Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

A TOPOLOGICAL DATA ANALYSIS APPROACH TO DETECTING CONGESTION IN PEDESTRIAN CROWDS

Naoyuki Kamiyama[1], Hiroaki Yamada[2], Takashi Kato[2], Shizuo Kaji[1], and Tetsuro Takahashi[2]

[1]Institute of Mathematics for Industry, Kyushu University, Fukuoka, JAPAN
[2]Fujitsu Limited, Kawasaki, JAPAN

ABSTRACT

This paper addresses the challenge of detecting heavy congestion in pedestrian crowds. While congestion detection is straightforward when monitoring predefined locations through pedestrian density measurements, identifying potential congestion areas in advance remains problematic. We propose a novel algorithm that eliminates the need for prior knowledge of congestion-prone locations by topological data analysis. Our approach transforms time-series pedestrian position data into topological representations, enabling more robust congestion detection. We validate our algorithm using both synthetic data from multi-agent simulations and real-world pedestrian measurements. The experimental results demonstrate that converting traditional positional data into topological representations significantly improves the performance of machine learning models in congestion detection tasks.

1 INTRODUCTION

In populated urban areas such as sports stadiums, amusement parks, and theaters, crowd accidents pose a significant safety risk (see, e.g., (Feliciani et al. 2022, Appendix A) and (Helbing et al. 2002, Table 1) for examples of crowd accidents). While these accidents can have multiple causes, they frequently occur in heavily congested areas of pedestrian crowds, which we term *clusters*. Detecting such clusters before they lead to fatal accidents is crucial for public safety.

Traditional approaches to cluster detection rely on monitoring predefined locations by measuring pedestrian density. However, this method requires prior knowledge of potential congestion points, which is not always feasible. Additionally, many existing algorithms that assess crowd risk using velocity measurements (see, e.g., (Feliciani and Nishinari 2018; Huang et al. 2015)) require person matching (or re-identification) in time-series data, raising privacy concerns and computational challenges.

We propose a novel algorithm that detects clusters without requiring advance knowledge of congestion-prone locations or individual pedestrian tracking (Figure 1). Our approach relies solely on pedestrian position data, independent of the spatial configuration. Using *topological data analysis* (see, e.g., (Buchet et al. 2018; Hiraoka et al. 2016)), we transform snapshots of pedestrian position data, namely point cloud data, into topological representations to identify cluster formation. This method builds on the assumption that congestion states can be encoded in pedestrian position patterns, which we quantify using persistent homology.

By using persistent homology as a feature vector, it is expected to improve prediction performance in various classification and regression tasks where point cloud data is used as input (Pun et al. 2022). Persistent homology is invariant to geometric and scale transformations. Therefore, it has the potential to improve extrapolation performance in some cases. In addition, since persistent homology only extracts topological features without relying on the detailed arrangement of data points, it has the potential to handle both noise-free simulation data and noisy real data without distinction. In other words, it can help to adapt a model trained in simulation to the real environment. The contribution of this paper is the demonstration that using persistent homology as a feature vector improves extrapolation performance and simulation to real

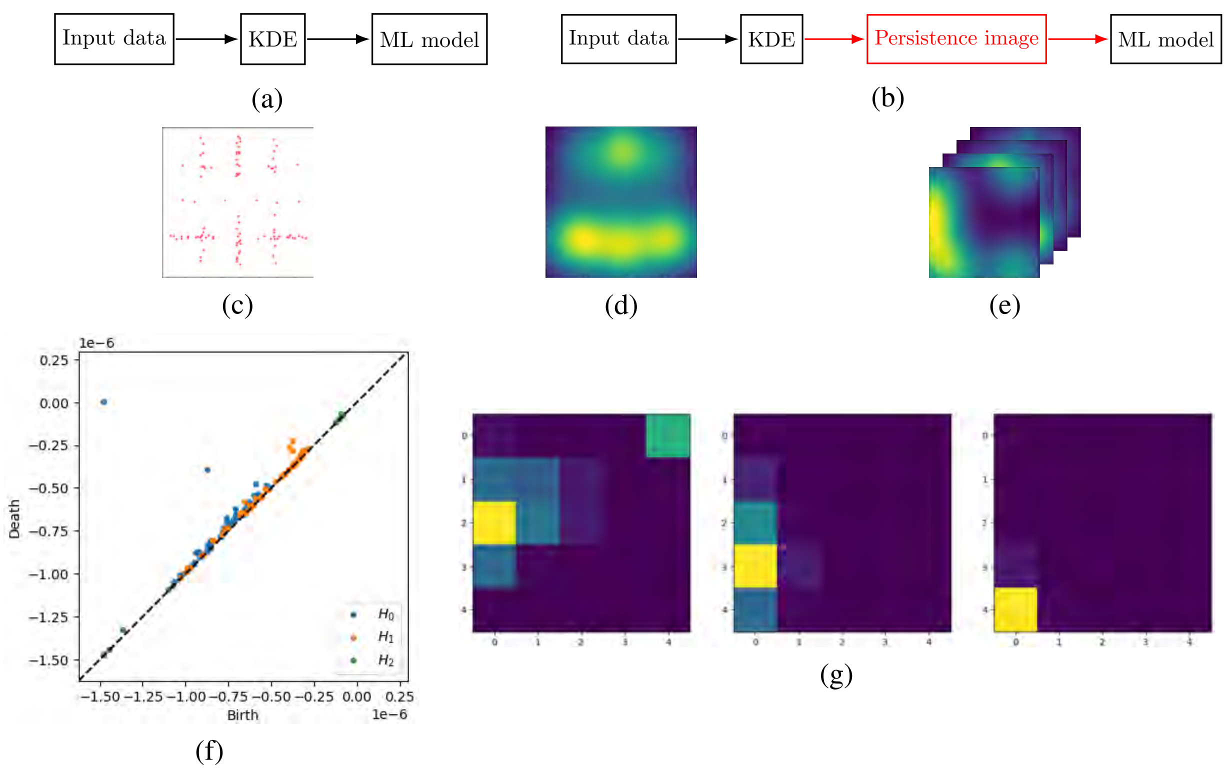

Figure 1: Topological feature learning. (a) The baseline algorithm (w/o topological feature extraction), (b) The proposed algorithm (w/ topological feature extraction). (c)-(g) is visualization of the proposed algorithm data processing steps: (c) Instantaneous pedestrian positions in spatial coordinates, (d) Two-dimensional density map generated through kernel density estimation, (e) Three-dimensional spatiotemporal representation created by chronological stacking of density maps, (f) Persistence diagram showing topological features and their lifetimes, and (g) Persistence image derived from the persistence diagram.

(Sim2Real) performance. To the best of our knowledge, there is no other study that has applied persistent homology to pedestrian cloud point data and demonstrated its effectiveness.

Validating benefits of our algorithm requires extensive pedestrian crowd data. However, collecting large-scale real-world data presents significant challenges, particularly for crowd accidents. To address this limitation, we employ a *multi-agent simulation* to generate comprehensive dataset. Our simulator produces diverse crowd scenarios with precise spatial and temporal resolution, while incorporating realistic pedestrian behaviors. Specifically, we implement detailed friction modeling, as friction effects are known to contribute significantly to congestion during evacuation and panic situations (Frank and Dorso 2011; Sticco et al. 2020). To complement our synthetic data, we also validate the algorithm using real-world measurements from the Pedestrian Dynamics Data Archive.

The remainder of this paper is structured as follows. Section 2 formally defines our problem, including input data specifications and objectives. Section 3 presents our topological data analysis approach. Section 4 presents the multi-agent simulation framework used to generate synthetic dataset. Section 5 presents experimental results from both synthetic and real-world datasets. (The code is available on request from the authors.) Finally, Section 6 concludes the paper.

2 PROBLEM FORMULATION

We formalize our problem setting as follows. An *instance* represents a time series of pedestrian positions captured at fixed time intervals (see Figure 2). At each time step, we record a *snapshot* containing only the spatial coordinates of pedestrians, without tracking individual identities. Thus, an instance consists of an ordered sequence of snapshots over a predefined time period. The input to our problem is a set of such instances, where each instance may contain different numbers of pedestrians. Each instance is labeled either True or False, indicating the presence or absence of congestion, respectively. Specifically, an instance is labeled True if at least one pedestrian fails to reach their destination within a predefined time limit, indicating congestion. Our objective is to develop an algorithm that can accurately predict the label of a previously unseen instance. The core data set consists of multiple instances with different numbers of pedestrians and different compositions of departure and destination.

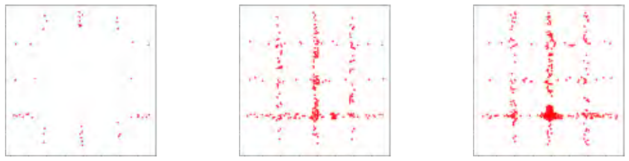
Figure 2: Example instance showing a series of snapshots. Red points indicate pedestrian positions.

In the synthetic dataset, we simulate pedestrian flows in a grid network (see Figure 3) over 300,000 time steps, with each step corresponding to 0.001 seconds. While each pedestrian has a designated destination they attempt to reach within a predefined time limit, our algorithm operates without knowledge of these start points or destinations. Importantly, we exclude several traditional data elements: Individual pedestrian identifiers (IDs), Spatial information about the environment (e.g., dimensions, geometry), and Predetermined congestion points or high-risk areas. Each instance in the synthetic dataset is labeled either True or False, where a True label indicates that at least one pedestrian failed to reach their destination within the specified time limit. The synthetic dataset is partitioned into: Training data: Used to build the model, Validation data: Used to select hyperparameters, Test data: Used to evaluate the model.

In the real-world dataset, similarly, each instance is labeled either True or False, where a True label indicates that at least one pedestrian failed to reach their destination within the specified time limit. We use the leave-one-out method for the experiments with the real data set because the size of the dataset is small. The leave-one-out method does not have validation data.

The primary objective is to develop a model using the training and validation data that can accurately predict labels for previously unseen instances in the test data.

3 ALGORITHMS

In this section, we present our topological data analysis framework for detecting congestion in pedestrian crowds. Our approach processes time-indexed point cloud data representing pedestrian positions to identify congestion patterns. We first describe a baseline algorithm using traditional density-based methods, which serves as a comparison benchmark in our synthetic data experiments. We then introduce our novel topological approach, which leverages persistent homology to capture the geometric and topological features of crowd dynamics.

3.1 Baseline Algorithm

Our baseline algorithm comprises three main steps (see Figure 1(a)): data normalization, density transformation, and model learning. The first two steps transform the raw input data into a format suitable for machine learning, while the final step performs the actual congestion detection.

3.1.1 Step 1: Data Normalization

To ensure consistent analysis, we first normalize the spatial coordinates of pedestrian positions at each time step. We sample the positions every 50 time steps (equivalent to 0.05 seconds) to reduce computational complexity while maintaining meaningful temporal resolution. Since pedestrians move an average of 0.0625 units per 50 time steps, we normalize all coordinates by this factor to achieve unit movement between consecutive samples.

3.1.2 Step 2: Density Transformation

We apply kernel density estimation (KDE) to each sample using an isotropic Gaussian kernel with a bandwidth of 0.5. This transforms the discrete point cloud data into continuous two-dimensional density images (see Figures 1(c) and (d)). These 2D density maps are then stacked chronologically to create a three-dimensional spatiotemporal representation (Figure 1(e)).

3.1.3 Step 3: Model Learning

For the learning component, we employ the separable 3D CNN model proposed by (Xie et al. 2018). This architecture is particularly well-suited for processing our three-dimensional spatiotemporal density representations.

3.2 Proposed Algorithm

Our proposed algorithm extends the baseline approach with an additional topological analysis step (see Figure 1(b)). While the first, second, and fourth steps remain identical to the baseline algorithm, we introduce a crucial topological data analysis component in the third step.

3.2.1 Topological Feature Extraction

The key innovation of our approach lies in the computation and vectorization of topological features. For each instance, we:

1. Generate a persistence diagram from the three-dimensional density image using Cubical Ripser (Kaji et al. 2020) (see Figure 1(f)). Each point in this diagram represents a topological feature ("hole") in the density field, characterized by its birth and death density values along the x and y axes, respectively. For a comprehensive treatment of persistence diagrams, we refer readers to (Edelsbrunner and Harer 2022).
2. Transform the persistence diagram into a fixed-length vector (75 dimensions in our implementation) using the persistence image technique (Adams et al. 2017) (see Figure 1(g)). This transformation effectively creates a two-dimensional histogram of the topological features, providing a stable vectorial representation suitable for machine learning.

3.2.2 Model Learning

For the learning component, we employ multiple machine learning architectures:

- Random forest (Breiman 2001)
- Multilayer perceptron with two hidden layers (MLP) (Rumelhart et al. 1986)
- Two-layered convolutional neural network (CNN) (LeCun et al. 1998)
- Separable 3D CNN (3D CNN) (Xie et al. 2018)

The objective is to investigate the impact of topological feature extraction on model performance, both for architectures that typically depend on manually constructed features (random forest) and for architectures that are generally considered less reliant on pre-processing (MLP, CNN, and 3D CNN). The 3D CNN is a neural network designed to automatically capture both spatial and temporal features, the CNN is designed to capture spatial features, and the MLP is a simple neural network without any special tricks.

4 SIMULATOR

The simulator used in this paper focuses on finely emulating a pedestrian-pedestrian friction and a pedestrian-object friction. This is because this kind of friction has been considered as one of the causes of congestion in panic or evacuation situations via “blocking clusters” (Parisi and Dorso 2005; Parisi and Dorso 2007). A blocking cluster is a human cluster that extends from a wall to a wall or from an obstacle to a wall. In a human cluster, it is extremely difficult for pedestrians to move due to friction with neighbors. Thus, blocking clusters in front of exit doors or narrow places clearly cause congestion. In addition, a blocking cluster is considered as a cause of the “faster is slower” effect (Frank and Dorso 2011). The faster is slower effect is that the faster pedestrians try to escape, the slower pedestrians get out from room (see also (Sticco et al. 2020)).

The simulator used in this paper was implemented by using the Python library `cromosim` for microscopic crowd motion simulation. This library is based on the well-known social force model (Helbing et al. 2000; Helbing and Molnár 1995). In the library `cromosim`, the following three forces are implemented. The first one is desire force, which leads agents to their final destinations. Roughly speaking, desire force is attraction force determined by the difference of the direction to the destination and the current velocity vector. The second one is social force, which arises between agents or between an agent and an obstacle. Basically, social force is repulsion force depending on the distance between the agents or between the agent and the obstacle. The last one is granular force, which is friction arising between pedestrians. Roughly speaking, granular force depends on the difference of the current velocity vectors of the pedestrians.

In our experiments, each parameter of the simulator basically follows that in (Frank and Dorso 2011). In addition to `cromosim`’s default three forces, we implemented friction between a pedestrian and a wall. This force is calculated in the same way as granular force between pedestrians by setting the velocity of a wall to be 0. The reason why we implemented friction between a pedestrian and a wall is that it creates blocking clusters.

5 NUMERICAL EXPERIMENTS

In this section, we evaluate the validity of the proposed algorithm by using synthetic dataset generated by the simulator in Section 4 and real-world dataset. First, we explain the parameters used in the experiments. Then we show the results for experiments with synthetic dataset. Finally, we show the results for an experiment with real-world dataset.

5.1 Parameters

The proposed algorithm was implemented by Python with the following existing packages.

5.1.1 Kernel Density Estimation

We used `gaussian_kde` of `scipy.stats` for the kernel density estimation. The window size of kernel density estimation was fixed as 0.5.

5.1.2 Topological Feature Extraction

We used `PersistenceImager` in `persim` for computing a persistence image. We multiply the values in the result of kernel density estimation by -1 to flip the filtration vertically. The parameters of `PersistenceImager` are set as follows.

- `pixel_size` in such a way that the size of the persistence image becomes 50×50 for each dimension $d \in \{0, 1, 2\}$. This resolution was selected to balance computational tractability with sufficient granularity to capture topological features.

- `birth_range` and `pers_range` to be the interval from 0 to the 99 percentile of the results of kernel density estimation for the training data. We chose the 99th percentile rather than the maximum to exclude potential outliers that could skew the persistence image representation. The choice of 99% captures nearly all topological features while maintaining robustness against extreme values.
- `sigma` of `kernel_params` to be 10^{-8} times the range of `birth_range` and `pers_range`. This small value ensures that the Gaussian kernels provide appropriate smoothing without over-blurring the topological structure. The scaling relative to the birth/persistence range ensures the parameter adapts appropriately to different datasets.

Since the definition of `pixel_size` implies that the size of the persistence image becomes 50×50 for each dimension $d \in \{0,1,2\}$, the total size of the persistence image is $50 \times 50 \times 3 = 7500$.

5.1.3 Model Learning

In the baseline algorithm, as a learning model, we use the separable 3D CNN (S3D) (Xie et al. 2018). We use the implementation of S3D in `torchvision` (version 0.19.1+cu124).

In the proposed algorithm, we use the random forest (RF) (Breiman 2001), the multilayer perceptron with two hidden layers (MLP) (Rumelhart et al. 1986), and the two-layered Convolutional neural network (CNN) (LeCun et al. 1998) as a learning model in addition to S3D. We use the implementation of RF in `sklearn` (version 1.5.2). Furthermore, we use the implementations of MLP and CNN in `pytorch` (version 2.4.1+cu124). The first linear layer of the MLP receives a flattened vector of KDE features and maps it to 100 hidden units. The second linear layer maps the 100 hidden units to 2 output units corresponding to the classification targets. The CNN consists of two convolutional layers followed by three fully connected layers for classification. Each convolutional layer is followed by a ReLU activation and a max pooling function; a dropout function is applied after the second convolutional layer. The three fully connected layers map the output of the convolutional layers to 120 hidden units, then from 120 to 84, and from 84 to 2 output units. The models were trained using a standard supervised learning procedure with cross-entropy loss. Details of the dataset and the splitting into train, test, and validation data are described in the following section.

5.2 Extrapolation on Synthetic Dataset

In this subsection, we show the results of extrapolation analysis on synthetic data generated by the simulator in Section 4. We examine whether persistence image improves extrapolation performance in rotation setting (**Dataset 1**) and rotation + reflection setting (**Dataset 2**). In the former dataset, the test data is rotationally symmetric with respect to the train/validation data, in the latter dataset, the test data is both rotationally and mirror symmetric with respect to the train/validation data. The datasets are carefully designed so that the train/validation data does not include any of its own rotational symmetry or mirror symmetry. Congestion occurs in some instances, but not in others. The trained model predicts whether congestion will occur or not from pedestrian point cloud data.

5.2.1 Dataset

The simulator used in this paper was implemented by using the Python library `cromosim` (see Section 4). Recall that, in our experiments, each parameter of the simulator basically follows that in (Frank and Dorso 2011). In addition to `cromosim`'s default three forces, we implemented friction between a pedestrian and a wall.

The geometry of this experiment is the grid network in Figure 3. The network has twelve entrances, i.e., left 1, left 2, left 3, right 1, right 2, right 3, up 1, up 2, up 3, down 1, down 2, and down 3. The network consists of three vertical corridors and three horizontal corridors connecting entrances. (For example, there exists a corridor connecting the entrance left 1 and the entrance right 1.) The width of each corridor is

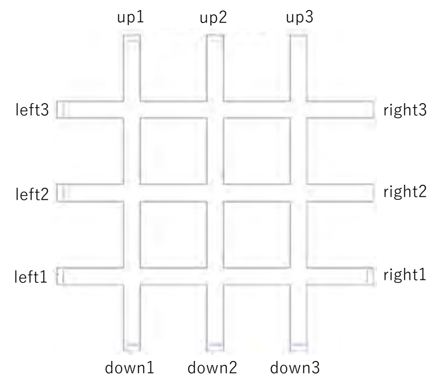

Figure 3: The geometry in the experiment with synthetic dataset.

4.0, and the length of each corridor is 76.0. Vertical corridors and horizontal corridors cross each other in such a way that corridors are divided into four equal parts.

The start point of each agent is one of the entrances. Then the destination of each agent is determined in such a way that the agent go straight through the corridor. That is, if an agent starts from the entrance left 1 (resp. up 1), then its destination is right 1 (resp. down 1). Conversely, if an agent starts from the entrance right 1 (resp. down 1), then its destination is left 1 (resp. up 1). On average, a pedestrian moves 0.0625 in 50 time steps.

The timing when an agent starts is determined as follows. The predefined time interval of our experiments is 300 seconds. In the time interval from the time step 0 to the time step 150, an agent uniform randomly starts during this interval. On the other hand, in the time interval after the time step 150, no agent starts. The position in the entrance from which an agent starts is uniform randomly determined in a small rectangle near the entrance.

In each instance of the experiments, for each $i \in \{1,2,3\}$, the number of pedestrians starting from the entrance left i (resp. up i) and the number of pedestrians starting from the entrance right i (resp. down i) are the same. Let LR_i (resp. UD_i) denote the number of pedestrians starting from the entrance left i (resp. up i) in an instance for each $i \in \{1,2,3\}$.

The training data consists of 200 instances, and the validation data consists of 50 instances. They are constructed as follows. For each integer $i \in \{2,3\}$, $\mathrm{LR}_i = 20$. In addition, $\mathrm{LR}_1 = 100$. Thus, in an instance of these datasets, congestion happens in the corridor connecting the entrance left 1 and the entrance right 1. Furthermore, for each $i \in \{1,2,3\}$, we set UD_i as follows.

- We prepare one instance such that $\mathrm{UD}_i = 50$ for every $i \in \{1,2,3\}$.
- For each $i \in \{1,2,3\}$ and each $m \in \{60,70,\dots,130\}$, we prepare the following instance.
  - $\mathrm{UD}_i = m$.
  - For each $j \in \{1,2,3\} \setminus \{i\}$, $\mathrm{UD}_j = 50$.

In our experiments, the ratio of instances with the label True in the training (resp. validation) data is 0.735 (resp. 0.74).

We prepare two test datasets. Each dataset consists of 250 instances. These datasets are constructed in such a way that the places where congestion happens are different from those in the test data and the validation data. That is, the datasets are constructed in such a way that they become extrapolation data for the training data and the validation data. In our problem, this means that the algorithm cannot use the knowledge about where congestion happens in the training data and the validation data.

Dataset 1. For each $i \in \{1,2\}$, $\mathrm{UD}_i = 20$. Furthermore, $\mathrm{UD}_3 = 100$. Thus, in this dataset, congestion happens in the corridor connecting the entrance up 3 and the entrance down 3. For each $i \in \{1,2,3\}$, we set LR_i as follows.

- We prepare one instance such that $\mathrm{LR}_i = 50$ for every $i \in \{1,2,3\}$.
- For each $i \in \{1,2,3\}$ and each $m \in \{60,70,\dots,130\}$, we prepare the following instance.
  - $\mathrm{LR}_i = m$.
  - For each $j \in \{1,2,3\} \setminus \{i\}$, $\mathrm{LR}_j = 50$.

The ratio of instances with the label True is 0.788.

Dataset 2. For each $i \in \{2,3\}$, $\mathrm{UD}_i = 20$. Furthermore, $\mathrm{UD}_1 = 100$. Thus, in this dataset, congestion happens in the corridor connecting the entrance up 1 and the entrance down 1. Furthermore, for each $i \in \{1,2,3\}$, we set LR_i as follows.

- We prepare one instance such that $\mathrm{LR}_i = 50$ for every $i \in \{1,2,3\}$.
- For each $i \in \{1,2,3\}$ and each $m \in \{60,70,\dots,130\}$, we prepare the following instance.
  - $\mathrm{LR}_i = m$.
  - For each $j \in \{1,2,3\} \setminus \{i\}$, $\mathrm{LR}_j = 50$.

The ratio of instances with the label True is 0.764.

5.2.2 Result

The first table (resp. second table) in Table 1 shows the results for the experiments with the dataset 1 (resp. dataset 2). Figure 4 shows the learning curve. Notice that the accuracy of the baseline algorithm is less than the ratio of the instances with the label True in the test data. On the other hand, the proposed method work well for the synthetic dataset even when the test data is extrapolation data for the training data and the validation data.

Table 1: The experimental results for the synthetic dataset. The row "S3D" shows the result for the baseline algorithm with S3D. For each X ∈ {RF, MLP, CNN, S3D}, the row "X + PI" shows the results for the X-based algorithm with persistence images. For each X ∈ {training, validation, test}, the column "X + acc" (resp. "X + F1") shows the accuracy (resp. F1-score) for the X data. Epoch of the models is 1000. RF has not been evaluated by validation data.

	Dataset 1					
	training acc	validation acc	test acc	training F1	validation F1	test F1
S3D	0.995	0.959	0.464	0.996	0.977	0.504
RF + PI	0.67	-	0.728	0.83	-	0.73
MLP + PI	0.920	0.859	0.836	0.943	0.906	0.894
CNN + PI	0.935	0.819	**0.851**	0.955	0.873	**0.901**
S3D + PI	1.00	0.779	0.796	1.00	0.870	0.878

	Dataset 2					
	training acc	validation acc	test acc	training F1	validation F1	test F1
S3D	1.00	0.959	0.751	1.00	0.977	0.782
RF + PI	0.67	-	0.772	0.83	-	0.77
MLP + PI	0.919	0.938	**0.892**	0.945	0.891	0.923
CNN + PI	0.915	0.819	**0.892**	0.942	0.876	**0.924**
S3D + PI	1.00	0.799	0.796	1.00	0.878	0.874

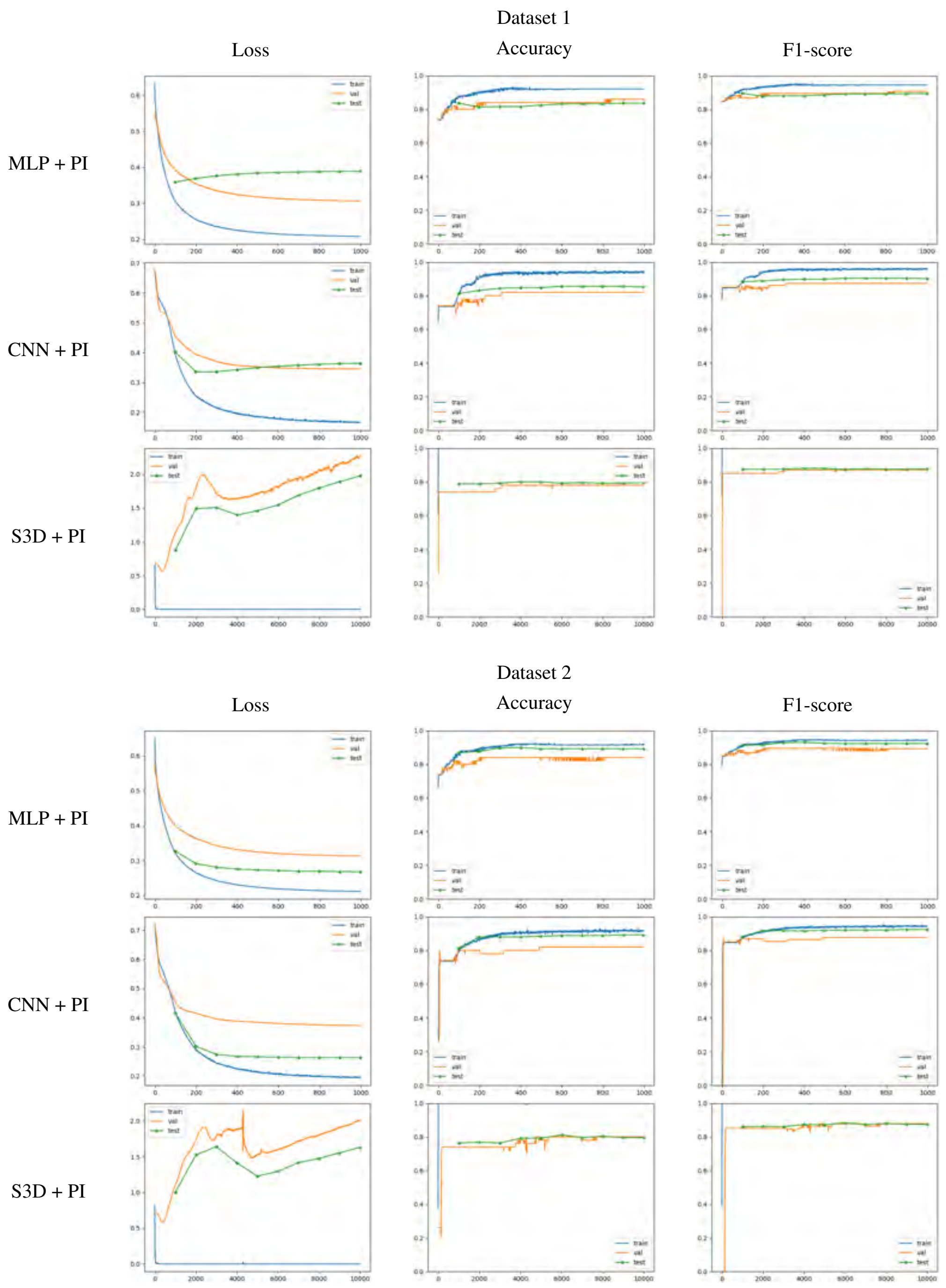

Figure 4: The learning curves. The blue lines show the results for the training data. The orange lines show the results for the validation data. The green lines show the results for the test data.

5.3 Sim2Real on Real-world Dataset

In this subsection, we show the results for an experiment with real data. More precisely, we used the data of Bidirectional flow in Pedestrian Dynamics Data Archive. In particular, we used 15 instances in

Bidirectional flow, in which 25 frames of the positions of pedestrians per second had been sampled. We also created a simulation that replicates the real-world dataset. First, we examine whether persistence image can predict congestion in the real-world dataset using leave-one-out cross-validation (**Realdata**). We selected one instance from 15 instances as a test data, and we did this for every instance. The average of 15 experiments is a score. Next, we examine whether a model trained on the synthetic data generated from the replicated simulation can predict congestion in the real-world dataset (**Sim2Real**). The objective of the second experiment is to evaluate whether the MLP+PI, CNN+PI, and S3D+PI architectures that demonstrated high performance on the synthetic dataset are also effective on the real-world dataset.

5.3.1 Dataset

The geometry of this experiment is the corridor in Figure 5. Pedestrians come from the left entrance or the right entrance.

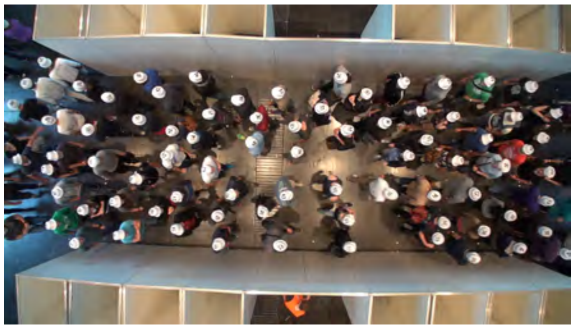

Figure 5: The geometry in the real-world dataset (cited from (Cao et al. 2017)).

We gave the labels True and False depending on whether pedestrians remain in the space at the moment when 20 seconds passes after the last pedestrian appears. If pedestrians remain, then we give the label True to this instance. The number of instances with the label True is 4. The synthetic data is also labeled using the same policy.

5.3.2 Result

Table 2 shows the experimental results. This shows that the proposed method works well for the real-world dataset. It also works well when a model is trained only in the simulator.

Table 2: The result for real-world dataset. Epoch of the models is 1000.

	Realdata		**Sim2Real**	
	training F1	test F1	training F1	test F1
MLP + PI	1.000	0.666	0.980	0.533
CNN + PI	1.000	**0.866**	0.982	0.466
S3D + PI	0.738	0.666	1.000	**0.800**

6 CONCLUSION

In this paper, we have developed a novel topological data analysis approach for detecting congestion clusters in pedestrian crowds. Our method successfully identifies potentially dangerous congestion patterns without requiring prior knowledge of high-risk locations or individual pedestrian tracking. We validated our approach using both multi-agent simulations incorporating realistic friction effects and real-world pedestrian movement data. The experiments demonstrate that using persistent homology as a feature vector improves extrapolation and Sim2Real performance.

This study suggests several promising directions for future work. First, validating our method across diverse spatial configurations and architectural layouts would strengthen its generalizability. Our multi-agent simulation based validation approach is compatible with the existing model evaluation benchmarks (e.g., multi-agent path-finding benchmark that provides diverse spatial configurations (Stern et al. 2019)); therefore, we can study the point by overcoming the limitations of real-world data collection. Second, investigating how our topological approach performs under different crowd densities and movement patterns could yield valuable insights. Finally, integrating our detection method with real-time monitoring systems could enhance its practical application in crowd management and safety.

ACKNOWLEDGMENTS

This work was supported by Fujitsu Limited and JST, PRESTO Grant Number JPMJPR236C, Japan. The fourth author was partially supported by NICT Commissioned Research No.22301 and JST Moonshot R&D Grant Number JPMJMS2021, Japan. This research was partially conducted by the Fujitsu Small Research Lab "Division of Fujitsu Mathematical Modeling for Decision Making," a joint research center between Fujitsu Limited and Kyushu University.

REFERENCES

Adams, H., T. Emerson, M. Kirby, R. Neville, C. Peterson, P. D. Shipman, *et al*. 2017. "Persistence Images: A Stable Vector Representation of Persistent Homology". *Journal of Machine Learning Research* 18(8):1–35.

Breiman, L. 2001. "Random Forests". *Machine Learning* 45(1):5–32.

Buchet, M., Y. Hiraoka, and I. Obayashi. 2018. "Persistent Homology and Materials Informatics". In *Nanoinformatics*, edited by I. Tanaka, 75–95. Singapore: Springer.

Cao, S., A. Seyfried, J. Zhang, S. Holl, and W. Song. 2017. "Fundamental Diagrams for Multidirectional Pedestrian Flows". *Journal of Statistical Mechanics: Theory and Experiment* 2017(3):033404.

Edelsbrunner, H., and J. Harer. 2022. *Computational Topology: An Introduction*. Providence, RI: American Mathematical Society.

Feliciani, C., and K. Nishinari. 2018. "Measurement of Congestion and Intrinsic Risk in Pedestrian Crowds". *Transportation Research Part C: Emerging Technologies* 91:124–155.

Feliciani, C., K. Shimura, and K. Nishinari. 2022. *Introduction to Crowd Management - Managing Crowds in the Digital Era: Theory and Practice*. Cham, Switzerland: Springer.

Frank, G. A., and C. O. Dorso. 2011. "Room Evacuation in the Presence of an Obstacle". *Physica A: Statistical Mechanics and its Applications* 390(11):2135–2145.

Helbing, D., I. Farkas, and T. Vicsek. 2000. "Simulating Dynamical Features of Escape Panic". *Nature* 407:487–490.

Helbing, D., I. J. Farkas, P. Molnár, and T. Vicsek. 2002. "Simulation of Pedestrian Crowds in Normal and Evacuation Situation". In *Pedestrian and Evacuation Dynamics*, edited by M. Schreckenberg and S. D. Sharma, 21–58. Berlin, Heidelberg, Germany: Springer.

Helbing, D., and P. Molnár. 1995. "Social Force Model for Pedestrian Dynamics". *Physical Review E* 51:4282–4286.

Hiraoka, Y., T. Nakamura, A. Hirata, E. G. Escolar, K. Matsue, and Y. Nishiura. 2016. "Hierarchical Structures of Amorphous Solids Characterized by Persistent Homology". *Proceedings of the National Academy of Sciences* 113(26):7035–7040.

Huang, L., T. Chen, Y. Wang, and H. Yuan. 2015. "Congestion Detection of Pedestrians using the Velocity Entropy: A Case Study of Love Parade 2010 disaster". *Physica A: Statistical Mechanics and its Applications* 440:200–209.

Kaji, S., T. Sudo, and K. Ahara. 2020. "Cubical Ripser: Software for Computing Persistent Homology of Image and Volume Data". *arXiv* arXiv:2005.12692.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. "Gradient-based Learning Applied to Document Recognition". *Proceedings of the IEEE* 86(11):2278–2324.

Parisi, D., and C. Dorso. 2005. "Microscopic dynamics of pedestrian evacuation". *Physica A: Statistical Mechanics and its Applications* 354:606–618.

Parisi, D., and C. Dorso. 2007. "Morphological and dynamical aspects of the room evacuation process". *Physica A: Statistical Mechanics and its Applications* 385(1):343–355.

Pun, C. S., K. Xia, and S. X. Lee. 2022. "Persistent-homology-based Machine Learning: A Survey and a Comparative Study". *Artificial Intelligence Review* 55:5169–5213.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986. "Learning Representations by Back-propagating Errors". *Nature* 323:533–536.

Stern, R., N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, *et al*. 2019. "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks". In *Proceedings of the 12th Annual Symposium on Combinatorial Search*. July 16-17, Napa, CA, 151–158.

Sticco, I. M., G. A. Frank, F. E. Cornes, and C. O. Dorso. 2020. "A Re-examination of the Role of Friction in the Original Social Force Model". *Safety Science* 121:42–53.

Xie, S., C. Sun, J. Huang, Z. Tu, and K. Murphy. 2018. "Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification". In *Proceedings of the 15th European Conference on Computer Vision*. September 8-14, Munich, Germany, 318–335.

AUTHOR BIOGRAPHIES

NAOYUKI KAMIYAMA is a Professor in Institute of Mathematics for Industry at Kyushu University in Fukuoka, Japan. His research interests include the theoretical aspect of discrete optimization and its applications to social system design. His email address is kamiyama@imi.kyushu-u.ac.jp and his website is https://imi.kyushu-u.ac.jp/~kamiyama/. This author contributed equally to manuscript development.

HIROAKI YAMADA is a Researcher at Fujitsu Ltd., Kawasaki, Japan. His research interests include social simulation and the practical application of systems science. His email address is yamadah@fujitsu.com. This author contributed equally to manuscript development.

TAKASHI KATO is a Researcher at Fujitsu Ltd. He earned his master degree in Information Science from Kyushu Institute of Technology. His email address is t_kato@fujitsu.com.

SHIZUO KAJI is a Professor at the Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan, and the Graduate School of Science, Kyoto University, Japan. His research interests include topology and its applications to science and industry. His email address is skaji@imi.kyushu-u.ac.jp and his website is https://www.skaji.org.

TETSURO TAKAHASHI is an Associate Professor in Graduate School of Science and Engineering at Kagoshima University in Kagoshima, Japan. His research fields are Natural Language Processing, Knowledge Discovery and Data/Text Mining. His email address is takahashi@ibe.kagoshima-u.ac.jp.