A Partitioned Experience Method for Trajectory Prediction of Pedestrians

MSc Thesis

Anish Sridharan

A Partitioned Experience Method for Trajectory Prediction of Pedestrians

MSc Thesis

by

Anish Sridharan

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday October 14, 2022.

Student number:	5311209	
Project duration:	December 1, 2021 – October 14, 2022	
Thesis committee:	Dr. L. Ferranti,	TU Delft, supervisor
	Ir. O. de Groot,	TU Delft, daily supervisor
	Prof. dr. J. Alonso Mora,	TU Delft
	Dr. L. Laurenti,	TU Delft

This thesis is confidential and cannot be made public until October 14, 2022.

TU Delft

Abstract

For travelling from point A to point B, autonomous vehicles generate a route between the points. During the mission, the vehicle uses a motion planning and controls algorithm to follow the planned route while avoiding static and dynamic obstacles. Motion planning algorithms generally plan over a future time horizon to smoothly follow the route and determine the car's optimal control (steering/acceleration). For planning through a future horizon, one requires the possible positions of all the relevant obstacles in future time-steps. Solutions for predicting an obstacle's future trajectory usually involve neural networks to perform sequence learning and generative algorithms to create multiple possibilities for a pedestrian's future state. This is done by attempting to learn the underlying distribution describing the obstacle motion. However, in practice, one cannot evaluate if this learnt distribution is accurate. This thesis addresses this issue by introducing a fully data-based alternative for trajectory prediction called the Partitioned Experience Method (PEM), which predicts future trajectories based solely on previously recorded data. In this way, it is not necessary to explicitly learn the underlying distribution of the pedestrian motion. The implemented trajectory prediction is validated using two metrics, recall and variance, introduced in this work. The trajectory prediction is also evaluated using a state-of-the-art motion planning algorithm. The results obtained from the motion planner indicate that using the Partitioned Experience Method (PEM) reduces the number of collisions and close contacts with other road users, and the corresponding trajectory followed by the car is closer to the reference trajectory.

Table of Contents

List of Figures

List of Tables

Acknowledgement

I want to thank Dr. Laura Ferranti for granting me the opportunity and supervising my thesis research. Her feedback and support helped me through all stages of the research. I am incredibly grateful to my daily supervisor Oscar. He has always been there, answering my every doubt and having long discussions that were valuable for completing my work.

I would like to thank my parents, Sridharan R and Banumathi S, and my brother Saish. Without them, my journey in TU Delft would not have been possible. It is with their love and prayers that I have been able to achieve my dreams

Finally, I would like to thank my friends, Denesh, Iva, Matthijs, Tarun, Revanth and Vivek, who have provided me with a lot of the support I needed throughout my TU Delft life and have also made the last couple of years more fun than I could imagine.

Delft, University of Technology
October 2022

Anish Sridharan

Chapter 1

Introduction

Historically, people lived and worked within a few-mile radius though-out their lives. The invention of automobiles and the mass production of the same by Ford transformed the way people lived. People no longer were bound by where they lived and could more easily travel long distances and live their life with more freedom and autonomy. With more nations developing and the richer the commoner became, the number of cars has also drastically increased [3].
The sudden and mass adoption of automobiles also has some adverse effects. First, the burning of fuel and car emissions have contributed to climate change affecting us all. Secondly, with the increase in the number of cars, the number of accidents, and the time taken to travel have also increased. The first challenge is being tackled by using cleaner fuel to run the car, such as electricity. The rise of self-driving cars looks to solve the problem of travel time and accidents. This thesis will look into self-driving cars and how they tackle the problem of predicting the motion of other obstacles around the car.

1-1 Automated Driving

Car manufacturers have been trying to implement assistive features called Advanced driver-assistance systems (ADAS), which would improve the driver's safety and comfort. Early ADAS included features such as cruise control, blind spot information, lane departure warning etc. [4]. These features still involve the human being as the main driver, while the ADAS acts as an assistant. With the rapid progress in computers and artificial intelligence, the number of tasks that can also be automated increased. This led to a standardization of the definition of the level at which a car is automated. SAE [5] defined six levels of automation of an autonomous vehicle, ranging from Level 0 to Level 5. Level 0 means the car is manually controlled, and each subsequent level has a higher degree of driver assistance and autonomy. Level 5 indicates that the car is fully autonomous and can handle any situation without a human driver. A lot of the current research focuses on developing Level 4, and Level 5 autonomous cars, with an increasing number of companies [6], [7] aiming to develop autonomous

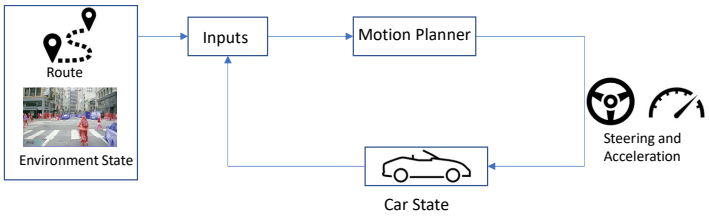


Figure 1-1: Motion Planning

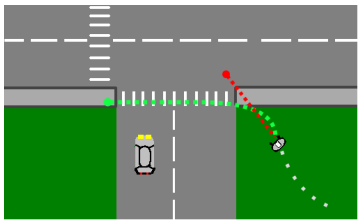


Figure 1-2: Trajectory Prediction [1]

taxis and other similar solutions to improve transport logistics.
A completely automated car can travel from starting to a goal location without human interference. This includes a route planning system, which determines the best route between A and B considering the time to travel and traffic situations, a perception system that detects the state of the environment and other obstacles in it, and finally, a state-estimation system that calculates the current position and movement of the car. The local motion planner takes all these as input and tries to follow the route by giving the required controls to the car.
Local motion planning algorithms try to plan over the future horizon by satisfying some constraints and then give the estimated control of acceleration and steering required to the car for the next step. This is done in order to ensure smooth motion. One of the constraints these algorithms need to satisfy is based on the current and future state of the environment. The environment consists of static obstacles such as sidewalks, poles, and lane dividers that do not move, and their position in a future time instant can be known. The environment also consists of dynamic obstacles, who are the other road-users such as pedestrians, cyclists and other vehicles on the road. These positions are not fixed or known beforehand. So for the motion planner to work, there needs to be an estimation or a prediction of where other road users could be at a future time instant, i.e. predict their trajectory for a given time horizon.
This thesis deals with trajectory prediction by introducing a new framework called the Partitioned Experience Method. The framework is then used for the trajectory prediction of the pedestrian. This framework is evaluated using a local motion planning method called the scenario approach to optimization [8].

1-2 Trajectory Prediction Approaches

Traditionally, trajectory prediction algorithms attempted to predict a single best prediction for an obstacle [9], [10]. These algorithms are known as uni-modal trajectory prediction algorithms. A uni-modal trajectory is shown in Fig. 1-3
Since the exact intention of an obstacle is unknown, predicting one trajectory per obstacle would not accurately represent its motion. Recently, the approach has turned towards predicting multiple possible trajectories for an obstacle [11], [12], [13], trying to cover the various possible intents of the obstacle. These approaches are known as multi-modal trajectory prediction algorithms. A multi-modal trajectory is shown in Fig. 1-4

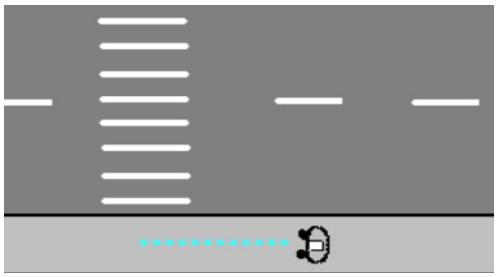


Figure 1-3: Uni-modal trajectory prediction

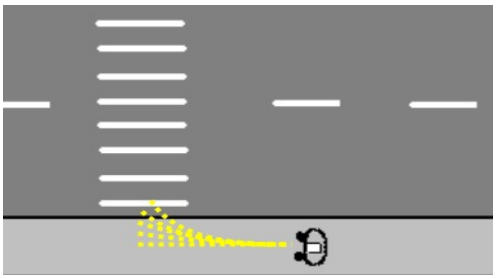


Figure 1-4: Multi-modal trajectory prediction

1-2-1 Problem Formulation

A generalized problem formulation of trajectory prediction is explained below: Let e_t be the state of the environment consisting of v obstacles, at current time t, o_t^v is the state of the v^{th} obstacle at the same time step. Then, $e_t - o_t^v$ is the state of all other obstacles at time t, other than the v^{th} obstacle. The problem is to predict the trajectory of the v^{th} object. The inputs for a general trajectory prediction problem would involve the state of the environment at the current and previous time steps.

$$X = \begin{bmatrix} o_t^v \ , \ e_t - o_t^v \\ o_{t-1}^v \ , \ e_{t-1} - o_{t-1}^v \\ o_{t-2}^v \ , \ e_{t-2} - o_{t-2}^v \\ . \\ . \\ . \\ o_{t-k}^v \ , \ e_{t-k} - o_{t-k}^v \end{bmatrix} \tag{1-1}$$

The states of the environment here can be any available feature of the obstacle of interest, such as the position and velocity of the obstacle.
The output of the problem consists of the position of the v^{th} obstacle at every time step in the future horizon N. For a uni-modal algorithm, m=1, and a multi-modal algorithm $m > 1$:

$$Y = f(x) = \begin{bmatrix} \hat{x}_{1(t+1)}^v \ , \ \hat{y}_{1(t+1)}^v \\ \hat{x}_{2(t+2)}^v \ , \ \hat{y}_{2(t+2)}^v \\ \hat{x}_{3(t+3)}^v \ , \ \hat{y}_{3(t+3)}^v \\ . \\ . \\ . \\ \hat{x}_{k(t+N)}^v \ , \ \hat{y}_{k(t+N)}^v \end{bmatrix}_1 , \begin{bmatrix} \hat{x}_{1(t+n)}^v \ , \ \hat{y}_{1(t+n)}^v \\ \hat{x}_{2(t+n)}^v \ , \ \hat{y}_{2(t+n)}^v \\ \hat{x}_{3(t+n)}^v \ , \ \hat{y}_{3(t+n)}^v \\ . \\ . \\ . \\ \hat{x}_{k(t+n)}^v \ , \ \hat{y}_{k(t+n)}^v \end{bmatrix}_2 ...., \begin{bmatrix} \hat{x}_{1(t+n)}^v \ , \ \hat{y}_{1(t+n)}^v \\ \hat{x}_{2(t+n)}^v \ , \ \hat{y}_{2(t+n)}^v \\ \hat{x}_{3(t+n)}^v \ , \ \hat{y}_{3(t+n)}^v \\ . \\ . \\ . \\ \hat{x}_{k(t+n)}^v \ , \ \hat{y}_{k(t+n)}^v \end{bmatrix}_m \tag{1-2}$$

A detailed literature survey is present in the related works section of the paper in Chapter 3.

1-3 Limitations in Current Research

1-3-1 Trajectory Prediction

Current state-of-the-art algorithms attempt to model obstacle behaviour through data. The underlying distribution of the data is modelled to learn information such as possible goal locations and intentions. When a query is made to predict an obstacle's trajectory, multiple possible intentions or goal positions are sampled from the distribution, giving an output of multiple trajectories.

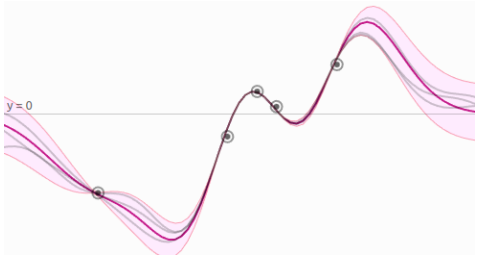


Figure 1-5: Uncertainty in any part of a learned distribution depends on the concentration of the data points at that part of the disribution [2]

There are some inherent issues with such algorithms. First, a large amount of data is necessary to learn the distribution accurately. Insufficient data might lead to high variance in a region of the learned distribution, as shown in Fig. 1-5, where the parts of the distribution defined by fewer data points have high variance. In such situations, an accurate trajectory is only produced if the query is in the well-defined part of distributions, leading to noise/erroneous predictions for edge cases and low-probability queries.[14] and [15] bring up these issues with Generative Adversarial Network (GAN) and the authors conclude that GANs might not accurately learn the target distribution, and even if they do succeed, there is a lack of an established way to prove the success, i.e., there is an assumption made on the accuracy of the modelled distribution. Thus, the risk associated with the predictions can only be found empirically, and the real risk is unknown.
Second, these algorithms try to predict a few modes (goals) and then sample trajectories from these modes [13], [16], [17]. This could mean that not all relevant possibilities of motion are predicted.
Third, multi-modality is usually achieved by sampling individual trajectories. Sampling multiple trajectories is a sequential task, the time taken increases with the number of samples required. This means that it is not possible to predict a high number of samples, which is helpful for constraint satisfaction in sampling-based motion planning algorithms, such as [8].

1-3-2 Trajectory Evaluation

The usual metrics to evaluate trajectory predictions are the Average Displacement Error (ADE) and the Final Displacement Error (FDE).

- Average Displacement Error (ADE) is the mean L_2 distance between the best predicted of N randomly sampled output trajectories and the ground-truth trajectory position at all times of predictions [18].
- Final Displacement Error(FDE) is the L_2 distance between the best predicted of N randomly sampled output trajectories and the ground-truth trajectory position at final time-step of prediction [18].

These metrics do not describe the distribution of the prediction. [13] introduced the use of recall and precision in trajectory prediction.

- Precision is the measure of how much of the predicted distribution is covered by the ground-truth distribution. A precision of one would mean that all the predicted trajectories are also a part of the ground truth distribution, and no predictions lie outside of the ground truth.
- Recall measures how much of the can be ground-truth is covered by the predicted distribution. A recall of one means the ground truth is completely covered by the predictions.

The precision and recall as described here tries to evaluate the entire distribution, rather than just figuring out how good the best trajectory is. This is important for motion planning scenarios where the entire distribution of the predicted trajectories helps in defining the obstacle space. A shortcoming of the metrics that are used in [13] is that they only work if the ground-truth consists of a multimodal output.

1-3-3 Research Direction

In order to overcome these issues, this thesis introduces the Partitioned Experience Method (PEM) for trajectory prediction. The method is motivated by the scenario approach [19]. Instead of learning the distribution from available data, the data itself is used as the output of the trajectory prediction. A dataset containing pedestrian motion is divided into multiple partitions. Each partition contains trajectories of a unique situation (walking, running or idle pedestrians). According to the state of motion of a detected obstacle, a specific partition is chosen, and the trajectories present in these partitions are directly used as the predicted output.
By directly using the data, Partitioned Experience Method (PEM) does not have to learn the underlying data distribution. Each sample represents a unique and real trajectory, thus representing a unique mode. More modes can be sampled, covering relevant possibilities. Finally, since the trajectories are directly sampled from a stored dataset and not generated individually, a high number of trajectories can be sampled quickly.

1-4 Contribution

- A new method for multi-modal trajectory predictions, the Partitioned Experience Method (PEM), based on the scenario approach, is introduced and implemented to predict pedestrian motion. A dataset containing pedestrian motion is divided into different partitions

separating different kinds of motion. Based on the observed pedestrian motion, a suitable partition is chosen, and trajectories are sampled from it.

- This effectiveness of the trajectory prediction is analysed using the scenario-based motion planner.
- New evaluation metrics, called recall and variance for evaluating trajectory prediction algorithms, which work on uni-modal ground truth data, are defined to measure how well the predictions capture the multi-modal distribution.

The rest of the report contains some preliminaries which would be useful in understanding the thesis, followed by the thesis, written in a research paper format.

Chapter 2

Preliminaries

This chapter introduces some background work related to the thesis, namely the scenario approach and different trajectory prediction algorithms. It is also included in the paper in the appendix.

2-1 Scenario Approach

The scenario approach [19] is an optimization method that uses data from historical observations as the constraints of the optimization problem. The number of historical samples used for the output affects the problem's risk of constraint violation with the cost of the solution. More predicted samples lead to a higher number of constraints, which reduces the risk of the solution but degrades the performance. The risk vs feasibility is explored in detail in [20] and [21], where the amount of data required to satisfy a particular risk is calculated for various problems. This method has also been used in other fields in time-series problems to predict future trends, such as power generation [22],[23], medical applications [24], and finance [25]. Similarly, trajectory prediction defined in this thesis is also a time-series problem, so the scenario approach is used to sample pedestrian motion from a partitioned database to predict future trajectories.

2-1-1 Scenario Approach in Motion Planning

In autonomous driving, the constraints are uncertain since the positions of obstacles (static and dynamic) are not precisely known. To overcome this, these positions are usually described using uncertainty around their *believed* position. The scenario approach comes into the picture here. [8] builds upon a Model Predictive Contouring Control (MPCC) framework called Local MPCC [26]. The way constraints are dealt with is inspired by [21], i.e. the current and possible future positions of the obstacles are represented as scenarios that act as constraints. The algorithm is as follows:

- The chance constraints of the dynamic obstacle are linearized.
- These linearized chance constraints are sampled, and deterministic constraints called scenarios are drawn.
- The risk of planning is linked with the number of scenarios, with lesser scenarios drawn implying higher risk.

Problem Definition

Consider a robot at its current state, $x_t \in \mathbb{X}$ navigating through an environment at state e_t containing both static and dynamic obstacles. The task followed by the robot is to follow a trajectory; the problem after linearizing the chance constraints and sampling the deterministic scenarios is given by:

$$\min_{u \in U} \quad \sum_{k=1}^{N} J(x_k, u_k) \tag{2-1a}$$

$$s.t. \quad x_{k+1} = f(xk, uk), x \in \mathbb{X} \tag{2-1b}$$

$$A_k^T(\delta_k^i, \hat{x}_k) x_k \leq b_k(\delta_k^i, \hat{x}_k) \tag{2-1c}$$

$$x_0 = x_{init} \tag{2-1d}$$

where x_k and u_k denote the states and inputs of the robot and $\hat{x}_k$) is the k-step ahead prediction of the robot,

$$A_k = \frac{\delta_k - \hat{x}_k}{||\delta_k - \hat{x}_k||}, b_k = A_k(\delta_k - A_k r) \tag{2-2}$$

is the linearized collision region with respect to the predictions of the robot state at a given stage k, u is the control input given to the robot.
The set of deterministic constraints is obtained by sampling each dynamic obstacle's position from uncertainty (δ_k^v) at a time-step k and getting the collision constraint. Each sample is a possible position of the dynamic obstacle v at a time-step k. The future positions of each obstacle are not observable and time-variant; that is, samples at each time step cannot be used for other time-steps.

Risk Bounding

The paper also has some theorems on risk bounding and some methods for sample pruning so that less number of samples are used in the optimization and the solution is tractable.
Theorem 1 of the paper states that the probability that the solution violates the chance constraint at a stage k in optimization is:

$$\mathbb{P}_k^{S_k} \left[V_k \left(\boldsymbol{u}_{SP}^* \right) > \epsilon_k \left(s_k^* \right) \right] \leqslant \beta_k \left(S_k \right) \tag{2-3}$$

where β is the confidence parameter that is defined by:

$$\beta_k \left(S_k \right) := \sum_{0}^{S_k - 1} \left(\begin{array}{c} S_k \\ s \end{array} \right) \left[1 - \epsilon_k(s) \right]^{S_k - s} \tag{2-4}$$

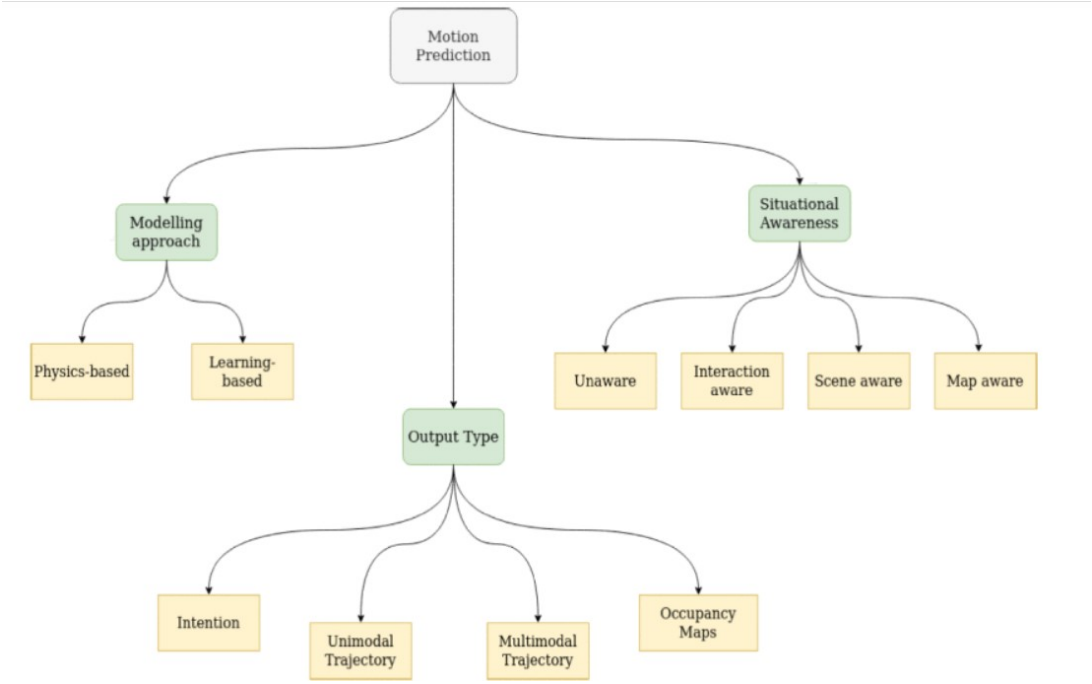


Figure 2-1: Taxonomy of trajectory prediction [1]

Thus the scenario approach for autonomous driving is a stochastic approach to the collision avoidance problem in local motion planning. The approach, unlike Local Model Predictive Contouring Control (LMPCC), is agnostic to the type of uncertainty of the dynamic obstacle. Another advantage of the approach is that the pruning of constraints, and the formal definition of the risk/ safety bounds, not only with the original scenario but also after pruning, allows the approach to be real-time.

2-2 Types of Trajectory Prediction Algorithms

This section describes the different ways in which trajectory prediction algorithms (as shown in Fig. 2-1) can be classified.

2-2-1 Based on Output

Trajectory prediction models can either be classifiers that try to classify the type of motion and object takes or predict one or multiple possible trajectories at every future time-step. The different types of possible outputs in a trajectory prediction model are described below:

Uni-Modal

These algorithms predict one trajectory per obstacle.

Multi-Modal

These algorithms suggest that one trajectory is not enough to cover all human intentions, hence predicting multiple possibilities of motion.

2-2-2 Based on Situational Awareness

Situational awareness refers to the data which the model takes as input. This would range from considering only one obstacle independently to considering multiple obstacles and their environment to give a more accurate prediction.

Unaware

These models try to predict the trajectory of an obstacle without considering its interaction with other obstacles/the entire scene itself. Some basic physics-based approaches described below, such as the constant velocity (CV) and constant acceleration (CA), are unaware models. These models are too simple, and since they do not consider the presence of other obstacles and the environment, they do not produce accurate long-term trajectories.

Interaction-Aware

Interaction-aware prediction models consider other obstacles in the path while estimating future trajectories. For a pedestrian, this would be using the trajectories of other pedestrians/vehicles and reasoning out that the pedestrian would try to avoid collisions and some possible paths. [9], [11] and [27] are some algorithms based on interaction aware modelling, where interaction between obstacles in a particular scene is considered by using Long Short-Term Memory (LSTM).

Scene Aware

These models use the interaction between all obstacles in the scene and the scene itself. For example, a pedestrian crossing the road would have a different type of movement when compared to walking straight ahead. These features are usually obtained from a sensor feed (like a camera) and fed to the model. Scene context adds physical constraints, which help in providing more realistic paths. [28] builds upon the social-LSTM work in [9], by embedding new factors encoding human-space interactions. [29] is another method that models both the physical terrain and the motion of other obstacles using a LSTM based Generative Adversarial Network (GAN) module.

Map Aware

Map Aware models use information from a High Definition (HD) map of the environment as additional information to estimate future trajectories. [30] makes use of HD maps. The approach uses maps to provide a structure to where obstacles can or cannot be. The future

trajectory of vehicles is predicted by constraining the vehicle path based on road geometry and constraints such as lane maintenance. [31] uses a neural network model called VectorNet to encode HD maps and agent dynamics in terms of vectors.

2-2-3 Based on Modelling Approach

The modelling approach refers to the algorithm based on a physics-based approach or a more general data-driven(learning-based) approach. Physics-based approaches rely on kinematics and dynamics of the obstacle to predict the motion. In learning approaches, the algorithm uses past obstacle data and other information like the road structure and motion of other obstacles to find a correlation with the obstacle's future trajectory.

Physics-Based Approaches

Physics-based approaches are algorithms where predictions are made following the rules of physics, either dynamic or kinematic models. Dynamic models consider the forces involved in creating the motion. For human trajectory predictions, it is not necessary to calculate these forces, and the dynamic modelling becomes complex and irrelevant [1]. Kinematic models describe the motion of the obstacles in mathematical form. There are many simple kinematic models where assumptions such as CV and CA are used. The obstacle is assumed to move at a constant velocity or acceleration based on its previous motion.

Learning-Based

Learning-based algorithms do not use complex physical-model to represent an obstacle; instead, they use data of similar obstacles and history to model the new trajectory. These approaches have become popular since the advent of Deep Learning (DL). The model might use sensor inputs such as velocity and position of the obstacle during its past trajectory and may also use the same to precept the scene in which the obstacle is present. Learning-based approaches can be uni-modal/ multi-modal, unaware/ interaction aware/ scene aware, or map aware. The output/ situational awareness type is usually based on the problem statement and the available sensor data. Literature [1] classifies learning algorithms into two types, sequential and non-sequential models.
Non-Sequential Models learn over the current data directly, and these models do not use data from previous frames to condition the output. [32] uses a 2D CNN for trajectory prediction and presents position normalization techniques and data augmentation techniques for the trajectory prediction problem. Sequential Models perform trajectory prediction based on the history of their past motion. These approaches try to capture long-term dependencies, using DL tools such as LSTM, Gated Recurrent Unit (GRU), Conditional Variational Autoencoders (CVAE) and GAN.
The table 2-1 summarizes different trajectory prediction algorithms and categorizes them based on the defined classification approaches.s

Trajectory Prediction			
Type of Learning Algorithms	**Type of Output**	**Type of Situational Awareness**	**Algorithms**
Sequential	Uni-Modal	Interaction Aware	Social LSTM [33], Group LSTM [10]
		Scene Aware	Graph2Kernel Grid-LSTM [34], Context Aware LSTM [35]
	Multi-Modal	Interaction Aware	Social GAN [11]
		Scene Aware	Social VRNN [36], Trajectron++ [12], Sophie [37], Social BiGAT [17], Goal-GAN [16], MG-GAN [13]

Table 2-1: Classifying Learning based state-of-the-art algorithms

Chapter 3

Paper

The thesis written in the form of a paper and is attached in the following few pages. A possible venue for the submission of the paper is IV Symposium 2023.

A Partitioned Experience Method for Trajectory Prediction of Pedestrians

Anish Sridharan
Cognitive Robotics
TU Delft
Delft, The Netherlands

***Abstract*—For travelling from point A to point B, autonomous vehicles generate a route between the points. During the mission, the vehicle uses a motion planning and controls algorithm to follow the planned route while avoiding static and dynamic obstacles. Motion planning algorithms generally plan over a future time horizon to smoothly follow the route and determine the car's optimal control (steering/acceleration). For planning through a future horizon, one requires the possible positions of all the relevant obstacles in future time-steps. Solutions for predicting an obstacle's future trajectory usually involve neural networks to perform sequence learning and generative algorithms to create multiple possibilities for a pedestrian's future state. This is done by attempting to learn the underlying distribution describing the obstacle motion. However, in practice, one cannot evaluate if this learnt distribution is accurate. This thesis addresses this issue by introducing a fully data-based alternative for trajectory prediction called the Partitioned Experience Method (PEM), which predicts future trajectories based solely on previously recorded data. In this way, it is not necessary to explicitly learn the underlying distribution of the pedestrian motion. The implemented trajectory prediction is validated using two metrics, recall and variance, introduced in this work. The trajectory prediction is also evaluated using a state-of-the-art motion planning algorithm. The results obtained from the motion planner indicate that using the PEM reduces the number of collisions and close contacts with other road users, and the corresponding trajectory followed by the car is closer to the reference trajectory.**

I. Introduction

Autonomous vehicles drive around in an unstructured environment, including pedestrians, cyclists and other obstacles. For safe navigation, modern motion planning algorithms plan over a future horizon. We require the various obstacles' current and future positions and motion.

Trajectory prediction is an estimation of the possible future position of an obstacle, given its past motion. Some methods predict one output trajectory per obstacle, called uni-modal prediction. Modern approaches, however, state that human intention is uncertain and cannot be predicted by just one trajectory. So multi-modal algorithms are used to predict multiple possible trajectories per obstacle.

Current state-of-the-art algorithms attempt to model obstacle behaviour through data. The underlying distribution of the data is modelled to learn information such as possible goal locations and intentions. When a query is made to predict an obstacle's trajectory, multiple possible intentions or goal positions are sampled from the distribution followed by the trajectories to achieve these.

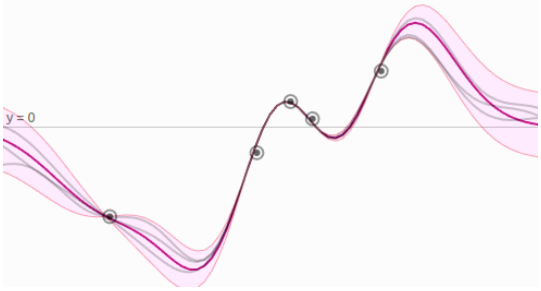


Fig. 1: Uncertainty in any part of a learned distribution depends on the concentration of the data points at that part of the distribution [3]

There are some inherent issues with such algorithms. First, a large amount of data is necessary to learn the distribution accurately. Insufficient data might lead to high variance in a region of the learned distribution, as shown in Fig. 1, where the parts of the distribution defined by fewer data points have high variance. In such situations, an accurate trajectory is only produced if the query is in the well-defined part of distributions, leading to noise/erroneous predictions for edge cases and low-probability queries. [1] and [2] bring up these issues with Generative Adversarial Network (GAN) and the authors conclude that GANs might not accurately learn the target distribution, and even if they do succeed, there is a lack of an established way to prove the success, i.e., there is an assumption made on the accuracy of the modelled distribution. Thus, the risk associated with the predictions can only be found empirically, and the real risk is unknown.

Second, these algorithms try to predict a few modes (goals) and then sample trajectories from these modes [4], [5], [6]. This could mean that not all relevant possibilities of motion are predicted.

Third, multi-modality is usually achieved by sampling individual trajectories. Sampling multiple trajectories

is a sequential task, the time taken increases with the number of samples required. This means that it is not possible to predict a high number of samples, which is helpful for constraint satisfaction in sampling-based motion planning algorithms, such as [7].

In order to overcome these issues, this thesis introduces the Partitioned Experience Method (PEM) for trajectory prediction. The method is motivated by the scenario approach [8]. Instead of learning the distribution from available data, the data itself is used as the output of the trajectory prediction. A dataset containing pedestrian motion is divided into multiple partitions. Each partition contains trajectories of a unique situation (walking, running or idle pedestrians). According to the state of motion of a detected obstacle, a specific partition is chosen, and the trajectories present in these partitions are directly used as the predicted output.

By directly using the data, PEM does not have to learn the underlying data distribution. Each sample represents a unique and real trajectory, thus representing a unique mode. More modes can be sampled, covering relevant possibilities. Finally, since the trajectories are directly sampled from a stored dataset and not generated individually, a high number of trajectories can be sampled quickly.

II. Related Work

A. Trajectory Prediction

1) Early Approaches: Initial approaches to trajectory predictions used approximations such as constant velocity/acceleration or simple physics-based rules to predict the trajectory [9], [10], [11]. Since it is not easy to model the exact dynamics of the scene, it is not practical to use purely physical models for the prediction of human motion.

Neural networks architectures based on Recurrent Neural Network (RNN) [12], like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) [13] became the prevalent way to model the long term dependencies of human motion. Social LSTM [14] was the first popular approach of this type, where LSTM networks were used to predict based on the motion of both the query pedestrians and other nearby pedestrians. Context-Aware LSTM in [15] added information about the surrounding scene to the better context of the motion. Group LSTM in [16] clustered pedestrians together in a group and used Social LSTM on the group as a whole.

These algorithms produce uni-modal trajectories. In case there are two available modes, the output ends up being the average of these, without representing either of the modes accurately [17], [18], [19], [20].

2) Multi-Modality: Variational and generative models such as the GAN are popular ways to solve the multi-modality problem of trajectory prediction.

Models using Variational Approaches

Conditional Variational Autoencoder (CVAE) [21] is an extension of Variational Autoencoder (VAE) [22], a neural network architecture that models latent variables and data to represent it as a lower dimension distribution, from which "fake" data can be sampled. Desire [23] is a method that uses CVAE to generate a diverse range of trajectories and then ranks the trajectories based on their likelihood. Trajectron++ [24], is another CVAE-based multi-modal approach to trajectory prediction, where the authors use dynamics and scene context, in addition to the ability to add sensors and High Definition maps for pedestrian/vehicle modelling.

A Variational Recurrent Neural Network (VRNN) [25] is also an extension of a VAE, that is used to model high dimensional sequences. Social-VRNN in [26] uses a scene-aware VRNN approach, where a Gaussian mixture model is used to generate multiple modes. The diversity of predictions is ensured by representing inputs as a gaussian distribution and sampling different inputs from it.

GAN Based Networks

GAN [27] is a neural network architecture containing a generator and a discriminator, contesting each other. The generators try to learn the underlying distribution of the data and create "fakes" from the same. The discriminator has to distinguish between real and fake data, thus pushing the generator to create more realistic data. Social-GAN [28] first used GAN for trajectory prediction. LSTMs are used for socially aware encoding of pedestrian motion, and the GAN attempts to generate multiple realistic trajectories from the distribution. Graph Attention Network (GAT) [6] is another approach that uses feature representations of multiple humans and static obstacles in the scene. Goal Gan [5] tries to predict multiple possible goal positions and then generate trajectories that lead to these goals. Finally, MG-GAN [4] is a recent work where the path mode network produces different modes. Multiple generators are used, each trying to sample from one generated mode. This ensures that every mode is independent of the other, effectively separating the different prediction modes.

Multi-modal algorithms have inherent issues, as described in the previous section. They attempt to model input data and learn the distribution, which could lead to the prediction of unrealistic trajectories. They also learn limited modes and then sample out of these modes instead of considering each possible trajectory as a separate mode by itself. Finally, since each trajectory is sampled individually from the learnt distribution, they are too slow to predict a high number of output trajectories.

3) Evaluation Metrics of Trajectory Predictive algorithms: The usual metrics to evaluate trajectory predictions are the Average Displacement Error (ADE) and the Final Displacement Error (FDE).

- Average Displacement Error (ADE) is the mean L_2 distance between the best predicted of N randomly sampled output trajectories and the ground-truth trajectory position at all times of predictions [29].
- Final Displacement Error (FDE) is the L_2 distance between the best predicted of N randomly sampled output trajectories and the ground-truth trajectory position at final time-step of prediction [29].

These metrics do not describe the distribution of the prediction. [4] introduced the use of recall and precision in trajectory prediction.

- Precision is the measure of how much of the predicted distribution is covered by the ground-truth distribution. A precision of one would mean that all the predicted trajectories are also a part of the ground truth distribution, and no predictions lie outside the ground truth.
- Recall measures how much of the can be ground-truth is covered by the predicted distribution. A recall of one means the predictions completely cover the ground truth.

The precision and recall, as described here, try to evaluate the entire distribution rather than just figuring out how good the best trajectory is. This is important for motion planning scenarios where the entire distribution of the predicted trajectories helps define the obstacle space. A shortcoming of the metrics used in [4] is that they only work if the ground truth consists of a multimodal output.

B. Scenario Approach

The scenario approach [8] is an optimization method that uses data from historical observations as the constraints of the optimization problem. The number of historical samples used for the output affects the problem's risk of constraint violation with the cost of the solution. More predicted samples lead to a higher number of constraints, which reduces the risk of the solution but degrades the performance. The risk vs feasibility is explored in detail in [30] and [31], where the amount of data required to satisfy a particular risk is calculated for various problems. This method has also been used in other fields in time-series problems to predict future trends, such as power generation [32], [33], medical applications [34], and finance [35]. Similarly, trajectory prediction defined in this thesis is also a time-series problem, so the scenario approach is used to sample pedestrian motion from a partitioned database to predict future trajectories.

C. Contribution

- A new method for multi-modal trajectory predictions, the Partitioned Experience Method (PEM), based on the scenario approach, is introduced and implemented to predict pedestrian motion. A dataset containing pedestrian motion is divided into different partitions separating different kinds of motion. Based on the observed pedestrian motion, a suitable partition is chosen, and trajectories are sampled from it.
- This effectiveness of the trajectory prediction is analysed using the scenario-based motion planner.
- New evaluation metrics, called recall and variance for evaluating trajectory prediction algorithms, which work on uni-modal ground truth data, are defined to measure how well the predictions capture the multi-modal distribution.

III. Problem Formulation

Consider an environment where an autonomous robot and obstacles are present. For motion planning, the autonomous robot needs to learn the current and future states of the environment accurately. The state of the environment at time t, is given by $e_t \in E$. The environment state at a given time is the combination of each obstacle present.

The problem of trajectory prediction can be defined as finding the environment state for N future time steps, given the environment states at current and m past time steps.

$$\begin{aligned} \text{Input} &: e = \{e_t, e_{t-1}, e_{t-2}, ..., e_{t-m}\} \quad \text{(History)} \\ \text{Output} &: \hat{e} = \{e_{t+1}, e_{t+2}, e_{t+3}, ..., e_{t+N}\} \quad \text{(Future)} \end{aligned} \tag{1}$$

Thus the prediction of future states of the environment can be represented as the combination of the trajectory prediction of each obstacle. In order to simplify the problem in Eq. (1), we will consider the problem to be the trajectory prediction of one obstacle, Obstacle i:

$$\begin{aligned} e_t &= \prod_{i=1}^{v} \mathbf{x}_{\text{obs}}^{i,t} \\ Input: &\quad \mathbf{x}_{\text{obs}}^{i} = \{\mathbf{x}_{\text{obs}}^{t}, \mathbf{x}_{\text{obs}}^{t-1}, ..., \mathbf{x}_{\text{obs}}^{t-m}\} \\ Output: &\quad \hat{\mathbf{x}}_{\text{obs}}^{i} = \{\mathbf{x}_{\text{obs}}^{t+1}, \mathbf{x}_{\text{obs}}^{t+2}, ..., \mathbf{x}_{\text{obs}}^{t+N}\} \end{aligned} \tag{2}$$

The obstacle state is not fully observable. Some information about the obstacle can be obtained (e.g. velocity, position, past motion). These are called observables. Other information on the obstacle state cannot be obtained (e.g. obstacle intention). These are the hidden states of the obstacle.

Let the obstacle state, $\mathbf{x}_{\text{obs}}^{i}$ be defined by a set of observables: $o_t^i \in \mathcal{O}$ and hidden states: $h_t^i \in \mathcal{H}$. The obstacle state is defined by: $\mathbf{x}_{\text{obs}}^{i} = O^i \cup \mathcal{H}^i$.

The predicted trajectories are a function of the current and past obstacle states. We assume that the hidden

states are generated by a time-variant distribution ($P^t_{\mathcal{H}}$). Data-centric approaches usually attempt to fit a model probability distribution for ($P^t_{\mathcal{H}}$) and thus make some assumptions regarding the distribution. Hence, the risk associated with the predictions can only be found empirically, and the real risk is unknown.

The PEM uses the scenario approach [36] and directly uses the samples from the distribution as the output predictions. In this case, no assumptions on the distribution are made, and thus the risk associated with the predicted trajectories can be accurately determined based on the number of trajectories predicted.

IV. Partitioned Experience Method

The PEM is an approach for trajectory prediction based on the scenario approach. It consists of two stages:

- Offline Stage: In this stage, we create a database called the experience dataset. The experience dataset is a collection of observables and the corresponding trajectory followed by obstacles. A partitioning algorithm divides the database into multiple partitions based on the collected observables, such that each partition contains trajectories of obstacles having a similar set of observables. The schematic of the offline stage is shown in Fig. 2.
- Online Stage: This stage is performed when the autonomous vehicle is driving. For every detected obstacle, the PEM predicts a multi-modal trajectory consisting of S_r trajectories. For this task, a set of observables is detected from the obstacle and fed to the partitioning algorithm, where the obstacle's partition is detected. From this partition, S_r trajectories (trajectories with the closest speed to that of the obstacles) are sampled. These trajectories are the predicted trajectories for the given obstacle. The schematic of the online stage is shown in Fig. 3.

Consider a dataset $\mathcal{D}$ of recorded obstacle motion. The experience dataset consists of a set of observables and their future trajectories. The hidden states cannot be recorded, and their distribution is defined by $P^t_{\mathcal{H}}$.

In the case of trajectory prediction, $P^t_{\mathcal{H}}$ is time-variant, i.e., the distribution of the hidden information is not static. This means that the trajectories collected in one instant of time cannot be readily used in another. To resolve this, we assume that the hidden states are correlated with the observables. That is, the intention of the obstacle is correlated with its observable state, such as velocity, direction, location etc. Then we can condition the hidden state based on the observables recorded, $P^t_{\mathcal{H}} = f(\mathcal{O})$, which is then assumed to be static with respect to time.

The idea of the method is that the more observable information is gained, the more is the variance of the distribution of the hidden state reduced. The more information we get about an obstacle state, the better we can predict the obstacle's intention.

A. Offline Stage

1) Data Collection: A recorded database, called the experience dataset, is collected at this stage. The experience dataset consists of observables and the corresponding trajectory for every observed obstacle. To make the experience dataset spatially stable, the collected trajectories do not consist of the actual obstacle position but rather contain the obstacle velocities at each step of the horizon. Let the set of m observables that represent an obstacle i be defined as, $\mathcal{O}^i_t = [\, O^{i,0}_t, O^{i,1}_t, O^{i,2}_t, ..., O^{i,m}_t \,]$.

The corresponding trajectory of this obstacle is $\hat{\mathbf{x}}^i_{\text{obs}} \in \mathbb{R}^{2N}$.

Let the experience dataset contain V datapoints recorded from various obstacles. Each data point is a combination of the observables recorded for a particular obstacle i and the corresponding trajectory.

$$\begin{aligned} &\text{Individual datapoint, } D_a \in \mathcal{O}^i_t \times \hat{\mathbf{x}}^i_{\text{obs}} \\ &\text{Experienced Dataset, } \mathcal{D} = [D_0, D_1, D_2, ..., D_V] \end{aligned} \tag{3}$$

2) Partitioning: By conditioning the hidden state on observables, the observables can directly be used to sample the relevant trajectories for a detected obstacle from the dataset, such that $\hat{\mathcal{O}}^i_t = \mathcal{O}_{\mathcal{D}}$. Here $\hat{\mathcal{O}}^i_t$ represents the set of observables of a detected obstacle, and $\mathcal{O}_{\mathcal{D}}$ represents an equivalent set of observables recorded in the experience dataset. But the probability of finding a point in a continuous space is zero ($P[X = x] = 0$).

Thus by definition, finding such a datapoint in the experience dataset where, $\hat{\mathcal{O}}^i_t = \mathcal{O}_{\mathcal{D}}$, is 0. To locate similar datapoints based on observables, it is necessary to divide the experience dataset into multiple partitions, where each partition contains volumes of datapoints with similar observables. These partitions are mutually exclusive, and each partition contains observables and the corresponding trajectory for specific cases of motion. Let experience dataset $\mathcal{D}$ be the collection of all individual partitions. Each partition is given by $\mathcal{D}^m$.

$$\mathcal{D} = \bigcup_{m=0}^{p} \mathcal{D}^m, \quad \mathcal{D}^i \cap \mathcal{D}^j = \emptyset, i \neq j, \quad \mathcal{D}\left[\hat{\mathcal{O}}^i_t \in \mathcal{D}\right] > 0, \quad \mathcal{D}^m \subseteq \mathcal{D} \tag{4}$$

The partition that an obstacle belongs to is a function of its observables. So, the partition m. to which an obstacle i, with a recorded set of observables $\mathcal{O}^i_t$, is given by:

$$\mathcal{D}^m = g(\mathcal{O}^i_t) \tag{5}$$

We propose partitioning the experience dataset using an unsupervised clustering algorithm, the k-means [37] algorithm. K-Means is an iterative clustering algorithm that divides the dataset into several (pre-defined number) partitions. This partitioning is done based on a set of features. K-Means randomly initializes centroids. Each datapoint is then assigned a cluster (partition) based on the nearest centroid, and the cluster is updated. A

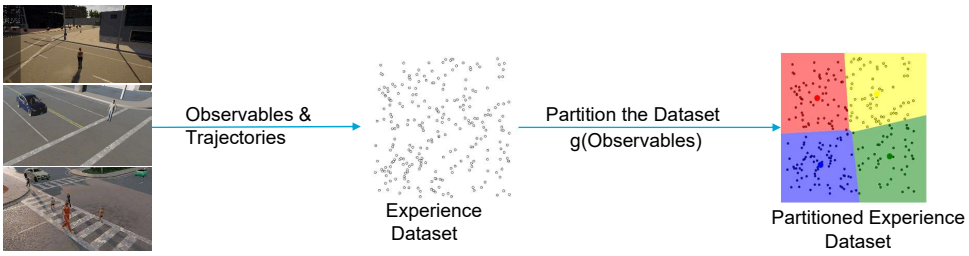


Fig. 2: Offline Stage

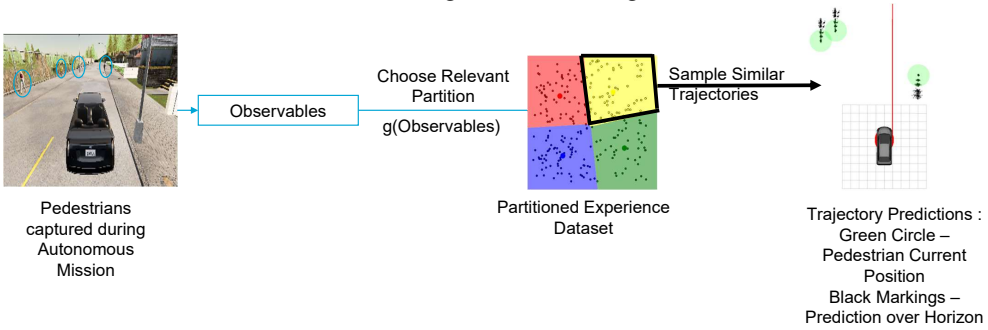


Fig. 3: Online Stage

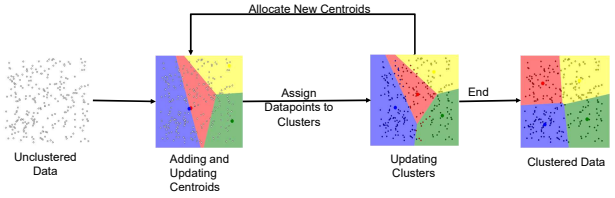


Fig. 4: K-Means Clustering Schematic [38]

new centroid is then assigned based on the mean of all the datapoints in the updated cluster. This process is iteratively performed till the assigned clusters are stable. Fig. 4 shows the schematic of K-Means for a two-dimensional set of features. The features used in PEM are high dimensional and cannot be visualized as easily.

K-Means is also an exclusive clustering method, meaning every datapoint can only belong to one cluster/partition. This point is crucial to our method, where by definition in Eq. (4), each datapoint can belong to one partition, and the partitions are mutually exclusive.

For the PEM, the features considered for the K-Means algorithm are the different observables recorded from obstacles. The more information the observables give on the obstacle state and thus condition the hidden state more accurately. This means that if more information about the obstacle state is known, the K-Means algorithm could better distinguish between the different intentions of the obstacle. Thus, each motion partition would contain only trajectories relating to a specific intention. The experience dataset is partitioned, so each partition contains at least S_r elements.

B. Online Stage

This stage occurs during motion planning, where the state of the environment needs to be predicted. Let us consider the problem defined in Eq. (2), trajectory prediction of one obstacle. A set of $\mathcal{O}_t^i$ is observables recorded for the obstacle at time t. The partition to which the obstacle belongs is given by Eq. (5). From this partition, S_r trajectories are sampled, and these samples are the predicted output of the problem. The output of the trajectory prediction is:

$$\hat{\mathbf{x}}_{\text{obs}}^{i} = [\hat{\mathbf{x}}_{\text{obs}}^{i,0}, \hat{\mathbf{x}}_{\text{obs}}^{i,1}, \hat{\mathbf{x}}_{\text{obs}}^{i,2}, ..., \hat{\mathbf{x}}_{\text{obs}}^{i,S_r}] \tag{6}$$

This process can be repeated for all obstacles in the environment, and the predicted future state of the environment is given by:

$$\hat{e} = \prod_{i=1}^{v} \hat{\mathbf{x}}_{\text{obs}}^{i,t} \tag{7}$$

V. Discussion

When an obstacle is detected, the Partitioned Experience Method (PEM) captures all the relevant information (Observables). It uses this information to predict the partition the obstacle belongs to, based on the K-Means clustering algorithm. A pre-determined (based on requirements) number of trajectories are then sampled from this partition and used as the output to the trajectory prediction problem.

In doing so, PEM alleviates three of the issues mentioned in Section I:

- Since the database is directly used to sample trajectories, one does not need to fit a model to learn

the prediction, and only realistic trajectories are sampled.
- Every trajectory that is sampled is unique and independent. Each trajectory represents an individual mode. Thus the trajectory prediction problem is not limited to a few modes, and low probability modes can be predicted.
- The output trajectories need not be sequentially sampled, the latency of predicting a high number of trajectories is lower than a comparative generative neural network based model.

VI. Implementation

The method is implemented and evaluated in the CARLA [39] simulator (shown in Fig. 5) along with a motion planning algorithm, the scenario-based motion planner. The scenario-based motion planner is a chance-constrained optimization problem where the constraints are the possible trajectories of various obstacles in the planning horizon. The number of trajectories needed to be predicted for each obstacle at every time-step is a function of the required risk (ϵ) and confidence (β) of the motion planner, $S_r \propto (\beta, \epsilon)$. This is calculated in [7], and the number of trajectories predicted for a risk factor of { 5, 10, 15 } are approximately { 1000, 500, 250 }. The table I gives further details on the implementation of the trajectory prediction.

TABLE I: Implementation details of the experiment

Type of Obstacles Considered	Pedestrians
Number of Predicted Timesteps	20
Timestep size	0.2 seconds

Fig. 5: Carla Scene

A. Implemented Algorithm

1) Inputs/ Observables: The implementation in the thesis is a map-aware approach, where the location of the sidewalks and crosswalks are known.
Carla provides the pedestrian's position and velocity with respect to the global centre of the map, so local motion is not known. To make the data spatially stable, the motion of the pedestrian must be recorded in a frame of reference that is easily generalized and can be used for any pedestrian.
The road network is converted into a graph with multiple nodes at the sidewalks' edge. Each consecutive node is joined together to form edges. All the observables of the pedestrian are calculated with respect to the close edge. The used observables:

- Distance from edge: Distance from the sidewalk edge, i.e. how far away pedestrian is from the road.
- Angle with respect to edge: Angle of the pedestrian velocity to the nearest sidwalk edge.
- Velocity with respect to edge: Velocity of the pedestrian with respect to the nearest sidwalk edge.
- Distance to crosswalk: Distance to the closest crosswalk from the pedestrian.
- Acceleration: Captures history, change in pedestrian velocity over past few timesteps.
- Direction Change : Captures history, change in pedestrian angle with the sidewalk edge over time.

These observations are recorded and sent to the partitioning module.

2) Experience dataset: The experience dataset consists of recorded pedestrian motion in Carla by recording the observables (defined above) for various pedestrians and the corresponding trajectories of obstacle velocities for 20 time-steps in the future. This dataset is recorded by collecting pedestrian information from multiple roads. In order to use the pedestrian trajectory in one road to satisfy the prediction of another road, all the motions are generalised and calculated in a local frame of reference as explained in the above Section VI-A1.

3) Partitioning: In this step, the k-means implementation of scikit learns [40] is used to divide the experience dataset into multiple partitions such that each partition has at least 1000 datapoints, ensuring that each partition can guarantee probabilistic safety.

4) Outputs: The output consists of S_r sampled trajectories, with each trajectory containing the velocity of the pedestrians for the next 20 time-steps. These velocities are integrated with respect to the observed current position of the obstacles to get the possible positions. So for each future time-step, the output contains 1000 possible positions of the obstacle. These outputs are used to construct constraints for the scenario-based motion planner.

VII. Experiments

For evaluation, two types of simulated experiments were carried out.

A. Stand-alone Trajectory Prediction Experiments

Here, the evaluation of the stand-alone trajectory prediction network is performed.

1) Evaluation Metrics: The recall and precision introduced in [4] is based on the convergence of ground-truth modes and the predicted modes, and the ground-truth itself is multi-modal. So it would not work for cases with just one ground truth per test case. Most datasets contain only one possible ground truth per test case, so such a method cannot be used. For the metrics to work when single ground truth is used, in this paper, the concept of recall is modified, and a new metric called variance is introduced to describe the shape of the predicted distribution. The predicted trajectory is clustered by a distance-based clustering algorithm, agglomerative clustering [41], from scikit learn [40]. This clustering attempts to separate the different prediction modes by keeping the nearby predicted trajectories close. The outer edges of each cluster are then found by fitting a bounding box on each cluster, using the convex hull algorithm of scipy [42]. The recall and variance are then calculated as follows:

- Recall - The recall defines if the predicted trajectories capture the ground truth for every time step of prediction. For example, in Fig. 6, the ground truth at the final stage of prediction is present inside the bounding box, and hence it is accurately captured by the predicted distribution. The recall of the algorithm of one time step is the percentage of predictions in which the distribution accurately represent the ground truth at that time step.
- Variance - Variance captures the size of every stage of the predicted distribution. In Fig. 6, this size for the final stage of prediction is given by the sum of areas of the bounding boxes A, B and C.
  The variance of an algorithm of one time step is the average of the areas of the predicted distribution of every prediction.

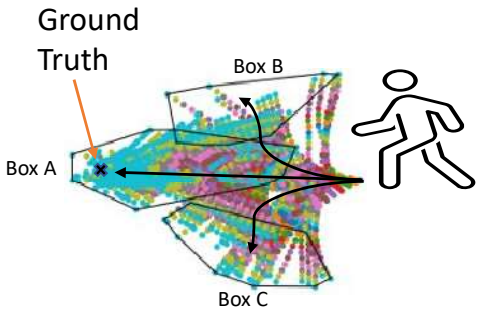


Fig. 6: Multi-Modal Trajectory Prediction of pedestrian, where each colour represents one time-step and the black 'x' represents the ground truth of the final position

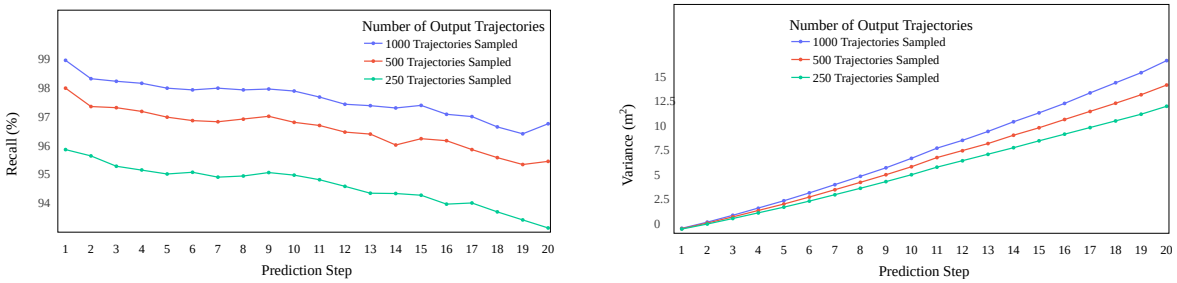
For calculating the recall and variance for the various experiments performed, a test set containing 10000 pedestrian observables and trajectories is separated from the recorded experience dataset and used.

2) Experiments Performed:

- Number of output trajectories sampled - The effect of different amounts of sampled trajectories is studied, and the results are demonstrated in graphs of Fig. 7a and Fig. 7b. The results show that the number of sampled trajectories greatly affects both the recall and variance. This goes per the research in scenario approach [36], as more trajectories sampled mean more variance in the predicted distribution, increasing the size of the predicted distribution and thereby increasing the probability of accurately predicting the ground truth.

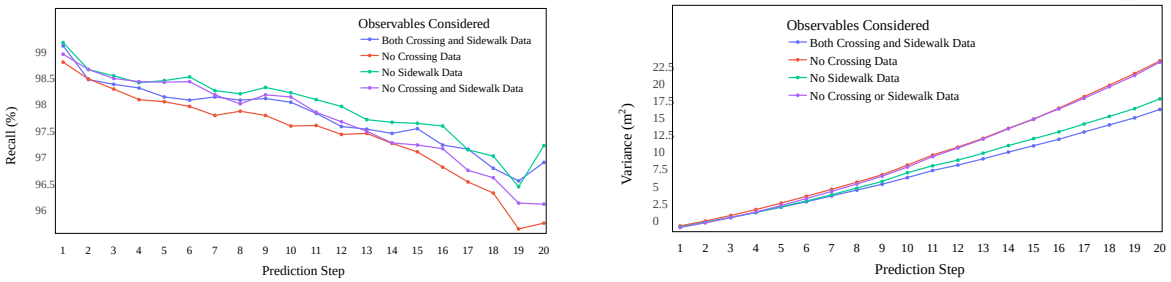
- Choice of Observables used - The effect of the choice of observables is studied, and the results are shown in graphs in Fig. 8a and Fig. 8b. The choice of the observables used greatly affects how partitioning is done and, thus, the kind of trajectories present in a partition. The more well-defined the obstacle's current state is, the less variance there is in the obstacle's predicted trajectories. For example, including a pedestrian's crosswalk data would help predict crossing intention, which will be reflected in the created partitions. If the pedestrian is near a crossing zone, the crossing intention would be higher, and the partition chosen would also reflect this. If crossing data is not included, this intention is not captured, and the chance of pedestrian crossing would be similar in each partition. The graphs in Fig. 8a and Fig. 8b show that the variance of predicted trajectories increases when the sidewalk or crosswalk data of the pedestrian is removed. This means that this data helps predict the pedestrian's intention and reduces the variance in the predicted trajectories.

- Size of the experienced dataset - Three datasets containing 100000 trajectories, 200000 trajectories and 800000 trajectories are considered and partitioned such that each partition contains at least 1000 trajectories. The results are shown in the graphs of Fig. 9a and Fig. 9b. The accuracy of the underlying distribution of motion is dependent on the size of the dataset. The distribution of a dataset with more samples is well-defined compared to a dataset with a less number of samples. This is demonstrated in the experiments in Fig. 9a and Fig. 9b. When there are fewer datapoints in the experience dataset, the number of trajectories representing each type of motion is also less. Due to this, there might not be S_r trajectories to represent the different intentions of the pedestrian, and hence adequate partitions cannot be made to cap-



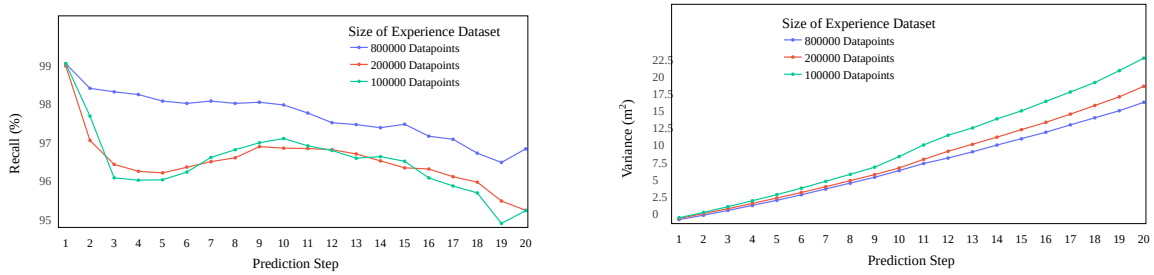
(a) Recall (b) Variance

Fig. 7: Recall and Variance for Different Number of Sampled Trajectories



(a) Recall (b) Variance

Fig. 8: Recall and Variance when different Observables are Used



(a) Recall (b) Variance

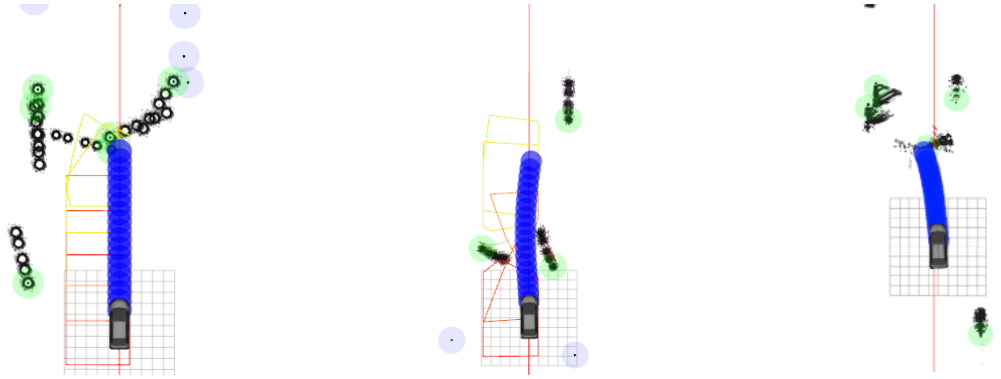
Fig. 9: Recall and Variance for Different Sizes of the Experience Dataset

ture each intention separately. Thus, each partition will contain a mixture of different intentions, and the variance of the predicted distribution will be larger. As the number of datapoints increases, the dataset can be partitioned such that more intentions of the pedestrian can be partitioned, containing S_r elements. Intentions can be better represented, and the variance of the predicted trajectory would be less. This test shows that the PEM does not eliminate the need for having an extensive database, and the more data contained in the experience dataset, the more well-defined the motion is.

B. Motion Planning Experiments

Here the PEM's performance is tested along with a motion planner. The performance is compared with two other trajectory prediction algorithms, the constant velocity trajectory predictions, where the pedestrian is projected to move at the observed constant velocity throughout the time horizon. The Social-VRNN [43] is a neural network based method that captures both social impact (Positions of other pedestrians and vehicles in the scene) and the position of static obstacles in the scene to generate possible trajectories. Twenty runs of the motion planner with the implementation in table I are run, with pedestrians spawning at different map locations in a different run. Then, the following metrics are collected and averaged out.

- Trajectory Followed - The trajectory of the car followed under different trajectory prediction algorithms is shown in Fig. 11.
- Close Obstacle Count - The motion planner is run



(a) Constant Velocity model (b) Social-VRNN model (c) PEM

Fig. 10: Rviz visualization of the motion planning problem, where the blue circles represents the generated plan for the car. The green markings represent the pedestrians and the black markings represent the pedestrian's predicted future trajectory

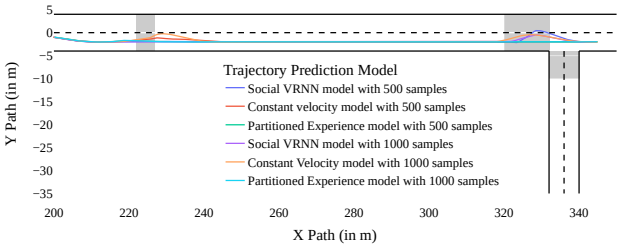


Fig. 11: Trajectories followed by various algorithms. The ego-vehicle travels from (200,-2) to (350,-1) in the mission. The shaded areas in the graph represent the different crossing zones, while the dashed lines represent the lane dividers

20 times, collecting metrics such as collisions, close encounters (where the distance to the obstacle is less than 0.5m) and minimum clearance per run. The results are shown in table II.

The results of the above two experiments demonstrate the ability of the PEM to learn crossing intentions before the pedestrian turns or enters the road. In the trajectories graph, it can be seen that both the constant velocity and the social-vrnn models (500 and 1000 trajectories sampled) have huge deviations in trajectories (Fig. 11), sometimes even entering the other lane. These deviations happen in the crossing zones, where the constant velocity and the social-vrnn models do not predict pedestrian crossing in time. The motion planner has to resort to reactive measures, such as sudden deviations, to avoid collisions. This is also reflected in the close encounters and collisions, as the ego-vehicle does not pre-emptively slow down for pedestrian crossing. These results are shown in in table II.

The PEM can learn crossing intentions by collecting appropriate data and simple partitioning approaches. Due to the crossing prediction, the ego-vehicle can slow down in time for pedestrians to cross, avoiding collisions and having very few close encounters.

TABLE II: Collision and close encounters

Algorithm	Collision Count	Close Encounters ($< 0.5m$)	Minimum clearance per run [m]
Constant Velocity (500 samples)	4	33	0.24
PEM(500 samples)	**0**	**9**	**0.63**
Social - VRNN (500 samples)	1	24	**0.27**
Constant Velocity (1000 samples)	2	25	**0.17**
PEM (1000 samples)	**0**	**5**	**0.74**
Social - VRNN (1000 samples)	2	18	0.41

- Time Taken to Goal - This is a measure of the average time to goal for each of the three algorithms is shown in table III.

  The early detection of crossing makes the ego-vehicle slow down in the crossing zones, unlike the social-vrnn and constant velocity prediction models where the ego-vehicle does not slow down until the pedestrian is on the road and crossing and thus uses reactive measures to avoid pedestrians and continue the mission. This results in the PEM being more conservative. The time taken to reach the goal is highest in the PEM compared to the other two methods (table III), but the safety of the planner improves significantly.

TABLE III: Average Time Taken to Goal

Algorithm	Average Time to Goal (seconds)
Constant Velocity (500 samples)	**46.1**
PEM (500 samples)	56.6
Social - VRNN (500 samples)	47.4
Constant Velocity (1000 samples)	**45.9**
PEM (1000 samples)	61.8
Social - VRNN (1000 samples)	48.2

TABLE IV: Time Taken to sample

Algorithm	Time Taken to sample trajectories (in ms)
Constant Velocity (500 samples)	<1
PEM (500 samples)	~25
Social - VRNN (500 samples)	~89
Constant Velocity (1000 samples)	<1
PEM (1000 samples)	~26.5
Social - VRNN (1000 samples)	~112.5

C. Time taken for Prediction

The table IV contains each algorithm's time to sample a specific number of trajectories for six obstacles. As can be seen, the constant velocity model is the fastest algorithm because it directly integrates current velocity over the horizon, without considering external factors. On the other hand, the Social-VRNN is a neural network-based algorithm that considers all the pedestrians in the scene to predict a particular pedestrian's position. It also requires sequential sampling trajectories to generate a diverse, multi-modal prediction, which leads to a high prediction time for a large number of sampled trajectories.
The PEM does not sequentially generate individual trajectories but directly samples trajectories from the experienced dataset, which is why the time difference between sampling 500 and 100 trajectories is low (1.5ms difference). The PEM can thus sample many trajectories required for the constraint satisfaction of multi-modal methods in real time.

VIII. Conclusion and Future Work

In this thesis, the current state-of-the-art trajectory prediction algorithms are studied. The underlying issues regarding learning the pedestrians' underlying distribution, amount of training data needed and algorithms' speed are raised. A new method for trajectory prediction of other obstacles, the PEM, is introduced to address these issues. The PEM does not try to learn a distribution but instead uses the recorded database and partitions it based on different kinds of motion. Based on the query obstacle, trajectories in the database, having a similar kind of motion as the query obstacles, are directly sampled as the output. The algorithm removes the need to learn the underlying distribution of the dataset and is time efficient for sampling a large number of trajectories. However, the PEM depends on the recorded experiences. Hence, a large amount of data is required to represent the various pedestrian motions accurately. The implemented method focuses on pedestrian motion and uses knowledge of pedestrian motion and map information such as the location of the footpaths and crossing zone to learn different types of pedestrian motion. This is evaluated in a CARLA simulation environment with a sampling-based motion planner, considering pedestrians as the other obstacles.

A. Future Work

- Currently, the PEM implemented in the thesis is also evaluated with a motion planner, so the CARLA simulation tool is used for the implementation. The next step would be to adapt the implementation to real-life datasets like SDD [44], ETH/UCY ([45], [46]), KITTI [47], NuScenes [48], which are the usually used datasets for the evaluation of trajectory prediction methods.
- Current implementation uses simple rules and a clustering-based partition method to predict different types of pedestrian motion. Future work could use neural networks' power to capture pedestrian motion and the environment scene. This could be used along with clustering to capture more intricate details for partitioning compared to the currently used approach.

References

[1] S. Arora, A. Risteski, and Y. Zhang, "Do GANs learn the distribution? Some Theory and Empirics," 2 2022.

[2] S. Arora and Y. Zhang, "Do GANs actually learn the distribution? An empirical study," 6 2017. [Online]. Available: http://arxiv.org/abs/1706.08224

[3] J. Görtler, R. Kehlbeck, and O. Deussen, "A Visual Exploration of Gaussian Processes," *Distill*, vol. 4, no. 4, p. e17, 4 2019. [Online]. Available: https://distill.pub/2019/visual-exploration-gaussian-processes

[4] P. Dendorfer, S. Elflein, and L. Leal-Taixé, "MG-GAN: A Multi-Generator Model Preventing Out-of-Distribution Samples in Pedestrian Trajectory Prediction," 8 2021. [Online]. Available: https://arxiv.org/abs/2108.09274v1

[5] P. Dendorfer, A. Ošep, and L. Leal-Taixé, "Goal-GAN: Multimodal Trajectory Prediction Based on Goal Position Estimation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12623 LNCS, pp. 405–420, 10 2020. [Online]. Available: https://arxiv.org/abs/2010.01114v1

[6] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, S. Hamid Rezatofighi, and S. Savarese, "Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks," *Advances in Neural Information Processing Systems*, vol. 32, 7 2019. [Online]. Available: https://arxiv.org/abs/1907.03395v2

[7] O. de Groot, B. Brito, L. Ferranti, D. Gavrilå, and J. Alonso-Morå, "Scenario-Based Trajectory Optimization in Uncertain Dynamic Environments," p. 2021.

[8] "The scenario approach: A tool at the service of data-driven decision making - ScienceDirect." [Online]. Available: https://www-sciencedirect-com.tudelft.idm.oclc.org/science/article/pii/S1367578821000791

[9] N. Kaempchen, K. Weiss, M. Schaefer, and K. C. Dietmayer, "IMM object tracking for high dynamic driving maneuvers," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 825–830, 2004.

[10] P. Lytrivis, G. Thomaidis, and A. Amditis, "Cooperative path prediction in vehicular environments," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 803–808, 2008.

[11] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang, "Vehicle Trajectory Prediction by Integrating Physics- and Maneuver-Based Approaches Using Interactive Multiple Models," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5999–6008, 7 2018.

[12] L. Jain and L. Medsker, "Recurrent Neural Networks: Design and Applications," *undefined*, 12 1999. [Online]. Available: https://www.taylorfrancis.com/books/9781420049176

[13] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997.

[14] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces."

[15] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, "Context-Aware Trajectory Prediction," *Proceedings - International Conference on Pattern Recognition*, vol. 2018-August, pp. 1941–1946, 5 2017. [Online]. Available: https://arxiv.org/abs/1705.02503v1

[16] N. Bisagno, B. Zhang, and N. Conci, "Group LSTM: Group Trajectory Prediction in Crowded Scenarios," pp. 0–0, 2018.

[17] B. Ivanovic and M. Pavone, "Injecting Planning-Awareness into Prediction and Detection Evaluation." [Online]. Available: https://github.com/BorisIvanovic/

[18] F. Fang, P. Zhang, B. Zhou, K. Qian, and Y. Gan, "Atten-GAN: Pedestrian Trajectory Prediction with GAN Based on Attention Mechanism," *Cognitive Computation 2022*, vol. 1, pp. 1–10, 6 2022. [Online]. Available: https://link.springer.com/article/10.1007/s12559-022-10029-z

[19] H. Cui, V. Radosavljevic, F. C. Chou, T. H. Lin, T. Nguyen, T. K. Huang, J. Schneider, and N. Djuric, "Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 2090–2096, 9 2018. [Online]. Available: https://arxiv.org/abs/1809.10732v2

[20] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, E. M. Wolff, and A. Company, "CoverNet: Multimodal Behavior Prediction using Trajectory Sets."

[21] K. Sohn, H. Lee, and X. Yan, "Learning Structured Output Representation using Deep Conditional Generative Models," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[22] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 12 2013. [Online]. Available: https://arxiv.org/abs/1312.6114v10

[23] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 2165–2174, 4 2017. [Online]. Available: https://arxiv.org/abs/1704.04394v1

[24] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-Feasible Trajectory Forecasting with Heterogeneous Data," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12363 LNCS, pp. 683–700, 8 2020. [Online]. Available: https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-3-030-58523-5_40

[25] J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio, "A Recurrent Latent Variable Model for Sequential Data," *Advances in Neural Information Processing Systems*, vol. 2015-January, pp. 2980–2988, 6 2015. [Online]. Available: https://arxiv.org/abs/1506.02216v6

[26] B. Brito, H. Zhu, W. Pan, and J. Alonso-Mora, "Social-VRNN: One-Shot Multi-modal Trajectory Prediction for Interacting Pedestrians," 10 2020. [Online]. Available: https://arxiv.org/abs/2010.09056v2

[27] I. Tolstikhin, O. Bousquet, B. Schölkopf, K. Thierbach, P. L. Bazin, W. de Back, F. Gavriilidis, E. Kirilina, C. Jäger, M. Morawski, S. Geyer, N. Weiskopf, N. Scherf, I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, E. Musk, Neuralink, M. A. Hjortsø, P. Wolenski, S. Ruder, W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, D. Duvenaud, and C. Doersch, "Generative Adversarial Networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11046 LNCS, no. NeurIPS, pp. 1–9, 6 2014. [Online]. Available: https://arxiv.org/abs/1406.2661v1

[28] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2255–2264, 3 2018. [Online]. Available: https://arxiv.org/abs/1803.10892v1

[29] R. Chandra, U. Bhattacharya, C. Roncal, A. Bera, and D. Manocha, "RobustTP: End-to-End Trajectory Prediction for Heterogeneous Road-Agents in Dense Traffic with Noisy Sensor Inputs," *Proceedings - CSCS 2019: ACM Computer Science in Cars Symposium*, 7 2019. [Online]. Available: https://arxiv.org/abs/1907.08752v1

[30] M. C. Campi and S. Garatti, "THE EXACT FEASIBILITY OF RANDOMIZED SOLUTIONS OF UNCERTAIN CONVEX PROGRAMS *," vol. 19, no. 3, pp. 1211–1230. [Online]. Available: http://bsing.ing.unibs.it/

[31] M. C. Campi, S. Garatti, and F. A. Ramponi, "A general scenario theory for non-convex optimization and decision making," *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, vol. 63, no. 12, 2018. [Online]. Available: http://ieeexplore.ieee.org.

[32] Z. E. Lee, Q. Sun, Z. Ma, J. Wang, J. S. MacDonald, and K. Max Zhang, "Providing Grid Services With Heat Pumps: A Review," *ASME Journal of Engineering for Sustainable Buildings and Cities*, vol. 1, no. 1, 2 2020. [Online]. Available: https://asmedigitalcollection.asme.org/sustainablebuildings/article/1/1/011007/1072202/Providing-Grid-Services-With-Heat-Pumps-A-Review

[33] X. Geng and L. Xie, "Data-driven decision making in power systems with probabilistic guarantees: Theory and applications of chance-constrained optimization," *undefined*, vol. 47, pp. 341–363, 1 2019.

[34] A. Borri, F. Cacace, A. De Gaetano, A. Germani, C. Manes, P. Palumbo, S. Panunzi, and P. Pepe, "Luenberger-Like Observers for Nonlinear Time-Delay Systems with Application to the Artificial Pancreas: The Attainment of Good Performance," *IEEE Control Systems*, vol. 37, no. 4, pp. 33–49, 8 2017.

[35] G. Arici, M. C. Campi, A. Carè, M. Dalai, and F. A. Ramponi, "A Theory of the Risk for Empirical CVaR with Application to Portfolio Selection," *Journal of Systems Science and Complexity 2021 34:5*, vol. 34, no. 5, pp. 1879–1894, 10 2021. [Online]. Available: https://link.springer.com/article/10.1007/s11424-021-1229-3

[36] M. C. Campi, S. Garatti, and M. Prandini, "Scenario Optimization for MPC." [Online]. Available: https://doi.org/10.1007/978-3-319-77489-3

[37] J. MacQueen, *Classification and analysis of multivariate observations*, 1967.

[38] "Visualizing K-Means Clustering." [Online]. Available: https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

[39] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," pp. 1–16, 10 2017. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html

[40] F. Pedregosa FABIANPEDREGOSA, V. Michel, O. Grisel OLIVIERGRISEL, M. Blondel, P. Prettenhofer, R. Weiss, J. Vanderplas, D. Cournapeau, F. Pedregosa, G. Varoquaux, A. Gramfort, B. Thirion, O. Grisel, V. Dubourg, A. Passos, M. Brucher, M. Perrot andÉdouardand, a. Duchesnay, and F. Duchesnay EDOUARDDUCHESNAY, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[41] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," 9 2011. [Online]. Available: https://arxiv.org/abs/1109.2378v1

[42] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, A. Vijaykumar, A. P. Bardelli, A. Rothberg, A. Hilboll, A. Kloeckner, A. Scopatz, A. Lee, A. Rokem, C. N. Woods, C. Fulton, C. Masson, C. Häggström, C. Fitzgerald, D. A. Nicholson, D. R. Hagen, D. V. Pasechnik, E. Olivetti, E. Martin,

E. Wieser, F. Silva, F. Lenders, F. Wilhelm, G. Young, G. A. Price, G. L. Ingold, G. E. Allen, G. R. Lee, H. Audren, I. Probst, J. P. Dietrich, J. Silterra, J. T. Webber, J. Slavič, J. Nothman, J. Buchner, J. Kulick, J. L. Schönberger, J. V. de Miranda Cardoso, J. Reimer, J. Harrington, J. L. C. Rodríguez, J. Nunez-Iglesias, J. Kuczynski, K. Tritz, M. Thoma, M. Newville, M. Kümmerer, M. Bolingbroke, M. Tartre, M. Pak, N. J. Smith, N. Nowaczyk, N. Shebanov, O. Pavlyk, P. A. Brodtkorb, P. Lee, R. T. McGibbon, R. Feldbauer, S. Lewis, S. Tygier, S. Sievert, S. Vigna, S. Peterson, S. More, T. Pudlik, T. Oshima, T. J. Pingel, T. P. Robitaille, T. Spura, T. R. Jones, T. Cera, T. Leslie, T. Zito, T. Krauss, U. Upadhyay, Y. O. Halchenko, and Y. Vázquez-Baeza, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods 2020 17:3*, vol. 17, no. 3, pp. 261–272, 2 2020. [Online]. Available: https://www-nature-com.tudelft.idm.oclc.org/articles/s41592-019-0686-2

[43] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 10 2020. [Online]. Available: https://arxiv.org/abs/2010.10190v1

[44] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9912 LNCS, pp. 549–565, 2016. [Online]. Available: https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-3-319-46484-8_33

[45] S. Pellegrini, A. Ess, and L. Van Gool, "Improving data association by joint modeling of pedestrian trajectories and groupings," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6311 LNCS, no. PART 1, pp. 452–465, 2010.

[46] A. Lerner, Y. L. Chrysanthou, D. Lischinski, A. Lerner, Y. L. Chrysanthou, and D. Lischinski, "Crowds by example," *Computer Graphics Forum*, vol. 26, no. 3, pp. 655–664, 2007. [Online]. Available: https://gnosis.library.ucy.ac.cy/handle/7/54385

[47] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset:," *http://dx.doi.org.tudelft.idm.oclc.org/10.1177/0278364913491297*, vol. 32, no. 11, pp. 1231–1237, 8 2013. [Online]. Available: https://journals-sagepub-com.tudelft.idm.oclc.org/doi/full/10.1177/0278364913491297

[48] H. Caesar, J. Kabzan, K. Seang, T. Whye, K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. O. Motional, "NuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles," 6 2021. [Online]. Available: https://arxiv.org/abs/2106.11810v4

[49] M. Gulzar, Y. Muhammad, and N. Muhammad, "A Survey on Motion Prediction of Pedestrians and Vehicles for Autonomous Driving," *IEEE Access*, vol. 9, pp. 137 957–137 969, 2021.

[50] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "A Data-driven Model for Interaction-aware Pedestrian Motion Prediction in Object Cluttered Environments."

[51] M. Lisotto, P. Coscia, and L. Ballan, "Social and Scene-Aware Trajectory Prediction in Crowded Spaces," 9 2019. [Online]. Available: http://arxiv.org/abs/1909.08840

[52] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, S. H. Rezatofighi, and S. Savarese, "SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints," pp. 1349–1358, 2019.

[53] Y. Yoon, T. Kim, H. Lee, and J. Park, "Road-Aware Trajectory Prediction for Autonomous Driving on Highways," *Sensors 2020, Vol. 20, Page 4703*, vol. 20, no. 17, p. 4703, 8 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/17/4703/htmhttps://www.mdpi.com/1424-8220/20/17/4703

[54] X. Fang, J. Huang, F. Wang, L. Zeng, H. Liang, and H. Wang, "ConSTGAT: Contextual Spatial-Temporal Graph Attention Network for Travel Time Estimation at Baidu Maps," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2697–2705, 8 2020.

[55] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces."

[56] S. Haddad and S. K. Lam, "Graph2Kernel Grid-LSTM: A Multi-Cued Model for Pedestrian Trajectory Prediction by Learning Adaptive Neighborhoods," 7 2020. [Online]. Available: https://arxiv.org/abs/2007.01915v2

[57] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 1349–1358, 6 2018. [Online]. Available: https://arxiv.org/abs/1806.01482v2

[58] S. Zamboni, Z. T. Kefato, S. Girdzijauskas, C. Norén, and L. Dal Col, "Pedestrian trajectory prediction with convolutional neural networks," *Pattern Recognition*, vol. 121, p. 108252, 1 2022.

Appendix

Preliminaries

A. Scenario Approach in Motion Planning

In autonomous driving, the constraints are uncertain since the positions of obstacles (static and dynamic) are not precisely known. To overcome this, these positions are usually described using uncertainty around their *believed* position. The scenario approach comes into the picture here. [7] builds upon a Model Predictive Contouring Control (MPCC) framework called Local MPCC [43]. The way constraints are dealt with is inspired by [31], i.e. the current and possible future positions of the obstacles are represented as scenarios that act as constraints. The algorithm is as follows:

- The chance constraints of the dynamic obstacle are linearized.
- These linearized chance constraints are sampled, and deterministic constraints called scenarios are drawn.
- The risk of planning is linked with the number of scenarios, with lesser scenarios drawn implying higher risk.

1) Problem Definition: Consider a robot at its current state, $x_t \in \mathbb{X}$ navigating through an environment at state e_t containing both static and dynamic obstacles. The task followed by the robot is to follow a trajectory; the problem after linearizing the chance constraints and sampling the deterministic scenarios is given by:

$$\min_{u \in U} \quad \sum_{k=1}^{N} J(x_k, u_k) \tag{8a}$$

$$s.t. \quad x_{k+1} = f(xk, uk), x \in \mathbb{X} \tag{8b}$$

$$A_k^T(\delta_k^i, \hat{x}_k) x_k \leq b_k(\delta_k^i, \hat{x}_k) \tag{8c}$$

$$x_0 = x_{init} \tag{8d}$$

where x_k and u_k denote the states and inputs of the robot and $\hat{x}_k$) is the k-step ahead prediction of the robot,

$$A_k = \frac{\delta_k - \hat{x}_k}{\|\delta_k - \hat{x}_k\|}, b_k = A_k(\delta_k - A_k r) \tag{9}$$

is the linearized collision region with respect to the predictions of the robot state at a given stage k, u is the control input given to the robot.
The set of deterministic constraints is obtained by sampling each dynamic obstacle's position from uncertainty (δ_k^v) at a time-step k and getting the collision constraint. Each sample is a possible position of the dynamic obstacle v at a time-step k. The future positions of each obstacle are not observable and time-variant; that is, samples at each time step cannot be used for other time-steps.

2) Risk Bounding: The paper also has some theorems on risk bounding and some methods for sample pruning so that less number of samples are used in the optimization and the solution is tractable.
Theorem 1 of the paper states that the probability that the solution violates the chance constraint at a stage k in optimization is:

$$\mathbb{P}_k^{S_k}\left[V_k\left(\boldsymbol{u}_{SP}^*\right) > \epsilon_k\left(s_k^*\right)\right] \leqslant \beta_k(S_k) \tag{10}$$

where β is the confidence parameter that is defined by:

$$\beta_k(S_k) := \sum_{0}^{S_k - 1} \binom{S_k}{s} [1 - \epsilon_k(s)]^{S_k - s} \tag{11}$$

Thus the scenario approach for autonomous driving is a stochastic approach to the collision avoidance problem in local motion planning. The approach, unlike LMPCC, is agnostic to the type of uncertainty of the dynamic obstacle. Another advantage of the approach is that the pruning of constraints, and the formal definition of the risk/ safety bounds, not only with the original scenario but also after pruning, allows the approach to be real-time.

B. Types of Trajectory Prediction Algorithms

This section describes the different ways in which trajectory prediction algorithms (as shown in Fig. 12) can be classified.

1) Based on Output: Trajectory prediction models can either be classifiers that try to classify the type of motion and object takes or predict one or multiple possible trajectories at every future time-step. The different types of possible outputs in a trajectory prediction model are described below:

- Uni-Modal: These algorithms predict one trajectory per obstacle.
- Multi-Modal: These algorithms suggest that one trajectory is not enough to cover all human intentions, hence predicting multiple possibilities of motion.

2) Based on Situational Awareness: Situational awareness refers to the data which the model takes as input. This would range from considering only one obstacle independently to considering multiple obstacles and their environment to give a more accurate prediction.

- Unaware: These models try to predict the trajectory of an obstacle without considering its interaction with other obstacles/the entire scene itself. Some basic physics-based approaches described below, such as the constant velocity (CV) and constant acceleration (CA), are unaware models. These models are too simple, and since they do not consider the presence of other obstacles and the environment, they do not produce accurate long-term trajectories.
- Interaction Aware: Interaction-aware prediction models consider other obstacles in the path while

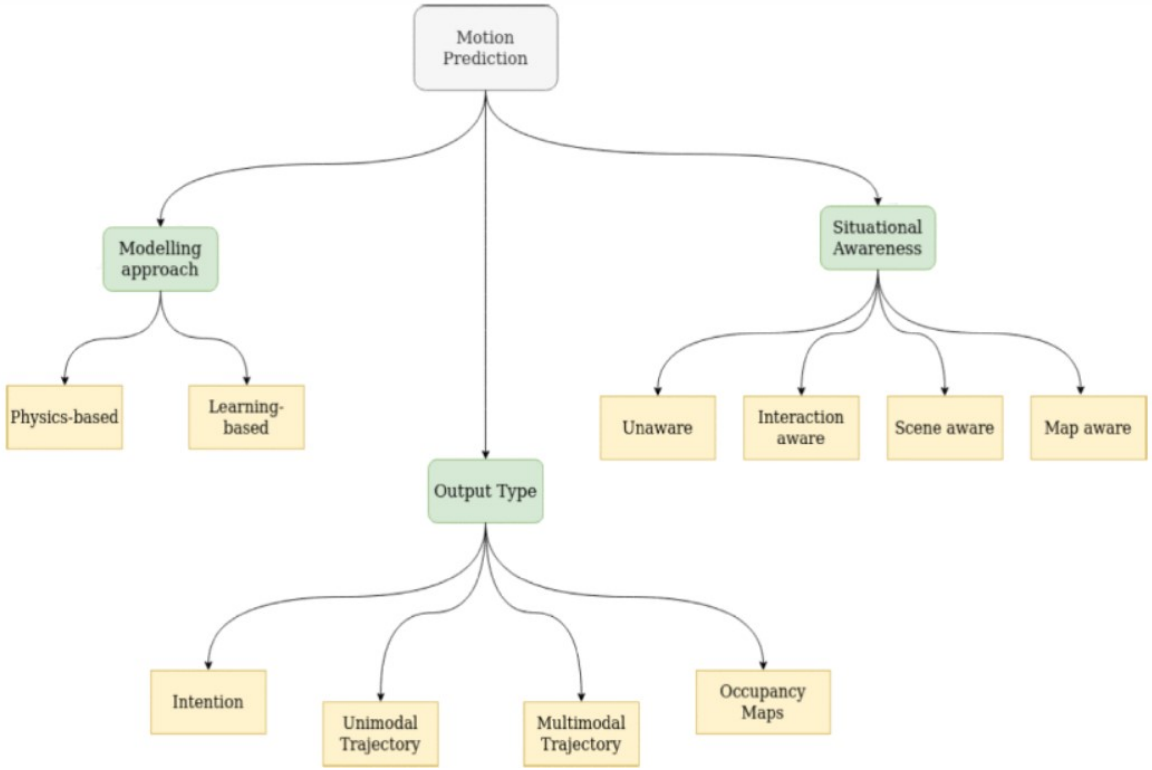


Fig. 12: Taxonomy of trajectory prediction [49]

estimating future trajectories. For a pedestrian, this would be using the trajectories of other pedestrians/vehicles and reasoning out that the pedestrian would try to avoid collisions and some possible paths. [14], [28] and [50] are some algorithms based on interaction aware modelling, where interaction between obstacles in a particular scene is considered by using LSTM.

- Scene Aware: These models use the interaction between all obstacles in the scene and the scene itself. For example, a pedestrian crossing the road would have a different type of movement when compared to walking straight ahead. These features are usually obtained from a sensor feed (like a camera) and fed to the model. Scene context adds physical constraints, which help in providing more realistic paths. [51] builds upon the social-LSTM work in [14], by embedding new factors encoding human-space interactions. [52] is another method that models both the physical terrain and the motion of other obstacles using a LSTM based GAN module.
- Map Aware: Map Aware models use information from a HD map of the environment as additional information to estimate future trajectories. [53] makes use of HD maps. The approach uses maps to provide a structure to where obstacles can or cannot be. The future trajectory of vehicles is predicted by constraining the vehicle path based on road geometry and constraints such as lane maintenance. [54] uses a neural network model called VectorNet to encode HD maps and agent dynamics in terms of vectors.

3) Based on Modelling Approach: The modelling approach refers to the algorithm based on a physics-based approach or a more general data-driven(learning-based) approach. Physics-based approaches rely on kinematics and dynamics of the obstacle to predict the motion. In learning approaches, the algorithm uses past obstacle data and other information like the road structure and motion of other obstacles to find a correlation with the obstacle's future trajectory.

- Physics-Based Approaches: These approaches are algorithms where predictions are made following the rules of physics, either dynamic or kinematic models. Dynamic models consider the forces involved in creating the motion. For human trajectory predictions, it is not necessary to calculate these forces, and the dynamic modelling becomes complex and irrelevant [49]. Kinematic models describe the motion of the obstacles in mathematical form. There are many simple kinematic models where assumptions such as CV and CA are used. The obstacle is assumed to move at a constant velocity or acceleration based on its previous motion.
- Learning-Based: Learning-based algorithms do not

Trajectory Prediction			
Type of Learning Algorithms	**Type of Output**	**Type of Situational Awareness**	**Algorithms**
Sequential	Uni-Modal	Interaction Aware	Social LSTM [55], Group LSTM [16]
		Scene Aware	Graph2Kernel Grid-LSTM [56], Context Aware LSTM [15]
	Multi-Modal	Interaction Aware	Social GAN [28]
		Scene Aware	Social-VRNN [26], Trajectron++ [24], Sophie [57], Social BiGAT [6], Goal-GAN [5], MG-GAN [4]

TABLE V: Classifying Learning based state-of-the-art algorithms

use complex physical-model to represent an obstacle; instead, they use data of similar obstacles and history to model the new trajectory. These approaches have become popular since the advent of DL. The model might use sensor inputs such as velocity and position of the obstacle during its past trajectory and may also use the same to precept the scene in which the obstacle is present. Learning-based approaches can be uni-modal/ multi-modal, unaware/ interaction aware/ scene aware, or map aware. The output/ situational awareness type is usually based on the problem statement and the available sensor data. Literature [49] classifies learning algorithms into two types; Sequential and non-sequential models.

- Non-Sequential Models learn over the current data directly, and these models do not use data from previous frames to condition the output. [58] uses a 2D CNN for trajectory prediction and presents position normalization techniques and data augmentation techniques for the trajectory prediction problem.
- Sequential Models perform trajectory prediction based on the history of their past motion. These approaches try to capture long-term dependencies, using DL tools such as LSTM, GRU, CVAE and GAN.

The table V summarizes different trajectory prediction algorithms and categorizes them based on the defined classification approaches.

Bibliography

[1] M. Gulzar, Y. Muhammad, and N. Muhammad, "A Survey on Motion Prediction of Pedestrians and Vehicles for Autonomous Driving," *IEEE Access*, vol. 9, pp. 137957–137969, 2021.

[2] J. Görtler, R. Kehlbeck, and O. Deussen, "A Visual Exploration of Gaussian Processes," *Distill*, vol. 4, p. e17, 4 2019.

[3] "Car History Timeline: From 3-Wheeled Buggies to Self-Driving Vehicles - HISTORY."

[4] A. Shaout, D. Colella, and S. Awad, "Advanced driver assistance systems - Past, present and future," *ICENCO'2011 - 2011 7th International Computer Engineering Conference: Today Information Society What's Next?*, pp. 72–82, 2011.

[5] "J3016_202104: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles - SAE International."

[6] "Home – Waymo."

[7] "Nuro - On a mission to better everyday life through robotics. | Nuro."

[8] O. de Groot, B. Brito, L. Ferranti, D. Gavrilå, and J. Alonso-Morå, "Scenario-Based Trajectory Optimization in Uncertain Dynamic Environments," p. 2021.

[9] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces,"

[10] N. Bisagno, B. Zhang, and N. Conci, "Group LSTM: Group Trajectory Prediction in Crowded Scenarios," 2018.

[11] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2255–2264, 3 2018.

[12] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-Feasible Trajectory Forecasting with Heterogeneous Data," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12363 LNCS, pp. 683–700, 8 2020.

[13] P. Dendorfer, S. Elflein, and L. Leal-Taixé, "MG-GAN: A Multi-Generator Model Preventing Out-of-Distribution Samples in Pedestrian Trajectory Prediction," 8 2021.

[14] S. Arora, A. Risteski, and Y. Zhang, "Do GANs learn the distribution? Some Theory and Empirics," 2 2022.

[15] S. Arora and Y. Zhang, "Do GANs actually learn the distribution? An empirical study," 6 2017.

[16] P. Dendorfer, A. Ošep, and L. Leal-Taixé, "Goal-GAN: Multimodal Trajectory Prediction Based on Goal Position Estimation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12623 LNCS, pp. 405–420, 10 2020.

[17] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, S. Hamid Rezatofighi, and S. Savarese, "Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks," *Advances in Neural Information Processing Systems*, vol. 32, 7 2019.

[18] R. Chandra, U. Bhattacharya, C. Roncal, A. Bera, and D. Manocha, "RobustTP: End-to-End Trajectory Prediction for Heterogeneous Road-Agents in Dense Traffic with Noisy Sensor Inputs," *Proceedings - CSCS 2019: ACM Computer Science in Cars Symposium*, 7 2019.

[19] "The scenario approach: A tool at the service of data-driven decision making - ScienceDirect."

[20] M. C. Campi and S. Garatti, "THE EXACT FEASIBILITY OF RANDOMIZED SOLUTIONS OF UNCERTAIN CONVEX PROGRAMS *," vol. 19, no. 3, pp. 1211–1230.

[21] M. C. Campi, S. Garatti, and F. A. Ramponi, "A general scenario theory for nonconvex optimization and decision making," *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, vol. 63, no. 12, 2018.

[22] Z. E. Lee, Q. Sun, Z. Ma, J. Wang, J. S. MacDonald, and K. Max Zhang, "Providing Grid Services With Heat Pumps: A Review," *ASME Journal of Engineering for Sustainable Buildings and Cities*, vol. 1, 2 2020.

[23] X. Geng and L. Xie, "Data-driven decision making in power systems with probabilistic guarantees: Theory and applications of chance-constrained optimization," *undefined*, vol. 47, pp. 341–363, 1 2019.

[24] A. Borri, F. Cacace, A. De Gaetano, A. Germani, C. Manes, P. Palumbo, S. Panunzi, and P. Pepe, "Luenberger-Like Observers for Nonlinear Time-Delay Systems with Application to the Artificial Pancreas: The Attainment of Good Performance," *IEEE Control Systems*, vol. 37, pp. 33–49, 8 2017.

[25] G. Arici, M. C. Campi, A. Carè, M. Dalai, and F. A. Ramponi, "A Theory of the Risk for Empirical CVaR with Application to Portfolio Selection," *Journal of Systems Science and Complexity 2021 34:5*, vol. 34, pp. 1879–1894, 10 2021.

[26] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 4, pp. 4459–4466, 10 2020.

[27] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "A Data-driven Model for Interaction-aware Pedestrian Motion Prediction in Object Cluttered Environments,"

[28] M. Lisotto, P. Coscia, and L. Ballan, "Social and Scene-Aware Trajectory Prediction in Crowded Spaces," 9 2019.

[29] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, S. H. Rezatofighi, and S. Savarese, "SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints," 2019.

[30] Y. Yoon, T. Kim, H. Lee, and J. Park, "Road-Aware Trajectory Prediction for Autonomous Driving on Highways," *Sensors 2020, Vol. 20, Page 4703*, vol. 20, p. 4703, 8 2020.

[31] X. Fang, J. Huang, F. Wang, L. Zeng, H. Liang, and H. Wang, "ConSTGAT: Contextual Spatial-Temporal Graph Attention Network for Travel Time Estimation at Baidu Maps," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2697–2705, 8 2020.

[32] S. Zamboni, Z. T. Kefato, S. Girdzijauskas, C. Norén, and L. Dal Col, "Pedestrian trajectory prediction with convolutional neural networks," *Pattern Recognition*, vol. 121, p. 108252, 1 2022.

[33] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces,"

[34] S. Haddad and S. K. Lam, "Graph2Kernel Grid-LSTM: A Multi-Cued Model for Pedestrian Trajectory Prediction by Learning Adaptive Neighborhoods," 7 2020.

[35] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, "Context-Aware Trajectory Prediction," *Proceedings - International Conference on Pattern Recognition*, vol. 2018-August, pp. 1941–1946, 5 2017.

[36] B. Brito, H. Zhu, W. Pan, and J. Alonso-Mora, "Social-VRNN: One-Shot Multi-modal Trajectory Prediction for Interacting Pedestrians," 10 2020.

[37] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 1349–1358, 6 2018.

Appendix A

Glossary

List of Acronyms

ADE	Average Displacement Error
CVAE	Conditional Variational Autoencoders
DL	Deep Learning
FDE	Final Displacement Error
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
HD	High Definition
LSTM	Long Short-Term Memory
LMPCC	Local Model Predictive Contouring Control
PEM	Partitioned Experience Method