Driver Modeling Using Continuous Reasoning Levels: A Game Theoretical Approach

Cem Okan Yaldiz[1] and Yildiray Yildiz[2]

***Abstract*— The focus of this paper is designing a game theoretical method using a continuous policy space for modeling human driver interactions on highway traffic. The proposed method is based on Gaussian Processes and developed as an enrichment to the hierarchical decision-making concept called "level-k reasoning". This concept conventionally assigns discrete levels of behaviors to agents. Although shown to be an effective modeling tool, the level-k reasoning approach may pose undesired constraints for predicting human decision making due to a limited number (usually 2 or 3) of driver policies it provides. The proposed approach is put forward to fill this gap in the literature by introducing a continuous domain framework that enables an infinite policy space. By using the approach presented in this paper, more accurate driver models are obtained, which can be employed for creating high-fidelity simulation platforms for the validation of autonomous vehicle control algorithms. The proposed method is validated on a traffic dataset and compared with the conventional level-k approach to demonstrate its contributions and implications.**

I. INTRODUCTION

Despite promising advances, the emergence of autonomous vehicles (AV) in regular commercial use is still pending mainly due to safety issues. Their trustworthiness must be validated before AVs become a usual sight at everyday traffic [1]. In [2], it is stated that to prove AVs' safety, hundreds of millions of miles are needed to be driven without fatality, which corresponds to years of observation. This particular result makes it difficult to validate AV technology by conducting only real road tests. Also, observing AV control algorithm behaviour in several different scenarios, including the edge cases, is a non-trivial task. On the other hand, computer simulations decrease the required time and effort significantly, and help create complex environments for assessing AVs' reactions in various test cases. Therefore, creating reliable simulation environments that can mimic the real world has utmost importance for the validation of AV control algorithms. To achieve this, modeling human driver interactions as accurate as possible is necessary.

Joint employment of reinforcement learning (RL) and game theory is demonstrated to be an effective tool for modeling human interactions. Examples for the utilization of this method to predict human-human and human-automation interactions can be found in [3], [4] and [5] for aviation, and in [6], [7] and [8] for automotive applications. The game theoretical approach used in these studies is called the "level-k reasoning" [9], [10], [11], [12]. This method is based on the idea that humans have different levels of decision making algorithms, where level-k is a best response to level-(k-1). Although level-k reasoning is shown to have a reasonable match with real human driving behavior [13], it suffers from the limited number of driver policies it offers. In this paper, we propose a solution to this problem. Our solution uses the limited policies suggested by the conventional level-k approach as inputs to a multi-output Gaussian Process (GP) to obtain driver levels in a continuous space. In other words, what distinguishes the method presented in this paper from the earlier ones is that we abandon the belief that humans can be successfully modeled via only discrete levels. Our contributions in this paper are as follows:

1) We introduce continuous reasoning levels to the level-k reasoning concept, which enables us to use an infinite policy space for modeling human behavior.
2) Continuous driver models are tested on a highway dataset and statistical analysis results are provided on the model prediction success. Furthermore, the results are compared with the earlier work using benchmark cases.

This paper is organized as follows: In Section II, we explain how conventional level-k reasoning is used to model human drivers, and we describe how discrete (integer-valued) levels are obtained using a specific highway scenario. We describe single-output and multi-output GPs in Section III. In Section IV, we introduce the method to obtain the real-valued levels that constitute the infinite driver policy space. In Section V, we detail how we validate our approach on traffic data. Validation results are shared in Section VI. Finally, a summary is provided in Section VII.

II. DRIVER MODELING USING CONVENTIONAL LEVEL-K REASONING CONCEPT

In this section, we explain how driver models are obtained using conventional level-k reasoning, $k \in \mathbb{N}$, together with deep reinforcement learning. We also provide the details of the high-way traffic scenario we are interested in.

A. Level-k Reasoning and Deep Reinforcement Learning

The concept of level-k reasoning assumes that humans have different levels of reasoning in decision-making. In this framework, a level-0 agent is a "non-strategic" agent and only follows predetermined rules. For $k > 1$, hierarchically, a level-k agent chooses its actions by maximizing its rewards based on the assumption that other agents in the environment are following a level-(k-1) reasoning.

[1]Cem Okan Yaldiz is with the Department of Electrical and Computer Engineering, Robotics, Georgia Institute of Technology, Atlanta, GA 30332, USA cyaldiz3@gatech.edu

[2]Yildiray Yildiz is with Faculty of Mechanical Engineering, Bilkent University, Ankara, 06800, Turkey yyildiz@bilkent.edu.tr

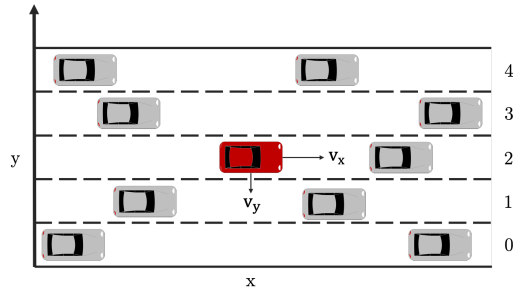

Fig. 1: Highway scenario with 5 lanes [13]

To obtain the best responses to level-(k-1) drivers, we use RL to maximize the level-k agent's utility in the time extended scenario. Since we have a large state space, instead of a tabular RL approach, we prefer to use the deep Q-network (DQN) [14]. In-depth description for the conjoint usage of level-k reasoning and DQN along with the network structure used in this paper can be found at [13].

B. Observation Space

The traffic scenario we study is given in Figure 1, where the ego vehicle is shown with red. We assume that a driver in lane h can observe the closest front and rear cars in lanes $h-2, h-1, h+1, h+2$, and the front car in lane h, which makes maximum 9 cars observable at a given instant. Let Δ_p and Δ_v denote the relative position and velocity, respectively. Then, the driver in lane h knows

1) The lane number h he/she is in,
2) Δ_p and Δ_v of the car in front him/her in lane h, and
3) Δ_p and Δ_v of both of the cars in front and in rear, in lanes $h+1, h-1, h+2$ and $h-2$.

The relative position Δ_p is quantized as *close*, *nominal* and *far*. Similarly, Δ_v is quantized as *stable*, *approaching* and *moving away*. Quantization intervals are determined to be representative of the selected scenario (see [13]), and are given as *close* if $\Delta_p < 11m$, *nominal* if $11m \leq \Delta_p < 27m$, *far* if $\Delta_p \geq 27m$, *approaching* if $\Delta_v < -0.1m/s$, *stable* if $-0.1m/s \leq \Delta_v < 0.1m/s$, and *moving away* if $\Delta_v \geq 0.1m/s$.

C. Action Space

There are two lateral actions and five longitudinal actions. The lateral actions *moving to the left lane* and *moving to the right lane* assumed to take 1 second. The five longitudinal actions are sampled from continuous distributions specified in [13].

D. Reward Function

The reward function R is defined as

$$R = w_1 c + w_2 s + w_3 d + w_4 e, \tag{1}$$

where the weights w_i are selected such that $w_1 > w_4 > w_3 > w_2$. The reward function terms are determined as given below.

- c: -1 if a crash occurs.
- s: The difference between the driver's speed and the average of the minimum speed and the maximum speed, normalized by the maximum speed.
- d: -1 if the car in front is in *close* position, 0 if the car in front is in *nominal* position, 1 otherwise.
- e: 0 if the action is *maintain*, -0.25 if the action is *accelerate* or *decelerate*, -0.5 if the action is *hard accelerate* or *hard decelerate*, -1 if the action is *moving to the left lane* or *moving to the right lane*.

E. Agent training

The scenario given in Figure 1 is used for the simulation, and the training is done in an episodic fashion. An episode is terminated if a crash occurs, and a new one is initialized. When training a level-k agent, 125 level-(k-1) drivers are placed into the environment including the ego agent. The policy of the level-0 driver, which acts as an anchor for higher levels, is determined with the following rules:

1) *hard decelerate* if the car in front is *close* and *approaching*.
2) *decelerate* if the car in front is *close* and *stable* or *nominal* and *approaching*.
3) *accelerate* if the car in front is *nominal* and *moving away* or *far*.
4) *maintain* otherwise.

III. GAUSSIAN PROCESSES

In this section, we explain two types of Gaussian Processes (GP), namely, the single output and multi-output GPs.

A. Single Output Gaussian Processes

A GP is a random process for which the joint distribution of any finite subset of random variables follows a multivariate Gaussian distribution. A function $f : \mathbb{R} \to \mathbb{R}$ that is modeled by a GP is expressed as

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x')), \tag{2}$$

where $\mu(x)$ is the mean function and $k(x, x')$ is the covariance function (kernel) which can be explicitly written as

$$\mu(x) = E[f(x)] \tag{3}$$

$$k(x, x') = cov(f(x), f(x')), \tag{4}$$

where $E[\ \ ]$ and $cov()$ represent the expectation and covariance, respectively. Any n samples chosen from the process will also be jointly Gaussian, which can be stated as

$$\begin{bmatrix} f(x_1) \\ . \\ . \\ . \\ f(x_n) \end{bmatrix} \sim N(\begin{bmatrix} \mu(x_1) \\ . \\ . \\ . \\ \mu(x_n) \end{bmatrix}, \Sigma) \tag{5}$$

where $\Sigma \in \mathbb{R}^{n \times n}$ is the covariance matrix containing $k(x_i, x_j)\ i, j = 1, 2, ..., n$, at the i^{th} row and j^{th} column. A valid Σ is positive definite, symmetric and invertible.

Given a GP, which is created by using observed samples $\mathcal{D} = \{(x_i, f(x_i))\},\ i = 1, ..., n$ with the mean $\mu(x)$ and

the kernel $k(x, x')$, we want to make predictions of $f(x_j^*)$ at points x_j^*, $j = 1, 2, ..., m$. The output vectors $\boldsymbol{f} = [f(x_1), ..., f(x_n)]^T$ and $\boldsymbol{f^*} = [f(x_1^*), ..., f(x_m^*)]^T$ are jointly Gaussian and their distribution can be expressed as

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f^*} \end{bmatrix} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{6}$$

where

$$\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma & \Sigma_*^T \\ \Sigma_* & \Sigma_{**} \end{bmatrix}, \boldsymbol{\mu} = \begin{bmatrix} \mu(x) \\ \mu(x^*) \end{bmatrix}. \tag{7}$$

The covariance sub-matrices $\Sigma \in \mathbb{R}^{n \times n}$, $\Sigma_* \in \mathbb{R}^{m \times n}$ and $\Sigma_{**} \in \mathbb{R}^{m \times m}$ contain $k(x_i, x_j)$, $k(x_i^*, x_j)$ and $k(x_i^*, x_j^*)$, respectively, as elements.

B. Multi-output Gaussian Processes

When the problem is to map an input $x \in \mathbb{R}^P$ to an output $y = f(x)$ using the mapping $f : \mathbb{R}^P \rightarrow \mathbb{R}^D$, the covariance matrix should be expanded to include the covariances between D outputs too. In this work, we use "Coregionalized Regression" approach to take into account the covariances between the outputs [15]. With this approach, each output of the multi-output GP is treated as a function that is a linear combination of different samples from the underlying single-output GPs whose priors are constructed by the specification of the kernel.

Let us assume that we have observations $X_d = [x_1^d, ..., x_{m_d}^d]$, where $x_i^d \in \mathbb{R}^P$, $i = 1, \ldots, m_d$, for the d^{th} function, $f_d \in \mathbb{R}$, where m_d stands for the number of observations for that function, and the corresponding outputs are $F_d = [f_d(x_1^d), ..., f_d(x_{m_d}^d)]$. The covariance matrix that encapsulates all the relations within the process is

$$\boldsymbol{\Sigma}(X, X) = \begin{bmatrix} \Sigma(X_1, X_1) & \ldots & \Sigma(X_1, X_D) \\ \vdots & \ddots & \vdots \\ \Sigma(X_D, X_1) & \ldots & \Sigma(X_D, X_D) \end{bmatrix}. \tag{8}$$

The diagonal terms in (8) modeling the auto-covariances can be constructed within the framework of single-output GP. What matters for the multi-output framework is the cross-covariance terms $\Sigma(X_i, X_j), i \neq j$. To model these matrices, "linear model of coregionalization" (LMC) can be used. Consider the set of functions $\{f_d\}_{d=1}^D$. In the LMC framework, these functions are modeled as

$$f_d(x) = \sum_{z=1}^{Z} \sum_{i=1}^{R_z} a_{d,z}^i u_z^i(x), \tag{9}$$

where Z is the number of kernels, $u_z^i(x)$ are independent latent functions with $cov[u_z^i(x), u_{z'}^{i'}(x)] = k_z(x, x')$ if $i = i'$ and $z = z'$, and zero, otherwise, $a_{d,z}^i$ are scalar coefficients, and R_z is the number of latent functions that share the same covariance. The covariance of two functions $k(x, x')_{d,d'} = cov[f_d(x), f_{d'}(x)]$ can be calculated as

$$k(x, x')_{d,d'} = \sum_{z=1}^{Z} \sum_{i=1}^{R_z} a_{d,z}^i a_{d',z}^i cov(u_z^i, u_{z'}^{i'}) \tag{10}$$

$$= \sum_{z=1}^{Z} \sum_{i=1}^{R_z} a_{d,z}^i a_{d',z}^i k_z(x, x') \tag{11}$$

$$= \sum_{z=1}^{Z} b_{d,d'}^z k_z(x, x'), \tag{12}$$

where $b_{d,d'}^z = \sum_{i=1}^{R_z} a_{d,z}^i a_{d',z}^i$. The covariance matrix is then determined as

$$k(x, x') = cov[f(x), f(x')] = \sum_{z=1}^{Z} B_z k_z(x, x'), \tag{13}$$

where $B_z \in \mathbb{R}^{D \times D}$ has $b_{d,d'}^z, d = 1, 2, ..., D, d' = 1, 2, ..., D$, as its $(d, d')^{th}$ element. Furthermore, B_z is rank R_z, and called the "coregionalization matrix" measuring interrelations of different outputs. Transitioning to matrix form for multiple inputs, we can write

$$\boldsymbol{\Sigma}(X, X) = \sum_{z=1}^{Z} B_z \otimes k_z(X, X), \tag{14}$$

where the symbol $\otimes$ stands for the Kronecker product.

For an observation set $\mathcal{D}$ containing D different functions with N observations from each, the posterior mean at a single point x^* is

$$\mu_{f^*|\mathcal{D}} = \Sigma_* \Sigma^{-1} f, \tag{15}$$

where $\Sigma \in \mathbb{R}^{ND \times ND}$ is the matrix in (8), $\Sigma_* \in \mathbb{R}^{D \times ND}$ has elements $k(x_i, x^*)_{d,d'}$ for $i = 1, \ldots, N$, and $d, d' = 1, \ldots, D$, and $f \in \mathbb{R}^{ND \times 1}$ is the output vector containing the outputs of D functions at N points.

IV. CONTINUOUS HIERARCHICAL MODELLING

Conventional level-k reasoning includes only the discrete levels which may be an obstacle for modeling certain behaviors. For instance, at a particular state, a level-0 driver might prefer the action "turn right" with a probability of 0.8, while a level-1 driver might take the same action with a probability of 0.15. This framework may not be representative enough for a driver who takes the same action with 0.4 probability. This limitation motivates us to expand the levels from the discrete domain to a continuous domain. In this study, we model continuous levels by employing a GP, which accepts discrete levels' action probability distributions (policies), obtained through RL, at every state of a particular driver, and form policies in continuous domain. These policies belong to real-valued reasoning levels, instead of integer-valued ones, in the new framework. This process is explained in the following sections. Throughout the paper, we use $(.)_l$, $l \in \mathbb{R}^+ \cup \{0\}$, and $(.)_k$, $k \in \mathbb{N}$, for "real-valued" and "integer-valued" concepts, respectively.

A. The Method

When we train a level-k agent using DQN, for each state $s \in \mathcal{S}$ we obtain a set of Q-values, where each value in this set belongs to an action $a \in \mathcal{A}$. The "Q-function" for each reasoning level-k, approximated by the corresponding DQN for level-k and denoted as $Q_k(s,a)$, is a mapping $Q_k : S \rightarrow \mathbb{R}^A$, where $A = |\mathcal{A}|$ is the number of available actions. We have a set $Q = \{Q_1, ..., Q_n\}$ corresponding to n different integer-valued levels. The policy of a level-k agent, denoted as $\pi_k(a_i|s)$, $a_i \in \mathcal{A}$, is defined as a probability distribution over actions conditioned on the given state and it is computed by a softmax function with

$$\pi_k(a_i|s) = \frac{e^{Q_k(s,a_i)}}{\sum_j e^{Q_k(s,a_j)}}, \tag{16}$$

for each a_i. Therefore, at a state s, we have a set $\pi = \{\pi_0, \pi_1, ..., \pi_n\}$ which is computed by using the set of Q-values through (16) where $\pi_k \in \mathbb{R}^A, k = 0, \ldots, n$.

Remark. Level-0 policy is generally hand crafted and not determined by an RL process. Formation of π_0 depends on the specific structure of the level-0 policy. For example, if at a particular state s, the level-0 agent deterministically chooses action $a_j, j \in \{1, 2, ..., A\}$, the policy of the agent is given as $\pi_0(a_j|s) = 1$, and $\pi_0(a_i|s) = 0\ \forall i \neq j$.

For each state s, we define a GP by using the realizations of π at discrete levels 0, 1, ..., n, denoted as $\pi_0(\mathbf{a}|s), \pi_1(\mathbf{a}|s)..., \pi_n(\mathbf{a}|s)$, $\mathbf{a} \in \mathbb{R}^A$. Therefore, the observation set $\mathcal{D}$ is defined as $\mathcal{D} = \{\pi_0(\mathbf{a}|s), \pi_1(\mathbf{a}|s), \ldots, \pi_n(\mathbf{a}|s)\}$. For a given state $s \in \mathcal{S}$, the distribution of the policy $\pi_l : \mathcal{S} \rightarrow \mathbb{R}^A$, which is the policy for any real-valued level l, can be calculated as

$$\pi_l(\mathbf{a}|s) = \begin{bmatrix} \pi_l(a_1|s) \\ \vdots \\ \pi_l(a_A|s) \end{bmatrix} \sim \mathcal{N}(\Sigma_* \Sigma^{-1} \boldsymbol{f}, \Sigma_{**} - \Sigma_* \Sigma^{-1} \Sigma_*^T), \tag{17}$$

through the GP framework, where $\pi_l(\mathbf{a}|s)$ is the vector containing the probabilities of all actions $a_i, i = 1, 2, ..., A$, at the state s, at level l, and $\boldsymbol{f} = [\pi_0(\mathbf{a}|s), ...\pi_n(\mathbf{a}|s)]$ is the observation vector consisting of the policies related to the integer-valued level-ks, $k = 1, 2, ..., n$. The covariance submatrices Σ and Σ_* are defined in (15). Σ_{**} is defined in (10) and has the elements $k(l,l)_{a,a'}$ for $a, a' = 1, \ldots, A$.

The function with the highest probability in the GP is the mean function. In other words, let $p(f)$ denote the probability of the function $f \sim GP(\mu, \Sigma)$. Then,

$$\arg\max_f p(f) = \mu. \tag{18}$$

Therefore, since a GP is a probability distribution over functions, we propose the assignment of the predictive mean (see (15)) as the policy for level-l. Then, at any real-valued level l^*, the corresponding policy is represented by $\widehat{\pi}_{l^*}(\mathbf{a}|s) = \mu(\pi_{l^*}(\mathbf{a}|s)) \in \mathbb{R}^A$.

Remark. In our method, GP nonlinearly predicts the policy of a real-valued level which is a probability distribution over actions. Therefore, for a real-valued level l, the constraint

$$\sum_{i=1}^{A} \widehat{\pi}_l(a_i|s) = 1 \tag{19}$$

should be satisfied. In [16], it is proved that for linear constraints, GP satisfies the constraint in its prediction when the inputs used in training also satisfies the linear constraint. Therefore, the policy constructed by GP satisfies (19). However, even though the constraint in (19) is satisfied, some of the action probabilities may be assigned negative values. To handle this issue, we shift and normalize the probabilities to satisfy (19). For instance, at a real-valued level l^* where GP predicts a negative value for one of the actions, we find

$$p_{shift} = \min_i \widehat{\pi}_{l^*}(a_i|s). \tag{20}$$

Then, the probabilities are updated as

$$\widehat{\pi}^{new}_{l^*}(a_i|s) \leftarrow \frac{\widehat{\pi}_{l^*}(a_i|s) + |p_{shift}|}{\sum_j [\widehat{\pi}_{l^*}(a_j|s) + |p_{shift}|]}. \tag{21}$$

Operation (21) defines a linear transformation on the probability value in the form

$$\widehat{\pi}^{new}_{l^*}(a_i|s) = \beta_1 \widehat{\pi}_{l^*}(a_i|s) + \beta_2, \tag{22}$$

and since the joint gaussian distribution property of the process is invariant under linear transformations, (21) can be employed, without violating GP properties.

V. Validation with Real Traffic Data

In this section, we use US101 data [17] to show how representative the real-valued reasoning levels are in terms of modeling driver behavior. To obtain the continuous, real-valued driver reasoning levels, we first need the conventional level-k models. We employ these discrete levels as inputs to the GP to obtain real-valued continuous levels, using the process given in Section IV. We use Kolmogorov-Smirnov (K-S) goodness of fit test [18], to compare the policies of the proposed model and that of real human drivers.

A. Construction of Gaussian Process

In constructing the multi-output GP in order to create the policy space, we chose Matérn kernel due to its finite differentiability, which makes it a more reasonable choice for most dynamical systems [19]. The Matérn kernel we used is given as

$$k(x,x') = \sigma^2(1 + \frac{\sqrt{3}(x-x')}{\beta})exp(-\frac{\sqrt{3}(x-x')}{\beta}) \tag{23}$$

where $\beta \in \mathbb{R}^+$ and $\sigma \in \mathbb{R}^+$ are the length scale and variance parameters, respectively. For optimization of the parameters and constructing the GP, we use GPy library [20]. The method presented in this paper uses the LMC model [15] with $R_Z = 7$ and $Z = 7$. Within these 7 kernels, one is a "bias kernel" that adjusts the mean of the function put into the GPy whereas the other six kernels are Matérn kernels

with different β constraints. We constrained β in six Matérn kernels to the values in the set $\{0.25,0.5,0.75,1.0,1.25,1.5\}$.

B. Comparison of Continuous Game Theoretical Policies and Policies Extracted from Driver Data.

We first provide definitions that we will be using extensively throughout the following sections. Then, we will detail the approach used in the comparison phase.

Definition 1. Discrete (integer-valued) game theoretical (DGT) policy is a probability distribution over actions. We use DGTs for comparison purposes. To see how DGTs are obtained, see [13]. In this paper, the discrete level policies are level-0, level-1, level-2 and level-3 policies. It is noted that these policies use a discrete observation space and a continuous action space sampled from discrete bins.

Definition 2. Continuous (real-valued) game theoretical (CGT) policy, the one proposed in this paper, is also a probability distribution over actions. However, the policies can be of a reasoning level-l, where l is a real number in the interval $[0, 3]$. It is noted that CGT policies use the same discrete observation space and the continuous action space sampled from discrete bins as DGTs.

Real driver policies are found by calculating the frequencies of the actions chosen by the driver at a given state. Once the driver policy is obtained from data, this policy is compared with the continuous policies created by the GP using the K-S test. K-S test measures the similarity of the policy at that continuous level with the real traffic data policy. During the comparison, a search over the continuous policy space is conducted using simulated annealing [21]. While the continuous levels within the interval $[0, 3]$ form the search space, the K-S test critical value is used as the cost function of simulated annealing. The continuous level that has the highest critical value, i.e. the one that is closest to the traffic data policy, is selected.

VI. Results and Discussion

Figure 2 shows the percentage of successfully modeled real driver states using DGT and CGT policies. The x-axes correspond to the driver ID numbers whereas the y-axes display the percentage of correctly modeled states. Therefore, each vertical line corresponds to the successfully modeled state percentage for an individual driver. The figure shows that CGT can model a significantly higher percentage of states, as expected. This is not a surprising result since CGT contains an infinitely large policy space while DGT contains only discrete levels. This figure is provided to demonstrate the flexibility obtained by employing real-valued reasoning levels instead of integer-valued ones.

What is more interesting to observe is the distribution of levels among real drivers. Figure 3 shows how this distribution appears for each driver in the traffic data. Each yellow dot on the figure indicates a certain level assigned for a given state of an individual driver. It is observed that most of the drivers' reasoning levels are accumulated between level-0 and level-1.75. The significant gap at level-1.75 and above indicates that in general human drivers in this dataset act as if other drivers in the traffic do not have reasoning levels of 1 or above. Curiously, this finding is in harmony with the experimental results reported in [22], [23], where it is found that in Beauty Contest game, humans behave mainly as level 1 and 2 and rarely level-3. In [23], unlike in this work, the level-0 reasoning is a simple strategy corresponding to an average guess. On the other hand, we choose a level-0 policy that can cope with different traffic scenarios by assigning proper actions for a given state. Consequently, it is a more complex strategy in comparison to [23]. This may explain the observation of real valued levels between 0 and 1.

VII. Summary

In this study, we propose an enrichment to level-k game theoretical reasoning by introducing continuous levels through Gaussian Processes. This fills a gap in level-k reasoning literature, where only a limited set of discrete levels for modeling is available. The proposed approach is validated on real traffic data using Kolmogorov-Smirnov goodness of fit test, where it is shown that the presented approach models the majority of the driver states correctly. It is also demonstrated that most drivers embrace a behavioral reasoning level that is consistent with the previous studies.

VIII. Acknowledgement

This work was supported by Turkish Academy of Sciences (TUBA) the Young Scientist Award Program.

References

[1] D. Fernández Llorca and E. Gómez, "Trustworthy autonomous vehicles," tech. rep., Joint Research Centre (Seville site), 2021.

[2] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?," *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[3] Y. Yildiz, A. Agogino, and G. Brat, "Predicting pilot behavior in medium-scale scenarios using game theory and reinforcement learning," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 4, pp. 1335–1343, 2014.

[4] B. M. Albaba and Y. Yildiz, "Modeling cyber-physical human systems via an interplay between reinforcement learning and game theory," *Annual Reviews in Control*, vol. 48, pp. 1–21, 2019.

[5] B. M. Albaba, N. Musavi, and Y. Yildiz, "A 3d game theoretical framework for the evaluation of unmanned aircraft systems airspace integration concepts," *Transportation Research Part C: Emerging Technologies*, vol. 133, 2021.

[6] C. Koprulu and Y. Yildiz, "Act to reason: A dynamic game theoretical driving model for highway merging applications," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 747–752, IEEE, 2021.

[7] S. Karimi and A. Vahidi, "Receding horizon motion planning for automated lane change and merge using monte carlo tree search and level-k game theory," in *2020 American Control Conference (ACC)*, pp. 1223–1228, IEEE, 2020.

[8] M. Jain, K. Brown, and A. K. Sadek, "Multi-fidelity recursive behavior prediction," *arXiv preprint arXiv:1901.01831*, 2018.

[9] C. F. Camerer, T.-H. Ho, and J.-K. Chong, "A cognitive hierarchy model of games," *The Quarterly Journal of Economics*, vol. 119, no. 3, pp. 861–898, 2004.

[10] D. O. Stahl and P. W. Wilson, "On players models of other players: Theory and experimental evidence," *Games and Economic Behavior*, vol. 10, no. 1, pp. 218–254, 1995.

[11] V. P. Crawford and N. Iriberri, "Level-k auctions: Can a nonequilibrium model of strategic thinking explain the winner's curse and overbidding in private-value auctions?," *Econometrica*, vol. 75, no. 6, pp. 1721–1770, 2007.

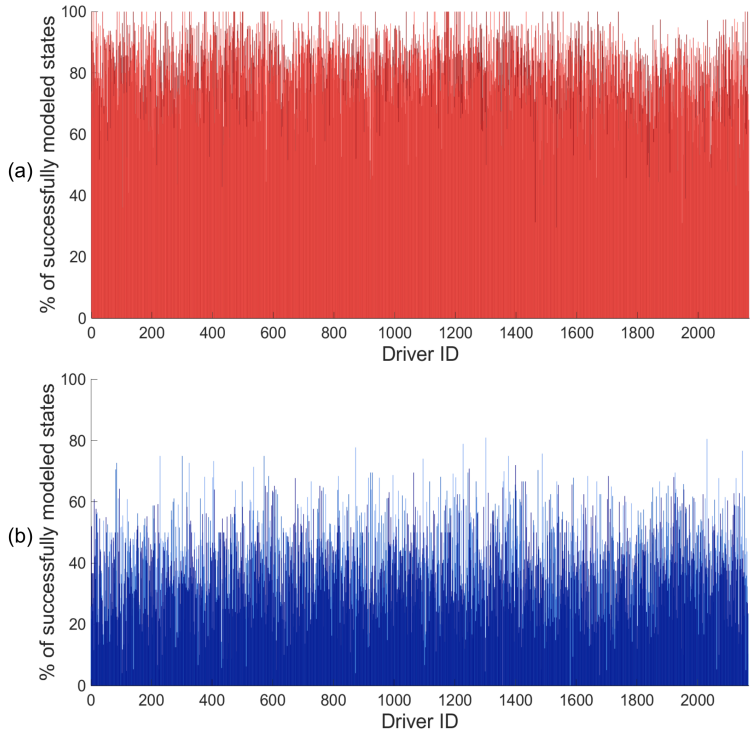

Fig. 2: Percentage of successfully modeled states, for each individual driver, by (a) CGT and (b) DGT policies.

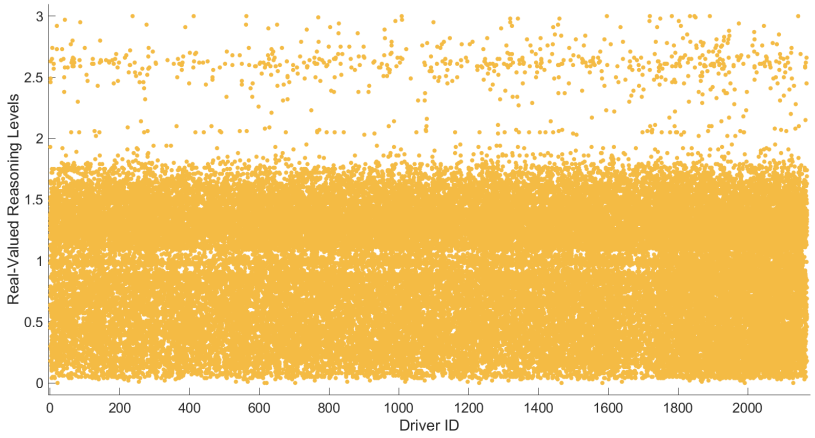

Fig. 3: For each driver, detected levels for all states that are correctly modeled are shown with markers.

[12] M. A. Costa-Gomes, V. P. Crawford, and N. Iriberri, "Comparing models of strategic thinking in van huyck, battalio, and beil's coordination games," *Journal of the European Economic Association*, vol. 7, no. 2-3, pp. 365–376, 2009.

[13] B. M. Albaba and Y. Yildiz, "Driver modeling through deep reinforcement learning and behavioral game theory," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 2, pp. 885–892, 2022.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[15] M. A. Alvarez, L. Rosasco, and N. D. Lawrence, "Kernels for vector-valued functions: A review," *arXiv preprint arXiv:1106.6251*, 2011.

[16] M. Salzmann and R. Urtasun, "Implicitly constrained gaussian process regression for monocular non-rigid pose estimation," *Advances in Neural Information Processing Systems*, vol. 23, pp. 2065–2073, 2010.

[17] U.F.H. Administration,, "Us101 dataset." https://www.fhwa.dot.gov/publications/research/operations/07030. Accessed: 2010-09-30.

[18] W. J. Conover, "A kolmogorov goodness-of-fit test for discontinuous distributions," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 591–596, 1972.

[19] M. L. Stein, *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media, 1999.

[20] GPy, "GPy: A gaussian process framework in python." http://github.com/SheffieldML/GPy, since 2012.

[21] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[22] C. F. Camerer, *Behavioral game theory: Experiments in strategic interaction*. Princeton university press, 2011.

[23] R. Nagel, "Unraveling in guessing games: An experimental study," *The American Economic Review*, vol. 85, no. 5, pp. 1313–1326, 1995.