Carnegie Mellon University
Research Showcase @ CMU

Robotics Institute | School of Computer Science

2006

Learning and Predicting Moving Object Trajectory: A Piecewise Trajectory Segment Approach

Patrick Pakyan Choi
Carnegie Mellon University

Martial Hebert
Carnegie Mellon University

Follow this and additional works at: http://repository.cmu.edu/robotics

Part of the Robotics Commons

This Technical Report is brought to you for free and open access by the School of Computer Science at Research Showcase @ CMU. It has been accepted for inclusion in Robotics Institute by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

Learning and Predicting Moving Object Trajectory: a piecewise trajectory segment approach

Patrick Pakyan Choi Martial Hebert

August 2006

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

© Carnegie Mellon University

Abstract

This paper presents an approach to predict future motion of a moving object based on its past movement. This approach is capable of learning object movement in an open environment, which is one of the limitions in some prior works. The proposed approach exploits the similarities of short-term movement behaviors by modeling a trajectory as concatenation of short segments. These short segments are assumed to be noisy realizations of latent segments. The transitions between the underlying latent segments are assumed to follow a Markov model. This predictive model was applied to two real-world applications and yielded favorable performance on both tasks.

1 Introduction

There are numerous proposed approaches on tackling the problem of clustering and predicting trajectory of a moving object. Some of these proposals are distance-based: Vasquez and Fraichard (2004) performs hierarchical clustering with Euclidean metric to partition training trajectories into clusters, learns the means and variances of the trajectories in the clusters, and predicts using the mean trajectory of the most probable clusters of the input trajectory segment; Piciarelli et al. (2005) builds a prefix tree of clusters found by minimizing normalized Euclidean distance, and uses Kalman filter to for trajectory prediction; Buzan et al. (2004) builds the clusters of trajectory by hierarchiical clustering with longest common subsequence distance; Junejo et al. (2004) applies min-cut algorithm on the Hausdorff distance matrix to partition the set of trajectory samples. Some approaches follow the path of generative modeling: Bennewitz et al. (2002) models the trajectory with a mixture of Gaussians; Alon et al. (2003) uses a model of mixture of hidden Markov models; Vasquez et al. (2005) uses a hidden Markov model incorporated with a goal variable representing the intention of the moving object; Gaffney and Smyth (2005) proposed a probabilistic model that jointly aligns and clusters trajectories. Alternative approaches include treating trajectories as functions of time instead of discrete data points and perform operations on the functional space (Ramsay and Silverman, 2002) (Ramsay and Silverman, 2005), and modeling trajectory by differential equations (Mertz, 2004).

A number of the clustering approaches assume that all trajectories are recorded on a fixed scene. This assumption is rather restrictive, for example, modeling vehicle movements in an urban environment violates the fixed scene assumption.

2 Modeling Trajectory

We propose a trajectory model that is based on a Markov model of piecewise trajectory segment. A moving object trajectory can be broken down into a series of short trajectory segments. These short trajectory segments are assumed to be generated, with perturbation of noise, from one of the many latent segments. The occurrence of latent segment that underlines an observed trajectory segment is dependent on the latent state of the previous segment. The relationship among the entire observed trajectory, observed trajectory segments, and the latent segments is shown in Figure 1. The latent segments, their associated noise model, and the stochastic model of transition between states are learned from data.

This piecewise segment model has an advantage over modeling explicity coordinates at each sampling intervals. Explicitly modeling the behavior of the moving object at every sampled instances may yield erroneous model on noisy data. Dropping the fine-grained model and adopting a model with a layer of abstraction and generalizatio may mitigate the effect of noisy and inaccurate measurements.

The stochastic nature of our proposed model produces forecasts in the form of a temporal-spatial distribution, which is not available in deterministic trajectory models. The forecast distribution is useful in applications such as path planning and collision avoidance, as the temporal-spatial forecast distribution allows probabilistic evaluations

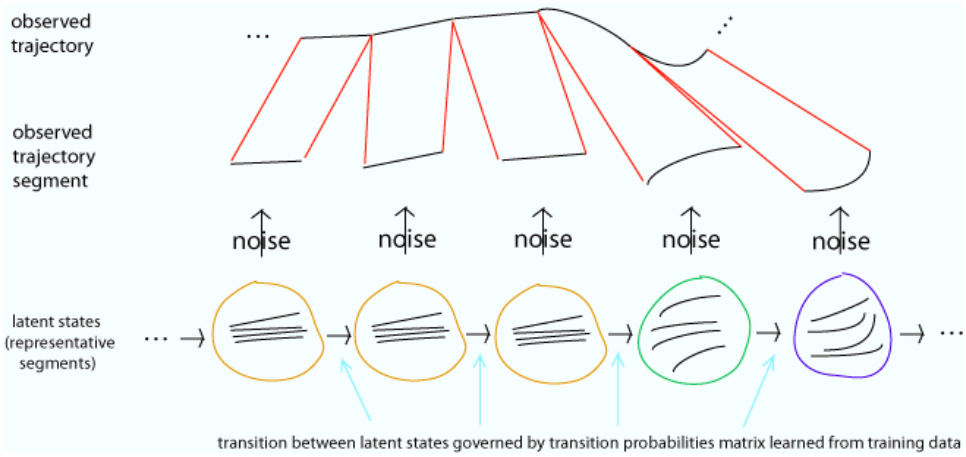


Figure 1: Modeling of trajectory of moving object: relationship among the entire observed trajectory, observed trajectory segments, and the latent representative segments.

of trajectory and risk.

The trajectory is modeled by joining together multiple segments, where one segment is only dependent on the location and speed of previous segment. This locally relative representation free us from considering the global location of the moving object, and thus would be applicable even if the scene is not fixed.

3 Data

We tested the trajectory model on two sets of data: PAT bus data and RoboReceptionist data. The PAT bus data recorded the movements and locations of buses in the Pittsburgh area over several months. The movement and location were measured by a device installed on each bus participated in the data collection procedure. Since the buses go on different routes to various parts in the city, this application requires a predictive model that works in an open environment.

The RoboReceptionist dataset recorded the pedestian movements at one of the entrances of the Newell-Simon Building at the Carnegie Mellon University. The locations of the pedestians were recorded by a robotic receptionist installed at the entrance.

Both datasets consist of time series of x- and y-coordinates of moving objects. The sampling frequency is 10Hz. The trajectory of each moving object was extracted and was saved into seperate files.

4 Model Implementation

4.1 Preprocessing

Smoothing the time series of the recorded object locations was necessary as the data collected is noisy. A Gaussian filter with full width at half maximum (FWHM) of 32 units was applied independently to both time series of x- and y-coordinates of the data. Gaussian filter smooths the data by replacing each data point by a function of that data point and its neighborhood, with closer points contributing more to smoothed value and points further away contributing less. The new value at the k-th position of the time series, x'_k is computed as

$$x'_k = \sum_i x_i \, G(k-i)$$

$$G(\delta) = \frac{e^{\frac{\delta^2}{2\sigma^2}}}{\sum_j e^{\frac{\delta^2}{2\sigma^2}}}$$

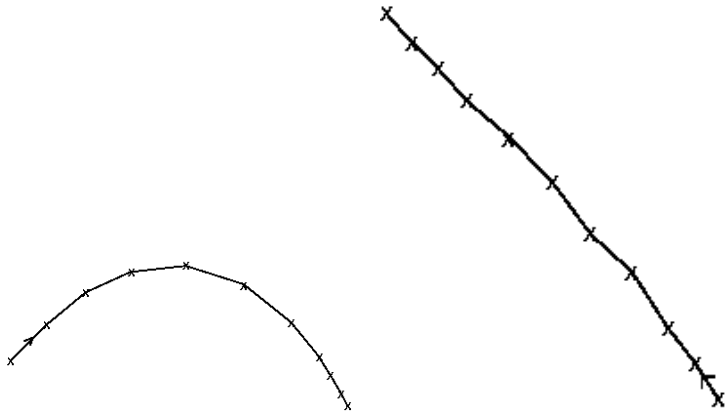
where $\sigma = \frac{\text{FWHM}}{\sqrt{8\,log(2)}}$.

4.2 Learning Latent Trajectory Segments

Short segment of observed vehicle trajectory is assumed to be a noisy realization of a latent trajectory segment. To find the set of latent trajectory segment from the data, we sampled from the data randomly to form a training set of observed short segments, and then applied clustering algorithm on this training set to find the representative segments. On the PAT bus dataset, we generated a training set consisted of 295000 randomly sampled segments. On the RoboReceptionist data, the training set has 63026 training segments.

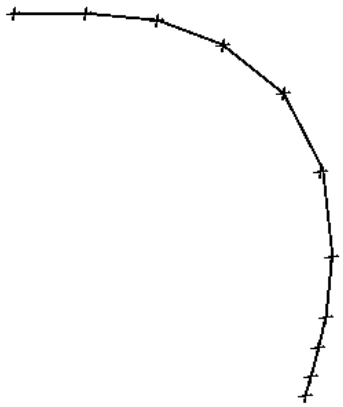
Since these trajectory segments have different directions, velocities, and starting locations, we had to normalize and align each segment prior to clustering. The normalization procedue translates each segment so that its initial position is the origin (0,0), rotates it so that its initial direction is $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, and scales it so that it have unit initial velocity. Figure 2 demonstrates the intermediate and final outputs of the normalization procedure.

K-means algorithm (Duda et al., 2000) was used to perform the clustering to find the latent segments. Each normalized training segment was transformed to a 22-dimensional vector consisted of the x- and y-coordinates at 0.1 second interval. K-means algorithm is a clustering algorithm that clusters data points into k partitions. The algorithm finds the cluster configuration that minimizes the intra-cluster variance, which is the sum of squared Euclidean distances between the cluster centroids and their associated data points. Alternatively, the algorithm can be viewed as a hard label assignment variant of the Expectation-Maximization algorithm (Dempster et al., 1977), assuming the data is generated from Gaussian distribution with a diagonal variance

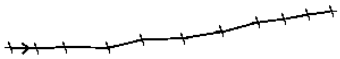


(a) Original trajectory segment #1

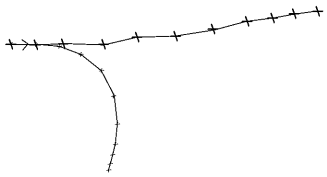
(b) Original trajectory segment #2



(c) Aligned trajectory segment #1



(d) Aligned trajectory segment #2



(e) Aligned and scaled trajectory segments #1 and #2

Figure 2: Normalization of the object trajectory segment. Each short segment is rotated so that the initial directory is aligned with the x-axis. In addition, the trajectory segments are scaled so that they have unit initial velocity.

matrix. K, the number of clusters, in the k-means algorithm has to be provided. We set this number with educated guesses. We used a fast implementation (Pelleg and Moore, 1999) of k-means algorithm which speeds up the k-means algorithm by using kd-trees to reduce the number of data points to be considered in each nearest-neighbor query.

Based on the result from clustering the training segments, we computed from training data, the noise model for each cluster. The noise model assumes that for each cluster the x- and y-coordinates at sampling interval are bivariate Gaussian distributed. The mean vectors and variance matrices were learned from the training data.

4.2.1 Noise Models

Each latent segment has a noise model which captures the means and variances of trajectory scaled to unit initial velocity and centered to origin. For each time-step t, the model of latent segment i has 5 parameters: mean x-coordinate $\bar{x}_{it}$, mean y-coordinate $\bar{y}_{it}$, x-coordinate variance V_{it}^{xx}, y-coordinate variance V_{it}^{yy}, and x-y-coordinate covariance V_{it}^{xy}.

Trajectory means of latent segment i at time-step t:

$$\begin{pmatrix} \bar{x}_{it} \\ \bar{y}_{it} \end{pmatrix}$$

Trajectory variances and covariance of latent segment i at time-step t:

$$\begin{bmatrix} V_{it}^{xx} & V_{it}^{xy} \\ V_{it}^{xy} & V_{it}^{yy} \end{bmatrix}$$

Because of the normalization procedure uses the first two time-steps as reference points for normalization, the noise models at the first two time-steps of all latent segments are identical. In particular, the mean trajectory coordinates at the first time-step is always $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$, and the mean coordinates at the second time-step is always $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$. The variances at both time-steps are always $\mathbf{0}$.

4.3 Learning Markov Model

Learning the Markov model involves learning the transition probabilities between the latent states. Given the noise model of latent state i, which is comprised of the means, variances, and covariances of the trajectory at each time-step t,

$$\begin{bmatrix} \bar{x}_{it_o} & \cdots & \bar{x}_{it_l} \\ \bar{y}_{it_o} & \cdots & \bar{y}_{it_l} \\ V_{it_o}^{xx} & \cdots & V_{it_l}^{xx} \\ V_{it_o}^{xy} & \cdots & V_{it_l}^{xy} \\ V_{it_o}^{yy} & \cdots & V_{it_l}^{yy} \end{bmatrix}$$

we can compute $\mathcal{L}$ the likelihood of the normalized input trajectory generated from latent state i by

$$\mathcal{L} = \prod_t Normal\left(\begin{pmatrix} x_t \\ y_t \end{pmatrix} | \begin{pmatrix} \bar{x}_{it_o} \\ \bar{y}_{it_o} \end{pmatrix}, \begin{bmatrix} V_{it}^{xx} & V_{it}^{xy} \\ V_{it}^{xy} & V_{it}^{yy} \end{bmatrix}\right)$$

where $Normal\,(\cdot|\vec{\mu}, \mathbf{\Sigma})$ is the bivariate Gaussian pdf.

Assuming a uniform prior on the latent states, application of Bayes Theorem shows that the posterior of the normalized input trajectory generated from latent state i is proportional to the likelihood of the trajectory generated from latent state i. Therefore, the latent state with the highest posterior probability of generating the observed segment is the one with the highest likelihood.

$$\begin{aligned} P(\text{latent state } i \,|\, \text{data segment}) &\propto \mathcal{L}(\text{data segment} \,|\, \text{latent state } i) P(\text{latent state } i) \\ &\propto \mathcal{L}(\text{data segment} \,|\, \text{latent state } i) \\ \text{argmax}_i\, P(\text{latent state } i \,|\, \text{data segment}) &= \text{argmax}_i\, \mathcal{L}(\text{data segment} \,|\, \text{latent state } i) \end{aligned}$$

The whole sequence of latent states underlying the observed training trajectory can be computed by first normalizing each segment and then finding the latent state with the highest likelihood. By counting and normalizing the number of transitions from one latent state to another along the entire training trajectory, we can learn the transition probabilities of the Markov model.

4.3.1 Higher-order Markov Model

Higher-order Markov model is an extension to the Markov model such that the next latent state is stochastically dependent on the latent states multiple time steps in the past. This extension yields a richer model that can better capture the dynamics of the trajectories. However, higher-order Markov model is impractical for even moderate number of time-step look-back, due to the exponential increase in the size of the transition probabilities matrix. In this paper we adopted a second-order Markov model which means the next latent state is stochastically dependent on both current and previous latent states.

4.4 Free Parameters and Design Choices

In our model formulation there are two parameters that have significant effects on the model performance.

4.4.1 Segment Length

In this paper, all models break up trajectories into segments of one second long. Increaseing the length of the segments will yield a richer set of movement patterns as longer segment will have more variability. However, the increase in variability in the trajectory segments will have to be accounted for by a larger set of latent states. As the number of latent states increases, more training samples will be required to properly learn the transition matrices. In addition, the time complexity increases as the number of latent states increases. Therefore, the length of segment will affect both the predictive accuracy and run-time speed of the model.

4.4.2 Number of Latent States

The number of latent states has to be provided to the k-means clustering algorithm to determine the number of clusters. The models applied to the PAT bus data used 100 latent states, and those applied to the RoboReceptionist data used 8. These number were educated guesses.

The number of latent states may affect the performance of the predictive model. Too many latent states will result in some latent states with very few data points and thus the trajectories in these states will have high variances. Too few latent states will cause the model fail to fully capture the structure of the trajectory segments. Moreover, more latent states will slow down the model.

We were later informed that there is a option in the fast k-means algorithm to automatically choose the number of clusters by BIC criterion. Using this option will solve the problem of arbitarily choosing the number of latent states by hand. However, there is no guarrantee that the number of clusters found using BIC criterion will yield the best predictive performance.

5 Evaluation

The predictive performances of the models are measured by the distances between the predicted and ground truth locations. Given a historic trajectory $\{(x_{t_o-h}, y_{t_o-h}),\ \dots\ , (x_{t_o}, y_{t_o})\}$, the model returns its forecast of the future trajectory $\{(\tilde{x}_{t_o}, \tilde{y}_{t_o}),\ \dots\ , (\tilde{x}_{t_o+1}, \tilde{y}_{t_o+1}),\ \dots\ , (\tilde{x}_{t_o+2}, \tilde{y}_{t_o+2}),\ \dots\}$. We can compute the prediction error, which is the distance between the predicted location and the actual location, at t-second into the future by calculating

$$\sqrt{(x_{t_o+t} - \tilde{x}_{t_o+t})^2 + (y_{t_o+t} - \tilde{y}_{t_o+t})^2}$$

Simulation of the Markov model on the given historic trajectory multiple times will generate a set of forecasted trajectories due to the stochastic nature of Markov model. Therefore we have a collection of prediction errors at each sampling interval. Suppose simulation is performed M times, each simulation is predictive T seconds into future, and the model handles trajectories at δ-second interval. Then the collection of prediction errors can be represented in matrix form,

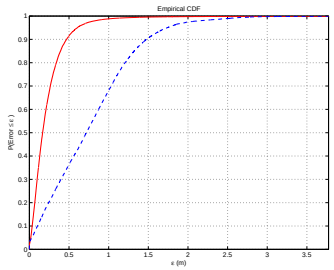
$$\begin{pmatrix} \epsilon_{1,\delta} & \epsilon_{1,2\delta} & \cdots & \epsilon_{1,T} \\ \epsilon_{2,\delta} & \epsilon_{2,2\delta} & \cdots & \epsilon_{2,T} \\ \vdots & \vdots & \ddots & \vdots \\ \epsilon_{M,\delta} & \epsilon_{M,2\delta} & \cdots & \epsilon_{M,T} \end{pmatrix}$$

The empirical cumulative distribution functions and percentiles of these errors at a particular time-step provide information regarding behavior of the performance of these predicitive models.

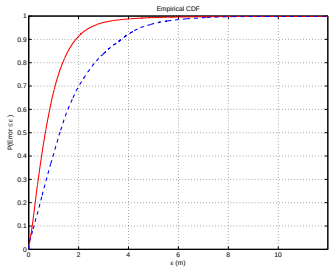
5.1 Empirical Results

We empirically evaluated the predictive performances of first-order Markov model and second-order Markov model on both PAT bus and RoboReceptionist datasets. On the

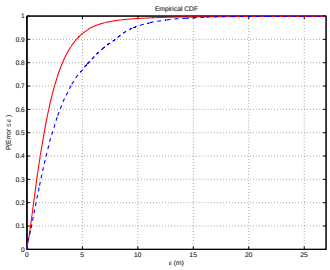
PAT bus data, we generated test sets of 4830 and 3209 randomly sampled trajectories for the first-order Markov model and second-order Markov model respectively. On the RoboReceptionist data, we generated test sets of 552 and 486 randomly sampled trajectories for the first-order Markov model and second-order Markov model respectively. For each trajectory in the test sets, 100 predictions were generated. The errors (in meters) are recorded and the cumulative distribution functions of these errors are plotted in Figure 3 for the PAT bus dataset and in Figure 4 for the RoboReceptionist dataset. Percentiles of the errors of the two models are shown in Table 1 for the PAT bus dataset and in Table 2 for the RoboReceptionist dataset.



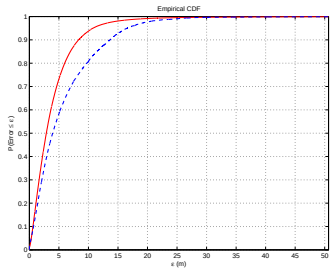
(a) Predicting 1 second ahead



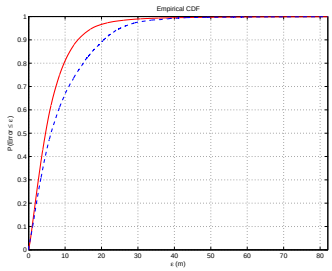
(b) Predicting 2 seconds ahead



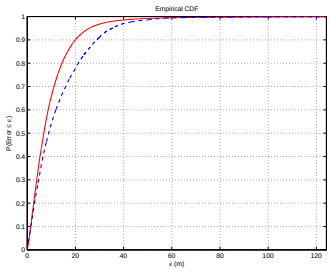
(c) Predicting 3 seconds ahead



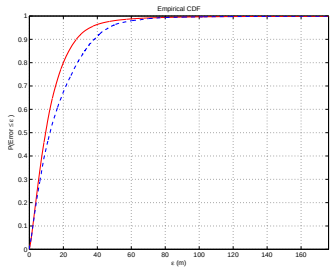
(d) Predicting 4 seconds ahead



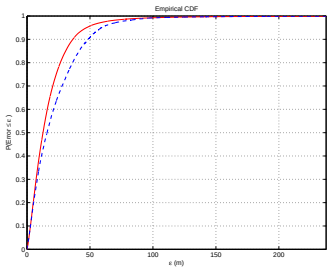
(e) Predicting 5 seconds ahead



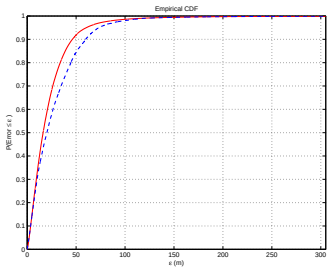
(f) Predicting 6 seconds ahead



(g) Predicting 7 seconds ahead



(h) Predicting 8 seconds ahead

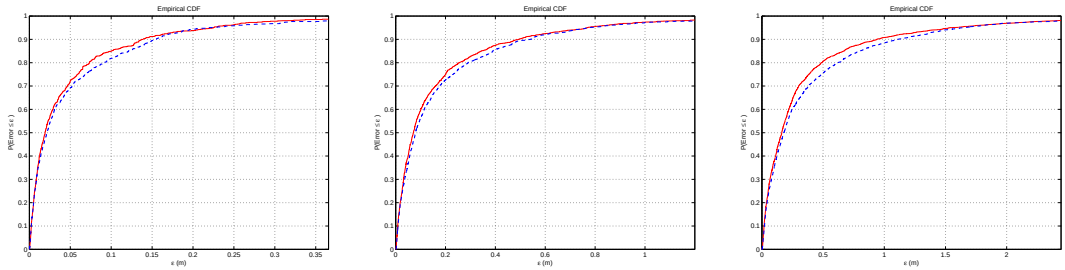


(i) Predicting 9 seconds ahead

Figure 3: Evaluation on the PAT bus dataset: Cumulative distribution function plots of empirical prediction error compared to the ground truth trajectories. From each plot, one can read from y-axis the probabilities of the prediction model making a prediction error less than the corresponding threshold in x-axis. Blue dotted lines represent the performances of first-order Markov models; red solid lines represent second-order Markov models. The uniform dominance of the red solid lines over the blue dotted lines in these plots show that the second-order Markov model is more likely than the first-order Markov model to make a prediction that is closer to the ground truth location.

Seconds into future	Percentiles of Prediction Error					
	50%		90%		95%	
	MM(1)	MM(2)	MM(1)	MM(2)	MM(1)	MM(2)
1	0.72 m	0.17 m	1.47 m	0.47 m	1.74 m	0.61 m
2	1.25 m	0.67 m	3.73 m	1.89 m	4.46 m	2.45 m
3	2.30 m	1.56 m	7.97 m	4.47 m	9.61 m	5.78 m
4	3.93 m	2.87 m	13.67 m	8.33 m	16.56 m	10.76 m
5	6.13 m	4.64 m	20.56 m	13.50 m	25.14 m	17.42 m
6	8.90 m	6.86 m	28.84 m	19.94 m	35.23 m	25.67 m
7	12.15 m	9.57 m	38.14 m	27.59 m	46.51 m	35.62 m
8	15.95 m	12.71 m	48.66 m	36.30 m	59.00 m	47.13 m
9	20.16 m	16.28 m	59.69 m	46.01 m	73.21 m	60.31 m
10	24.56 m	20.23 m	71.58 m	56.82 m	88.57 m	74.84 m

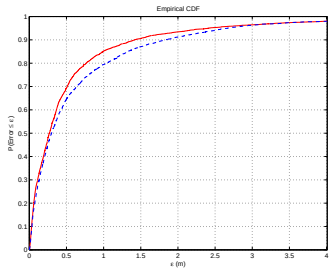
Table 1: Evaluation on the PAT bus dataset: 50%, 90%, and 95% percentiles of prediction error made by first-order Markov model (MM(1)) and second-order Markov model (MM(2)). The test set of first-order Markov model consisted of 4830 randomly sampled trajectories; the test set of second-order Markov model consisted of 3209 trajectories. Each trajectory in the test sets was compared to 100 simulated predictions by the corresponding model.



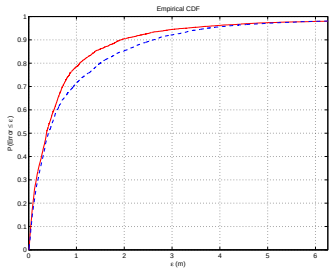
(a) Predicting 1 second ahead

(b) Predicting 2 seconds ahead

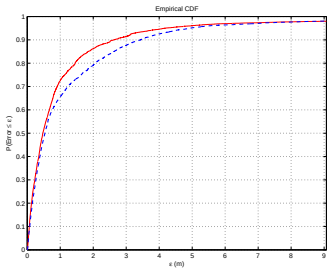
(c) Predicting 3 seconds ahead



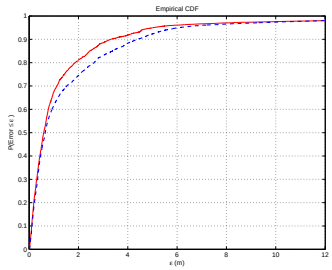
(d) Predicting 4 seconds ahead



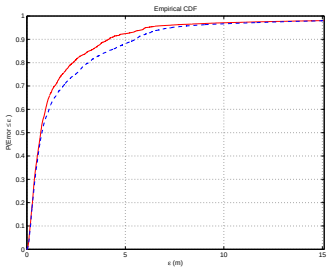
(e) Predicting 5 seconds ahead



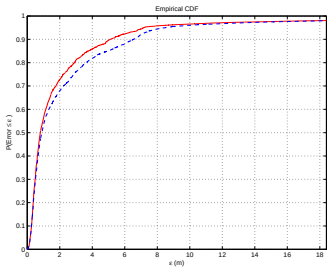
(f) Predicting 6 seconds ahead



(g) Predicting 7 seconds ahead



(h) Predicting 8 seconds ahead



(i) Predicting 9 seconds ahead

Figure 4: Evaluation on the RoboReceptionist dataset: Cumulative distribution function plots of empirical prediction error compared to the ground truth trajectories. From each plot, one can read from y-axis the probabilities of the prediction model making a prediction error less than the corresponding threshold in x-axis. Blue dotted lines represent the performances of first-order Markov models; red solid lines represent second-order Markov models. The uniform dominance of the red solid lines over the blue dotted lines in these plots show that the second-order Markov model is more likely than the first-order Markov model to make a prediction that is closer to the ground truth location.

Seconds into future	Percentiles of Prediction Error					
	50%		90%		95%	
	MM(1)	MM(2)	MM(1)	MM(2)	MM(1)	MM(2)
1	0.021 m	0.019 m	0.153 m	0.138 m	0.218 m	0.225 m
2	0.080 m	0.071 m	0.537 m	0.494 m	0.775 m	0.768 m
3	0.180 m	0.163 m	1.107 m	0.940 m	1.640 m	1.533 m
4	0.299 m	0.269 m	1.818 m	1.413 m	2.667 m	2.409 m
5	0.412 m	0.373 m	2.610 m	1.928 m	3.738 m	3.268 m
6	0.536 m	0.474 m	3.429 m	2.615 m	4.944 m	4.176 m
7	0.654 m	0.593 m	4.387 m	3.326 m	6.070 m	5.010 m
8	0.764 m	0.704 m	5.503 m	4.147 m	7.329 m	6.021 m
9	0.902 m	0.810 m	6.534 m	5.112 m	8.484 m	7.179 m
10	1.085 m	0.956 m	7.424 m	6.170 m	9.405 m	8.329 m

Table 2: Evaluation on the RoboReceptionist dataset: 50%, 90%, and 95% percentiles of prediction error made by first-order Markov model (MM(1)) and second-order Markov model (MM(2)) on the Roboreceptionist datset. The test set of first-order Markov model consisted of 552 randomly sampled trajectories; the test set of second-order Markov model consisted of 486 trajectories. Each trajectory in the test sets was compared to 100 simulated predictions by the corresponding model.

6 Discussion

We formulated a predictive trajectory model based on piecewise segments with stochastic transition and observation noises. The trajectory model was tested on two very different datasets. One of the dataset was collected in a small fixed scene while the other was collected in vast open urban environment; one dataset captures pedestrians movements while the other captures bus movements.

Empirically we found that the second-order Markov model outperforms the first-order Markov model. Over the range of look-ahead length from one to ten seconds, the second-order Markov model forecasted locations that are closer than that of the first-order model to the actual locations on both datasets. On the bus movement forecasting task, the second-order Markov model made a prediction with a error less than 4.5 meters 90% of time the second-order Markov model at a three-second look-ahead length.

A Codes

1. For each data set */path/data_filename*
   (a) set DATA_DIR */path/data_filename*
   (b) $ tankAnalyzer timegui.cfg > [output_filename]
   (c) $ parse_tankanalyzer_output.pl [output_filename]

This step extracts the x- and y-coordinates of the moving objects from the raw data. The sampling frequency can be modified in timegui.cfg.

2. In MATLAB, run

   $ gen_vocab_training_samples.m

   This script preprocesses the data, randomly samples a training set of short trajectory segments, and aligns, centers, and rescales these segments. This training set is then used to find the representative set of segments through clustering.

3. In command prompt, run

   $ segment_clustering.pl [Number_of_clusters]

   This operation clusters the short segment samples into clusters, with the maximum number of clusters defined by [Number_of_clusters].

4. In MATLAB, run

   $ genTrajCluster_external.m

   This script learns the means and covariances of each clusters from the clustered samples.

5. In MATLAB, run

   $ mm_o2_learn.m

   This steps learns the first-order and second-order transition matrices of the Markov Model using preprocessed data and learned cluster model. The complete model is then saved as SAVE_MODEL.mat

B Notes

B.1 Nearest-Neighbor Prediction

We implemented a predictive model that matches the input trajectory segment to the segments in the training dataset, finds the trajectories that are close (in terms of Euclidean distance) to the input segment, and then return the mean of the future segments of the found trajectories as prediction. Emprically this approach did not yield good prediction result.

B.2 Dynamic Time Wraping

We considered using dynmaic time wraping (DTW) to perform the normalization in the preprocessing step. While DTW may be a better tool for normalizing the input segment, DTW cannot provide the scaling factor used in the prediction step. The latent trajectory segments are normalized, therefore they have to be scaled back to actual length based on the input historic trajectory.

B.3 Modeling Speed of Moving Object

We attempted an alternative formulation of the Markov model which takes the speed of the moving object into account in the transition of latent states. In the implementation, we binned the initial speed (equivalently, the scaling factor) into 10 bins and built a transition matrix where the next state is dependent on the state and speed bin of the last segment. The performance comparison of this model over the first-order model was inconculsive.

References

Alon, J., Sclaroff, S., Kollios, G., and Pavlovic, V. (2003). Discovering clusters in motion time-series data. *CVPR*, 01:375.

Bennewitz, M., Burgard, W., and Thrun, S. (2002). Learning motion patterns of persons for mobile service robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, Washington, DC. ICRA.

Buzan, D., Sclaroff, S., and Kollios, G. (2004). Extraction and clustering of motion trajectories in video. *ICPR*, 02:521–524.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2nd Edition)*, pages 526–528. Wiley-Interscience.

Gaffney, S. J. and Smyth, P. (2005). Joint probabilistic curve clustering and alignment. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 473–480. MIT Press, Cambridge, MA.

Junejo, I. N., , and Shah, M. (2004). Multi feature path modeling for video surveillance. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 716–719.

Mertz, C. (2004). A 2d collision warning framework based on a monte carlo approach. In *Proceedings of ITS America's 14th Annual Meeting and Exposition*. accepted for publication.

Pelleg, D. and Moore, A. (1999). Accelerating exact k -means algorithms with geometric reasoning. In *Knowledge Discovery and Data Mining*, pages 277–281.

Piciarelli, C., Foresti, G., and Snidara, L. (2005). Trajectory clustering and its applications for video surveillance. In *Proc. of the IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 40–45.

Ramsay, J. O. and Silverman, B. W. (2002). *Applied functional data analysis: methods and case studies*. Springer series in statistics.

Ramsay, J. O. and Silverman, B. W. (2005). *Functional Data Analysis*. Springer Science+Business Media, Inc., New York.

Vasquez, A. D. and Fraichard, T. (2004). Motion prediction for moving objects: a statistical approach. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3931–3936, New Orleans, LA (US).

Vasquez, A. D., Fraichard, T., Aycard, O., and Laugier, C. (2005). Intentional motion on-line learning and prediction. In *Proc. of the Int. Conf. on Field and Service Robotics*, Port Douglas (AU).