digital devices, and simulations. Data-driven methods have demonstrated the ability to achieve flexible and precise dynamics predictions, even without knowing the governing equation of the systems.

This thesis focuses on crowd dynamics modeling with the Koopman operator, an infinite-dimensional linear operator that acts on functions defined on the state space of a dynamical system. The primary objective is parameterizing the Koopman operator to accommodate variable geometries and conditions in crowd dynamics prediction. The Koopman operator offers insights into system dynamics through eigen-components via extended dynamic mode decomposition, which approximates the Koopman operator with a linear combination of eigenfunctions, eigenvalues, and modes in infinite dimension.

The data on crowd dynamics is sampled using Vadere, an open-source crowd simulation software. Through approximation of the Koopman operator, the essential components of the dynamic system are identified and represented by eigenfunctions. It enabled successful predictions of crowd behavior. Further investigation of the eigen-components of the Koopman operator is conducted to characterize coherent structures in dynamic systems. The Koopman eigenfunctions reveal wave-like data intrinsic patterns and parameter-dependent trends in crowd behaviors.

Contents

1 Introduction

The study of crowd dynamics modeling and forecasting is crucial in urban planning, safety measures, and event coordination. This field primarily aims to understand the collective behavior of individuals within large gatherings, examining how crowds navigate, interact, and respond to external stimuli like obstacles, emergencies, or changes in their environment [1, 2, 3].

Numerous modeling approaches for crowd behavior exist, where researchers try to capture the underlying principles of crowds. Several researchers tried to describe the crowd behavior on a micro-scale by modeling the pedestrian's behaviors [4]. However, it still requires bridging the gap between the local behaviors of pedestrians during interactions and the broader dynamics of the crowds. Moreover, external factors such as weather conditions, natural disasters, and construction activities present additional challenges to equation-based modeling approaches. Consequently, computer simulation has emerged as a prevalent tool for predicting crowd dynamics, offering flexibility in handling complex scenarios and manipulating various factors as parameters [5]. However, a significant drawback of computer simulation is that as scenarios grow in complexity and scale, the simulation requires considerable time to complete. Moreover, simulations need to be repeated numerous times to account for variable parameter combinations, with the addition of new parameters increasing the number of simulations exponentially. Recent advancements in artificial intelligence and machine learning greatly influence crowd dynamics modeling by leveraging vast amounts of data from sensors, digital devices, and simulations. Data-driven methods have shown the potential to make more accurate predictions without relying on traditional equation-based approaches [6, 7]. One of the significant challenges in data-driven methods is that the model tends to be a black box, and the internal process of making predictions is unclear. The Koopman operator stands out for its ability to allow the analysis of the model, offering insights into the dynamics through the eigen-components. Utilizing operator approximation methods such as Extended Dynamic Mode Decomposition (EDMD), researchers have successfully employed the Koopman operator to model dynamical systems even with nonlinear dynamical patterns.

Hence, this thesis selects the Koopman operator model for investigating crowd dynamics. The objective is to construct the approximated model of the Koopman operator to provide predictions of the crowd dynamics. To apply the Koopman operator in crowd dynamics modeling, it needs to consider that the operator should be capable of representing variable conditions, such as geometries and the size of crowds. In other words, the Koopman operator models should be dependent on some parameters. EDMD is a data-driven method requiring only a dataset to approximate the Koopman operator model. The parameter information should also be included within the dataset to take it into account in the model. The method employed in this thesis is to increase the number of features in the dataset as many as the number of parameters. The dataset is extended by adding a parameter as a column of constant value, facilitating the operator's prediction depending on the parameter value.

Without the approximation of the Koopman operator, the practical application of the operator in crowd dynamics is challenging. One major obstacle is that the operator can be infinite-dimensional, which makes the direct computation of the operator unfeasible. Additionally, no efficient computational method is available to calculate the Koopman operator directly. The emergence of approximation methods made a significant breakthrough in the practical application of the Koopman operator. Approximation methods offer finite dimensional approximations of the Koopman operator, making it feasible to apply to real-world problems. EDMD, an extension of the Dynamic Mode Decomposition method, is one of the methods used to approximate the Koopman operator, which produces Koopman eigenfunctions, modes, and associated eigenvalues [8]. These components offer a structured linear representation of nonlinear dynamics, establishing an intrinsic coordinate system where the system operates within the function space. Additionally, Koopman eigenfunctions are linked to geometric system properties and coherent structures. EDMD offers

greater flexibility in approximating the Koopman operator, thus attracting attention among the approximation methods in recent studies.

In this thesis, Vadere, an open-source software designed for crowd dynamics simulation [9], is employed to acquire data on crowd dynamics. Vadere offers a platform to simulate crowd models in two-dimensional geometry. Section 2 explains the theoretical foundations of the Koopman operator and its approximation methods in detail. Additionally, the recent progress in crowd dynamics modeling is summarized. Section 3 presents the main project, applying the parameterized Koopman operator model for crowd dynamics. The section is further divided into the background, methodology, results evaluation, and model analysis. Finally, this thesis is summarized in the conclusion.

2 State of the art

2.1 Koopman operator

The Koopman operator was first introduced by Bernard Koopman in 1931 [10]. Koopman revealed that there is a linear operator associated with each dynamical system. In this paper, he explored the foundations of Hamiltonian mechanics and investigated systems that exhibit a steady n-dimensional flow of a fluid of positive density. He used the Hilbert spaces as spaces of observables to define the operator for the Hamiltonian systems. A Hilbert space is a mathematical construct used to describe infinite-dimensional vector spaces. This leads to a concept of linear transformations in Hilbert's space, which maps one state to another while preserving inner products and norms. The Koopman operator is a linear operator that acts on state observables [11]. It maps an observable, a function of the systems's state, at one time to the same observable at a later time.

A continuous dynamical system can be defined as

$$\frac{dx}{dt} = f(x). \tag{2.1}$$

The system continuously updates state variable x by a continuous function f. A discrete-time update can also express this

$$\begin{aligned} F_t(x(t_0)) &= x(t_0 + t) \\ &= x(t_0) + \int_{t_0}^{t} f(x_t)dt, \end{aligned} \tag{2.2}$$

where F_t is a function at time t. Here, x can be any state space, such as a manifold. The Koopman operator K acts on the observables g of state X

$$K_t g = g \circ F_t, \tag{2.3}$$

where $\circ$ denotes the composition of the observable function g and function F. In many cases, employing a Hilbert space F of functions is useful as the observables the Koopman operator acts on. Throughout the thesis, we will assume this is the case. The Koopman operator updates the observable of the current state x_k one step ahead and gives the observable x_{k+1}. Therefore, it is a linear way to advance observable functions in time. In other words, the Koopman operator defines a new dynamical system that governs the evolution of observables in discrete time. Even when function F is nonlinear, the Koopman operator K is linear. The observables are usually in infinite dimensions; hence, solving an infinite dimensional linear system of the equation is required to compute the Koopman operator directly. Therefore, it requires an approximation of the Koopman operator in practice, using several methods described in sections 2.2 and 2.3.

Figure 2.1 shows a diagram of how the Koopman operator works. The Koopman operator provides another way to evolve a state over time. The dynamical system defined by the function F and the one defined by the Koopman operator K are two different parameterizations, but they represent the same fundamental behavior of the dynamics. The links between the two expressions are the observable function $g(x) = x$. Eigendecomposition of the Koopman operator provides another form of the formulation de-

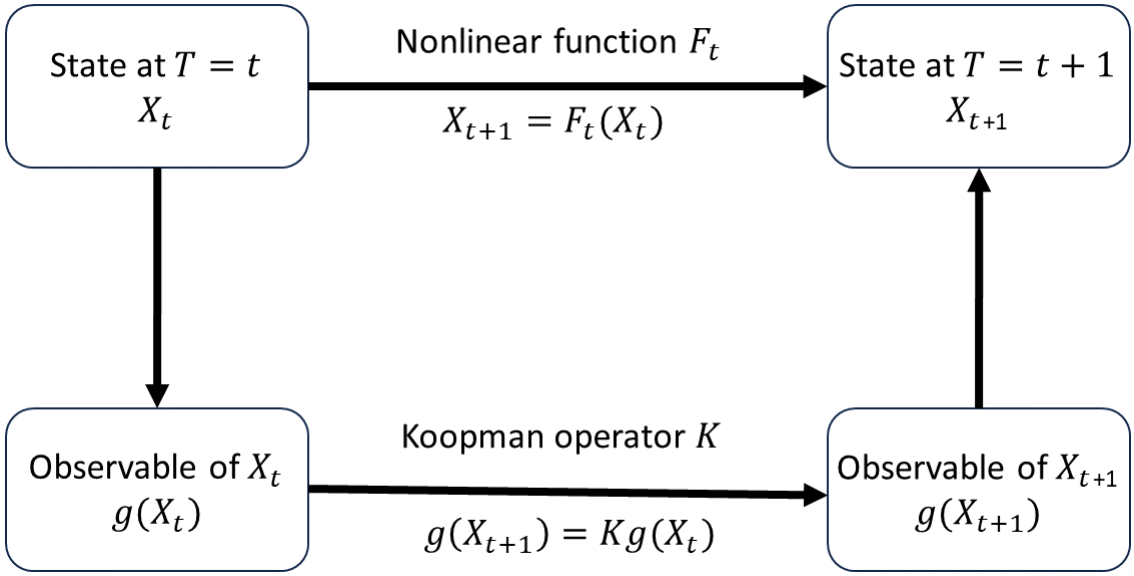


Figure 2.1 Flow chart of time series prediction with Koopman operator. A nonlinear function is substituted by the infinite-dimensional linear function, the Koopman operator, by representing the states with the observable functions and updating the state as observable functions. The updated state is obtained by inversing the mapping from observable to the state.

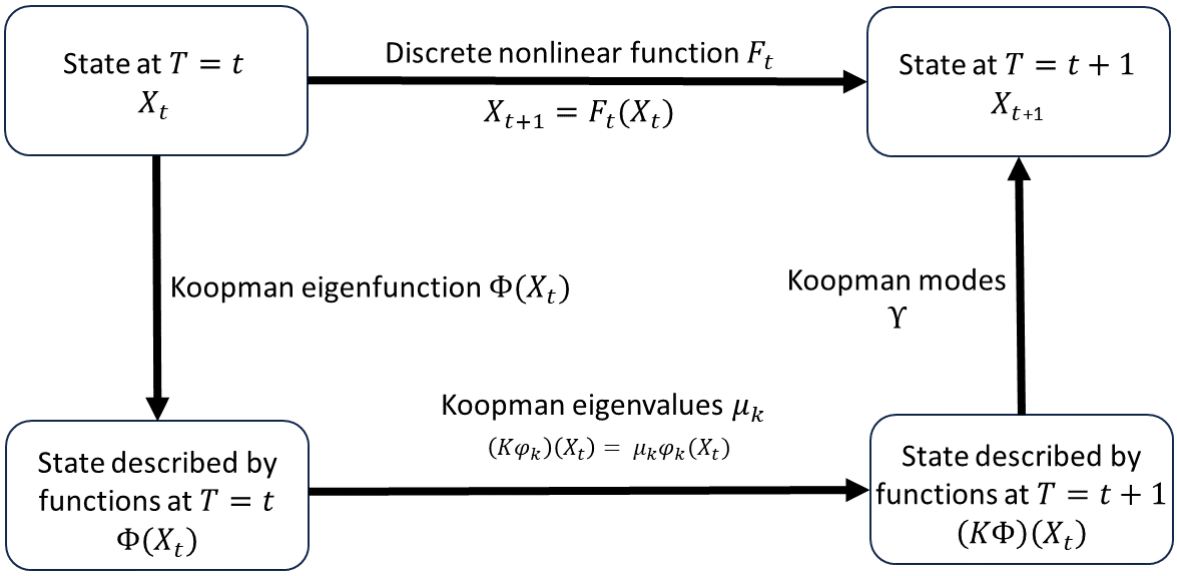


Figure 2.2 Flow chart of time series prediction with the Koopman operator's eigenfunctions and eigenvalues. Koopman eigenfunction embeds the states, and eigenvalues update the state over time. Koopman mode is a vector that transforms the embedded space back to the original state space.

fined by $(\mu_k, \phi_k, v_k)_{k=1}^K$, the Koopman eigenvalue, eigenfunction, and mode. The formulation of the time evolution by the Koopman operator can be written as

$$\begin{aligned} F(x) &= (Kg)(x) \\ &= \sum_{k=1}^{K} v_k (K\phi_k)(x) \\ &= \sum_{k=1}^{K} \mu_k v_k \phi_k(x). \end{aligned} \tag{2.4}$$

Figure 2.2 shows the roles of the three components. The Koopman eigenfunction represents the state X based on the eigenfunctions on which the eigenvalues are applied to update the state. The Koopman modes restore the state from the eigenfunction basis. In this formulation, the links of the two expressions are the Koopman eigenfunctions and the modes. Indeed, the full state observable $g = [g_1, g_2, ...]$ can be obtained by multiplying the two components

$$g_i = \sum_{k=1}^{K} v_{ik} \phi_k. \tag{2.5}$$

Consequently, state variable x can also be obtained as

$$x = g(x) = \sum_{k=1}^{K} v_k \phi_k(x), \tag{2.6}$$

where v_k is the k-th Koopman mode and ϕ_k is the k-th Koopman eigenfunction.

Such representation based on eigen-components enables analysis of the operator because the eigenfunctions preserve system characteristics, and the eigenvalues determine the significance of corresponding eigenfunctions.

Due to the linearity of the Koopman operator, numerical approximations are simplified, which is later introduced as extended dynamic mode decomposition (EDMD). The linearity also provides the operator with interpretability and explainability. The spectral components of the operator can analyze the Koopman operator as "building blocks." It enables better interpretation of the dynamics than other methods, such as simulations and deep learning-based methods. The approximation of the Koopman operator can be done entirely using a data-driven approach. Hence, it does not require a governing equation of the state.

The introduction of approximation methods such as Dynamic Mode Decomposition (DMD) provided a breakthrough for the practical application of the Koopman operator. Those approximation methods offer finite-dimensional approximations of the Koopman operator, enabling its application to real-world problems. In section 2.2, DMD, one of the significant methods for Koopman operator approximation, is introduced. Also, the extension of the DMD (EDMD) is explained in section 2.3.

Koopman operator is applied to various fields due to its ability to linearize nonlinear systems and reveal critical geometric properties. The Koopman operator has been applied in various areas of control theory, including observability, controllability analysis, stability, model predictive control, optimal control, feedback stabilization, and identification. Especially the introduction of the approximation of the Koopman operator promoted research in these fields [12]. Ian Abraham et al. explored the application of the Koopman operator to control theory in the field of robotics [13]. He used the Koopman operator to generate data-driven models with utility for model-based control. The Koopman operator improved the controller performance in various terrains. In physics, the Koopman operator is often called a compositional operator and is frequently used, especially in dynamic systems. Stefan Klus et al. investigated the application of the Koopman operator to Quantum systems [14]. He compares the ground-state transformation and Nelson's stochastic mechanics, demonstrating how data-driven methods developed for the approximation of the Koopman operator can be used to analyze quantum physics problems. Yoshihiko Susuki et al. explored the Koopman operator theory for power systems technology [15]. The applications include coherency identification of swings in coupled synchronous generators, precursor diagnostic of instabilities in the coupled

swing dynamics, and stability assessment of power systems without using mathematical models. These applications demonstrate the potential of the Koopman operator theory in addressing complex problems in power systems technology.

2.2 Dynamic Mode Decomposition

Dynamic Mode Decomposition (DMD) is a data-driven technique to obtain linear reduced order models for high dimensional complex systems [16]. DMD can also extract spatial-temporal coherent structures or patterns that dominate the observed measurement data from that dynamical system. It is a purely data-driven method; therefore, it does not require an underlying equation for the system. DMD was initially introduced in fluid dynamics and later connected to the nonlinear dynamical systems theory by Peter Schmidt [17]. DMD is later applied to the approximation of the Koopman operator by Igor Mezic et al. [12].

Given measured data $X = [x_1, x_2, ...]$, where each column is a feature or spatial information of the state. $X' = [x_{n+1}, x_{n+2}, ...]$ is a matrix of n time step ahead of X. Now assume an operator A which satisfy

$$X' = AX. \tag{2.7}$$

In such a case, an operator A can be derived as

$$A = X'X^{-1}. \tag{2.8}$$

Operator A evolves X to X'. However, the calculation of A is often computationally too expensive as the number of features or spatial information increases. Therefore, A is approximated with the smaller dimensional operator. Such an approximation can be realized by approximating the leading eigenvalue-eigenvector pairs of A without actually calculating A. In other words, DMD tries to approximate the dominant eigenvalues and corresponding eigenvectors of matrix A. Once the leading eigenvectors are obtained, these eigenvectors can be reshaped into the original state space. Also, these eigenvectors show coherent time-varying dynamics such as sine and cosine waves, exponential decay, or growth. The eigenvalues allow the prediction of the evolution of the dynamics.

The first step of DMD computes the Singular Value Decomposition (SVD) of the matrix X. Assuming that X is a $M \times N$ matrix, SVD is a factorization of the matrix into three matrices

$$X = UDV^T, \tag{2.9}$$

where U is $M \times M$ matrix containing the orthonormal eigenvectors of XX^T, V^T is transpose of a $N \times N$ matrix of the orthonormal eigenvectors of X^T. D is a matrix with r elements equal to the root of the positive eigenvalues of XX^T or X^TX. After applying SVD to the data matrix X, the variance of X is organized from the most significant to the least significant. Then X in 2.7 is substituted and obtain

$$X' = AUDV^T. \tag{2.10}$$

Then, in order to isolate matrix A, matrix V, D^{-1}, and complex conjugate U^* are multiplied from the right hand side. SVD is advantageous in inverting V because it is a unitary matrix.

$$\begin{aligned} U^*X'VD^{-1} &= U^*AU \\ &= A'. \end{aligned} \tag{2.11}$$

Here, a large matrix A is projected onto the dominant singular vectors U^* and U and a smaller matrix A' is obtained. Matrix A' has the same non-zero eigenvalues as matrix A. Therefore, the eigenvalues of A can be computed from the eigendecomposition of A'

$$A'W = W\Lambda. \tag{2.12}$$

W contains the eigenvectors of A' and Λ contains the eigenvalues of A'. Finally, the eigenvectors of A are computed as follows

$$\Phi = X'VD^{-1}W. \tag{2.13}$$

Φ is the approximated eigenvectors of A. Φ is also called DMD mode because it shows the spatial correlations between measurements. If the initial condition of X is given as X_0, the future state can be predicted by

$$X'(k\delta t) = \Phi\Lambda^t X_0. \tag{2.14}$$

Using equation 2.14, the future state of X can be predicted from the DMD mode and corresponding eigenvalues. The Λ term in the equation is powered by the time t. Hence, the DMD modes of small eigenvalues are diminished in the long-term prediction. Therefore, the analysis of the DMD modes is quite helpful in understanding the major components that affect the prediction. Figure 2.3 shows an example of DMD modes. This DMD mode is derived from a simple time series of sine functions with randomly generated noise, with 200-time steps for the computation of the fitting DMD model. The DMD modes are sorted in the descending order of their corresponding eigenvalues. In other words, the DMD modes are sorted by the significance of the components. The heatmap visualization shows that each component of DMD modes has a different frequency of repeating patterns, and the significant components have less frequency. The first component represents the major component of the function, the sine function, and the other components represent the noise terms. The DMD model comprehends the dynamics as a composition of factors with different behaviors and then represents the factors using the DMD modes.

In many applications, DMD is used as a numerical approximation of the Koopman operator, which gives Koopman modes and corresponding eigenvalues. As in the definition of DMD, data matrices X and Y are defined. As long as X and Y are linearly consistent, the relationship between DMD and Koopman modes can be defined as follows. Suppose that DMD modes are computed from X and Y and thus obtain eigenvectors and eigenvalues that satisfy

$$A\phi_j = \lambda_j\phi_j, \tag{2.15}$$

where $A = YX^+$. $+$ denotes a pseudoinverse. If the matrix A has a full set of eigenvectors, then each column x_k of X can be expanded as

$$h(z_k) = x_k = \sum_{j=0}^{n-1} c_{jk}\phi_j \tag{2.16}$$

for some constants c_{jk}. For linearly consistent data, each column of data matrices is updated by linear operator A, thus $Ax_k = y_k$, therefore,

$$\begin{aligned} h(f(z_k)) = y_k &= Ax_k \\ &= \sum_{j=0}^{n-1} c_{jk}A\phi_j \\ &= \sum_{j=0}^{n-1} \lambda_j c_{jk}\phi_j. \end{aligned} \tag{2.17}$$

Recalling that the Koopman operator can be expressed as

$$h(f(z)) = \sum_{j=0}^{n-1} \lambda_j\theta_j(z)\phi_j, \tag{2.18}$$

where $\theta(z)$ is the eigenfunction of the Koopman operator, it is seen that in the case of linearly consistent data matrices, the DMD modes correspond to Koopman modes and the DMD eigenvalues to the Koopman eigenvalues. Constants c_jk are given as Koopman eigenfunction $\theta_j(z_k)$ in this case. If the data arise from a sequential time series, the modes can be scaled to subsume the constant c_jk.

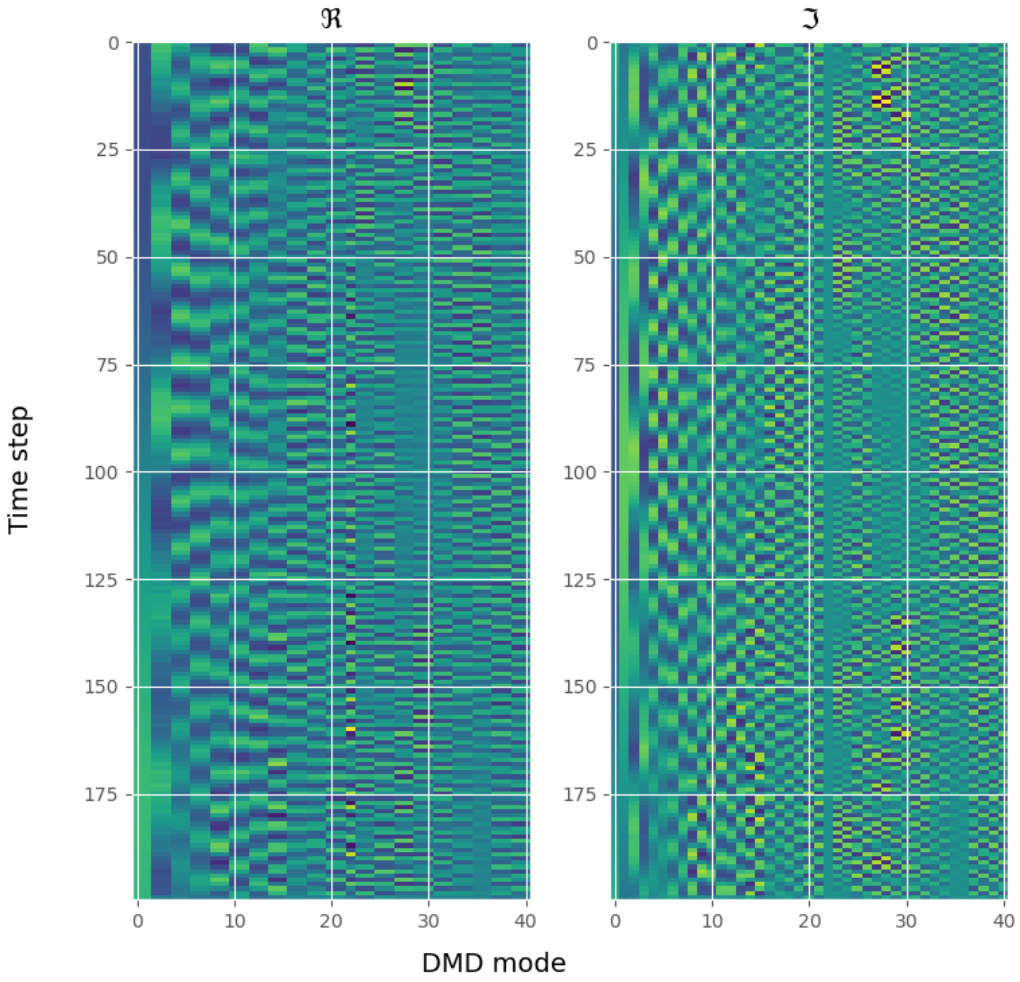


Figure 2.3 Visualization of the DMD modes. The DMD model is derived from data of sine function with randomly generated noise terms. The model is fitted with 200 time steps. The figure on the left side shows the DMD modes in real values, and the figure on the right side shows the imaginary values of DMD modes. DMD modes are sorted in the descending order of corresponding eigenvalues.

DMD has been applied to various fields, especially in the fluid dynamics. P. J. Schmid applied DMD to Schlieren snapshots of a helium jet and to time-resolved Particle Image Velocimetry measurements of an unforced and harmonically forced jet [18]. DMD allowed the breakdown of a fluid process into dynamically relevant and coherent structures. It aided in the characterization and quantification of physical mechanisms in fluid flow.

DMD approximates the Koopman operator with a best-fit linear model. However, DMD is based on linear measurements, which do not span a Koopman invariant subspace for many nonlinear systems. Also, DMD has a limited ability to represent the Koopman operator as it is stated in 2.18, that the DMD approximates the Koopman operator in a particular case where the Koopman eigenfunctions can be represented with constant values.

2.3 Extended Dynamic Mode Decomposition

Extended Dynamic Mode Decomposition (EDMD) is a method that approximates the Koopman operator and, therefore, the Koopman eigenvalue, eigenfunction, and mode based on the choice of a dictionary. [8]. Koopman eigenvalue is the eigenvalue of Koopman operator, and Koopman eigenfunctions are corresponding eigenfunctions of the Koopman operator. Koopman eigenfunctions provide a principled linear embedding of nonlinear dynamics, resulting in an intrinsic coordinate system in which the system is closed under the action of the Koopman operator. Koopman eigenfunctions can also be associated with geometric system properties and coherent structures [19]. Koopman mode is a vector that enables the reconstruction of the system's state as a linear combination of the Koopman eigenfunctions.

Several other algorithms can computationally approximate subsets of Koopman eigenfunctions, eigenvalues, and modes.

- DMD
  As discussed in the previous section, DMD is a method that has been successfully used to analyze nonlinear systems by approximating Koopman modes and corresponding eigenvalues [12].
- Generalized Laplace Analysis (GLA)
  GLA can approximate both the Koopman modes and eigenfunctions. It has been used to extract meaningful spatio-temporal structures from sensor data [20].
- Ulam Galerkin Method
  Ulam Galerkin Method approximates the eigenfunctions and eigenvalues. It is used to identify coherent structures and almost invariant sets based on the singular value decomposition of the Koopman operator [8].

EDMD is developed to approximate all the components: Koopman eigenfunctions, eigenvalues, and modes from a data set of successive snapshot pairs and a dictionary of observables that spans a subspace of the scalar observables

EDMD is introduced by O. Williams [8]. It is an extension of DMD explained in section 2.2.

Given a pair of time series data from simulation or measurement of the dynamical system $X = [x_1, x_2, ...]$ and $X' = [x_{n+1}, x_{n+2}, ...]$ of the system state, where X' is n time ahead of X. It is written as

$$X' = F(X), \tag{2.19}$$

where F is a nonlinear flow. Dictionary of observables $g = \psi_1, \psi_2, ..., \psi$ spans in the finite dimensional function space. Vector-valued observable functions can also be defined as

$$\Psi = [\psi_1(x)\ \psi_2(x)\ ...\ \psi_N(x)]. \tag{2.20}$$

The finite-dimensional approximation of the Koopman operator, K, is computed based on the selection of the dictionary. By definition, a function f can be written as

$$f = \sum_{l=1}^{L} a_l \psi_l = \Psi \mathbf{a}, \tag{2.21}$$

where L is the number of elements in the dictionary with weights $\mathbf{a}$. Since f is typically not an invariant subspace of the Koopman operator $\mathcal{K}$,

$$\mathcal{K}f = (\Psi \odot F)\mathbf{a} = \Psi(K\mathbf{a}) + r, \tag{2.22}$$

K is the approximated Koopman operator in finite dimension, and r is the residual term. To determine the approximation of the operator K, the following minimization problem

$$\begin{aligned} J &= \frac{1}{2}\sum_{m=1}^{M} |r(x_m)|^2 \\ &= \frac{1}{2}\sum_{m=1}^{M} |((\Psi \odot F)(x_m) - \Psi(x_m)K)\mathbf{a}|^2 \\ &= \frac{1}{2}\sum_{m=1}^{M} |(\Psi(x'_m) - \Psi(x_m)K)\mathbf{a}|^2 \end{aligned} \tag{2.23}$$

must be solved. Equation 2.23 is a least square problem. The solution yields K, a finite-dimensional approximation of $\mathcal{K}$, that maps f to some other f' by minimizing the residuals. As a result, the K that minimizes the above is

$$K = G^{+}C, \tag{2.24}$$

where G^{+} is a pseudoinverse of

$$G = \frac{1}{M}\sum_{m=1}^{M} \Psi(x_m)\Psi(x_m), \tag{2.25}$$

and

$$C = \frac{1}{M}\sum_{m=1}^{M} \Psi(x_m)\Psi(y_m). \tag{2.26}$$

The approximation of the Koopman eigenfunctions ϕ_j can be derived from the obtained approximation of the Koopman operator K. Assume ν_j is j-th eigenvector of K, the approximation of an eigenfunction is

$$\phi_j = \Psi\nu_j. \tag{2.27}$$

In the next step, the approximation of the Koopman modes Υ is computed. Full state observable g can be written as

$$g_i = \sum_{n=1}^{N} \psi_n b_{n,i}, \tag{2.28}$$

where b_i is some appropriate vector of weights. The entire vector-valued observable can be expressed

$$g = B^T\Psi^T = (\psi B)^T, \tag{2.29}$$

where $B = [b_1, b_2, ..., b_N]$. From the equation 2.27, observable Ψ can be expressed in terms of Koopman eigenfunction Φ as

$$\Psi = \Phi V, \tag{2.30}$$

where $V = [\nu_1, \nu_2, ..., \nu_k]$. ν_i is the i-th eigenvector of K associated with eigenvalue μ_i. Therefore, observable ψ_i can be written as a function of ϕ_i by inverting V_t. Its inverse is

$$V^{-1} = W = [w_1, w_2, ..., w_k], \tag{2.31}$$

where w_i is the i-th left eigenvector of K also associated with μ_i. From equation 2.29 and 2.27, full state observable can be expressed using Koopman modes Υ

$$g = \Upsilon\Phi^T = \sum_{m=1}^{M} \upsilon_m\phi_m, \tag{2.32}$$

where the Koopman modes $\Upsilon = [\upsilon_1, \upsilon_2, ..., \upsilon_m]$.

While DMD approximates Koopman modes and eigenvalues but not Koopman eigenfunctions, EDMD additionally produces a better approximation of Koopman eigenfunctions. They produce the same set of Koopman modes and eigenvalues for any set of snapshot pairs; however, they are only equivalent to specific and restricted dictionary selection. From equation 2.19, the linear operator of DMD K_{DMD} can be derived as

$$K_{DMD} = X'X^{+}. \tag{2.33}$$

The DMD modes are defined as the eigenvectors of the matrix K_{DMD}, where the j-th mode is associated with the j-th eigenvalue of K_{DMD}, μ_j. Here, $+$ denotes the pseudoinverse. The Koopman modes computed using EDMD with a particular choice of dictionary $\mathcal{D} = e_1^*, e_2^*, ..., e_N^*$ are equivalent to the DMD modes. Recall that the full state observable of the Koopman operator $g(x) = x$ is a vector-valued observable that can be expressed as

$$\begin{aligned} g(x) &= [g_1(x), g_2(x), ..., g_N(x)] \\ &= [e_1^* x, e_2^* x, ..., e_N^* x], \end{aligned} \tag{2.34}$$

where e_i is the i-th unit vector in $\mathbb{R}^N$. All g_i is assumed to be $g_i \in F_D$. Because the elements of the full state observable are the dictionary elements, $B = I$. The Koopman modes are expressed as

$$\Upsilon = (W^* B)^T. \tag{2.35}$$

Since $B = I$, the Koopman modes are the complex conjugates of the left eigenvectors of K. Hence, $\upsilon_i^T = w_i^*$. Furthermore, $G^T = \frac{1}{M} X X^*$ and $A^T = \frac{1}{M} X' X^*$. Then

$$\begin{aligned} K^T &= A^T G^{T+} \\ &= Y X^* (X X^*)^+ \\ &= Y X^+ \\ &= K_{DMD}. \end{aligned} \tag{2.36}$$

It reveals that DMD and EDMD approximations are identical regarding a specific dictionary choice and if the pseudo inverse is the matrix inverse. Moreover,

$$\begin{aligned} K_{DMD} \upsilon_i &= (\upsilon_i^T K_{DMD}^T)^T \\ &= (w_i^* K)^T \\ &= (\mu_i w_i^*)^T \\ &= \mu_i \upsilon_i. \end{aligned} \tag{2.37}$$

Hence, all the Koopman modes computed by EDMD are eigenvectors of K_{DMD} and, thus, the DMD modes. DMD can be thought of as producing the approximation of Koopman eigenfunctions with a set of linear monomials as basis functions. However, most of the systems are nonlinear. EDMD is an extension of DMD that retains additional terms in the expansion, where the elements of D determine these additional terms [8].

EDMD offers more flexibility in approximating the Koopman operator than DMD. EDMD improved upon the classical DMD by including a flexible choice of dictionary of observables that spans a finite-dimensional subspace on which the Koopman operator can be approximated. However, selecting a proper dictionary to approximate the Koopman operator remains challenging. This is particularly challenging for high-dimensional and highly nonlinear systems. Qianxiao Li et al. proposed an iterative approximation algorithm that couples the EDMD with a trainable dictionary represented by an artificial neural network to address this issue [21]. This approach allowed the dictionary to adapt to the problem at hand, achieving good reconstruction accuracy without choosing a fixed dictionary a priori. Furthermore, Hiroaki Terao et al. suggested a method to find a dictionary using neural ordinary differential equations (NODEs) [22]. NODEs is a neural network equipped with a continuum of layers with high parameter and memory efficiencies. He proposed an algorithm to perform EDMD using NODEs and showed that it finds a parameter-efficient dictionary that provides an excellent finite-dimensional approximation of the Koopman operator.

2.4 Crowd dynamics modeling and prediction

Crowd dynamics modeling and prediction have gained attention due to their relevance in various domains, including urban planning, safety, and event management. The main objectives of this field are to understand the collective behavior of individuals in large groups and how the crowds move, interact, and respond to external factors such as obstacles, emergencies, or changes in the environment.

Preceded research tried to understand the underlying principles of crowd behavior from governing equations. Mehdi Moussaïd et al. attempted to uncover the principles based on two modeling methods: outcome models and process models [4]. On the one hand, outcome models describe the movements of a pedestrian through repulsive forces or probabilities to move from one place to another. On the other hand, process models generate movement from the bottom up by describing the underlying cognitive process pedestrians use during navigation. However, the authors emphasized that the crowd of pedestrians is a complex system exhibiting a variety of self-organized collective behaviors, and it requires linking the local behavior of pedestrians during interactions and the global dynamics of the crowd at high density to understand the dynamics. Also, external factors need to be considered reality, such as weather, natural disasters, and constructions, which makes the equation-based modeling approach challenging.

Computer simulation is also frequently used for predicting crowd dynamics due to its reproducibility of the results, extensivity of the scenarios to be more complex and realistic, and availability of controlling and manipulating various factors as variables. R. A. Saeed et al. presented a simulation methodology for pedestrian crowd movement in large and complex environments [5]. It uses a multi-group microscopic model that tracks all individual pedestrians rather than aggregating the behaviors of crowds. Hence, it accepts the characteristics of pedestrians, such as gender, age, position, velocity, and energy, which can generate more realistic trajectories of pedestrians. While computer simulation is a powerful tool for studying crowd dynamics, it suffers from the required computational resources and time for simulation as there are more variables to be considered in the scenarios. Simulations must be repeatedly carried out to obtain different conditions, which is time-consuming.

Artificial intelligence and machine learning advancements have opened up new possibilities for crowd dynamics modeling. Much data is accessible from sensors, digital devices, and simulations. Data-driven methods can leverage large amounts of data to make more accurate predictions. Furthermore, traditional equation-based modeling often requires very strong assumptions to hold, which limits the complexity and variability that the model can handle. Meanwhile, data-driven methods do not require the underlying equations; they can learn the dynamics from data independently. Sample data about crowd flows are generally high-dimensional. Hence, this approach also has to deal with the increasing computational complexity. Zhai Zhongyi et al. presented a novel method for predicting citywide crowd flows using deep learning techniques [6]. He addressed the challenge by introducing a data reconstruction mechanism to reduce the dimension of sample data. The mechanism also includes a spatiotemporal data-oriented prediction model constructed based on a bottleneck residual network, which effectively reduces the computational complexity of model training while improving the prediction accuracy of crowd flows. Marco Cardia et al. introduced a novel solution, CrowdNet, for predicting crowd flow based on graph convolutional networks [7]. One of the limitations most deep learning approaches face is that they cannot explain their predictions sufficiently. CrowdNet can be used with regions of irregular shapes and provide meaningful explanations of the predicted crowd flows. However, the explainability and interpretability of the predictions are still challenges that should be improved because data-driven methods tend to be a black box as they lack the governing equation of the dynamic system. The models are complex, so it is difficult to root the cause.

The Koopman operator owns a great advantage as it offers the "building blocks" like structure of the model to be analyzed. In other words, the Koopman operator is described as a set of eigenfunctions, eigenvalues, and modes, which are associated with one component of the linear evolution of the observables. Hence, the Koopman operator enables the dynamics analysis by eigenpair components. Lehmberg used the Koopman operator to model the crowd dynamics on a city scale. He collected the measurement data of the number of pedestrians across Melbourne as shown in Figure 2.4, which are influenced by non-recurrent factors such as city layout, weather, and accidents by sensors. The Koopman operator is

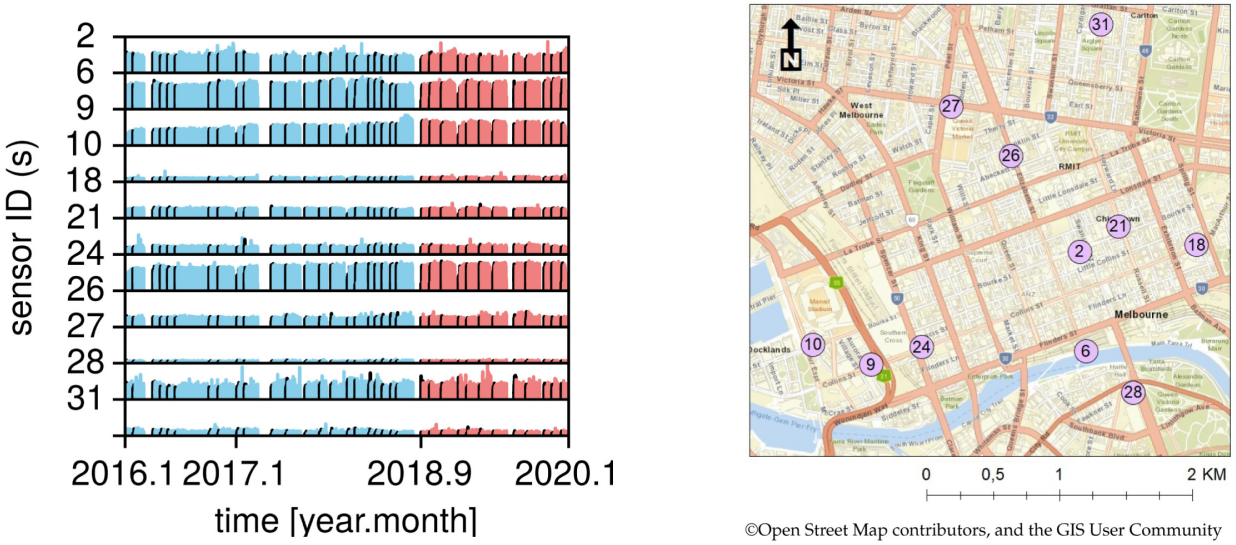


Figure 2.4 Overview of the sensor data and locations in the city map taken from [23]. Left: Schematic overview of data selection of 11 sensors in four years. A single traffic state consists of the scalar measurements of all sensors and describes the Melbourne pedestrian traffic state. Right: Map of Melbourne, Australia, with corresponding sensor locations.

approximated using Extended Dynamic Mode Decomposition (EDMD) [8] with a dictionary composed of two methods: Taken's time-delay embedding and diffusion maps. It showed that the trained model predicts the crowd behavior quite well even though the data contains nonlinear behavior. He emphasized the advantage of the Koopman operator by analyzing the eigenfunctions of the Koopman operator and the diffusion maps to understand how the operator interpreted the crowd dynamics. It showed repeating patterns representing the intrinsic patterns of the geometrically dominant direction of the data-inferred manifold [23].

3 Parameterized Koopman operator models for crowd dynamics

3.1 Background and objectives

Crowd simulation is one of the most commonly used methods for studying crowd dynamics since it provides the flexibility of the geometry and model parameters such as the number of pedestrians in the crowds and walking speed. With the increasing computational ability and resources, crowd simulation has gained popularity in this field. However, crowd simulation has significant drawbacks. Each simulation takes a long time until it terminates. The time required grows even larger as the scenario becomes more complex and on a larger scale. On top of that, the simulations have to be repeated as many times as the number of combinations of the parameters. If a new parameter is added to a scenario, the number of simulations increases exponentially. This makes the crowd simulation unpractical for scenarios with a variety of parameters. Crowd simulation has especially attracted attention in the field of city planning. The behavior of the crowds is crucial to determining the geometry and structures of a city to make it without over-crowding people or causing traffic jams. Hence, methodologies to simulate crowd behavior in a city-like geometry are of great interest in this field. A city involves many parameters: the width of streets, the size of buildings, the number of people on the streets, and the structures of the buildings. It requires simulations with the combinations of these parameters to investigate each case. This process costs a lot of time, so using crowd simulation in such a case is impractical.

Therefore, instead of using crowd simulations to investigate crowd dynamics, the Koopman operator model is selected to model the behavior of the crowds in this thesis. In order to accommodate various conditions and geometries that affect crowd dynamics, the Koopman operator is parameterized. The figure 3.1 shows the diagram, describing the parameterized Koopman operator. The parameterized Koopman operator should be able to model the fundamental dynamics of the crowd and provide accurate predictions of the crowd's behavior depending on parameters such as the width of a path and the number of pedestrians.

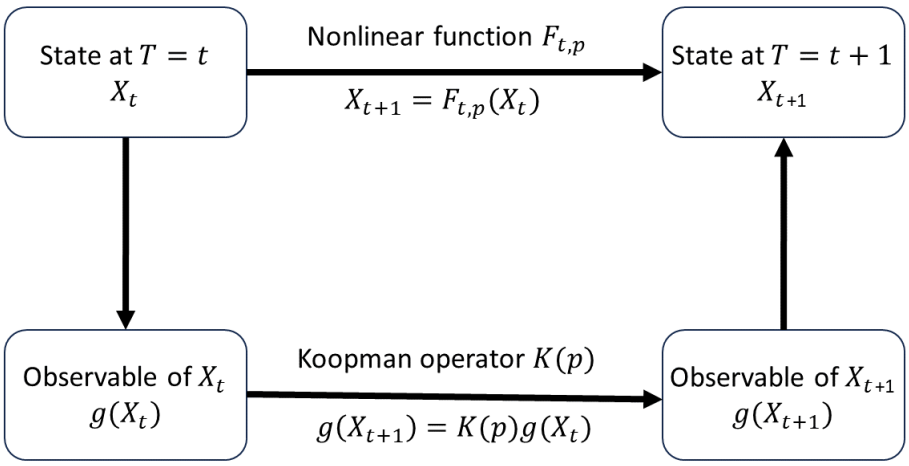


Figure 3.1 A diagram of a parameterized Koopman operator. The evolution of the state is based on a parameter p.

Time	Value 0	Value 1	Value 2
T_0	V_0^0	V_0^1	V_0^2
T_1	V_1^0	V_1^1	V_1^2
T_2	V_2^0	V_2^1	V_2^2
T_3	V_3^0	V_3^1	V_3^2
...	...	...	...

Time	Value 0	Parameter 0	Value 1	Value 2
T_0	V_0^0	P^0	V_0^1	V_0^2
T_1	V_1^0	P^0	V_1^1	V_1^2
T_2	V_2^0	P^0	V_2^1	V_2^2
T_3	V_3^0	P^0	V_3^1	V_3^2
...	...	...	...	...

Figure 3.2 Visualization of an example data matrix. The data matrix above is a sampled time-series data with three feature values. Assuming that the value of feature 0 is dependent on a parameter, the parameter value is added to a new column. The parameter value can be the width of a street and the number of population in a district. Hence, it is a constant value over the time series. Therefore, the added parameter column contains a constant parameter value.

3.2 Methodology

In the first part, this section proposes the methodology for parameterizing the Koopman operator. Then, the workflow is presented, from the preparation of the dataset to the prediction of future data.

3.2.1 Parameterization of Koopman operator

In this section, the method to parameterize the Koopman operator is presented. The basic idea is that the parameter value becomes a new column of constant value, where the value is the parameter value, and the column is added to the data matrix. Assume a data matrix X of a time-series data. The columns of the data matrix represent the features. Suppose that a feature value is dependent on a parameter value. The data matrix can include the parameter value as a new column in the data matrix. Figure 3.2 shows an example of parameterization of the data matrix. The original data matrix is a time series of several feature values. If the feature 0 depends on a parameter, the parameter value can be added as a new column, as shown in the figure. A constant value is only for a time series, meaning that if there are multiple time series in the data matrix, the column should contain multiple constants, one for each time series. This parameter column provides the information of the parameter when the operator is approximated and enables the model to provide predictions dependent on the parameter value. The modified data matrix can be handled like the original data matrix. However, the output should only give the prediction of feature values because the parameters are considered known and do not need to be predicted. Therefore, the parameter columns are eliminated while reconstructing the predicted state values.

3.2.2 Workflow

Figure 3.3 presents the research workflow. It consists of four steps: "data sampling," "data preprocessing," "EDMD framework," and "visualization," as described next.

1. Data sampling
   First, the crowd behavior data is sampled using the simulation software Vadere. The simulation scenario contains information such as the geometry, the number of pedestrians, walking speed, sampling rate, and sampling method. Scenarios are created as many as the range of the parameter values. The scenarios are then simulated on Vadere, and the target information is sampled at each time step during the simulation.

2. Data preprocessing
   Sampled data is preprocessed in this step. The sampled data contains uninterested parts, such as waiting time. Therefore, those parts of the data are truncated. The data is then sorted and merged into a large single data matrix.

3. EDMD framework
   EDMD approximates the Koopman operator from the provided dataset. It fits the model using a training dataset and tests the performance by predicting the test dataset with initial conditions. The test dataset is reconstructed and compared with the original test dataset. EDMD returns the RRMSE value as a score of the prediction.

4. Visualization
   Reconstructed test data is compared visually with the original test data. The eigenfunctions of the EDMD model are also visualized to understand how the EDMD model gives predictions by investigating eigencomponents.

The following sections describe each step in detail.

3.3 Data Sampling

To obtain the crowd dynamics data, Vadere, an open-source software for crowd dynamics simulation [9], is used to create an environment for crowd simulation and sample data. Vadere v3.0 is used in this thesis. Vadere provides the framework for crowd simulation and other models, such as cars and granular flow. Users can construct two-dimensional geometry in the graphical user interface, visualizing the simulation results and offering data analysis tools. Figure 3.4 shows an example of a geometry created by Vadere. The green rectangle on the left is the "source," which generates pedestrians represented by blue dots. The pedestrians walk towards a target, displayed as an orange rectangle. The red area in the middle is the measurement area, which counts the number of pedestrians at each time step. Geometry structure is constructed with obstacles, represented by grey rectangles. The pedestrians cannot go over obstacles.

Four scenarios with different conditions and geometries are created:

- *Scenario 1*
  A straight corridor with one gate with variable width between one source and one target. The number of pedestrians generated in the source is constant. There is one measurement area covering the source and the gate.

- *Scenario 2*
  A corridor with three rooms, each room is equipped with one source and one gate with a fixed width at the exit. The number of pedestrians is variable at each source. There are three measurement areas. Each of them covers one whole room area and over a gate.

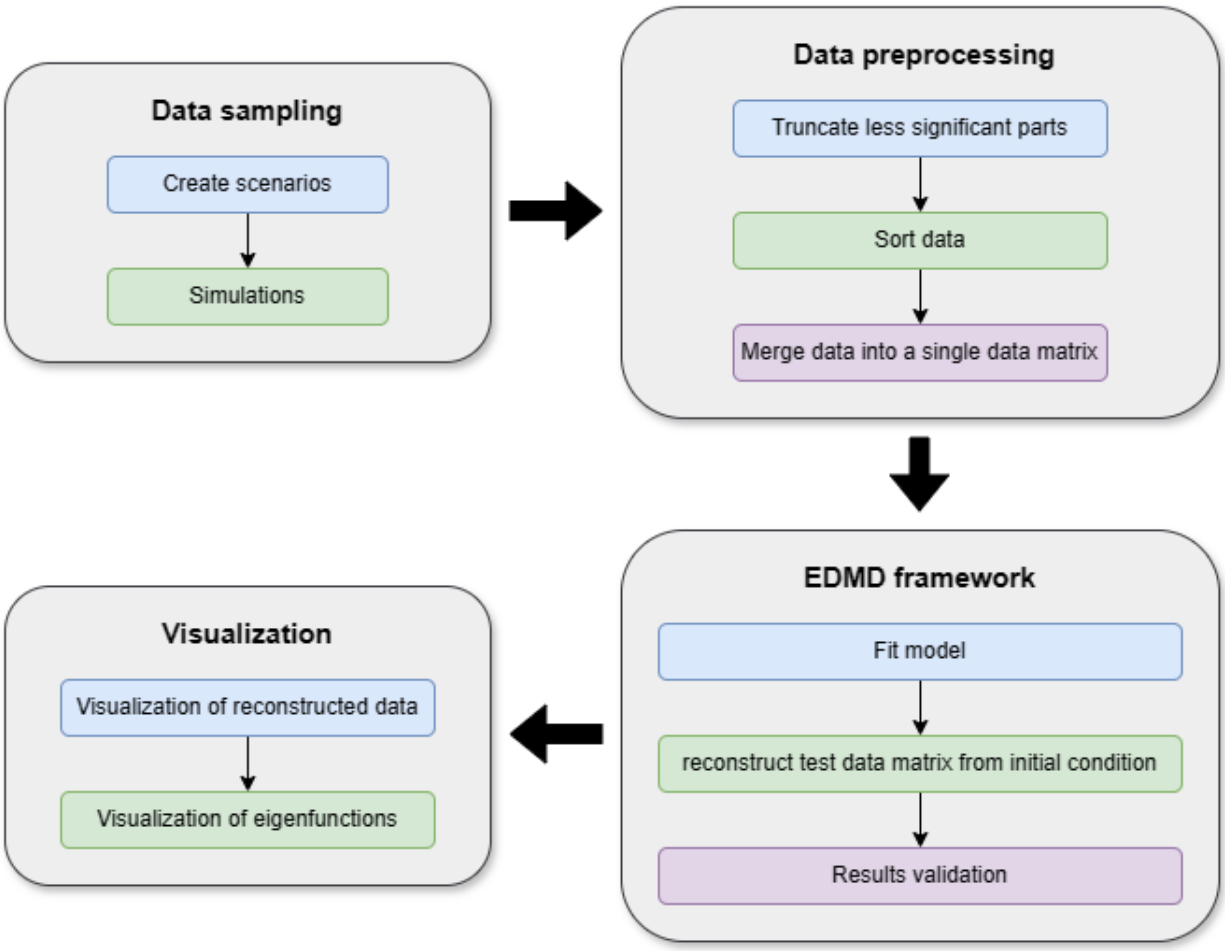


Figure 3.3 A diagram of the workflow. It is composed of 4 parts: Data sampling, data preprocessing, EDMD framework, and Visualization. In the data sampling, the simulation scenario is created and simulated using a simulation software called Vadere to obtain data on crowd behavior under defined conditions. The data is preprocessed in the next step and passed to the EDMD framework. EDMD fits the model based on the train data. Then, it reconstructs test data with a given initial condition and computes the error with the measurement data. Finally, the reconstructed data are visualized and compared. Additionally, the eigenfunctions are also visualized to analyze the significant edge components of the EDMD model.

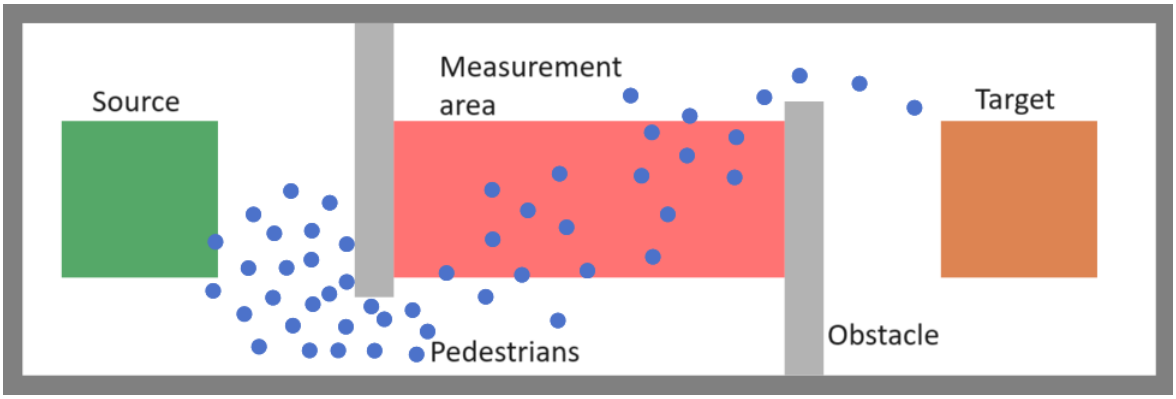


Figure 3.4 An example of a simulation scenario created using Vadere. The green rectangle is a pedestrian source, and the orange rectangle is a target that pedestrians head toward. The red rectangle represents the measurement area, counting pedestrians displayed as blue dots. Grey areas are objects that pedestrians cannot go through.

Table 3.1 Table of configurations of four scenarios.

Scenario	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Finish time	60.0	100.0	100.0	100.0
Simulation time step length	1.0	1.0	1.0	1.0
Real-time simulation time ratio	0.1	0.1	0.1	1.0
Digits per coordinate	2	2	2	2
Use fixed speed	True	True	True	True
Model	OSM	OSM	OSM	OSM
Number of sources	1	3	3	6
Number of targets	1	1	1	1
Number of measurement areas	1	3	3	6
Number of gates	1	3	3	6
Width of the gates	[1.1, 11.0] m	2.0 m	[1.0, 4.0] m	[1.0, 4.0] m
Event element count	50	[1, 50]	[1, 50]	[1, 50]
Constrains element max	50	50	50	50
Constrains time start	0.0	0.0	0.0	0.0
Constrains time end	0.0	0.0	0.0	0.0
Event position random	True	True	True	True
Event position free space	True	True	True	True
Distribution	Constant	Constant	Constant	Constant

- *Scenario 3*
  A corridor with three rooms, each with one source and one variable gate at the exit. The number of pedestrians is variable. There are three measurement areas. Each of them covers one whole room area and over a gate.

- *Scenario 4*
  A corridor connected to six rooms, three on the left and the others on the right. Each room contains a pedestrian source, which generates a random number of pedestrians. Gates between the corridor and rooms have a variable width. A measurement area overlaps each room and gate. Two measurement areas at the same level also overlap each other.

These scenarios are created in the graphical user interface in Vadere and are outputted as JSON files. The variation of the gate and the number of pedestrians are realized by modifying the JSON file using Python codes. Vadere runs all scenarios and generates CSV files that store the number of pedestrians in each measurement area at each time step. The Optimal Steps Model (OSM) is selected to model the behavior of pedestrians, which takes an individual's personal space into account when it updates the location of the pedestrians. OSM uses a circle around pedestrians to determine the next step location. Each scenario has one or more measurement areas that count the number of pedestrians in the area at each time step. The following subsections describe the scenario settings in detail. Also, the table 3.1 summarizes the configurations of the three scenarios.

3.3.1 Scenario 1: a simple corridor

This scenario consists of a corridor and a door in the middle. The pedestrian source is placed on the left side of the door, and the target is placed on the other side. Generated pedestrians then walk through the gate and reach the target. It counts the number of pedestrians in the measurement area that spans the gate and surrounding region. The gate width w [m] is defined as $w = x + 1$ [m], x in [0, 10] [m], where x is a variable. In this simulation, x takes the value at every 0.1 m, hence, 100 different values in total. The number of pedestrians is 50 and fixed for all cases. Scenario 1 has a very simple geometry to evaluate the effect of variable x. Figure 3.5 displays three simulation snapshots. The first picture is at the

Table 3.2 Geometry information of scenario 1. x is a parameter of gate width $w = x + 1$ [m]. The values in the table are in [m].

	X-coord	Y-coord	width	height
Source	2.0	2.0	4.0	10.0
Target	25.0	6.0	2.0	2.0
Measurement area	0.5	0.5	16.5	13.0
Objects	15.0	7.5 + $\frac{x}{2}$	1.0	6.0 - $\frac{x}{2}$
	15.0	0.5	1.0	6.0 + $\frac{x}{2}$

Table 3.3 Geometry information of scenario 2. The values in the table are in [m].

	X-coord	Y-coord	width	height
Sources	1.0	28.0	6.0	6.0
	1.0	20.5	6.0	6.0
	1.0	13.0	6.0	6.0
Target	16.5	1.0	8.5	3.5
Measurement areas	0.5	27.5	25.0	7.0
	0.5	20.0	25.0	7.0
	0.5	12.5	25.0	7.0
Objects	0.5	27.0	15.0	0.5
	0.5	19.5	15.0	0.5
	0.5	12.0	15.0	0.5
	15.5	32.0	0.5	2.5
	15.5	24.5	0.5	5.5
	15.5	17.0	0.5	5.5
	15.5	0.5	0.5	14.5
	25.5	0.5	0.5	34.0

beginning of the simulation. The pedestrians locate the source and start walking towards the gate as they keep their distance from each other. The second picture is when the pedestrians pass through the gate. Some pedestrians get stuck around the gate due to its width. Therefore, the number of pedestrians who stay before the gate depends on the gate's width. The pedestrians reach the target in the last picture, and the simulation terminates. Table 3.2 summarizes the detailed geometry structure.

3.3.2 Scenario 2: a corridor connected to three rooms

Scenario 2 has a more complicated geometry than scenario 1. Three sources are on the left side of the geometry, and each source is placed in its room. Hence, there are three rooms in total. The rooms and the corridor are connected via gates that are 2.0 m wide. The sources generate a random number of pedestrians ranging n in [1, 50], walking towards the target at the end of the corridor. Therefore, different groups of crowds get together in the corridor, which is supposed to create more complicated behavior in the crowd. Measurement areas are placed over each room and gate. In this scenario, the gate width is fixed, and the number of pedestrians is variable. Figure 3.6 shows the beginning of scenario 2 and the middle of the simulation when the pedestrians pass through the gates. Measurement areas cover all the rooms and the gates. Therefore, the initial value of the measurement is the number of pedestrians. As the pedestrians exit from the rooms, the measured value starts declining in the top measurement area. In the middle and the bottom measurement areas, the pedestrians from higher rooms flow in the measurement areas. Therefore, the measured values either decrease or increase when the number of incoming pedestrians exceeds that of the outgoing pedestrians. Table 3.3 describes the detailed geometry information.

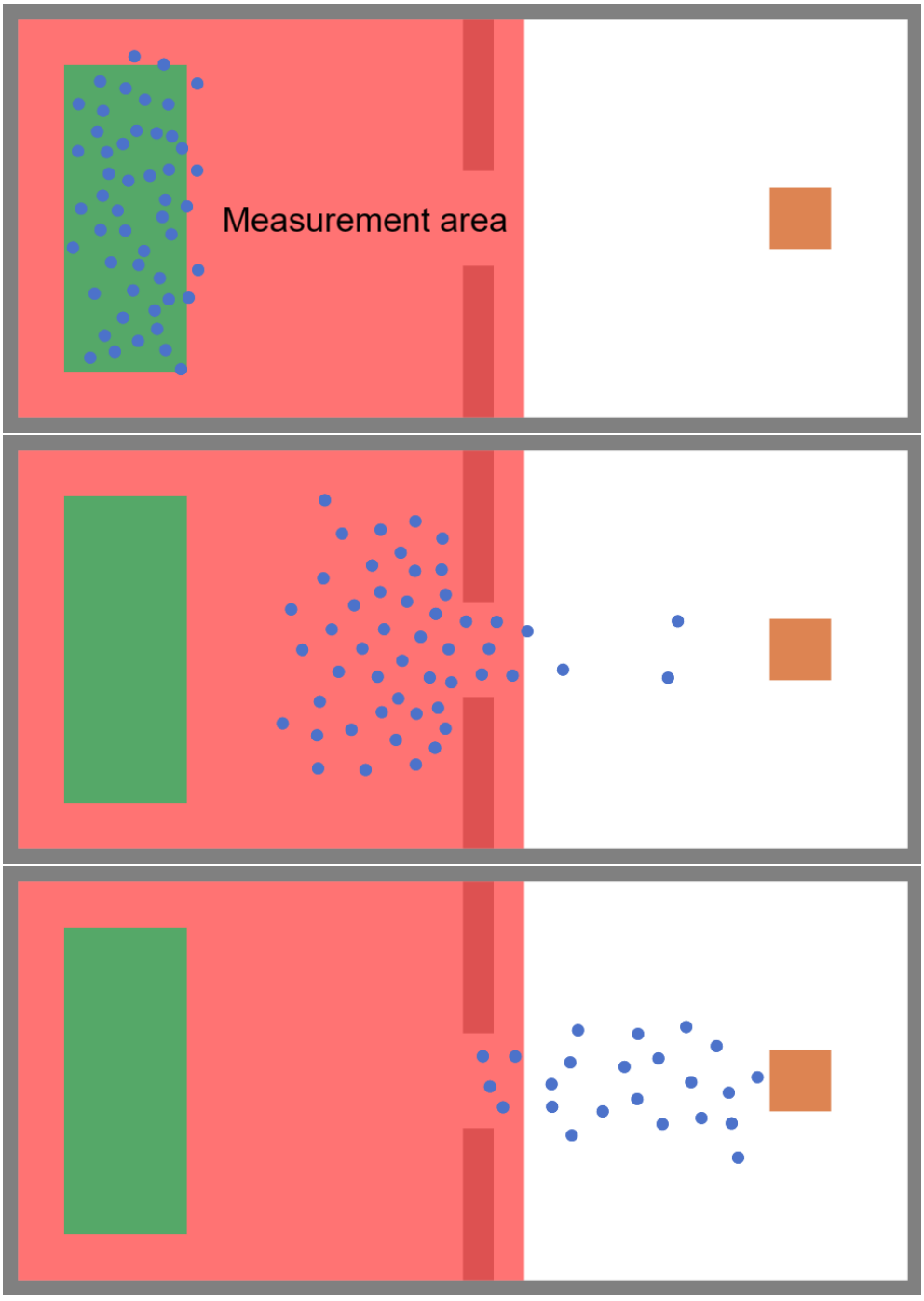


Figure 3.5 Visualization of the geometry and pedestrian flow of scenario 1. Top: Pedestrians are generated from the source. The number of pedestrians is 50. Middle: Pedestrians go through the gate. The width of the gate is within the range of [1.0, 11.0], and it is different in each simulation. Bottom: Pedestrians reach the target.

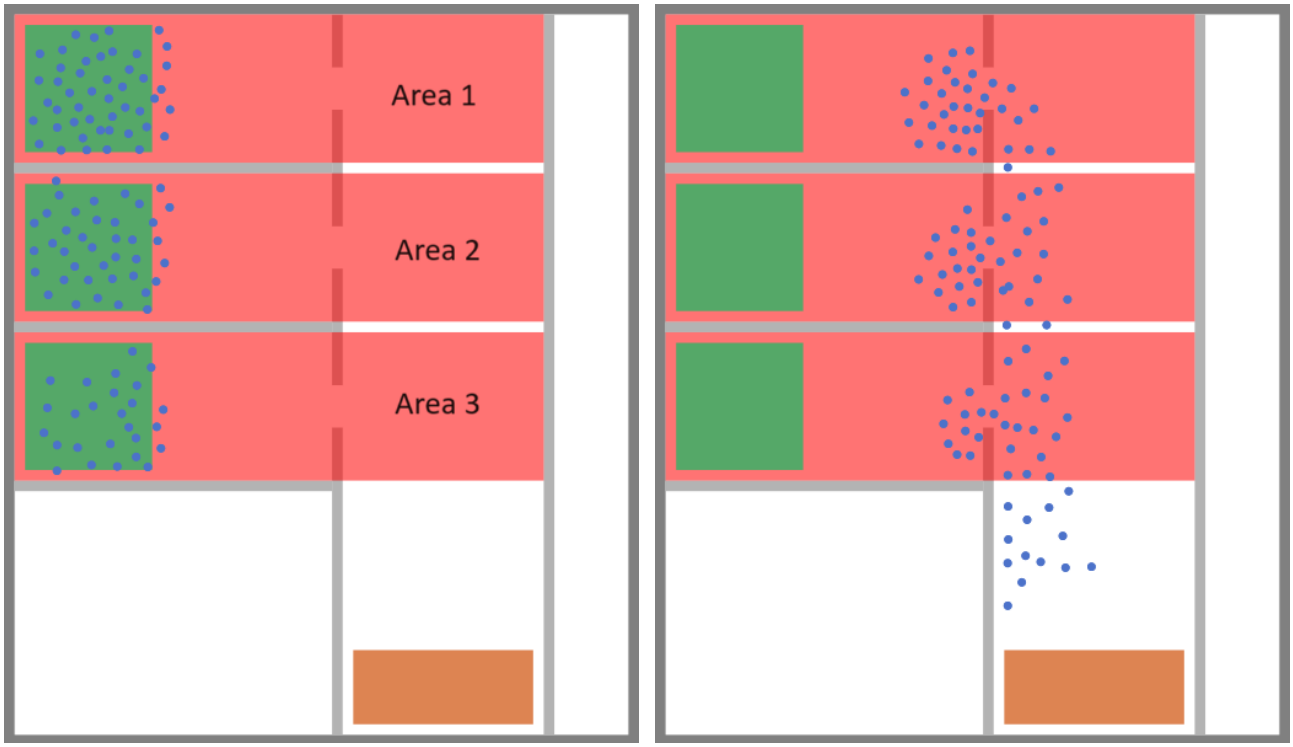


Figure 3.6 Visualization of the geometry and pedestrian flow of scenario 2. Left: Randomly generated pedestrians at three pedestrian sources. The number of pedestrians ranges [1, 50]. Each simulation randomly determines the number of pedestrians at each pedestrian source. Right: Pedestrians go through the gate and merge with other groups of pedestrians in the corridor. The width of the gate is fixed, and it is 2.0 m.

3.3.3 Scenario 3: a corridor connected to three rooms with varying gates

Scenario 3 modifies scenario 2 partly, and it has both variable gate width and variable number of pedestrians. Its geometry is the same as scenario 2. Hence, three pedestrian sources generate pedestrians in the range [1, 50]. The gates have the same width. Therefore, the number of variables is 4 in total. The gate width varies in the range [1, 5]. Figure 3.7 shows the beginning and the middle of the scenario 3. As observed in the simulations, the pedestrians get stuck at the gate when the width of the gate is small. Therefore, the variable gate width adds slightly more complexity to the other scenarios.

3.3.4 Scenario 4: a corridor connected to six rooms

Scenario 4, displayed in figure 3.8, extends scenario 3 by creating three more rooms on the other side of the corridor. The pedestrian sources have the same condition as scenario 3, randomly generating [1, 50] pedestrians in each source. However, the new sources in the rooms on the right are located 2.5 m farther away from the gate than on the left. The pedestrians from the rooms on the right enter the corridor later, creating delays compared to the other side, which provides further complexity in the dynamics. The number of pedestrians is counted and measured in the red areas. In the corridor, the measurement areas overlap at each level (top, middle, bottom). The gates are again variable, having the range of [1.0, 5.0] m, and the six gates are the same width.

3.4 Dictionary setting

Dictionary choice is a key factor in successfully constructing the Koopman operator. A dictionary consists of one or a composition of multiple observable functions g_i. In this thesis, the dictionary is determined by referring to preceding research from D. Lehmberg et al. [23], which used a composition of two observable

Table 3.4 Geometry information of scenario 3. The values in the table are in [m].

	X-coord	Y-coord	width	height
Sources	1.0	28.0	6.0	6.0
	1.0	20.5	6.0	6.0
	1.0	13.0	6.0	6.0
Target	16.5	1.0	8.5	3.5
Measurement areas	0.5	27.5	25.0	7.0
	0.5	20.0	25.0	7.0
	0.5	12.5	25.0	7.0
Objects	0.5	27.0	15.0	0.5
	0.5	19.5	15.0	0.5
	0.5	12.0	15.0	0.5
	15.5	32.0 + $\frac{x}{2}$	0.5	2.5 - $\frac{x}{2}$
	15.5	24.5 + $\frac{x}{2}$	0.5	5.5 - x
	15.5	17.0 + $\frac{x}{2}$	0.5	5.5 - x
	15.5	0.5	0.5	14.5 - $\frac{x}{2}$
	25.5	0.5	0.5	34.0

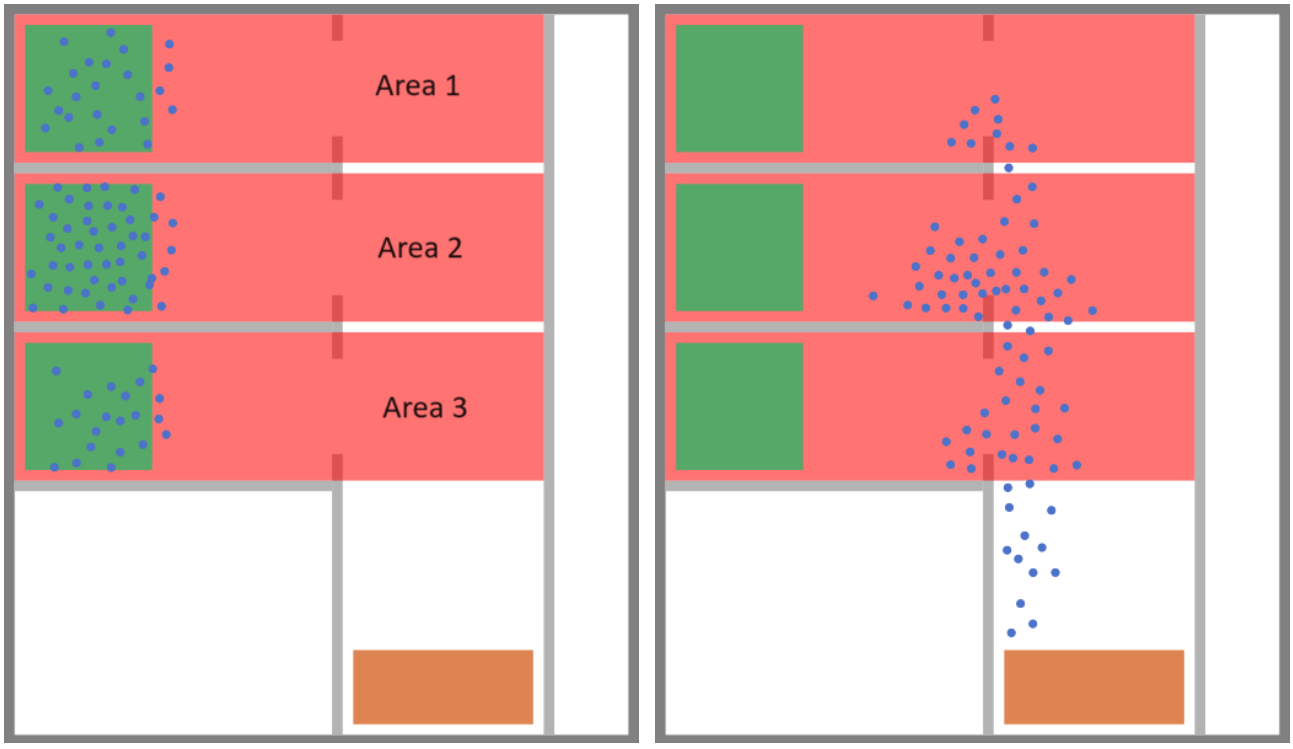


Figure 3.7 Visualization of the geometry and pedestrian flow of scenario 3. Left: Randomly generated pedestrians at three pedestrian sources. The number of pedestrians ranges [1, 50]. Each simulation randomly determines the number of pedestrians at each pedestrian source. Right: Pedestrians go through the gate and merge with other groups of pedestrians in the corridor. The width of the gate is randomly determined within the range of [1.0, 5.0] m.

Table 3.5 Geometry information of scenario 4. The values in the table are in [m].

	X-coord	Y-coord	width	height
Sources	1.0	28.0	6.0	6.0
	1.0	20.5	6.0	6.0
	1.0	13.0	6.0	6.0
	37.0	28.0	6.0	6.0
	37.0	20.5	6.0	6.0
	37.0	13.0	6.0	6.0
Target	16.5	1.0	8.5	3.5
Measurement areas	0.5	27.5	25.0	7.0
	0.5	20.0	25.0	7.0
	0.5	12.5	25.0	7.0
	16.0	27.5	27.5	7.0
	16.0	20.0	27.5	7.0
	16.0	12.5	27.5	7.0
Objects	0.5	27.0	15.0	0.5
	0.5	19.5	15.0	0.5
	0.5	12.0	15.0	0.5
	15.5	32.0 + $\frac{x}{2}$	0.5	2.5 - $\frac{x}{2}$
	15.5	24.5 + $\frac{x}{2}$	0.5	5.5 - x
	15.5	17.0 + $\frac{x}{2}$	0.5	5.5 - x
	15.5	0.5	0.5	14.5 - $\frac{x}{2}$
	0.5	27.0	17.5	0.5
	0.5	19.5	17.5	0.5
	0.5	12.0	17.5	0.5
	25.5	32.0 + $\frac{x}{2}$	0.5	2.5 - $\frac{x}{2}$
	25.5	24.5 + $\frac{x}{2}$	0.5	5.5 - x
	25.5	17.0 + $\frac{x}{2}$	0.5	5.5 - x
	25.5	0.5	0.5	14.5 - $\frac{x}{2}$

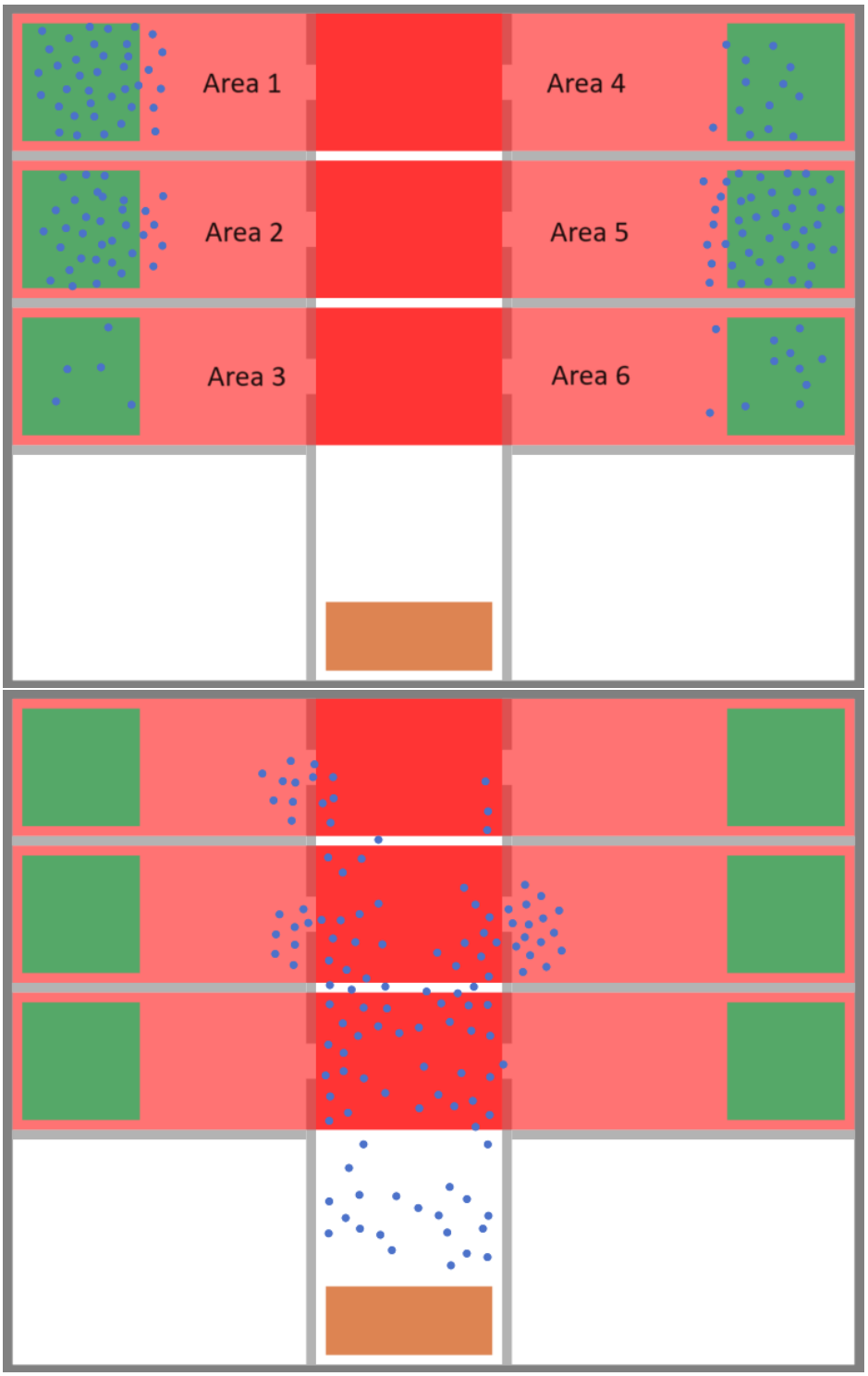


Figure 3.8 Visualization of the geometry and pedestrian flow of scenario 4. Top: Randomly generated pedestrians at six pedestrian sources. Three sources are located on the left and right sides of the central passage. The number of pedestrians ranges [1, 50]. Each simulation randomly determines the number of pedestrians at each pedestrian source. The right sources are placed 2.5 m further away from the gates. This is due to a delay in entering the passage from the right rooms. Bottom: Pedestrians go through the gate and merge with other groups of pedestrians in the corridor. The width of the gate is randomly determined within the range of [1.0, 5.0] m.

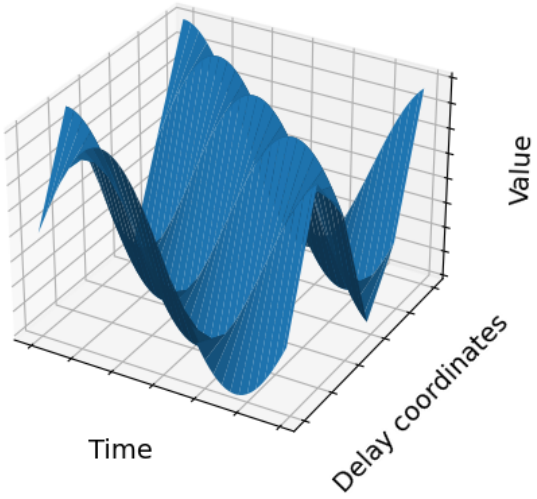


Figure 3.9 Taken's time-delay embedding of a sine curve $x = \sin(t)$. The embedding adds a delayed time series as a new coordinate of the system, which extends the spatial dimension.

functions: Taken's time-delay embedding g_1 and Diffusion maps g_2. The dictionary g is a composition of g_1 and g_2, $g = g_2 \circ g_1$. Dictionary transforms the state space variables x into observable functions $z = g(x_t)$, then Koopman operator K updates the observable functions of state space by $g(x_{t+1}) = Kz = Kg(x_t)$. A dictionary can be written in the form of vector values

$$g_i = \phi_i(x) = [\phi_{i1}(x), \phi_{i2}(x) \dots]. \tag{3.1}$$

Then, ϕ_i must belong to a observable space under which $\phi_i(x_t)$ is K-invariant for all t. This ensures that the Koopman operator comprehensively describes the flow of the state evolution. The dictionary components, Taken's time-delay embedding and diffusion maps, are introduced in the following subsection.

3.4.1 Taken's time-delay embedding

Takens' time-delay embedding is a method for embedding a time-series into a higher dimensional space. It reconstructs the dynamics of a system from a sequence of observations, preserving the properties of the system that do not change under smooth coordinate changes. The basic idea is to use a delay coordinate map, which concatenates previous outputs of a dynamical system into a vector. This vector forms a new space known as the embedding space. Therefore, Each point of the embedded space corresponds to observations at different times of the time series.

Suppose X is a vector of time-series data and $x_1, x_2, \dots x_n$ represents the state of different times. Then the Takens time-delay embedding constructs a new vector X' of higher dimension k by taking k states of different time as new space $X'_1 = [x_1, x_2, \dots, x_k]$, and X' evolves to $X'_2 = [x_2, x_3, \dots, x_{k+1}], X'_3 = [x_3, x_4, \dots, x_{k+2}], \dots, X'_{n-k} = [x_{n-k}, x_{n-k+1}, \dots, x_n]$. The reason for including time-delay embedding is that measured states are often not dynamically closed (or Markovian). For example, if only the current pedestrian measurements are available, the information about how the number of pedestrians changes is missing. Time-delay embedding enriches a state with temporal context. If sufficiently many prior states are included, the time delay embedding introduces a new geometry on the data that reconstructs the system's qualitatively equivalent state space, on which the dynamics are well-defined. Figure 3.9 shows an example of the time-delay embedding of a time series data. A sine curve $x = sin(t + \frac{\pi \triangle t}{2})$ is extended in the spatial dimension by adding new coordinates using five delay coordinates; each coordinate has a different number

of time delays. As the figure shows, the time-delay embedding concatenates lags of the sine curve and reconstructs dynamics, which have observations at different times in spatial directions.

3.4.2 Diffusion maps

Diffusion maps are a machine learning method for dimensionality reduction or identifying geometric structures. This method is beneficial for analyzing non-linear structures in high-dimensional data. Coifman first introduces it [24]. It computes a family of embeddings of a dataset into Euclidean space whose coordinates can be computed from the eigenvectors and eigenvalues of a diffusion operator on the data.

The algorithm is based on the concept of a random walk on a graph. Each data point is represented as a node in the graph, and the edges between nodes represent the probability of transitioning from one data point to another in a single step of the random walk. This probability is determined by the similarity between data points, typically measured using a kernel function. It first computes the distance matrix of the given data matrix X by

$$D(x_i, x_j) = ||x_i - x_j|| \tag{3.2}$$

where x_i, x_j are pairs of all points. A distance matrix is a square matrix whose element represents the distance of a pair of points. A distance matrix provides the data's geometry information. Then, this distance matrix is transformed into an affinity matrix

$$A(x_i, x_j) = \exp(-\frac{D(x_i, x_j)^2}{\sigma}), \tag{3.3}$$

where σ is a scaling parameter that determines the width of the Gaussian. In this definition, a Gaussian kernel is used. It returns a value between 0 and 1, where 1 indicates that the two points are identical, and 0 indicates that they are infinitely far apart. An affinity matrix represents the affinity or similarity of each pair of data points. An affinity matrix becomes the basis of the diffusion process. Then, take the row normalized affinities, which is equal to degree inverse times the affinity matrix, and get the row stochastic diffusion operator

$$\begin{aligned} P(x_i, x_j) &= \frac{A(x_i, x_j)}{\sum_j A(x_i, x_j)} \\ &= D^{-1}A, \end{aligned} \tag{3.4}$$

which is a Markov chain. The diffusion map is the right eigenvectors weighted by the corresponding eigenvalues. It creates new coordinates of n-dimension for n-data points. Since the eigenvalues are one or less than one, the magnitudes of eigenvalues are the significance of the corresponding eigenvectors. Hence, the first several eigenvectors with the greatest eigenvalues construct new reduced-dimensional coordinates. Figure 3.10 visualize an example of diffusion maps. The left figure is a three-dimensional helix data, which consists of 1000 data points, and $x = r\sin\theta$, $y = r\cos\theta$, $z = r$, where $r = [-2, 2], \theta = [-4\pi, 4\pi]$. The figure on the right maps the data in two new coordinates created from diffusion maps. It is observed that the diffusion maps extract the geometrical feature of the helix data while preserving the relationships of each pair of data points.

One of the critical advantages of diffusion maps is that they preserve the data's local and global geometric structure. This makes them particularly useful for dimensionality reduction methods such as manifold learning, where the goal is to uncover the underlying low-dimensional structure of the data. Additionally, diffusion maps are robust to noise since the random walk tends to smooth the local fluctuations.

3.5 Datafold Framework

In this thesis, the time-series data from crowd dynamics simulation is given as the input to the EDMD, a data-driven Koopman operator approximation method. EDMD is chosen for applicability to linear dynamics and flexibility of dictionary setting, which resulted in high accuracy in preceding research [23]. EDMD can

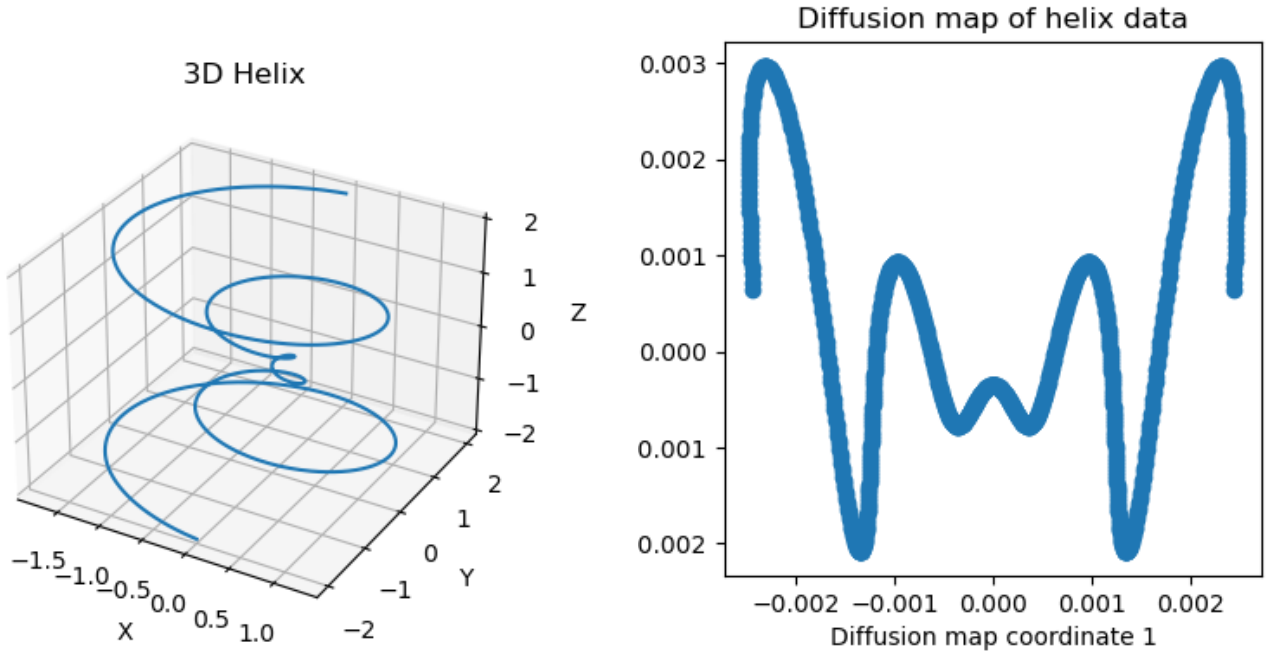


Figure 3.10 Diffusion maps of three-dimensional helix data. $x = r\sin\theta$, $y = r\cos\theta$, $z = r$, where $r = [-2, 2], \theta = [-4\pi, 4\pi]$. New coordinates from diffusion maps capture the main modes of variation in the data.

also provide the Koopman eigenfunctions, eigenvalues, and modes that are useful for analyzing dynamics vector by vector. A package called datafold is used to build the EDMD model.

Datafold is a Python package that provides a framework to construct operator-theoretic models to identify dynamical systems from time series data and infer geometrical structures from point clouds. It is open-source software and was introduced in the publication from Lehmberg et at. [25]. Datafold has three layers of workflow hierarchies that enable a high degree of modularity. In this project, the dataset is transformed to TSCDataFrame, a data frame for time series collections, and the EDMD is constructed using datafold.appfold package.

3.5.1 Parameter setting

EDMD requires several parameters in its process, and those parameters mainly appear in the dictionaries. In this work, the dictionary is a composition of Taken's time-delay embedding and diffusion map, and both algorithms have parameters to be configured. Some model parameters are determined based on the theory and dataset, and others are obtained from grid search.

Taken's embedding requires the number of delays as a parameter, the lag before embedding starts, and the frequency at which the samples are embedded. In this thesis, the model is assumed to perform prediction of the crowd dynamics from a given few data points as an initial condition, at most four data points. Therefore, the number of time delays must be small as well to be able to apply the embedding to the provided initial data. The number of time delays is investigated by grid search, and the outcome is shown in figure 3.11. It compares the accuracies of the predictions using Root Mean Square Error (RMSPE) with different numbers of time delays. 1, 2, 3, and 4 time delays are compared. Scenario 1 has a minimum RMSPE value at the time delay 1, and increasing delays do not improve the model accuracy. In scenarios 2 and 3, no overall linear correlations exist between the number of time delays and RMSPE values. However, a smaller time delay maximizes the model's ability. In scenario 4, a growing number of time delays degrades the model's prediction accuracies in areas 1 and 4, the top measurement areas. However, areas 3 and 6, the bottom measurement areas, have the opposite tendency, that their RMSPE is improved. Areas 2 and 5 seem to be independent of the number of time delays. A drawback of adding more time delays is that it enlarges the size of the data matrix, requiring more computational cost in the later steps. It could be a crucial issue, especially if the computational resource is limited. Therefore, one

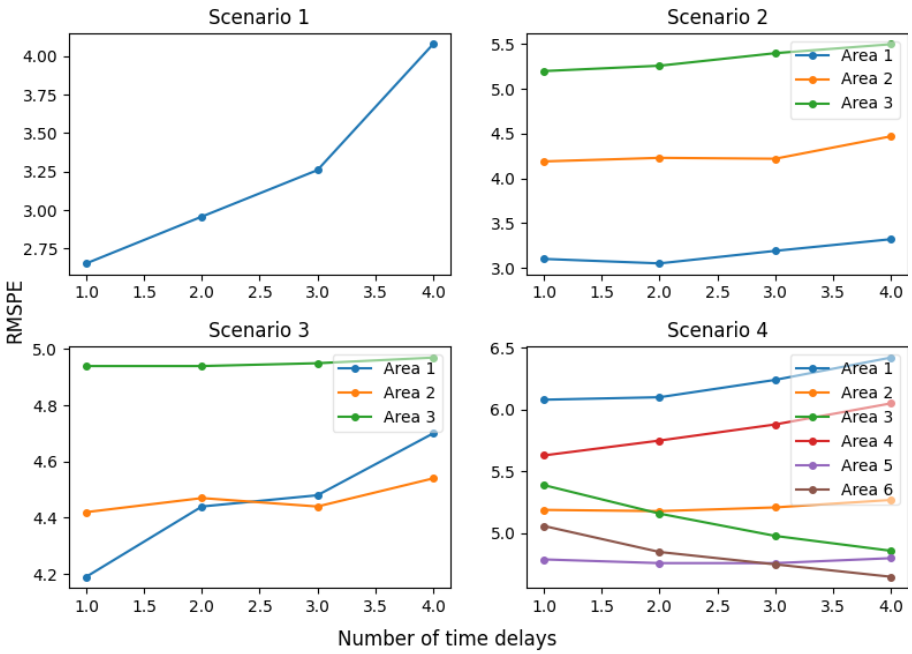


Figure 3.11 Comparisons of Root Mean Square Percentage Errors (RMSPE) with different numbers of time delays for each scenario. The numbers of time delays vary from 1 to 4, incrementing by 1. RMSPEs are calculated for each measurement area, showing the accuracies of the predictions.

time delay is chosen for scenario 4 as well. The sampled data does not require lag for the time delays, meaning that the delayed coordinates start from time 0. Also, the embedding frequency is set to 1, which samples every data point from the head of the data matrix. Additionally, there is a parameter $k >= 0$, which is a weight of exponential factor in delayed coordinates $e^{-dk}(x_{-d})$ where $d = 0, \ldots, delays$ is the delay index. Results from grid search showed that any positive k value $k > 0$ results in a larger MSE. Hence, $k = 0$ is selected.

Diffusion maps need to define the connectivity based on a chosen kernel. In this thesis, the Gaussian kernel is used, which is a function of the following form:

$$K(x, y) = \exp(\frac{-||x - y||^2}{2\sigma}), \tag{3.5}$$

where σ is a bandwidth and $\sigma = median(||y_j - y_i||_2^2)$. Internally, It calculates the norms of every point to get the distance matrix. Hence, the Gaussian kernel measures how close every two points are in the space and assigns a higher weight to nearby points. Diffusion maps also require the number of eigenpairs to compute from the kernel matrix. This number is decided by a grid search, where the range of the grid spans [10, 100]. Figure 3.12 compares prediction results with different numbers of eigenpairs. It evaluates the prediction accuracies with Root Mean Square Percentage Error (RMSPE) for each measurement area. In this thesis, the number of eigenpairs is determined as the number from which the model's prediction ability does not improve significantly by adding more eigenpairs. In the figures, once a plateau appears in the graph, the number at the beginning is the optimal number of eigenpairs. By doing so, the model minimizes computational cost while maintaining an acceptable prediction accuracy. The change of RMSPE becomes flat at 70 eigenpairs in scenario 1. Hence, the number is set to 70 in scenario 1. In scenario 2, RMSPE at areas 2 and 3 reaches a plateau at 60 eigenpairs, while the value in area 3 becomes flat at 70 eigenpairs. 70 is chosen for the number of eigenpairs in scenario 2 so that the model performs sufficiently well in every measurement area. In scenario 3, the value in area 2 continuously decreases as the number of eigenpairs grows; therefore, the number is determined by considering the other two areas, which gives 70 eigenpairs as the most suitable number. Scenario 4 shows two plateaus, from 40 to 70 and 70 to 100

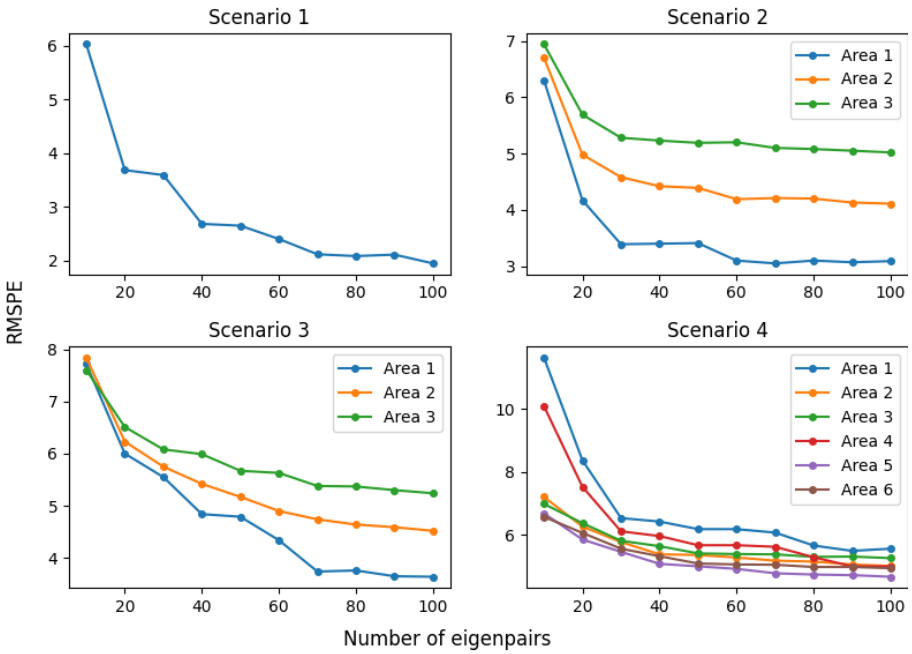


Figure 3.12 Comparisons of Root Mean Square Percentage Errors (RMSPE) with different numbers of diffusion maps eigenpairs for each scenario. The numbers of eigenpairs vary from 10 to 100, incrementing by 10. RMSPEs are calculated for each measurement area, showing the accuracies of the predictions.

eigenpairs. Considering the previous scenarios, more eigenpairs are thought to be required to represent higher dimensional space. Hence, 80 eigenpairs are obtained to construct a model in scenario 4.

Table 3.6 describes the detailed parameter settings for all scenarios.

3.5.2 Dataset description

Datafold offers a specific data structure, TSCDataFrame, for the time-series data. It is a multi-indexed data frame based on the Pandas DataFrame class. The row index must have two levels: ID and time

- ID
  The time series ID. In this project, the ID number corresponds to the simulation number. Therefore, there are as many IDs as the number of simulations in each scenario.
- time
  Time values of the time series data.

Therefore, the simulation data are concatenated into a single large data matrix, and each time-series data is labeled by adding an ID number. The detailed data description in each scenario is described in the following sections.

Scenario 1: a simple corridor

A simulation of Scenario 1 outputs data consisting of time and the number of pedestrians counted in one measurement area at each time for 60-time steps. This creates 60 rows and two columns of the data matrix. A new column is added to include the gate width as a parameter in the time series. The column has a constant value x in the range [0, 10] corresponding to the gate width. The variable x starts from 0.1 and increases by 0.1 in each simulation. Therefore, the number of simulations is 100. The training dataset consists of 80 results randomly selected from the 100 results. The remaining results are used as a test dataset. Therefore, there is no duplicated data in training and test datasets. Some parts of the time

Table 3.6 Table of parameters for EDMD setup.

Scenario	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Time series length	39	34	34	34
Number of pedestrians	50	[1, 50]	[1, 50]	[1, 50]
Gate width	[1.0, 11.0]	2.0	[1.0, 5.0]	[1.0, 5.0]
DMD model	DMDFull	DMDFull	DMDFull	DMDFull
Include ID state	False	False	False	False
Dict preserves ID state	infer	infer	infer	infer
Stepwise transform	False	False	False	False
Sort Koopman triplets	True	True	True	True
Use transform inverse	False	False	False	False
Time-delays	1	1	1	1
lag	0	0	0	0
frequency	1	1	1	1
kappa	0	0	0	0
dmap kernel	Gaussian	Gaussian	Gaussian	Gaussian
num eigenpairs	70	70	70	80
alpha	1	1	1	1
time exponent	0	0	0	0
is stochastic	True	True	True	True
symmetrize kernel	True	True	True	True

series contain no meaningful data on dynamics. Therefore, the beginning and ending of the time series are eliminated, and every time series is truncated to 39 time steps. Figure 3.13 shows all the results of simulations in scenario 1. As the variable x grows, the number of pedestrians counted in the measurement area converges to 0 earlier. This is due to the decrease in the number of pedestrians stuck at the gate as the gate becomes wider. Furthermore, it is observed that after x reaches around 5, the results almost stay the same even though the width of the gate grows. This is because, after a certain width of the gate, pedestrians can pass through the gate without getting stuck around the gate. Therefore, the results with larger gate widths show the same data trend.

Scenario 2: a corridor connected to three rooms

A simulation of scenario 2 has 100 time steps. Three measurement areas create three columns of measured pedestrian numbers. Also, the initial numbers of pedestrians at each source are the parameters. Hence, the time series data appends three additional columns representing the parameter values. Therefore, the data matrix consists of 7 columns: time steps, three measurement data, and three parameters. The scenario is repeatedly simulated with different parameter values 1000 times to obtain 1000 samples. Some parts of the time series are truncated to keep only the meaningful data, leaving 34 time steps in each time series. All simulation results are concatenated and become one data matrix. Repeating the same procedure, two data matrices of this shape are obtained. One is assigned as a training dataset, and the other is assigned as a test dataset. Hence, the training and test datasets are different data from the same scenario with randomly generated parameters. Four samples of data are visualized in figure 3.14. The three lines represent the measured number of pedestrians in each measurement area. Area 1 is the measurement area over the top room, area 2 is located in the middle, and area 3 is over the bottom room. The number of pedestrians measured in area 1 decreases monotonically over time. Meanwhile, the other two values show complicated behavior. The values in area 2 decrease in most time steps; however, sometimes they go up as the pedestrians from the top room flow in area 2. In the same way, area 3 measured many pedestrians who came into the area in the middle of the simulations. It is observable that the values of area 2 and area 3 depend on the change of values in area 1 and area 2. Figure 3.15 displays the

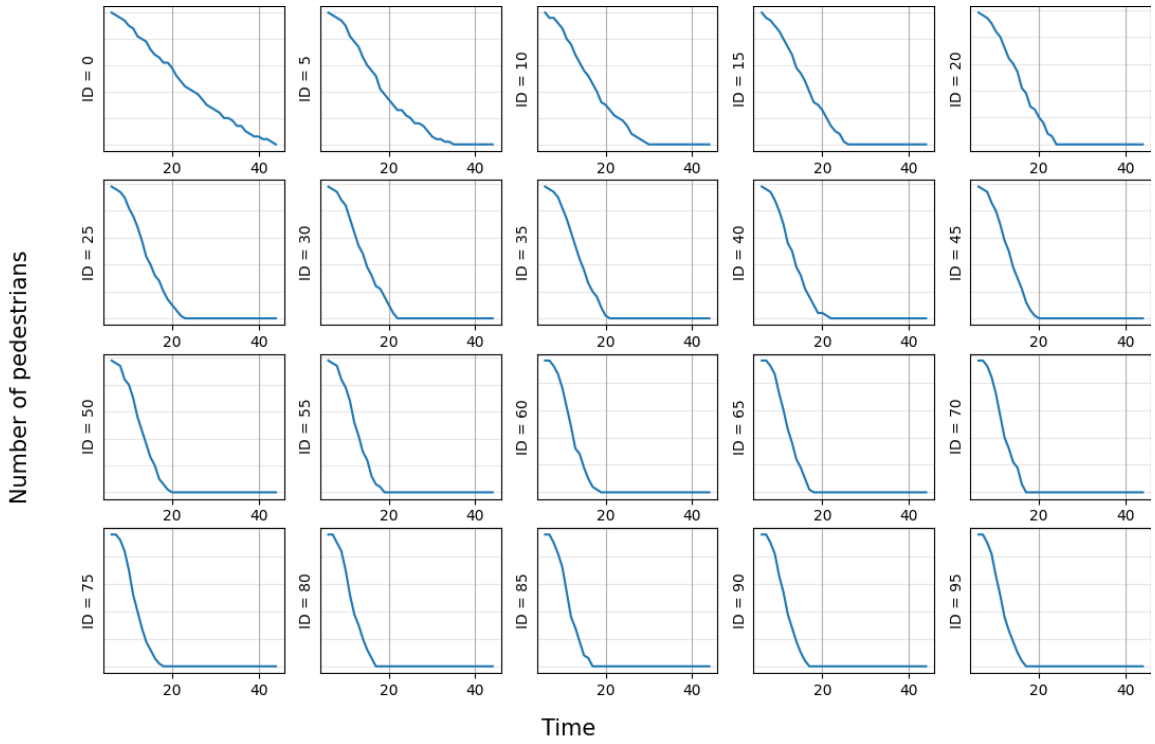


Figure 3.13 Visualization of the simulation results from scenario 1. The y-axis shows the number of pedestrians, and the x-axis shows the time steps. The number on the left of each figure represents the id of the simulation. Id 1 has a gate width of 1.1, and as the number of the id increases by 1, the gate width also gets larger by 0.1. Hence, the simulation with id 100 has a gate width of 11.0.

transition of the measured values over time in 3D coordinates. Each axis represents a measured value in an area: area 1 at the top of the geometry, area 2 in the middle, and area 3 at the bottom. The elapsed time is visualized by color: blue represents the beginning of a simulation, while red represents the end. The initial conditions are fixed: The number of pedestrians generated at the sources in areas 1 and 2 is fixed at 30. Area 3 has a variable initial number, ranging between 5 to 50, and equally distanced by 5.

The Visualization describes how the state evolves. Area 1 converges to 0 first. Then, the other values also go towards the value 0. The number counted in Area 3 increases when the initial number is small. However, it is not observed when the initial number is large. This is because the number of incoming pedestrians surpasses the number of leaving pedestrians when the initial number in area 3 is small. It is observed that the values are dependent on higher area values, and the values smoothly transition over time.

Scenario 3: a corridor connected to three rooms with varying gates

The data matrix structure in scenario 3 is similar to the one in scenario 2. Scenario 3 also has 100 time steps in each simulation. However, the data matrix of scenario 3 has another parameter column that represents the gate's width. Hence, the data matrix has eight columns. Scenario 3 is simulated 1000 times with different parameters for both the training and test datasets. Unnecessary parts of the time series are truncated, yielding a time series with 34 time steps. Figure 3.16 shows four samples of scenario 3. The overall trend of the values is similar to those of scenario 2. However, the new parameter, the width of the gate, changes the decreasing rate of the number of pedestrians.

Figure 3.17 describes the evolution of the data, starting from the initial value to the termination of simulations, with different lines representing different gate widths. All the data have the same initial number of pedestrians, 50, at the sources. The width of the gate is modified by adding 0.4 m for each line, starting from 1.0 m up to 5.0 m, yielding ten lines in total. The color on the lines displays the elapsed time. The

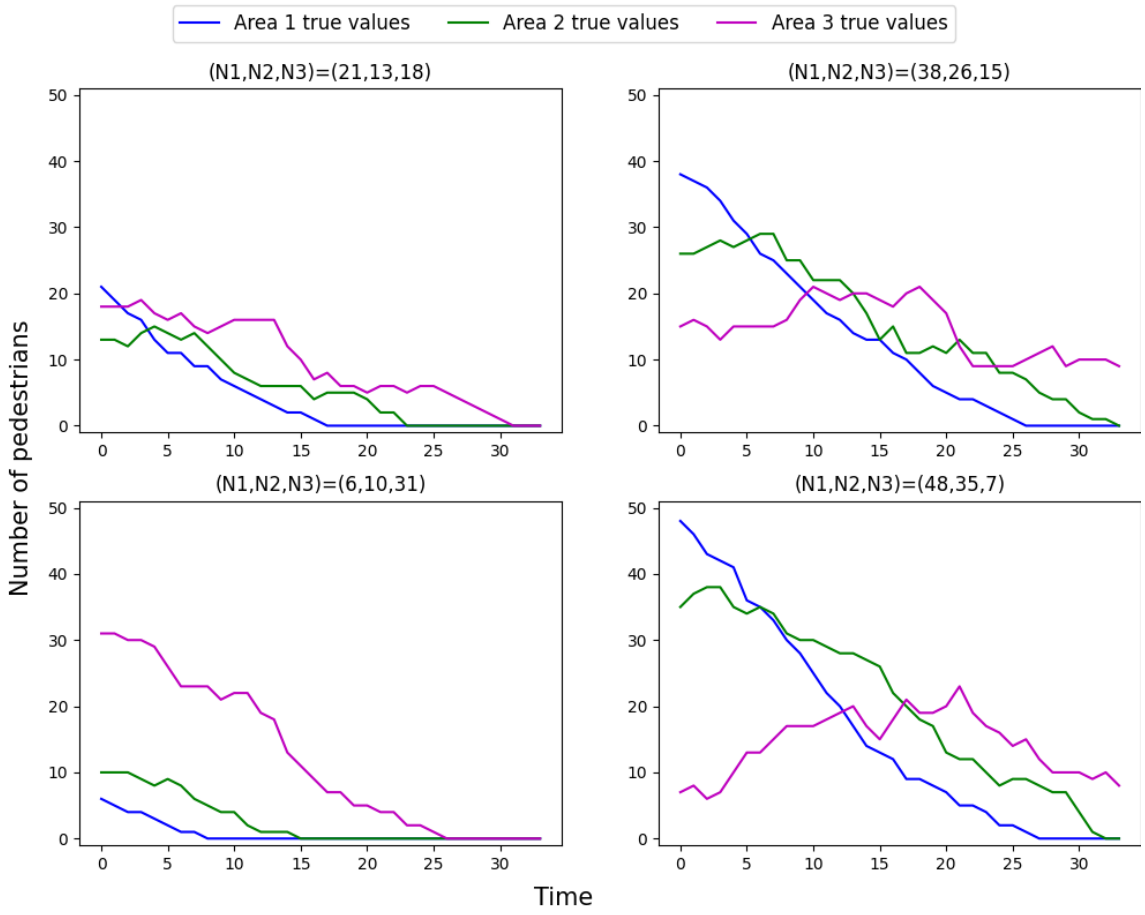


Figure 3.14 Visualization of the simulation results from scenario 2. The y-axis shows the number of pedestrians, and the x-axis shows the time steps. Three lines display the number of pedestrians at each measurement area. The initial number of pedestrians in each source is shown above each figure, where N1, N2, and N3 represent the pedestrian numbers at the top, middle, and bottom sources, respectively.

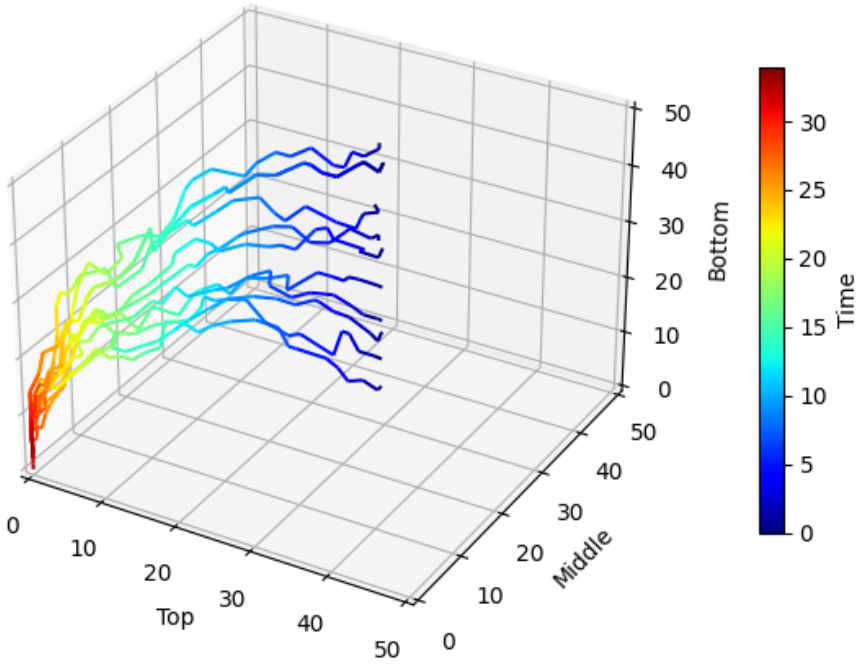


Figure 3.15 Visualization of several data in 3d space, where each axis represents the number of pedestrians counted in each measurement area. Area 1 is located on the top of the region, area 2 in the middle, and area 3 in the bottom. The number of pedestrians generated in each source is fixed: Area 1 and area 2 generate 30 pedestrians in each source, and area 3 generates 5 to 50 pedestrians, equally distanced by 5. The color on the lines shows the elapsed time.

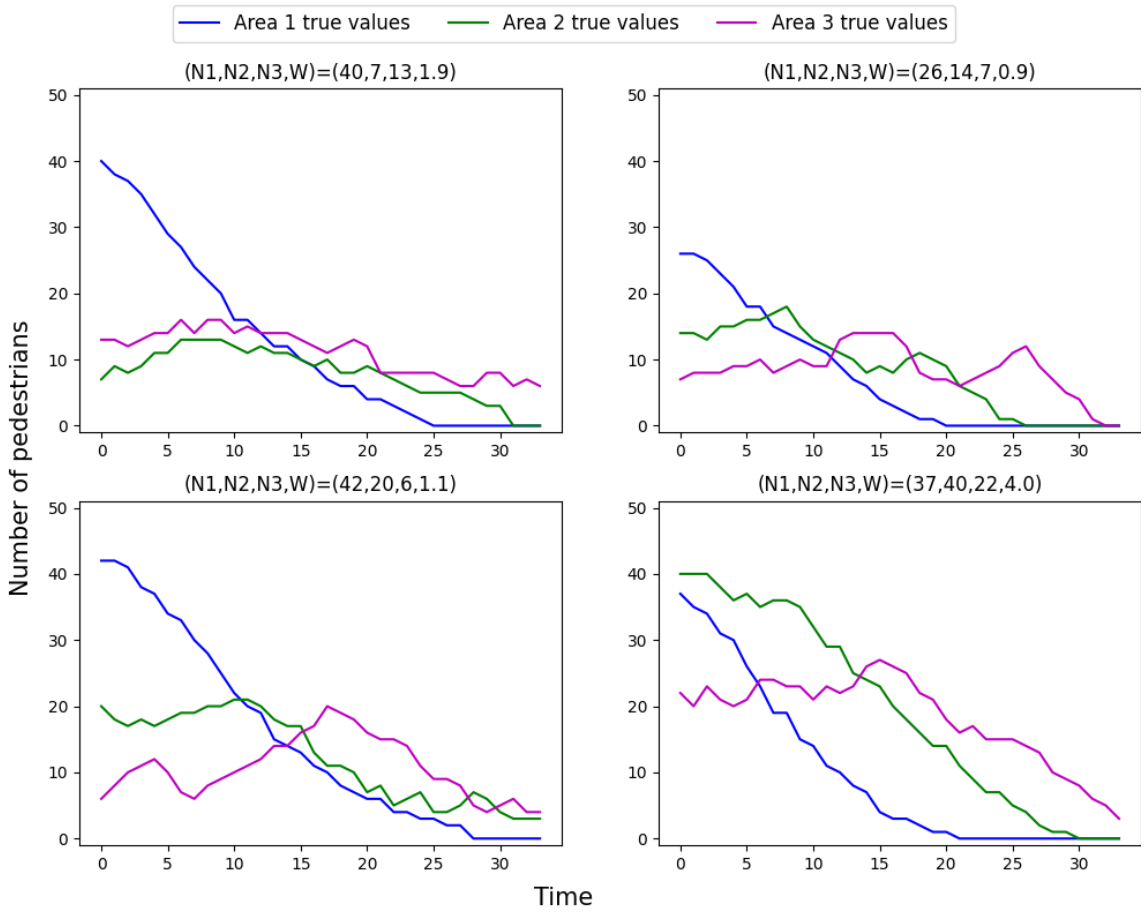


Figure 3.16 Visualization of the simulation results from scenario 2. The y-axis shows the number of pedestrians, and the x-axis shows the time steps. Three lines display the number of pedestrians at each measurement area. The initial number of pedestrians in each source is shown above each figure, where N1, N2, and N3 represent the pedestrian numbers at the top, middle, and bottom sources, respectively. W represents the gate width.

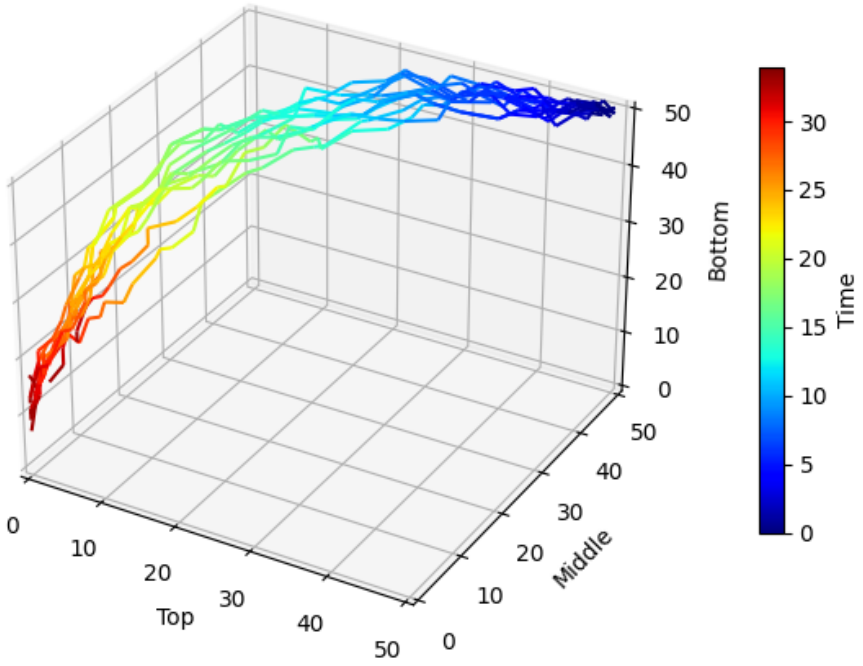


Figure 3.17 Visualization of several data in 3d space, where each axis represents the number of pedestrians counted in each measurement area. Area 1 is located at the top of the region, area 2 is in the middle, and area 3 is at the bottom. The number of pedestrians generated in each source is fixed to 50, and the width of the gate is only variable. The gate width spans 1.0 m to 5.0 m, increasing by 0.4 m for each sample. The color on the lines shows the elapsed time.

color shows that the difference in the gate width creates the gradation of the color: The length of each color segment is shorter for the line width narrow gate width.

Scenario 4: a corridor connected to six rooms

1000 simulations for the training dataset and another 1000 for the test dataset are carried out for scenario 4. The same procedures as the other scenarios are applied to preprocess the data matrices. The simulations have randomly generated different parameter values: the initial number of pedestrians at each source and the width of all gates. Each room has one measurement area. Hence, there are six measured values: area 1 is located on the top left, area 2 is in the middle left, area 3 is in the bottom left, and area 4, and area 5 and area 6 are on the right side in the same order. Therefore, the pedestrians observed in area 1 and area 4 go through area 2, area 5, area 3, and area 6 after they leave the higher measurement areas. This movement of pedestrians from one area to another creates one or several peaks in lower measurement areas. 3.18 shows the observed crowd data from four simulations. This Visualization of the results describes different peaks in the measurement. Areas 1 and 4 have peaks at the beginning of the data since the crowds only leave the areas. Hence, the graphs show a monotonous decrease in the numbers. Meanwhile, the peaks are observed after several steps in area 2, area 5, area 3, and area 6. This is due to the incoming pedestrians from higher areas. Area 2 and area 5 have different phases from area

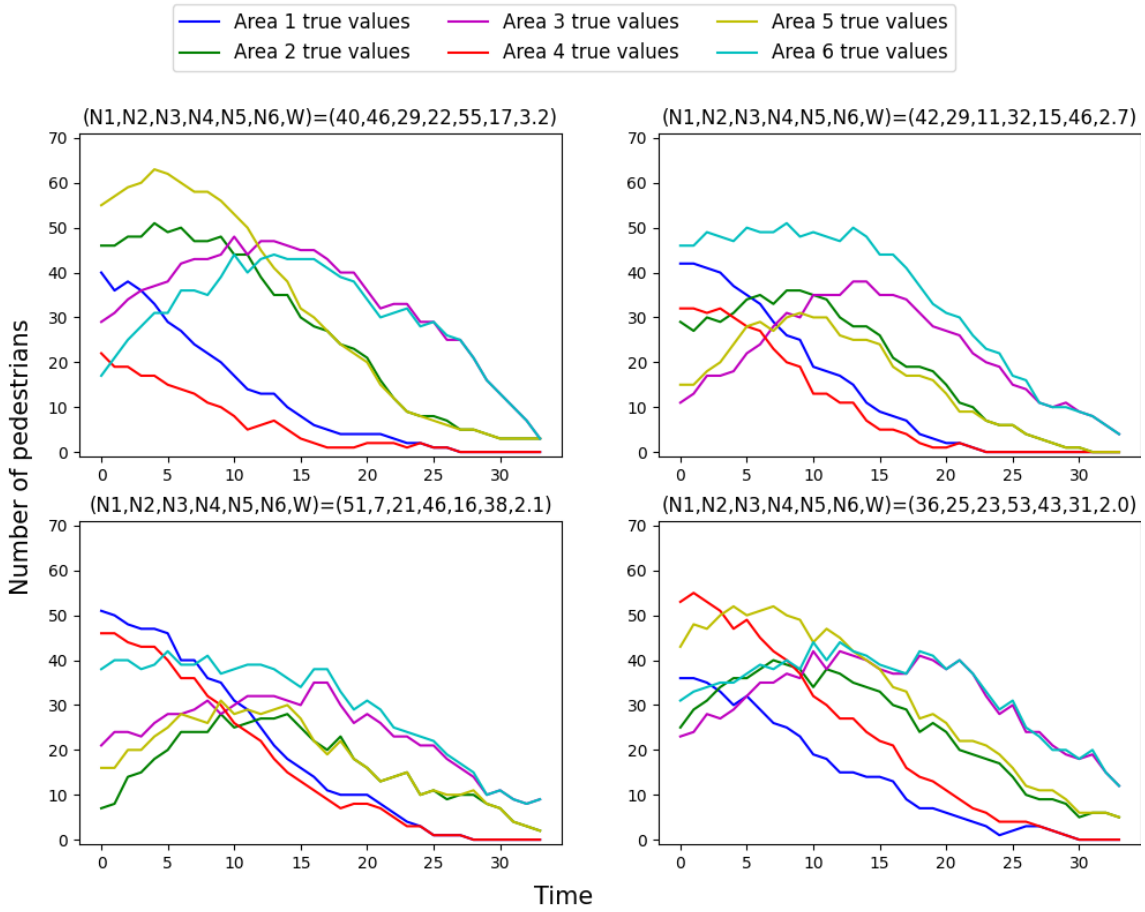


Figure 3.18 Visualization of the simulation results from scenario 2. The y-axis shows the number of pedestrians, and the x-axis shows the time steps. Three lines display the number of pedestrians at each measurement area. The initial number of pedestrians in each source is shown above each figure, where N1, N2, and N3 represent the pedestrian numbers at the top, middle, and bottom sources, respectively. W represents the gate width.

3 and area 6 as a result. These four values from areas 2, 3, 5, and 6 are characterized by parabolic shape and small fluctuations due to incoming and outgoing pedestrians, creating more dynamics complexity.

3.6 Crowd dynamics prediction

3.6.1 Scenario 1: a simple corridor

Training dataset X_{train} is mapped to the dictionary representation in the training process. At first, Taken's time-delay embedding expands the spatial dimension using one delayed coordinate. Due to the embedding, the length of the matrix is shortened by one time step. Then, the data matrix proceeds to the diffusion map procedure, which computes the 70 eigenvectors of significant eigenvalues from the Gaussian kernel of the data matrix's diffusion operator. This process yields a data matrix with 70 columns. Each column is a new coordinate created using diffusion maps. Different samples are assigned different IDs, so the training treats each sample as a different time series. X_{train} is then passed to a method that approximates the Koopman operator by solving a least square problem, and also, the Koopman mode, which is the inverse map from dictionary states to the original states, is set up. After the model fits the given training data matrix, the model performance is tested with test data X_{test}. X_{test} is also transformed to the dictionary representation by the sequence of time-delay embedding and diffusion map. Hence, the Koopman eigenfunction, which maps the original state to the dictionary state, is computed for the test data matrix.

Table 3.7 Performance evaluation of the model in scenario 1. From the top, Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), Root Mean Square Percentage Error (RMSPE), R^2.

Measurement area	Area 1
MAE	0.65
MAPE	1.40 %
RMSE	1.22
RMSPE	2.65 %
MBE	0.07
MBPE	0.16 %
R^2	0.99

Table 3.8 Performance evaluation of the model in scenario 2. From the top, Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), Root Mean Square Percentage Error (RMSPE), R square.

Measurement area	Area 1	Area 2	Area 3
MAE	0.66	1.19	1.64
MAPE	2.20 %	3.12 %	4.00 %
RMSE	0.93	1.59	2.13
RMSPE	3.10 %	4.19 %	5.20 %
MBE	-0.01	0.01	0.05
MBPE	-0.03 %	0.02 %	0.13 %
R^2	0.99	0.99	0.99

The Koopman operator reconstructs the test data matrix from the initial condition. In this test, the initial condition is the first row of the X_{test} in dictionary representation.

The reconstructed test data is compared with the original test data from the simulations in figure 3.19. The two values overlap most of the time except for noisy regions. The result shows that the model accurately predicts the test data with a relative root mean squared error of 2.65 %. The high R^2 value proves that it captures the primal trend of the data. Also, it is observed that there is no phase shift between the prediction and true values.

3.6.2 Scenario 2: a corridor connected to three rooms

The training of the Koopman operator for scenario 2 is similar to that of scenario 1. The number of time delays is set to 1, which creates six new coordinates: 3 coordinates for the measurement data at $t = 1$ and the others for the number of initial pedestrian numbers, which are constant parameter values. The number of diffusion map eigenpairs stays the same as in scenario 1, which yields 70 new coordinates from the eigenvectors of the diffusion operator.

Figure 3.20 compares the predicted and the original values. It is observed that for all three measurement data, the predictions capture the dynamics with low errors. Measurement at the bottom area shows a larger up and down of the number of pedestrians over time, but the prediction follows such nonlinear behavior quite well. Relative root mean squared error shows that all three predictions have a 3 to 5 % error, which is a maximum three pedestrians difference, meaning it is not a significant error. It also shows that predicting values at the bottom measurement area yields a more significant error than the other two. This is because the bottom area involves the most significant complexity of the crowd dynamics due to the incoming pedestrians from the above two areas. R^2 scores show that all three predictions are highly accurate.

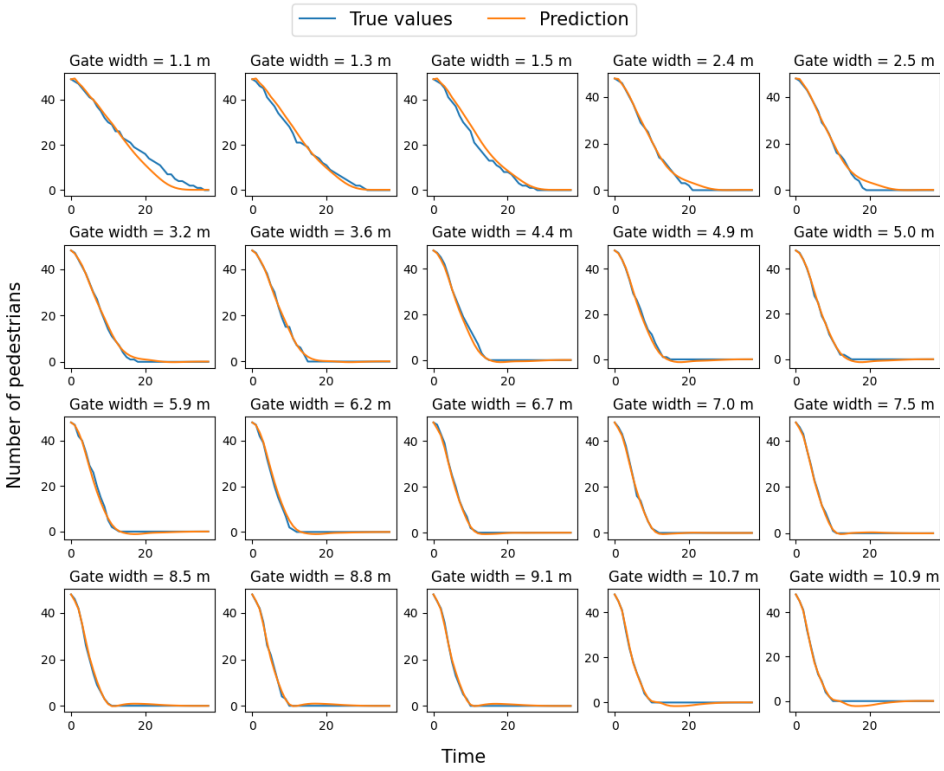


Figure 3.19 Comparison of the measured pedestrian number and prediction in scenario 1. 10 test samples are visualized. Blue lines show the measured values, while the orange lines represent predictions.

3.6.3 Scenario 3: a corridor connected to three rooms with varying gates

The training process of the Koopman operator for scenario 3 is based on the process used in scenario 2. The same parameters for the EDMD are used. Therefore, the number of delays is again 1, which creates seven new columns in this case, as scenario 3 has the width of the gate as an additional parameter. Figure 3.21 shows the prediction results for several cases and the corresponding original values. The results show that gate width is also well parameterized since the gate width does not affect the prediction accuracy compared to the results from scenario 2. Relative root mean square error is also nearly the same as that of scenario 2, which implies that the variable gate width does not degrade the prediction quality in this model.

Table 3.9 Performance evaluation of the model in scenario 3. From the top, Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), Root Mean Square Percentage Error (RMSPE), R square.

Measurement area	Area 1	Area 2	Area 3
MAE	0.75	1.30	1.66
MAPE	2.49 %	3.53 %	4.14 %
RMSE	1.12	1.76	2.16
RMSPE	3.74 %	4.75 %	5.39 %
MBE	0.07	-0.01	0.04
MBPE	0.24 %	-0.04 %	0.10 %
R^2	0.99	0.99	0.99

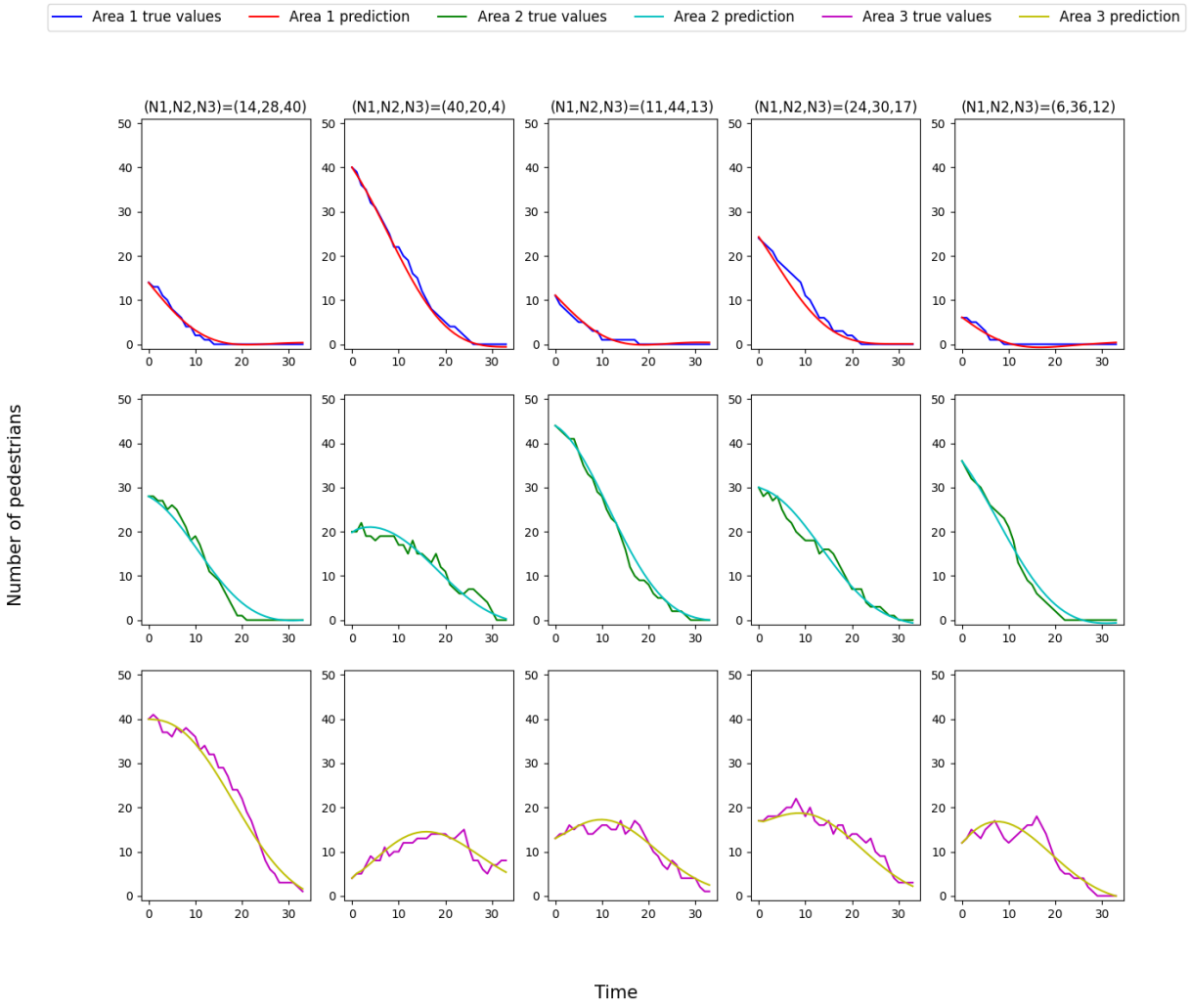


Figure 3.20 Comparison of the measured pedestrian number and prediction in scenario 2. The first row is the values measured in area 1, located on the top of the geometry. The second and third rows are area 2 and area 3, respectively. Area 2 is in the middle of the corridor, and area 3 is the nearest measurement area to the target. N1, N2, N3 represent the initial number of pedestrians in each pedestrian source. The respective prediction values are displayed in the same figures.

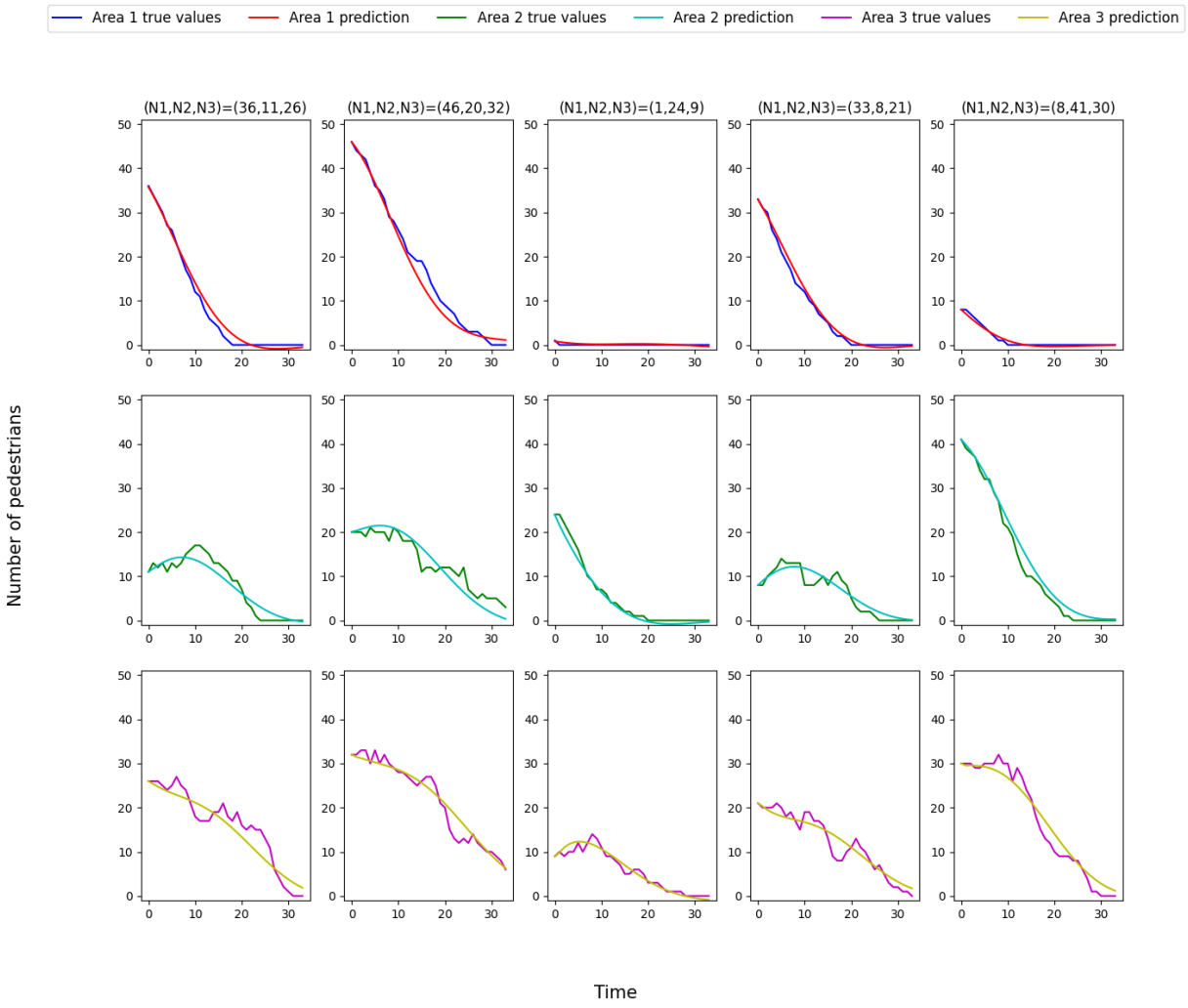


Figure 3.21 Comparison of the measured pedestrian number and prediction in scenario 3. Likewise, in scenario 2, each row represents a measurement area: area 1 in the first row, area 2 in the second, and area 3 in the third. Also, the initial number of pedestrians at each source is represented as N1, N2, and N3, respectively.

Table 3.10 Performance evaluation of the model in scenario 4. From the top, Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), Root Mean Square Percentage Error (RMSPE), R square.

Measurement area	Area 1	Area 2	Area 3	Area 4	Area 5	Area 6
MAE	1.50	2.25	2.80	1.56	2.39	2.84
MAPE	4.30 %	4.90 %	5.49 %	4.09 %	4.78 %	5.27 %
RMSE	2.12	2.94	3.61	2.12	3.13	3.67
RMSPE	6.05 %	6.40 %	7.07 %	5.58 %	6.26 %	6.80 %
MBE	0.00	0.01	-0.03	-0.02	0.04	0.03
MBPE	0.00 %	0.02 %	-0.06 %	-0.05 %	0.08 %	0.06 %
R^2	0.98	0.98	0.99	0.98	0.98	0.99

3.6.4 Scenario 4: a corridor connected to six rooms

The time delay is set to the same number of delays, 1, as the other scenarios. Therefore, the time-delay embedding creates 14 columns of the data matrix. The number of diffusion maps eigenfunctions is set to 80, representing the state space with 80 eigenfunctions. 1000 simulation data are given as a training dataset, and the same number of simulation data, 1000 data, are also sampled with different parameter values for the test. EDMD model is trained on the training dataset, and the Koopman operator is approximated. The model is then given the first two steps as the initial condition due to the one time delay. The operator updates the value in the space created on the Koopman eigenfunctions up to the 34th time step. Then, it reconstructs the original state space to obtain the predicted test data.

The predicted test values are compared with the measured test values from the simulation. Figure 3.22 compares three test cases. Each row represents one measurement area. The parameter values of each test case are displayed at the top of each column. It reveals that the predicted values overlap the actual values overall. It ignores small fluctuations of the values and returns smooth lines, which will approximate the actual behavior of the dynamics. It means that the model is robust to the noise in the data. The dynamics are different in each layer of the measurement area. However, the Koopman operator captures the phase difference well. It results in 6-7 % of RRMSE, meaning that the predictions produce an error of 2-3 pedestrians count on average, which is relatively small considering that the maximum number of pedestrians at one source is 50.

3.7 Model analysis

One of the critical features of the Koopman operator is its building block-like structure of eigenfunctions and corresponding eigenvalues. The Koopman eigenfunctions are vector values that lift the state space into the observable function basis, and the eigenvectors update the state in the observable function basis. Therefore, the eigenfunctions reveal how the Koopman operator maps the state space into the observables and describes it in a linear system.

In this section, the eigenfunctions are investigated to analyze the model. Diffusion maps are selected as a dictionary component in this thesis. Hence, the eigenfunctions from the diffusion maps are also explored. The eigenfunctions are visualized using heatmaps, which show the values along time on the x-axis and parameters on the y-axis.

3.7.1 One Parameter Scenario

Figure 3.23 visualizes two eigenfunctions of the model in scenario 1: the eigenfunctions of the diffusion maps on the top of the figure and the Koopman eigenfunctions on the bottom. The color in the figure shows the magnitude of the values. The Y-axis is the simulations' ID number, representing the parameter

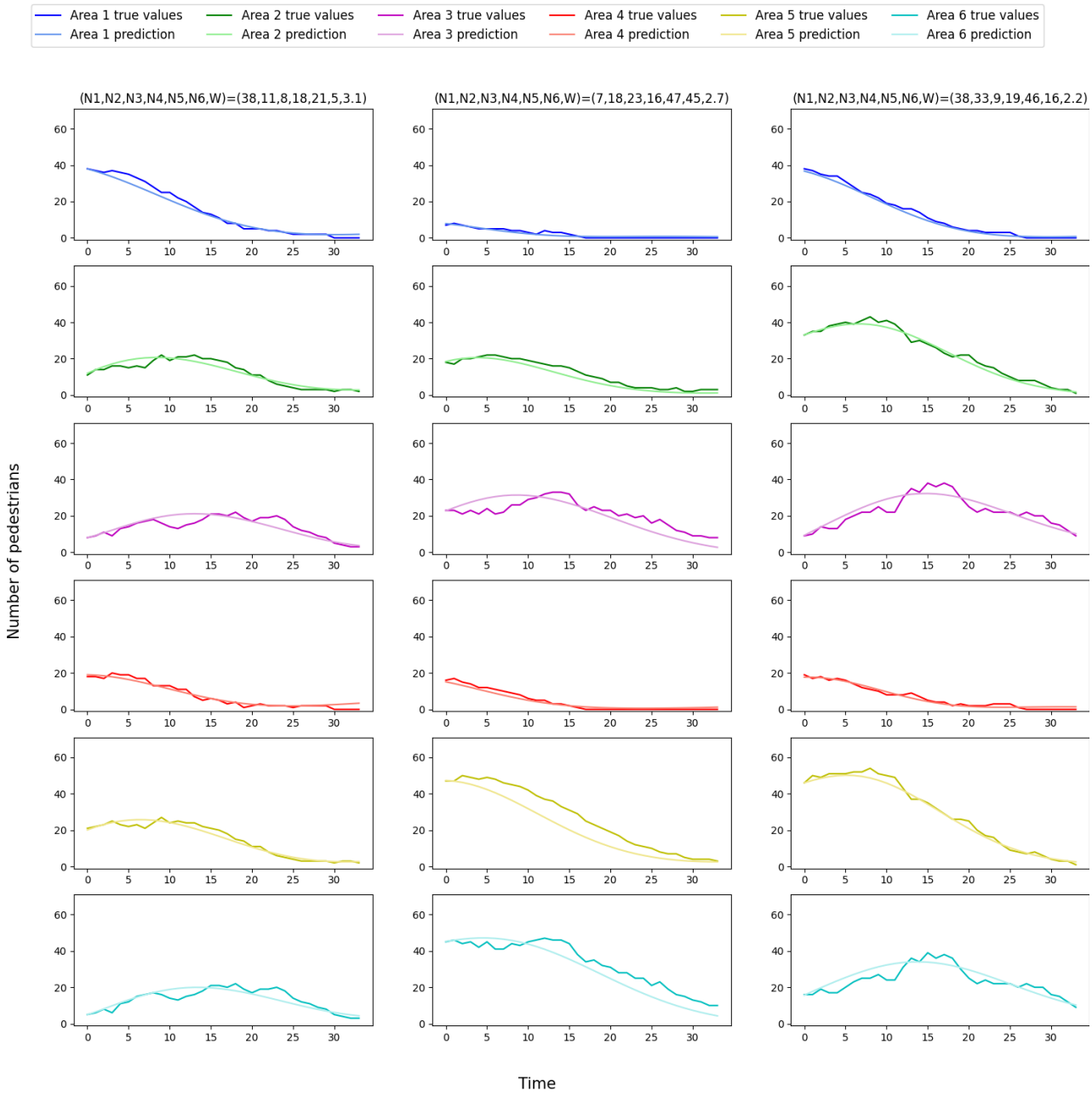


Figure 3.22 Three comparisons of the measured pedestrian numbers and predicted numbers in scenario 4. Each row represents a measurement area: area 1 in the first row, area 2 in the second, area 3 in the third, area 4 in the fourth, area 5 in the fifth, and area 6 in the last row. Also, the initial number of pedestrians at each source is represented as N1, N2, N3, N4, N5, and N6 respectively. The width of the gate is represented by parameter W.

values. Small IDs represent narrower gate widths, and the gate width becomes wider as the number of IDs grows. When ID is 0, the width of the gate is 1.0 m, and when 100, the gate width is 11.0 m.

Diffusion maps arrange the eigenfunctions by the order of the corresponding eigenvalues. Hence, the first eigenfunctions remain for the long term since the eigenvalues are close to 1, and the latter eigenfunctions shrink as time passes by, leaving tiny influences in the long term. Therefore, the first 20 eigenfunctions are displayed in the figure to observe the dominant components. The first eigenfunction is a constant containing no meaningful information about the model. From the second eigenfunction, it is observable that there is a pattern that appears in almost all eigenfunctions. Along the x-axis, there is a point where the value suddenly changes. The point depends on the ID; as the ID grows, the point appears earlier. It draws a form like a logarithmic function in the heatmaps. This pattern corresponds to the trend in data seen in figure 3.13, where small ID simulations show a slow transition of the values, while the larger parameters lead to faster convergence. This reveals that the diffusion maps' eigenfunctions capture a data trend. Furthermore, the eigenfunctions show that these forms have different frequencies. Eigenfunctions ϕ_2 and ϕ_3 have only one point where the value changes. Meanwhile, The values change twice in ϕ_4 and ϕ_5. The number of points increases further in the latter eigenfunctions. The difference in the number of points represents the frequency of the feature that the eigenfunction describes. The first few eigenfunctions have small frequencies, describing a long-term trend of the data. Meanwhile, the latter eigenfunctions contain higher frequencies, meaning that those eigenfunctions represent small fluctuations in the data, such as noises. Moreover, there is a point where the value transitions in the vertical direction in some eigenfunctions, such as $\phi_7, \phi_8, \phi_{10}$. This transition describes another characteristic in the data. The data does not change depending on the width after a certain width, around $width = 6.0$ m. The vertical transition represents where the data becomes independent of the width.

Several Koopman eigenfunctions are selected and displayed at the bottom of the figure. The first row visualizes the real values of the eigenfunctions, and the imaginary values are shown in the second row. Complex conjugates are placed next to each other. Similarly to diffusion maps, eigenfunctions capture the data's intrinsic behaviors. The first two eigenfunctions represent when the values start decreasing in the data. The latter eigenfunctions correspond to the diffusion maps; they describe the repeating patterns in the horizontal direction. Also, different frequencies are visible in $\phi_{30}, \phi_{31}, \phi_{47}, \phi_{48}$. The similarity between the diffusion maps eigenfunctions and the Koopman eigenfunctions is because the derivation of the Koopman eigenfunctions is based on the dictionary as equation 2.27 states.

3.7.2 Multiple Parameters Scenarios

Scenarios 2, 3 and 4 have multiple parameters that affect the dynamics. In order to visualize the eigenfunctions, only one parameter changes, and the other parameters are fixed specific values. Scenario 2 has three dimensions of freedom: the number of generated pedestrians. The parameters for areas 1 and 2 are fixed to 30, and the parameter for area 3 remains variable. ID number corresponds to the number of pedestrians in area 3. For example, if ID is 5, the initial number of pedestrians in area 3 is 5. This way, the variation is limited to one axis, and thereby, the influence of one parameter is observable.

Figure 3.24 shows the eigenfunctions of diffusion maps and eigenfunctions of the approximated Koopman operator. 18 eigenfunctions of diffusion maps are selected and visualized in the top figure. In eigenfunctions $\phi_7, \phi_{22}, \phi_{26}, \phi_{31}$, there are points where the pattern changes in the vertical direction, around ID 20 to 30. The relationship between incoming and outgoing pedestrians causes this. The number of incoming pedestrians surpasses that of outgoing from area 3 for IDs smaller than 30. However, the inequality is reversed for IDs larger than 30. This creates the transition from behaviors with ups and downs to a monotonic decrease of the number measured in area 3. The eigenfunctions capture this transition as the patterns change along the ID number. Additionally, some eigenfunctions, such as eigenfunctions ϕ_5, ϕ_{18}, represent the overall behaviour of the data in the horizontal direction, starting from an initial number, decreasing, and converging to 0. Eigenfunctions also tend to be noisy compared to scenario 1 due to the increased uncertainty created by multiple pedestrian sources.

The Koopman eigenfunctions also describe the same trend, the transition in the vertical direction. It is represented in the eigenfunctions $\phi_{18}, \phi_{47}, \phi_{48}, \phi_{54}, \phi_{55}, \phi_{56}$.

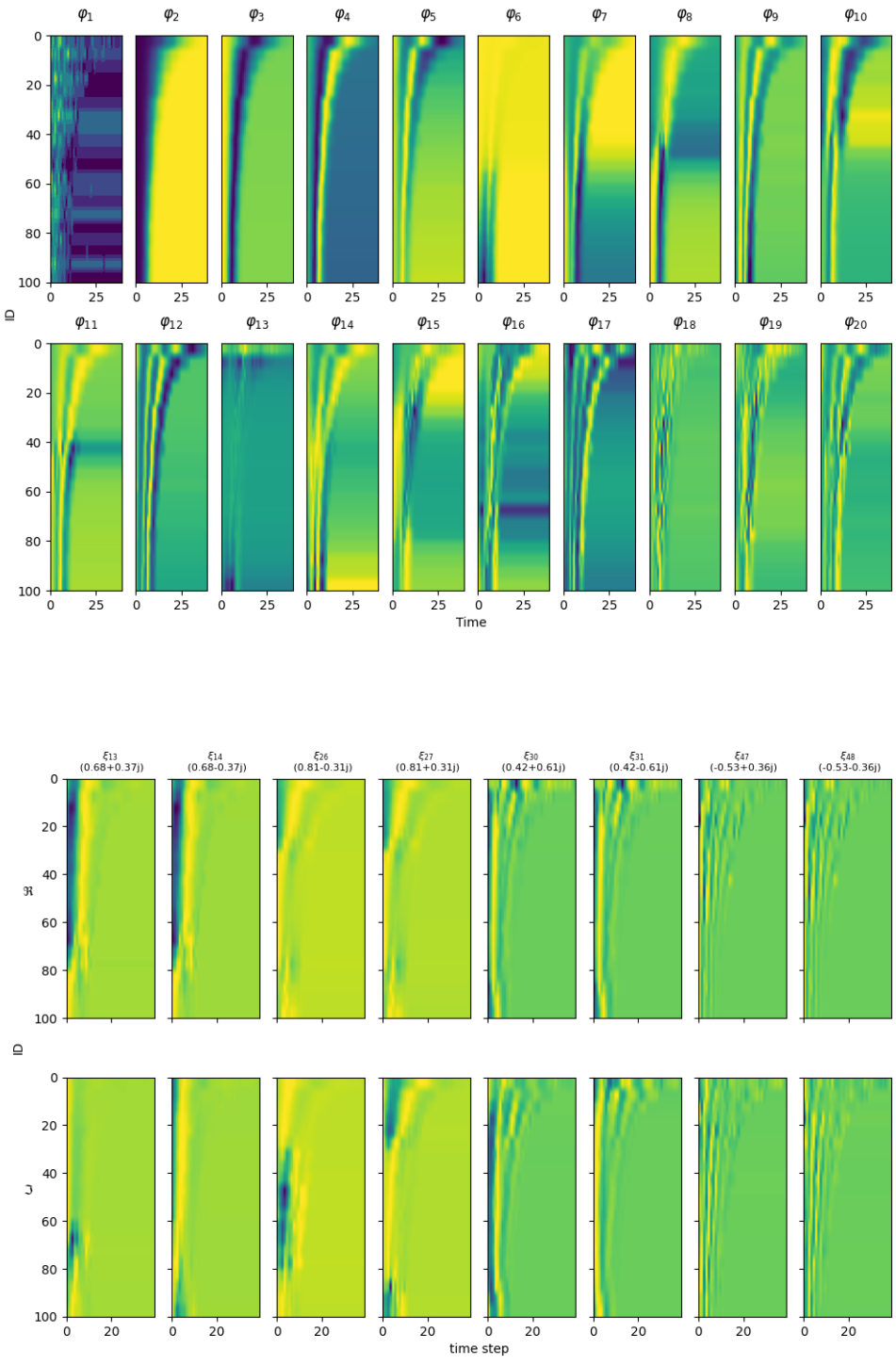


Figure 3.23 Visualization of the diffusion maps eigenfunctions (Top) and the Koopman eigenfunctions (Bottom) of the model in scenario 1. Eigenfunctions are ordered according to the decreasing order of the corresponding eigenvalues. Top: 20 most significant eigenfunctions are displayed. The Y-axis represents the ID number, an identifier for each simulation. In scenario 1, the IDs are ordered along with the value of parameter x. Hence, ID = 0 means x = 0.0, and x = 10.0 when ID = 100. The X-axis shows the time steps. Bottom: The figure shows 8 Koopman eigenfunctions, the real values on the first row and the imaginary values in the second row. Also, the eigenvalues are displayed on the top of each plot.

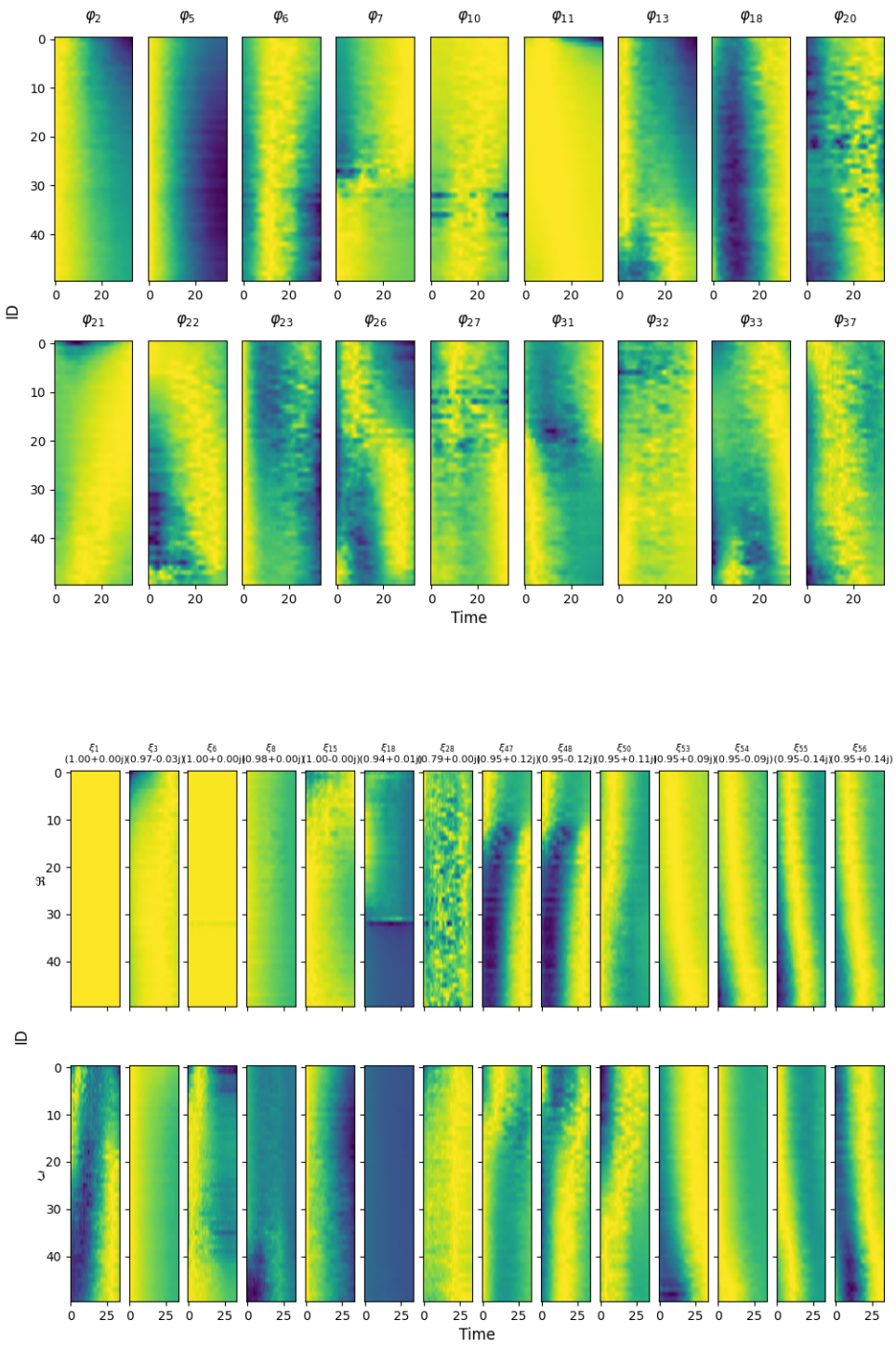


Figure 3.24 Visualization of the diffusion maps eigenfunctions (Top) and the Koopman eigenfunctions (Bottom) of the model in scenario 2 under the condition where the initial condition of two pedestrian sources are fixed. Eigenfunctions are ordered according to the decreasing order of the corresponding eigenvalues. Top: 20 eigenfunctions are selected and displayed. The Y-axis represents the ID number, an identifier for each simulation. The X-axis shows the time steps. Bottom: The figure shows 14 Koopman eigenfunctions, the real values on the first row and the imaginary values in the second row. Also, the eigenvalues are displayed on the top of each plot.

Scenario 3 has four parameters: the width of the gates and the number of generated pedestrians. Therefore, the model is equipped to describe the influence of the width of the gates on the number of pedestrians measured in three measurement areas. The initial number of pedestrians at three sources is fixed at 50 to visualize the effects of the gate width as a parameter in the y-axis. ID represents the gate width from 1.0 m to 5.0 m: If ID is 15, the gate width is 2.5 m. Figure 3.17 shows the eigenfunctions of diffusion maps and Koopman eigenfunctions. The eigenfunctions show a little variance along the y-axis. This is mainly visible in eigenfunctions $\phi_4, \phi_6, \phi_{15}, \phi_{31}, \phi_{32}, \phi_{55}$. These patterns in the vertical direction represent the influence of the variable gate width. Also, the eigenfunctions can be classified into three types: the ones with the highest or lowest values at the beginning, such as ϕ_5, ϕ_9, ϕ_{11}, ϕ_{12} and ϕ_{24}, the ones with the highest or lowest values in the middle of the simulation around $Time = 12$ such as ϕ_{14}, ϕ_{41}, ϕ_{42} and ϕ_{57}, and the ones without any pattern along the time axis such as ϕ_{55}. The first ones represent a monotonic decrease of the values observed mostly in area 1. In contrast, the second ones are observed in areas 2 and 3, where the values once increase due to the incoming pedestrians from higher areas and then start decreasing. The last one does not contain information depending on parameters, that is, in this case, the effect of the variable gate width.

The Koopman eigenfunction in the bottom figure shows similar trends.

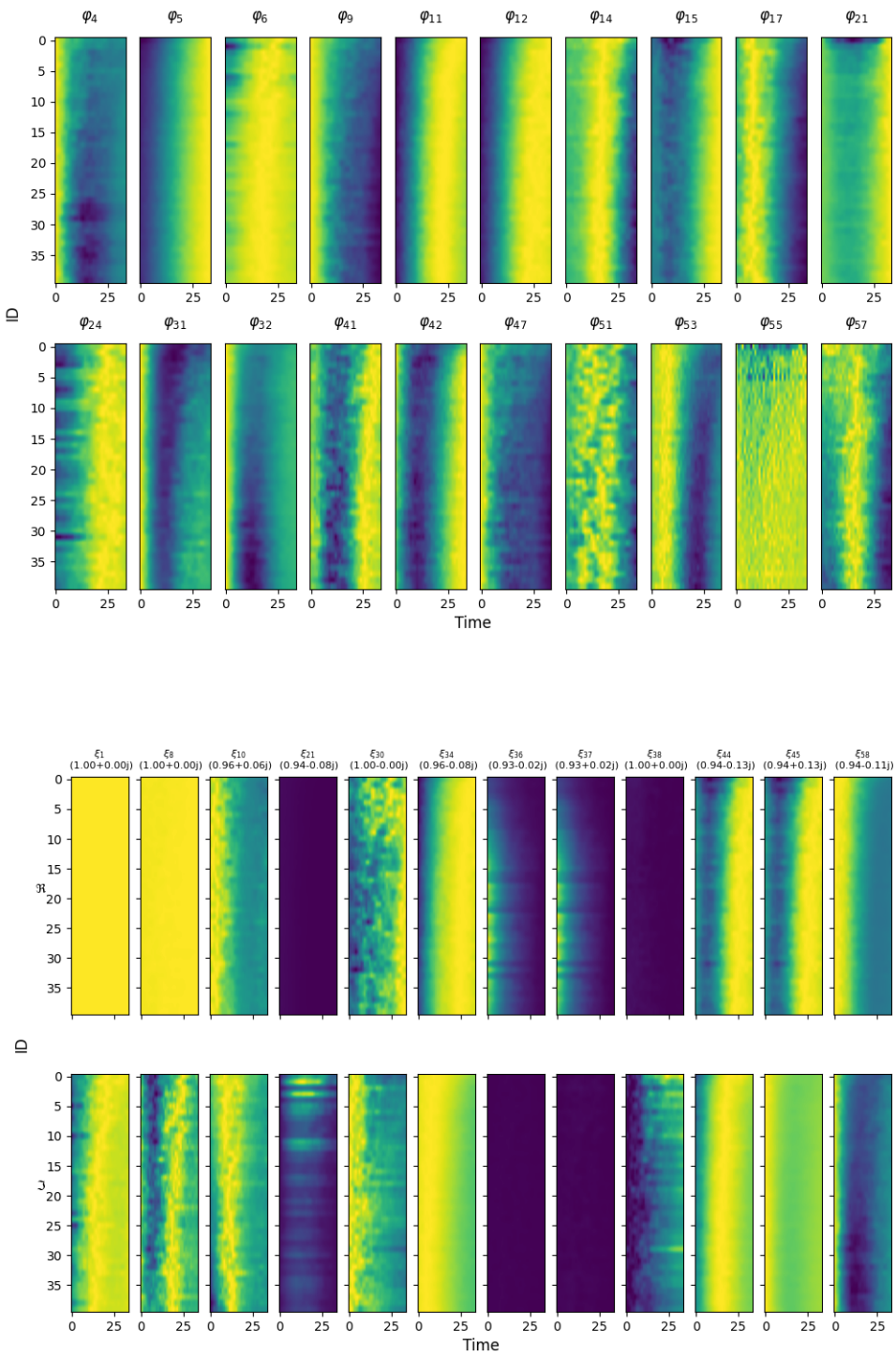


Figure 3.25 Visualization of the diffusion maps eigenfunctions (Top) and the Koopman eigenfunctions (Bottom) of the model in scenario 3 under the condition where the initial number of pedestrians is fixed. Eigenfunctions are ordered according to the decreasing order of the corresponding eigenvalues. Top: 20 eigenfunctions are selected and displayed. The Y-axis represents the ID number, an identifier for each simulation. The X-axis shows the time steps. Bottom: The figure shows 12 Koopman eigenfunctions, the real values on the first row and the imaginary values in the second row. Also, the eigenvalues are displayed on the top of each plot.

4 Conclusions

The primary objective of this thesis is to construct a parameterized Koopman operator specifically for crowd dynamics modeling. The parameterized Koopman operator is designed to capture the intrinsic dynamics of the crowd behavior with variable parameters, including the crowd's size and the environment's geometry. The model's performance is assessed, and further analysis is conducted on the eigen-components of the operator to comprehend the dominant components of the dynamics and the model's ability to represent crowd behaviour.

The Koopman operator acts on observables of the state, which is usually defined in an infinite dimensional space. Infinite dimensionality imposes challenges in computing and using the operator in practice. Several approximation methods have been developed to address this difficulty. One such method is EDMD, characterized by its capability to effectively approximate subsets of Koopman eigenfunctions based on data and dictionary selection. In this thesis, EDMD is selected to compute the approximation of the Koopman operator. EDMD requires data on the crowd dynamics and a dictionary. The data is sampled from crowd simulation results. This thesis exploited four scenarios of different conditions about crowds' geometry and size. Sampled data consists of multiple time series, whose values are the number of pedestrians counted in measurement areas. A dictionary is a set of observable functions that span the space in which the Koopman operator is approximated. It lifts the state space to the space described by observables. This thesis employs a combination of two techniques as a dictionary: Taken's time delay embedding and diffusion maps. Time delay embedding enriches the dataset with time-dependent information by introducing delayed coordinates, extending the spatial dimension. Diffusion maps extract the leading eigen-components of the diffusion operator of the data matrix, yielding new coordinates that encapsulate significant information about the data.

The four scenarios introduce different crowd dynamics with different levels of complexity. The first scenario is designed to demonstrate the simplest crowd behaviour, equipped with a corridor and a gate with variable width, which blocks the flow of the crowd along the corridor, with one source of pedestrians, one target, and one measurement area. The second scenario tests the model's ability to adjust to another type of parameter, the size of crowds, and the capability of dealing with more dimensions. The geometry is extended to have three sources of crowds and three measurement areas covering each source, creating three parameters in total. The third scenario adds one more degree of freedom, the gate width, to the second scenario. The last scenario expands it further, having six pedestrian sources, six measurement areas, and a variable gate width, increasing the dimensions to test the model's capability. The model predictions result in high accuracy, with R^2 scores close to 1 in all scenarios, meaning that the model fits quite well to the ground truth.

Scenario 1 demonstrates the model's ability to learn monotonic tendencies in data, revealing that the model adjusts the decreasing rate of the pedestrians in the measurement area by the parameter values. Monotonic increase or decrease is simple and can be regressed by a linear model. Therefore, the other scenarios integrate more variability in the data by having multiple pedestrian sources, requiring modeling of the nonlinear curves. The results in scenarios 2 and 3 show that the model can capture the nonlinear behavior of the dynamics with high accuracy. Moreover, it is robust to the noise, as it follows the long-term trend of the data even though the data contains small fluctuations. Scenario 4 examines the model's ability to increase dimensions. The number of samples is the same as in scenarios 2 and 3, meaning the data matrix becomes sparser. Although a slight drop in accuracy is observed from the statistical analysis, the model can still provide accurate forecasting in all dimensions.

The operator's eigenfunctions, the model's building blocks, are analyzed further in depth. The heat maps of selected eigenfunctions displayed that the eigenfunctions preserve intrinsic dynamics patterns. The patterns indicate the transition of the states, such as the beginning of an increase and decrease in

the measured values. Some patterns have some frequencies; Significant eigenfunctions have lower frequencies, implying that the major trends are preserved for the long term. Small fluctuations are considered noise and diminished after several time steps, multiplied by small eigenvalues. Hence, the eigenfunctions capture the overall tendencies of the dynamics over time and extract the characteristics.

This thesis shows that the parameterized Koopman operator can model crowd dynamics, which involves several variable factors, such as geometrical structures and the conditions of the crowds. Successful prediction of multi-variable situations such as scenario 4 tells that the models can be further extended to more complicated scenarios, where more variables are involved in a larger space. It proves that the parameterized Koopman operator models can be applied to the research in city planning in the future thesis. Additionally, the Koopman operator models offer an in-depth analysis of the crowd dynamics in city planning. It enables a better understanding of the tendency of crowd behavior in the city in the long-term and short-term from the Koopman eigenfunctions. Deep learning-based models are attracting attention in this field; however, the significant drawback of deep learning models is their "black-box" nature, which makes it difficult to root out the cause of the outcomes. The parameterized Koopman operator can supplement the lack of interpretability of deep learning-based models, providing insights into the models and allowing humans to comprehend the prediction-making mechanisms.

However, this thesis is still limited to restricted and ideal scenarios. In reality, it deals with numerous variables which determine crowd behaviors. Future research is supposed to extend the scenarios to be more complicated, with an increasing number of variables. Furthermore, utilising an actual city map as the simulation geometry would provide more realistic modeling of the crowd dynamics in a city.

Bibliography

[1] Xueyan Yin, Genze Wu, Jinze Wei, Yanming Shen, Heng Qi, and Baocai Yin. A comprehensive survey on traffic prediction. *arXiv preprint arXiv:2004.08555v2*, 2020.

[2] Attila Matyas Nagy and Vilmos Simon. Survey on traffic prediction in smart cities. *Pervasive and Mobile Computing*, 50:148–163, 2018.

[3] Masoomeh Zameni, Mengyi He, Masud Moshtaghi, Zahra Ghafoori, Christopher Leckie, James C. Bezdek, and Kotagiri Ramamohanarao. Urban sensing for anomalous event detection: Distinguishing between legitimate traffic changes and abnormal traffic variability. In *Machine Learning and Knowledge Discovery in Databases*, pages 553–568. Springer, Cham, 2019.

[4] Moussaïd Mehdi and Nelson Jonathan D. Simple heuristics and the modelling of crowd behaviours. In *Pedestrian and Evacuation Dynamics*, pages 75–90. Springer, Cham, 2014.

[5] Saeed R.A., Recupero Diego Reforgiato, and Remagnino Paolo. Modelling group dynamics for crowd simulations. *Personal and Ubiquitous Computing*, 26:1299–1319, 2022.

[6] Zhai Zhongyi, Liu Peipei, Zhao Lingzhong, Qian Junyan, and Cheng Bo. An efficiency-enhanced deep learning model for citywide crowd flows prediction. *International Journal of Machine Learning and Cybernetics*, 12:1897–1891, 2021.

[7] Marco Cardia, Massimiliano Luca, and Luca Pappalardo. Enhancing crowd flow prediction in various spatial and temporal granularities. *WWW '22 Companion: Companion Proceedings of the Web Conference 2022*, 2022.

[8] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data-driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(5):1307–1346, 2015.

[9] Benedikt Kleinmeier, Benedikt Zönnchen, Marion Gödel, and Gerta Köster. Vadere: An open-source simulation framework to promote interdisciplinary understanding. *Collective Dynamics*, 4:1–36, 2019.

[10] B. O. Koopman. Hamiltonian systems and transformations in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[11] Alessandro Alla and J. Nathan Kutz. Nonlinear model order reduction via dynamic mode decomposition. *arXiv preprint arXiv:1602.05080*, 2016.

[12] Marko Budišić, Ryan Mohr, and Igor Mezić. Applied koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510, 2012.

[13] Ian Abraham, Gerardo De La Torre, and Todd D. Murphey. Model-based control using koopman operators. *Robotics: Science and Systems Proceedings*, 2017.

[14] Stefan Klus, Feliks Nüske, and Sebastian Peitz. Koopman analysis of quantum systems. *Quantum Physics (quant-ph)*, 2022.

[15] Yoshihiko Susuki, Igor Mezic, Fredrik Raak, and Takashi Hikihara. Applied koopman operator theory for power systems technology. *Nonlinear Theory and Its Applications, IEICE*, 7(4):430–459, 2016.

[16] Mustaffa Alfatlawi and Vaibhav Srivastava. An incremental approach to online dynamic mode decomposition for time-varying systems with applications to eeg data modeling. *arXiv preprint arXiv:1908.01047*, 2019.

[17] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656(10):5–28, 2010.

[18] P. J. Schmid, L. Li, M. P. Juniper, and O. Pust. Applications of the dynamic mode decomposition. *Theoretical and Computational Fluid Dynamics*, 25:249–259, 2011.

[19] Eurika Kaiser, J. Nathan Nutz, and Steven L. Brunton. Data-driven discovery of koopman eigenfunctions for control. *arXiv preprint arXiv:1707.01146*, 2017.

[20] Igor Mezić. On numerical approximations of the koopman operator. *Mathematics*, 10(7):1180, 2020.

[21] Qianxiao Li, Felix Dietrich, Erik M. Bollt, and Ioannis G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: a data-driven adaptive spectral decomposition of the koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10):103111, 2017.

[22] Hiroaki Terao, Sho Shirasaka, and Hideyuki Suzuki. Extended dynamic mode decomposition with dictionary learning using neural ordinary differential equations. *Nonliear Theory and Its Application, IEICE*, 12(4):626–638, 2021.

[23] Daniel Lehmberg, Felix Dietrich, and Gerta Köster. Modeling melburnians - using the koopman operator to gain insight into crowd dynamics. *ELSEVIER*, 2021.

[24] Ronald R. Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21:5–30, 2006.

[25] Lehmberg Daniel, Dietrich Felix, and Köster Gerta. datafold: data-driven models for point clouds and time series on manifolds. *Journal of Open Source Software*, 5(51):2283, 2020.