

G-RIPS SENDAI 2023

IHI GROUP

Final Report

Authors:

MELISSA ADRIAN¹

NAMITO ASANO²

ANN MWANGI³

TOMOKI OKAMOTO²

AARON SCHEINER⁴

Mentors:

NATSUO MIYATAKE⁺

MASAO ONO^{*}

¹ The University of Chicago

² Musashino University

³ African Institute for Mathematical Sciences

⁴ Northwestern University

⁺ Academic Mentor, Tohoku University

^{*} Industrial Mentor, IHI Corporation

Date of presentation August 8, 2023.

Finalized by industrial
mentor May 16, 2024.

Title: Mathematics for Trajectory Extrapolation Using Vehicle and Human Traffic Data
Towards Zero Traffic Fatalities

IHI document number: HAPD117219

Contents

1	Introduction	2
2	Objectives	2
3	Data Description	3
4	Methods	4
4.1	Car-Following Models	4
4.2	Gipps Model.....	5
4.3	Ensemble Kalman Smoother	7
4.3.1	Assumptions.....	7
4.3.2	Algorithms	8
4.3.3	Specifying Known Quantities.....	9
5	Results	10
5.1	Objective 1: Assigning Labels to Unknown Trajectory Directions	10
5.2	Objective 2: Reducing Noise in the LiDAR Sensors	11
5.3	Objective 3: Extrapolating Trajectories.....	14
6	Conclusion	14
7	Further Directions	14
7.1	Improvements to the Forward Model	14
7.1.1	2D Car-Following Models	15
7.1.2	Joint Forward Model Learning and State Space Estimation	15
7.2	Improvements to the Data Assimilation Framework.....	15
7.2.1	Covariance Tapering	15
7.2.2	Assimilating Temporally Intermediate Observations.....	15
	References	16
	Appendix A. Data Preprocessing	17
A.1	Encoding Reference Trajectories	17
A.2	Spatial Region of Interest.....	18
A.3	2D to 1D Coordinate Conversion	18

1 Introduction

Traffic sensors are increasingly being applied to intelligent transportation systems (ITS). In particular, Light Detection And Ranging (LiDAR) can precisely capture shapes of vehicles and pedestrians as 3D point cloud data. Owing to this capability, the LiDAR technology holds significant potential for improving both traffic safety and sustainability to a great extent.

Traditional traffic measurements rely on video cameras for object detection. However, poor visibility in inclement weather or halation at night often deteriorates the detection accuracy. To solve this problem, IHI has been developing more reliable sensing systems using 3D Laser RadarTM [1], which is a proprietary LiDAR sensor of IHI. One of the strengths of these systems is high performance object tracking by detection algorithms refined over 20 years of the development and operation of 3D Laser Radar [2][3].

The LiDAR sensors, including 3D Laser Radar, use lasers instead of radio waves to scan objects on the road. The shorter wavelengths of lasers provide a finer resolution for observing the shapes of objects compared to conventional radar sensors. The LiDAR sensing typically gathers point cloud data by leveraging the Time-of-Flight (ToF) principle, which measures the time taken for laser light to return. The ToF sensing facilitates object detection even in environments imperceptible to the human eye.

In addition, the detection algorithms implemented in 3D Laser Radar systems enable to accurately track and classify each object along its trajectory. To date, over 2,400 units of 3D Laser Radar sensors have been monitoring vehicles and pedestrians across the world, especially at railroad crossings [3]. Recently, the development has been extended to various ITS applications such as intersections and highways [4]. This advancement helps both drivers and autonomous vehicles to drive safely as eyes on the roadside.

While the LiDAR technology is in high demand in accident-prone areas, object tracking using the ToF sensors can be a challenge, particularly in high-traffic intersections and merging sections. For example, if one vehicle is obscuring another from the view of a roadside sensor, the sensor cannot detect the hidden one. This is called occlusion and often makes tracking performance more vulnerable to detection errors. Therefore, to fully observe all trajectories of vehicles and pedestrians, we need to cover the measurement areas with a large number of sensors and eliminate any blind spots.

However, a significant disadvantage of the LiDAR, its high implementation cost, restricts the number of locations where the sensors can be installed. Consequently, our primary objective in this study is to minimize the requisite number of LiDAR sensors to capture accurate trajectories by harnessing mathematics. Our team proposes a novel approach to smooth partial LiDAR data and estimate complete trajectories by using mathematical car-following models and offline smoothing methods. This methodology can pave the way for broader applications of LiDAR sensors, contributing to the realization of safe and sustainable transportation systems.

2 Objectives

The primary goals in this project are as follows:

1. Assign labels that represent traveling directions to vehicles labeled as ‘unknown’ directions by using geographic and geometric information on a specified intersection as well as given LiDAR data.
2. Reduce noises and generate probable trajectories by eliminating measurement errors caused by data fusion of multiple LiDAR sensors.

3. Design mathematical models to extrapolate the trajectories of vehicles.

If time permits, we can explore the following additional goals related to this project:

4. Develop an alternative approach for assigning direction labels to vehicles labeled as ‘unknown’ directions.
5. Evaluate safety levels of the intersection by analyzing trajectories and labels generated by our proposed methods.

3 Data Description

In this project, we studied mathematical models by using traffic flow datasets that were observed by four LiDAR systems temporarily installed on the roadside. These measurements were taken in real traffic situations (and not on specific test courses), such as multilane traffic intersections with traffic lights in downtown Tokyo (Figure 1).

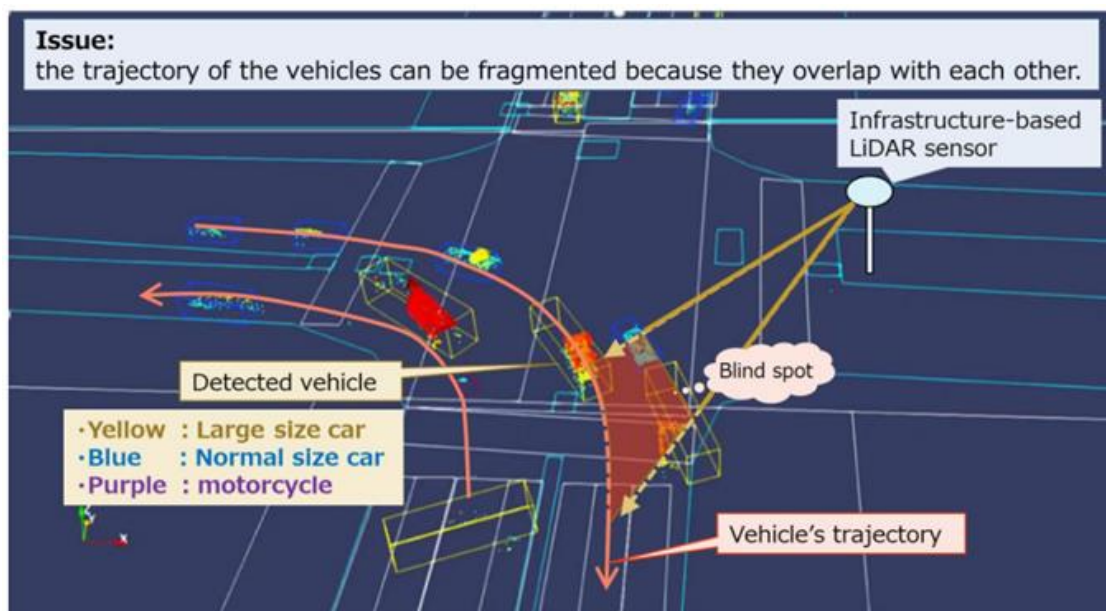


Figure 1: Traffic situations of datasets.

The measurements were conducted under the following conditions:

- Date: 11/24/2021 8:00 am to 5:00 pm (JST)
- Number of LiDAR sensors: four units
 - The data was generated by fusing all LiDAR sensors.
 - The data contains measurement noises caused by synchronization errors at multiple time periods.

The dataset consists of time series data of each object. The columns are shown in Figure 2.

- The following data are assigned to each vehicle:
 - Type: normal car, large car, bike, and pedestrian
 - Size: height [m], length[m], and width [m]
- The following data are recorded in each index (frame):
 - x-coordinate of vehicle's head [m] (on local coordinate system)
 - y-coordinate of vehicle's head [m] (on local coordinate system)

No.	Column name	Description	Unit
1	Frame	Serial number (ascending-order)	$\times 0.1$ ms
2	Time	Japan Standard Time (JST) of each frame	YYYY/MM/DD hh:mm:ss.000
3	Delta_time	Sampling time (diff from last frame)	ss.000
4	ID	Unique ID of each object	—
5	x	x-coordinate (head) of each ID	m
6	y	y-coordinate (head) of each ID	m
7	Type	Unique type of each ID	<ul style="list-style-type: none"> • Normal • Large • Bike • Pedestrian*
8	Length	Unique length of each ID	m
9	Width	Unique width of each ID	m
10	Height	Unique height of each ID	m

★ Although the dataset includes pedestrian trajectories, the mathematical modeling is excluded from our research objects mentioned in [Section 2](#).

Figure 2: Measurement items of datasets.

4 Methods

In achieving our research objectives, we explore a variety of car-following models, especially one-dimensional models mentioned in [\[5\]](#). Furthermore, we use a framework of Ensemble Kalman Smoother (EnKS) [\[6\]](#) to integrate the car-following models and actual LiDAR data.

4.1 Car-Following Models

Car-following models are commonly classified into three categories: General Motors (GM)-type models, safety-distance models, and psychophysical car-following models. In the GM-type models, follower vehicle's acceleration is proportional to three factors: (1) follower's speed, (2) difference in speed between the follower vehicle and the leading one, and (3) spacing between the follower vehicle and the leading one. In the safety-distance models, the followers maintain their safe distance to the leading vehicles. Lastly, in the psychophysical car-following models, the follower's driving behavior is determined by "perceptual thresholds." All drivers can react to changes in spacing or relative velocity only when certain variables exceed their own thresholds. For instance, one of the thresholds is the minimum value of relative velocity that the followers can perceive [\[5\]\[7\]](#).

The GM-type car-following models are based on the simple GM model, in which the acceleration of the following vehicle is calculated by subtracting the current speed of the leader from that of the follower. The Gazis-Herman-Rothery model incorporates the concept of distance into the simple GM model and can describe more detailed driving behavior [\[5\]\[7\]](#).

The Gipps car-following model, which is the most famous safety-distance model, assumes the following

two constraints: (1) drivers should not exceed their desired speed; (2) they should always maintain their own safe distance [5][7][8].

The Wiedemann model, a commonly referenced psychophysical car-following model, postulates that characteristics of each driver depend on the following four factors: (1) driving capabilities for perception, reaction, and estimation of surrounding traffic environment, (2) safety requirements, (3) desired speed, and (4) aggressiveness towards the maximum acceleration and deceleration. Based on these premises, each driver determines subsequent driving behavior, as determined by perceptual thresholds, and transitions between the following four regimes: (1) free driving, (2) approaching/closing, (3) following/closing, and (4) emergency [5][7].

Additionally, we also address other models and several applications.

Optimal velocity model (OVM) describes the dynamics of traffic congestion using the equation of motion of each vehicle. In the OVM, all drivers possess constant sensitivities. Each driver's velocity alters toward the optimal velocity, which is determined by a function of the following headway, representing the distance to the preceding vehicle. The optimal velocity, $V(\Delta x)$, is empirically known to follow $V(\Delta x) = \tanh(\Delta x - 2) + \tanh(2)$, where Δx represents the following headway [9].

The car-following models described above have been utilized for driving control systems of vehicles such as Adaptive Cruise Control (ACC) and Cooperative Adaptive Cruise Control (CACC). The ACC serves as driver assistance systems, while the CACC is an advancement of the ACC system, incorporating vehicle-to-vehicle (V2V) communication systems [5].

In addition to the car-following models, a lane-change model was proposed by Peter G. Gipps, the originator of the Gipps car-following model. The lane-change model entails several rules governing lane-changing behavior. Lane changes are further categorized into two categories: mandatory lane changes, which are necessary, and discretionary lane changes, which are optional [5][7].

In this study, we opted for the Gipps model for three reasons. The foremost reason pertains to the expressive power of the Gipps model. This model can closely emulate real-world driving behavior with relatively simple mathematical model. Furthermore, this concept of the safe distance inherent in this model appears to align with car-following behavior within the intersections shown in Section 3. Finally, the parameters of the Gipps model can be identified from our intersection data. From these reasons, we believe that the Gipps model, or the safety-distance model, is appropriate for our research.

4.2 Gipps Model

The Gipps model focuses on the concept of the safety distance from a following vehicle to its preceding one. As mentioned above, this model has two constraints:

- (1) Drivers' velocity should not exceed their desired velocity.
- (2) Drivers should maintain their own safety distance.

These constraints appear to align with typical behavior of actual drivers following the preceding vehicles. In this model, the velocity of the following vehicle after reacting to any behavior of the preceding one, is described as follows:

$$v_n(t+T) = \min \{v_n^a(t+T), v_n^b(t+T)\}, \quad (4.1)$$

where $v_n^a(t+T)$ and $v_n^b(t+T)$ are given by

$$v_n^a(t+T) = v_n(t) + 2.5 a_n^{\max} T \left\{ 1 - \frac{v_n(t)}{v_n^{\text{desired}}} \sqrt{0.025 + \frac{v_n(t)}{v_n^{\text{desired}}}} \right\}, \quad (4.2)$$

$$v_n^b(t+T) = d_n^{\max} T + \sqrt{(d_n^{\max} T)^2 - d_n^{\max} \left[2\{v_{n-1}(t) - s_{n-1} - x_n(t)\} - v_n(t) T \frac{v_{n-1}^2(t)}{d_{n-1}} \right]}, \quad (4.3)$$

where

- T is a reaction time of the following vehicle.
- n is the follower's index and $n - 1$ is the leader's index as shown in [Figure 3](#).
- $v_n(t)$ and $v_{n-1}(t)$ are vehicles' velocities of the follower and the leader at the time t , respectively.
- v_n^{desired} is the follower's maximum desired velocity.
- a_n^{\max} is the follower's maximum acceleration.
- d_n^{\max} is the follower's deceleration at the most severe braking.
- \hat{d}_{n-1} is the leader's d_n^{\max} estimated by the follower n .
- $x_n(t)$ and $x_{n-1}(t)$ are locations of the follower and the leader at the time t , respectively.
- s_{n-1} is inter-vehicle spacing at a stop, which is the follower's desired inter-vehicle spacing at the stop including the length of the leader's vehicle.

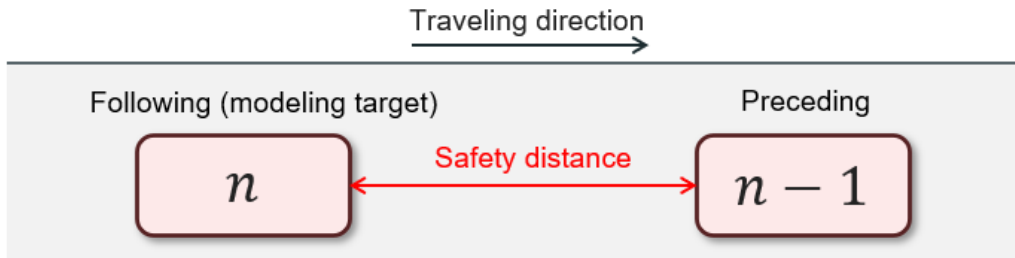


Figure 3: Indexes of leader and follower.

Each vehicle's velocity $v_n(t+T)$ at the time $t+T$ depends on the previous value at $t+T-1$. The constraint (1) is considered by $v_n^a(t+T)$ and drivers' velocities $v_n(t+T)$ accelerate toward their desired velocities. In other words, $v_n^a(t+T)$ adjusts the accelerating behavior of the follower n . On the other hand, $v_n^b(t+T)$ takes into account the constraint (2) and decelerates the follower n when there is a possibility that the vehicle will break its safe distance and approach the leading vehicle $n-1$.

The transition between $v_n^a(t+T)$ and $v_n^b(t+T)$ occurs smoothly because $v_n^b(t+T)$ is put into effect

before the maximum braking is required. The following vehicles can continue to accelerate again even after a transition from accelerating behavior to decelerating one [8].

4.3 Ensemble Kalman Smoother

Our team considers the ensemble Kalman smoother (EnKS) presented in [6] to be a useful approach for addressing objective 2. Kalman filters can be utilized in an online setting, whereas Kalman smoothers are better suited to be used in an offline setting. Since the setting for our denoising task is offline, Kalman smoothing methods are more appropriate.

We propose to utilize EnKS for two particular reasons: (1) this approach can handle cases where our forward model is a nonlinear operator, therefore avoiding the need for a linear adjoint as in the case of a generic Kalman smoother, and (2) we would like to utilize information from future time points to inform our estimate at each time point. In the case of the ensemble Kalman filter (EnKF), this method estimates quantities of interest across time only using information from $[0, t]$ for each t , does not utilize observational data across the entire timespan and therefore leads to a less smooth fit [11].

In our setting, the forward model that we plan to pair with Kalman methodology is a nonlinear operator of the latent state space, namely the car-following models mentioned in Section 4.1. Therefore, we need to utilize the EnKS to smooth the data using information across the entire timespan without constructing an operator mapping backward in time.

To achieve the same goal of denoising the vehicle data, our group will investigate results obtained by using a cubic spline and compare these results to the EnKS.

4.3.1 Assumptions

We can formalize the assumptions underlying this Kalman smoother with the following Gaussian state-space model [12]:

$$X_t = F(X_{t-1}) + \xi_t, \quad \xi_t \sim N(0, Q_t), \quad 1 \leq t \leq T, \quad (4.4)$$

$$Y_t = H_t X_t + \eta_t, \quad \eta_t \sim N(0, R_t), \quad 1 \leq t \leq T, \quad (4.5)$$

$$X_0 = p_0(X_0), \quad (4.6)$$

where $X_t \in \mathbb{R}^{d_{x_t}}$ is a vector of all quantities of interest to reconstruct at time t , including vehicle locations, velocities, and accelerations at time t . $Y_t \in \mathbb{R}^{d_{y_t}}$ is a vector of all observed quantities of interest at time t , which in our setting, includes positional readings from the LiDAR sensors for vehicles in the intersection at time t . In our application setting, the dimension of the space we would like to estimate is larger than the dimension of our observational space ($d_{x_t} \gg d_{y_t}$). We emphasize that the dimension of the problem is also changing with time, denoted by the subscripts t in the dimension notations d_{x_t} and d_{y_t} . F refers to the car-following models that estimate vehicle variables at time t using traffic and pedestrian information at time $t - 1$. $H_t \in \mathbb{R}^{d_{x_t} \times d_{y_t}}$ is a linear observation operator mapping the latent quantities of interest to the observation space. The initial distribution p_0 is assumed to be known, meaning we assume that we

have access to a complete characterization of the latent space at time zero [12]. Lastly, we assume independent Gaussian noise statistics for the forward model errors and the observation errors with respective known covariance matrices Q_t and R_t [12].

4.3.2 Algorithms

Algorithm 1 and **Algorithm 2** provide pseudo-code for EnKF and EnKS, respectively. The EnKF algorithm alternates between two steps at each time point: forecasting into the future based on the previous time point, and correcting this forecast based on observational data. EnKS extends on this idea utilizing future observations; first, the EnKF solution is computed across time points $1 \leq t \leq T$, then this solution is improved utilizing observations up to L time points in the future.

Algorithm 1. Ensemble Kalman Filter (EnKF) [12]	
1: Input: $Y_{1:T}$, $X_0^{1:N}$. (If $X_0^{1:N}$ is not specified, draw $X_0^n \sim \text{i.i.d. } p_0(X_0)$)	
2: for $t = 1, \dots, T$ do	
3: Set $\hat{X}_t^n = F(X_{t-1}^n) + \xi_t^n$, where $\xi_t^n \sim \text{i.i.d. } N(0, Q_t)$.	▷ Forecast
4: Compute \hat{m}_t and \hat{C}_t using (4.7) and (4.8), and set $\hat{K}_t = \hat{C}_t H_t^T (H_t \hat{C}_t H_t^T + R_t)^{-1}$.	
5: Set $X_t^n = \hat{X}_t^n + \hat{K}_t (Y_t + \eta_t^n - H_t \hat{X}_t^n)$, where $\eta_t^n \sim \text{i.i.d. } N(0, R_t)$.	▷ Analysis
6: end for	
7: Output: $X_{1:T}^{1:N}$	

$$\hat{m}_t = \frac{1}{N} \sum_{n=1}^N \hat{X}_t^n \quad (4.7)$$

$$\hat{C}_t = \frac{1}{N-1} \sum_{n=1}^N (\hat{X}_t^n - \hat{m}_t)(\hat{X}_t^n - \hat{m}_t)^T \quad (4.8)$$

Algorithm 2. Ensemble Kalman Smoother (EnKS) [6]	
1: Input: $Y_{1:T}$, $X_0^{1:N}$. (If $X_0^{1:N}$ is not specified, draw $X_0^n \sim \text{i.i.d. } p_0(X_0)$)	
2: Compute $\hat{X}_{1:T}^{1:N}$ using Algorithm 1 .	▷ EnKF
3: for $t = 1, \dots, T$ do	
4: Compute \hat{m}_t and \hat{C}_t using (4.7) and (4.8).	
5: Compute $\hat{K}_{t,t+l} = \hat{C}_{t,t+l} H_{t+l}^T (H_{t+l} \hat{C}_{t,t+l} H_{t+l}^T + R_{t+l})^{-1}$.	
6: Set $X_t^n = \hat{X}_t^n + \sum_{l=1}^L \hat{K}_{t,t+l} (Y_{t+l} + \eta_{t+l}^n - H_{t+l} \hat{X}_{t+l}^n)$, where $\eta_{t+l}^n \sim \text{i.i.d. } N(0, R_{t+l})$.	▷ Look ahead time steps
7: end for	
8: Output: $X_{1:T}^{1:N}$	

4.3.3 Specifying Known Quantities

One main challenge in applications of EnKF and EnKS is that some quantities are assumed to be known a priori, but in practice, these quantities are potentially not known by the researchers. Particularly, the error covariances Q_t and R_t are assumed to be known, as well as the parameterization of the forward model F and the complete initial condition X_0 . In practice, these quantities are estimated prior to the EnKF/EnKS analysis.

In this work, we do not have a prior sense of the magnitudes of Q_t and R_t , and the forward model parameters. We can fully specify the initial condition using our observation data.

Dimensionality of the latent and observation states.

The dimensionality of the system in terms of latent states and observations is dynamic in this traffic system since vehicles are entering and exiting our region of interest. The latent state at time t is comprised of all vehicles' positions, speeds, and accelerations in the intersection. This vector contains all quantities that we would like to estimate at the given time t . The number of quantities we would like to estimate at time t is denoted by d_{x_t} . The observations at time t are the vehicles' positions detected by the LiDAR sensor. The number of vehicles' positions we detect at time t is denoted by d_{y_t} . We note in this setting, $d_{x_t} \gg d_{y_t}$.

Choosing the ensemble size.

Since the ensemble size is directly tied to how well \hat{C}_t in (4.8) is estimated, we would like to choose this user-defined hyper-parameter as large as possible. However, setting this hyper-parameter to a considerable number increases the storage and computational space needed for the problem. This trade-off is problem specific and depends on the size of d_{x_t} . For this project, we set $N = 100$ as a starting point to assess the performance.

Creating the initial ensemble.

To create the initial ensemble, we choose to initialize the system using observational data that corresponds to the first time point in the dataset that we have both velocity and acceleration information, which corresponds to the first timepoint in the dataset where we have both velocity and acceleration information for all observed vehicles in the dataset. We denote this initial condition as X_0 .

To create the initial ensemble, we add Gaussian noise to the initial condition in a way that describes our uncertainty of this initial condition. More formally,

$$X_0^n = X_0 + \varepsilon^n, \quad \varepsilon^n \sim N(0, B), \quad (4.9)$$

where B is specified a priori.

Forward model.

The forward model is a user-chosen model that maps the latent states X_t to a prediction at time $t + T$. There are a variety of different choices for forward models for our system, and our choice of forward model is the Gipps model described in Section 4.2. Since this model describes traffic behavior for a 1D system along a straight line, some preprocessing steps were taken to create a dataset compatible with this forward model. Appendix A will describe the preprocessing steps that were applied to the dataset. We will mention alternative car-following models in Section 7.1.1 that describe 2D traffic models, which may be more

suitable for our dataset.

Forward model error covariance.

We assume that the forward model error covariance in equation (4.4), Q_t , is a matrix with nonzero diagonal variances and nonzero covariances between vehicles that are spatially close to each other.

Observation noise error covariance.

We assume that the observation error covariance in equation (4.5), R_t , is a diagonal matrix with dimension $d_{y_t} \times d_{y_t}$. More specifically, in this work, we assume this matrix has the form $R_t = rI_{d_{y_t}}$, where $r \in \mathbb{R}$. This choice of matrix assumes that the LiDAR sensor's measurement errors for different objects are independent (based on the diagonal specification), and the errors across vehicles are distributed with the same magnitude (based on the scalar r specification).

Observation operator.

In our setting, this matrix is in $\{0,1\}^{d_{y_t} \times d_{x_t}}$. This matrix has the following form:

$$H_{ij,t} = \begin{cases} 1 & \text{if } Y_{i,t} \text{ is the observation of quantity } X_{j,t}, \\ 0 & \text{otherwise,} \end{cases} \quad (4.10)$$

where $i = 1, \dots, d_{y_t}$ and $j = 1, \dots, d_{x_t}$. For example, if $Y_{i,t}$ is the observation for a certain vehicle's position and $X_{j,t}$ is the true position for that same vehicle, then $H_{ij,t} = 1$. On the other hand, if $X_{j,t}$ were the velocity for that same vehicle, $H_{ij,t}$ would be zero since $Y_{i,t}$ would not be a measurement of the quantity $X_{j,t}$.

5 Results

5.1 Objective 1: Assigning Labels to Unknown Trajectory Directions

The procedure for assigning labels is as follows:

(1) We extracted data with unknown trajectory directions from the given dataset. The extracted data contains the following information of each vehicle: ID, labels of traveling directions (from, to), the first measured time, the last measured time, initial x , initial y , last x , last y , initial velocity, and initial acceleration.

(2) We quantified the similarities between the observed trajectory and all reference trajectories by calculating mean squared error (MSE) described below:

$$\text{MSE}_j = \frac{1}{N} \sum_i^N \{y_i - f_j(x_i)\}^2, \quad (5.1)$$

where

- y_i and x_i are the i th observed coordinate of y and x , respectively.
- $f_j(x_i)$ is the y -coordinate of the j th reference trajectory on the i th observed x -coordinate x_i .

Please refer to [Appendix A](#) for details on f .

- N is the number of observed points (x_i, y_i) of any objects. (x_1, y_1) and (x_N, y_N) are the initial point and the last point, respectively.

(3) [Figure 4](#) shows an example of label assignment. Using a vehicle trajectory (orange line) starting from "East" and ending at "Unknown", we replaced "Unknown" with the direction label of a reference trajectory. The orange line represents a partial trajectory of the vehicle, which was lost from the LiDAR sensors' view before exiting the intersection. After calculating MSE_j for all reference trajectories, we assigned the vehicle as the label "North" of the reference trajectory with the minimum MSE_j (blue line).

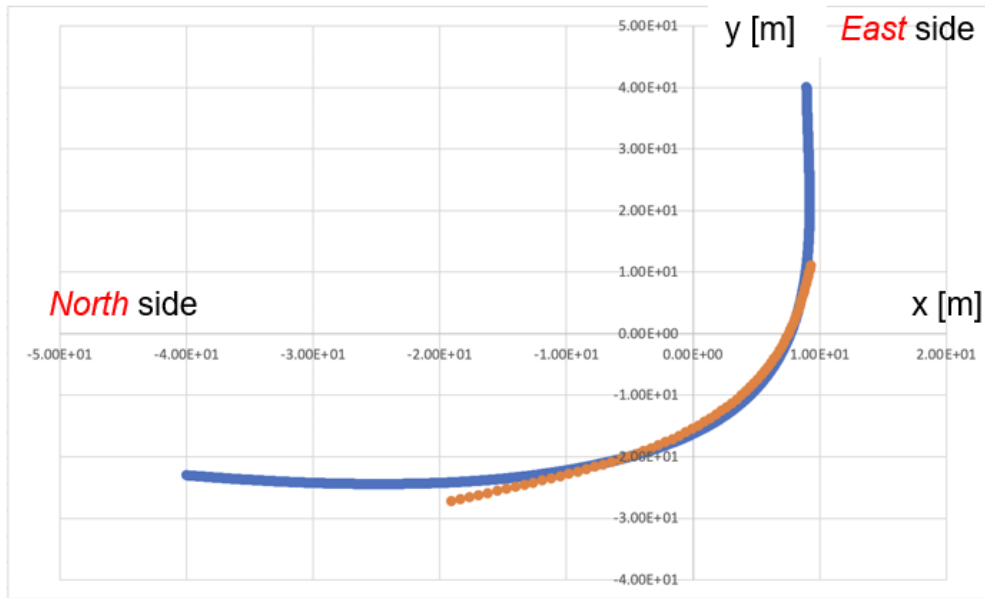


Figure 4: An example of label assignments.

5.2 Objective 2: Reducing Noise in the LiDAR Sensors

To reduce noise of the LiDAR sensor data, we can utilize the EnKS framework. Results shown in this section contain estimated positions, velocities, and accelerations over time for one vehicle. The estimates for this vehicle were computed independently from any other vehicles in the dataset, which does not account for information about surrounding cars that would aid in estimating this vehicle's information. We note this limitation in mind when interpreting our results.

The results show EnKF, EnKS, and Gipps forecasting compared to the observational LiDAR data. In these results, for EnKS, we set $L = 7$, and in both EnKF and EnKS, we set $R = rI$, where $r = 0.05$, and $Q = qI$, where $q = 0.5$, and $N = 100$. The parameters of the Gipps model in this example are shown in [Table 1](#).

Table 1: The parameter values in this study [10]

Parameter	Value	Unit
v_n^{desired}	15.3	m/s
a_n^{max}	1.77	m/s ²
$ d_n^{\text{max}} $	3.51	m/s ²
$ \hat{d}_{n-1} $	6.3	m/s ²
s_{n-1}	5.9	m
T	0.6	s

The estimated time series of positions, velocities, and accelerations are shown in Figure 5, Figure 6, and Figure 7, respectively. Figure 5 and Figure 6 demonstrate that Gipps forecast and EnKF/EnKS analysis can be assimilated into the trend of positions and velocities. As shown in (4.1)-(4.3), the forward model has no information on second or higher derivative. Thus, the acceleration cannot be estimated, and the noise is amplified compared to the complete observation.

Meanwhile, Figure 6 indicates that the forward model highly contaminated the smoothed velocities due to its large uncertainty. Based on the current specification of the Gipps model, specifying q to be large and r to be small would be most optimal since we see that the Gipps forecasts are unreliable. If we were to change q and r in this way, we would receive EnKS and EnKF results that closely match the observations. To obtain results that are less noisy compared to the observational data, we need a forward model whose errors are on a smaller scale compared to the observations. With this in mind, we suggest future directions for consideration for improving the forward model in Section 7.1.

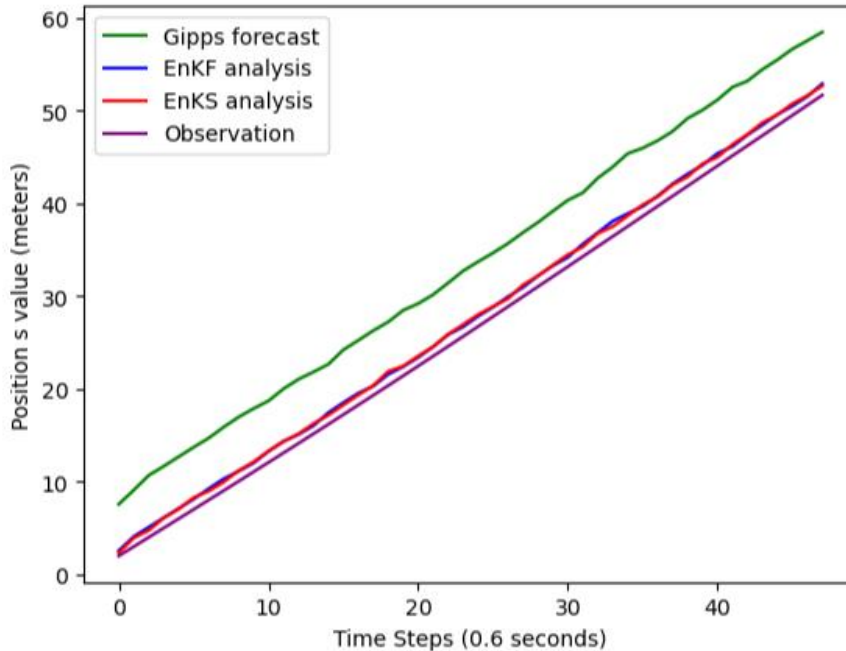


Figure 5: Comparison of time series of positions.

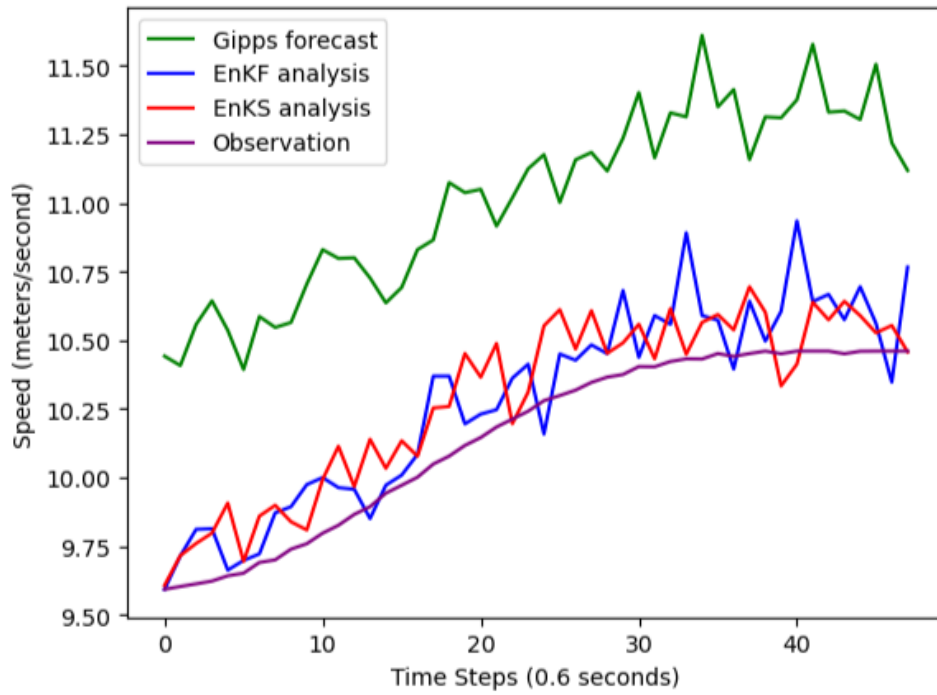


Figure 6: Comparison of time series of velocities.

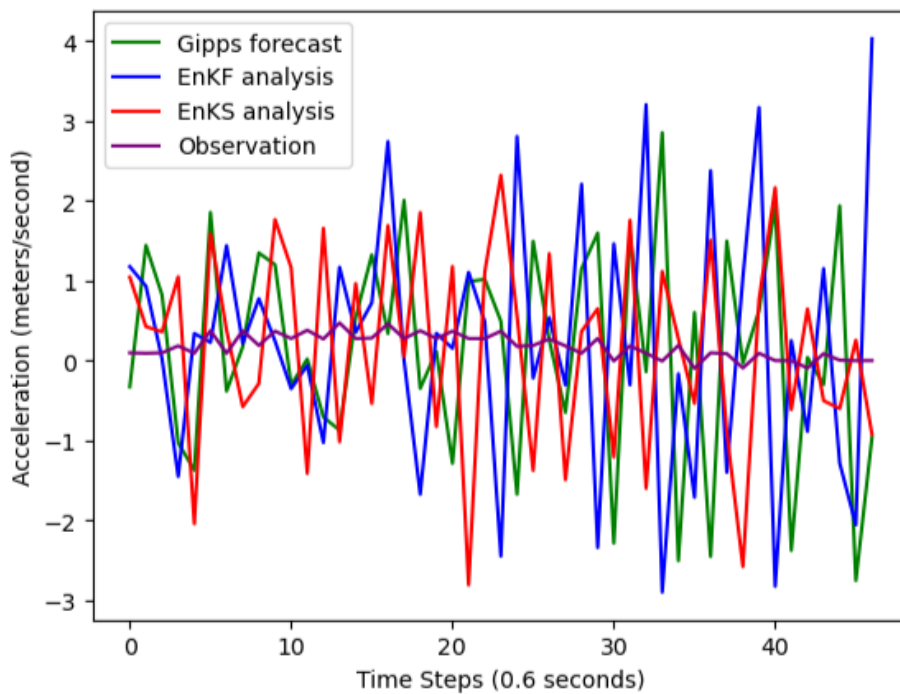


Figure 7: Comparison of time series of accelerations.

5.3 Objective 3: Extrapolating Trajectories

We present pseudo code below in [Algorithm 3](#) for how EnKS can extrapolate vehicle trajectories that are hidden in the dataset, and how to account for the dynamic nature of the latent space and observation space dimensions. We provided code to IHI related to the steps of [Algorithm 3](#), however, we note that we have not integrated all steps of this algorithm in our code.

Algorithm 3. Extrapolating Trajectories

- 1: **Initialize** $X_0^{1:N}$.
 - 2: **for** $t = 1, \dots, T$ **do**
 - 3: Apply the Gipps forward model from equation (4.1)-(4.3) to $X_{t-1}^{1:N}$.
 - 4: Check if the latent dimension is correct.
 (remove vehicles no longer in our region of interest, unless their observed position is still in the region, and add vehicles that enter the region)
 - 5: Construct a new H_t based on which latent states are observed versus unobserved including hidden vehicles.
 - 6: Construct R_t and Q_t based on d_{y_t} and d_{y_t} , and apply the EnKS from [Algorithm 2](#).
 - 7: **end for**
 - 8: **Output:** $X_{1:T}^{1:N}$, which includes estimated speeds, positions, and accelerations for both hidden and observed vehicles.
-

6 Conclusion

We have implemented a method for assigning labels to trajectories with “unknown” travel directions. This approach applies to our data preprocessing steps, which includes mapping the data from a 2D format to a set of 1D trajectories. Regarding our EnKS results, we acknowledge that work still needs to be done in optimizing the user-chosen parameters, as well as specifying the best setup for EnKS. We provide more details on further directions in [Section 7](#).

7 Further Directions

This preliminary attempt to denoise observed traffic trajectories and extrapolate missing trajectories can be extended to account for more realistic assumptions for an intersection. For example, this approach can be extended to include a 2D car-following model as the forward model, which would capture information about vehicle lane changes in the intersection. Additionally, we suggest methods for jointly learning the forward model parameters and estimating the state space quantities of interest to further improve the results.

7.1 Improvements to the Forward Model

Our team sees two avenues to improve upon the EnKS results regarding the specification of the forward model, the first of which would be to consider vehicle prediction models that can produce predictions in a 2D space, and the second of which would be to consider implementing an approach to jointly learn or tune the forward model dynamics while estimating the latent state space. We consider these two potential avenues to be the most fruitful in improving the results.

7.1.1 2D Car-Following Models

There is less research literature on 2D car-following models compared to 1D models. However, new modeling methods have been discussed in recent years, and further expansion is expected in the future [13].

7.1.2 Joint Forward Model Learning and State Space Estimation

This framework can be further improved to utilize the observational data to improve the forward model parameterization. One such method is the auto-differentiable ensemble Kalman filter [14], which alternates between estimating the latent states X_t given a set of parameters and updating this parameter set by autodifferentiating the loss function and taking a gradient step to minimize the loss function. This method illustrated superior performance compared to an expectation-maximization approach to learning the forward model parameters.

7.2 Improvements to the Data Assimilation Framework

We see a few areas of improvement that could be made regarding our application of EnKS, specifically related to including covariance tapering and assimilating temporally intermediate observations. We consider these two directions to be important to consider improving upon the current results.

7.2.1 Covariance Tapering

Covariance tapering is a standard technique in applications of data assimilation where the latent states represent some sort of spatial information, such as in our case of tracking information about individual cars in an intersection. In our setting, we can define a tapering matrix $\rho_t \in \{0,1\}^{d_{x_t} \times d_{x_t}}$ that describes which values of \hat{C}_t that we would like to estimate [15]. Covariance tapering is applied in the EnKF and EnKS formula by replacing \hat{C}_t with $\rho_t \circ \hat{C}_t$, where \circ denotes the Hadamard product [15]. The goal of this matrix is to zero out correlations between states that we would expect to be zero, such as two vehicles that are spatially far apart or on different roads, which increases our effective ensemble size and leads to better approximations [15].

7.2.2 Assimilating Temporally Intermediate Observations

We note in this work that we discard observations that are temporally in between the forward model time jump. For example, the forward model maps the latent states from time t to time $t + T$. Observations from time points t and $t + T$ are assimilated, but the intermediate observations at $t + \frac{T}{6}$, $t + \frac{T}{3}$, etc. are not assimilated. A further extension of this work could be to extend the current implementation to be able to assimilate these temporally intermediate observations.

References

- [1] Realizing a sustainable mobility society with traffic flow data. IHI Engineering Review, 56(1), 2023.
<https://www.ihi.co.jp/en/technology/techinfo/contents`no/1199408`13586.html>
- [2] 3D Laser Radar. IHI Corporation.
<https://www.ihi.co.jp/3DLaserRadar/en/index.html>.
- [3] Level crossing obstacle detection system. IHI Corporation.
<https://www.ihi.co.jp/3DLaserRadar/en/products/01.html>.
- [4] Intelligent Transport Systems(ITS) Equipment. IHI Corporation.
<https://www.ihi.co.jp/3DLaserRadar/en/products/02.html>.
- [5] Hafiz Usman Ahmed, Ying Huang, and Pan Lu. A review of car-following models and modeling tools for human and autonomous-ready driving behaviors in micro-simulation. Smart Cities, 4(1):314-335, 2021.
- [6] Geir Evensen, Peter Jan, and Van Leeuwen. An ensemble kalman smoother for nonlinear dynamics. Monthly Weather Review, 128:1852-1864, 2000.
- [7] Johan Janson Olstam and Andreas Tapani. Comparison of Car-following models. Swedish National Road and Transport Research Institute: Linkoping, Sweden, 2004.
- [8] P.G. Gipps. A behavioural car-following model for computer simulation. Transportation Research Part B: Methodological, 15:105-111, 1981.
- [9] The Mathematical Society of Traffic Flow. Optimal velocity model.
<http://traffic.phys.cs.is.nagoya-u.ac.jp/~mstf/sample/ov`e.html>.
- [10] Luis Vasconcelos, Ana B. Silva, Alvaro M. Seco, Paulo Fernandes, and Margarida C. Coelho. Turboroundabouts: Multicriterion assessment of intersection capacity, safety, and emissions. Transportation Research Record, 2402(1):28-37, 2014.
- [11] Roger R Labbe Jr. Kalman and Bayesian Filters in Python. 2020.
- [12] Geir Evensen. The Ensemble Kalman Filter: Theoretical formulation and practical implementation. Ocean Dynamics, 53(4):343-367, 2003.
- [13] Jing Zhao, Victor L Knoop, and Meng Wang. Two-dimensional vehicular movement modelling at intersections based on optimal control. Transportation Research Part B: Methodological, 138:1-22, 2020.
- [14] Yuming Chen, Daniel Sanz-Alonso, and Rebecca Willett. Auto-differentiable ensemble kalman filters,

2021.

- [15] Thomas M. Hamill, Jeffrey S. Whitaker, and Chris Snyder. Distance-dependent filtering of background error covariance estimates in an ensemble kalman filter. *Monthly Weather Review*, 129(11):2776 - 2790, 2001.
- [16] T.D. Barfoot and C.M. Clark. Motion planning for formations of mobile robots. *Robotics and Autonomous Systems*, 46(2):65-78, 2004.

Appendix A. Data Preprocessing

Because the car-following model of interest operates only in scenarios where the road is a straight path with only one lane, we needed to preprocess our 2D dataset so that it is compatible with the methodology we would like to use. This preprocessing does throw away useful traffic information, such as lane changes between vehicles, but we noted this limitation in this document and suggest remedies for this limitation in [Section 7](#).

A.1 Encoding Reference Trajectories

We experimented with linear interpolation, cubic spline, and quadratic spline techniques to produce suitable smooth curves representing reference trajectories. [Figure 8](#) shows an example of interpolated results of the reference trajectory shown in [Figure 4](#).

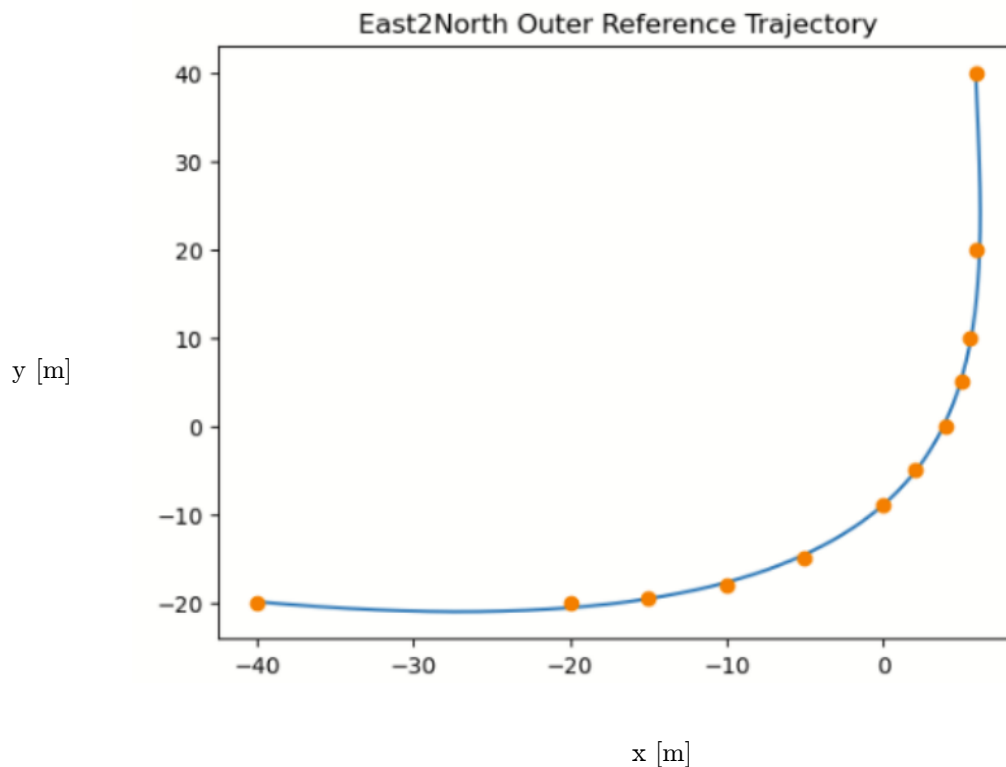


Figure 8: An example of label assignments.

A.2 Spatial Region of Interest

While we have traffic data for each of the four directions prior to the intersection, we are primarily concerned with the behavior of the cars in the intersection. Thus, we restrict our data to the points contained in the box representing the intersection, in which each edge is a crosswalk. The below image as shown in [Figure 9](#) demonstrates the lanes on which the cars can travel in the intersection (in blue) as well as the restricted box (in orange).

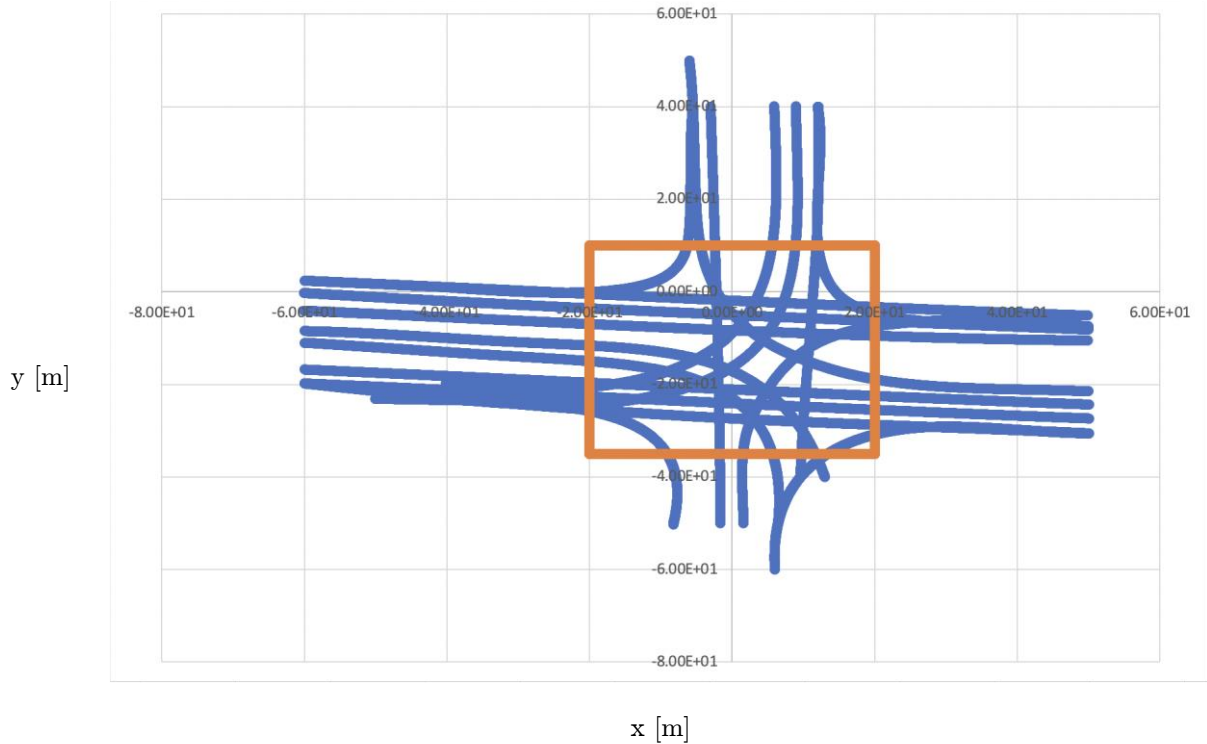


Figure 9: Spatial region of interest in this project.

A.3 2D to 1D Coordinate Conversion

Our smoothing and project goals necessitate one-dimensional coordinates as inputs to the forward model. Thus, we convert from Cartesian coordinates to curvilinear coordinates. However, we specifically choose the length along the reference trajectory, s , as our one-dimensional coordinate of interest. We initially used the curvilinear conversion seen in [\[16\]](#). This entailed the conversion from Cartesian to curvilinear coordinates in which:

$$s = \frac{1}{\kappa_b} \arctan\left(\frac{\kappa_b x}{1 - \kappa_b y}\right) \quad , \quad (\text{A.1})$$

and

$$n = \frac{1}{\kappa_b} - \frac{x}{\sin(\kappa_b s)} \quad . \quad (\text{A.2})$$

This resulted in (s, n) coordinates that closely resembled (x, y) coordinates. We expected s to be the distance along the reference trajectory, and we expected n to be the closest distance from a point to the reference trajectory. This would imply only positive s , for example. However, our s values and x values were equivalent, and the same was true for our n and y values. Even before this issue, we intended to only consider the s coordinate. In the intersection box, there are rarely lane changes, so n can be discounted without significant impact.

However, our s results were not appearing as we had expected them to, so we found an alternative way to calculate s . For a given Cartesian car trajectory coordinate, we find the closest point α along the reference trajectory and calculate the distance along the reference trajectory from its start to α . As we previously discussed encoding the reference trajectories, we measure distance along the reference trajectory via summing linear distances between discretely sampled points along the reference trajectory. We use this method to generate s coordinates.