CS5340 Final Report

Pedestrian Trajectory Prediction and Crossing Pedestrian Detection

Li Xiaochen
A0231442U
e0703474@u.nus.edu

Zhi Jun
A0243717A
e0833564@u.nus.edu

Shi Yuang
A0229927N
e0686126@u.nus.edu

Wang Binluo
A0224877R
e0575511@u.nus.edu

Abstract

To increase the comforts and safety of the driver and the passengers, and avoid accidents, human activity detection and analysis attracts increasing number of researchers in self-driving domain. In this project, we aim to build a pedestrian behavior probabilistic model to predict whether a pedestrian crosses the road or not. We propose two algorithms, Hidden Markov Model (HMM) and Kalman Filter (KL), for trajectory measurement and predictions, and pedestrian crossing detection. We conduct experiments on a large-scale walking pedestrians dataset and analysis the results of HMM and KL in details.

1 Motivation

In the global background, human activity detection and analysis is used for wide-range applications, including surveillance systems, environment understanding, robotics application, content-based image retrieval, video annotation, assisted living, intelligent vehicles, and advanced user interfaces [1]. Pedestrian detection is an indispensable part of self-driving cars and vision-based driver-assisted systems. To avoid collisions and guarantee the passengers' and drivers' experience, when the probability of a pedestrian crossing the road is small enough, the car will not stop and give way to the pedestrian. To this end, our system aims to ensure the least amount of stopping of self-driving cars on the condition of safety, reduce the times of passengers waiting and improve passengers' experience. To this end, our objective is to design a mode which can effectively forecast the future trajectory of the pedestrian present in a scene, and then detect the pedestrian crossing the street.

2 Methodology

2.1 Problem Statement

Figure 1 shows the scenario of our task. The model takes as input the trajectory of the person in a scene denoted by $\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n\}$. The position and velocity of the pedestrian at time-step t is denoted by $\mathbf{x}^t = (x^t, y^t)$ and $\mathbf{v}^t = (v_x^t, v_y^t)$. Our task is to forecast the corresponding future trajectory $\mathbf{Y}$, and then classify whether the pedestrian intends to cross the street. Specifically, we receive the positions of the pedestrian at time-steps $t = 1, \ldots, T_{obs}$ and want to forecast the future positions from time-steps $t = T_{obs} + 1$ to T_{pred}. We denote our predictions using $\hat{\mathbf{Y}}$. According to the predictions, we then detect the pedestrian as a pedestrian crossing road or not with probability p_c.

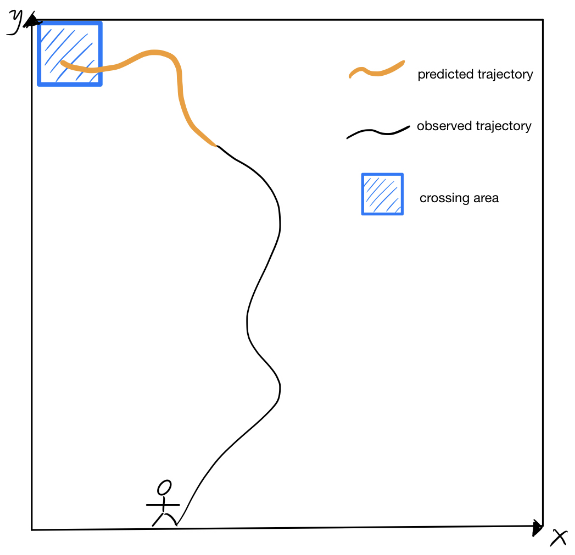


Figure 1: Example of the proposed problem. The goal is to predict the trajectory of the pedestrian given the current observed trajectory and detect whether the pedestrian intends to cross the road. The pedestrian will be regarded as a pedestrian crossing road when he/she reaches the crossing area.

2.1.1 Crossing Detection Under Uncertainty

We will design experiments to predict the possible continuous position of pedestrians in the future, and add prediction error to describe the possible position distribution of pedestrians. Limited by the constraints of aleatory uncertainty and epistemic uncertainty, our model will produce relative uncertainty in the process of training. Therefore, by estimating each prediction point on the future trajectory and combining the model variance, we can draw a group of ellipses with different radius, as shown in Fig 2.

We can see that with the growth of prediction time, the area of ellipse becomes larger and larger. Therefore, our goal is to make the two groups of models reduce the error of the actual prediction process by constantly observing the data in the training process, so as to make the prediction of the model more and more accurate. During the experiment, we counted the last step of pedestrian trajectory prediction. If the last step of prediction happens to fall in the square specified by us, we can assume that the probability of pedestrians passing through the road is 100%. We describe the accuracy of our model in predicting pedestrian intention by comparing the actual number of people crossing the road in the data with the predicted number of people crossing the road in our model. And compare the gap between the two groups of models in prediction performance.

2.2 Trajectory Prediction and Crossing Detection

We consider two methods for trajectory prediction: (1) Hidden Markov Models (HMM) [2], and (2) Kalman Filter (KF) [3]. HMM is a Markov model with hidden state that can be used to track processes with hidden state. KL is a recursive real time algorithm that can calculate and adjust the predicted measurement according to the belief of the algorithm.

2.2.1 Hidden Markov Models

For a system which internal factors can not be observed directly but the events depend on the factors can be observable, the Hidden Markov Models can be used to describe it. Basically, the observed event in the Hidden Markov Models will not be corresponding to its step-by-step status but related to a set of probability distributions. So the Hidden Markov model is a probabilistic model which is used to derive the probabilistic characteristic of any random process. Besides, the Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a

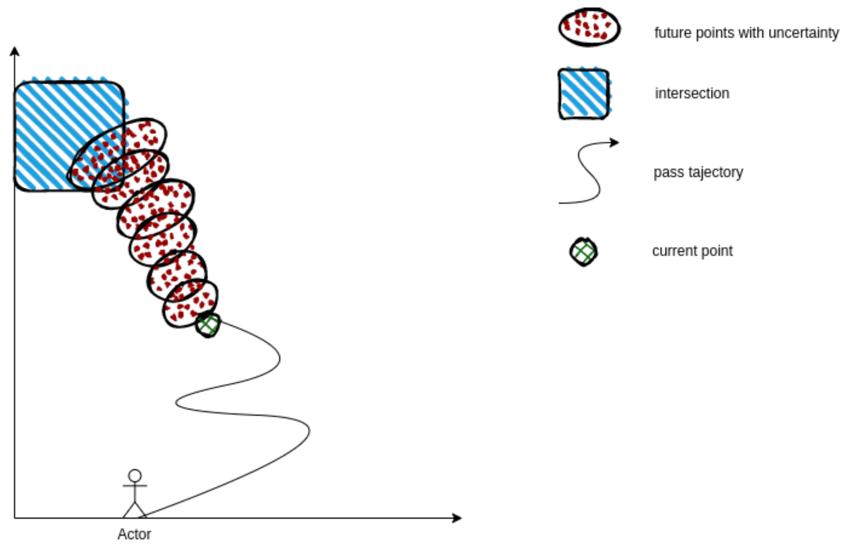


Figure 2: Mechanism for crosswalk probability

Markov process with hidden states X. However, the process Y whose outcomes are "influenced" by the outcomes of X is observable. So in HMM, our goal is to learn about X by observing Y.

The HMM flow chart is shown as below.

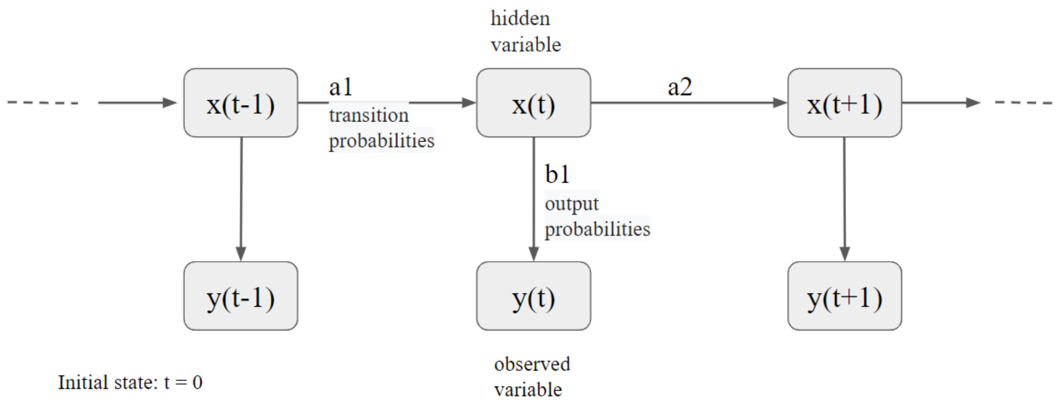


Figure 3: HMM updating flow chat

In the Hidden Markov Model, let X_n and Y_n be discrete-time stochastic processes and $n \geq 1$. The pair (X_n, Y_n) is a hidden Markov model if

$$\mathbf{P}\left(Y_n \in A \mid X_1 = x_1, \ldots, X_n = x_n\right) = \mathbf{P}\left(Y_n \in A \mid X_n = x_n\right) \tag{1}$$

where for every $n \geq 1$, $x_1, \ldots, x_n$ and every Borel set A. And $X/_n$ is a Markov process whose behavior is not directly observable ("hidden");

HMM also applies to continuous-time system. Let X_t and Y_t be continuous-time stochastic processes. The pair (X_t, Y_t) is a hidden Markov model if

$$\mathbf{P}\left(Y_{t_0} \in A \mid \{X_t \in B_t\}_{t \leq t_0}\right) = \mathbf{P}\left(Y_{t_0} \in A \mid X_{t_0} \in B_{t_0}\right) \tag{2}$$

where for every t_0, every Borel $setA$, and every family of Borel sets $\{B_t\}_{t \leq t_0}$, and same to the discrete-time's, X_t is a Markov process with hidden behavior.

The Hidden Markov Model has three pivotal problems, the filtering, the smoothing and the decoding problem. The Filtering task is to compute, given the model's parameters and a sequence of observations, the distribution over hidden states of the last latent variable at the end of the sequence. The smoothing task is similar to filtering, but asks about the distribution of a latent variable somewhere in the middle of a sequence. The decoding task, unlike the previous two, requires finding a maximum over all possible state sequences.

In our experiment, the hidden variable is the speed of the pedestrian in the x, y direction, the observed variable is x_hat, y_hat, which is also the Pedestrian trajectory.

There are two major limitations of HMM in our task. First, we need to discretize the problem into finite sets of states and actions for HMM, which results in discretized trajectories. Such trajectories cannot well fit the trajectories in the real world. Second, we ignore the noise from data sampling and pre-processing, which may leads to bias in learning and prediction phase. Therefor, we consider KF as a comparable method for trajectory prediction. KF is regarded as analogous to HMM, with the difference that the hidden state variables have values in a continuous space as opposed to a discrete state space as for the hidden Markov model. Also, KF can filter noise effectively, thus predicting smoother trajectory compared to the prediction of HMM.

2.2.2 Kalman Filter

An article published in 1960 by Kalman [3], demonstrated a new algorithm that provides estimation on unknown variables given measurements(prone to uncertainty) over time. It is a recursive real time algorithm that can calculate and adjust the predicted measurement according to the belief of the algorithm or, AKA, the Kalman gain. In this project, the team could use it to predict the trajectory of passengers using predicted and measured values, take into account of various uncertainty constraints and map out the best predictions. In this subsection, we would look at the online updating algorithm that estimates states based on linear dynamical systems in a state space with each step. Fig. 4 shows the flow of the Kalman Filter.

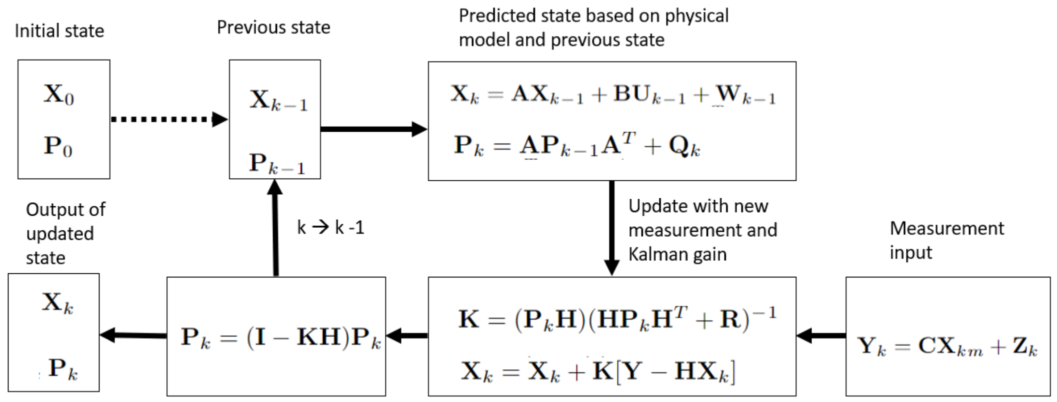

Initial state
$\mathbf{X}_0$
$\mathbf{P}_0$
Previous state
$\mathbf{X}_{k-1}$
$\mathbf{P}_{k-1}$
Predicted state based on physical model and previous state
$\mathbf{X}_k = \mathbf{A}\mathbf{X}_{k-1} + \mathbf{B}\mathbf{U}_{k-1} + \mathbf{W}_{k-1}$
$\mathbf{P}_k = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}_k$
Update with new measurement and Kalman gain
Measurement input
$\mathbf{Y}_k = \mathbf{C}\mathbf{X}_{km} + \mathbf{Z}_k$
$\mathbf{K} = (\mathbf{P}_k\mathbf{H})(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$
$\mathbf{X}_k = \mathbf{X}_k + \mathbf{K}[\mathbf{Y} - \mathbf{H}\mathbf{X}_k]$
$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$
k → k -1
Output of updated state
$\mathbf{X}_k$
$\mathbf{P}_k$

Figure 4: Kalman Filter updating flow chat

The time update equation can be broken down into five components [4], i.e. (i) State Prediction, (ii) Error Covariance Prediction, (iii) Kalman Gain, (iv) State Update and (v) Error Covariance Update.

State prediction can be formulated as

$$\mathbf{X}_k = \mathbf{A}\mathbf{X}_{k-1} + \mathbf{B}\mathbf{U}_{k-1} + \mathbf{W}_{k-1} \tag{3}$$

where $\mathbf{A}$ is the state transition matrix, and $\mathbf{B}$ is a control-input matrix. The state vector $\mathbf{X}$ typically consist of a position and vector matrix denoted by $\mathbf{p}$ and $\mathbf{v}$, respectively. While $\mathbf{U}$ is a control variable matrix that represents any object accelerations, gravity etc, if any. $\mathbf{W}$ is the predicted state noise matrix with an assumed zero mean Gaussian with covariance $\mathbf{Q}$. Typically represented in the format of $\mathbf{w}_{k-1}\ \mathbf{N}(0, \mathbf{Q})$.

We then formulate the error covariance prediction as

$$\mathbf{P}_k = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}_k \tag{4}$$

where $\mathbf{A}$ is a state transition matrix and $\mathbf{Q}$ is a process noise covariance matrix. Both Eq. 3 and Eq. 4 are the representation of predicted state space.

We represent Kalman gain by the equation below

$$\mathbf{K} = (\mathbf{P}_k\mathbf{H})(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1} \tag{5}$$

where the weight of the equation depends heavily on the $\mathbf{R}$ matrix, that is the measurement of noise covariance. A higher valued $\mathbf{R}$ would result in a low valued $\mathbf{K}$. The lower $\mathbf{K}$ is, the more stable the algorithm think the predictions are and put less weight on the measurements, which would in-turn trust the prediction more, vice versa, and eventually affect the updating process in Eq. 6 and Eq. 7. In this two part, the algorithm would decide if it would weight more on the measurement or the prediction based on the Kalman gain calculated in Eq. 5.

We then update the state prediction proportional to the value $\mathbf{K}$, where a low $\mathbf{K}$ means high uncertainty in measurement and should only take a small portion of the measurement value - $\mathbf{Y}$ to update. The equation of state update is shown below

$$\mathbf{X}_k = \mathbf{X}_k + \mathbf{K}[\mathbf{Y} - \mathbf{H}\mathbf{X}_k] \tag{6}$$

We update error covariance in following step

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k \tag{7}$$

Eq. 7 has a similar updating property as Eq. 6, where $\mathbf{I}$ is an identity matrix that subtracts $\mathbf{K}$ which would boost in the error/uncertainty correction if $\mathbf{K}$ is low and converge if $\mathbf{K}$ is high.

In our actual implementation, however, we are going to ignore $\mathbf{W}$, $\mathbf{Q}$ and $\mathbf{Z}$ for simplicity and also because they are the error associated with the actual process of calculating the matrices. For example, measurement delays etc.

Algorithm 1 shows the procedure of KL. $\mathbf{A}$ is the identity matrix, ΔP_x is the coordinate error in the process covariance matrix, ΔP_v is the velocity error in the process covariance matrix, ΔX is the coordinate observation error, ΔXV is the velocity observation error. Two outer loops is performed, one for each coordinate, e.g., x and y. We assumed an equal uncertainty/error in both the x and y direction, thus, the value of ΔP_x,ΔP_v,ΔX,,ΔXV is shared for both the x and y coordinate.

Algorithm 1 Kalman Filter

```
Initial: $\mathbf{X}_{k-1} = [\mathbf{X}_0\mathbf{V}_0], \mathbf{P}_{k-1} = [[0.64, 0][0, 0.64]]$,
         $u_k = 0, \mathbf{Q} = 0, \mathbf{W} = 0, \mathbf{Z} = 0, \mathbf{A} = \Phi$,
         $\Delta\mathbf{P}_x = 0.8, \Delta\mathbf{P}_v = 0.8, \Delta\mathbf{X} = 0.5, \Delta\mathbf{XV} = 1.5$
 1: for $I = 1, 2, \ldots$ do
 2:     for $J = 1, 2, \ldots, N$ do
 3:         $\mathbf{X}_k, predicted\_state \Leftarrow Eq.3(\mathbf{X}_{k-1}, \Delta\mathbf{P}_x, \Delta\mathbf{P}_v)$
 4:         $predicted\_pcm \Leftarrow Eq.4(\mathbf{A}, \Delta\mathbf{P}_{k-1}, \mathbf{Q})$
 5:         $kalman\_gain \Leftarrow Eq.5(predicted\_pcm, \Delta\mathbf{P}_x, \Delta\mathbf{P}_v)$
 6:         $new\_observation \Leftarrow NewObservation(iterationJ, \Delta\mathbf{X}_k, \Delta\mathbf{X}, \Delta\mathbf{XV})$
 7:         $\mathbf{X}_k \Leftarrow Eq.6(\mathbf{X}_k, kalman\_gain, new\_observation)$
 8:         $\mathbf{P}_k \Leftarrow Eq.7(kalman\_gain, predicted\_pcm)$
 9:         $\mathbf{X}_{k-1} \Leftarrow \mathbf{X}_k$
10:         $\mathbf{P}_{k-1} \Leftarrow \mathbf{P}_k$
11:     end for
12: end for
```

During implementation, we tried to train our model by taking all the trajectory data in the data set as a whole. However, we find that because the trajectory of each pedestrian has its characteristics, if all pedestrian data are used to learn our predicting model, the learned state transition matrix cannot be generalized. The state transition matrix tends to be overfitted with the majority proportion of trajectories with the same characteristics, so some trajectories cannot be predicted in the right direction. Therefore, to avoid such problems, we learn each pedestrian trajectory separately in the process of off-line iteration, that is, each pedestrian has its parameters such as state transition matrix, emission matrix, noise variance, and mean value. The parameters of different pedestrians are not shared.

In practice, we initialize the above parameters and iterate the pedestrian trajectories in the data set in turn. We initially set the emission matrix and state transition matrix with NumPy matrices in a dimension of two by two with ones on the diagonal, and zeros in other entries. We set the model variance, process noise, and model noise as random numbers with 2×2 between 0 and 1 respectively, and finally set the mean as a matrix composed of random numbers directly and uniformly sampled between 0 and 1 inclusively with 2×1. In the iterative process, we use the current parameters to derive the Kalman constant K throughout the forward propagation process and update our model parameters μ_t and v_t. Then we smooth the mean and variance of our model by backward propagation to re-estimate and maximize the conditional probability of our hidden states. In the EM process, we calculate the expectations of Z_t, $\{Z_t, Z_{t-1}\}$ and $\{Z_t, Z_t\}$ under the conditional probability respectively, and update our parameter matrices through the Q function, to obtain the hidden state trajectory of pedestrians, that is, the real trajectory of pedestrians. At this time, the learned state transition matrix $\mathbf{A}$ and emission matrix $\mathbf{C}$ are the parameters we use to predict the pedestrian trajectory, and the parameter μ is the real pedestrian trajectory.

3 Evaluation

3.1 Experiment Data and Settings

3.1.1 Dataset

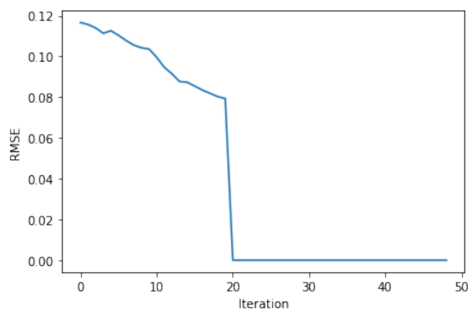
For the dataset, we use ETH Walking Pedestrians Dataset[5], which is extensively used in Human Trajectory Prediction literature. To compare the accuracy of model training and ensure the relative linearity of pedestrian trajectories, we select that the data path of pedestrians with steps between 30 and 40 conforms to the linear relationship. Due to the directionality of pedestrian paths, to unify the movement direction of pedestrians, we specify that the paths of pedestrian tracks along the x-axis from large to small is the positive direction of pedestrians' walking. Because of the symmetry of the intersection, we count the number of pedestrians crossing the road according to the real intentions of pedestrians in the video, and under the condition of following this provision, we symmetrically reversed the pedestrian trajectories with negative directions and combined the pedestrian trajectories with positive directions together to form our final data set. We drew an area in the coordinate as our intersection area according to the actual structure of the roads, the number of pedestrians stepping into this area at the end of their trajectories data is calculated as the number of people having the intention to cross the road.

3.1.2 Validation

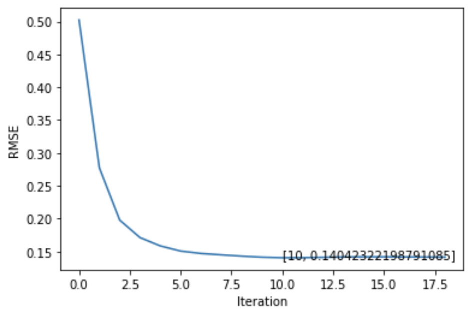
We adopt 5-fold cross-validation given that the size of the training set is small. Specifically, the training set is divided into 5 blocks, and each block is used as the validation set in turn. Eventually, we calculate the average accuracy of the validation set for each model under 5 times of training and select the model with the highest average accuracy. Any resulting model's function in the 5 times of training results is then trained through the testing set to obtain the final model.

Trajectory Prediction When we measure the accuracy of a predicted trajectory, we are comparing the sum of error (RMSE) between the real trajectory generated by our smoothed hidden states and the trajectories we predict through the state transition matrix and the emission matrix. Also, to assert the accuracy of predicted intentions we calculate the differences between the number of the predicted intentions with the number of actual intentions in crossing the intersection, denoted by the red box as mentioned above.

To measure the accuracy of the above, we predict the future trajectory of each pedestrian from step 20, calculate the hidden state of the pedestrian from step 20 by iterating the number of remaining trajectory points of the pedestrian and multiplying it by the state transition matrix, and finally get our predicted trajectory for the current pedestrian through the emission matrix. In particular to Kalman Filter, we record the variance of our transition matrix to show the uncertainty of our model in process of learning and draw the uncertainty area as shown in Fig. 2.



(a) The changing trend of RMSE for HMM.



(b) The changing trend of RMSE for KL.

Figure 5: Changing Trend of RMSE of HMM and KL for Trajectory Prediction.

Crossing Detection We draw the receiver operating characteristic (ROC) curve by plotting true positive rate (TPR) the classifiers positive rate (FPR) as functions of the threshold t for the decision boundary, for classifier $p(y = 1|x, w) \geq t$, to analyze the best decision boundary. The larger the area under the curve (AUC) is, the better the performance of the classifier is. If there are lots of negative samples, precision is preferred over FPR.

3.2 Trajectory Prediction Results

For evaluation, we use RMSE to evaluate our KF model point-to-point. We calculate the derivation between each predicted point and our true underlying point. Fig. 5 shows the trend of RMSE of HMM and KL. We can find that HMM converges after 20 iterations while KL requires about 10 iterations for convergence, which demonstrates that KL is more efficient than HMM during training. Besides, the trend of RMSE of HMM is not stable compared to that of KL, indicating that KL is more robust on the task of trajectory prediction. In conclusion, the results show that KF can better explore the potential relationship between linear data in the prediction process with better robustness.

3.3 Crossing Classification Results

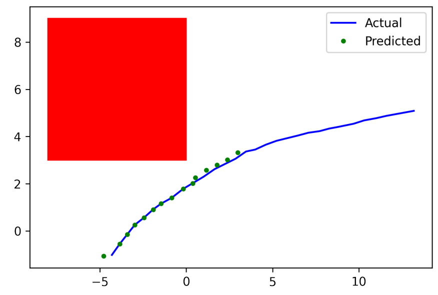
In order to digitally measure the accuracy of our model, we calculated AUC and Precision values respectively according to the actual label and prediction label of the model. As shown in Table 1, by comparing the actual label and predicted label of pedestrians' intention, HMM achieved AUC and precision of 0.8571 abd 0.9285, respectively, while KL got about 10% and 0.6% improvement on precision and AUC (i.e., 0.9380 on precision and 0.9340 on AUC) when compared with HMM, demonstrating that KL outperforms HMM in crossing detection task.

Table 1: Comparison Performance of HMM and KL for Crossing Detection

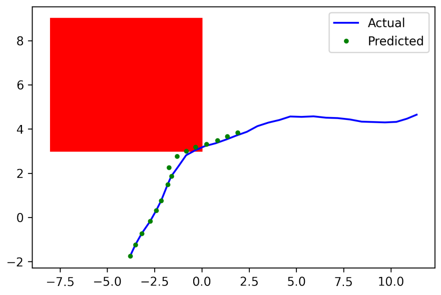
Model	Precision	AUC
HMM	0.8571	0.9285
KL	**0.9380**	**0.9340**

3.4 Visualization of Crossing Pedestrian Detection

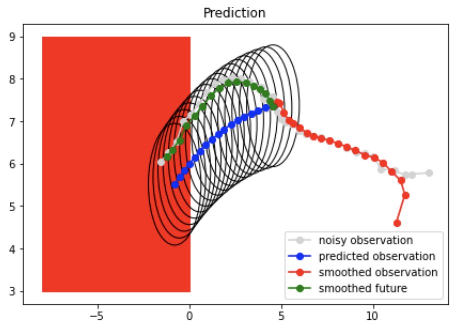
We further conduct visualization of trajectory prediction and Fig. 6) shows the results. Fig. 6(a) and Fig. 6(b) portray the prediction of HMM, where the red dots represent the predictions and the blue lines represent the ground-truth. As can be observed, the predicted results are close to the actual trajectory, showing that HMM performs well on trajectory prediction. The results of KL are shown in Fig. 6(c) and Fig. 6(d) which reveal the uncertainty of trajectory prediction. We use green lines to represent the real pedestrian trajectory from t_{20}, red to represent the real trajectory before t_{20}, and blue dots to represent the pedestrian trajectory predicted. The ellipse in the shadow represents the uncertainty of the overall prediction. It can be observed that in the process of each iteration, prediction



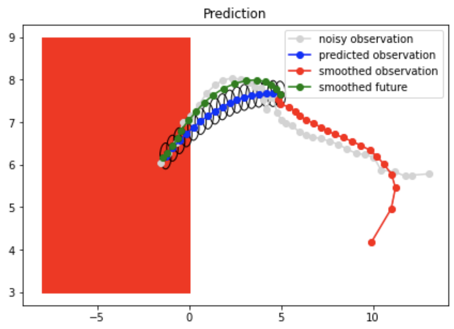
(a) Prediction of HMM for a random pedestrian.



(b) Prediction of HMM for a random pedestrian.



(c) Prediction of KL for a random pedestrian under uncertainty.



(d) Prediction of KL for a random pedestrian under uncertainty.

Figure 6: Visualization of Trajectory Prediction.

uncertainty is gradually decreasing, and because we learn pedestrian trajectory parameters through the complete data of a single pedestrian trajectory, there is no need to iteratively update our transition matrix and emission matrix by adding new path points. Therefore, in the process of each iteration, we predict that the uncertainty of each path point in the future corresponds to a consistent variance, and this variance will continue to be decreased each iteration of offline learning and reflected in the prediction uncertainty. That is, the model will update and modify various parameters in KF with the continuous iteration of the trajectory data of a single pedestrian. The reason for designing in this way is that we can simultaneously observe the model's estimation of the real pedestrian trajectory and the prediction of the future trajectory. So that it helps us visualize the update status and compare the relationship between our predicted trajectory, real trajectory, and model uncertainty in each iteration of learning parameters until the model converges.

4 Conclusion

In this project, we propose Hidden Markov Model (HMM) and Kalman Filter (KL) for trajectory predictions and crossing pedestrian detection. HMM is a Markov model with hidden state which can be utilized to track the pedestrian trajectory with hidden state. Given the limitations of HMM which can hardly handle the noise and is impossible to process hidden state in continuous space, HMM is regarded as a naive method. On the contrary, KL is proposed to resolve the issues of HMM. To compare the performance and give quantitative analysis of HMM and KL, we conduct several experiments on ETH Walking Pedestrians Dataset for these two methods. We also visualize the uncertainty of trajectory prediction of KL. The experiment results show that KL is more effective and efficient than HMM both in trajectory prediction task and crossing pedestrian detection task.

References

[1] Joko Hariyono and Kang-Hyun Jo. Detection of pedestrian crossing road. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 4585–4588. IEEE, 2015.

[2] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, Inc., 1993.

[3] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[4] D.G Patel, H.A Thakore. Moving object tracking using kalman filter. 2013.

[5] Javad Amirian, Bingqing Zhang, Francisco Valente Castro, Juan Jose Baldelomar, Jean-Bernard Hayet, and Julien Pettre. Opentraj: Assessing prediction complexity in human trajectories datasets. In *Asian Conference on Computer Vision (ACCV)*, number CONF. Springer, 2020.