sensors

MDPI

Article

Bimodal Extended Kalman Filter-Based Pedestrian Trajectory Prediction

Chien-Yu Lin [1,*], Lih-Jen Kau [1] and Ching-Yao Chan [2]

[1] Department of Electronic Engineering, National Taipei University of Technology, Taipei 106344, Taiwan
[2] California Partners for Advanced Transportation Technology, University of California, Berkeley, CA 94804, USA
* Correspondence: t107368003@ntut.org.tw

Abstract: We propose a pedestrian trajectory prediction algorithm based on the bimodal extended Kalman filter. With this filter, we are able to make full use of the dual-state nature of the pedestrian movement, i.e., the pedestrian is either moving or remains stationary. We apply the dual-mode probability model to describe the state of the pedestrian. Based on this model, we construct the proposed bimodal extended Kalman filter to estimate pedestrian state distribution. The filter obtains the state distribution for each pedestrian in the scene, respectively, and use that state distribution to predict the future trajectories of all the people in the scene. This prediction method estimates the prior probability of each parameter of the model through the dataset and updates the individual posterior probability of the pedestrian state through the bimodal extended Kalman filter. Our model can predict the trajectory of every individual, by taking the social interaction of pedestrians as well as the surrounding physical obstacles into account, with less than fifty model parameters being used, while with the limited parameter, our model could be nearly accurate as other deep learning models and still be comprehensible for model users.

Keywords: pedestrian trajectory prediction; bimodal extended Kalman filter; point-cloud

check for updates

Citation: Lin, C.-Y.; Kau, L.-J.; Chan, C.-Y. Bimodal Extended Kalman Filter-Based Pedestrian Trajectory Prediction. *Sensors* **2022**, *22*, 8231. https://doi.org/10.3390/s22218231

Academic Editor: Hossam A. Gabbar

Received: 25 September 2022
Accepted: 24 October 2022
Published: 27 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Pedestrian trajectory prediction and pedestrian state filtering are often applied in various fields such as self-driving systems [1,2] and pedestrian tracking in surveillance systems [3,4]. In the self-driving system, the state estimation and the tracking algorithm play an important role in the system.

A good prediction algorithm can help the autonomous driving system compute a better plan according to the encountered situation [2]. By performing tracking for the moving pedestrian around its surroundings, and taking the prediction of the future trajectories of pedestrians into account, the path planner would able to avoid the potential collision with the moving pedestrians or planning in a less oppressive route to bypass its surrounding pedestrians.

On the other hand, the prediction algorithm also plays a key component in the Multi-Object Tracking (MOT) algorithm in surveillance systems. A good prediction model would lower the trajectory prediction error for trackers [5] and therefore help the data association and tracker to track the moving object more accurately in the Multi-Object Tracking algorithm in the surveillance system.

Recently, deep learning-based models such as social GAN [6] showed their ability to predict socially acceptable trajectories via the deep neural network. These models often suffer from lacking the interpretability of the prediction models. It is challenging for the model users to comprehend the models, as the meaning of the weights and biases in the deep neural networks are not obvious. Such a situation motivates us to pursue a model with interpretability while keeping the multi-modality and the social behavior of pedestrians in mind.

Literature Review

For the current pedestrian trajectory prediction algorithms, the algorithms could be divided into two categories: one is the deep learning-based algorithm [6–10] and the other is the traditional model-based algorithm [11–14].

Deep learning-based pedestrian prediction models [6–10] are built on top of various neural network structures. This kind of algorithm works by reading the pedestrian trajectories and the context of the surrounding environment, generating the features via deep neural networks, and finally predicting the future pedestrian trajectories based on these features. On the other hand, the traditional algorithms [11–14] predict the trajectories by first manually analyzing the context in which the pedestrians are moving. After that, the prediction model is built to extract the features based on the analysis and predict the future trajectories on top of the extracted features.

For the deep learning-based pedestrian trajectory prediction models, take the social LSTM [7] as an example. The social LSTM [7] is a deep learning-based pedestrian trajectory prediction model and is proposed by Alahi Alexandre et al. They use the LSTM [15] to consume the observed trajectories and construct the trajectory features for every individual in the scene, and propose a social pooling mechanism to calculate the features of the interactions between pedestrians. This algorithm is a classic deep learning-based method that takes social interaction into consideration.

Continuing from the social LSTM [7], Agrim Gupta et al. propose the social GAN [6]. They employ the GAN [16] architecture in solving the pedestrian trajectory prediction problem. They also adjust the social pooling mechanism from that of social LSTM [7]. A new maximum pooling-based social pooling mechanism is proposed to construct social features. Furthermore, they improve the pooling mechanism and reduce the time of social feature calculation by computing interaction features per scene instead of computing the interaction for every frame.

For the traditional pedestrian algorithms, Dirk Helbing and Peter Molnar propose the concept of social force and use this concept to build the social force model. They treat pedestrians as moving particles, which are affected by the social force from the surrounding. For example, a pedestrian is attracted by the goal and therefore he/she is affected by the attraction force from the goal and trying to move toward the goal. Furthermore, a pedestrian might want to avoid collision with obstacles and other pedestrians, which is modeled as the repulsive force to the pedestrians. The repulsive force discourages the pedestrian from moving in that direction. The social force model is built with the consideration of these forces in its model dynamics.

Yamaguchi Kota et al. propose an energy-based pedestrian behavior model [14]. They construct a series of energy functions to describe how pedestrians see the environment and use the energy function to model the motion of the pedestrians. For example, pedestrians prefer moving at a certain speed and maintaining their current velocity, pedestrians want to move toward the destination, and pedestrians in the same group tend to maintain the same speed and same moving direction. They assume that pedestrians have the nature to move in a way such that the energy is minimized. Therefore they propose to take the negative gradient of the mixture energy function as the pedestrian's velocity and use the velocity to predict future trajectories.

As the traditional model-based pedestrian trajectories prediction models have the advantage of comprehensibility for the model users. The model users can see how the model work. Furthermore, it is much easy for the users to tweak the internal parameters compared to that of the deep learning-based models. Therefore, this study chooses the traditional model as our research direction.

In this research, we employ the concept of the social force model [11] and propose the use of a dual-mode Kalman filter. We consider the static and moving mode of pedestrian motion and predict the trajectory according to the mode of the pedestrians.

2. Materials and Methods

2.1. Dataset Preparation

In this research, we aim to create a motion model of pedestrians and predict their future trajectories based on observed data in the scene. In order to construct the pedestrian trajectory prediction algorithm, optimize the model parameters, or evaluate the performance of the algorithm, relevant data need to be prepared for follow-up research work. There are two parts of the data to be prepared in this research: one is the pedestrian trajectories in the scene, and the other one is the map of static obstacles in the environment.

To build the dataset for the experiment, we first collect the data at the front square of the National Taipei University of Technology (NTUT), with the help of the Velodyne VLP-16 3D LiDAR sensor. We installed the LiDAR on top of a table, which is near the center of the square, so that the sensor would have a better field of view of the surrounding pedestrians. The environment and setup is shown in Figure 1. The point-cloud data is then collected at the frame rate of 10 Hz for 90 min starting at 16:00 on 1 November 2020.



Figure 1. The setup of the Velodye LiDAR sensor.

After the point-cloud data is collected, the data have to be processed into the format that is suitable for our research needs. The data would be processed and transformed into an environment obstacle map and pedestrian trajectories dataset. To extract the map and trajectories, a list of steps is executed sequentially.

To extract the trajectories data, firstly, the background point-cloud extraction is performed on all the collected frames of point-clouds by building the histogram of ray-hitting distance for every direction of the laser ray and extracting the most frequently hitting point for every ray direction. In such a way, we are able to extract the background point cloud.

Secondly, after the background point-cloud is retrieved, we take this background and the collected point-clouds as input. We apply the background subtraction method to extract the foreground pedestrian point-clouds. Then we perform the clustering and multi-object tracking algorithm to extract the pedestrian trajectories as shown in the algorithm [17]. The tracked cluster and trajectories are shown in Figure 2.

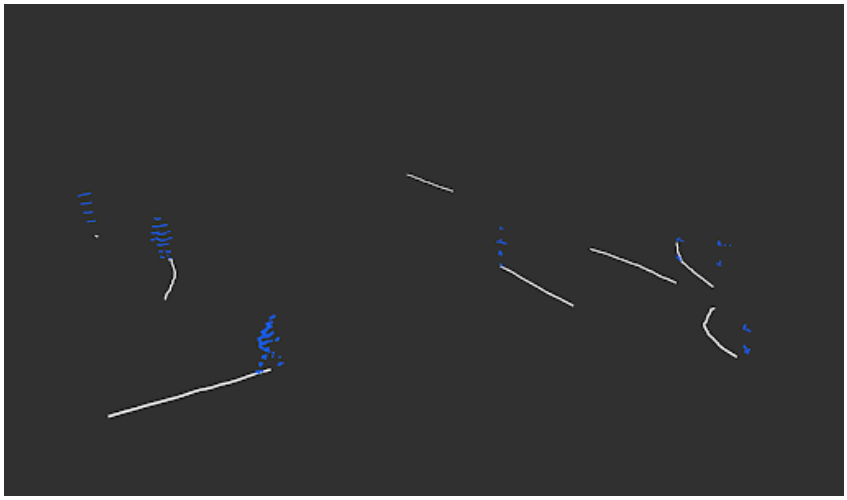


Figure 2. Example of tracked pedestrian trajectories.

Finally, the extracted trajectories are manually adjusted and annotated, so that the trajectories would not suffer from the errors during the tracking algorithm in the previous step. The frame rate for the trajectories data is also down-sampled to 2.5 frames per second, which is aligned with that in trajectories dataset [18,19].

To obtain the environment obstacle map, we perform the SLAM algorithm [1] to build the point-cloud map of the experiment and align the point-cloud map to the background map with the scan-matching algorithms [20,21]. Then, we keep the points in our range of interests and flatten the 3D point cloud into a 2D one. Finally, we down-sample the point cloud to reduce the points and obtain the obstacle map eventually. This map represents the static obstacles in the experiment environment as shown in Figure 3.

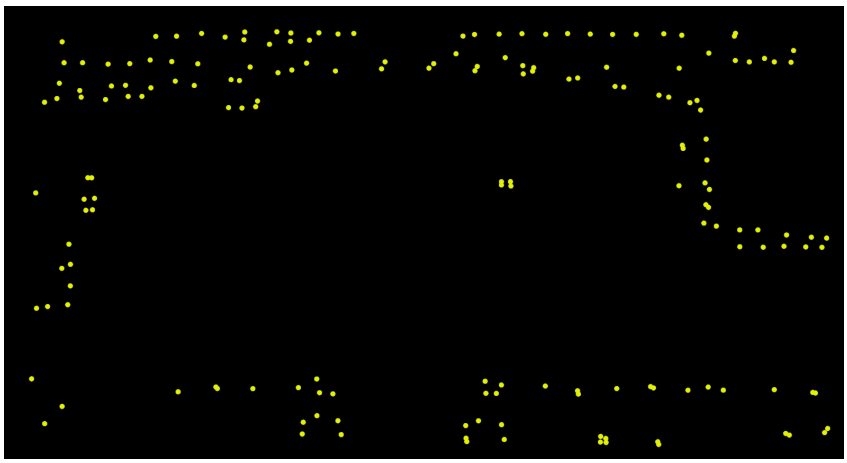


Figure 3. Down-sampled obstacle map.

2.2. Proposed Model

We proposed a bimodal extended Kalman filter-based pedestrian trajectory prediction model. This model adopts the bimodal extended Kalman filter, which is a special implementation of the Bayes filter. This filter is applied to encode the observed trajectories of the pedestrians and their surrounding obstacles into the state representation. After the state representation is encoded via the filter, the representation is then passed to the decoder to predict the future trajectories of the pedestrians.

2.2.1. Problem Formulation

In this study, we proposed a pedestrian trajectories prediction model $\mathcal{G}$. The model consumes the observed pedestrian trajectories X and obstacles O of their walking environment, and predict the future trajectories $\hat{Y}$ of the pedestrians in the scenes. We formulate the problem as follows:

$$\hat{Y} = \mathcal{G}(X, O) \tag{1}$$

1. Observed trajectories $X \triangleq \langle \zeta_{\text{obs},i} \rangle_{i=1:I}$: The observed trajectories X are the collection of the observed pedestrian trajectory $\zeta_{\text{obs},i}$. The pedestrian index number i is used in the symbol subscript to denote a specific pedestrian trajectory in the scene. The pedestrian trajectory $\zeta_{\text{obs},i} \triangleq \langle \mathbf{p}_i^t \rangle_{t=1:t_{\text{obs}}}$ is defined as a sequence of 2D positions $\mathbf{p}_i^t \triangleq \langle x_i^t, y_i^t \rangle \in \mathbb{R}^2$. It describes how a specific pedestrian is being located throughout a specific period of frames, which starts from frame $t = 1$ to $t = t_{\text{obs}}$. In this paper, we set the observed sequence length t_{obs} to 8.
2. Predicted trajectories $\hat{Y} \triangleq \langle \hat{\zeta}_{\text{pred},i} \rangle_{i=1:I}$: The predicted trajectories $\hat{Y}$ are the collection of the predicted pedestrian trajectory $\hat{\zeta}_{\text{pred},i}$, which are generated via the prediction model $\mathcal{G}$. The pedestrian index number i is used in the symbol subscript to denote a specific pedestrian trajectory in the scene. The pedestrian trajectory $\hat{\zeta}_{\text{pred},i} \triangleq \langle \hat{\mathbf{p}}_i^t \rangle_{t=t_{\text{obs}}+1:t_{\text{obs}}+t_{\text{pred}}}$ is defined as a sequence of 2D positions $\hat{\mathbf{p}}_i^t \triangleq \langle x_i^t, y_i^t \rangle \in \mathbb{R}^2$. It describe how a specific pedestrian is located throughout a specific period of time frames, which starts from frame $t = t_{\text{obs}}$ to $t = t_{\text{obs+pred}}$. In this paper, we set the predicted sequence length t_{pred} to 8.
3. Ground-truth predicted trajectories $Y \triangleq \langle \zeta_{\text{pred},i} \rangle_{i=1:I}$: The ground-truth predicted trajectories represent the pedestrian trajectories in the future like the predicted trajectories, but they are not generated by a prediction model $\mathcal{G}$. Instead, they describe the pedestrians' real trajectories in the dataset throughout the same time frames as the predicted trajectories.
4. Environment Obstacles $O \triangleq \langle \mathbf{o}_k \rangle_{k=1:K}$: The environment obstacles are a set of obstacles in the pedestrian walking environment. An obstacle is a 2D point representing where the obstacle is located.

2.2.2. State Representation

In our work, we choose the setting for the state so that it is able to represent the state information of that pedestrian in the specific frame. The state $\mathbf{s} \triangleq \langle m, \mathbf{x} \rangle$ is a tuple that consists of two variables.

The first variable $m \in \{0, 1\}$, named mode, is the discrete random variable describing the pedestrian's moving mode. We consider the pedestrians has two moving modes, hence this variable is a binary variable. We define $m = 0$ being the static mode, any pedestrian in this mode prefers to stay in the same location. On the other hand, we defined the mode $m = 1$ as the moving mode, any pedestrian in such mode would prefer moving but not staying in the same location.

The second variable $\mathbf{x} \triangleq [\mathbf{p}^\mathsf{T}\ \mathbf{v}^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^4$, named base state, is a 4-dimensional vector which contains the 2-dimensional position $\mathbf{p}$ and 2-dimensional velocity $\mathbf{v}$ of the pedestrian.

With such a setting, we are able to estimate the state and predict the trajectories in a mode-aware manner.

2.2.3. State Distribution

To represent the belief distribution of the state variable, we first decompose the belief state distribution $bel(\mathbf{s})$ into the product of the mode distribution $p(m)$ and the base state distribution $p(\mathbf{x} \mid m)$ conditioning on the mode. For the mode distribution $p(m)$, it is a Bernoulli distribution; therefore, the distribution is parameterized with the weight parameter w_m to represent the probability under the specific mode. For the mode-conditioned base state distribution $p(\mathbf{x} \mid m)$, we parameterize the distribution as a multivariate normal distribution, with $\boldsymbol{\mu}_m$ as its mean and Σ_m as its covariance matrix. The subscript m shows

the mode that the parameter is conditioning on. We denote the belief state distribution as $bel(\mathbf{s}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_m, \Sigma_m)w_m$.

2.2.4. The Proposed Model

For the proposed prediction model, we first encode the trajectories into the belief state distribution, which is the state distribution built on top of the observed data. To calculate the state distributions for the pedestrians in the scene. We introduce the bimodal extended Kalman filter to help encode the observed trajectories into the state distributions. The algorithm is performed as follows.

Just like the extended Kalman filter [22] is an implementation of the Bayes filter, the proposed bimodal extended Kalman filter is an implementation of the bimodal Bayes filter. We will first show the algorithm of the bimodal Bayes filter, as it provides the overview of the workflow for the bimodal extended Kalman filter. Then we will illustrate the algorithm of the bimodal extended Kalman filter.

The bimodal Bayes filter is a specialization of the Bayes filter in which the state consists of a binary mode variable and a continuous multi-dimensional variable. The estimated state is processed in two phases. For the first phase, the current state distribution is predicted based on the previous state distribution with the help of the state prediction model. In the case of the bimodal extended Kalman filter, there are three minor steps in the prediction phase. Firstly, the mode is being predicted with the mode transition model $p(m' \mid m) \triangleq t_{m'|m}$. Secondly, the base state for each mode is predicted via the mode-specific motion model $\mathbf{x}' = g_{m'|m}(\mathbf{x}, \boldsymbol{\epsilon})$. Finally, the predicted base states would be merged into the predicted mode-conditioned base state distribution. After the first phase is completed, the second phase is followed. The second phase is the correction phase, the predicted distribution would be corrected based on the observed data in this. In the correction phase, there are also three minor steps. Firstly, the likelihood function of the state is being computed based on the observed pedestrian position $\mathbf{z}$ and the observation model $\mathbf{z} = h_{m'}(\mathbf{x}, \boldsymbol{\delta})$. Then, the mode is corrected with the likelihood function via the Bayes rule. Finally, the predicted base state is corrected.

To estimate the state with the bimodal Bayes filter, one should provide the previous belief state distribution and the state observation, with a properly configured state transition model and observation model. The state-transition model takes the previous state as input and computes the current state. The state transition model contains two parts: one is the mode transition model and the other one is the base state transition model. The mode transition model $p(m' \mid m)$ represent the mode transition probability and is parameterized as $t_{m'|m}$. The base state transition model $g_{m'|m}$ is a function that takes the previous base state $\mathbf{x}$ and a standard normal noise vector $\boldsymbol{\epsilon}$ as input and produces the current base state $\mathbf{x}'$, and shows the base state transition from the previous mode m to the current m'. The transition process could be denoted as $\mathbf{x}' \sim p(\mathbf{x}' \mid \mathbf{x}, m, m') \iff \mathbf{x}' = g_{m'|m}(\mathbf{x}, \boldsymbol{\epsilon}), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I)$. The observation model models the observation process of the state. It takes the state s and the observation noise vector $\boldsymbol{\delta}$ as input, and produces the observation vector $\mathbf{z}$ as output. The observation process could be denoted as $p(\mathbf{z}' \mid \mathbf{x}', m') \iff \mathbf{z}' = h_{m'}(\mathbf{x}', \boldsymbol{\delta}), \boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, I)$. The pseudo-code for this algorithm is shown in Algorithm 1.

As the bimodal extended Kalman filter is the implementation of the bimodal Bayes filter, the bimodal extended Kalman filter further assumes the base state is normally distributed and linearizes the base state transition model as well as the observation model to approximate posterior distribution, which is similar to how the extended Kalman filter [22] extends the Kalman filter [23]. The prediction phase and the correction phase are performed sequentially in the filtering algorithm. The pseudo-code for this algorithm is shown in Algorithm 2.

Algorithm 1: Bimodal Bayes Filter.

Input: belief distribution $bel(\mathbf{s})$ of the previous state s
Output: belief distribution $bel(\mathbf{s}')$ of the current state s'
// Prediction step
$\overline{bel}(\mathbf{s}') = \overline{bel}(\mathbf{x}' \mid m')\overline{bel}(m')$ where
  // Mode Prediction
  $\overline{bel}(m') = \sum_m p(m' \mid m)bel(m)$
  // Base State Prediction
  $p(\mathbf{x}' \mid m, m') = \int p(\mathbf{x}' \mid x, m, m')bel(\mathbf{x} \mid m)dx$
  $p(m \mid m') = \frac{p(m'|m)bel(m)}{\overline{bel}(m')}$
  $\overline{bel}(\mathbf{x}' \mid m') = \sum_m p(\mathbf{x} \mid m, m')p(m \mid m')$
// Correction step
$bel(\mathbf{s}') = bel(\mathbf{x}' \mid m')bel(m')$ where
  // Mode Correction
  $p(\mathbf{z}' \mid m') = \int p(\mathbf{z}' \mid x', m')\overline{bel}(\mathbf{x}' \mid m')dx'$
  $\eta_{m'} = 1/\sum_{m'} p(\mathbf{z}' \mid m')\overline{bel}(m')$
  $bel(m') = \eta_{m'}p(\mathbf{z}' \mid m')\overline{bel}(m')$
  // Base State Correction
  $\overline{bel}(\mathbf{z}', \mathbf{x}' \mid m') = p(\mathbf{z}' \mid x', m')\overline{bel}(\mathbf{x}' \mid m')$
  $bel(\mathbf{x}' \mid m') = \overline{bel}(\mathbf{z}', \mathbf{x}' \mid m')/p(\mathbf{z}' \mid m')$
return $bel(m', x')$

To estimate the current state distribution of the algorithm, the belief distribution of the previous state $bel(\mathbf{s}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_m, \Sigma_m)w_m$ is passed into the filter as the input, and it is followed by the prediction phase (Algorithm 2 line 1) and the correction phase (Algorithm 2 line 10), and finally the belief distribution of the current state $bel(\mathbf{s}') = \mathcal{N}(\mathbf{x}'; \boldsymbol{\mu}'_{m'}, \Sigma'_{m'})w'_{m'}$ is produced as the output.

In the prediction phase, the mode is predicted using the Bayes rule with the mode transition model $p(m' \mid m) \triangleq t_{m'|m}$ (Algorithm 2 lines 2 and 3). Then, the base state distribution $p(\mathbf{x}' \mid m, m') \triangleq \mathcal{N}(\mathbf{x}'; \bar{\mu}'_{m',m}, \bar{\Sigma}'_{m',m})$ is predicted with the linearized base state transition model (Algorithm 2 lines 4–6). The Jacobian matrices G and E of the base state transition model $g_{m',m}$ with respect to the base state $\mathbf{x}$ and the standard multivariate normal noise $\boldsymbol{\epsilon}$ are computed. These Jacobian matrices are used for computing the first-order approximation of the predicted state distribution. The predicted modal mean $\bar{\boldsymbol{\mu}}'_{m',m}$ is approximated by the application of the base state transition model $g_{m',m}$ at the mean of previous base state $\boldsymbol{\mu}_m$ and the zero mean of the standard multivariate-normal noise. The predicted modal covariance $\bar{\Sigma}'_{m',m}$ is computed as the sum of the transformed previous base state covariance $G\Sigma_m G^\mathsf{T}$ and the transformed noise covariance EE^T. Finally, the predicted modal base state distributions are merged into the predicted base state distribution. As the mixture of two normal distributions may not distribute normally, the mixture normal distribution is merged approximately by preserving the mean and covariance of the mixture distribution. After the prediction phase is completed, the correction phase is followed (Algorithm 2 lines 8 and 9).

Algorithm 2: Bimodal Extended Kalman Filter.

Input: belief distribution $bel(\mathbf{s}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_m, \Sigma_m) w_m$ of the previous state s
Output: belief distribution $bel(\mathbf{s}') = \mathcal{N}(\mathbf{x}'; \boldsymbol{\mu}'_{m'}, \Sigma'_{m'}) w'_{m'}$ of the current state s'
// Prediction step
$\overline{bel}(\mathbf{s}') = \mathcal{N}(\mathbf{x}'; \bar{\boldsymbol{\mu}}'_{m'}, \bar{\Sigma}'_{m'}) \bar{w}'_{m'}$ where
  // Mode Prediction
  $\bar{w}'_{m'} = \sum_m t_{m'|m} w_m$
  $\bar{w}'_{m',m} = \frac{t_{m'|m} w_m}{\bar{w}'_{m'}}$
  // Predict the base state for each mode
  $G = G_{m',m} = \frac{\partial g_{m',m}}{\partial \mathbf{x}}$
  $E = E_{m',m} = \frac{\partial g_{m',m}}{\partial \epsilon}$
  $\bar{\boldsymbol{\mu}}'_{m',m} = g_{m',m}(\boldsymbol{\mu}_m, \mathbf{0})$
  $\bar{\Sigma}'_{m',m} = G \Sigma_m G^\mathsf{T} + E E^\mathsf{T}$
  // Merge the base state
  $\bar{\boldsymbol{\mu}}'_{m'} = \sum_m w_{m',m} \boldsymbol{\mu}_{m',m}$
  $\bar{\Sigma}'_{m'} = \sum_m \bar{w}'_{m',m} \bar{\Sigma}'_{m',m} + \sum_m \bar{w}'_{m',m} \bar{\boldsymbol{\mu}}'_{m',m} \bar{\boldsymbol{\mu}}'^\mathsf{T}_{m',m} - \bar{\boldsymbol{\mu}}'_{m'} \bar{\boldsymbol{\mu}}'^\mathsf{T}_{m'}$
// Correction step
$bel(\mathbf{s}') = \mathcal{N}(\mathbf{x}'; \boldsymbol{\mu}'_{m'}, \Sigma'_{m'}) w'_{m'}$ where
  // Compute observation distribution $p(\mathbf{z}' \mid m') \triangleq \mathcal{N}(\mathbf{z}'; \bar{\boldsymbol{\nu}}'_{m'}; \bar{\Pi}'_{m'})$
  $H = H_{m'} = \frac{\partial h_{m'}}{\partial \mathbf{x}}$
  $D = D_{m'} = \frac{\partial h_{m'}}{\partial \delta}$
  $\bar{\boldsymbol{\nu}}'_{m'} = h_{m'}(\bar{\boldsymbol{\mu}}'_{m'}, \mathbf{0})$
  $\bar{\Pi}'_{m'} = H \bar{\Sigma}'_{m'} H^\mathsf{T} + D D^\mathsf{T}$
  // Mode correction
  $\eta = 1 / \sum_{m'} \bar{w}'_{m'} \mathcal{N}(\mathbf{z}'; \bar{\boldsymbol{\nu}}'_{m'}, \bar{\Pi}'_{m'})$
  $w'_{m'} = \eta \cdot \bar{w}'_{m'} \mathcal{N}(\mathbf{z}'; \bar{\boldsymbol{\nu}}'_{m'}, \bar{\Pi}'_{m'})$
  // Base state correction
  $K = \bar{\Sigma}'_{m'} H^\mathsf{T} \bar{\Pi}'^{-1}_{m'}$
  $\boldsymbol{\mu}'_{m'} = \bar{\boldsymbol{\mu}}'_{m'} + K(\mathbf{z}' - \bar{\boldsymbol{\nu}}^-_{m'})$
  $\Sigma'_{m'} = (I - KH) \bar{\Sigma}'_{m'}$
return $w'_{m'}, \boldsymbol{\mu}'_{m'}, \Sigma'_{m'}$

In the correction phase, the predicted distribution would be corrected based on the observed data $\mathbf{z}'$. There are also three steps in this phase. First, the observation distribution $p(\mathbf{z}' \mid m') \triangleq \mathcal{N}(\mathbf{z}'; \bar{\nu}'_{m'}; \bar{\Pi}'_{m'})$ of the state is computed based on the observed data $\mathbf{z}'$ and the linearized observation model $h_{m'}$ (Algorithm 2 lines 11–14). The Jacobin matrices D and H of the observation $h_{m'}$ with respect to the base state $\mathbf{x}'$ and the standard multivariate normal noise δ are computed. The mean $\bar{\boldsymbol{\mu}}'_{m',m}$ is approximated by the application of the observation model $h_{m'}$ at the mean of the predicted base state $\bar{\boldsymbol{\mu}}'_{m'}$ and the zero mean of the standard multivariate-normal noise. The covariance $\bar{\Pi}'_{m'}$ is computed as the sum of the transformed predicted base state covariance $H \bar{\Sigma}'_{m'} H^\mathsf{T}$ and the transformed noise covariance $D D^\mathsf{T}$. Then, the mode is corrected via the Bayes rule (Algorithm 2 lines 15–16). Finally, the Kalman gain K is computed and the predicted base state distribution is corrected with the Kalman gain (Algorithm 2 lines 17–19).

In our setting, the base state $\mathbf{x}$ represents the pedestrian's position $\mathbf{p}$ and velocity $\mathbf{v}$, the mode m to represent the static and the moving mode pedestrian, and the observation $\mathbf{z}$ represents the position of the pedestrian. The required models for the bimodal extended Kalman filter are specified as follows:

1. For the observation model, it is parameterized with the standard deviation σ_p of the observation noise.

$$\mathbf{z}' = h_{m'}(\mathbf{x}', \delta) \triangleq \mathbf{p}' + \sigma_p \delta, \text{ where } \delta \sim \mathcal{N}(\mathbf{0}, I) \tag{2}$$

2. For the mode transition model, it is defined as:

$$p(m' \mid m) \triangleq t_{m'|m} \tag{3}$$

3. For the base state transition model, we apply different motion-model according to the mode. When current mode m' is in static mode, we model the motion as constant position model. On the other hand, when current mode m' is in moving mode, we apply the social force model [11] as the motion model. For both scenario, an extra noise is added to the velocity term to model the uncertainty of the pedestrian's plan. The model is defined as follows.

$$R(\mathbf{v}) \triangleq R(\begin{bmatrix} v_x \\ v_y \end{bmatrix}) = \frac{1}{\sqrt{v_x^2 + v_x^2}} \begin{bmatrix} v_x & -v_y \\ v_y & v_x \end{bmatrix} \tag{4a}$$

$$L_{m'} \triangleq \begin{bmatrix} \sigma_{vx,m'} & 0 \\ 0 & \sigma_{vy,m'} \end{bmatrix} \tag{4b}$$

$$\mathbf{x}' = g_{m',m}(\mathbf{x}, \boldsymbol{\epsilon}) \triangleq \begin{bmatrix} \mathbf{v}' \Delta t \\ \bar{\mathbf{v}}'_{m'} + L_{m'} R(\mathbf{v}) \boldsymbol{\epsilon} \end{bmatrix},$$
$$\text{where } \bar{\mathbf{v}}_{m'} = m' \cdot \text{SocialForceModel}(\mathbf{x}), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I) \tag{4c}$$

In the encoding step, we adopt the bimodal extended Kalman filter to estimate the current state distribution of every pedestrian. We take the observed pedestrian position sequence $\mathbf{p}^{1:t_{\text{obs}}}$ as the input data, and estimate the state distribution frame by frame recursively. For each frame, the filter takes the observed position and the previous state distribution, and estimates the current state distribution accordingly. Through the steps, the current state distribution is computed, which contains the parameterized variables representing the mode distribution and the base state distribution.

After the current state distribution is estimated, the decoding phase is then applied to the distribution to generate the predicted pedestrian future trajectories. To decode and predict the future trajectory of the pedestrians. First, states are sampled from the computed state distributions. The sampled state is taken as the initial input. The input then goes into the state transition models and predicts the state. Then, the state is generated recursively via the state transition model by taking the previous state as the input, until the required prediction sequence length is reached. To reduce the prediction variance, the velocity noise $\boldsymbol{\epsilon}$ is applied to the only states for the output predicted trajectories.

2.3. Model Optimization

To obtain the optimal model for predicting the pedestrian trajectories. We have to optimize the proposed model and obtain the optimal parameters. We will illustrate the optimization process for the proposed model.

2.3.1. Observation Model Optimization

For the position observation model $\mathbf{z} = h_m(\mathbf{x}, \delta) \triangleq \mathbf{p} + \sigma_p \delta$, it is parameterized via the parameter σ_p to represent the standard deviation of the observation noise.

To compute such parameter, we first compute the smoothed pedestrian trajectories via the BSpline [24] algorithm, and we denote the smooth trajectory as $\tilde{\mathbf{p}}_{1:t_{\text{pred}}}$. Now, for every pedestrian position $\mathbf{p}$ there is a smoothed version of position $\tilde{\mathbf{p}}$. We want to file

the parameter σ_p such that it maximizes the sum of the log likelihood for all positions in the dataset in following equation.

$$\max \prod_{i=1}^{N} \mathcal{N}(\mathbf{p}_i; \tilde{\mathbf{p}}_i, \sigma_p^2 I_{2\times 2}) \tag{5}$$

The optimal parameter could be computed by following equation.

$$\sigma_p = \sqrt{\frac{1}{2N} \sum_{i=1}^{N} \|\mathbf{p}_i - \tilde{\mathbf{p}}_i\|^2} \tag{6}$$

2.3.2. Mode Transition Model Optimization

The mode transition model is a part of the state transition model, and it is used to describe the mode transition probability between the transition of states. The mode transition model is defined as follows.

$$p(m' \mid m; T) = t_{m'|m} \tag{7}$$

We parameterize the state transition as follows:

$$T \triangleq \begin{bmatrix} t_{0|0} & t_{0|1} \\ t_{1|0} & t_{1|1} \end{bmatrix} \tag{8}$$

If we let the previous mode distribution be $p(m) = w_m$, the current mode distribution $p(m') = w'_{m'}$, and the predict current mode distribution be $\overline{bel}(m') = \bar{w}_{m'}$. The process of the prediction could be formulated as follows.

$$\bar{w}'_{m',i} = \sum_{m} t_{m'|m} w_{m,i} \tag{9}$$

We want to compute the optimal parameter T^* such that the sum of square errors between the predicted probability $\bar{w}'_{m',i}$ and the ground-truth $w'_{m',i}$ is minimized.

$$T^* = \arg\min_{T} \sum_{m} (w'_{m',i} - \bar{w}'_{m',i})^2 \tag{10}$$

Then, we reformulate the equation in the form of matrix multiplication.

$$T^* = \arg\min_{T} \sum_{m'} (\mathbf{w}'_{m'} - W\mathbf{t}_{m'})^{\mathsf{T}} (\mathbf{w}'_{m'} - W\mathbf{t}_{m'}) \tag{11}$$

where

$$\mathbf{w}'_{m'} = \begin{bmatrix} w'_{m',1} \\ \vdots \\ w'_{m',N} \end{bmatrix}, W = \begin{bmatrix} w_{0,1} & w_{1,1} \\ \vdots & \vdots \\ w_{0,N} & w_{1,N} \end{bmatrix}, \mathbf{t}_{m'} = \begin{bmatrix} t_{m'|0} \\ t_{m'|1} \end{bmatrix} \tag{12}$$

Solve the equation we have the optimal transition as follows.

$$T^* = \begin{bmatrix} \mathbf{t}_0 & \mathbf{t}_1 \end{bmatrix}^{\mathsf{T}}, \text{ where } \mathbf{t}_{m'} = (W^{\mathsf{T}} W)^{-1} W^{\mathsf{T}} \mathbf{w}'_{m'} \tag{13}$$

2.3.3. Mode Probability Model Optimization

In order to obtain the mode probability $w'_{m,i}$ for optimizing the mode transition model, we construct the mode probability model and compute the mode probability base on this

model. First, we assume the speed of the pedestrians is distributed as the mixture normal distribution, the distribution is defined as follows:

$$p(v;\Theta) \triangleq \sum_{m=0}^{1} w_{v,m}\mathcal{N}(v;\mu_{v,m},\sigma_{v,m}^2) \tag{14}$$

The model is parameterized with six parameters: two for the mode weight $w_{v,0:1}$, two for the speed mean $\mu_{v,0:1}$, and two for the standard deviation $\sigma_{v,0:1}$, the subscript denotes the mode.

$$\Theta \triangleq \langle w_{v,0:1}, \mu_{v,0:1}, \sigma_{v,0:1}\rangle \tag{15}$$

For all the speed data in the pedestrian trajectory dataset, we would like to search for the optimal parameters so that the sum of model log-likelihood could be maximized as follows:

$$\arg\max_{\Theta} \sum_{i} \log p(v_i;\Theta) \tag{16}$$

As we assume the pedestrian prefers to move in either static or moving mode, the velocity noise is different mode is thereby distributed, respectively. We model the velocity noise as normal distribution and distributed, respectively, in each mode. We use the notation $\Theta_{v,m}$ to denote the velocity noise parameter under the specific mode m', it is defined as follows:

$$w'_{m',i} = \frac{w_{v,m'}\mathcal{N}(v'_i;\mu_{v,m'},\sigma_{v,m'}^2)}{p(v'_i;\Theta)} \tag{17}$$

2.3.4. Velocity Noise Parameter Optimization

As we assume the pedestrian prefer to move in either static of moving mode. The velocity noise is different mode is therefore distributed, respectively. We model the velocity noise as normal distribution and distributed, respectively, in each mode. We use the notation $\Theta_{v,m}$ to denote the velocity noise parameter under the specific mode m', it is defined as follows:

$$\Theta_{v,m'} \triangleq \begin{bmatrix}\sigma_{vx,m'}\\ \sigma_{vy,m'}\end{bmatrix} \tag{18}$$

From the motion models, we assumes the velocity difference $\Delta\mathbf{v}'_{m'}$ between ground-truth velocity $\mathbf{v}$ and the predicted velocity $\bar{\mathbf{v}}'_{m'}$ is distributed under a scaled and rotated standard normal distribution:

$$\Delta\mathbf{v}'_{m'} \triangleq \mathbf{v}' - \bar{\mathbf{v}}'_{m'} = RL_{m'}\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},I_{2\times 2}), R = \frac{1}{v_x^2+v_y^2}\begin{bmatrix}v_x & -v_y\\ v_y & v_x\end{bmatrix}, L_{m'} = I_{2\times 2}\Theta_{v,m'} \tag{19}$$

We want to maximize the sum of weighted log likelihood as follows:

$$\arg\max_{\Theta_{v,m'}} \sum_{i} w'_{m',i}\cdot\log\mathcal{N}(\Delta\mathbf{v}'_{m',i};\mathbf{0},RL_{m'}^2R^{\mathsf{T}}) \tag{20}$$

where

$$w'_{m',i} = \left.\frac{w_{v,m'}\mathcal{N}(v'_i;\mu_{v,m'},\sigma_{v,m'}^2)}{p(v'_i;\Theta)}\right|_{v'_i=\|\mathbf{v}'_i\|} \tag{21}$$

The parameter could be calculated by the following formula, note that the square and square root operators are applied in an element-wise manner.

$$\Theta_{v,m'} = \begin{bmatrix}\sigma_{vx,m'}\\ \sigma_{vy,m'}\end{bmatrix} = \sqrt{\frac{\sum_i w'_{m',i}(R^{-1}\Delta\mathbf{v}'_i)^2}{\sum_i w'_{m',i}}} \tag{22}$$

2.3.5. Social Force Model Optimization

For the pedestrians' in the moving mode, we model the moving behavior with a motion model:

$$\hat{\mathbf{x}}' = \text{SocialForceMotionModel}(\mathbf{x}; \Theta_{\text{SFM}}) \tag{23}$$

We define the model loss to be the average of expected weighted displacement error for pedestrians in the moving mode:

$$\mathcal{L}(\Theta_{\text{SFM}}) = \frac{1}{n} \sum_{i=1}^{n} w'_{m',i} \|\hat{\mathbf{p}}'_i - \mathbf{p}'_i\| \Big|_{m=1} \tag{24}$$

We optimize the loss by minimizing the loss via the gradient decent algorithm, the parameters are then eventually retrieved.

3. Results

In our experiment, we divide the collected data into two parts. One is the training set for training the models. The other one is the testing set, which is not involved in the training process. The constant velocity model, social force model [11], and social GAN [6] model are selected to be compared with the proposed model (with and without noise). The proposed model is trained and tested in a leave-one-out manner.

As for the configuration details of the testing models. Our model is configured to be tested with both the deterministic and the stochastic prediction settings. The deterministic model uses the same parameters as those of the stochastic model. However, the sampling process is degenerated: the normal distribution is replaced with Dirac delta distribution and the mode is always transit to the most likely one. In the deterministic model the most likely sample is always picked, and the randomness is canceled.

The social force model is configured to adopt the repulsive force terms from other pedestrians and the obstacles but discard the attractive force terms for reaching the destination and some other objects of interest, as those terms are not obvious during the prediction process. Two parameter sets of the social force model [11] are chosen for this experiment. One follows the original parameters settings as shown in [11], and the other follows the parameters that are refined through the training process.

For the social GAN model, the dimension of the hidden state is set to 32 for the encoder and decoder of the generator, and is configured to 64 for the encoder of the discriminator. The input coordinates are embedded as 16-dimensional vectors. The generator and discriminator are iteratively trained with a batch size of 16 using Adam [25] with an initial learning rate of $5 \cdot 10^{-4}$ for the generator and 10^{-3} for the discriminator. The k-variety loss with $k = 10$ and the pooling mechanism are applied for training this model.

3.1. Hardware and Software Environment

In thisstudy, we used Dell XPS 15 laptop as our main computing platform. All calculations and experiments described in the paper were computed with this device, including the collection of the pedestrian point-cloud data, pedestrian trajectory extraction, data curation, and evaluation for all models. The detailed hardware specifications of this computing platform are shown in Table 1.

Table 1. Hardware specification.

Hardware	Specification
Computer	Dell XPS 15 9570
RAM	8ATF1G64HZ-2G6E1 SODIMM DDR4 Micron 8GiB * 2
CPU	Intel Core i7-8750H CPU @ 2.20 GHz; 6 core; 12 threads
GPU	GeForce GTX 1050 Ti with Max-Q Design/PCIe/SSE2
Storage	THNSN5512GPUK NVMe TOSHIBA 512GB
LiDAR	Velodyne VLP-16

For the part of the software environment, we used Ubuntu 20.04 as the operating system of our developing environment. We used the C++ programming language and the robotic operating system (ROS) for point-cloud data processing, mapping, and trajectory data curation. We used the Python programming language and the Pytorch machine learning framework for building and testing models. The detailed software specifications are shown in Table 2.

Table 2. Software environment.

Software	Version
OS	Ubuntu 20.04.1 LTS
ROS	Noetic Ninjemys
C++	14
GCC	9.4.0
Python	3.8.10
PyTorch	1.12.1+cu113
NVIDIA Driver	470.74
CUDA	11.4

3.2. Metrics and Results

We adopt several metrics to help evaluate the complexity and the performance of the proposed model. The metrics include the model parameters count, model prediction run-time, average displacement error, final displacement error, minimum social distance, and minimum physical distance. We will illustrate the definition of these metrics and show the results in the following paragraphs. The metric with the best performance among the compared models is marked with boldface, and the metric with the second-best performance is underlined.

3.2.1. Model Complexity

To evaluate the complexity of the models, we count the number of parameters of the prediction model to show the space complexity of the model. We also measure the average trajectory prediction run-time per scene in the dataset to show the time complexity of the model. The run-time of the model is tested and measured on both the CPU device and GPU device.

Table 3 shows the metrics for all the models to be compared. Note that we only compared the parameters count for the prediction model, but we do not include the auxiliary models such as the discriminator in the social GAN [6] model.

Table 3. Model Complexity.

	Number of Parameters	Run-Time (CPU)	Run-Time (GPU)
Constant Velocity	**0**	**36.2** µs	**73.7** µs
Social GAN	60,234	3.3 ms	3.0 ms
SFM (trained)	7	9.0 ms	15.1 ms
SFM (original)	7	9.9 ms	15.3 ms
Proposed (deterministic)	14	66.3 ms	78.1 ms
Proposed (stochastic)	14	68.5 ms	78.1 ms

3.2.2. Average Displacement Error

The average displacement error (ADE) is the metric that shows the average displacement between the predicted pedestrian trajectories $\hat{Y}$ and the ground-truth future trajectories Y. To compute this metric, the magnitude of the difference between ground-truth position and the prediction position is computed first, then the average is taken for all the

frames in a scene. The formula to compute ADE is defined as follows, in which the variable n denotes the number of pedestrians in the specified scene.

$$\mathrm{ADE}(Y, \hat{Y}) \triangleq \frac{1}{n} \cdot \frac{1}{t_{pred}} \sum_{i=1}^{n} \sum_{t=t_{\mathrm{obs}}+1}^{t_{\mathrm{obs}}+t_{\mathrm{pred}}} \|\mathbf{p}_i^t - \hat{\mathbf{p}}_i^t\| \tag{25}$$

The average displacement error (ADE) is the metric for single-scene evaluation. To evaluate the performance of the scenes in the dataset. We derive two metrics based on this metric, which are the mean ADE metric and the minimum ADE metric.

The mean ADE metric (meanADE) computes the average ADE for all sampled scenes in the specified dataset. Furthermore, in order to capture the performance of the stochastic models, we sample $N_{\mathrm{samples}} = 10$ trajectory predictions for a scene, and compute the average over all the samples as well.

$$\mathrm{meanADE} \triangleq \frac{1}{N_{\mathrm{scenes}}} \sum_{X,Y} \left(\frac{1}{N_{\mathrm{samples}}} \sum_{k} \mathrm{ADE}(Y, \hat{Y}^k) \right) \Bigg|_{\hat{Y}^k = \mathcal{G}(X,O)} \tag{26}$$

The minimum ADE metric (minADE) is a metric to capture the average minimum error among the samples. To compute this metric, first, the minimum ADE is evaluated for all prediction samples in each scene, and then the average is taken over the scenes. The computation formula for this metric is defined as follows:

$$\mathrm{minADE} \triangleq \frac{1}{N_{\mathrm{scenes}}} \sum_{X,Y} \left(\min_{k} \mathrm{ADE}(Y, \hat{Y}^k) \right) \Bigg|_{\hat{Y}^k = \mathcal{G}(X,O)} \tag{27}$$

The experiment result for the ADE metrics is shown in Table 4.

Table 4. Evaluation of ADE metrics.

	Training Set		**Testing Set**	
	meanADE	**minADE**	**meanADE**	**minADE**
Ground Truth	0 ± 0	0 ± 0	0 ± 0	0 ± 0
Constant Velocity	0.262 ± 0.088	0.262 ± 0.088	0.260 ± 0.086	0.260 ± 0.086
Social GAN	0.231 ± 0.071	$\mathbf{0.196 \pm 0.062}$	0.251 ± 0.079	$\mathbf{0.215 \pm 0.070}$
SFM (trained)	0.270 ± 0.085	0.270 ± 0.085	0.268 ± 0.083	0.268 ± 0.083
SFM (original)	0.468 ± 0.395	0.468 ± 0.395	0.469 ± 0.273	0.469 ± 0.273
Proposed (deterministic)	$\mathbf{0.218 \pm 0.087}$	0.218 ± 0.087	$\mathbf{0.215 \pm 0.084}$	0.215 ± 0.084
Proposed (stochastic)	0.331 ± 0.120	0.243 ± 0.084	0.329 ± 0.114	0.241 ± 0.079

3.2.3. Final Displacement Error

The final displacement error (FDE) is the metric that shows the average displacement in the last predicted frame between the predicted pedestrian trajectories $\hat{Y}$ and the ground-truth future trajectories Y. To compute this metric, the magnitude of the difference between the ground-truth position and the prediction position is computed first, then the average is taken for the last frame in the scene. The formula to compute FDE is defined as follows, in which the variable n denotes the number of pedestrians in the specified scene.

$$\mathrm{FDE}(Y, \hat{Y}) \triangleq \frac{1}{n} \sum_{i=1}^{n} \left(\|\mathbf{p}_i^t - \hat{\mathbf{p}}_i^t\| \right) \Bigg|_{t = t_{\mathrm{obs}} + t_{\mathrm{pred}}} \tag{28}$$

The final displacement error (FDE) is the metric for single-scene evaluation. To evaluate the performance of the scenes in the dataset. We derive two metrics based on this metric, which are the mean FDE metric and the minimum FDE metric.

The mean FDE metric computes the average FDE for all sampled scenes in the specified dataset. Furthermore, in order to capture the performance of the stochastic models, we sample $N_{\text{samples}} = 10$ trajectory predictions for a scene, and compute the average over all the samples as well.

$$\text{meanFDE} \triangleq \frac{1}{N_{\text{scenes}}} \sum_{X,Y} \left(\frac{1}{N_{\text{samples}}} \sum_{k} \text{FDE}(Y, \hat{Y}^k) \right) \Bigg|_{\hat{Y}^k = \mathcal{G}(X,O)} \tag{29}$$

The min ADE metric is a metric to capture the average minimum error among the samples. To compute this metric, first the minimum ADE is evaluated for all prediction samples in each scene, and then the average is taken over the scenes. The computation formula for this metric is defined as follows:

$$\text{minFDE} \triangleq \frac{1}{N_{\text{scenes}}} \sum_{X,Y} \left(\frac{1}{N_{\text{samples}}} \sum_{k} \text{FDE}(Y, \hat{Y}^k) \right) \Bigg|_{\hat{Y}^k = \mathcal{G}(X,O)} \tag{30}$$

The experiment result for the FDE metrics is shown in Table 5.

Table 5. Evaluation of FDE metrics.

	Training Set		**Testing Set**	
	meanFDE	**minFDE**	**meanFDE**	**minFDE**
Ground Truth	0 ± 0	0 ± 0	0 ± 0	0 ± 0
Constant Velocity	0.480 ± 0.169	0.480 ± 0.169	0.476 ± 0.165	0.476 ± 0.165
Social GAN	0.412 ± 0.138	$\mathbf{0.342 \pm 0.119}$	0.453 ± 0.154	$\mathbf{0.381 \pm 0.137}$
SFM (trained)	0.504 ± 0.164	0.504 ± 0.164	0.500 ± 0.160	0.500 ± 0.160
SFM (original)	0.887 ± 1.427	0.887 ± 1.427	0.887 ± 0.823	0.887 ± 0.823
Proposed (deterministic)	$\mathbf{0.402 \pm 0.172}$	0.402 ± 0.172	$\mathbf{0.395 \pm 0.164}$	0.395 ± 0.164
Proposed (stochastic)	0.594 ± 0.240	0.407 ± 0.164	0.590 ± 0.226	0.403 ± 0.155

3.2.4. Minimum Social Distance

The minimum social distance (MSD) is the metric that computes the minimum distance between the pedestrians among all predicted frames. The metric helps us grasp the idea of how the model handles and avoids social collisions.

$$\text{MSD}(\hat{Y}) \triangleq \min_{t,i,j \neq i} \|\hat{\mathbf{p}}_i^t - \hat{\mathbf{p}}_j^t\| \tag{31}$$

To further apply the metric for the scenes in the dataset. We derive three metrics based on this metric, these three metrics are the min MSD metric, the 5th% MSD metric, and the Social collision ratio (SCR) metric. To compute these metrics, the MSD metric for each scene in the dataset is computed first. Then, the minimum MSD metrics among all the MSD metrics are found and defined as the minimum MSD metric. The 5th% MSD metric is computed by taking the 5th% value of the percentile of the MSD metrics in the dataset. As for the Social collision ratio, it is defined as the ratio of collision between pedestrians in the dataset, we consider a scene has collision if the MSD metric for the scene is less than 20 cm.

The experiment result for the MSD metrics is shown in Table 6.

Table 6. Evaluation of MSD Metrics.

	Training Set			Testing Set		
	minMSD	**5th%MSD**	**SCR**	**minMSD**	**5th%MSD**	**SCR**
Ground Truth	0.204	0.266	0%	0.227	0.286	0%
Constant Velocity	0.006	0.110	12.3%	0.014	0.115	12.7%
Social GAN	0.003	0.176	6.4%	0.006	0.176	6.4%
SFM (trained)	0.086	0.350	0.7%	0.127	0.351	0.7%
SFM (original)	**0.106**	**0.583**	**0.2%**	**0.207**	**0.588**	**0.2%**
Proposed (deterministic)	0.033	0.312	0.9%	0.085	0.328	1.0%
Proposed (stochastic)	0.012	0.209	4.5%	0.021	0.211	4.6%

3.2.5. Minimum Physical Distance

The minimum physical distance (MPD) is the metric that computes the minimum distance between the pedestrians among all predicted frames. This metrics help us the grasp the idea how the model handle and avoid the physical collisions.

$$\mathrm{MPD}(\hat{Y}, O) \triangleq \min_{t,i,k} \|\hat{\mathbf{p}}_i^t - \mathbf{o}_k\| \tag{32}$$

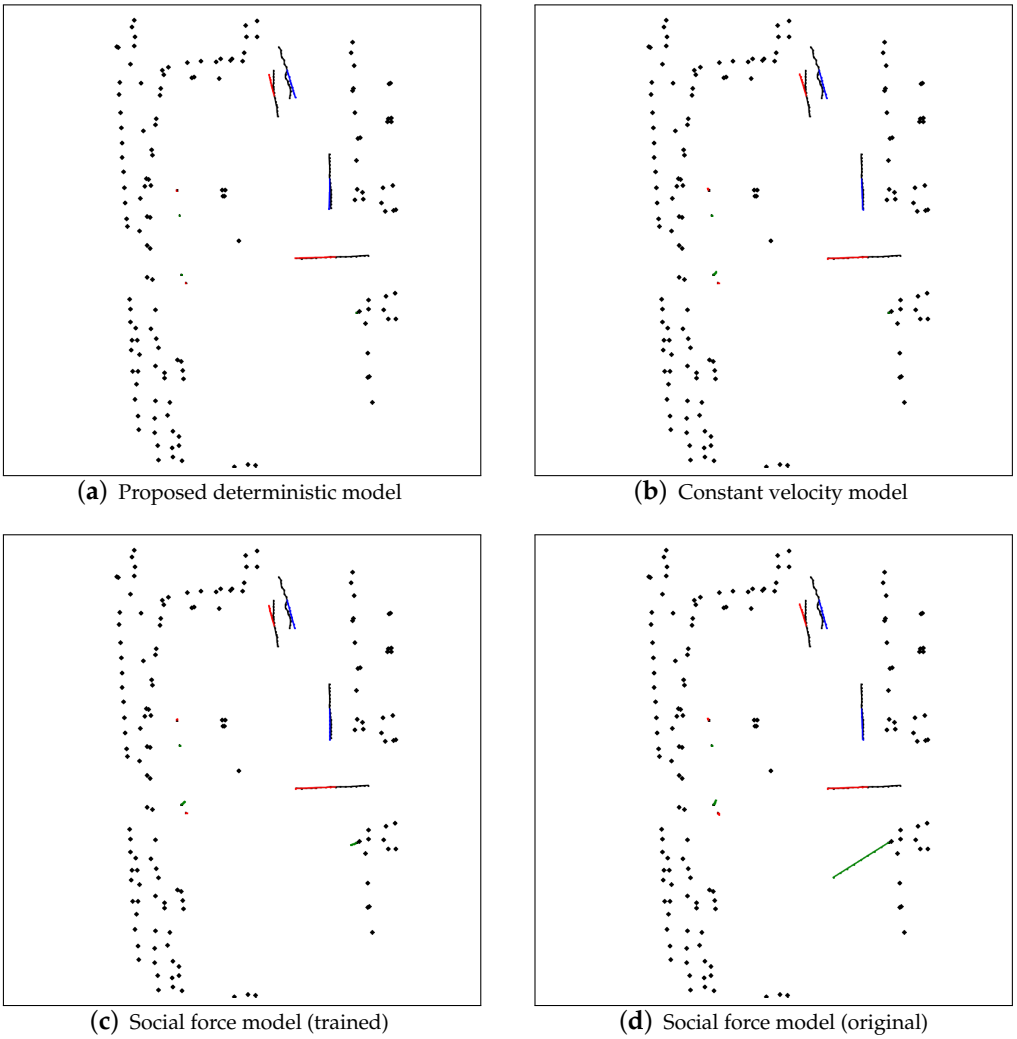
To further apply this metric to the scenes in the dataset. We derive three metrics based on this metric, these three metrics are the min MPD metric, the 5th% MPD metric, and the physical collision ratio (PCR) metric. To compute these metrics, the MPD metric for each scene in the dataset is computed first. Then, the minimum MPD metrics among all the MPD metrics are found and defined as the minimum MPD metric. The 5th% MPD metric is computed by taking the 5th% value of the percentile of the MPD metrics in the dataset. As for the physical collision ratio, it is defined as the ratio of collision between the pedestrian and the physical obstacles in the dataset, we consider a scene has a collision if the MPD metric for the scene is less than 20 cm. The experiment result for the MPD metrics is shown in Table 7.

Table 7. Evaluation of MPD metrics.

	Training Set			Testing Set		
	minMPD	**5th%MPD**	**PCR**	**minMPD**	**5th%MPD**	**PCR**
Ground Truth	0.319	0.472	0.0%	0.408	0.674	0.0%
Constant Velocity	0.023	0.382	1.4%	0.072	0.429	1.3%
Social GAN	0.014	0.405	0.9%	0.016	0.442	1.2%
SFM (trained)	0.073	0.491	0.3%	**0.219**	0.679	0.2%
SFM (original)	**0.102**	**0.623**	0.2%	0.188	0.619	0.2%
Proposed (deterministic)	0.072	0.477	**0.2%**	0.167	**0.697**	**0.2%**
Proposed (stochastic)	0.011	0.401	0.8%	0.030	0.527	0.8%

3.3. Graphical Results

Figure 4 shows the predicted trajectories and comparison results with the ground-truth trajectories for deterministic models, including our proposed model, constant velocity model and the social force model [11] in two configurations. The predicted trajectories of the pedestrians in this scene are lines colored either red, green, or blue for every individual. The ground-truth trajectories are represented with black lines. The black dots mark the locations of the physical obstacles in the experiment environment. Each figure shows the 60 by 60 square meters area and is represented in the resolution of 600 $\times$ 600 pixels.



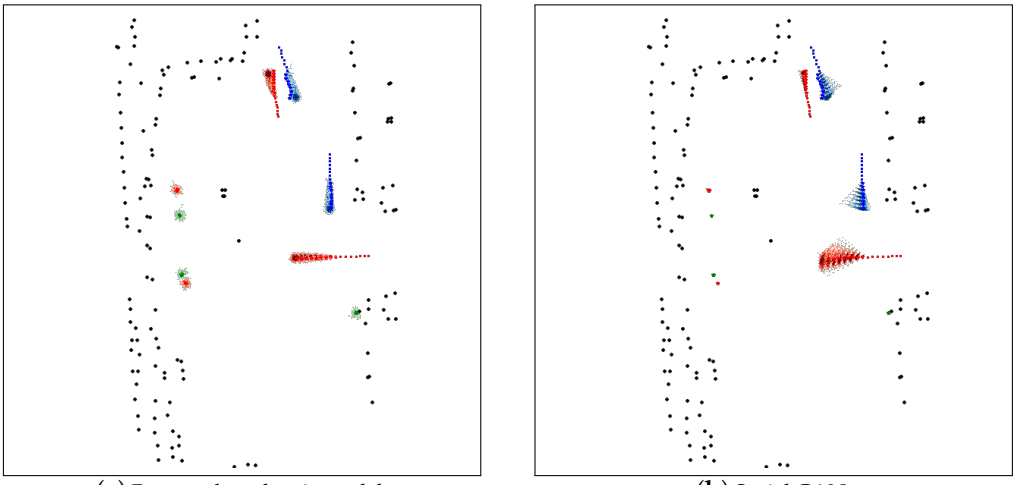
(**a**) Proposed deterministic model (**b**) Constant velocity model

(**c**) Social force model (trained) (**d**) Social force model (original)

Figure 4. Graphical result of the compared deterministic models.

Figure 5 shows the predicted trajectories distribution compared with the ground-truth trajectories for two stochastic models. The trajectories of the pedestrians in this scene are colored either red, green, or blue for every individual. The predicted trajectories distribution is generated by stacking 1000 predictions into the 2D histogram and is represented using the heatmap. The ground-truth trajectories are represented with solid dots. The black dots mark the locations of the physical obstacles in the experiment environment. Each figure shows the 60 by 60 square meters area and is represented in the resolution of 600 $\times$ 600 pixels.

These figures demonstrate that the proposed model can predict in a bimodal manner, i.e., the pedestrian prefers to maintain his/her position in static mode while moving with the consideration of the environment in the moving mode. When compared with other deterministic models such as constant velocity and the social force model, we see those models often suffers drifts for the pedestrian which is supposed to standing still at the same spot, as shown by those 4 pedestrians in the left and the pedestrian at the right-bottom by an obstacle in Figure 4. Furthermore, the ground-truth trajectories fall in the predicted distribution for both our proposed stochastic model and the social GAN model [6], which showcase the prediction coverage of the models.



(**a**) Proposed stochastic model (**b**) Social GAN

Figure 5. Graphical result of the compared stochastic models.

4. Discussion

From the experimental result in the previous section, we could see that our traditional model-based model uses slightly more parameters than the social force model [11], as we introduce the bimodal extended Kalman filtering mechanism in addition to the social force model. Nevertheless, our model uses much fewer parameters than that of the deep learning-based model such as the social GAN [6] model.

The run-time of the proposed models is longer than that of all the other compared models, which is mainly due to the following three reasons. First, our model has an encoding step, which processes the observed trajectories and estimates the states of the pedestrians. This cause our model appears slower than that of the constant velocity model and social force model [11]. Second, our model considers and computes the social and physical interaction on a frame-by-frame basis. When compared with the social GAN [6], which only computes the social interaction feature once per trajectory. Thus, our frame-by-frame approach requires extras computations more than those that do not. Finally, our model requires the computation of Jacobian matrices for approximation during the encoding phase, which is also a time-consuming operation compared with normal feedforward operations.

For the performance of the ADE and FDE metrics. Our proposed deterministic model outperforms all compared models on the mean ADE metric and the mean FDE metric. Our proposed models also outperform all compared models except for the social GAN [6] model, when comparing the minimum ADE metric and the minimum FDE metric. By considering the bimodal moving nature of the pedestrian with the proposed bimodal extended Kalman filter, our proposed model shows the improvement in performance when compared with the constant velocity model and the social force model as they do not utilize such property. Regarding the metrics compared with the social GAN [6] model, such a result is reasonable, as social GAN [6] is optimized on its variety loss. Optimizing the variety loss will encourage the variety of the predicted trajectories while keeping the minimum displacement error as small as possible. However, such loss might also cause the mean ADE to be larger due to the diversity of the prediction.

As for the performance of the MSD and MPD metrics, our proposed models and the social force model [11] perform better than that by the constant velocity model and the social GAN model for all the MSD and MPD metrics while remaining competitive with that by the social force model. The proposed models have larger distances in terms of MSD and MPD metrics and lower collision ratio values. These metrics show that our model does gain the social avoidance ability thanks to the social force model [11] being applied in the moving mode, while the original social force model performs the best in these metrics, it is

reasonable as the model is configured to have a larger repulsive force for both the social and physical obstacles.

5. Conclusions

We build a trajectory dataset from the point-cloud data collected by the 3D LiDAR sensor in a semi-automatic labeling manner and proposed a bimodal extended Kalman filter to model the bimodal moving nature of the pedestrian. We then adopt the social force model [11] as the motion model for the pedestrian in moving mode, and the constant position model for the pedestrian in the static mode. With this approach, the proposed model shows increased performance in ADE and FDE metrics than of those models which are not aware of the bimodal motion properties. Furthermore, our model inherits the ability from the social force model to avoid collisions with pedestrians and physical obstacles. In addition, our proposed approach is fully explainable as comparing with those deep learning-based model, which justifies the usefulness of the proposed approach.

Author Contributions: Conceptualization, C.-Y.L. and L.-J.K.; methodology, C.-Y.L.; software, C.-Y.L.; validation, C.-Y.L. and L.-J.K.; formal analysis, C.-Y.L. and L.-J.K.; investigation, C.-Y.L.; resources, C.-Y.L. and L.-J.K.; data curation, C.-Y.L.; writing—original draft preparation, C.-Y.L.; writing—review and editing, L.-J.K.; visualization, C.-Y.L.; supervision, L.-J.K. and C.-Y.C.; project administration, L.-J.K.; funding acquisition, L.-J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Science and Technology Council, Taiwan, under Grant MOST 109-2221-E-027-086-MY2.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are openly available at https://github.com/cylin-at-research/ntut_library (last accessed on 23 October 2022).

Acknowledgments: The authors would like to thank Berkeley DeepDrive for its great support in many aspects.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LSTM	Long Short-Term Memory
GAN	Adversarial Generative Network
SFM	Social Force Model
MOT	Multi-Object Tracking
LiDAR	Light Detection Furthermore, Ranging
ROS	Robotic Operating System
ADE	Average Displacement Error
FDE	Final Displacement Error
MSD	Minimum Social Distance
MPD	Minimum Physical Distance
SCR	Social Collision Ratio
PCR	Physical Collision Ratio

References

1. Kato, S.; Takeuchi, E.; Ishiguro, Y.; Ninomiya, Y.; Takeda, K.; Hamada, T. An Open Approach to Autonomous Vehicles. *IEEE Micro* **2015**, *35*, 60–68. [CrossRef]
2. Phillips, M.; Likhachev, M. SIPP: Safe interval path planning for dynamic environments. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shangai, China, 9–13 May 2011; pp. 5628–5635.
3. Zhou, X.; Ren, H.; Zhang, T.; Mou, X.; He, Y.; Chan, C.-Y. Prediction of Pedestrian Crossing Behavior Based on Surveillance Video. *Sensors* **2022**, *22*, 1467. [CrossRef] [PubMed]

4. Wang, Z.; Li, M.; Lu, Y.; Bao, Y.; Li, Z.; Zhao, J. Effective multiple pedestrian tracking system in video surveillance with monocular stationary camera. *Expert Syst. Appl.* **2021**, *178*, 114992 [CrossRef]
5. Schubert, R.; Richter, E.; Wanielik, G. Comparison and evaluation of advanced motion models for vehicle tracking. In Proceedings of the International Conference on Information Fusion, Cologne, Germany, 30 June–3 July 2008; pp. 1–6.
6. Gupta, A.; Johnson, J.; Fei-Fei, L.; Savarese, S.; Alahi, A. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 2255–2264.
7. Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L.; Savarese, S. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 961–971.
8. Zhang, P.; Ouyang, W.; Zhang, P.; Xue, J.; Zheng, N. Social Ways: Learning Multi-Modal Distributions of Pedestrian Trajectories With GANs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 16–17 June 2019; pp. 2964–2972.
9. Amirian, J.; Hayet, J.-B.; Pettré, J. SR-LSTM: State Refinement for LSTM towards Pedestrian Trajectory Prediction. In Proceedings of the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–17 June 2019; pp. 12085–12094.
10. Huang, Y.; Bi, H.; Li, Z.; Mao, T.; Wang, Z. STGAT: Modeling Spatial-Temporal Interactions for Human Trajectory Prediction. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 6272–6281.
11. Dirk, H.; Peter, M. Social Force Model for Pedestrian Dynamics. *Phys. Rev. E* **1998**, *51*, 4282–4286.
12. Pellegrini, S.; Ess, A.; Schindler, K.; Van Gool, L. You'll never walk alone: Modeling social behavior for multi-target tracking. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Kyoto, Japan, 27 September–4 October 2009; pp. 261–268.
13. Scovanner, P.; Tappen., M. Learning pedestrian dynamicsfrom the real world. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Kyoto, Japan, 27 September–4 October 2009; pp. 381–388.
14. Yamaguchi, K.; Berg, A.C.; Ortiz, L.E.; Berg, T.L. Who are you with and where are you going? In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Colorado Springs, CO, USA, 20–25 June 2011; pp. 1345–1352.
15. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
16. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. *Commun. ACM* **2014**, *63*, 139–144. [CrossRef]
17. Häselich, M.; Jöbgen, B.; Wojke, N.; Hedrich, J.; Paulus, D. Confidence-based pedestrian tracking in unstructured environments using 3d laser distance measurements. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 4118–4123.
18. Leal-Taixé, L.; Fenzi, M.; Kuznetsova, A.; Rosenhahn, B.; Savarese, S. Learning an Image-Based Motion Context for Multiple People Tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014; pp. 3542–3549.
19. Pellegrini, S.; Ess, A.; Van Gool, L. Improving Data Association by Joint Modeling of Pedestrian Trajectories and Groupings. In Proceedings of the 11th European Conference on Computer Vision, Heraklion, Greece, 5–11 September 2010; pp. 452–465.
20. Biber, P.; Straßer, W. The Normal Distributions Transform: A New Approach to Laser Scan Matching. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, 27 October–1 November 2003; pp. 2743–2748.
21. Besl, P.J.; McKay, N.D. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 239–256. [CrossRef]
22. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*; The MIT Press: Cambridge, MA, USA, 2005; pp. 54–61.
23. Kálmán, R.E. A New Approach to Linear Filtering and Prediction Problems. *J. Basic Eng.* **1960**, *82*, 35–45. [CrossRef]
24. de Boor, C. On calculating with b-spline. *J. Approx. Theory* **1972** ,*6*, 50–62. [CrossRef]
25. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference for Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015;