END-TO-END, REAL TIME AND ROBUST BEHAVIORAL PREDICTION MODULE WITH ROS FOR AUTONOMOUS VEHICLES

Tolga KAYIN[1] and Çağatay Berke ERDAŞ[1]

[1]Department of Computer Engineering, Başkent University, Ankara, TÜRKİYE

ABSTRACT. In the world where urbanization and population density are increasing, transportation methods are also diversifying and the use of unmanned vehicles is becoming widespread. In order for unmanned vehicles to perform their tasks autonomously, they need to be able to perceive their own position, the environment and predict the possible movements/routes of environmental factors, similar to living things. In autonomous vehicles, it is extremely important for the safety of the vehicle and the surrounding factors to be able to predict the future position of the objects around it with high performance so that the vehicle can plan correctly. Due to the stated reasons, the behavioral prediction module is a very important component for autonomous vehicles, especially in moving environments. In this study, fast and successful robotic behavioral prediction module has been developed to enable the autonomous vehicle to plan more safely and successfully.

1. INTRODUCTION

The function and importance of autonomous vehicles are increasing day by day. It is foreseen that autonomous vehicles will play an important role in the future in order to reduce the density in transportation and to eliminate human-induced accidents. Apart from transportation, autonomous vehicles are becoming more and more common in areas such as agriculture, health and education.

Autonomous vehicles, inspired by living things; It consists of modules such as perception to detect the environment, localization to determine its own position, planning to where and how to go, control for its movement and behavioral prediction for possible movement routes of surrounding objects. Middleware such as Robotic Operating System (ROS) [1], ZeroMQ (ZMQ) [2], Robotic Operating System 2 (ROS2) [3] are needed for these modules to communicate with each other correctly and completely. These middlewares enable modules to transmit the desired message to the relevant module. Thanks to the ROS middleware tools that is used in the study, it also provides benefits such as visualizing, recording and observing data.

The behavioral prediction module is the one of the most important factor for the accurate result of the planning module in autonomous vehicles. The behavioral

prediction module generates output that predicts future positions by keeping the past positions of objects around the ego vehicle. This output creates an input to the planning module by combining with the objects found by the detection module. An autonomous vehicle without a behavioral prediction module will consider all objects as static and plan accordingly, but in highway conditions or urban traffic scenarios, an accident will be inevitable if the possible routes of vehicles or pedestrians are not taken into account. To give an example from scenarios that are frequently experienced in daily life, in order for an autonomous vehicle to consider a pedestrian preparing to cross the street, the autonomous vehicle must know the pedestrian's possible route. Similarly, while the autonomous vehicle is changing lanes, it must calculate the possible route according to the speed of the vehicle from behind, otherwise there will most likely be an accident.

The developed behavioral prediction module is based on ROS and works in real time. Features such as ROS middleware, dynamic history hold and release structure, direction error correction, covariance distribution visualization, and message type matching suitable for planning have been added to the multi modal CVAE-based model [4].Thus, an end-to-end autonomy module structure was created that will send the possible routes of the surrounding vehicles to the planning module.

In the next part of the study, first of all, behavioral prediction approaches and studies in the literature will be summarized, then information about the methodology used in the study will be given and the developed module will be explained in detail. Afterwards, the results obtained with the test data will be shared. Finally, the result of the study will be expressed and suggestions for future studies will be shared.

2. MATERIALS AND METHOD

2.1. **Related Methods.** There is less research in the field of behavioral prediction compared to areas such as perception, localization, and planning of autonomous vehicles. The biggest reason for this is that it is more difficult to determine the location of environmental factors in the future than the problems in other areas. When the trajectory prediction approaches are examined [5] [6], although there are approaches such as representation, output types, modeling, situational awareness, the modeling approach will be used as the main approach in categorizing the studies in this article. In addition, information will be provided in terms of representation, output and situational awareness types for the studies. When the studies are examined in terms of modeling methods; Behavioral prediction methods as shown in Figure 1; it consists of physics-based, machine learning-based, deep learning-based and reinforcement learning-based methods.

Physics-based methods take information from the dynamics and kinematics of the vehicle. It consists of Single Trajectory, Kalman Filtering, Monte Carlo Methods.

Models using single trajectory mostly use the kinematic information of the vehicle. Although there are also models that use the dynamic information of the vehicle, these models are more complex. Dynamic models consider all forces that govern motion. Dynamic models are highly complex due to the factors involved. For example, for a vehicle, the dynamic model considers the forces acting on the tires, the driver's actions and their effects on the vehicle's engine and transmission. For trajectory prediction, it doesn't make much sense to use a dynamic model to model such complex behavior unless you intend to run a control-oriented application [7]. Kinematic models are more commonly used than dynamic models due to their simpler structure. One of the most commonly used is Constant Velocity (CV). A simple example of a kinematic model is the CV model used in [8]. The CV model assumes that the recent relative motion of an object determines its future trajectory. Similarly Ammoun et al. [9] and Schubert et al. [10] They estimated the possible trajectories of the vehicle using the Constant Acceleration (CA) method. The CA method estimates the future acceleration of the vehicle from the past acceleration data, these acceleration estimates are converted into position information and the possible position of the vehicle is found. Lytrivis et al. [11] using Constant Turn Rate and Velocity (CTRL) and Constant TurnRate & Acceleration (CTRA) with a similar approach, Batz et al. [12] Constant SteeringAngle & Velocity (CSAV) and Constant SteeringAngle & Acceleration (CSAA) models were used by adding wheel data to the model.
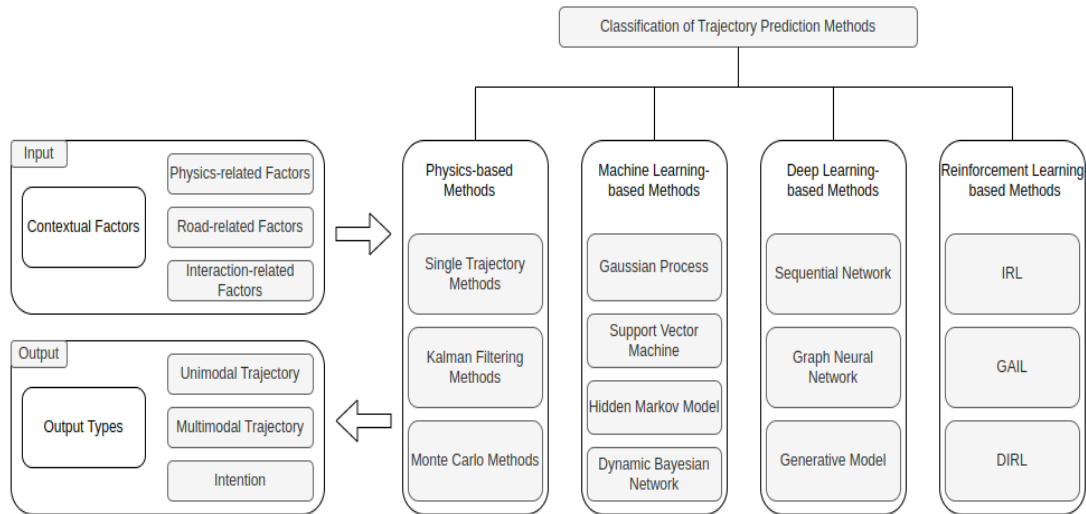

FIGURE 1. Trajectory prediction methods.

Unlike physics-based methods, machine learning methods are based on the principle of obtaining predicted trajectory by data mining. In comparison to physics-based models which are limited to low-level properties of motion and cannot estimate well the long term dependencies in motion, the learning-based models on the other hand tend to capture and incorporate long term dependencies and changes caused by external factors. The most widely used main machine learning methods are Gaussian Process (GP), Support Vector Machine (SVM), Hidden Markov Model (HMM), Dynamic Bayesian Network (DBN), K-Nearest Neighbors (KNN), Decision Tree methods. Since SVM can output the characteristics of classification probability, Kumar et al. [13] propose a layered architecture method combining SVM and Bayesian filtering to identify lane-changing maneuvers to obtain more accurate identification results

In real life, it could only be observed the tangible states that are visible on surface, but we cannot intuitively express the hidden states. Thus, it is needed to develop a Markov process involving hidden states and find the intrinsic state of an event by the set of observable states related to the probability of the hidden state. This is the so-called hidden Markov model. Based on HMM, Qioa et al. [14] presents an algorithm named HMTP* that adaptively chooses parameters to imitate real scenery with dynamically changing speeds. In [15], her HMM connection with fuzzy logic is applied to predict driver maneuvers. The author of [16] presents his DBN representing driver behavior and vehicle trajectory. DBNs have Markov properties. We can extend the state with more information to satisfy the Markov assumption. In [16] this is done by adding all relevant information of the process to the DBN in the form of a vector.

Although the outputs of the studies in this method are mostly multimodal, it has been observed that the model's performance increases as the situational awareness states such as map-aware, scene aware, interaction aware increase.

Deep-learning based methods consist of sequential network, graph neural network (GNN) and Generative Model methods. Sequential network methods consist of Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), RNN and CNN and Attention Mechanism, while Generative Model methods consist of Generative Adversarial Network and Conditional Variational Auto Encoder methods.

One of the most popular study with RNN & CNN is DESIRE [17], whose goal is to predict the future positions of multiple interacting agents in a dynamic (driving) scene. This takes into account the multimodal nature of future projections. For example even in the same situation, the future can be different. It can predict potential future outcomes and make strategic predictions based on them, making inferences based not only on past movement history, but also on scene context and agent interactions.

An example of work with sequential network is [18] a modified version of LSTM i.e. ST-LSTM (Spatio-temporal LSTM) is used in [18] where the interaction of

multiple vehicles and its effect on trajectory of Value of Information (VOI) is estimated.

Deep learning-based studies can provide more comprehensive output and input compared to physics and machine learning-based studies. These studies mostly take interaction-aware inputs and provide multimodal or intention type output.

The reinforcement learning approach, which has been extensively studied in recent years, also appears in predicting trajectory. Reinforcement learning method is based on the decision-reward principle, focusing on finding the decision that will maximize the reward.

According to studies using these approaches; In Sun et al. [19] study, interaction related elements are taken into account to achieve probabilistic estimation for AVs by using IR. Future trajectory distribution is defined by driving manoeuvers. Kufler et al. [20] Extending GAIL to his RNN optimization to show the behavior of a human driver, discriminators evaluate steps and actions. Choi et al. [21] combine the partially observable Markov decision process (POMDP) within the GAIL framework and propose a method to train the model using the discriminator reward function. The prediction problem is nonlinear, so nonlinear mapping should be used for generalizable function approximation. Wulfmeier et al. [22] propose a deep inverse reinforcement learning (DIRL) framework for approximating complex nonlinear reward functions. Some D-IRL approaches get history tracks as input. Considering driving characteristics and route shape, the authors [23] initially practiced RL to develop MDP, then learned the optimum driving procedure from IRL, and used deep neural network (DNN) to generate a reward function. In Jung et al., [24] this work proposes a convolutional LSTM to extract feature maps from his LIDAR and trajectory data considering inertia, environment, and society. This feature map is integrated with output reward map to forecast the traversability map.

Reinforcement learning methods, similar to deep learning methods, can take extensive inputs such as road and scene related factors and provide comprehensive outputs in the form of unimodal and intention.

2.2. **Datasets.** Datasets are required for training or testing the above mentioned methods or models. These datasets consist of various sensor data, map data and their annotations. The main ones are KITTI [25], nuScenes [26], Argoverse [27] and NGSIM [28] datasets. The model that is worked on in this article, trained on nuScenes dataset. NuScenes dataset is a large-scale autonomous driving dataset which has several distinct dataset such as nuPlan [29] for planning, nuScenes for perception, nuImages [30] for image level operations. In the following sections, comparisons of various studies on these data sets will be given.

2.3. **Evaluation Metrics.** Reliable and generic metrics are needed to measure the success of the studies. Some metrics that can be used to compare related works are given below.

Root Mean Squared Error (RMSE): "RMSE computes the square root of the mean squared forecast error" ([6], p.14). As shown in the equation (1) while $\hat{y}_i$ stands for estimated value (m), y_i indicates observed value and n is the number of samples.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{1}$$

Average displacement error (ADE): "The average distance between the predicted trajectory and the ground truth" ([6], p.14). In the formulas (2) given below, x_i and y_i stand for predicted trajectory for one second interval in meters at x and y axes respectively, x_i^{GT} and y_i^{GT} indicate observed trajectory for one second interval in meters at x and y axes respectively, and T is time in seconds.

$$ADE = \frac{1}{T}\sum_{t=1}^{T} \sqrt{(xi - x_i^{GT})^2 + (y_i - y_i^{GT})^2} \tag{2}$$

Final displacement error (FDE): "The distance between the final prediction results and the corresponding ground truth location" ([6], p.14). In the equation (3) given below, x_T and y_T stand for predicted trajectory for one second interval in meters at x and y axes respectively, x_T^{GT} and y_T^{GT} indicate observed trajectory for one second interval in meters at x and y axes respectively, and T is time in seconds.

$$FDE = \sqrt{(xT - x_T^{GT})^2 + (y_T - y_T^{GT})^2} \tag{3}$$

Miss Rate (MR): "Based on the L2 distance of the final positions, the ratio of cases where the estimated trajectory isn't within 2.0 meters of the ground truth" ([6], p.14). In the equation (4) given below, 'misses' is the predicted trajectory not within 2.0 meters of the ground truth and 'hits' is the predicted trajectory within 2.0 meters of the ground truth.

$$MR = \frac{misses}{hits+misses} \tag{4}$$

When the Tables 1 and 2 are examined, although physics-based and machine-learning-based methods require low computational load, their performance decreases as the estimated time (2s>) increases. Compared to these two methods, deep learning and reinforcement-based learning methods can predict longer time successfully, although they overlay more computation load.

When deep learning and reinforcement-based methods are compared, it is seen that the deep learning-based method is more successful. An open source, ROS structured, end to end, configurable, robust, multi-class behavioral prediction module could not be found. The most related work found [31] is one that predicts pedestrian-only trajectories with ROS.

TABLE 1. Comparison of the trajectory prediction RMSE results of models using various methods trained on NGSIM dataset under highway condition.

Classification Methods	Models	RMSE(m)				
		1s	2s	3s	4s	5s
Single Trajectory	Constant Velocity[32]	0.73	1.78	3.13	4.78	6.68
Kalman Filtering	IMM-KF[33]	0.58	1.36	2.28	3.37	4.55
HMM	C-VGMM+VIM[34]	0.66	1.56	2.75	4.24	5.99
RNN	M-LSTM[35]	0.58	1.26	2.12	3.24	4.66
RNN	MFP-1[36]	0.54	1.16	1.90	2.78	3.83
CNN and RNN	CS-LSTM(M)[37]	0.62	1.29	2.13	3.20	4.52
Attention Mechanism	MHA-LSTM[38]	0.41	1.01	1.74	2.67	3.83
GNN	GRIP++[39]	0.38	0.89	1.45	**2.14**	**2.94**
GNN	GISNet[40]	**0.33**	**0.83**	**1.42**	2.14	3.23
Generative Model	MATF-GAN[41]	0.66	1.34	2.08	2.97	4.13
Generative Model	TS-GAN[42]	0.60	1.24	1.95	2.78	3.72
IRL	L-IRL[43]	1.12	2.29	2.31	3.38	4.45
GAIL	GAIL-GRU[44]	0.69	1.51	2.55	3.65	4.71
DIRL	MEDIRL[45]	1.35	2.57	2.83	3.69	4.88
DIRL	DN-IRL[46]	0.54	1.02	1.91	2.43	3.76

TABLE 2. Comparison of the trajectory prediction FDE, ADE and MR results of models using various methods trained in Argoverse data set under urban condition.

Classification Methods	Models	$K^1=6$			$K^1=1$		
		minFDE	minADE	MR	minFDE	minADE	MR
Physics-based	CV[47]	7.57	3.39	0.82	7.89	3.53	0.84
Machine Learning-based	NN+map[47]	4.03	2.08	0.58	8.12	3.65	0.84
RNN	LSTM+map[47]	5.44	2.34	0.69	6.81	2.96	0.81
RNN	Jean[48]	1.49	0.93	0.19	4.18	1.86	0.63
Attention Mechanism	SceneTransformer[49]	**1.23**	**0.80**	0.13	-	-	-
Attention Mechanism	mmTransformer[50]	1.34	0.84	0.15	-	-	-
GNN	LaneGCN[51]	1.36	0.87	0.16	3.78	1.71	0.59
GNN	DenseTNT[52]	1.45	0.93	**0.11**	-	-	-
GNN	LaneRCNN[53]	1.45	0.90	0.12	**3.69**	**1.69**	**0.57**
Generative Model	PRIME[54]	1.56	1.22	0.12	3.82	1.91	0.59

[1] output trajectory numbers

3. SOFTWARE DESCRIPTION

Trajectron++ has been used as the model in this article for the following reasons;
- There are many studies that show that graph-structured recurrent models are more successful,
- As it is stated in the literature review section it has a multi-classification structure,
- It is trained on a new and comprehensive data set for vehicle route estimation (nuScenes),
- It outputs both unimodal and intentional and is a scene aware model,

3.1. **Software Architecture.** Trajectron++ is in a map and interaction aware structure as shown in Figure 2, Studies on this model;

ROS middleware has been added, the data from the perception module has been adapted to the input data format in the model. In addition, a dynamic history hold/drop structure has been created, so the history of tracked objects is accumulated, and the untracked object is prevented from entering the model.

3.2. **Software Functionalities.** Due to the added ROS middle-ware, this software processes the incoming data from the perception subsystem, finds possible trajectories and sends them to the planning subsystem. In this way, the vehicle also considers the trajectories of its moving objects while planning.
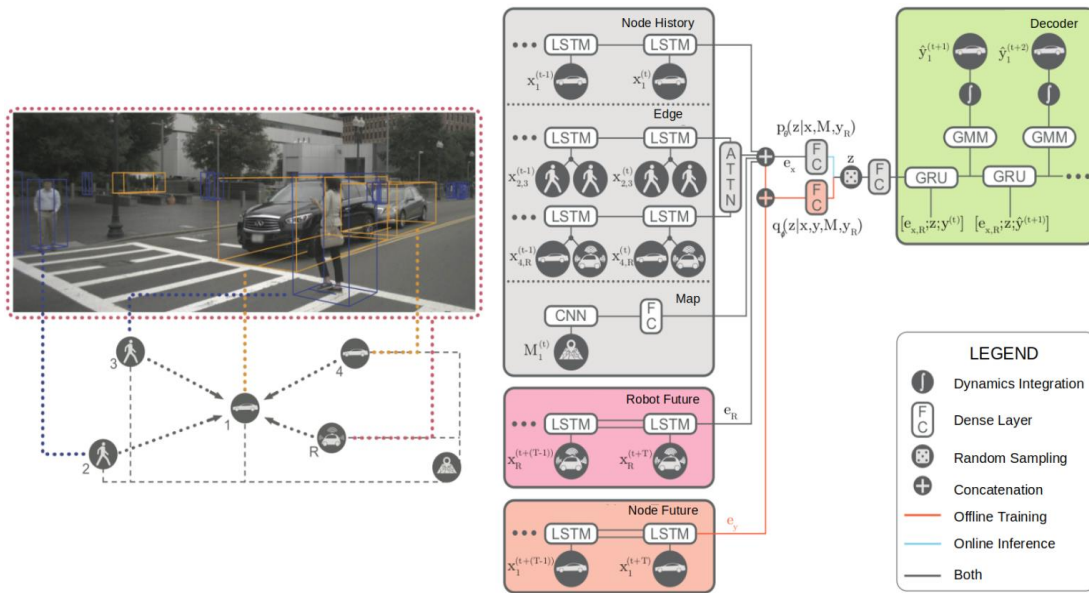

FIGURE 2. Trajectron++ model architecture.

3.3. **Sample Code Snippets Analysis.** The developed Behavioral prediction module is shown in Figure 3 as a pseudo code. First of all, by listening to the output of the perception module, information such as the position, class and orientation of the surrounding objects is obtained, and then dictionaries are created for the object at a certain speed and attention radius. As long as the ROS connection is open, the object information from the perception output is accumulated in the relevant dictionaries. When the objects reach enough history, they are converted into a data structure suitable

for the model and entered as an input to the model. 6-second predicted trajectories are output from the model. This estimated trajectory information is converted into the message types for planning module and visualization. These messages are published to relevant modules by ROS middleware.

If the functions are examined in more detail, with the append history function shown in Figure 4, the heading, position and classification information of the tracked objects are passed by the distance and speed filters, and the history is appended in the related python dictionaries. With the function shown in Figure 5, the untracked object is deleted from the dictionary and only vehicle and pedestrian type entries are entered into the model.

Heading data entered as input to the model is obtained from both the direction information coming from the segmentation output and the direction information found from the vehicle position as shown in Figure 4. Although the segmentation output gives the right direction of the vehicle, it often reverses the direction by 180 degrees. While heading from the position information, the heading value is incorrect, especially when the vehicle is maneuvering. If the heading value from the position information and the value from the segmentation are more than 90 degrees, the direction information from the segmentation is rotated 180 degrees, thus a more robust heading information structure is created. In addition, a velocity and attention radius filter has been created for the tracked objects, and objects that are far from the ego vehicle or at very low speeds from the detection system do not enter the model, so that the algorithm works more efficiently. The predicted trajectory information from the model was converted into a message type suitable for the planning module and visualized. A configuration file was created as in Figure 6 in order to easily understand and configure Rostopic and message types.

```
print("Append object information and history hold/release structure")

subscribe perception output

initialize history dictionary

while ROS is UP

    if object is in attention radius and has speed

        Append object id,x,y,heading,yaw in dictionary

    if object is in dictionary

        Append object id, x, y, heading, yaw in dictionary
    else
        delete object dictionary

    if object has 20 history timestep

        filter heading data
        convert history dictionary to model input
        input object history to model

    create trajectory message from model output
    create visualization marker from model output

    publish trajectory message
    publish visualization message
```

FIGURE 3. Pseudo code of behavioral prediction module.

```python
def appendTimestep(self, obj):
    """
Appends Position,Classification and heading of Objects to dictionaries according to relative filters and checks
Input
    :param obj: object from 3D Worldmodel data.
    """
    if math.sqrt(abs(obj.objectGeometry[0].position.x)*abs(obj.objectGeometry[0].position.x) + \
        abs(obj.objectGeometry[0].position.y)*abs(obj.objectGeometry[0].position.y)) < self.attention_radius \
        and abs(obj.objectGeometry[0].position.x*obj.objectGeometry[0].speed)>self.min_prediction_speed:
        if not self.timestep_map.__contains__(obj.trackID):
            #his_queue = deque(maxlen = HISTORY_SIZE)
            #his_queue.append(obj)
            frame_id = 0

            if obj.objectGeometry[0].heading < 0:
                obj.objectGeometry[0].heading = obj.objectGeometry[0].heading+360

            elif obj.objectGeometry[0].heading > 360:
                obj.objectGeometry[0].heading = obj.objectGeometry[0].heading % 360
            else:
                pass

            if abs(obj.objectGeometry[0].boxYaw-(obj.objectGeometry[0].heading)) < 90:
                heading1 = obj.objectGeometry[0].heading
            elif abs((obj.objectGeometry[0].boxYaw+360)-(obj.objectGeometry[0].heading))<90:
                heading1 = obj.objectGeometry[0].heading
            else:
                heading1 = obj.objectGeometry[0].heading+180 #= obj.objectGeometry[0].boxYaw

            x = deque(maxlen = HISTORY_SIZE)
            x.append(obj.objectGeometry[0].position.x)

            y = deque(maxlen = HISTORY_SIZE)
            y.append(obj.objectGeometry[0].position.y)

            heading = deque(maxlen = HISTORY_SIZE)
            heading.append(heading1)

            boxyaw = deque(maxlen = HISTORY_SIZE)
            boxyaw.append(obj.objectGeometry[0].boxYaw)

            node={"x":x,"y":y,"heading":heading,"boxyaw":boxyaw, "type":obj.type.label}

            self.timestep_map[obj.trackID] = frame_id
            self.node_map[obj.trackID] = node
            self.updated_track_id[obj.trackID] = True
        else:
            if abs(obj.objectGeometry[0].boxYaw-(obj.objectGeometry[0].heading+360)) < 90:
                heading = obj.objectGeometry[0].heading
            elif abs((obj.objectGeometry[0].boxYaw+360)-(obj.objectGeometry[0].heading))<90:
                heading = obj.objectGeometry[0].heading
            else:
                heading = obj.objectGeometry[0].heading+180 #= obj.objectGeometry[0].boxYaw

            self.node_map[obj.trackID]["x"].append(obj.objectGeometry[0].position.x)
            self.node_map[obj.trackID]["y"].append(obj.objectGeometry[0].position.y)
            self.node_map[obj.trackID]["heading"].append(heading)
            self.node_map[obj.trackID]["boxyaw"].append(obj.objectGeometry[0].boxYaw)
            self.timestep_map[obj.trackID]+= 1
            self.updated_track_id[obj.trackID] = True
    else:
        pass
```

FIGURE 4. Function to append object information.

```python
347    def updateHistoryMap(self,data):
348        """
349    Updates History of Objects. If object isn't tracked, it's node is deleted
350    Otherwise objects position,types,orientation are appended .
351    Input
352        :param data: 3D Worldmodel data.
353
354        """
355
356        for k, v in self.updated_track_id.items():
357            self.updated_track_id[k] = False
358
359        worldmodel = data
360        self.seq+=1
361        for i in range(len(worldmodel.objects)):
362
363            for j in range(len(worldmodel.objects[i].objectGeometry)): #[0] yap
364                worldmodel.objects[i].objectGeometry[j].header.seq = self.seq
365
366                if not LABEL_FILTER:
367                    self.appendTimestep(worldmodel.objects[i])
368                else:
369                    if worldmodel.objects[i].type.label == 1 or worldmodel.objects[i].type.label == 2:
370                        self.appendTimestep(worldmodel.objects[i])
371        k_list=[]
372        for k, v in self.updated_track_id.items():
373
374            if v is False:
375                k_list.append(k)
376
377        for k in k_list:
378            del self.updated_track_id[k]
379            del self.timestep_map[k]
380            del self.node_map[k]
381
```

FIGURE 5. History hold/release function.

```
trajectron > conf > ⚙ waypointprediction.conf
 1  <?xml version="1.0" ?>
 2  <AdapterConfig>
 3      <Adapter>
 4          <Parameter name="SensorFusion" />
 5          <Parameter mode="SUBSCRIBER" />
 6          <Parameter message_history="10" />
 7          <Parameter topic="/sensor_fusion" />
 8      </Adapter>
 9      <Adapter>
10          <Parameter name="PredictedDetectedObjectArray" />
11          <Parameter mode="PUBLISHER" />
12          <Parameter message_history="10" />
13          <Parameter topic="/predicted_trajectories" />
14      </Adapter>
15      <Adapter>
16          <Parameter name="TrajectoryVisual" />
17          <Parameter mode="PUBLISHER" />
18          <Parameter message_history="10" />
19          <Parameter topic="/predicted_markers" />
20      </Adapter>
21      <Adapter>
22          <Parameter name="PointCloud2" />
23          <Parameter mode="PUBLISHER" />
24          <Parameter message_history="1" />
25          <Parameter topic="/predicted_pc" />
26      </Adapter>
27  </AdapterConfig>
```

FIGURE 6. Rostopic configuration file.

4. RESULTS

In order to test the work done, a Rosbag which was shown at Table 3, was collected with size 19.2 GB and 17.8 GB by 10 and 12 minutes driving in Mustafa Kemal district Ankara/Turkey. Rosbag data consist of ego vehicle's localization output, perception output obtained by sensor fusion which includes positions, orientations, speeds, sizes of adjacent objects, transform information, camera images and visualization markers which includes bounding box markers. In detail ego vehicle's localization output includes ego vehicle's position, orientation and speed, perception output includes positions, orientations, speeds, sizes of adjacent objects, transform information includes relative positions and orientations of global map, local map and sensors, visualization markers includes bounding box markers of vehicles, pedestrians and unknown objects.

Trajectory prediction module was tested by playing this Rosbag on the laptop which has T1000 graphic card, i7 9th Gen. Processor, 16 GB Ram. The developed software was run Docker and Conda virtual environment with CUDA 11.3 and PyTorch.

The part of the collected Rosbag under highway conditions (Sabancı Boulevard) and the inner city (Mustafa Kemal Mah.) parts are handled separately. The part in urban conditions is between 1.min and 8.min in Rosbag, and the part in highway conditions is between 8.min and 13.min. Trajectory predictions were obtained and visualized with green sphere markers by running the trajectory prediction ROS module on the Rosbag.

As can be seen in Figure 7 when driving in an urban area, the vehicle has entered dense environments many times. Although the speed and attention radius filter were added, incorrect speed information from the perception layer to the standing vehicles caused the algorithm to work slowly in a dense environment. Sudden changes of direction of vehicles and pedestrians in dense areas are a factor that reduces the performance of the behavior prediction module.

Rosbag's highway condition driving between 8 min and 13 min has achieved better results compared to the urban section. Similar to the urban roads section, although the incorrect perception output or the inability of the perception module to track the surrounding vehicle is a factor that reduces my performance, erroneous data has not been received from the perception module very often. One of the reasons for incorrect data coming from the perception module is incorrect data coming from GPS and IMU. However, since the data was collected at an off-peak time of the day, there was few erroneous data. Approximately 3 km of driving has been done under highway condition. Nine vehicles have been tracked by perception without error. From time to time, trajectory predictions of two or three vehicles have been made at the same time without loss of performance. The average and each 5 seconds RMSE of these nine vehicles are given in Table 4. The longer the estimation period, the greater the amount of error. Visualization of multi vehicle and four single vehicle are given in Figure 8, 9, 10, 11, 12 respectively.

During the tests, when there is one vehicle/pedestrian in the environment, the module works at 5fps, and as the environmental factors increase, the operating frequency of the module decreases inversely with the number of factors. In order not to reduce the performance, model parallelization methods such as converting Tensorrt can be used. However, the Tensorrt library does not yet support the conversion of some modules of the Trajectron++ model, such as GRU and GMM. By installing an attention radius and a speed filter, distant, stationary or very slow objects were prevented from entering the model, so it was seen that the model worked more efficiently.

Considering that the distance between the two lines is about 3m and considering that lane changes take about 2-4 seconds, according to the results of the vehicles on the highway given in Table 4, it has been found that vehicles up to two seconds have

an acceptable level of RMSE. RMSE can be reduced by getting more accurate data from the perception layer. Speed and heading information, which is one of the data from the perception layer, comes to the perception layer from the localization layer and it comes to localization layer from GPS and IMU sensors. Therefore, accurate data should be received from GPS and IMU sensors for the successful performing of the behavior prediction module, otherwise incorrect results will emerge from the speed and heading data entered as input to the model.

At the drives under urban condition, as seen at the dense environment Figure 7 and above inputs are entered into the model at the same time, which slows down the operating speed of the model. However, the fact that the route that both pedestrians and vehicles will take is more uncertain than highway roads is one of the negative factors affecting this structure in urban use. Similar to highway roads, speed and heading information from GPS and IMU are also important in urban areas. Even in the city, since the speeds of vehicles are lower, errors in speeds affect the system more. Although the slow speed makes it difficult to find the heading data correctly, more accurate heading data can be obtained thanks to the added binary heading data structure. In order for the behavior prediction module to be used in the city, the environment must be controlled and run on more advanced computers. Controlled environment refers to the environments in which the algorithm is fed with HDMap, traffic rules, traffic signs, node interactions and objects move within the framework of these rules.

TABLE 3. Collected Rosbag for testing.

| Dataset Type: Rosbag	
Size : 39 GB	**Duration: 13 min**
Data	*Data Description*
• Image Data	• Image data from camera
• Localization data	• Position and orientation data from localization module
• Camera Object Detections	• Object detections from camera
• Sensor fusion output	• Object detections from perception module
• Tf and Tf static	• Static and dynamic transformations of sensors, local and global map
• Lidar Data	• Pointcloud data from Lidar
• Visualization Markers	• Visualization Markers for 3D World model
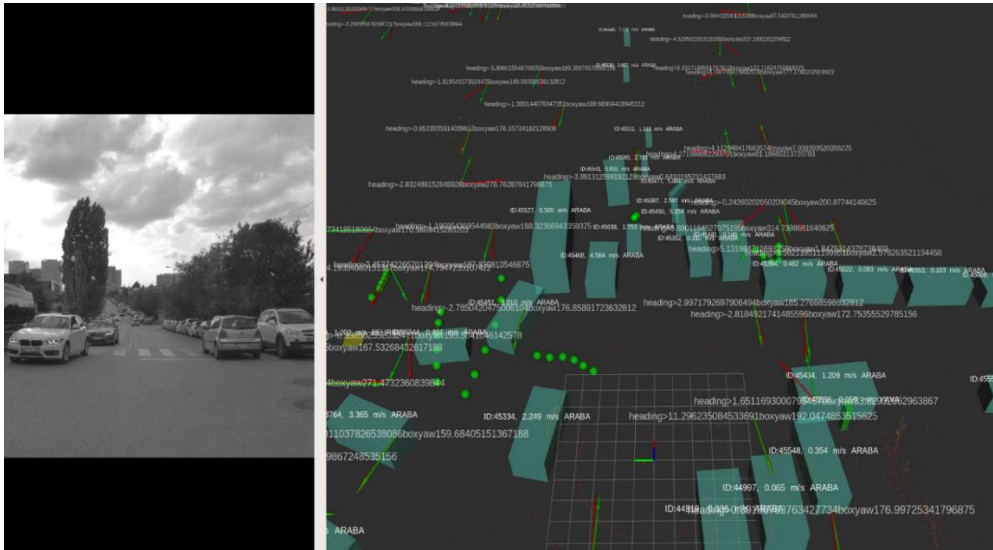
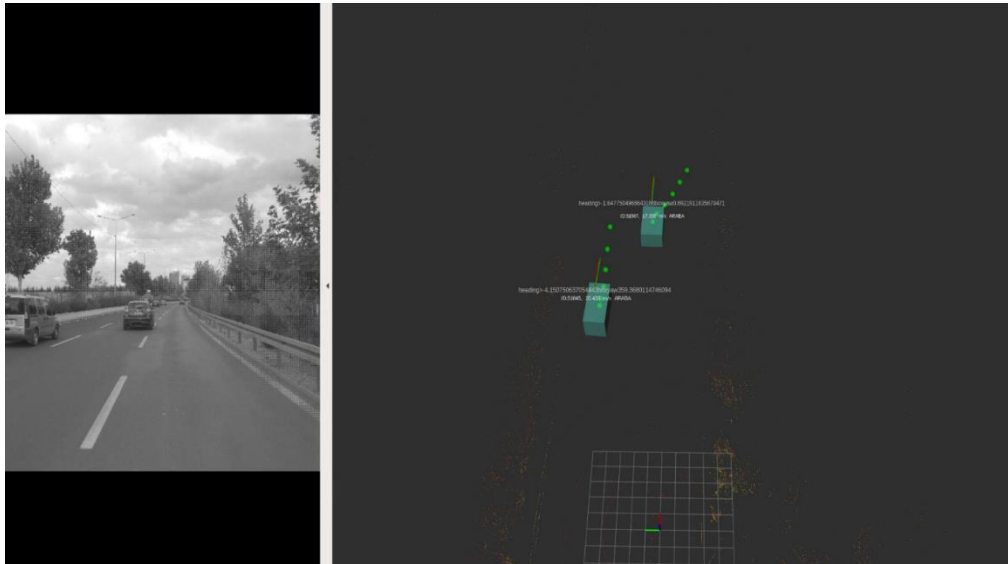FIGURE 7. Visualization of dense environment trajectory prediction.

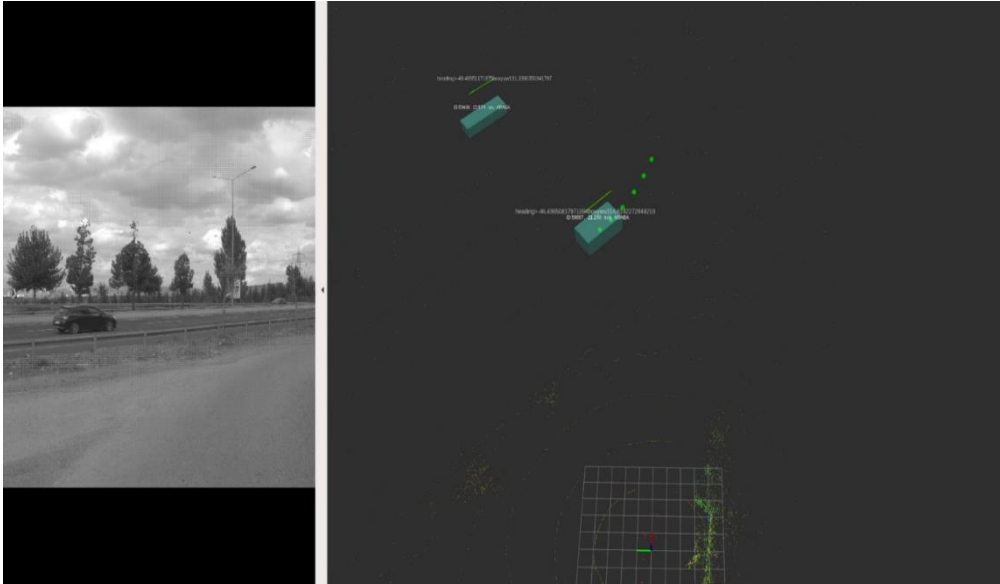FIGURE 8. Visualization of multi vehicle trajectory prediction.

FIGURE 9. Visualization of vehicle-1 trajectory prediction.
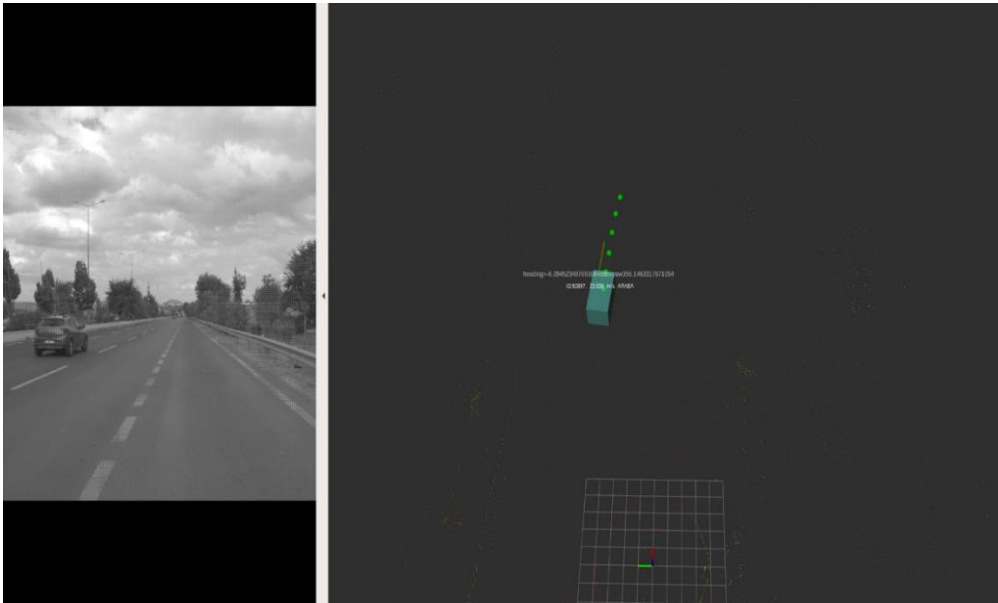

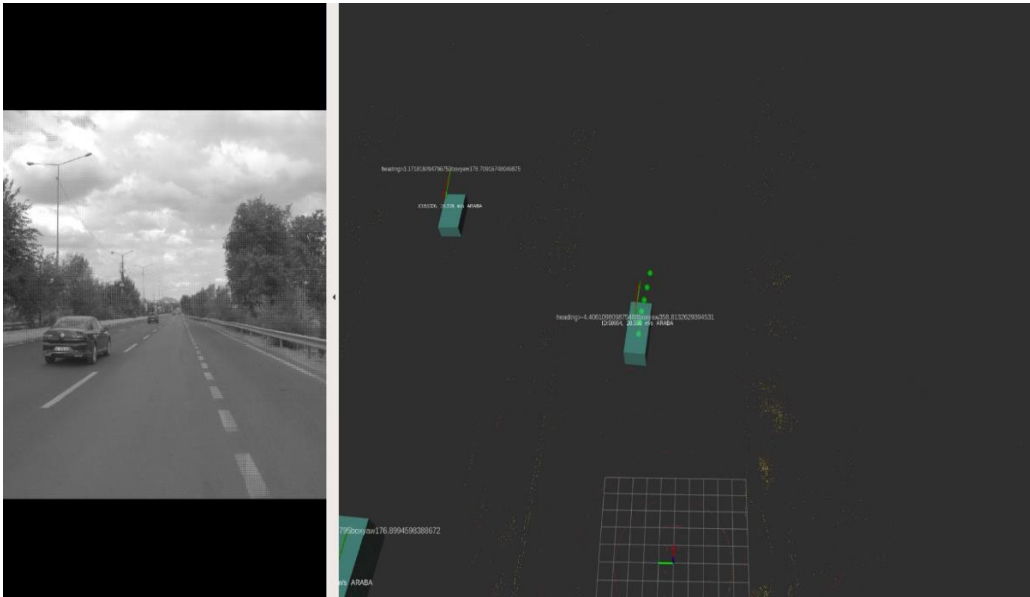FIGURE 10. Visualization of vehicle-2 trajectory prediction.

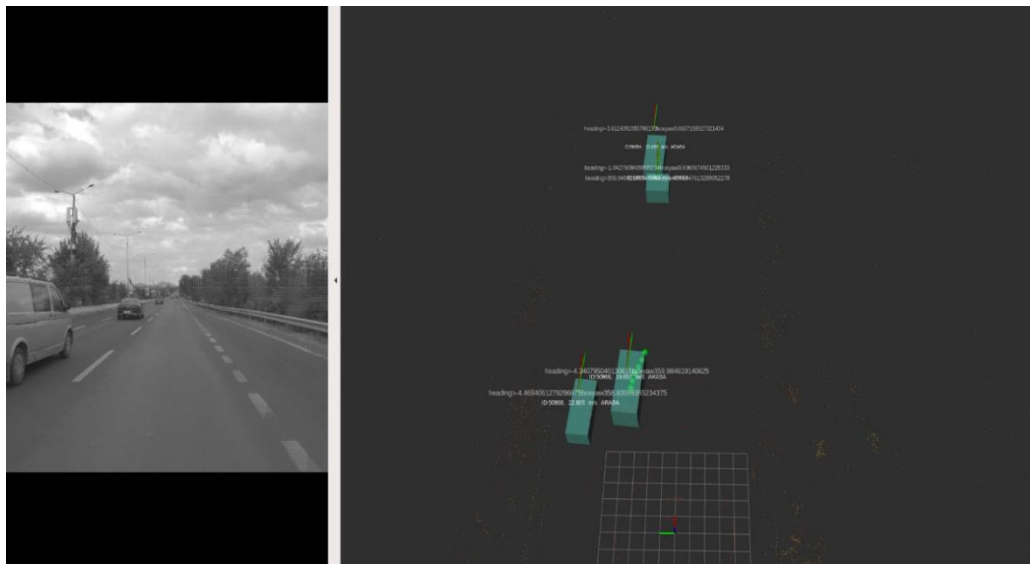FIGURE 11. Visualization of vehicle-3 trajectory prediction.

FIGURE 12. Visualization of vehicle-4 trajectory prediction.

TABLE 4. RMSE results of tested Rosbag.

Objects	RMSE(m)(1)				
	1s	2s	3s	4s	5s
Vehicle1	0.87	2.13	4.14	5.68	7.25
Vehicle2	0.92	2.09	4.28	5.82	6.43
Vehicle3	0.70	1.90	3.91	5.11	6.12
Vehicle4	1.20	2.42	4.72	6.23	8.04
Vehicle5	0.78	1.75	3.78	5.02	6.10
Vehicle6	1.45	2.10	3.99	4.97	7.05
Vehicle7	0.75	2.78	3.41	6.01	7.73
Vehicle8	1.28	2.35	4.66	5.28	6.87
Vehicle9	1.12	1.99	4.03	6.36	8.80
Overall	1.07	2.17	4.10	5.61	7.15

5. CONCLUSION

With the behavioral prediction module obtained as a result of this study, the probability of an autonomous vehicle making a mistake in the environment of moving objects has been significantly reduced. An end-to-end, real-time and robust behavioral prediction structure has been established from the detection module to the planning module, with contributions such as writing real-time inference to the current model, passing it to the ROS infrastructure, correcting or filtering faulty data.

From a broader perspective apart from transportation vehicles, this software can be used in any autonomous ground vehicle such as health care robots, agricultural robots, shuttle services etc. and it increases accuracy of planning module. In the future, it is planned to add ConvLSTM to the model for increasing accuracy and accelerate this model using TensorRT.

In the field of behavioral prediction, as it can be understood from the related work section, many models have been studied and continue to be studied. Although the studied models generally give successful results for the test sets of their own data-sets, their speed and performance decrease in dense environments such as urban areas. Successful results can be obtained by creating more complex data-sets and developing better model architectures.

In this study, trajectory prediction methods were examined, compared with each other, and as a result of these comparisons, graph structured structures were seen to be more successful. Afterwards, the data was collected and the content of the data was explained. The developed software is explained, tested and the results are expressed.

Once for all, an open-source behavioral prediction module for autonomous vehicle is developed. Although Trajectron++ is used as the model, there is no open source study for end-to-end integration of a model into an autonomous vehicle. The study is innovative in this aspect. The module works more efficiently and robustly due to the added history hold/drop structure and data filtering and correction features.

Author Contribution Statements The authors equally worked on the study. All authors read and approved the final copy of the manuscript.

Declaration of Competing Interests The authors declare that there is no conflict of interest regarding the publication of manuscript.

REFERENCES

[1] Koubaa, A., Robot Operating System (ROS): The Complete Reference (Volume 2), Springer, 2017, https://doi.org/10.1007/978-3-319-54927-9.

[2] Hintjens, P., ZeroMQ: Messaging for Many Applications, O'REILLY, CA, 2013.

[3] Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W., Robot operating system 2: Design, architecture, and uses in the wild, *Sci. Robot.*, 7 (66) (2022), https://doi.org/10.48550/arXiv.2211.07752.

[4] Salzmann, T., Ivanovic, B., Chakravarty, P., and Pavone, M., Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data, *European Conference on Computer Vision*, 12363 (2020), 683-700, https://doi.org/10.1007/978-3-030-58523-5_40.

[5] Huang, Y., Du, J., Yang, Z., Zhou, Z., Zhang, L. and Chen, H., A survey on trajectory-prediction methods for autonomous driving, *IEEE Trans. Intell. Veh.*, 7 (3) (2022), https://doi.org/10.1109/TIV.2022.3167103.

[6] Gulzar, M., Muhammad, Y. and Muhammad, N., A survey on motion prediction of

pedestrians and vehicles for autonomous driving, *IEEE Access*, 9 (2021), 137957-137969, https://doi.org/10.1109/ACCESS.2021.3118224.

[7]   Lin, C. F. and Ulsoy, A. G., Vehicle dynamics and external disturbance estimation for vehicle path prediction, *IEEE Trans. Control Syst. Technol.*, 8 (3) (2000), 508-518, https://doi.org/10.1109/87.845881.

[8]   Lefèvre, S., Vasquez, D. and Laugier, C., A survey on motion prediction and risk assessment for intelligent vehicles, *ROBOMECH J.*, 1 (1) (2014), 1-14, https://doi.org/10.1186/s40648-014-0001-z.

[9]   Schöller, C. Aravantinos, Lay, V. F. and Knoll, A., What the constant velocity model can teach us about pedestrian motion prediction, *IEEE Robot. Autom. Lett.*, 5 (2) (2020), 1696-1703, https://doi.org/10.48550/arXiv.1903.07933.

[10] Ammoun, S. and Nashashibi, F., Real time trajectory prediction for collision risk estimation between vehicles, *IEEE 5th International Conference on Intelligent Computer Communication and Processing*, (2009), 417-422, https://doi.org/10.1109/ICCP.2009.5284727.

[11] Schubert, R., Richter, E. and Wanielik, G., Comparison and evaluation of advanced motion models for vehicle tracking, *11th International Conference on Information Fusio*n, (2008), 1-6.

[12] Lytrivis, P., Thomaidis, G. and Amditis, A., Cooperative path prediction in vehicular environments, *11th International IEEE Conference on Intelligent Transportation Systems*, (2008), 803-808, https://doi.org/10.1109/ITSC.2008.4732629.

[13] Batz, T., Watson, K. and Beyerer, J., Recognition of dangerous situ ations within a cooperative group of vehicles, *IEEE Intelligent Vehicles Symposium*, (2009), 907-912, https://doi.org/10.1109/IVS.2009.5164400.

[14] Kumar, P., Perrollaz, M., Lefevre, S. and Laugier, C., Learning-based approach for online lane change intention prediction, *IEEE Intelligent Vehicles Symposium (IV)*, (2013), 797- 802, https://doi.org/10.1109/IVS.2013.6629564.

[15] Qiao, S., Shen, D., Wang, X., Han, N. and Zhu, W., A self-adaptive parameter selection trajectory prediction approach via hidden markov models, *IEEE Trans. Intell. Transp. Syst.*, 16 (1) (2015), 284-296, https://doi.org/10.1109/TITS.2014.2331758.

[16] Deng, Q. and Soffker, D., Improved driving behaviors prediction based on fuzzy logic-hidden markov model (fl-hmm), *IEEE Intelligent Vehicles Symposium (IV)*, (2018), 2003-2008, https://doi.org/10.1109/IVS.2018.8500533.

[17] Gindele, T., Brechtel, S. and Dillmann, R., A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments, *13th Int. IEEE Conf. Intell. Transp. Syst.*, (2010), 1625-1631, https://doi.org/10.1109/ITSC.2010.5625262.

[18] Lee, N., Choi, W., Vernaza, P., Chor, C. B., Torr, P. H. S. and Chandraker, M. K., DESIRE: distant future prediction in dynamic scenes with interacting agents, *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), (2017), 2165-2174,

https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.233.

[19] Dai, S., Li, L. and Li, Z., Modeling vehicle interactions via modified LSTM models for trajectory prediction, *IEEE Access*, 7 (2019), 38287-38296, https://doi.org/10.1109/ACCESS.2019.2907000.

[20] Sun, L., Zhan, W. and Tomizuka, M., Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning, *IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*, (2018), 2111-2117, https://doi.org/10.1109/ITSC.2018.8569453.

[21] Kuefler, A., Morton, J., Wheeler, T. and Kochenderfer, M., Imitating driver behavior with generative adversarial networks, *IEEE Intelligent Vehicles Symposium (IV)*, (2017), 204-211, https://doi.org/10.1109/IVS.2017.7995721.

[22] Choi, S., Kim, J. and Yeo, H., Trajgail: Generating urban vehicle trajectories using generative adversarial imitation learning, *Transp. Res. Part C Emerg. Technol.*, 128 (2021), 103091, https://doi.org/10.48550/arXiv.2007.14189.

[23] Wulfmeier, M., Ondruska, P. and Posner, I., Maximum entropy deep in verse reinforcement learning, (2015), https://doi.org/10.48550/arXiv.1507.04888.

[24] You, C., Lu, J., Filev, D. and Tsiotras, P., Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning, *Robot. Auton. Syst.*, 114 (2019), 1-18, https://doi.org/10.1016/j.robot.2019.01.003.

[25] Jung, C. and Shim, D. H., Incorporating multi-context into the traversability map for urban autonomous driving using deep inverse reinforcement learning, *IEEE Robot. Autom. Lett.*, 6 (2) (2021), 1662-1669, https://doi.org/10.1109/LRA.2021.3059628.

[26] Geiger, A., Lenz P. and Urtasun, R., Are we ready for autonomous driving? The KITTI vision benchmark suite, *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, (2012), 3354-3361, https://doi.org/10.1109/CVPR.2012.6248074.

[27] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G. and Beijbom, O., nuScenes: A multimodal dataset for autonomous driving, *CVPR*, (2020), 11618-11628, https://doi.ieeecomputersociety.org/10.1109 /CVPR42600.2020.01164.

[28] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D. and Hays, J., Argoverse: 3D Tracking and Forecasting With Rich Maps, *IEEE Conference on Computer Vision and Pattern Recognition*, (2019), 8748-8757, https://doi.org/10.1109/CVPR.2019.00895.

[29] Coifman, B. A critical evaluation of the next generation simulation (NGSIM) vehicle trajectory dataset, *Trans. Res. B Methodol.*, 105 (2017), 362-377, https://doi.org/10.1016/j.trb.2017.09.018.

[30] Caesar, H., Kabzan, J., Tan, K., nuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles, *CVPR ADP3 workshop*, (2021), https://doi.org/10.48550/arXiv.2106.11810.

[31] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G. and Beijbom, O., nuScenes: A multimodal dataset for autonomous driving, *CVPR*, 2020.

[32] Enyen, Deep Trajectory Prediction, (2017), https://github.com/enyen/Deep-Trajectory-Prediction.

[33] Deo, N. and Trivedi, M. M., Convolutional social pooling for vehicle trajectory prediction, *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, (2018), 1549-15498, https://doi.org/10.1109/CVPRW.2018.00196.

[34] Lefkopoulos, V., Menner, M., Domahidi, A. and Zeilinger, M. N., Interaction-aware motion prediction for autonomous driving: A multiple model kalman filtering scheme, *IEEE Robotics and Automation Letters*, 6 (1) (2021), 80-87, https://doi.org/10.1109/LRA.2020.3032079.

[35] Deo, N., Rangesh, A. and Trivedi, M. M., How would surround vehicles move? a unified framework for maneuver classification and motion prediction, *IEEE Trans. Intell. Veh.*, 3 (2) (2018), 129-140, https://doi.org/10.1109/TIV.2018.2804159.

[36] Deo, N. and Trivedi, M. M., Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms, *IEEE Intelligent Vehicles Symposium (IV)*, (2018), 1179-1184, https://doi.org/10.1109/IVS.2018.8500493.

[37] Tang C. and Salakhutdinov, R. R., Multiple futures prediction, *Adv. Neural Inf. Process. Sys.*, 32 (2019), 15424-15434, https://doi.org/10.48550/arXiv.1911.00997.

[38] Deo, N. and Trivedi, M. M., Convolutional social pooling for vehicle trajectory prediction, *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, (2018), 1468-1476.

[39] Messaoud, K., Yahiaoui, I., Verroust-Blondet, A. and Nashashibi, F., Attention based vehicle trajectory prediction, *IEEE Trans. Intell. Veh.*, 6 (1) (2020), 175-185, https://doi.org/10.1109/TIV.2020.2991952.

[40] Li, X., Ying, X. and Chuah, M. C., Grip++: Enhanced graph-based interaction-aware trajectory prediction for autonomous driving, arXiv:1907.07792, (2019), https://doi.org/10.48550/arXiv.1907.07792.

[41] Zhao, Z., Fang, H., Jin, Z. and Qiu, Q., Gisnet: Graph-based information sharing network for vehicle trajectory prediction, *IEEE International Joint Conference on Neural Networks (IJCNN)*, (2020), 1-7, https://doi.org/10.48550/arXiv.2003.11973.

[42] Zhao, T., Xu, Y., Monfort, M., Choi, W., Baker, C., Zhao, Y., Wang, Y. and Wu, Y. N., Multi-agent tensor fusion for contextual trajectory prediction, *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2019), https://doi.org/10.48550/arXiv.1904.04776.

[43] Wang, Y., Zhao, S., Zhang, R., Cheng, X. and Yang, L., Multi-vehicle collaborative learning for trajectory prediction with spatio-temporal tensor fusion, *IEEE Transactions on Intelligent Transportation Systems*, 23 (1) (2022), 236-248, https://doi.org/10.1109/TITS.2020.3009762.

[44] Saleh, K., Hossny, M. and Nahavandi, S., Long-term recurrent predictive model for intent prediction of pedestrians via inverse reinforcement learning, *Digital Image*

Computing: Techniques   and   Applications   (DICTA),   (2018),   1-8,
https://doi.org/10.1109/DICTA.2018.8615854.

[45] Kuefler, A., Morton, J., Wheeler, T. and Kochenderfer, M., Imitating driver behavior
with generative adversarial networks, *IEEE Intelligent Vehicles Symposium (IV)*, (2017),
204-211, https://doi.org/10.1109/IVS.2017.7995721.

[46] Wulfmeier, M., Rao, D., Wang, D. Z., Ondruska, P. and Posner, I., Large-scale cost
function learning for path planning using deep inverse reinforcement learning, *Int. J.
Rob. Res.*, 36 (10) (2017), 1073-1087, https://doi.org/10.1177/0278364917722396.

[47] Fernando, T., Denman, S., Sridharan, S. and Fookes, C., Neighbourhood context
embeddings in deep inverse reinforcement learning for predicting pedestrian motion
over long time horizons, *IEEE/CVF International Conference on Computer Vision
Workshops*, (2019), 1179-1187, https://doi.org/10.1109/ICCVW.2019.00149.

[48] Chang, M.-F., Lambert, J., Sangkloy,P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr,
P., Lucey, S., Ramanan, D. et al., Argoverse: 3d tracking and forecasting with rich maps,
IEEE/CVF Conference on Computer Vision and Pattern Recognition, (2019), 8748-
8757.

[49] Mercat, J., Gilles, T., El Zoghby, N., Sandou, G., Beauvois, D. and Gil, G. P., Multi-
head attention for multi-modal joint vehicle motion forecasting, *IEEE International
Conference   on   Robotics   and   Automation   (ICRA)*,   (2020),   9638-9644,
https://doi.org/10.1109/ICRA40945.2020.9197340.

[50] Ngiam, J., Caine, B., Vasudevan, V., Zhang, Z., Chiang, H.-T. L., Ling, J., Roelofs, R.,
Bewley, A., Liu, C., Venugopal, A., et al., Scene transformer: A unified multi-task
model for behavior prediction and planning, *International Conference on Learning
Representations (ICLR)*, (2021), https://doi.org/10.48550/arXiv.2106.08417.

[51] Liu, Y., Zhang, J., Fang, L., Jiang, Q. and Zhou, B., Multimodal motion prediction with
stacked transformers*, IEEE/CVF Conference on Computer Vision and Pattern
Recognition*, (2021), 7577-7586, https://doi.org/10.1109/CVPR46437.2021.00749.

[52] Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S. and Urtasun, R., Learning lane
graph representations for motion forecasting, *European Conference on Computer Vision
(ECCV)*, (2020), 541-556, https://doi.org/10.48550/arXiv.2007.13732.

[53] Gu, J., Sun, C. and Zhao, H., Densetnt: End-to-end trajectory prediction from dense goal
sets, *IEEE/CVF International Conference on Computer Vision*, (2021), 15303-15312,
https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.01502.

[54] Zeng, W., Liang, M., Liao, R. and Urtasun, R., Lanercnn: Distributed representations
for graph-centric motion forecasting, *IEEE/RSJ International Conference on Intelligent
Robots   and   Systems   (IROS)*,   (2021),   532-539,
https://doi.org/10.1109/IROS51168.2021.9636035.

[55] Song, H., Luan, D., Ding, W., Wang, M. Y. and Chen, Q., Learning to predict vehicle
trajectories   with   model-based   planning,   arXiv:2103.04027,   (2021),
https://doi.org/10.48550/arXiv.2103.04027.