Distributional Prediction of Human Driving Behaviours using Mixture Density Networks

Karen Leung Edward Schmerling Marco Pavone

***Abstract*— Confident predictions of human driving behaviours are necessary in designing safe and efficient control policies for autonomous vehicles. A better understanding of how human drivers react to their surrounding may avoid the design of overly-conservative control policies which require greater cost (e.g., time, traffic flow disruption) to achieve their objective. In this paper, we explore ways to learn distributions over human driver actions that are typical of a highway setting. We use actions filtered from Next Generation SIMulation (NGSIM) vehicle trajectory data gathered on the US 101 highway as training data for a Recurrent Neural Network. In particular, we use a Mixture Density Network (MDN) model to represent predicted driver actions as a Gaussian Mixture Model. We present and discuss exploratory results on the filtering of the raw NGSIM data and design of the MDN model.**

I. Introduction

Advances in artificial intelligence and robotics are being incorporated into nearly every aspect of daily life. With more than 13,000 average annual miles driven per person in the United States [1], it is no surprise that in the future "self-driving" autonomous cars will have huge impact on everyday life. This area of research has multi-faceted implications, particularly the improvement of safety for drivers, reduction in congestion and carbon emissions, and greater mobility. There are many avenues one can take in planning the autonomy of cars. In general, these avenues can be divided into four hierarchies: (i) route planning, (ii) path planning, (iii) maneuver choice and (iv) trajectory planning [2]. In this work, we look at the foundations that support the latter two hierarchies; in particular we aim to predict how human drivers behave in order to formulate the optimal maneuver choice and trajectory planning for an autonomous vehicle. One central assumption in this work is that the *intent* of the human driver is *unknown*, the actions and reactions of human drivers are inconsistent even within a fixed driving scenario. To predict exactly how humans behave is very difficult because human intent can be random or highly dependent on external factors (e.g., emotions, distractions). That is to say that there is no "correct" way to drive, but rather a distribution of possible actions drivers may take. Thus a human driving a car may be thought of as a random process and it is important to model the associated distributions as accurately as possible. Many approaches have been taken to learn driver intent and behaviours directly from the human source, such as using eye and facial trackers [3], [4]. In this work we approach the modeling problem from an external perspective, that is, learning how drivers act using only information from outside the vehicle (e.g., position on the road, neighbouring vehicle states). We use techniques from machine learning to learn distributions over behaviours given contextual information. Given these distributions, we can minimise conservatism in the decision-making policies of autonomous vehicles that interact with human drivers by planning around and anticipating possible behaviours they may take. This has the potential to both improve safety and efficiency of autonomous vehicles. This paper investigates the actions of a target vehicle given the state of its neighbouring vehicles, like the set-up shown in Fig. 1, and discusses how we can interpret these results and potentially use them to construct an optimal autonomous control policy.

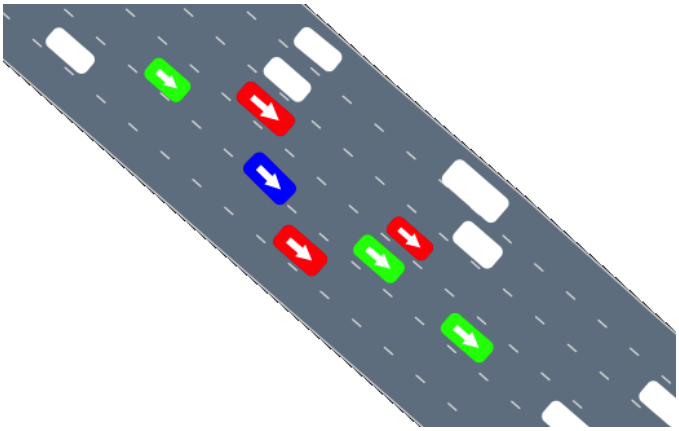


Fig. 1: Three nearest longitudinal (green) and lateral (red) vehicles of a target vehicle (blue) during a lane change maneuver.

A. Literature

One of the main problems in creating self-driving cars is the difficulty of modeling a human driver. To truly capture how a human thinks, it would require knowing exactly how the human brain works and knowing how every single driver in the world drives. Clearly this is not a viable option. Instead, a common approach is to gather real driving data and use machine learning to learn driver behaviour from it [5], [6], [7], [8]. Currently, a state-of-the-art method in learning driving behaviour is Inverse Reinforcement Learning (IRL) [9]. A reward function governing human action choice is learned by maximising the likelihood of expert demonstrations assuming an exponential family distribution [5], [10].

However, this method assumes that the expert demonstrations are locally optimal, which in general is not true for drivers and it relies on handcrafted features. By formulating

Karen Leung and Marco Pavone are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, {karenl7, pavone}@stanford.edu.

Edward Schmerling is with the Institute for Computational & Mathematical Engineering, Stanford University, Stanford, CA 94305, schmrlng@stanford.edu.

a single reward function from the expert demonstrations, [9] assumes that all drivers on the road are essentially optimising over the same reward function which is clearly not entirely true. Further, the reward function is not unique and depends on the handcrafted features. Although this promotes the tractability of the problem, it possesses several limitations, in particular, it relies on domain knowledge and human experience in order to select 'good' features. Even with well-selected features, its discriminative power is often unknown before the learning process, and its effectiveness is very context-based. In addition, a generic set of features is difficult to create since driving patterns can differ significantly in different driving contexts. For example, driving in the US is different to driving in Australia, India or Vietnam. Nonetheless, IRL does offer a simple and tractable way of modeling human drivers, and in a simple setting such as one ego-vehicle and one human driver on a road, it has shown to be effective [5], [11].

In [11], the authors use IRL to model human drivers and construct a control policy using techniques from game theory. Their central insight is that drivers do not operate in isolation; the actions of the autonomous vehicle will affect the actions of the human driver and vice versa. The problem formulation is set up in a Stackelberg manner [12] and solving for the optimal control policy is made tractable by using Model Predictive Control (MPC). The model was derived using deterministic techniques yet the resulting control policy was promising when tested with humans in the loop. This perspective of using game theory is a critical aspect that should be further investigated because the actions of the ego-vehicle undoubtedly will affect human drivers and we can leverage this to create more efficient and communicative driving.

Alternatively, to enrich the model, uncertainty can be added into the problem formulation, especially through environment sensing and environment predictability [13]. While there are existing frameworks that readily admit dynamic obstacles [14], the difficulties in the modeling and prediction of real-world domains such as cars are typically more complex. One of the main objectives in [13] is to use pattern-based approaches to predict the trajectories of an observed car and plan a path with uncertainty in the dynamics. A similar idea of predicting possible maneuvers of a car is presented in [15]. The authors present an integrated inference and decision-making approach that models the possible maneuvers of the car as a discrete set of closed loop policies when reacting to the actions of other agents. They employ a Bayesian approach and used observed history of states of nearby cars to estimate a probability distribution over the potential maneuvers that each nearby car might be executing. Unlike the IRL approach where the human drivers are assumed to be optimal planners, Bayesian-based approaches describe the actions of a human driver more accurately because it encapsulates the central assumption of not knowing exactly the intent of human drivers. As such, this distributional representation of human driving actions offers a richer model for the construction of decision-making control policies for autonomous vehicles.

A Mixture Density Network (MDN) [16] combines elements from machine learning and probability distributions. The idea of a MDN is that rather than learning a single output value using neural networks, it predicts an entire probability distribution for the output using a Gaussian Mixture Model (GMM) [17], a convex combination of Gaussians. The GMM can potentially describe the multimodal nature of driving behaviours and describe better possible intentions drivers may have given a situation. The distribution is represented as a linear combination of kernel functions [16]. Since each kernel function represents the distribution of a particular intent, it may reflect upon the driving style of that particular driver (e.g., aggressive, slow, prone to lane changes). The advantage of using neural networks over IRL is that the features are naturally constructed rather than pre-determined. As such, neural networks are able to extract high level discriminative features from the input data and combined with a probability distribution over driving actions, this can result in a better representation of human driving behaviours.

Relating back to [11], it is very natural to formulate the problem using game theory [12], [18]. Since we have some understanding of how human drivers behave given their surroundings, an autonomous vehicle can exert some level of control authority over them by moving in some particular desired state. Since there is uncertainty in the human driver's intent, this problem can potentially be placed in the context of Bayesian game theory, a game of incomplete information about the intent and behaviours of players. Such games can be converted into a complete game but with imperfect information by introducing Nature as a player [19]. In a Bayesian game, a player, or in this context the ego-vehicle, has a belief about the intentions other players (human drivers) and will take actions based on this belief and may change based on past actions other players have made. Thus there is a potential to combine MDN and Bayesian games to help predict the intentions of human drivers and design an optimal control policy for an autonomous vehicle. Ideally the policy incorporates some notion of *anticipation*, predicting what human drivers might do based on their intent, and *reaction*, how they will react to changes in the surrounding.

B. Motivation of proposed work

The ultimate goal within the autonomous car industry is to design safer, more efficient and interactive vehicles while maintaining the persona of an "ideal" realistic human driver. Interactive driving in particular is an interesting avenue; a future vision would involve fleets of autonomous vehicles on the road that can communicate with each other. Possible applications include improving traffic flow, monitoring isolated roads and aid in keeping vehicles within the law.

If we anticipate the actions of human drivers by predicting their intent, we can potentially design less overly-conservative control policies which can lead to more efficient traffic flows yet still maintaining an acceptable level of safety. Thus it is important to develop an accurate model for predicting driving behaviours. A probabilistic model is used because it is able to capture all possible actions that drivers may take.

Even though highway driving poses a more simplistic environment than city driving, it comprises a large portion of time spent on the road by drivers in the US. Especially

with a higher speed limit, accidents on highways are often catastrophic. Thus highway driving is an important domain to investigate and we aim to develop a model to predict human driving behaviours based off collected vehicle trajectory data.

C. Statement of work

The work presented in this paper can be separated into the following components: (i) filtering and state estimation of the raw NGSIM data, (ii) designing a MDN framework to produce a probability distribution of driving behaviours and (iii) formulating a state representation for the MDN model. The primary contribution is the filtering of relevant information from the raw data and using this as training data for the MDN model. By gaining intuition of the results, we gain a deeper insight into how we can improve the distributional model and also methods we can use to construct an optimal control policy for an autonomous vehicle.

Highway data from NGSIM of the 101 highway [20] was used as training data for the MDN model. Before the learning process, the data was filtered using the Extended Kalman Filter (EKF) [21], [22] and transformed into Frenet coordinates such that it is highway agnostic. Then the $k-$nearest neighbour algorithm was applied in order to extract contextual information with respect to a target vehicle. Using this information, we learn an output probability distribution for the possible control outputs a human driver may take given its surroundings. A discussion on the implications and interpretation of the results is given and this gives more insight into possible avenues we can take to add more complexity into the model.

II. Problem formulation

Our goal is to produce probability distribution over the current driving actions of a target vehicle given some representation of the state. In particular, we are looking at vehicles in a highway setting. Essentially we aim to give a representation of the conditional probability $P(\mathbf{u} = \mathbf{U}|\mathbf{x} = \mathbf{X})$ where $\mathbf{u}$ is the vector of driving actions and $\mathbf{x}$ is some representation of the state which encapsulates information about the target vehicle and its surroundings.

The continuous time system given in (1) [14] is used to model the dynamics of the vehicles. The vector of control inputs is $\mathbf{u}(t) = [\delta(t), a(t)]^T$ and the *state (dynamic) vector* is given by $\mathbf{x}_d(t) = [x(t), y(t), \theta(t), v(t)]^T$. $\delta(t)$ is the steering input, $a(t)$ is acceleration, $x(t)$ and $y(t)$ are the global position coordinates, $\theta(t)$ is the global heading angle and $v(t)$ is the forward velocity of the vehicle. A slight simplification to the model in [14] is made by defining $\delta = \frac{\tan u_\phi}{L}$. Further, if the highway is not perfectly straight, we will later make a transformation which makes the coordinates highway agnostic.

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos\theta(t) \\ v(t)\sin\theta(t) \\ v(t)\delta(t) \\ a(t) \end{bmatrix} \tag{1}$$

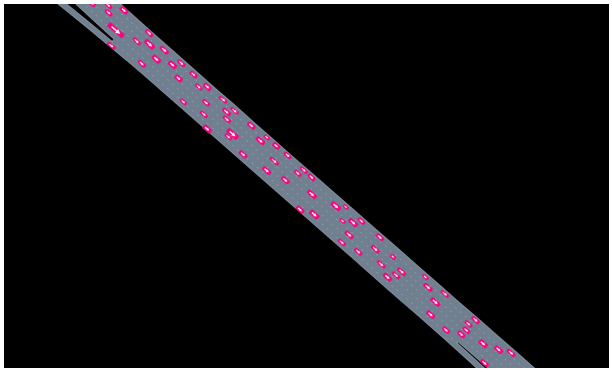


Fig. 2: Test section of Highway 101 from the NGSIM data. Vehicles are pink and moving towards the lower right.

III. Proposed Solution

The problem can be separated into three parts: filtering the NGSIM data, constructing a sufficient representation of the state and designing the MDN model. The following subsections describe the details of each part.

A. Filtering NGSIM Data

The training data is taken from the NGSIM program [20] where detailed vehicle trajectory data (position, velocity and acceleration) on southbound US 101 was collected over a 45 minute period. The study area is approximately 640 meters in length and include five lanes of traffic plus an auxiliary lane for the on/off-ramps and this is seen in Fig. 2. The velocity and acceleration data are very noisy, thus an EKF is used to give better estimates of the data. For brevity, the EKF formulation will not be described here but can be found in [21] and [22]. However, to prepare the training data, the control inputs $\delta(t)$ and $a(t)$ also need to be estimated since it is not part of the raw data set. To include the control inputs as part of the EKF estimation, we augment the system by appending the state vector with the control inputs. The augmented continuous time system is given in (2).

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \\ \dot{v}(t) \\ \dot{\delta}(t) \\ \dot{a}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos\theta(t) \\ v(t)\sin\theta(t) \\ v(t)\delta(t) \\ a(t) \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

During implementation, the equivalent discrete time system ($\Delta t = 0.1s$) was used, including terms up to $\mathcal{O}(\Delta t^2)$.

Since the portion of highway that is studied is relatively straight and flat, we assume a zero bias on our prior of δ and a. Naturally we also need to consider observation noise and process noise (assume zero mean). Due to the augmentation, the primary tuning parameters for this EKF are the variances on the prior and the variances on the process noise on the $\dot{\delta}$ and $\dot{a}$ components. This will dictate the smoothness of the state trajectories and controls, and rigidity to the dynamics.

Once the estimation is complete, the states need to be transformed into Frenet coordinates, an intrinsic coordinate system which allows the states to be road/highway agnostic [23]. Frenet coordinates are parameterised by (s, τ, ϕ).

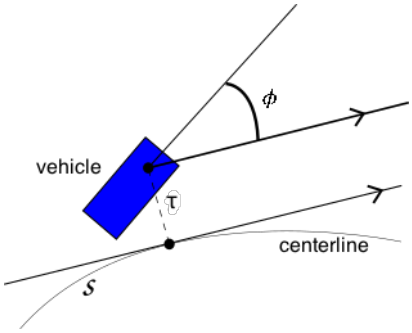


Fig. 3: Illustration of Frenet coordinates, describing the (s, τ, ϕ) coordinates.

Essentially the centerline of the road is known from the NGSIM data and the closest point from the car and centerline is found. At that point, the distance along the centerline is s, the distance between the car and centerline is τ and the difference in angle between the car and the tangent at the point on the centerline is $\phi = \theta - \theta_{road}$. This is illustrated in Fig. 3. In terms of the steering input, the transformation to this intrinsic coordinate system essentially becomes $\delta_{frenet} = \hat{\delta} = \delta - \delta_{road}$ where δ_{road} represents the curvature of the road.

1) EKF Results: The EKF for (2) was tuned to give physically realistic results. The results for a particular car ID 2133, is given in Fig. 4. The EKF can be continued to be tuned for better results, but for the purpose of using it for an MDN, this is sufficient.

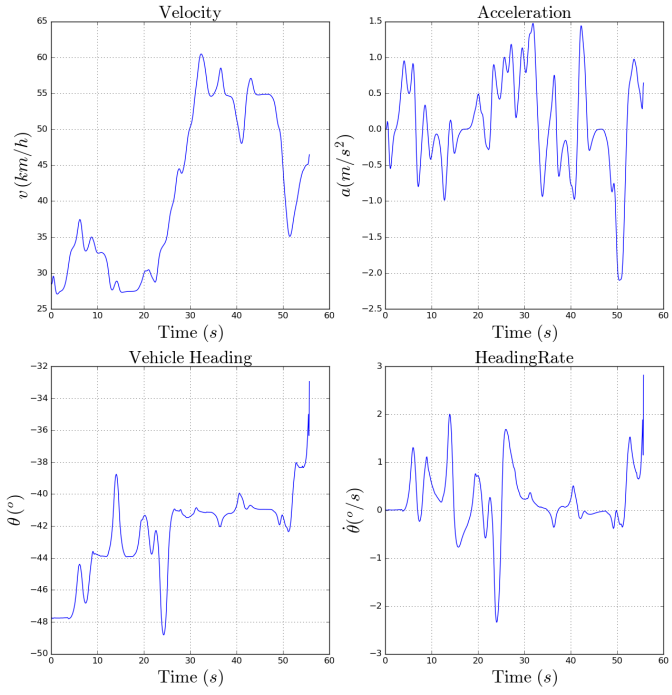


Fig. 4: Estimation of states and control action from EKF. (Car ID 2133)

The filtered (dynamic) states and the corresponding control actions are transformed into Frenet coordinates, then the control actions are normalised to have zero mean and unit standard deviation. This filtered and normalised data will be used to generate the training data for the MDN framework.

B. Recurrent Neural Network

Neural Networks are useful in many applications because they are universal function approximators. They are capable of extracting hierarchal features from complicated inputs. A Recurrent Neural Network (RNN) is used rather than a conventional neural network because it takes into account an ordered sequence of states rather than just the current state. In particular, we use a RNN within the MDN model, but this subsection focuses on the set up of the RNN.

The formulation of the sequence of states used for the RNN is described here. For a *target* vehicle at a particular time t, its *vehicle state* $\mathbf{x}_v^{(t)}$ is described by

$$\mathbf{x}_v^{(t)} = [s(t),\, \tau(t),\, l(t),\, \phi(t),\, v(t)]^T \qquad \text{(vehicle state)}$$

where (s, τ, ϕ) are the Frenet coordinates, l is the lane number and v is the velocity of the vehicle. The actions of the target vehicle depend on its surrounding environment. Hence we define a *scene state* $\mathbf{x}_s^{(t)}$ which is a vector of the target car's vehicle state concatenated by the vehicle states of the $k-$nearest neighbours (nn1, nn2, ..., nnk).

$$\mathbf{x}_s^{(t)} = [\mathbf{x}_{v_{target}}^{(t)}; \mathbf{x}_{v_{nn1}}^{(t)}; \mathbf{x}_{v_{nn2}}^{(t)}; \dots \mathbf{x}_{v_{nnk}}^{(t)}] \quad \text{(scene state)}$$

The input into the RNN, called the *model state* $\mathbf{x}(t)$, is composed of a history of scene states (the current scene state concatenated with the previous T scene states).

$$\mathbf{x}(t) = [\mathbf{x}_s^{(t)}; \mathbf{x}_s^{(t-1)}; \mathbf{x}_s^{(t-2)}; \dots \mathbf{x}_s^{(t-T)}] \quad \text{(model state)}$$

Using $\mathbf{x}(t)$, we can predict the current control actions $\mathbf{u}(t) = [\hat{\delta}(t), a(t)]^T$ using an MDN model.

Apart from the hyper parameters inherent to a RNN such as regularisation terms, key parameters to tune are the number of nearest neighbours and history length. For this paper, we consider a history length of ten time steps, which is equivalent to a one second interval and the number of nearest neighbours will be discussed later. The benefit of using RNN over other methods that utilise a time series of states such as a hidden Markov model is that RNNs are scalable to higher dimensions. Weights can be learned by training on a particular history length and the same weights can be used for a different history length by simply adding more cells into the RNN but not altering the weights.

C. MDN Model

By using a MDN to learn distributions over control inputs given model states, we obtain a GMM - the weighted sum of many Gaussians with different means and standard deviations. The conditional probability of a particular control action $\mathbf{u}(t) = (\hat{\delta}(t),\, a(t))$ given the model state $\mathbf{x}(t)$ is given by (3). (We drop the superscript t to be concise but this is for a particular time t.)

$$P(\mathbf{U} = \mathbf{u}|\mathbf{X} = \mathbf{x}) = \sum_{k=0}^{K-1} \pi_k(\mathbf{x}) \phi(\mathbf{u},\, \boldsymbol{\mu}_k(\mathbf{x}), \sigma_k(\mathbf{x})) \quad (3)$$

$\phi(\mathbf{u},\, \boldsymbol{\mu}_k(\mathbf{x}), \sigma_k(\mathbf{x}))$ is the k-th kernel function; we shall restrict ourselves to Gaussians for this problem [16]. $\boldsymbol{\mu}_k(\mathbf{x})$ and

$\boldsymbol{\sigma}_k^2(\mathbf{x})$ are the mean and variance vector of the k-th kernel function respectively. It is assumed that the components of the output vector are statistically independent within each component of the distribution. To add complexity to the model, each control action is represented by its own standard deviation. Theoretically, this complication is not necessary since a Gaussian mixture model can approximate any given density function to arbitrary accuracy [17]. Thus the kernel function is simply a multivariate normal distribution. $\pi_k(\mathbf{x})$ are the weights, or mixing coefficients, on each distribution. Since they represent the probability of each Gaussian occurring, (4) must be satisfied.

$$\sum_{k=0}^{K-1} \pi_k(\mathbf{x}) = 1 \tag{4}$$

Typically, the loss function for the neural network is given by minimising the negative log-likelihood plus regularisation terms to prevent over fitting. This assumes that the training data is drawn independently from the distribution. Minimising negative log-likelihood is a natural choice for a loss function because we are trying to produce accurate predictions. However, to capture the "interesting" cases when the cars are not simply moving straight and at constant velocity, weights are used to penalise the interesting cases. Thus the loss function is given by (5) where Q is the number of training data, w_q is a weighting on each term where

$$w_q = \kappa|\hat{\delta}_q| + |a_q| + 1, \qquad \kappa = 1.5$$

and $\mathcal{W}_i$ are the matrices associated with the RNN. The Frobenius norm is used on the matrices for regularisation and is weighted by γ_i. In other words, we penalise vehicles that have control actions that are much different to the nominal (zero steering and acceleration) actions.

$$E = \sum_{q=1}^{Q} w_q E^{(q)} + \sum_{i}^{W} \gamma_i \|\mathcal{W}_i\|_F \tag{5}$$

$$\text{where} \quad E^{(q)} = -\ln\left(\sum_{k=0}^{K-1} \pi_k(\mathbf{x}^{(q)}) \phi_k(\mathbf{u}^{(q)}|\mathbf{x}^{(q)})\right)$$

The risk of significantly affecting the prediction of the nominal behaviour is minimal since they make up a large portion of the data. An additional weight is placed on the δ_q term because steering is more difficult to predict; not only does it depend on the presence of neighbouring cars, but also on human intent. For example, in a situation where it is safe to change lanes, not all drivers do so and this may depend on factors such as the driver needing to take the next exit in the next mile, or feeling more safe in lanes farther to the right.

Once the training is done, we can can obtain all the probability coefficients $\boldsymbol{\pi}_k$, $\boldsymbol{\mu}_k$ and $\boldsymbol{\sigma}_k$ given some model state $\mathbf{x}(t)$. The output of the neural network, $\mathbf{z} = (\mathbf{z}_\pi, \mathbf{z}_{\boldsymbol{\sigma}}, \mathbf{z}_{\boldsymbol{\mu}})$, will be a vector of length $(1+2M)K$ where K is the number of mixture models, M the number of control inputs. This is because of the mixing coefficients, standard deviations and means for each of the control inputs. To ensure that (4) is satisfied, the softmax function is used on $\mathbf{z}_\pi$, the π portion of $\mathbf{z}$. For the standard deviation, it is convenient to represent them in terms of exponentials, $\sigma_k = exp(z_{\sigma_k})$. While $\boldsymbol{\mu}_k$ is the mean of the control inputs. Within the Bayesian framework, this corresponds to the choice of an un-informative Bayesian prior [16].

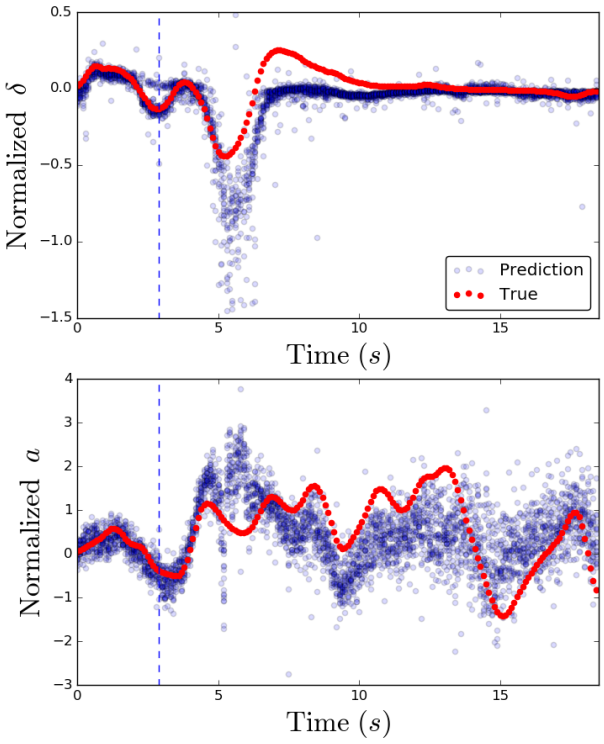


Fig. 5: Control actions predicted by the MDN model based on any $3-$nearest neighbours configuration. (Car ID 2133)

IV. Training Results

Since we are analysing highway driving data, a majority of the data will represent straight driving with almost zero acceleration and steering. A basis for the measure of a good prediction is to see how well the MDN model predicts interesting cases of highway driving. In particular, we shall consider the following interesting cars:

1) Car ID 2133: This car is approaching a traffic jam. It edges closer to the edge of the lane before executing a rapid lane change to avoid the jam. This car is in lane 4 and transitions right to lane 5
2) Car ID 181: The car is approaching a traffic jam, but as it is attempting to execute a lane change, the traffic jam frees up and the car remains in the lane and accelerates forward.

In the following sections, we will investigate different ways to construct the model state in order to best predict the interesting behaviour on highway driving. Since this is a preliminary investigation, we will analyse the results qualitatively and offer some intuition behind it. Log-likelihood is not a valid metric to use for performance because we

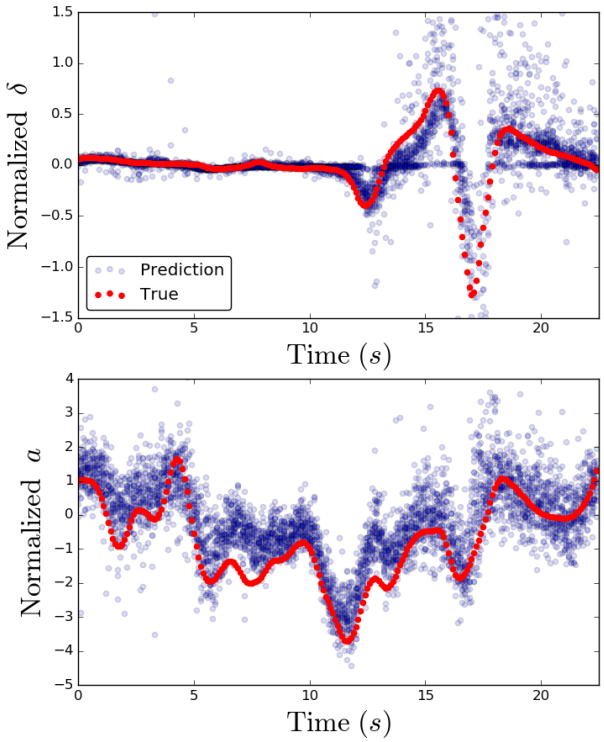


Fig. 6: Control actions predicted by the MDN model based on any $3-$nearest neighbours configuration. (Car ID 181)

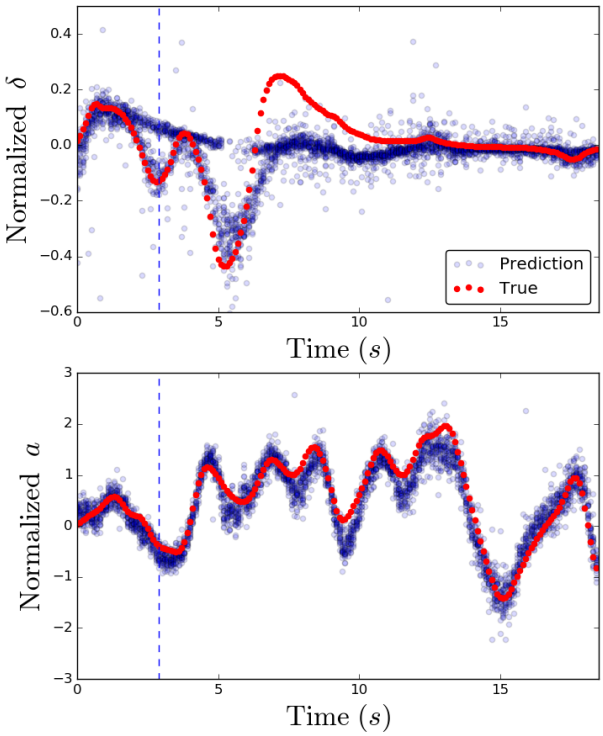


Fig. 7: Control actions predicted by the MDN model based on any $6-$nearest neighbours configuration. (Car ID 2133)

are concerned with only predicting a small portion of the overall data. Results are presented and accompanied with brief comments. In Section V a more in depth discussion and interpretation of the results is given.

From experimentation, it was found that a Long Short Term Memory (LSTM) network (a type of RNN) was the most efficient; it was able to offer similar results compared to using a Gated Recurrent Unit (GRU) network (another type of RNN) but with significantly fewer mixture density models (four as opposed to twenty). Thus this is the one that will be used for the following investigation.

A. *Any k-nearest Neighbours*

In this set-up the scene state consists of the vehicle states of the target vehicle and the vehicle states of *any* k-nearest neighbours in terms of euclidean distance. This set-up represents the perspective that the driver is only concerned with their immediate vicinity regardless of how the k-nearest neighbours are distributed. We shall consider two cases, (i) $3-$nearest neighbours and (ii) $6-$nearest neighbours. Case (i) is a simple case where the scene state is relatively small yet attempts to maintain a realistic number of nearest neighbours that drivers remain cognisant of. While case (ii) is considered to give a fair comparison with the other approach of decoupling longitudinal and lateral neighbours.

1) Any 3-nearest Neighbours: The three nearest vehicles were used for the scene state. Three was chosen to represent the simple ideal scenario of a driver keeping track of the vehicle directly in front, to the left and right. This is to represent a "near-sighted" driver. With this set-up, as seen in Fig. 5, it was able to capture the general trend of the steering input but the acceleration was not generally very accurate. Even though the prediction of acceleration followed the general trend of the true acceleration, it was not very confident in the prediction. For car ID 181, shown in Fig. 6, the MDN model predicts the steering actions of car ID 181 quite accurately, but like with car ID 2133, the acceleration not so well. A possible explanation for this better prediction in steering is car ID 181 is in an edge lane (lane 1) so the search space for the nearest three neighbours is smaller than cars not on the edge lanes. This suggests than in increase in k could potentially improve the results.

2) Any 6-nearest Neighbours: An increase in the number of nearest neighbours has the potential to capture cars that are several cars in front of the target vehicle. This offers the ability to anticipate the onset of a traffic jam while still possibly keeping some information about adjacent vehicles.

For car ID 2133, this set-up was able to predict the steering input and acceleration with much greater confidence than $3-$nearest neighbouts, evident in Fig. 8. There is a significant improvement in the acceleration prediction and the density in the steering for the lane changing portion is slightly different (will be discussed later). For car ID 181 on the other hand, the steering prediction is not as accurate but like the case with car ID 2133, the acceleration prediction is significantly more accurate. This is illustrated in Fig. 8. This strongly suggest

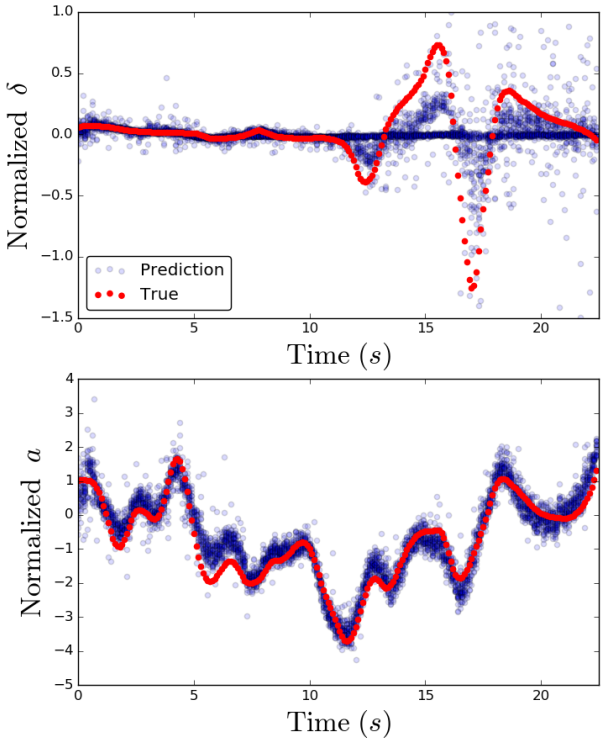


Fig. 8: Control actions predicted by the MDN model based on any $6-$nearest neighbours configuration. (Car ID 181)

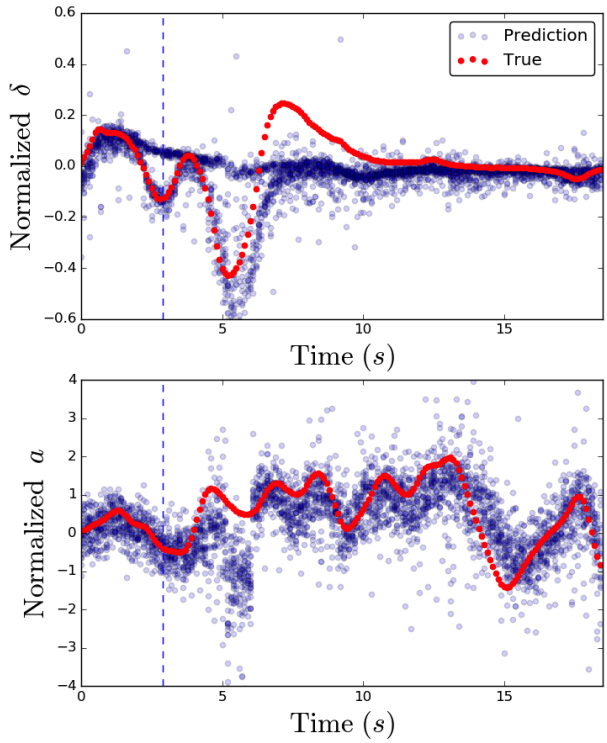


Fig. 9: Control actions predicted by the MDN model based on the nearest longitudinal and lateral neighbours configuration. (Car ID 2133)

that greater knowledge of the surrounding vehicles improve the acceleration prediction, but perhaps not necessarily in steering.

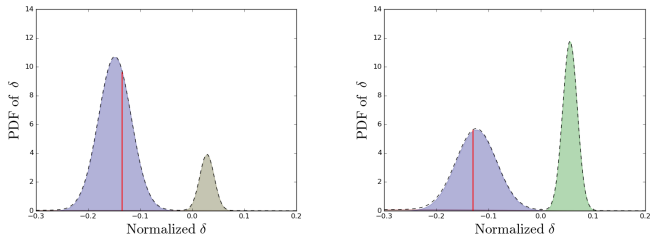
B. Longitudinal and Lateral k-nearest Neighbours

Here, we discriminate between longitudinal (same lane) and lateral (adjacent lanes) neighbours. To compare with the any $6-$nearest neighbours set-up, we look at the $3-$nearest longitudinal and lateral neighbours (total 6). The motivation for this is that it potentially maintains the driver's awareness of vehicles both in the same and adjacent lanes. Enforcing the separation between longitudinal and lateral vehicles give rise to the possible foresight of an upcoming traffic jam *and* possible gap for a lane change. For example, in Fig. 1 the target vehicle (blue) can see that there it is approaching a traffic jam since two of its three longitudinal nearest neighbours (green) are in front. At the same time, it notices a gap on its right adjacent lane since two of the three lateral neighbours (red) are on the right.

It was found that this gave similar, if not less reliable results than the $6-$nearest neighbour set-up. For brevity, we will only show the results for car ID 2133 and this is given in Fig. 9. A possible explanation may be that because we separate longitudinal and lateral neighbours, depending on the situation, not all the longitudinal or lateral vehicles will be relevant. While the any $6-$nearest neighbours set-up have the benefit that all the neighbours will be somewhat relevant.

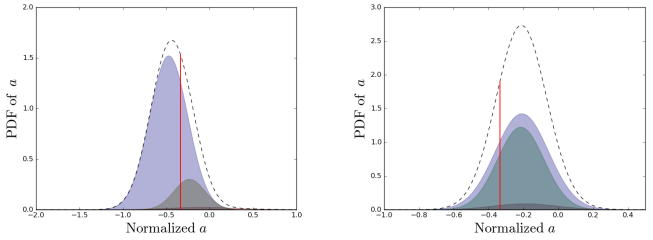
V. Discussion

In this section, we discuss the implications of the results and offer some interpretations. A key observation from results is the bimodal behaviour in the steering input of both vehicles (they both execute a lane change, or an attempt otherwise). Since we are predicting the current single control action given a sequence of states, we are neglecting correlation between the current and future control actions. Once the vehicle has begun its lane change, the heading angle is going to change (away from the heading of the road) due to the dynamic constraints. The next few time steps (0.1 seconds each) immediately after the vehicle has begun the steering, the heading angle is still quite close to nominal and so the MDN model also predicts steering to be around the nominal value. Once the heading angle is sufficiently large, our MDN model predicts that the driver is most likely executing a lane change thus the steering is most probably away from the nominal value (zero). Otherwise this would imply the vehicle is traverse diagonally across the lanes which is very unlikely and does not exist in the training data. This is most evident around the five and sixteen seconds mark of car ID 2133 and 181 respectively for the case with $3-$nearest neighbours. We can see that there is a lower probability density around the region when steering is zero. The $6-$nearest neighbour case gives a greater probability that the vehicle stays in the nominal driving action which does not abide to this logic



(a) 3−nearest neighbours (b) 6−nearest neighbours

Fig. 10: Steering input δ gaussian mixture model at 2.9 seconds of car ID 2133



(a) 3−nearest neighbours (b) 6−nearest neighbours

Fig. 11: Acceleration a gaussian mixture model at 2.9 seconds of car ID 2133.

and this could be a point for further investigation. This may perhaps be an artifact of over fitting. The probability density functions of this swerving action is given in Fig. 10. We can see that at the peak of the swerve, the 3−nearest neighbour model has a higher probability of the car swerving back which follows the above logic. However, using 6−nearest neighbours is very effective in predicting the acceleration as it has a higher probability density than the 3−nearest neighbours set-up. This is evident in Fig. 11. Further the behaviour is essentially unimodal. A plausible explanation is that if the vehicle ahead is close, the most obvious action the target vehicle must take is to slow down in order to avoid a collision.

VI. Conclusions

A. Future Work

Immediate future work would include addressing limitations to this MDN model and eventually applying it to a controls framework. This paper gives qualitative measures of performance of the model because it was only the outliers that we were interested in. A next step is to come up with a more systematic and quantitative measure which would include an unbiased method of parsing out interesting vehicle behaviours from the data. With better metrics for performance, we can tune the system better and produce more confident predictions. In addition, a more informative data set could potentially improve the our predictions such as knowledge of whether neighbouring vehicles have their turn signals on.

Further, the MDN model explored in this paper only predicts the current control actions given a sequence of past states. This does not offer information about the correlation between consecutive of control actions. A deeper investigation would include considering the prediction of the next N number of control actions. This distribution could possibly be learned by extending the output vector, but this may require more training data and would not be very scalable as N increases. Alternatively, since we enforce smoothness on our control actions through the EKF, a spline could be used to estimate the sequence of control actions. This reduces the dimensionality of the problem because it is only the coefficients that needs to be learned and the size is dependent on the degree of the spline used and not the prediction length. This is reminiscent of Gaussian Processes (GP) and in [24], the authors define Recurrent Gaussian Processses (RGP) models, a general family of Bayesian nonparametric models with recurrent GP priors which are able to learn dynamics patterns from sequential data. We can potentially apply this model for future work.

Once the driving model is developed, the next step is to formulate decision-making control policies that optimises over some cost and test it in interesting scenarios. Based on a distributional model, we can possibly design control policies through a Bayesian game framework which is essentially a game of imperfect information. Ways to validate the policy would involve having a human driver be surrounded by all autonomous vehicles which we can control. For example, in Fig. 1, the blue car could represent the target human driver and the all the neighbouring green and red cars can be autonomous vehicles which we can control. We can direct the autonomous vehicles in certain configurations and see if the human behaves in the manner we expect it to. Further, we can use the fleet of autonomous vehicles to interact and to some extent manipulate the human driver into exhibiting a certain behaviour, similar to the validation testing done in [11]. This would represent a situation where we have very high control authority and further extensions could include situations with less control authority and where there are multiple human drivers that share the same neighbouring autonomous vehicle.

B. Conclusions

A recurrent MDN model was used to predict actions drivers are likely to take when driving on a highway. Possible ways to construct the model state was discussed and it was found that taking any 6−nearest neighbours was effective in predicting acceleration and to some extent the steering input. Equipped with intuition of what the results represent, further complexities can be added to potentially improve the model. In particular predicting control actions multiple time steps ahead. With this probabilistic model of human driving behaviours, future work will be directed towards designing quantitative measures for performance and constructing a more complex MDN model. Ultimately, we aim to design a control policy that anticipates the intent of human drivers and have methods to validate this.

REFERENCES

[1] Federal Highway Administration U.S. Department of Transportation. Average annual miles per driver by age group, 2016.

[2] Pravin Varaiya. Smart cars on smart roads: Problems of control. In *IEEE Transactions on Automatic Control*, volume 38, February 1993.

[3] Maria E. Jabon, Jeremy N. Bailenson, Emmanuel D. Pontikakis, Leila Takayama, and Clifford Nass. Facial expression analysis for predicting unsafe driving behavior. *IEEE Pervasive Computing*, 10(4):84–95, 2011.

[4] Eric Wahlstrom, Osama Masoud, and Nikos Papanikolopoulos. *Vision-Based methods for driver monitoring*, volume 2, pages 903–908. Institute of Electrical and Electronics Engineers Inc., 2003.

[5] Sergey Levine and Vladlen Koltun. Continuous inverse optimal control with locally optimal examples. In *ICML '12: Proceedings of the 29th International Conference on Machine Learning*, 2012.

[6] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 19–27. Curran Associates, Inc., 2011.

[7] Karkus Kudere, Shilpa Gulati, and Wolfram Burgard. Learning driving styles for autonomous vehicleas from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2015.

[8] W. Dong, J. Li, R. Yao, C. Li, T. Yuan, and L. Wang. Characterizing Driving Styles with Deep Learning. *ArXiv e-prints*, July 2016.

[9] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *In Proceedings of the Twenty-first International Conference on Machine Learning*. ACM Press, 2004.

[10] Brian D. Ziebart, Andrew Maas, J. Andrew (Drew) Bagnell, and Anind Dey. Maximum entropy inverse reinforcement learning. In *Proceeding of AAAI 2008*, July 2008.

[11] Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. Planning for autonomous cars that leverage effects on human actions. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

[12] Rufus Isaacs. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. Dover books on mathematics. Dover Publications, 1999.

[13] Georges Aoude, Brandon Luders, Joshua Joseph, Nicholas Roy, and Jonathan P. How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Auton. Robots*, 35(1):51–76, 2013.

[14] Steven M LaValle. *Planning Algorithms*. Cambridge, 2004.

[15] Enric Galceran, Alexander G. Cunningham, Ryan M. Eustice, and Edwin Olson. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction. In *Proceedings of Robotics: Science and Systems (RSS)*, Rome, Italy, July 2015.

[16] Christopher M. Bishop. Mixture density networks. Technical report, 1994.

[17] G.J. McLachlan and K.E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, 1988.

[18] Tamer Basar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory*. Academic Press, 1999.

[19] John Harsanyi. Games with incomplete information played by "bayesian" players, i-iii part i. the basic model. *Management Science*, 14(3):159–182, 1967.

[20] Federal Highway Administration U.S. Department of Transportation. Next generation simulation (ngsim): Us highway 101 dataset, 2016.

[21] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[22] Simon Haykin. *Kalman Filtering and Neural Networks*. Published Online, 2002.

[23] Oliver Bauchau. *Flexible Multibody Dynamics*. Springer, 2011.

[24] César Lincoln C Mattos, Zhenwen Dai, Andreas Damianou, Jeremy Forth, Guilherme A Barreto, and Neil D Lawrence. Recurrent Gaussian processes. *International Conference on Learning Representations (ICLR)*, 2016.