2019 IEEE Intelligent Vehicles Symposium (IV)
Paris, France. June 9-12, 2019

Driving Behavior Modeling Based on Hidden Markov Models with Driver's Eye-Gaze Measurement and Ego-Vehicle Localization

Naoki Akai[1], Takatsugu Hirayama[2], Luis Yoichi Morales[2], Yasuhiro Akagi[2], Hailong Liu[1], and Hiroshi Murase[1]

***Abstract*— This paper presents a comparison of driving behavior modeling methods based on hidden Markov models (HMMs) with driver's eye-gaze measurement and ego-vehicle localization. Original HMMs are sometimes insufficient to model real-world scenarios. To overcome these limitations, extended HMMs have been proposed, e.g., autoregressive input–output HMMs (AIOHMMs). This paper first details AIOHMMs and presents ways to use them for driving behavior modeling. We compare the performance for behavior modeling and maneuver discrimination for six types of HMMs. The driving data for this work was gathered in our university campus with a car-like vehicle. Experimental results suggest that the hidden states can properly represent the average of the driving actions when the driving behaviors are accurately modeled by the HMMs. It is also suggested that surrounding and past information can be used to flexibly model the relationship between driving actions and related information.**

I. Introduction

Advanced driver assistance systems (ADASs) contribute to reduce traffic accidents. However, the alerts of the ADASs are sometimes harsh and unpleasant for the drivers because ADASs are unaware of the driver intentions and can generate undesirable alerts. The prediction of the driver's intention and the anticipation of driving maneuvers are necessary to make the alerts effective [1], [2], [3], [4], [5]. This paper focuses on driving behavior modeling for maneuver anticipation.

An example of a driving scene is shown in Fig. 1. The intention of a driver depends on the surroundings of an ego vehicle because drivers obtain much information through eyesight when they decide on driving actions. We focus on the movement in the eye-gaze direction and the control of the steering wheel and gas pedal as the driving action. We assume that the driver has several driving states and that driving actions are generated based on driving states. Additionally, we assume that driving maneuvers such as turning a corner are based on a set of surrounding information, driving states, and actions. In this work, we try to model these relationships and discriminate driving maneuvers using hidden Markov models (HMMs). Recently, deep learning-based modeling has become popular, e.g., [6], [7], but we focus on more explanatory modeling methods.

[1]Naoki Akai, Hailong Liu, and Hiroshi Murase are with the Graduate School of Informatics, Nagoya University, Nagoya 464-8603, Japan

[2]Takatsugu Hirayama, Luis Yoichi Morales, and Yasuhiro Akagi are with the Institute of Innovation for Future Society (MIRAI), Nagoya University, Nagoya 464-8601, Japan

E-mail of the corresponding author akai@nagoya-u.jp

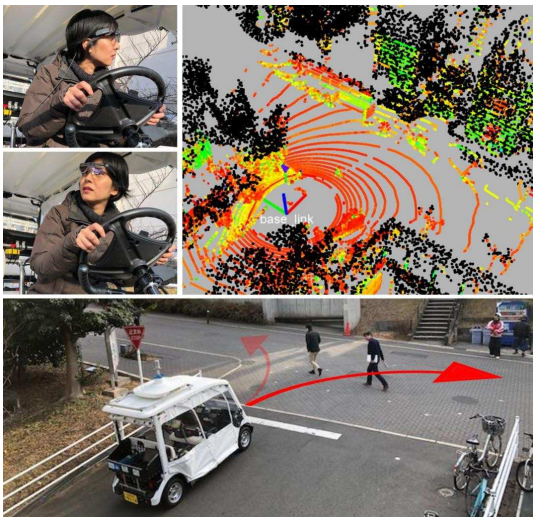

Fig. 1. Driving actions and an ego-vehicle localization result on a 3D point cloud map in a turning right scene. We model driving behaviors using various HMMs with the driver's eye-gaze measurement and the localization results. The driver wears an eye-gaze measurement device. Note that the ground points were removed from the 3D map to make it more readable.

Original HMMs have discrete hidden states and emission variables [8]. In [9], [10], [11], [12], HMMs are used to model driving behaviors and to estimate a risk level. Hidden states and emission variables can be viewed as corresponding to the driving states and actions. However, the original HMMs are sometimes insufficient to model real world scenarios because they cannot handle past and input information. Input information such as an external sensor and map data affect the hidden states and emission variables. To overcome these limitations, extended HMMs have been proposed, including the autoregressive HMM (AHMM) [13], [14], input-output HMM (IOHMM) [15], and autoregressive input-output HMM (AIOHMM) [16].

The first contribution of this work is to detail AIOHMMs and provide comparison results for the driving behavior modeling based on various types of HMMs. The temporal modeling using AIOHMMs is described in [16], [17], but our implementation is a bit different from the both of these two previous works. In our implementation, more hyperparameters are introduced, and this paper describes how these parameters are optimized. We evaluate the learning and discrimination performances of six types of HMMs with the driving dataset that was gathered at our university using

978-1-7281-0559-8/19/$31.00 ©2019 IEEE

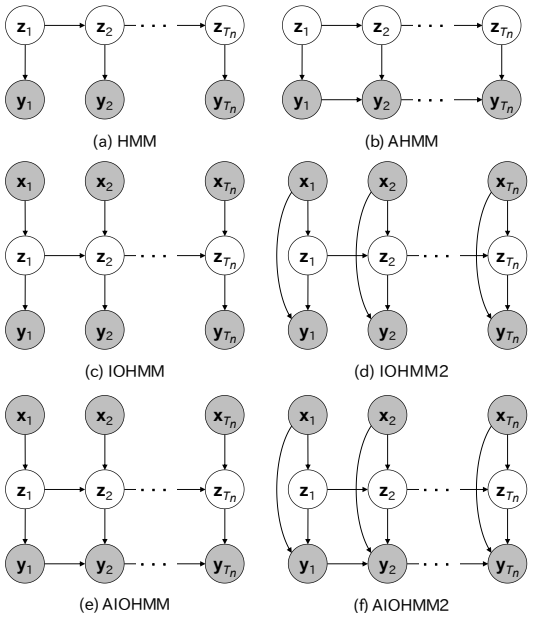

Fig. 2. The graphical models of HMMs used in this work. White and gray nodes illustrate latent and observable variables, respectively.

the experimental platform shown in Fig. 1. As a second contribution, we suggest that the hidden states can properly represent the average of the driving actions when the driving behaviors are accurately modeled by the HMMs. It is also suggested that the surroundings and past information can be used to flexibly model the relationship between driving actions and related information.

The rest of this paper is organized as follows. Section II details the modeling using AIOHMMs. Section III describes the driving behavior modeling in this work. Section IV describes the comparison results of the driving behavior modeling and maneuver discrimination using six types of HMMs. Section V concludes this work.

II. Modeling using AIOHMMs

Figure 2 illustrates the graphical models of HHMs used in this work; (a) HMM, (b) AHMM, (c) and (d) IOHMM, and (e) and (f) AIOHMM. $\mathbf{x}$ and $\mathbf{y}$ denote the input and output features and $\mathbf{z}$ denotes the discrete hidden states. In the IOHMM2 and AIOHMM2, the output features depend on the input features. This section details the AIOHMM2 and the optimization of its hyperparameters.

A. Learning

In the learning phase, we have N sequences of input and output features as a training dataset. One training dataset (one sequence) has a time sequence from 1 to T_n; these are denoted as $\mathbf{x}_{1:T_n}^{(n)}$ ($\mathbf{x}_t^{(n)} \in \mathcal{R}^M$) and $\mathbf{y}_{1:T_n}^{(n)}$ ($\mathbf{y}_t^{(n)} \in \mathcal{R}^L$), and the training dataset is denoted as $\mathcal{D} = \{(\mathbf{x}_{1:T_n}^{(n)}, \mathbf{y}_{1:T_n}^{(n)}) | n = 1, ..., N\}$. We also have discrete hidden states denoted as $\mathbf{z}_{1:T_n}^{(n)}$ ($\mathbf{z}_t^{(n)} \in \mathcal{R}^S$) with $z_{i,t}^{(n)} \in \{0, 1\}$ and $\sum_{i \in S} z_{i,t}^{(n)} = 1$. The objective of the learning phase is to find hyperparameters, $\mathbf{\Theta}$, that maximize the data log likelihood defined as:

$$l(\mathbf{\Theta}; \mathcal{D}) = \sum_{n=1}^{N} \sum_{t=1}^{T_n} \ln p(\mathbf{y}_t^{(n)} | \mathbf{x}_t^{(n)}; \mathbf{\Theta}). \tag{1}$$

It is, however, difficult to directly maximize equation (1) because the discrete states are latent. We can use the iterative expectation-maximization (EM) procedure, called the EM algorithm, to maximize equation (1). The EM algorithm includes two steps called the E-step and the M-step.

In the E-step, we calculate the expected value of the complete data log likelihood, $Q(\mathbf{\Theta}; \hat{\mathbf{\Theta}})$, defined as:

$$Q(\mathbf{\Theta}; \hat{\mathbf{\Theta}}) = E[l_c(\mathbf{\Theta}; \mathcal{D}_c) | \hat{\mathbf{\Theta}}, \mathcal{D}], \tag{2}$$

where $\hat{\mathbf{\Theta}}$ is the current hyperparameter, l_c is the complete data log likelihood, and $\mathcal{D}_c$ is the complete data, $\mathcal{D}_c = \{(\mathbf{x}_{1:T_n}^{(n)}, \mathbf{y}_{1:T_n}^{(n)}, \mathbf{z}_{1:T_n}^{(n)}) | n = 1, ..., N\}$. Because $\mathcal{D}_c$ includes the hidden states, it is called the "complete" data. The complete log likelihood is defined as:

$$l_c(\mathbf{\Theta}; \mathcal{D}_c) = \sum_{n=1}^{N} \sum_{t=1}^{T_n} \ln p(\mathbf{y}_t^{(n)}, \mathbf{z}_t^{(n)} | \mathbf{x}_t^{(n)}; \mathbf{\Theta}). \tag{3}$$

In the M-step, we maximize the expected value and update the hyperparameters as follows:

$$\mathbf{\Theta} = \arg\max_{\mathbf{\Theta}} Q(\mathbf{\Theta}; \hat{\mathbf{\Theta}}). \tag{4}$$

The hyperparameters can be optimized by deriving their closed form update expectations. Some parameters, however, have constraints that might not be satisfied by simply applying a closed form update. In these cases, the Lagrange multipliers can be used. It should be noted that the expected values are calculated using the current hyperparameters, $\hat{\mathbf{\Theta}}$, and these are regarded as constants in the M-step.

B. Probabilities

We model the AIOHMM2 with the following hyperparameters; $\mathbf{\Theta} = \{\boldsymbol{\mu}_i, \mathbf{\Sigma}_i, \mathbf{a}_{i,1}, \mathbf{a}_{i,}, \mathbf{b}_i, \boldsymbol{\pi}_l, \mathbf{W}_l\}$ $(i = 1, ..., S, l = 1, ..., L)$. This subsection details the hyperparameters and their optimization.

1) Emission probability: Let us assume that the output features of the hidden states are consistent with a Gaussian distribution. A simple Gaussian can be used in a case where an output feature depends on a current hidden state. In this work, a linear Gaussian model is used in cases where an output feature depends on variables except for a current hidden state. Means are linearly converted using coefficient vectors in linear Gaussian models.

In the AIOHMM2, a current output feature, $\mathbf{y}_t^{(n)}$, depends on current input and previous output features, $\mathbf{x}_t^{(n)}$ and $\mathbf{y}_{t-1}^{(n)}$. The emission probability is denoted as:

$$\begin{aligned} p(\mathbf{y}_t^{(n)} | \mathbf{x}_t^{(n)}, \mathbf{y}_{t-1}^{(n)}, \mathbf{z}_t^{(n)}) &= \prod_{i=1}^{S} \mathcal{N}(\mathbf{y}_t^{(n)} | \boldsymbol{\mu}_{i,t}^{(n)}, \mathbf{\Sigma}_i)^{z_{i,t}^{(n)}}, \\ \boldsymbol{\mu}_{i,t}^{(n)} &= \begin{cases} (1 + \mathbf{a}_{i,1}^T \mathbf{x}_t^{(n)}) \boldsymbol{\mu}_i & (t = 1) \\ (1 + \mathbf{a}_i^T \mathbf{x}_t^{(n)} + \mathbf{b}_i^T \mathbf{y}_{t-1}^{(n)}) \boldsymbol{\mu}_i & (t \geq 2), \end{cases} \end{aligned} \tag{5}$$

where $\mathbf{a}_{i,1}$, $\mathbf{a}_i$, and $\mathbf{b}_i$ are coefficient vectors related to the i-th hidden state. $\mathbf{a}_{i,1}$ is not used in [16], [17], but we introduce it as $t = 1$ because the dependency of the initial output feature is less than that of the other output features.

2) Initial probability: As the exception of the initial one, the hidden states of HMMs depend on a previous hidden state. This transition can be expressed using a transition matrix (this will be described next). In HMMs with no input features, the initial hidden state is completely independent, and its probability is defined as:

$$p(\mathbf{z}_1^{(n)}) = \prod_{i=1}^{S} \pi_i^{z_{i,1}^{(n)}}, \tag{6}$$

where $\boldsymbol{\pi} \in \mathcal{R}^S$ with $(\boldsymbol{\pi})_i = \pi_i \geq 0$ and $\sum_{i=1}^{S} \pi_i = 1$.

In HMMs with input features, the initial state depends on the initial input feature. Let us assume that the input features can be divided into L groups. In the implementation, we apply K-means clustering before the learning and obtain the means and covariances of the clusters. A variable $c_t^{(n)} \in \{1, ..., L\}$, that indicates the class number of the cluster to which an input feature, $\mathbf{x}_t^{(n)}$, is belonging, is introduced. For notational convenience, we write $c_t^{(n)}$ as c when it is used as an index and denote an initial probability as

$$p(\mathbf{z}_1^{(n)}|\mathbf{x}_1^{(n)}) = \prod_{i=1}^{S} \pi_{ic}^{z_{i,1}^{(n)}}, \tag{7}$$

where $(\boldsymbol{\pi}_c)_i = \pi_{ic}$. Equation (7) means that the initial probability is determined based on the class of an initial input feature. Note that the initial probability is not used in [16], [17].

3) Transition probability: In HMMs with no input features, the current hidden state only depends on the previous hidden state. Because there are S discrete states possible, the conditional probability is denoted using an $S \times S$ matrix as

$$p(\mathbf{z}_t^{(n)}|\mathbf{z}_{t-1}^{(n)}) = \prod_{i=1}^{S}\prod_{j=1}^{S} w_{ij}^{z_{i,t}^{(n)} z_{j,t-1}^{(n)}}, \tag{8}$$

where $p(z_t^{(n)} = j|z_{t-1}^{(n)} = i) = (\mathbf{W})_{ij} = w_{ij} \geq 0$. To preserve the consistency of the probabilistic distribution, the following condition must be satisfied:

$$\sum_{j=1}^{S} w_{ij} = 1. \tag{9}$$

Lagrange multipliers is used in the M-step to maintain this.

As we mentioned before, we assume that the input features are composed of L clusters. Thus, the transition probabilities in HMMs with input features are defined as:

$$p(\mathbf{z}_t^{(n)}|\mathbf{z}_{t-1}^{(n)}, \mathbf{x}_t^{(n)}) = \prod_{i=1}^{S}\prod_{j=1}^{S} w_{ijc}^{z_{i,t}^{(n)} z_{j,t-1}^{(n)}}, \tag{10}$$

where $p(z_t^{(n)} = j|z_{t-1}^{(n)} = i, \mathbf{x}_t^{(n)}) = (\mathbf{W}_c)_{ij} = w_{ijc}$.

In [16], [17], the transition probability is modeled as:

$$p(z_t^{(n)} = j|z_{t-1}^{(n)} = i, \mathbf{x}_t^{(n)}) = \frac{\exp\left(\mathbf{w}_{ij}^T \mathbf{x}_t^{(n)}\right)}{\sum_{l=1}^{S} \exp\left(\mathbf{w}_{il}^T \mathbf{x}_t^{(n)}\right)}, \tag{11}$$

where $\mathbf{w}_{ij} \in \mathcal{R}^M$ are the hyperparameters. The elements of the initial and transition probabilities easily converge to zero or one as these are optimized using the log likelihood. To flexibly model the transitions between hidden states, using several transition probabilities according to the current input feature could be an effective approach.

C. Optimization using the EM algorithm

1) E-step: We maximize the data log likelihood shown in equation (1) using the EM algorithm. We first calculate the expected value of the complete data log likelihood. By using equations (5), (7), and (10), we can re-write equation (3) as:

$$\begin{aligned} l_c(\boldsymbol{\Theta}; \mathcal{D}_c) = &\sum_{n=1}^{N}\sum_{t=1}^{T_n}\sum_{i=1}^{S} z_{i,t}^{(n)} \ln \pi_{ic} \\ &+ \sum_{n=1}^{N}\sum_{t=1}^{T_n}\sum_{i=1}^{S} z_{i,t}^{(n)} \ln \mathcal{N}(\mathbf{y}_t^{(n)}|\boldsymbol{\mu}_{i,t}^{(n)}, \boldsymbol{\Sigma}_i) \\ &+ \sum_{n=1}^{N}\sum_{t=2}^{T_n}\sum_{i=1}^{S}\sum_{j=1}^{S} z_{i,t}^{(n)} z_{j,t-1}^{(n)} \ln w_{ijc}. \end{aligned} \tag{12}$$

Here, we define the expected values of the probabilistic distribution of $\mathbf{z}_t^{(n)}$ and the joint probabilistic distribution of $\mathbf{z}_t^{(n)}$ and $\mathbf{z}_{t-1}^{(n)}$ with all of the observed variables in one sequence and hyperparameters as:

$$\gamma_{i,t}^{(n)} \stackrel{\text{def}}{=} p(z_{i,t}^{(n)} = 1|\mathbf{x}_{1:T_n}^{(n)}, \mathbf{y}_{1:T_n}^{(n)}, \boldsymbol{\Theta}), \tag{13}$$

$$\xi_{ij,t}^{(n)} \stackrel{\text{def}}{=} p(z_{i,t}^{(n)} = 1, z_{j,t-1}^{(n)} = 1|\mathbf{x}_{1:T_n}^{(n)}, \mathbf{y}_{1:T_n}^{(n)}.\boldsymbol{\Theta}), \tag{14}$$

With equations (13) and (14), the expected value can be re-written as:

$$\begin{aligned} Q(\boldsymbol{\Theta}; \hat{\boldsymbol{\Theta}}) = &\sum_{n=1}^{N}\sum_{t=1}^{T_n}\sum_{i=1}^{S} \gamma_{i,t}^{(n)} \ln \pi_{ic} \\ &+ \sum_{n=1}^{N}\sum_{t=1}^{T_n}\sum_{i=1}^{S} \gamma_{i,t}^{(n)} \ln \mathcal{N}(\mathbf{y}_t^{(n)}|\boldsymbol{\mu}_{i,t}^{(n)}, \boldsymbol{\Sigma}_i) \\ &+ \sum_{n=1}^{N}\sum_{t=2}^{T_n}\sum_{i=1}^{S}\sum_{j=1}^{S} \xi_{ij,t}^{(n)} \ln w_{ijc}. \end{aligned} \tag{15}$$

To efficiently calculate $\gamma_{i,t}^{(n)}$ and $\xi_{ij,t}^{(n)}$, we use the forward-backward algorithm. Additionally, we use the scaling algorithm to avoid an unstable computation, i.e., underflow. In this paper, we describe the minimal essential equations to calculate $\gamma_{i,t}^{(n)}$ and $\xi_{ij,t}^{(n)}$. For readers who want to know more details, please refer to [18].

$$\gamma_{i,t}^{(n)} = \hat{\alpha}_{i,t}^{(n)} \hat{\beta}_{i,t}^{(n)}, \tag{16}$$

$$\xi_{ij,t}^{(n)} = \frac{\hat{\alpha}_{i,t}^{(n)} w_{ijc} \mathcal{N}(\mathbf{y}_t^{(n)}|\boldsymbol{\mu}_{i,t}^{(n)}, \boldsymbol{\Sigma}_i) \hat{\beta}_{j,t+1}^{(n)}}{s_{t+1}^{(n)}}, \tag{17}$$

$$s_t^{(n)}\hat{\alpha}_{i,t}^{(n)} = \left(\sum_{j=1}^{S}\hat{\alpha}_{j,t-1}^{(n)}w_{ijc}\right)\mathcal{N}(\mathbf{y}_t^{(n)}|\boldsymbol{\mu}_{i,t}^{(n)},\boldsymbol{\Sigma}_i), \quad (18)$$

$$s_t^{(n)} = \sum_{i=1}^{S}\left(\sum_{j=1}^{S}\hat{\alpha}_{j,t-1}^{(n)}w_{ijc}\right)\mathcal{N}(\mathbf{y}_t^{(n)}|\boldsymbol{\mu}_{i,t}^{(n)},\boldsymbol{\Sigma}_i), \quad (19)$$

$$s_{t+1}^{(n)}\hat{\beta}_{i,t}^{(n)} = \sum_{j=1}^{S} w_{ijc}\mathcal{N}(\mathbf{y}_t^{(n)}|\boldsymbol{\mu}_{i,t}^{(n)},\boldsymbol{\Sigma}_i)\beta_{j,t+1}^{(n)}. \quad (20)$$

$\hat{\alpha}_{i,t}^{(n)}$ and $\hat{\beta}_{i,t}^{(n)}$ can be recursively calculated and their initial values are set to the following: $\hat{\alpha}_{i,1}^{(n)} = \pi_{ic}\mathcal{N}(\mathbf{y}_1^{(n)}|\boldsymbol{\mu}_{i,1}^{(n)},\boldsymbol{\Sigma}_i)$ and $\hat{\beta}_{i,T_n}^{(n)} = 1$.

2) M-step: In the M-step, $\gamma_{i,t}^{(n)}$ and $\xi_{i,t}^{(n)}$ are treated as constant. We derive the partial derivatives of $Q(\boldsymbol{\Theta};\hat{\boldsymbol{\Theta}})$ with respect to each hyperparameter regarding the emission probabilities and update them as:

$$\mathbf{a}_{i,1} = \left(\sum_{n=1}^{N}\gamma_{i,1}^{(n)}\mathbf{x}_1^{(n)}\mathbf{x}_1^{(n)T}\right)^{-1} \sum_{n=1}^{N}\gamma_{i,1}^{(n)}\left(\frac{\mathbf{x}_1^{(n)}\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}^{-1}\mathbf{y}_1^{(n)}}{\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i} - \mathbf{x}_1^{(n)}\right), \quad (21)$$

$$\mathbf{a}_i = \left(\sum_{n=1}^{N}\sum_{t=2}^{T_n}\gamma_{i,t}^{(n)}\mathbf{x}_t^{(n)}\mathbf{x}_t^{(n)T}\right)^{-1} \sum_{n=1}^{N}\sum_{t=2}^{T_n}\gamma_{i,t}^{(n)}\left(\frac{\mathbf{x}_t^{(n)}\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}^{-1}\mathbf{y}_t^{(n)}}{\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i} - \mathbf{x}_t^{(n)} - \mathbf{b}_i^T\mathbf{y}_{t-1}^{(n)}\mathbf{x}_t^{(n)}\right), \quad (22)$$

$$\mathbf{b}_i = \left(\sum_{n=1}^{N}\sum_{t=2}^{T_n}\gamma_{i,t}^{(n)}\mathbf{y}_{t-1}^{(n)}\mathbf{y}_{t-1}^{(n)T}\right)^{-1} \sum_{n=1}^{N}\sum_{t=2}^{T_n}\gamma_{i,t}^{(n)}\left(\frac{\mathbf{y}_{t-1}^{(n)}\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}^{-1}\mathbf{y}_t^{(n)}}{\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i} - \mathbf{y}_{t-1}^{(n)} - \mathbf{a}_i^T\mathbf{x}_t^{(n)}\mathbf{y}_{t-1}^{(n)}\right), \quad (23)$$

$$\boldsymbol{\mu}_i = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T_n}c_{i,t}^{(n)}\gamma_{i,t}^{(n)}\mathbf{y}_t^{(n)}}{\sum_{n=1}^{N}\sum_{t=1}^{T_n}c_{i,t}^{(n)2}\gamma_{i,t}^{(n)}}, \quad (24)$$

$$\boldsymbol{\Sigma}_i = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T_n}\gamma_{i,t}^{(n)}\mathbf{V}_{i,t}^{(n)}}{\sum_{n=1}^{N}\sum_{t=1}^{T_n}\gamma_{i,t}^{(n)}}, \quad (25)$$

where $c_{i,1}^{(n)} = 1 + \mathbf{a}_{i,1}^T\mathbf{x}_1^{(n)}$, $c_{i,t}^{(n)} = 1 + \mathbf{a}_i^T\mathbf{x}_t^{(n)} + \mathbf{b}_i^T\mathbf{y}_{t-1}^{(n)}$ $(t \geq 2)$, and $\mathbf{V}_{i,t}^{(n)} = (\mathbf{y}_t^{(n)} - c_{i,t}^{(n)}\boldsymbol{\mu}_i)(\mathbf{y}_t^{(n)} - c_{i,t}^{(n)}\boldsymbol{\mu}_i)^T$. It should be noted that $\mathbf{a}_{i,1}$, $\mathbf{a}_i$, and $\mathbf{b}_i$ cannot be updated when $N = 1$ because the inverse matrices cannot be defined.

The initial probabilities are updated as:

$$\pi_{il} = \frac{\sum_{n=1}^{N}\mathbb{1}(c_t^{(n)} = l)\gamma_{i,1}^{(n)}}{\sum_{n=1}^{N}\sum_{j=1}^{S}\mathbb{1}(c_t^{(n)} = l)\gamma_{j,1}^{(n)}}, \quad (26)$$

where $\mathbb{1}(\cdot)$ is an indicator function which is equal to 1 when the condition within the bracket is true, and 0 otherwise.

As shown in equation (9), the transition matrices reflect the constraints. To satisfy the maximization constraints, we introduce a new function defined as:

$$J(\boldsymbol{\Theta};\hat{\boldsymbol{\Theta}}) \stackrel{\text{def}}{=} Q(\boldsymbol{\Theta};\hat{\boldsymbol{\Theta}}) + \sum_{j=1}^{S}\sum_{l=1}^{K}\lambda_{jl}\left(1 - \sum_{i=1}^{S}w_{ijl}\right), \quad (27)$$

where λ_{jl} are the Lagrange multipliers. Taking the derivatives of this function with respect to w_{ijl}, we obtain:

$$\frac{\partial J(\boldsymbol{\Theta};\hat{\boldsymbol{\Theta}})}{\partial w_{ijl}} = \sum_{n=1}^{N}\sum_{t=2}^{T_n}\mathbb{1}(c_t^{(n)} = l)\xi_{ij,t}^{(n)}\frac{1}{w_{ijl}} - \lambda_{jl}, \quad (28)$$

Consequently, w_{ijl} is updated as:

$$w_{ijl} = \frac{\sum_{n=1}^{N}\sum_{t=2}^{T_n}\mathbb{1}(c_t^{(n)} = l)\xi_{ij,t}^{(n)}}{\sum_{i=1}^{S}\sum_{n=1}^{N}\sum_{t=2}^{T_n}\mathbb{1}(c_t^{(n)} = l)\xi_{ij,t}^{(n)}}. \quad (29)$$

$\lambda_{jl} = \sum_{i=1}^{S}\sum_{n=1}^{N}\sum_{t=2}^{T_n}\mathbb{1}(c_t^{(n)} = l)\xi_{ij,t}^{(n)}$ can be obtained by imposing the constraint $\sum_{i=1}^{S}w_{ijl} = 1$.

In [16], [17], they optimized the coefficients of the transition matrix, $\mathbf{w}_{ij}$ shown in equation (11), using gradient descent because the closed form-based update and Lagrange multipliers cannot be used. However, they did not provide the definition of the gradient.

III. Driving behavior modeling

A. Experimental platform

We used the car like platform shown at the bottom of Fig. 1 to gather driving data for this work. The platform was equipped with a 3D LiDAR (HDL-32e) which allowed the vehicle to precisely recognize its own position [19], [20]. Measurements of the 3D LiDAR and an environment-metric map were used to obtain input features. The inertial movement can be measured via CAN and IMU. Additionally, we used the eye-gaze measurement device, Tobii Pro Glasses 2. The inertial movement and eye-gaze information are used to create the output features.

B. Input features

We first create a virtual 2D scan from the raw 3D scan points. The minimum and maximum angles of the 2D scan are set to -135 and 135 degrees. First, the points included in a region ranging from a -1.5 m to 0.2 m height from the LiDAR are clipped. Second, a horizontal plane, which is parallel to the surface of the ground, is radially divided into 25 regions. The minimum distances of each region are used as elements of an input feature. The experimental vehicle recognizes its own position while driving and has an occupancy grid map. The same 25 regions are set according to the estimated pose and the minimum distance from the vehicle to obstacles on the map are used as elements of an input feature as well. Consequently, one input feature is composed of 50 elements.

C. Output features

The eye-gaze measurement device and the experimental vehicle provide a 3D eye-gaze vector and linear and angular velocities. We calculate the average and variance values for the raw, absolute, difference, and absolute difference values for each measurement for 0.5 s and these are used as elements of an output feature. Here, the difference is computed using the current and previous measurements. Consequently, one output feature is composed of 40 elements.

D. Dimension reduction

In the M-step, we need to compute inverse matrices. Its computation is heavy and unstable if the size of the matrix is large. Additionally, input and output features might include useless information. Thus, we apply a principal component analysis (PCA) to reduce dimensionality of the input and output features. The data provided is normalized with respect to the maximum norm of the data before applying the PCA. We choose the reduction size consistent with the cumulative contribution ratio exceeding 90 % in both input and output features. Consequently, the input and output features are compressed to 30 and 3 dimensions, respectively.

E. Initial values of hyperparameters

The initial values of the emission probability, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$, are given through K-means clustering results. The numbers of K-means clusters and hidden states of the HMMs are equal. The coefficient vectors computing the linear Gaussian model shown in (5), $\mathbf{a}_{i,1}$, $\mathbf{a}_i$, and $\mathbf{b}_i$, are set to $\mathbf{0}$. The initial and transition probabilities, $\boldsymbol{\pi}_l$ and $\mathbf{W}_l$, are set as uniform.

IV. Comparison Experiments

A. Driving data

We gathered the driving data for six individuals in our university. Lane change maneuvers could not be observed as there are no multi-lane roads in our university. Several interaction scenes with cars and pedestrians occurred and the drivers slowed down in these scenes. We saw that the drivers followed the traffic participants in such interactive scenes. Thus, we divided the driving maneuvers into four maneuvers; go straight, turn left, turn right, and follow participants.

We gathered approximately 4 hours of driving data that included 394 total maneuvers, 192 "go straight", 72 "turn left", 87 "turn right", and 43 "follow participants" maneuvers. Approximately half the data was used for learning and the remainder for verification.

B. Learning results

We first compared the learning results with various numbers of hidden states; 3, 5, and 10. Figure 3 shows the K-means clustering results of the output features of the "go straight" dataset. The emission probabilities were initialized using the means and covariances of the clustering results. Additionally, Fig. 4 shows the clustering results ($K = 10$) of the input features of the "go straight" dataset (top) and the corresponding output features (bottom). It is difficult to find the direct relationship between input and output features.

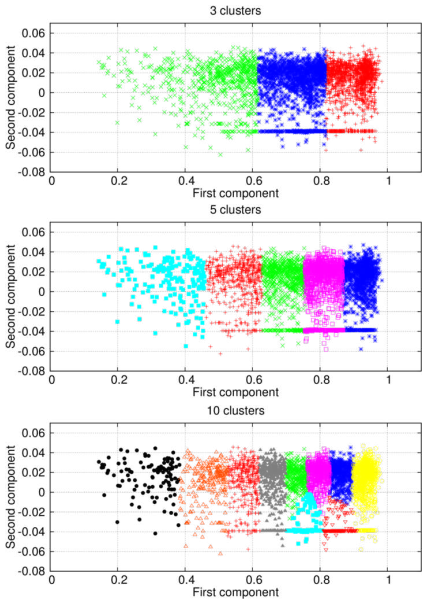

Fig. 3. The K-means clustering results of the output features of the "go straight" dataset. The cluster numbers are 3, 5, and 10 from the top.

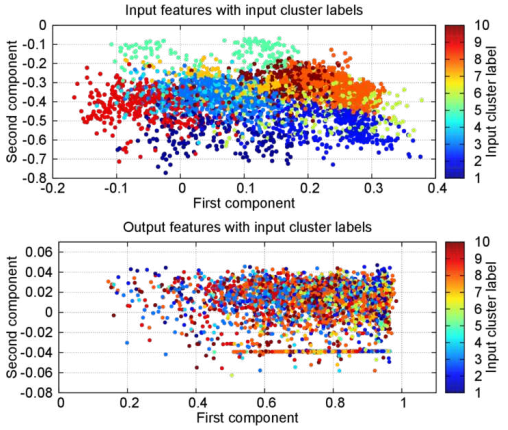

Fig. 4. The K-means clustering result of the input features of the "go straight" dataset (top) and corresponding output features (bottom).

Note that these input and output features were visualized using the first and second components of the PCA results.

Figure 5 shows the transitions of the complete data log likelihood during the learning phase. A result with high log likelihood means that the output features fitted the emission probabilities well. We could see a trend where the HMMs for which there was a dependency of the current output features on the input and past output features fit these output features well. In fact, the log likelihoods of the original HMMs, that do not have input and past dependencies against the output features, were almost smaller than that of the other HMMs.

Figure 6 shows the fitting results of the output features of the "go straight" dataset for the HMMs. The blue points show the means of the linear Gaussian model (we refer to these as the "converted means"). The converted means enabled the HMMs to flexibly model the output features and this yielded

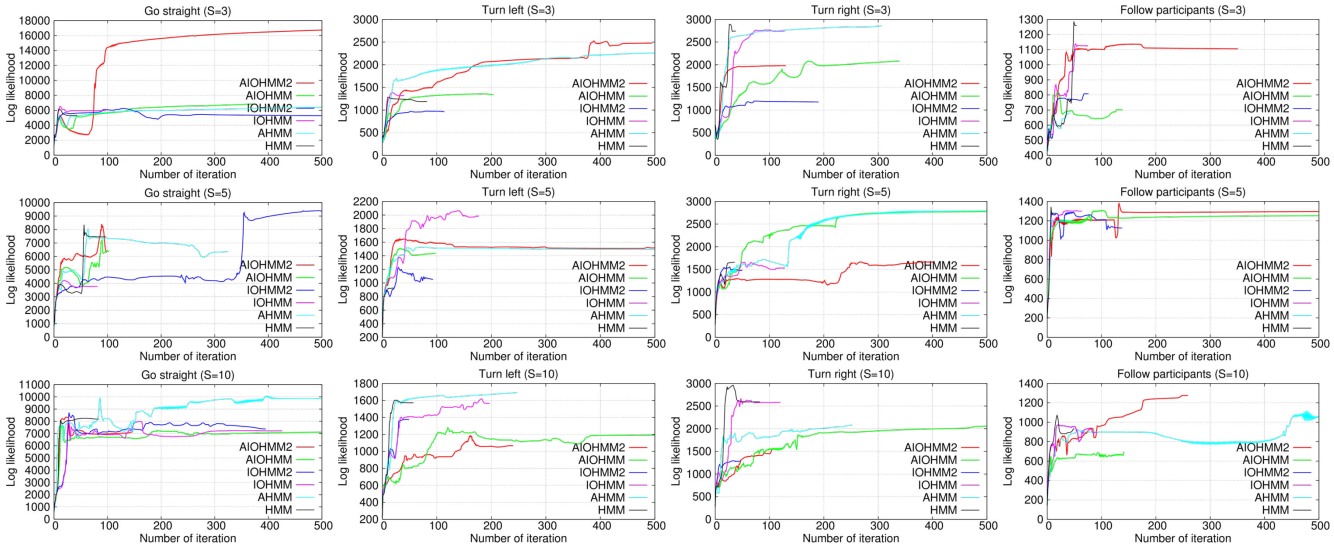

Fig. 5. Transitions of the complete data log likelihood shown in equation (3) for each HMM with various numbers of hidden states, S. The top, middle, and bottom figures show cases where the number of hidden states is 3, 5, and 10, respectively.

a trend where the HMMs for which there was a dependency of the output features on the input and the past resulted in high log likelihoods.

As shown in Fig. 4, the direct relationship between the input and output features is not visible, but the input features could be used to increase the log likelihoods. This suggested that using the hidden states is effective in modeling relations between the driving actions and the surrounding information.

C. Discrimination results

We can determine the probability that an output feature sequence, $\mathbf{y}_{1:T_n}^{(n)}$, is obtained with the learnt HMMs using the forward algorithm [18] as:

$$p(\mathbf{y}_{1:T_n}^{(n)}) = \sum_{i=1}^{S} \alpha_{i,T_n}^{(n)}, \tag{30}$$

where $\alpha_{i,T_n}^{(n)}$ is defined by omitting $s_t^{(n)}$ from equation (18). We evaluated the discrimination performances of the driving maneuvers for the HMMs using the test dataset.

Figure 7 shows the confusion matrices for all of the HMMs. The diagonal of the matrices shows the precision. A trend emerged, as the HMMs with the dependencies on the output features resulted in a better performance.

The AIOHMM2 with five hidden states and the AHMM with ten hidden states had the best and worst precision, respectively. As can be seen in the middle row of Fig. 5, the AIOHMM2 did not always result in the highest log likelihood. By contrast, as can be seen in the bottom row, the AHMM usually resulted in a higher log likelihood. These results show that results with high log likelihood means that the output features are well fitted to the emission probabilities, however this sometimes leads to overfitting where the output features are not accurately modeled.

D. Discussion

Figure 8 shows the fitting results of the output features of the "turn left", "turn right", and "follow participants" datasets for the AIOHMM2 with five hidden states and the AHMM with ten hidden states. As can be seen in Figs. 6 and 8, almost all the means are seen inside the output features and the converted means could capture the discriminative patterns of the output features. By contrast, several means are seen outside of the output features in the AHMM fitting results. Additionally, the shift in converted means is larger here than it is in the AIOHMM2 fitting results. The hidden states can be seen as adequately representing average driving actions and the coefficients for computing linear Gaussian models flexibly represent the relationship between driving actions and related information when the driving behaviors are accurately modeled. In contrast, the AHMM fitting results could be interpreted as over-fitting because the output features were forcibly fitted using the converted means and this does not suitably represent the relationship between driving actions and related information.

In [21], driving features were extracted using the deep autoencoder with driving information such as CAN. The benefit of deep learning-based methods is to extract complex nonlinear relationships and these approaches successfully managed to classify and visualize driving behaviors based on extracted 3D features. It is, however, still difficult to describe driving behaviors with extracted features because understanding relations between the features extracted, and the driving actions is challenging. By contrast, HMMs-based methods could tell us the relationships between the hidden states and driving actions as we can access the emission probability of each hidden state. However, the descriptive and extraction abilities of the HMM-based methods for complex nonlinear relations are not higher than that of the deep learning-based methods. In fact, we can only consider linear

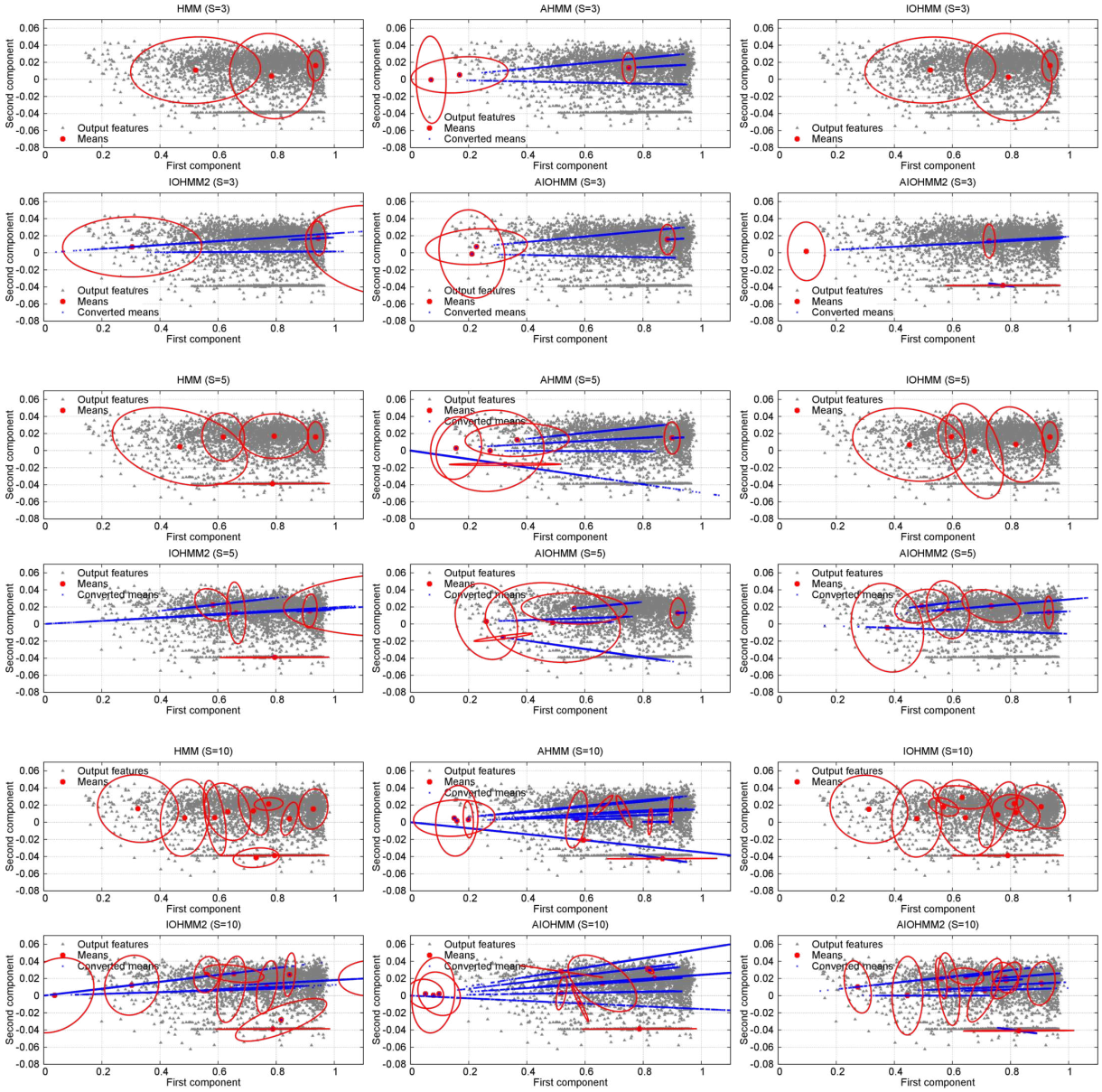

Fig. 6. Fitting results of the output features of the "go straight" dataset for the six HMMs with various numbers of hidden states.

relations between driving actions and input and past information in AIOHMMs. Combining both benefits is necessary for further analyzing the relations between driving actions and related information.

V. Conclusion

This paper has presented six driving behavior models based on HMMs with measurement of the driver's eye-gaze and ego-vehicle localization. We first detailed AIOHMMs and their implementation and then compared the modeling and driving maneuver discrimination performance of the HMMs. The comparison results suggested that the hidden states can properly represent the average of the driving actions when the driving behaviors are accurately modeled by the HMMs. It is also suggested that surrounding and past information can be used to flexibly model relations between the driving actions and related information.

Acknowledgment

This work was supported by the Center of Innovation Program (Nagoya-COI) funded by the JST Agency, JST-Mirai Program (Grant Number JPMJMI17C6), and Technova Inc. We would also like to thank Profs. Takayuki Morikawa and Kzuya Takeda who helped with the experiments.

References

[1] L. Fletcher et al. Correlating driver gaze with the road scene for driver assistance systems. *Rob. and Auton. Systems*, 52:71–84, 2005.

[2] A. Tawari et al. Looking-in and looking-out vision for urban intelligent assistance: Estimation of driver attentive state and dynamic surround for safe merging and braking. In *Proc. of the IEEE Intel. Veh. Symp.*, pages 115–120, 2014.

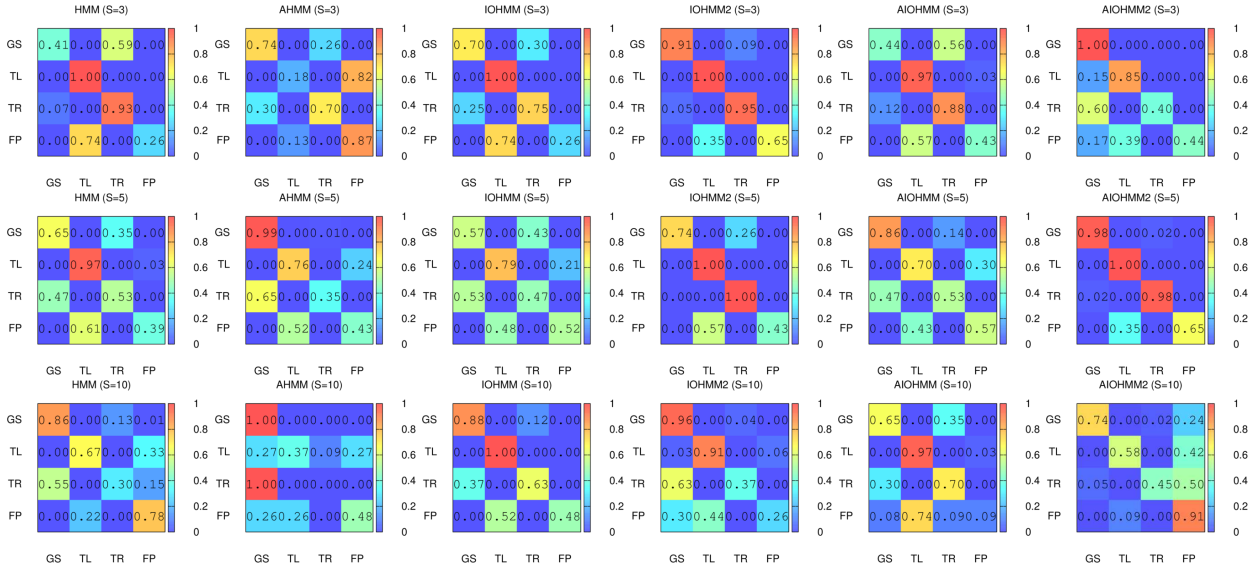

Fig. 7. The confusion matrices for HMMs. GS, TL, TR, and FP denote "go straight", "turn left", "turn right", and "follow participants", respectively.

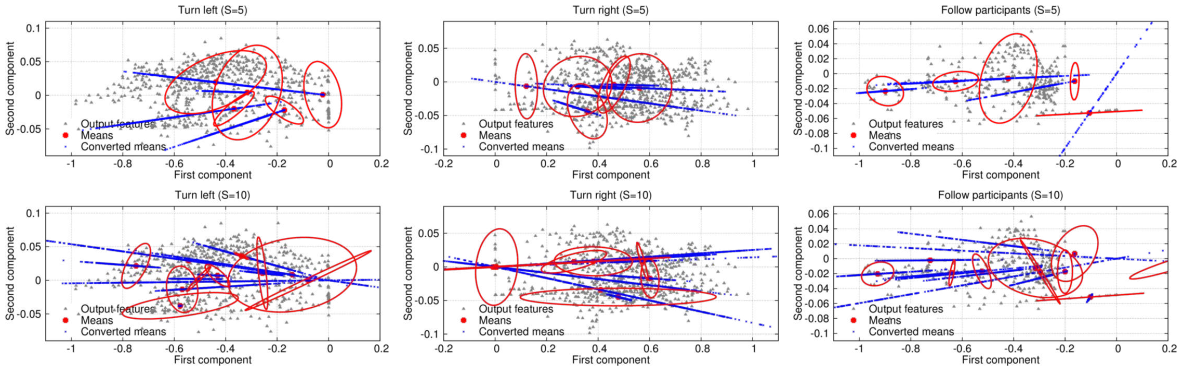

Fig. 8. The fitting results of the output features of the "turn left", "turn right", and "follow participants" for the AIOHMM2 with five hidden states (top) and the AHMM with ten hidden states (bottom). The AIOHMM2 and the AHMM had the best and worst precision, respectively.

[3] M. Rezaei et al. Look at the driver, look at the road: No distraction! no accident! In *Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2014.

[4] F. Vicente et al. Driver gaze tracking and eyes off the road detection system. *IEEE Trans. on Intel. Transp. Systems*, 16(4):2014–2027, 2015.

[5] N. Rhinehart et al. R2P2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *The European Conference on Computer Vision*, 2018.

[6] A. Jain et al. Brain4Cars: Car that knows before you do via sensory-fusion deep learning architecture. *CoRR*, abs/1601.00740, 2016.

[7] F. Codevilla et al. End-to-end driving via conditional imitation learning. In *Proc. of the IEEE Int. Conf. on Rob. and Autom.*, pages 4693–4700, 2018.

[8] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proc. of the IEEE*, volume 77, pages 257–286, 1989.

[9] N. Oliver et al. Graphical models for driver behavior recognition in a SmartCar. In *Proc. of the IEEE Intel. Veh. Symp.*, pages 7–12, 2000.

[10] H. Berndt et al. Continuous driver intention recognition with hidden Markov models. In *Proc. of the IEEE Int. Conf. on Intel. Transp. Systems*, pages 1189–1194, Oct 2008.

[11] H. Hou et al. Driver intention recognition method using continuous hidden Markov model. *Int. J. of Computational Intel. Systems*, 4(3):386–393, 2011.

[12] M. Mori et al. Integrated modeling of driver gaze and vehicle operation behavior to estimate risk level during lane changes. In *Proc. of the IEEE Int. Conf. on Intel. Transp. Systems*, 2013.

[13] B.-H. Juang et al. Mixture autoregressive hidden Markov models for speech signals. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 33:1404–1413, 1985.

[14] Y. Kishimoto et al. A modeling method for predicting driving behavior concerning with driver's past movements. In *Proc. of the IEEE Int. Conf. on Veh. Elec. and Safety*, pages 132–136, 2008.

[15] Y. Bengio et al. An input output HMM architecture. In *Proc. Int. Conf. on Neural Information Processing Systems*, pages 427–434, 1994.

[16] A. Jain et al. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. *CoRR*, abs/1504.02789, 2015.

[17] E. L. Zec et al. Statistical sensor modelling for autonomous driving using autoregressive input-output HMMs. In *Proc. of the IEEE Int. Conf. on Intel. Transp. Systems*, pages 1331–1336, 2018.

[18] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg, 2006.

[19] N. Akai et al. Robust localization using 3D NDT scan matching with experimentally determined uncertainty and road marker matching. In *Proc. of the IEEE Intel. Veh. Symp*, pages 1357–1364, 2017.

[20] N. Akai et al. Autonomous driving based on accurate localization using multilayer lidar and dead reckoning. In *Proc. of the IEEE Int. Conf. on Intel. Transp. Systems*, pages 1147–1152, 2017.

[21] K. Sama et al. Driving feature extraction and behavior classification using an autoencoder to reproduce the velocity styles of experts. In *Proc. of the IEEE Int. Conf. on Intel. Transp. Systems*, pages 1337–1343, 2018.