EINDHOVEN UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING

CONTROL SYSTEMS GROUP

SAFE MOTION PLANNING IN DYNAMIC ENVIRONMENT USING MPC AND TEMPORAL LOGIC SPECIFICATIONS

Master Thesis

BHASHIK KAMBLE

GRADUATION

Program:	System and Control
Supervisor 1:	Prof. Dr. ir. Sofie Haesaert
Supervisor 2:	Zengjie Zhang
Student ID:	1731033
Date of defense:	January 30, 2024

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

12/15/2023

……..……..

Name

Kamble, Bhashik

……..……..

ID-number

1731033

……..……..

Signature

……..……..

Submit the signed declaration to the student administration of your department.

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU

Safe Motion Planning in Dynamic Environment Using MPC and Temporal Logic Specifications

Bhashik Kamble
Control Systems Group, Eindhoven University of Technology, Eindhoven, The Netherlands
b.kamble@student.tue.nl

***Abstract*—In advancing pedestrian safety in autonomous vehicle navigation, this contemporary research integrates Long Short-Term Memory (LSTM) networks that are sophisticated variants of Recurrent Neural Networks (RNN) with Model Predictive Control (MPC) and Signal Temporal Logic (STL). It focuses on LSTM's capability for predicting pedestrian trajectories, leveraging its advanced data processing ability to capture complex temporal dynamics and patterns in pedestrian movement. A novel aspect of this work is the incorporation of conformal prediction, improving the reliability of LSTM-based pedestrian trajectory forecasts. This predictive framework is integrated into an MPC system, which is further refined by STL to rigorously define safety constraints. Empirical simulations demonstrate the autonomous system's ability to adapt to real-time pedestrian movements, maintaining adherence to stringent safety specifications. Through the use of intricate techniques such as STL and conformal prediction, the research offers a robust and dynamic solution for pedestrian safety in urban environments.**

***Index Terms*—Conformal Prediction, Signal Temporal Logic, Model Predictive Control, Recurrent Neural Network.**

I. Background and Introduction

The advent of A.I and Machine learning techniques have made many future technologies, including autonomous driving possible. The responsiveness of autonomous vehicles to avoid obstacles, make complex maneuvers, maintain a safe driving distance, and, at the same time, follow all traffic signals with high fidelity has significantly increased in recent times. However, safety remains a cause of concern. According to the US National Highway Safety Administration (NHSTA), in 10 months between 1 July 2021, and 15 May 2022, automakers reported approximately 400 accidents of vehicles equipped with partially automated driver-assist systems [1]. Out of 98 crashes with injury reports, 11 resulted in serious injuries, and 5 were fatal. The occurrences of such accidents, however rare, pose as impediments to the adoption of autonomous vehicles by the general populous at large. To overcome these impediments, autonomous vehicles need to assess and react more effectively to the uncertainties that manifest in their environment. The report published by NHSTA also delineates that 11 crashes involved vulnerable road users like pedestrians and bicyclists [1]. The uncertainties encountered by autonomous vehicles, in the form of pedestrians and bicyclists, are difficult to gauge since their intentions are unknown, and their movement can be erratic. At the same time, any collision of an autonomous vehicle, especially with pedestrians, is probabilistically more fatal because pedestrians are not protected by any structure or safety equipment, such as seat belts present in motor vehicles. Therefore, it is in our interest to predict pedestrians' trajectories with uncertainty quantification as a fail-safe so that it can be leveraged by the autonomous vehicle's motion planner to safely navigate around pedestrians in various scenarios at different times. Most of the previous works in the area of autonomous vehicle behavior are predominantly focused on the reactive domains. For instance, the paper on multi-agent navigation via dynamic window approach [2]. However, the reactive approach typically considers simplified dynamics and, consequently, does not optimize performance. However, the predictive approach considers the predictions of the agent's future motion, and the control action is not governed by predefined rules or heuristics, so it can optimize performance. More recently, a growing interest has been in interactive approaches, i.e., considering inter-agent interactions, as illustrated in the works of [3], [4]. The research presented in this paper lies at the intersection of these three approaches.

Firstly, the motion of the agent (pedestrians) is predicted in the future using a Long Short-Term Memory (LSTM) neural network. We do not impose any assumptions on the underlying trajectory-generating distribution. These predictions are then considered to appropriately react by the autonomous vehicle's motion planner. While we focus on non-interactive planning algorithms, we do implicitly consider interactions, which will be described later and provide valid safety guarantees. For providing safety guarantees, we use conformal prediction to construct prediction regions that quantify the prediction uncertainty of predictions made using LSTM neural network. Thus, the contributions are two-fold and are as follows:

1) Prediction of trajectories for pedestrians using LSTM trajectory predictor, and an algorithm that quantifies the uncertainty associated with the point predictions using conformal prediction, a statistical tool for uncertainty quantification.
2) A Model Predictive Controller (MPC) motion planner for the autonomous vehicle that considers the uncertainty quantification made using conformal prediction. These uncertainty quantification are subsequently incorporated into the STL specifications, enabling the planning of probabilistic safe paths for the autonomous vehicle.

Supervision: Dr. ir. Sofie Haesaert, Electrical Engineering - Control systems group, Eindhoven University of Technology, s.haesaert@tue.nl.
Zengjie Zhang, Electrical Engineering - Control systems group, Eindhoven University of Technology, z.zhang3@tue.nl

LSTM, or Long short-term memory, is a type of recurrent neural network (RNN) that allows a network to learn long-term dependencies between time steps in time series or sequential data. It correlates past and current information, making it ideal for time series forecasting. The equations of a Long Short-Term Memory (LSTM) unit in a recurrent neural network (RNN) are composed of several components that govern the flow of information and control memory retention and update [5]. Trajectory predictors, such as the LSTMs, however, provide no information about prediction uncertainty. Since we are interested in trajectory predictions of pedestrians, it is paramount to know the uncertainties that manifest in their trajectory predictions. Else, even a slight error in point predictions made by the LSTM neural network can lead the autonomous vehicle to endanger the lives of pedestrians. Consequently, to monitor the prediction quality of trajectory predictors and predictors' false negative rate, prediction monitors were presented in the works of [6], [7].

Conformal prediction (CP) is a type of prediction monitor that was introduced as a statistical tool to quantify prediction uncertainty of prediction algorithms, as illustrated in the works of [8], [9]. In essence, CP is a novel framework for constructing prediction intervals that attain valid coverage in finite samples, without assuming the distribution of samples. However, since its conception, it has undergone various developments to address problems such as constant or weakly varying prediction intervals, illustrated in the work of [10], and is now being applied to real-world problems in classification and regression, as highlighted in the work of [11].

Building upon the advancements in predictive uncertainty quantification through conformal prediction, we can see its practical implications and application. Particularly in the context of control systems, where the motion planner, such as MPC, can leverage these uncertainty quantification. The Model Predictive Control strategy is based on iterative, finite horizon optimization over the model of the plant, which is the system to be controlled. For a given time t, the current plant state is observed, and an optimal scheme is computed for some finite time horizon in the future. Then, an online computation is executed to explore the trajectories originating from the current state; subsequently, an optimal control strategy is computed for a time horizon $t + H$. To avoid any anomalies with respect to modeling errors, only the first step of the computed optimal control strategy is implemented. The plant state is re-sampled, and a new computation is performed on a horizon H starting from the new current state. By doing this, the system's robustness with respect to exogenous disturbances and modeling uncertainties is improved. This, in addition to MPC's ability to handle constrained dynamic systems, along with the ability to handle conformalized uncertainty quantification made using conformal prediction, makes it an ideal candidate as the motion planner for the autonomous vehicle.

As we explore the integration of conformal prediction with MPC in autonomous vehicles, ensuring safety and responsiveness becomes paramount. This leads us to the incorporation of Signal Temporal Logic (STL) in our framework. STL offers a robust and precise language for defining and verifying complex temporal properties, such as safety and responsiveness, which are critical in autonomous vehicle navigation. By combining the predictive accuracy and control optimization of MPC with rigorous safety specifications specified using STL, we can create a more comprehensive and reliable system. This synergy not only enhances the vehicle's ability to predict and react to dynamic environments but also ensures adherence to stringent safety requirements, making autonomous driving safer and more efficient.

Conceptually closest to our proposed research objectives are the works from [12] and [13]. [12] obtained prediction uncertainty quantification using conformal prediction, which they utilize to design Model Predictive Controller (MPC). Although [12] provides probabilistic safety guarantees for the motion planner, they do not consider STL specifications to define the safe behavior of the self-driving autonomous system. Additionally, the method we employ to calculate conformal prediction regions in our work differs from their proposed approach.

As stated before, we mainly focus on non-interactive planning algorithms, which manifests in the work of [14], [15], [16], where they present non-interactive sampling-based motion planner. Long time horizons for non-interactive methods cause an autonomous vehicle to come to a deadlock because of large prediction uncertainty, as illustrated in the work of [17]. To mitigate this, a receding horizon planning strategy (RHC) was deployed in [18]. While interactive approaches, as highlighted in the works of [4] and [19], can also address the issue by modeling social interactions and constructing learned interaction models, they can become convoluted in their implementation. In search for a decent predictive model, intent-driven models have been proposed in [20], [21] for human agents, where uncertainty is estimated using Bayesian inference and subsequently used for planning. However, these works impose strong assumptions on the prediction algorithm and the agent model, including, for instance, the consideration of Gaussian distributions. Our primary focus remains on the predictive perks of neural networks, particularly, RNNs such as LSTM for the prediction of pedestrian trajectories and using distribution-free conformal prediction framework to provide probabilistic safety guarantees.

The outline for the remaining paper is as follows. Section II sets the foundation by presenting the mathematical preliminaries. Section III, explicitly describes the problem formulation with four interconnected sub-problems. Section IV illustrates pedestrian trajectory forecasting with LSTM. Building upon this, Section V explores the utilization of conformal prediction for calculating prediction intervals. Finally, Section VI illustrates simulation results of the Model Predictive Control (MPC) framework with conformal prediction and Signal Temporal Logic (STL) specifications to plan probabilistic safe path for the autonomous vehicle.

II. Preliminaries

A. Long Short Term Memory Network

LSTM networks known for their efficacy in sequence prediction [22], recursively update hidden states and output

predictions. At each time step τ, with $\tau \leq t$, the network's transformations are governed by

$$\begin{aligned} h_\tau^1 &= \mathcal{H}(y_\tau, h_{\tau-1}^1) \\ h_\tau^i &= \mathcal{H}(y_\tau, h_{\tau-1}^i, h_\tau^{i-1}) \quad \forall i \in 2,3,4,...d \\ \hat{y}_{\tau+1|\tau} &= \mathcal{Y}(h_\tau^d) \end{aligned} \tag{1}$$

where the parameters y_τ are the inputs that are sequentially applied, $\mathcal{H}$ parameterizes LSTM network, d is depth of the RNN, h_τ^1, h_τ^2,..., h_τ^d are the sequestered states [12]. The function $\mathcal{Y}$ utilizes its learned parameters to generate point predictions for the next time step based on the information encoded in the final hidden state h_τ^d.

The RNN transforms a sequence of observations, $y_0, y_1, \ldots, y_t$, into a series of future predictions. Let, the term H denotes the prediction horizon, which is the number of future time steps for which the model makes predictions. Therefore, the RNN generates predictions ranging from $\hat{y}_{t+1|t}$, the forecast for y_{t+1} based on observations up to time t, to $\hat{y}_{H|t}$, the prediction for the H-th time step in the future. To determine subsequent predictions for $y_{t+2}, \ldots, y_H$, the RNN employs a recursive approach, using each successive prediction, from $\hat{y}_{t+1|t}$ to $\hat{y}_{H-1|t}$, as new input to sequentially generate the next step's prediction.

B. Conformal Prediction

The conformal prediction (CP) framework leverages any underlying point forecasting prediction model to produce multi-step prediction intervals with coverage guarantees across the prediction horizon. For a significance level α, CP returns a prediction interval Γ^α that guarantees to contain the true value with a probability of at least (1-α).

Assumption 1 (Exchangeability) [23]: In a dataset of l observations $\left\{(x^i, y^i)\right\}_{i=1}^l$ any of the $l!$ permutations are equiprobable. Note if an identically distributed sequence is independent, then the sequence is exchangeable; however, the converse is false.

Given a set of observations $\mathcal{D} = \left\{(x^i, y^i)\right\}_{i=1}^l$ and a new example x^{l+1}, the Inductive Conformal Prediction (ICP) algorithm returns a prediction interval Γ^α such that property of validity is satisfied [8], [24], [25]. Here, $x^i \in \mathbb{R}^d$ is the vector of attributes for example i, and $y^i \in \mathbb{R}$ is the label of that example.

Property 1 (Validity) [8]: Under exchangeability assumption as described above, any conformal predictor will return the prediction region $\Gamma^\alpha(x^{(i)})$ such that the probability of error $y^{(l+1)} \notin \Gamma^\alpha(x^{(l+1)})$ is not greater than α, alternatively it is given as

$$\mathbb{P}[y^{(l+1)} \in \Gamma^\alpha(x^{(l+1)}) \mid \mathcal{D} \geq 1 - \alpha]. \tag{2}$$

The conformal prediction framework is distribution-free, i.e., it does not consider any assumptions on the distribution of the underlying data and applies virtually to any predictive model. The only assumption made is that of exchangeability. We use an inductive variant of conformal prediction that is described in Section V on conformal prediction.

C. Signal Temporal Logic

An STL formula is recursively defined according to

$$\phi ::= \pi^\mu \mid \neg\pi^\mu \mid \phi \wedge \psi \mid \phi \vee \psi \mid \Box_{[a,b]}\psi \mid \phi_1 \mathcal{U}_{[a,b]}\psi \tag{3}$$

where π^μ is an atomic predicate $\mathbb{R}^n \to \mathbb{B}$ whose truth value is determined by the sign of a function μ: $\mathbb{R}^n \to \mathbb{R}$ and ψ is an STL formula [26]. The validity of a formula ϕ with respect to the discrete-time signal $\mathbf{x}$ at time t_k, represented as $(\mathbf{x}, t_k) \vDash \phi$ is defined inductively as follows:

$$\begin{aligned} (\mathbf{x}, t_k) &\vDash \pi^\mu && \Leftrightarrow \mu(x_k) > 0 \\ (\mathbf{x}, t_k) &\vDash \neg\pi^\mu && \Leftrightarrow \neg((\mathbf{x}, t_k) \vDash \pi^\mu) \\ (\mathbf{x}, t_k) &\vDash \phi \wedge \psi && \Leftrightarrow (\mathbf{x}, t_k) \vDash \phi \wedge (\mathbf{x}, t_k) \vDash \psi \\ (\mathbf{x}, t_k) &\vDash \phi \vee \psi && \Leftrightarrow (\mathbf{x}, t_k) \vDash \phi \vee (\mathbf{x}, t_k) \vDash \psi \\ (\mathbf{x}, t_k) &\vDash \Box_{[a,b]}\phi && \Leftrightarrow \forall t_{k'} \in [t_k + a, t_k + b], (\mathbf{x}, t_{k'}) \vDash \phi \\ (\mathbf{x}, t_k) &\vDash \phi \mathcal{U}_{[a,b]}\psi && \Leftrightarrow \exists t_{k'} \in [t_k + a, t_k + b] \text{ s.t. } (\mathbf{x}, t_{k'}) \\ & && \quad \vDash \psi \wedge \forall t_{k''} \in [t_k, t_{k'}], (\mathbf{x}, t_{k''}) \vDash \phi. \end{aligned} \tag{4}$$

A signal $\mathbf{x} = (x_0, x_1, x_2, \ldots)$ satisfies ϕ, denoted by $\mathbf{x} \vDash \phi$, if $(\mathbf{x},t_0) \vDash \phi$. Informally, $\mathbf{x} \vDash \Box_{[a,b]}\phi$ if ϕ holds at every time step between time points a and b. Furthermore, $\mathbf{x} \vDash \phi\ \mathcal{U}_{[a,b]}\ \psi$, if ϕ holds at every time step before ψ holds at some time between step between a and b. An STL (Signal Temporal Logic) formula ϕ is considered bounded-time if all its operators are bound to specific time limits. The bound of ϕ is determined by calculating the maximum sum of all upper bounds on its nested temporal operators. This value serves as a conservative estimate for the maximum trajectory length needed to assess whether ϕ can be satisfied. For instance, for $\Box_{[0,10]}\ \Diamond_{[1,7]}\phi$, a trajectory of length $N \geq 10 + 7 = 17$ is sufficient to determine if the formula is satisfiable.

III. Problem formulation

The primary objective is to control an autonomous vehicle navigating in an environment so that it does not collide with dynamically moving pedestrian. The vehicle's dynamics are modeled by a discrete-time dynamical system

$$x_{t+1} = f(x_t, u_t), \quad x_0 = \zeta \tag{5}$$

where $x_t \in \mathcal{X} \subseteq \mathbb{R}^n$ and $u_t \in \mathcal{U} \subseteq \mathbb{R}^m$ denote the state and the control input at time $t \in \mathbb{N} \cup 0$ respectively. The sets $\mathcal{U}$ and $\mathcal{X}$ denote the set of permissible control input and the state of the system, respectively. The function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ describes the dynamics of the autonomous vehicle and $\zeta \in \mathbb{R}^n$ is the initial condition of the system. Let, $x = (x_0, x_1, x_2, \ldots)$ denote the trajectories of the dynamical system in (5) with a control sequence $u = (u_0, u_1, u_2, \ldots)$. For the purposes of control and prediction, we consider a finite prediction horizon of N discrete time steps, over which the discrete-time system's (5) future states are predicted and control decisions are made. To achieve our objective we define four sub-problems that we solve sequentially.

Sub-Problem 1: Pedestrian Trajectory Prediction

- Train an LSTM neural network ($\mathcal{H}$) to predict pedestrian trajectories based on dataset D with N dynamic pedestrians with unknown distribution $\mathcal{D}$ over the pedestrian

trajectories without imposing any restriction on the distribution $\mathcal{D}$, for instance, Gaussian or Binomial distribution.

- Let, Y_t = (Y_0, Y_1, Y_2,…) $\sim \mathcal{D}$ represent the random trajectory where the joint agent state $Y_{t,j}$ = ($Y_{t,1}$, $Y_{t,2}$, $Y_{t,3}$,…, $Y_{t,N}$) at time $t \in \mathbb{N} \cup 0$ is drawn from $\mathbb{R}^{nN}$. The state of each dynamic agent is n-dimensional. $Y_{t,j}$ is the state of the agent j at time t. Lowercase letters y_t are used to refer to a realization of Y_t, and we assume at time t to have knowledge of (y_0, y_1, y_2,…, y_t). Thus, the objective is to predict the trajectory of pedestrians $\hat{y}_{[(t+1):(t+H)]|t}$ from time $(t+1)$ to $(t+H)$ given the information up to time t using an LSTM neural network ($\mathcal{H}$) for a prediction horizon H. The training, calibration, and test data are drawn from $\mathcal{D}$.

Sub-Problem 2: Uncertainty Quantification

- Quantify the uncertainty in pedestrian trajectory predictions with conformal prediction (CP) framework and subsequently obtain prediction intervals $\Gamma^{\alpha}_{t+h} = [\hat{y}^{L}_{t+h}, \hat{y}^{U}_{t+h}]$ for each step $h \in \{1, \ldots, H\}$, with a coverage probability of $(1-\alpha)$, where α is a user-defined confidence level.
  - $\hat{y}^{L}_{t+h}$ represents the lower bound of the prediction interval at time $(t+h)$. This bound is the minimum expected position of the pedestrian, considering the uncertainty in the prediction.
  - $\hat{y}^{U}_{t+h}$ represents the upper bound of the prediction interval at time $(t+h)$. This bound is the maximum expected position of the pedestrian, considering the uncertainty in the prediction.

Sub-Problem 3: Formulate STL Safety Specification

- Given a pedestrian's predicted position, denoted by $\hat{y}_{t+h}$, for each time step $(t+h)$ within the prediction horizon H, and the associated prediction interval Γ^{α}_{t+h}. The STL safety specification ϕ can be expressed as:

$$\phi = \Box_{[a,b]}\, \mu \tag{6}$$

  Here, $\Box_{[a,b]}$ is the STL temporal operator meaning "always in the interval $[a, b]$" and μ is an atomic predicate.
- Based on the STL formulas provided in Section II-C, we define an atomic predicate μ, as part of our STL formula ϕ, to encapsulate safety conditions for the autonomous vehicle. The predicate μ can be expressed in the context of the STL framework as follows

$$\mu = (x_t \notin \Gamma^{\alpha}_{t+h}). \tag{7}$$

  This predicate ensures that the state of the vehicle x_t at time t should not be within the prediction interval Γ^{α}_{t+h} of the pedestrian. Additionally, the predicate μ also encodes that the vehicle is required to adjust its velocity to a predefined safe velocity in the vicinity of the pedestrian. Thus, the predicate μ enforces adherence to safety margins around the pedestrian's predicted position.

Sub-Problem 4: MPC Framework

- Integrate the pedestrian trajectory predictions, uncertainty quantifications, and safety specification into an MPC framework to compute an optimal control sequence for the autonomous vehicle.
- The MPC problem is formulated as follows:

$$\begin{aligned} \underset{u_{[t:t+N]}}{\text{minimize}} \quad & J(x_{t:t+N}, u_{t:t+N}) \\ \text{subject to} \quad & x_{\tau+1} = f(x_\tau, u_\tau), \quad \forall \tau \in \{t, \ldots, t+N-1\}, \\ & x_\tau \in \mathcal{X}, \quad u_\tau \in \mathcal{U}, \quad \forall \tau \in \{t, \ldots, t+N\}, \\ & x_\tau \models \phi, \qquad\qquad \forall \tau \in \{t, \ldots, t+N-1\} \end{aligned} \tag{8}$$

  Here, N represents the prediction horizon in discrete time steps, over which the MPC optimizes the control sequence. $J(x_{t:t+N}, u_{t:t+N})$ denotes the cost function, encompassing terms for tracking performance and penalizing control effort. The term ϕ signifies the STL safety constraints, which are derived from pedestrian trajectory predictions and their associated uncertainties. These constraints ensure that the vehicle's planned path maintains safety standards, taking into account the uncertainties in pedestrian movements.

Thus, the problem formulation methodically tackles the intricate challenge of ensuring the safety of autonomous vehicles operating in environments with dynamic pedestrians. This is achieved through four interrelated sub-problems, each addressing a critical aspect of the overall challenge. By integrating predictive analysis for pedestrian movement, rigorous uncertainty assessment, formal methods encapsulated in STL specifications, and advanced control theory in the MPC framework, the formulation holistically targets both safety and efficiency. This comprehensive approach is designed to enhance the capability of autonomous vehicles to navigate complex, real-world scenarios reliably.

IV. Pedestrian trajectory forecasting with LSTM

A. Dataset description and preprocessing

The dataset D used for pedestrian trajectory prediction is a publicly available ETH pedestrian dataset [27], which has been processed to get sequential time series data corresponding to each pedestrian. The dataset captures pedestrian movements at the entrance of ETH University's main building. It is annotated at 2.5 fps or every 0.4 seconds from a bird's eye view and consists of 360 pedestrians in total with approximately 20 data points per pedestrian. The dataset's features consist of 'frame', 'pedestrian id', 'x-position', 'y-position', 'z-position', 'x-velocity', 'y-velocity', 'z-velocity'. We extract the features 'x-position' and 'y-position' from the dataset to train the LSTM neural network for trajectory prediction. Since the video is recorded from a top view, the attributes 'z-position' and 'z-velocity' are all zeros.

Figure 1 illustrates the scatter plot of all pedestrians' trajectories from ETH University's dataset. The abscissa depicts the 'x-position' and the ordinate depicts the 'y-position' of the pedestrians. The trajectories exhibit a notable directional flow, as evidenced by the elongated distribution of data points along the abscissa. This pattern suggests a predominant pedestrian route reflecting the layout of the walkway of foot traffic in the observed area. The distribution does not appear to be uniform, which implies the existence of preferred paths or the influence of environmental factors on pedestrian movement.

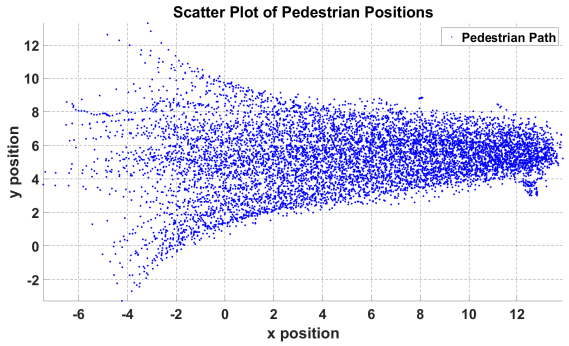

Fig. 1: Pedestrian Trajectories in ETH dataset

The dataset employed in this research, while a valuable resource, presents certain limitations concerning environmental diversity and dimensionality. The dataset is collected from a specific location around ETH University's main building, which might not represent the diversity of pedestrian behaviors in different environments. This lack of variability can introduce a significant bias, as the dataset may not adequately capture the complexities and nuances of pedestrian movements in varied urban contexts. Moreover, without the z-axis information, the dataset cannot capture changes in elevation, such as slopes, stairs, or hills, which can significantly affect pedestrian dynamics. Thus, the LSTM model may oversimplify pedestrian behavior by assuming a perfectly flat 2D plane, potentially overlooking the complexities of real-world pedestrian movement.

B. Model selection

Figure 2 presents a three-dimensional scatter plot delineating the trajectories of five distinct pedestrians from dataset D as captured over a sequence of time frames. The abscissa (x-axis) quantifies the temporal dimension in frames, the ordinate (y-axis) denotes the lateral displacement (x-position), and the applicate (z-axis) corresponds to the longitudinal displacement (y-position) within a two-dimensional observational plane. The paths followed by pedestrians display a variety of directional shifts and deviations from linear progression. Notably, the trajectories of Pedestrians 1 and 5 are characterized by curvilinear patterns, which accentuate the occurrence of turns and directional changes. These observations corroborate the inherently non-stationary nature of the pedestrian trajectories.

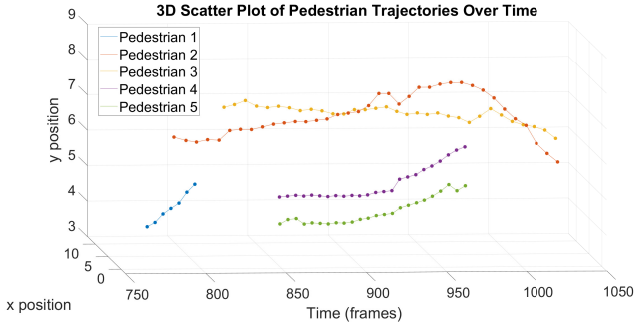

Fig. 2: 3D Scatter Plot of Pedestrian Trajectories Over Time

This non-stationarity in pedestrian trajectories can often be mitigated by applying different transformations at the cost of removing trends and seasonality [28]. LSTM is proven to perform satisfactorily without accounting for non-stationarity [29], [30], which makes it ideal for pedestrian trajectory forecasting. However, LSTM computations are prone to data scaling, consequently, the input and target sequence are normalized by calculating mean and standard deviation for training the LSTM neural network.

Figure 3 provides a schematic representation of the LSTM network, detailing the various layers and the nodes within each layer that are instrumental in the training process. The sequential layer is used as an input layer and is utilized to train weights and biases that are used to make predictions. Next, an LSTM layer is defined which is the core of the recurrent neural network. It processes sequences of data and captures temporal dependencies. After the LSTM layer, a fully connected layer is added. This layer takes the output of the LSTM layer and performs a linear transformation on it. Finally, the regression layer is used as the output layer. This layer computes the loss function used during training. The goal of the regression layer is to minimize the mean squared error (MSE) between the predicted values and the actual target values. During training, the weights are initialized randomly and the biases are initialized as one to prevent information loss. The initialization of cell state and hidden state vectors is done with zero. The model training is done on a single GPU NVIDIA GeForce GTX 1050 Ti on a Windows PC with Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz. The training code is compiled in MATLAB 2023a.

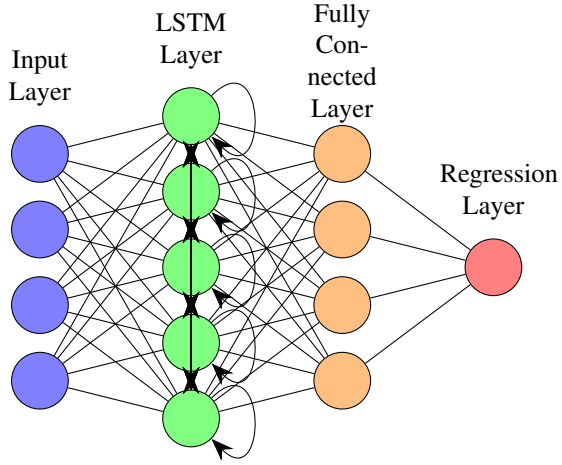

Fig. 3: Neural Network Architecture

C. Hyperparameter selection

To identify the optimal hyperparameters, we conducted a grid search, focusing on three key parameters: Mini-batch size, Learning rate, and the number of hidden units. This involved testing various permutations of these parameters to determine the combination that minimizes the Root Mean Square Error (RMSE) between the predicted and actual pedestrian

trajectories in the calibration dataset D_{cal}. The following list details the hyperparameters and their respective optimal values along with other crucial parameters that are involved while specifying the LSTM neural network architecture:

1) **Mini-batch Size:** A mini-batch is a subset of the training set that is used to evaluate the gradient of the loss function and update the weights. Based on the grid search performed, a mini-batch size of 64 is used.
2) **Maximum Epochs:** A maximum epoch of 150 is experimentally determined.
3) **Optimizer and Learning Rate:** The Adam optimizer was employed for its adaptive learning rate capabilities and its robust performance in recurrent neural network training [29]. The initial learning rate was set to 0.1, as determined by the grid search to provide an optimal convergence rate.
4) **Loss Function:** Mean Squared Error (MSE) was chosen as the loss function to quantify the prediction accuracy, providing a clear measure for the optimization process.
5) **Number of Hidden Units:** The number of hidden units defines the memory capacity of the LSTM layer, with an excessive number potentially leading to overfitting. As a result of the grid search, 50 hidden units were selected to achieve an appropriate balance between model complexity and generalization ability.

D. Training

For training the LSTM model, the input sequence is created to be a normalized, concatenated array of all the pedestrians' x and y positions. The target sequence comprises all the subsequent, normalized x and y positions, each corresponding to an individual pedestrian. This specification of correct input and target sequence is crucial for correct training and ultimately obtaining an appropriate pedestrian prediction model. The dataset is split into training and test datasets. The training set is further split into two subsets i.e. proper training set and calibration set. The LSTM model is trained on the proper training set.

Let D be the entire dataset, D_{train} be the training set, D_{test} be the test set, $D_{\text{train}}^{\text{proper}}$ be the proper training set, and D_{cal} be the calibration set. Given a proportion p for the test set and q for the calibration set, where $p, q \in (0, 1)$ and $p + q < 1.$, the dataset splits is defined as follows:

- Split the entire dataset D into training and test sets. Here, p is set to be 0.3.

$$\begin{aligned} D &= D_{\text{train}} \cup D_{\text{test}} \\ |D_{\text{test}}| &= p \cdot |D| \\ |D_{\text{train}}| &= (1-p) \cdot |D|. \end{aligned}$$

- Further split the training set D_{train} into proper training and calibration sets. Here, q is set to be 0.2.

$$\begin{aligned} D_{\text{train}} &= D_{\text{train}}^{\text{proper}} \cup D_{\text{cal}} \\ |D_{\text{cal}}| &= q \cdot |D_{\text{train}}| \\ |D_{\text{train}}^{\text{proper}}| &= (1-q) \cdot |D_{\text{train}}|. \end{aligned}$$

where $|\cdot|$ denotes the size (number of samples) of the set. The LSTM model is then trained using $D_{\text{train}}^{\text{proper}}$ as the input. Note that $D_{\text{train}}^{\text{proper}}$, D_{cal}, and D_{test} are all disjoint sets:

$$\begin{aligned} D_{\text{train}}^{\text{proper}} \cap D_{\text{cal}} &= \emptyset \\ D_{\text{train}}^{\text{proper}} \cap D_{\text{test}} &= \emptyset \\ D_{\text{cal}} \cap D_{\text{test}} &= \emptyset. \end{aligned}$$

The calibration set is used to tune the model's hyperparameters and for early stopping. The RMSE between the predicted and actual trajectory on the calibration set is observed to be 0.09.

E. LSTM predictions

The trained LSTM model is used for predicting the trajectory on a completely unseen test dataset D_{test} that consists of m pedestrians. The predicted trajectory is obtained using open-loop prediction in an auto-regressive fashion such that the next time step in the sequence is predicted using the ground truth trajectory at each time step as shown below in Algorithm 1.

Algorithm 1 Pedestrian Trajectory Prediction using LSTM

Input: Test dataset $D_{\text{test}} = \{y_t^{(i)}\}_{i=1}^{m}$, where each $y_t^{(i)} \in \mathbb{R}^{2\times T}$ is a pedestrian trajectory corresponding to the i-th pedestrian, represented as a sequence of 2D positions over T time steps.
Output: Root Mean Square Error (RMSE) and Average Displacement Error (ADE) for trajectory predictions.
for $i = 1$ to m **do**
  Reset the state of the LSTM network, $\mathcal{H}$.
  Let T be the number of time steps in the i-th pedestrian's trajectory.
  for $t = 1$ to T **do**
    Set input $\xi \leftarrow y_t^{(i)}$, where $\xi \in \mathbb{R}^2$ represents the 2D position at time t.
    Predict next position: $\hat{y}_{t+1}^{(i)} \leftarrow \mathcal{H}(\xi)$.
  end for
  Compute RMSE and ADE for the i-th pedestrian's trajectory.
end for

For evaluating the LSTM model's predictive accuracy, two key metrics are employed: Root Mean Square Error (RMSE) and Average Displacement Error (ADE), each of which serves to quantify distinct aspects of the prediction error.

1) Root Mean Square Error (RMSE): RMSE is a robust measure that emphasizes larger errors in predictions, which is crucial in contexts where such errors are particularly impactful. It is defined as the square root of the average of the squared differences between the actual trajectories $y_t^{(i)}$ and their corresponding predictions $\hat{y}_t^{(i)}$. Here, $y_t^{(i)} \in \mathbb{R}^{2\times T}$ is a pedestrian trajectory corresponding to the i-th pedestrian, represented as a sequence of 2D positions over T time steps. For m pedestrians in the test dataset D_{test}, RMSE is mathematically expressed as:

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_t^{(i)} - \hat{y}_t^{(i)})^2}. \tag{9}$$

While RMSE provides a comprehensive measure of prediction accuracy, its sensitivity to outliers can sometimes lead to a skewed representation of the model's performance.

2) Average Displacement Error (ADE): As a counterbalance to RMSE, ADE is utilized to offer a more straightforward assessment of the average error magnitude in the model's predictions. It calculates the mean of the absolute differences between the predicted and actual trajectory points, thus providing an aggregate measure of error per trajectory. ADE is defined as:

$$ADE = \frac{1}{m}\sum_{i=1}^{m}|y_t^{(i)} - \hat{y}_t^{(i)}|. \tag{10}$$

Here, $y_t^{(i)}$ and $\hat{y}_t^{(i)}$ represent the actual and forecasted trajectories for the i-th observation, respectively, with m denoting the total number of observations.

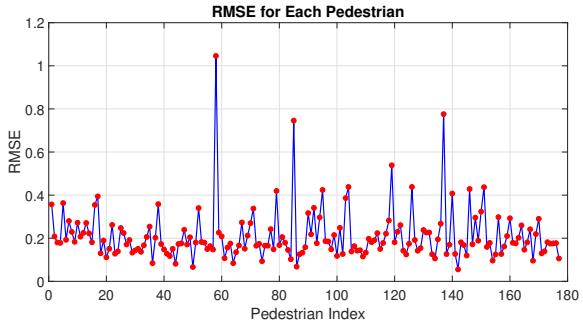

Fig. 4: Distribution of RMSE Across Different Pedestrians in the Test Dataset D_{test}

The RMSE for each pedestrian in the test set D_{test} is visualized as shown in Figure 4, calculated using Algorithm 1. The RMSE values are observed to be within acceptable limits, suggesting that the trained LSTM model makes predictions with good fidelity. Furthermore, the mean of all RMSE and ADE values across all pedestrians in the test dataset is as follows:

1) Overall RMSE: 0.093131
2) Overall ADE: 0.42127

The overall RMSE from the test set is similar to the RMSE computed on the calibration set prior. This indicates that the LSTM model has generalized effectively to a novel dataset.

V. Conformal Prediction

In high-stakes applications such as autonomous driving, the point predictions made using LSTM are inadequate. For such safety-critical applications, estimates of uncertainty are required for accurate risk assessment and decision-making. We use a computationally effective conformal forecasting framework that can leverage any underlying point forecasting model to produce multi-step prediction intervals with coverage guarantees across the prediction horizon. Specifically, for regression problems such as pedestrian trajectory prediction, we adopt Inductive Conformal Prediction (ICP). ICP modifies conformal prediction by incorporating an additional calibration set and an underlying model, enhancing its applicability for the problem of pedestrian trajectory forecasting.

For multi-horizon forecasting of pedestrian trajectories ($\hat{y}_{t+1|t}$, $\hat{y}_{t+2|t}$,..., $\hat{y}_{t+H|t}$) given the history of observed values (y_0, y_1, y_2,..., y_t) where H is the prediction horizon. We compute the uncertainty associated with the forecast for each step $h \in \{1, \ldots, H\}$. We obtain prediction intervals of the form $[\hat{y}_{t+h}^L, \hat{y}_{t+h}^U]$, where $\hat{y}_{t+h}^L$ and $\hat{y}_{t+h}^U$, represent the lower and upper bounds of the prediction interval, respectively, ensuring that the ground truth value y_{t+h} is contained within the interval with a sufficiently high probability. We achieve this by setting the significance level α so that the ground truth value of the pedestrian trajectory is contained within the interval. Algorithm 2 delineates the implementation of an inductive conformal prediction framework on our pedestrian dataset.

Algorithm 2 Inductive Conformal Prediction

Input: A trained model $\mathcal{H}$ producing H-steps (5-steps) ahead predictions in our case. A calibration dataset $D_{cal} = \{(y_t^{(i)}, y_{t+1:t+H}^{(i)})\}_{i=1}^{l}$, significance level $\alpha = 0.05$.
Output: Prediction interval for novel pedestrian time series.
for $i = 1$ to l **do**
  Reset the state of the LSTM network, $\mathcal{H}$.
  Let $t = 7$. Set input $\rho \leftarrow y_t^{(i)}$, where $\rho \in \mathbb{R}^2$ represents the 2D position at time t.
  for $h = 1$ to H **do**
    Predict next position: $\hat{y}_{t+h}^{(i)} \leftarrow \mathcal{H}(\rho)$.
    Set $\rho \leftarrow \hat{y}_{t+h}^{(i)}$
    NC score $\longrightarrow \|y_{t+h}^{(i)} - \hat{y}_{t+h}^{(i)}\|$
  end for
end for
Computation of quantile value
$\hat{\varepsilon} = Q(1-\alpha) \times \frac{(l+1)(1-\alpha)}{l}$
for a new pedestrian time series y_t^* **do**
  $\mathcal{H}(y_t^*) \longrightarrow \hat{y}_{t+1:t+H}^*$
  return intervals $y_{t+1}^* \pm \hat{\varepsilon}, \ldots, y_{t+H}^* \pm \hat{\varepsilon}$
end for

Lemma 1 (Conformal prediction validity): Let $\mathcal{D} = \left\{(y_{1:t}^{(i)}), (y_{t+1:t+H}^{(i)})\right\}_{i=1}^{l}$ be the dataset of l exchangeable time-series corresponding to l pedestrians and their H-step forecasts is obtained from same underlying probability distribution. $\mathcal{H}$ parameterizes LSTM, for a significance level $\alpha \in$ [0,1], the intervals obtained from ICP-based conformal prediction algorithm will have error rate of at most α given as

$$\mathbb{P}(y_{t+h} \in [\hat{y}_{t+h} - \hat{\varepsilon}_h, \hat{y}_{t+h} + \hat{\varepsilon}_h]) \geq 1 - \alpha \quad \forall h \in 1, 2, .., H. \tag{11}$$

The proof follows from the conditional validity of ICP illustrated in [31]. The proper training set is used to train the LSTM model and the calibration set is used to obtain the non-conformity scores. The non-conformity score is the Euclidean distance between the actual and predicted trajectory which is given as

$$R_{t+h|t}^{(i)} = \|y_{t+h}^{(i)} - \hat{y}_{t+h|t}^{(i)}\|. \tag{12}$$

Figure 5 shows the histogram of non-conformity scores calculated on the calibration dataset.

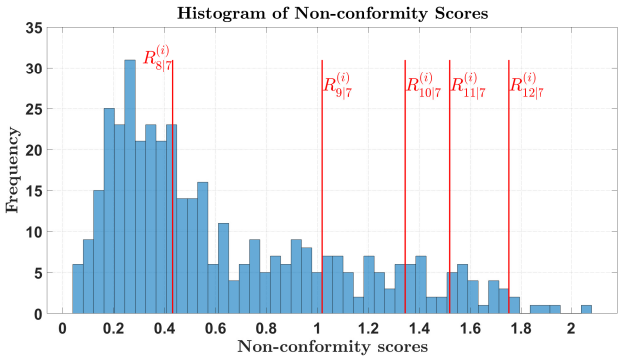

Fig. 5: Histogram of non-conformity scores

In the context of conformal prediction, the empirical distribution of nonconformity scores is leveraged to determine the critical nonconformity score $\hat{\varepsilon}$. For each future time step $t+h$ in a new pedestrian trajectory from the test set D_{test}, the model computes a prediction interval, Γ^{α}_{t+h}, which encapsulates the forecasted position $\hat{y}_{t+h}$ with an adjustment for prediction uncertainty. This interval is defined as $[\hat{y}_{t+h} - \hat{\varepsilon}, \hat{y}_{t+h} + \hat{\varepsilon}]$ for each $h \in \{1, \ldots, H\}$, where H is the prediction horizon. The inclusion of $\hat{\varepsilon}$ in the interval calculation ensures that the true position of the pedestrian is likely to fall within this range with a confidence level of $1-\alpha$. Mathematically, the prediction interval is represented as:

$$\Gamma^{\alpha}_{t+h} = [\hat{y}_{t+h} - \hat{\varepsilon}, \hat{y}_{t+h} + \hat{\varepsilon}]. \tag{13}$$

This formulation provides a quantifiable measure of the confidence in the model's predictions over the specified future time steps. Based on the non-conformity scores, for a random pedestrian from the test set, we calculate the prediction interval using Algorithm 2 and plot the prediction interval which is illustrated in Figure 6.

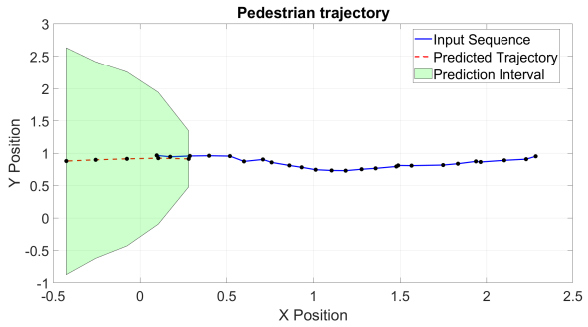

Fig. 6: Prediction interval using conformal prediction

VI. Model Predictive Control with Conformal Prediction Regions and Signal Temporal Logic

The integration of MPC with conformal prediction regions and STL enables the control of complex systems with formal guarantees and robustness to uncertainties. MPC provides an optimization-based control strategy to handle dynamic systems, while conformal prediction regions quantify uncertainty in the model's predictions. STL formalizes high-level requirements, ensuring the control system meets desired specifications. STL allows the specification of properties of dense-time, real-valued signals, and the automatic generation of monitors for testing properties on individual simulations.

The collision avoidance STL specifications are encoded as Mixed-Integer Linear Programming or MILP constraints, which are combined with MILP constraints representing the system dynamics to efficiently solve the resulting state-constrained optimization problem. The conversion of STL to MILP is achieved by using the BluSTL toolbox [32], the closed-loop framework for which is shown in Figure 7. Solving general MILPs is NP-complete and can be difficult to solve in real time. However, the advantage of using the temporal logic specification unlike natural language specification is that it can be formulated precisely, without leaving room for ambiguity. Leveraging such a formal approach makes the solution correct by construction [33], that is, the trajectories of the closed-loop system are guaranteed to satisfy the temporal logic specification. Furthermore, STL has the advantage of admitting robust semantics as defined in Appendix A which in addition to a yes/no solution on whether a specification satisfies or not, provides real numbers that illustrate the quality of satisfaction.

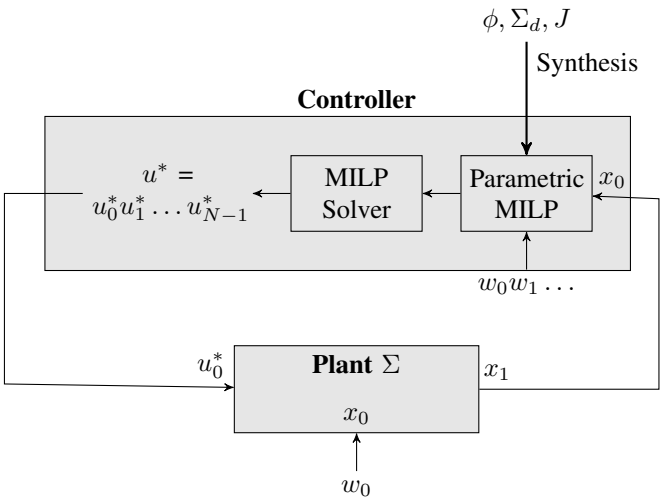

Fig. 7: Closed-loop: A parametric MILP is synthesized from the specifications, dynamics, and cost function. However, at each time step, only the first optimal input is used by the plant.

A. Constraint on System Evolution

The system constraints are designed to ensure valid finite-horizon trajectories of length N for the discrete-time system, as described by Equation (5). These constraints are applicable and hold if and only if the system's trajectories adhere to the dynamics specified in Equation (5). The constraint on system evolution not only captures the dynamics of the system but also incorporates execution tracking using binary "done" variables, done_t. These variables indicate whether a particular state x_t

and control input u_t are part of the computed history (and thus immutable) or if they are predictions (and thus subject to optimization).

For states, the dynamic constraints with "done" values can be mathematically represented as:

$$x_{t+1} = \begin{cases} x_{\text{done},t+1} & \text{if done}_t = 1, \\ A_t x_t + B_t u_t & \text{if done}_t = 0, \end{cases} \tag{14}$$

where $x_{\text{done},t+1}$ is the immutable state at time $t+1$, A_t is the state transition matrix, and B_t is the control input matrix at time t.

However, we cannot directly use conditional statements in linear programming. Instead, the big-M method, employing a large constant M, is used to formulate these as linear constraints:

$$x_{t+1} \leq x_{\text{done},t+1} + M \cdot (1 - \text{done}_t), \tag{15}$$

$$x_{t+1} \geq x_{\text{done},t+1} - M \cdot (1 - \text{done}_t), \tag{16}$$

$$x_{t+1} \leq A_t x_t + B_t u_t + M \cdot \text{done}_t, \tag{17}$$

$$x_{t+1} \geq A_t x_t + B_t u_t - M \cdot \text{done}_t. \tag{18}$$

Here, when done_t is 1 (indicating the past), the large value of M effectively deactivates the constraints involving the system dynamics, and the state x_{t+1} is forced to equal $x_{\text{done},t+1}$. When done_t is 0 (indicating the future), the constraints related to the system dynamics are activated, and x_{t+1} is calculated based on the state transition equation.

For control inputs:

$$u_t \leq u_{\text{done},t} + M \cdot (1 - \text{done}_{t-1}), \tag{19}$$

$$u_t \geq u_{\text{done},t} - M \cdot (1 - \text{done}_{t-1}). \tag{20}$$

In this case, when done_{t-1} is 1, the control input u_t is constrained to the previously determined value $u_{\text{done},t}$. If done_{t-1} is 0, the control input u_t is free to be optimized.

B. Complexity Analysis

The computational complexity in encoding Signal Temporal Logic (STL) specifications into Mixed-Integer Linear Programming (MILP) constraints is predominantly determined by the number of variables and constraints generated. This complexity is significantly influenced by the binary variables required for the logical structure of the STL formula. The complexity can be quantified as follows: the number of continuous variables introduced scales linearly with the product of the prediction horizon and the complexity of the STL formula, represented as $\mathcal{O}(N \cdot |\phi|)$. Here, N signifies the number of time steps within the prediction horizon, and $|\phi|$ denotes the length of the STL formula, quantified by the count of temporal operators it contains. The reader is guided to [34] for additional details.

VII. Case study: Simulation for collision avoidance with Conformal Prediction, MPC, and STL

In this case study, we consider the problem of controlling an autonomous vehicle operating in the presence of a potentially adversarial pedestrian whose trajectory is predicted using the trained LSTM neural network from Section IV. For the predicted pedestrian trajectory, the conformalized regions are constructed using an inductive conformal prediction algorithm as delineated in Section V.

The state-space model for the autonomous vehicle is given by a simple kinematic bicycle model. There are 4 states (x, y, θ, v) and two manipulated variables or inputs (T, δ). Here, x represents the global X position, y represents the global Y position, θ represents the heading angle, and v represents the velocity of the autonomous vehicle. T and δ represent throttle and steering angle respectively, while C_L represents the car length. However, the bicycle model represents the nonlinear dynamics of the autonomous vehicle. Therefore, to construct the linear prediction model for the ego vehicle we derive the analytical jacobians from the nonlinear state-space model. The nonlinear model that describes the dynamics of the ego car is given by

$$\begin{aligned} \dot{x} &= \cos(\theta) \cdot v \\ \dot{y} &= \sin(\theta) \cdot v \\ \dot{\theta} &= \frac{\tan(\delta)}{C_L} \cdot v \\ \dot{v} &= 0.5 \cdot T. \end{aligned}$$

A linear prediction model, derived using the analytical Jacobian around a nominal operating point characterized by the ego car traveling eastward at a steady velocity of 20 meters per second, is presented as follows:

$$\dot{x} = Ax + Bu \tag{21}$$

where

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -v\sin(\theta) & \cos(\theta) \\ 0 & 0 & v\cos(\theta) & \sin(\theta) \\ 0 & 0 & 0 & \tan(\delta)/cL \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \\ v \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & (v(\tan^2(\delta)+1))/\text{cL} \\ 0.5 & 0 \end{bmatrix} \begin{bmatrix} T \\ \delta \end{bmatrix}.$$

Assume all the states are measurable. The term θ appears in the linear model, and during the process of linearization, this term is evaluated at the nominal operating point. Given that the car travels primarily in a straight, eastward direction, the nonlinear terms $\sin(\theta)$ and $\cos(\theta)$ can be linearly approximated. Specifically, $\sin(\theta)$ can be approximated as θ and $\cos(\theta)$ as 1, assuming small deviations from the nominal operating point. Similarly, for the steering angle δ, we utilize nominal values in the linear model. This approach simplifies the model while maintaining an acceptable level of accuracy for small deviations from the nominal conditions.

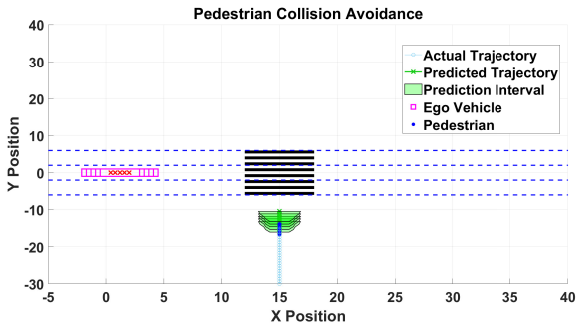

Fig. 8: Pedestrian collision 2D simulation environment

Figure 8 shows the simulation environment and illustrates the current and past positions of the ego vehicle indicated by a magenta square, and the current and past positions of the pedestrian indicated by a blue circle for the first 5 simulation steps. It also indicates conformal prediction regions for 5 steps-ahead point prediction made using LSTM neural network at each simulation time step indicated by the green trapezoidal region. The general trajectory of the pedestrian is upwards along a positive y direction. The ego vehicle approaches the upwards-moving pedestrian who is trying to cross a street with three lanes as indicated by dashed bold blue lines. The street also has a pedestrian crossing illustrated by alternate stripes of black and white.

The initial condition (x_0) for the ego vehicle is set as a vector [0, 0, 0, 0] which corresponds to the 4 initial states of the ego vehicle. For the pedestrian trajectory prediction, a novel pedestrian trajectory $\mathbf{y}_t^* \in \mathbb{R}^{2\times T}$ is considered where T represents the length of the pedestrian trajectory. We define the initial or starting position of the pedestrian by κ. κ is specified such that it does not directly begin at the pedestrian cross-walk. Instead, the initial position κ of the pedestrian starts a few units before along the y-axis, so that the pedestrian eventually crosses the pedestrian cross-walk as illustrated in Figure 8. For making the pedestrian trajectory prediction, we consider the history of the pedestrian trajectory till the initial position κ. After which the rest of the predictions are made in a close-loop autoregressive manner. This is highlighted in (Lines 3-10) of Algorithm 3. At every new simulation time step we again consider the ground truth trajectory of the pedestrian to make the next 5 steps ahead prediction along with conformalized prediction region corresponding to those predictions. This ensures the MPC algorithm which operates in a receding horizon manner remains feasible and does not become too conservative in its motion planning.

The system dynamics are discretized with a sampling time of Δt, and denoting N as the prediction horizon (in terms of the number of time steps), we set $\Delta t = 0.02$ seconds and the horizon $N = 20$ for this case study. The cost function $J \in \mathbb{R}$ for MPC evaluates the performance over the prediction horizon N based on the state trajectory $x_{[t:(t+N)]}$ and control inputs $u_{[t:(t+N)]}$. It is formulated as:

$$J(x_{t:t+N}, u_{t:t+N}) = \sum_{\tau=t}^{t+N-1} \Big[(x_{\tau+1} - x_{\mathrm{f}})^\top Q (x_{\tau+1} - x_{\mathrm{f}}) + u_\tau^\top R u_\tau \Big], \tag{22}$$

where x_{f} denotes the desired final state that the ego vehicle must reach, Q is the state weighting matrix, and R is the control input weighting matrix. The objective of MPC is to minimize this cost function, subject to the system dynamics and constraints. The optimization problem is thus formulated as:

$$\begin{aligned} \underset{u_{[t:t+N]}}{\text{minimize}} \quad & J(x_{t:t+N}, u_{t:t+N}) \\ \text{subject to} \quad & x_{\tau+1} = f(x_\tau, u_\tau), \quad \forall \tau \in \{t, \dots, t+N-1\}, \\ & x_\tau \in \mathcal{X}, \quad u_\tau \in \mathcal{U}, \quad \forall \tau \in \{t, \dots, t+N\}, \\ & x_\tau \models \phi, \qquad\qquad \forall \tau \in \{t, \dots, t+N-1\}, \end{aligned} \tag{23}$$

where f represents the system dynamics function, $\mathcal{X}$ and $\mathcal{U}$ are the feasible sets for the states and controls, respectively, and ϕ encapsulates STL safety specification.

In this case study, the Signal Temporal Logic (STL) specification aims to prevent collision between the dynamically moving pedestrian and the autonomous vehicle. To achieve this, we define a hypothetical buffer zone around the crosswalk along the x-axis. This buffer zone begins 5 units before the start of the crosswalk and extends to the end of the crosswalk. Firstly, the STL formula is specified such that the autonomous vehicle is required to reduce its speed to a safe velocity when a pedestrian is detected or predicted to be within the crosswalk. More specifically, if the pedestrian's current or forecasted position falls within the crosswalk, the ego vehicle must slow down for a specified duration of time to a safe velocity. The STL specification is specified as follows:

$$\phi_{\text{avoid}} := \Box(x_4(t) < \epsilon) \tag{24}$$

where:

- $\Box$ represents the *always* operator.
- $x_4(t)$ denotes the speed of the ego vehicle at time t.
- ϵ is the safe velocity threshold. Here, ϵ is defined to be 0.1.

In complement to the primary STL specification focused on pedestrian presence on the crosswalk, we introduce an auxiliary STL specification that accounts for prediction intervals, denoted as Γ^{α}_{t+h}. These intervals are derived for each predicted time step $h \in 1, \dots, H$ for the pedestrian, by employing Algorithm 2. The intervals are designed to maintain a coverage probability of $(1-\alpha)$, where α is a predefined confidence level set by the user. This secondary STL specification is formulated to ensure that the autonomous vehicle refrains from entering the conformalized prediction region given by Γ^{α}_{t+h}. By doing so, the vehicle is not only responsive to the real-time position of pedestrians but also respects the uncertainty bounds of their predicted future positions. The STL specification is given as

$$\phi_{\text{elude}} := \Box((x_1(t) < lb_x \vee x_1(t) > ub_x) \vee (x_2(t) < lb_y \vee x_2(t) > ub_y)) \quad (25)$$

where:

- $x_1(t)$ and $x_2(t)$ denote the states corresponding to x and y position of the ego vehicle, respectively.
- lb_x, ub_x, lb_y, and ub_y are the lower and upper bounds of the prediction intervals for the pedestrian's x and y positions.
- $\vee$ is the logical *or* operator.

This specification asserts that at every time step, the ego vehicle must not be within the predicted region where the pedestrian is expected to be, thus avoiding any possible collisions.

Algorithm 3 MPC with Conformal Prediction Regions and Signal Temporal Logic Specification

Require: Trained LSTM model $\mathcal{H}$ for H-step pedestrian trajectory predictions, novel pedestrian series $\mathbf{y}_t^* \in \mathbb{R}^{2\times T}$, significance level α, quantile values ε, ego vehicle dynamics f, initial state x_0, STL specification ϕ.

Ensure: Optimized control inputs, updated states.

1: **for** $t = 1$ **to** step **do**
2: Reset the state of the LSTM network, $\mathcal{H}$.
3: Let κ be the initial position of pedestrian taken from novel pedestrian series $\mathbf{y}_t^*$.
4: Update $\kappa = \kappa + t - 1$ for the current simulation step.
5: **Predict Trajectory:** $\mathcal{H}(\mathbf{y}^*_{1:\kappa}) \rightarrow \hat{\mathbf{y}}^*_{2:(\kappa+1)}$.
6: **Initialize First Prediction:** $\hat{\mathbf{y}}^*_{\kappa+1} = \hat{\mathbf{y}}^*_{2:(\kappa+1)}(\cdot, \text{end})$.
7: **for** $h = 2$ **to** H **do**
8: $\rho = \hat{\mathbf{y}}^*_{\kappa+1}(\cdot, h-1)$, where $\rho \in \mathbb{R}^2$ represents the 2D position of the pedestrian.
9: $\hat{\mathbf{y}}^*_{\kappa+h} = \mathcal{H}(\rho)$.
10: $\hat{\mathbf{y}}^*_{\kappa+1}(\cdot, h) = \hat{\mathbf{y}}^*_{\kappa+h}$.
11: **end for**
12: **Generate CP Intervals:** $\Gamma^{\alpha}_{\kappa+h} = [\hat{\mathbf{y}}^*_{\kappa+h} \pm \epsilon]$ for $h = 1, \ldots, H$.
13: **Specify STL specification:** $\phi = \phi_{\text{avoid}} + \phi_{\text{elude}}$
14: **Encode STL specification (ϕ) to MILP** [34]
15: Consider the optimization problem (23) with cost function J (22).
16: Apply the first input computed and update s.s model (21).
17: **end for**

Figures 9, 10, and 11 illustrate the outcomes of the state variables and control inputs, respectively, for the autonomous vehicle following the application of Algorithm 3 for pedestrian collision avoidance. A critical observation from Figure 11 that represents the control inputs over time is the behavior of the throttle input—the first control input—which registers a value of zero over a specific time duration. This period coincides with the vehicle's entry into the hypothetical buffer zone, which is strategically positioned a few units ahead of the pedestrian crosswalk.

During this phase, the vehicle's trajectory, particularly the x position (the first state), undergoes a notable transition from an exponential increase to a linear progression. This change is directly attributed to the enforcement of the STL specification, which is designed to decelerate the vehicle within the buffer zone. Concurrently the STL specification also enforces the criterion that the ego vehicle should stay outside the conformalized prediction region. Thus, the implementation of ϕ effectively prevents the vehicle from encroaching into the conformalized prediction region and also enforces the criterion that the ego vehicle should maintain a safe velocity within the vicinity of the pedestrian.

Figure 12 depicts the movement of an autonomous vehicle (ego vehicle) and a moving pedestrian, over time. The red points represent the position of the ego vehicle at various moments. The blue points indicate the obstacle's current position, while the green 'x' marks denote the predicted future positions of the obstacle. The filled black lines are meant to illustrate the prediction intervals, providing a visual margin of where the pedestrian might be with a certain confidence level. The dashed blue lines symbolize lanes, and the visualization's 3D nature—with time as the vertical axis—provides a dynamic sense of how the vehicle and obstacle interact and move relative to each other and the prediction intervals over time. The simulation results demonstrate the autonomous vehicle's proficiency in avoiding not only the actual, dynamically moving pedestrian but also the projected positions of the pedestrian with the conformalized prediction region. These predicted positions, encapsulating a range of potential future locations of the pedestrian, are effectively circumnavigated by the vehicle, underscoring the application of Algorithm 3 in maintaining safety. The vehicle's adaptive response to both real-time and anticipated pedestrian movements highlights the efficacy of the implemented control strategy in achieving collision avoidance while adhering to the predetermined navigational objectives.

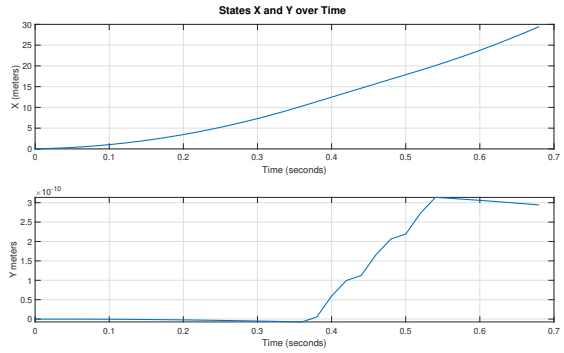

Fig. 9: States 1 and 2 over time

VIII. Conclusion and Recommendations

We posited a Model Predictive Controller (MPC) that is subjected to collision avoidance Signal Temporal Logic (STL) specification which embeds conformal prediction regions for probabilistic safe planning in dynamic environments. To predict the complex trajectories of pedestrians we used long short-term memory networks. However, we do not impose this choice, in practice, any underlying trajectory predictor

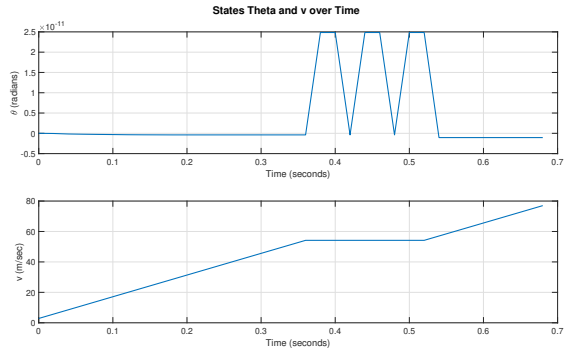

Fig. 10: States 3 and 4 over time

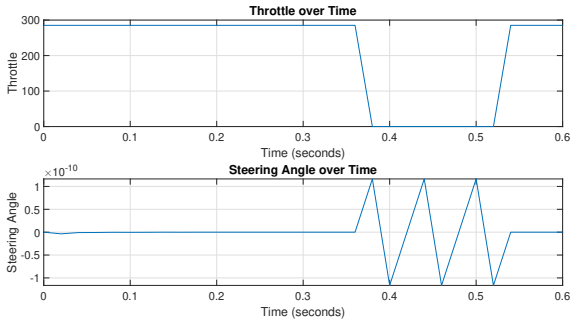

Fig. 11: Control inputs over time

can be used for pedestrian trajectory forecasting and the inductive conformal prediction algorithm would give valid prediction regions that quantify uncertainties. To the best of our knowledge, these are the first results that provide valid safety guarantees for motion planning in a dynamic environment with no assumptions on the predictor and use STL specification as a constraint within the motion planning algorithm. To validate our framework of safe motion planning, we provided simulation results that consider a self-driving vehicle that safely navigates under the presence of a dynamically moving pedestrian.

For future work, we plan to consider a more haphazard trajectory for the pedestrian and increase the number of pedestrians under which the self-driving car operates. Additionally, we plan to do a comparative study using different trajectory predictors and analyze how the results are altered or improved. Another interesting avenue for future work would be to consider the pedestrian trajectory and the conformalized prediction regions to be a part of the state space of the self-driving vehicle.

APPENDIX

A. Quantitative semantics

Quantitative or robust semantics for STL are defined by providing a real-valued function ρ^{ϕ} of signal $\mathbf{x}$ and time t such that $\rho^{\phi}(\mathbf{x}, t) > 0 \implies (\mathbf{x}, t) \vDash \phi$. It is defined recursively as

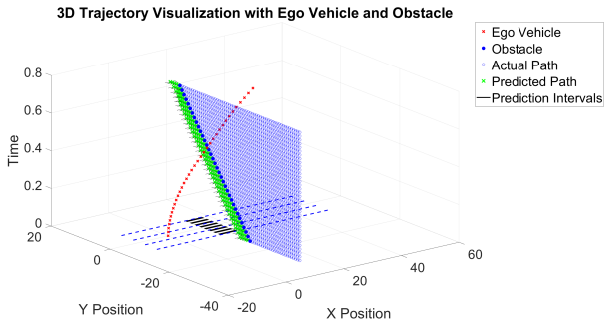

Fig. 12: 3D visualization of simulation for collision avoidance

follows:

$$\begin{aligned}
\rho^{\pi^{\mu}}(\mathbf{x}, t_k) &= \mu(x_k, y_k, u_k, w_k) \\
\rho^{\neg\psi}(\mathbf{x}, t_k) &= -\rho^{\psi}(\xi, t_k) \\
\rho^{\phi_1 \wedge \phi_2}(\mathbf{x}, t_k) &= \min(\rho^{\phi_1}(\xi, t_k), \rho^{\phi_2}(\xi, t_k)) \\
\rho^{\phi_1 \vee \phi_2}(\mathbf{x}, t_k) &= \max(\rho^{\phi_1}(\xi, t_k), \rho^{\phi_2}(\xi, t_k)) \\
\rho^{\Diamond_{[a,b]}\psi}(\mathbf{x}, t_k) &= \max_{t_{k'} \in [t+a, t+b]}(\rho^{\psi_1}(\xi, t_{k'})), \\
\rho^{\phi_1 \mathcal{U}_{[a,b]} \psi \phi_2}(\mathbf{x}, t_k) &= \max_{t_{k'} \in [t+a, t+b]}(\min(\rho^{\phi_2}(\mathbf{x}, t_{k'}), \\
&\qquad \min_{t_{k''} \in [t_k, t_{k'}]} \rho^{\phi_2}(\mathbf{x}, t_{k''}))).
\end{aligned} \tag{26}$$

For simplicity, we denote $\rho^{\pi^{\mu}}$ as ρ^{μ} for the remainder of this paper. The robustness of satisfaction for an arbitrary STL formula is computed recursively from the above semantics by propagating the values of the functions associated with each operand using min and max operators corresponding to the various STL operators. The robust satisfaction of $\mu_1 \wedge \mu_2$ is the minimum of ρ^{μ_1} and ρ^{μ_2}. Temporal operators are considered as conjunctions and disjunction along time axis: since we deal with discrete time, the robustness of satisfaction of $\phi = \Box_{[0,2]}\mu_1$ is

$$\rho^{\phi}(x, t) = \min_{t_k \in \{0,1\}} \rho^{\mu_1}(x, t_k) = \min\{(x_0 - 3, x_1 - 3, ..., x_k - 3)\} \tag{27}$$

where $0 \leq t_0 < t_1 < \cdots < t_k \leq 2 < t_{k+1}$.

The absolute value of robustness is interpreted as the distance of signal $\mathbf{x}$ from the set of trajectories violating or satisfying ϕ, in the space of projections with respect to the function μ that defines the predicates of ϕ.

B. Quantitaive Encoding

The advantage of using robust encoding is that it allows us to maximize the value of r_0^{ϕ}, which results in a trajectory that maximizes the robustness of satisfaction. Furthermore, it allows STL constraints to be softened or hardened as required. The downside is that robustness-based encoding is more expensive to compute because of additional binary variables that are introduced during boolean operation. Additionally, incorporating robustness as an objective makes the cost function non-convex thereby making the existence of many local minima possible which makes it difficult to optimize. But, the robustness constraints can also be relaxed which allows for a simpler cost function thereby making the

problem more tractable. Additionally, the temporal constraints such as always, eventually, and until which are fundamental to any STL specification are also specified in the appendix. These are borrowed from the work of Alexandre Donze et al [34]. The boolean operations of negation, conjunction, and disjunction are defined as follows:

1) Negation

$$r_t^{\psi} = \neg r_t^{\varphi} \quad \text{or equivalently} \quad r_t^{\psi} = -r_t^{\varphi} \tag{28}$$

Negation is the logical NOT operation. If r_t^{ϕ} is the variable representing the truth of a formula ϕ at time t, then negation is represented by r_t^{ψ}, which will have the opposite truth value. In MILP, this is implemented by negating the value of r_t^{ϕ}, so if r_t^{ϕ} = 1(true), then r_t^{ψ} = -1 (false).

2) Conjunction

$$r_t^{\psi} = \bigwedge_{i=1}^{m} r_t^{\varphi_i} \tag{29}$$

$$\sum_{i=1}^{m} p_{t_i}^{\psi} = 1 \tag{30}$$

$$r_t^{\psi} \leq r_t^{\varphi_i}, \quad \text{for } i = 1, \ldots, m \tag{31}$$

$$r_t^{\varphi_i} - (1 - p_{t_i}^{\psi})M \leq r_t^{\psi} \leq r_t^{\varphi_i} + (1 - p_{t_i}^{\psi})M \tag{32}$$

Conjunction is the logical AND operation. It requires that all operands ϕ_i to be true for ψ to be true. The MILP encoding introduces a binary helper variable $p_{t_i}^{\psi}$ for each operand to ensure that r_t^{ψ} is the minimum of all $r_t^{\phi_i}$. Equation 30 ensures that exactly one $p_{t_i}^{\psi}$ is true. Equation 31 ensures that r_t^{ψ} is less than $r_t^{\varphi_i}$. Equation 32 ensures that $r_t^{\varphi_i}$ is equal to r_t^{ψ}, if and only if $p_{t_i}^{\psi}$ = 1. The constant M in the equation is used as a part of the "big-M" method in MILP formulations to enforce logic conditions and allows $p_{t_i}^{\psi}$ as a switch without explicitly using non-linear logical operators.

3) Disjunction

$$r_t^{\psi} = \bigvee_{i=1}^{m} r_t^{\varphi_i} \tag{33}$$

$$\sum_{i=1}^{m} p_{t_i}^{\psi} = 1 \tag{34}$$

$$r_t^{\psi} \geq r_t^{\varphi_i}, \quad \text{for } i = 1, \ldots, m \tag{35}$$

$$r_t^{\varphi_i} - (1 - p_{t_i}^{\psi})M \leq r_t^{\psi} \leq r_t^{\varphi_i} + (1 - p_{t_i}^{\psi})M \tag{36}$$

Disjunction is the logical OR operation. It requires that at least one operand ϕ_i be true for ψ to be true. The MILP encoding for disjunction is similar to conjunction but ensures that r_t^{ψ} is the maximum of the $r_t^{\varphi_i}$. This is implemented in MILP by using constraints similar to those for conjunction but designed to ensure that r_t^{ψ} takes the value of the largest $r_t^{\varphi_i}$ that is true.

1) Temporal constraints:

1) Always

$$\psi = \square_{[a,b]}\varphi \tag{37}$$

$$\text{Let } a_N^t = \min(t+a, N) \text{ and } b_N^t = \min(t+b, N) \tag{38}$$

$$r_t^{\psi} = \bigwedge_{i=a_t^N}^{b_t^N} r_i^{\varphi} \wedge \Big(\bigvee_{j=1}^{N} l_j \wedge \bigwedge_{i=\hat{a}_t^N}^{\hat{b}_t^N} r_t^{\phi}\Big) \tag{39}$$

This definition means that for a given formula ϕ, the "Always" operator $\square_{[a,b]}$ ensures that ϕ holds at every time step in the interval from a to b, inclusive, relative to time step t.

2) Eventually

$$\psi = \Diamond_{[a,b]}\varphi \tag{40}$$

$$r_t^{\Diamond_{[a,b]}\varphi} = \bigvee_{i=a_N^t}^{b_N^t} r_i^{\varphi} \wedge \Big(\bigvee_{j=1}^{N} l_j \wedge \bigvee_{i=\hat{a}_t^N}^{\hat{b}_t^N} r_i^{\phi}\Big) \tag{41}$$

The "Eventually" operator asserts that ϕ is true at some point within the interval from a to b, relative to time step t.

3) Until

$$\psi = \varphi_1 \, U_{[a,b]} \, \varphi_2 \tag{42}$$

$$\varphi_1 \, U_{[a,b]} \, \varphi_2 = \square_{[0,a]}\varphi_1 \wedge \Diamond_{[a,b]}\varphi_2 \wedge \square_{[a,a]}(\varphi_1 \, U \, \varphi_2) \tag{43}$$

REFERENCES

[1] Standing general order on crash reporting. https://www.nhtsa.gov/laws-regulations/standing-general-order-crash-reporting. Accessed: 2023-07-11.

[2] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, 1997.

[3] Michael Everett, Yu Fan Chen, and Jonathan P How. Collision avoidance in pedestrian-rich environments with deep reinforcement learning. *IEEE Access*, 9:10357–10377, 2021.

[4] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] Alec Farid, Sushant Veer, Boris Ivanovic, Karen Leung, and Marco Pavone. Task-relevant failure detection for trajectory predictors in autonomous vehicles. In *Conference on Robot Learning*, pages 1959–1969. PMLR, 2023.

[7] Rachel Luo, Shengjia Zhao, Jonathan Kuck, Boris Ivanovic, Silvio Savarese, Edward Schmerling, and Marco Pavone. Sample-efficient safety assurances using conformal prediction. In *Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics*, pages 149–169. Springer, 2022.

[8] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

[9] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

[10] Yaniv Romano, Evan Patterson, and Emmanuel Candes. Conformalized quantile regression. *Advances in neural information processing systems*, 32, 2019.

[11] Nasir Uddin and Tuwe Lofstrom. Applications of conformal regression on real-world industrial use cases using crepes and mapie. In *Conformal and Probabilistic Prediction with Applications*, pages 147–165. PMLR, 2023.

[12] Lars Lindemann, Matthew Cleaveland, Gihyun Shim, and George J Pappas. Safe planning in dynamic environments using conformal prediction. *arXiv preprint arXiv:2210.10254*, 2022.

[13] Yuxiao Chen, Ugo Rosolia, Chuchu Fan, Aaron Ames, and Richard Murray. Reactive motion planning with probabilisticsafety guarantees. In *Conference on Robot Learning*, pages 1958–1970. PMLR, 2021.

[14] Mike Phillips and Maxim Likhachev. Sipp: Safe interval path planning for dynamic environments. In *2011 IEEE international conference on robotics and automation*, pages 5628–5635. IEEE, 2011.

[15] Venkatraman Renganathan, Sleiman Safaoui, Aadi Kothari, Benjamin Gravell, Iman Shames, and Tyler Summers. Risk bounded nonlinear robot motion planning with integrated perception & control. *Artificial Intelligence*, 314:103812, 2023.

[16] Georges S Aoude, Brandon D Luders, Joshua M Joseph, Nicholas Roy, and Jonathan P How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35:51–76, 2013.

[17] Antony Thomas, Fulvio Mastrogiovanni, and Marco Baglietto. Probabilistic collision constraint for motion planning in dynamic environments. In *Intelligent Autonomous Systems 16: Proceedings of the 16th International Conference IAS-16*, pages 141–154. Springer, 2022.

[18] Richard M Murray, John Hauser, Ali Jadbabaie, Mark B Milam, Nicolas Petit, William B Dunbar, and Ryan Franz. Online control customization via optimization-based control. *Software-Enabled Control, Information technology for dynamical systems*, pages 149–174, 2003.

[19] Peter Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: the case for cooperation. In *2013 IEEE international conference on robotics and automation*, pages 2153–2160. IEEE, 2013.

[20] Jaime F Fisac, Andrea Bajcsy, Sylvia L Herbert, David Fridovich-Keil, Steven Wang, Claire J Tomlin, and Anca D Dragan. Probabilistically safe robot planning with confidence-based human predictions. *arXiv preprint arXiv:1806.00109*, 2018.

[21] David Fridovich-Keil, Andrea Bajcsy, Jaime F Fisac, Sylvia L Herbert, Steven Wang, Anca D Dragan, and Claire J Tomlin. Confidence-aware motion prediction for real-time collision avoidance1. *The International Journal of Robotics Research*, 39(2-3):250–265, 2020.

[22] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 1672–1678. IEEE, 2018.

[23] Bruno De Finetti. *Theory of probability: A critical introductory treatment*, volume 6. John Wiley & Sons, 2017.

[24] Harris Papadopoulos and Haris Haralambous. Reliable prediction intervals with regression neural networks. *Neural Networks*, 24(8):842–851, 2011.

[25] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings 13*, pages 345–356. Springer, 2002.

[26] Vasumathi Raman, Alexandre Donzé, Dorsa Sadigh, Richard M Murray, and Sanjit A Seshia. Reactive synthesis from signal temporal logic specifications. In *Proceedings of the 18th international conference on hybrid systems: Computation and control*, pages 239–248, 2015.

[27] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th international conference on computer vision*, pages 261–268. IEEE, 2009.

[28] Victor M Guerrero. Time-series analysis supported by power transformations. *Journal of forecasting*, 12(1):37–48, 1993.

[29] Weicong Kong, Zhao Yang Dong, Youwei Jia, David J Hill, Yan Xu, and Yuan Zhang. Short-term residential load forecasting based on lstm recurrent neural network. *IEEE transactions on smart grid*, 10(1):841–851, 2017.

[30] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 1394–1401. IEEE, 2018.

[31] Vladimir Vovk. Conditional validity of inductive conformal predictors. In *Asian conference on machine learning*, pages 475–490. PMLR, 2012.

[32] Alexandre Donzé, Vasumathi Raman, G Frehse, and M Althoff. Blustl: Controller synthesis from signal temporal logic specifications. *ARCH@ CPSWeek*, 34:160–8, 2015.

[33] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57(11):2817–2830, 2012.

[34] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, pages 81–87. IEEE, 2014.