2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Transfer Learning for Driver Model Adaptation via Modified Local Procrustes Analysis

Chao Lu, Fengqing Hu, Wenshuo Wang, *Member, IEEE*, Jianwei Gong*, *Member, IEEE*, Zeliang Ding

***Abstract*— A new driver model adaptation (DMA) method is proposed in this paper to help the model adaptation between different individual drivers. This method is based on transfer learning which can improve the DMA process at data level. The Gaussian mixture model (GMM)-based method is used to model the steering behaviour of drivers during the overtaking manoeuvre. Based on the GMM model, an alignment-based transfer learning technique named local Procrustes analysis (LPA) is modified to formulate the transfer learning problem for driver steering behaviour. A series of experiments based on the data collected from a driving simulator are carried out to evaluate the proposed modified LPA (MLPA). The experimental results verify the ability of MLPA for knowledge transfer. Compared with the GMM-only method and LPA, MLPA shows better performance on the prediction accuracy with much lower predicting errors in most cases.**

I. Introduction

Exact prediction of driver behaviours plays an essential role for developing an effective advanced driver assistance system (ADAS). Traditional driver behaviour modelling methods based on the hybrid dynamical systems [1-3], hidden Markov models (HMM) [4, 5] and artificial neural networks [6-8] have shown their effectiveness on predicting driver behaviours in various scenarios. However, when the personalised driver model is required for each individual driver, such kind of methods needs a large amount of data collected from each driver involved [9]. Collecting sufficient driving data from each individual driver is a time-consuming work and requires considerable financial support. This issue will get even worse when the number of drivers considered is increasing. In this case, driver model adaptation (DMA) method is required to adapt the models at hand to fit the characteristics of the newly-involved drivers without collecting a large amount of data to train the new models.

*This work was supported by the National Natural Science Foundation of China under Grants 61703041 and 91420203, Beijing Institute of Technology Research Fund Program for Young Scholars and Key Laboratory of Biomimetic Robots and Systems, Beijing Institute of Technology, Ministry of Education (Project Grant No. 2017CX02005).

Chao Lu is with Beijing Institute of Technology, Beijing 100081, China (e-mail: chaolu@bit.edu.cn).

Fengqing Hu is with Beijing Institute of Technology, Beijing 100081, China (e-mail: 2120170338@bit.edu.cn).

Wenshou Wang is with Beijing Institute of Technology, Beijing 100081, China and also with University of Michigan, Ann Arbor, MI 48109, USA (e-mail: wwsbit@gmail.com).

*Jianwei Gong is with Beijing Institute of Technology, Beijing 100081, China (corresponding author, e-mail: gongjianwei@bit.edu.cn).

Zeliang Ding is with Beijing Institute of Technology, Beijing 100081, China (e-mail: 1242703731@qq.com).

Although some methods, such as the adaptive predictive control (APC) [10] and Gaussian mixture model (GMM) [9, 11] have been proposed to deal with the problem of DMA, a pre-trained driver model is usually required by these methods to start the adaptation. The DMA using these methods is conducted by training a model first based on the historical data and then adapting the parameters of this model to fit the newly-observed data. In this paper, a new DMA method based on transfer learning is proposed to realise the adaptation at data level without pre-training a driver model. In this way, the data collected from one driver can be directly transferred to the model training process of another driver, which offers a more flexible mechanism for DMA and improves the whole adaptation procedure.

The basic objective of transfer learning is to transfer the knowledge obtained from a familiar domain to another relevant new domain and help to solve the problem faced in this new domain. Based on the same principle, the terminology of transfer learning in the machine learning community is used to refer to the technique that can transfer knowledge between different data domains and improve the model training process in the newly-observed domain [12]. Because of this promising feature, transfer learning has been successfully applied to solve the problems of image recognition [13], language processing [14], robot model learning [15-17] and even the driver behaviour transfer [18]. As an early attempt, a transfer learning approach is proposed in [18] to transfer the driver behaviour between two different computer racing games. The objective of this work is to transfer the driver behaviour model embedded in a racing game to another one with different simulation environment, and thus improve the development process of the new game. Modelling and transferring the realistic driving behaviour between different drivers are not the focus of the work presented in [18]. This paper, instead, uses transfer learning to improve the model adaptation between different realistic drivers. One of the alignment-based transfer learning techniques named local Procrustes analysis (LPA) is modified in this study to model and transfer the steering behaviour of drivers. The overtaking scenario is selected to formulate the whole transfer learning problem, and the driving data are collected from realistic drivers using the driving simulator. When the modified LPA (MLPA) is used for DMA, a pre-trained model is not required, and the data collected from the source driver can be transferred to the target driver by an alignment function, which improves the model training of the target driver.

The remainder of this paper is organised as follows. Section II formulates the whole transfer learning problem and gives the details of the model and relevant algorithm. Section III designs a group of experiments to evaluate the proposed

MLPA based on the data collected from driving simulator. Conclusions and future work are presented in Section IV.

II. Problem Formulation

The overtaking scenario considered in this paper is shown in Figure 1.

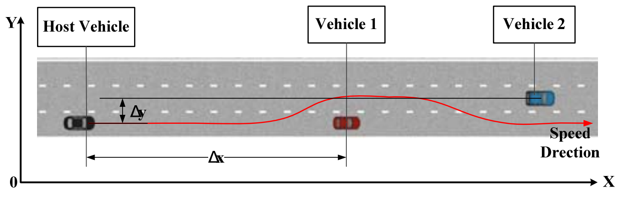

Figure 1. The overtaking scenario

On a typical 3-lane urban road, the host vehicle is aiming to overtake Vehicle 1 and get back to its original lane without collision as soon as possible. Except for the host vehicle, the other two vehicles are driven in the same constant speed and kept in their own lanes during the whole test.

A. Modelling the steering behaviour

The careful control of steering wheel plays an essential role for guaranteeing a successful overtaking manoeuvre and thus is the focus of this paper. The steering behaviour of drivers is expressed by a model with the relative state of vehicles as the model input and the steering wheel angle as the model output. For the overtaking scenario considered in Figure 1, the relative state of vehicles can be modelled by a state vector with 6 elements as shown below.

$$s_i = [\Delta x_{1,i} \quad \Delta y_{1,i} \quad \Delta v_{1,i} \quad \Delta x_{2,i} \quad \Delta y_{2,i} \quad \Delta v_{2,i}]^T \tag{1}$$

where i is the time index, Δx_1 denotes the relative distance on direction X between the host vehicle and Vehicle 1, Δy_1 denotes the relative distance on direction Y between the host vehicle and Vehicle 1, Δv_1 denotes the relative velocity between host vehicle and Vehicle 1, while Δx_2, Δy_2 and Δv_2 convey the same meaning respectively between the host vehicle and Vehicle 2.

The model output at time i is given by:

$$a_i = \delta_i \tag{2}$$

where δ_i is the steering wheel angle in degrees at time i.

By combining the model input and output we can get the extended feature vector z_i for the model.

$$z_i = [s_i^T \quad a_i]^T \tag{3}$$

Similar to the work presented in [19, 20], the widely-used GMM is selected here to construct the relationship between the model input and output. Under the framework of GMM, the extended feature vector satisfies the following conditional probability distribution.

$$p(z \mid \Pi) = \sum_{k=1}^{K} \pi_k N(z, \mu_k^z, \Sigma_k^{zz}) \tag{4}$$

where $\Pi = \{\pi_k, \mu_k^z, \Sigma_k^{zz}\}$ is the GMM parameters, K denotes the number of mixture components, π_k denotes the priori probability of each component.

$$\mu_k^z = \begin{bmatrix} \mu_k^s \\ \mu_k^a \end{bmatrix} \quad \text{and} \quad \Sigma_k^{zz} = \begin{bmatrix} \Sigma_k^{ss} & \Sigma_k^{sa} \\ \Sigma_k^{as} & \Sigma_k^{aa} \end{bmatrix} \tag{5}$$

where Σ_k^{aa} and Σ_k^{s} are auto-covariance matrices for the k th mixture component related to s_i and a_i, Σ_k^{sa} and Σ_k^{as} are cross-covariance matrices, μ_k^s and μ_k^a are mean vectors for s_i and a_i.

Given μ_k^z and Σ_k^{zz}, the steering wheel angle a_{i+1} at time step i+1 can be calculated by the following equations according to [9]:

$$\hat{\mu}_k^a(s_i) = \mu_k^a + \Sigma_k^{as}(\Sigma_k^{ss})^{-1}(s_i - \mu_k^s) \tag{6}$$

$$h_k(s_i) = \frac{\pi_k N(s_i \mid \mu_k^s, \Sigma_k^{ss})}{\sum_{k'=1}^{K} \pi_{k'} N(s_i \mid \mu_{k'}^s, \Sigma_{k'}^{ss})} \tag{7}$$

$$a_{i+1} = \sum_{k=1}^{K} h_k(s_i)\hat{\mu}_k^a(s_i) \tag{8}$$

Parameters related to GMM ($\Pi = \{\pi_k, \mu_k^z, \Sigma_k^{zz}\}$) can be estimated using the standard EM (Expectation Maximization) algorithm [21].

B. Transfer learning

When the training data for all the drivers considered are sufficient, the GMM-based model described above can be obtained for each individual driver, and in this case model adaptation is not required. In this paper, we are concerned about the situation in which enough training data are available for the source driver, while there is a lack of data for the target driver. Transfer learning offers a way to help the training process of target driver model by transferring the data collected from source driver to target driver. The data domain of the source driver is usually named the source domain, while the data domain for the target driver is the target domain.

The LPA-based transfer learning is used here to facilitate the data transfer. Let us assume that the dataset for source ($\mathrm{D^s}$) and target domain ($\mathrm{D^t}$) is collected from two manifolds $\mathrm{M^s}$ and $\mathrm{M^t}$ respectively. The goal of LPA is to find a non-linear mapping between these two manifolds. Usually it is very difficult to get this non-linear mapping directly. To simplify the problem, LPA approximates the non-linear mapping by dividing it into several locally linear mappings. Clustering algorithms can be used to split the datasets $\mathrm{D^s}$ and $\mathrm{D^t}$ into K regions first. Then, the traditional PA algorithm is applied independently to each region to learn the local mappings.

C. Modified EM initialisation

The GMM described in Subsection A can be directly used here to do the data clustering. Each mixture component of the GMM corresponds to a data region for PA. As mentioned in Subsection A, the EM algorithm can be applied to estimate the parameters of GMM. As we know, the performance of EM highly relies on the initialisation of GMM parameters. In the original LPA work presented in [15], the initialisation of GMM is carried out using a hierarchical clustering scheme. There is only one cluster for all the data involved at the beginning of the initialisation algorithm. Then this cluster is spitted into two or more iteratively when some predefined conditions are not satisfied. Although this method can find the initial number of clusters automatically, for large dataset it suffers from low efficiency. For the dataset with many changing points, such as the overtaking data considered in this paper, the original algorithm cannot cluster the data around the changing points properly, which leads to poor prediction performance for LPA. To avoid this problem, an EM initialisation algorithm based on the K-means method is proposed in TABLE I.

TABLE I. INITIALISATION ALGORITHM FOR EM

```
1. IN: Training sets $D^s$ and $D^t$ , cluster parameters $K_1$ and $K_2$
2. Compute local control changing rates $G^s \Leftarrow \text{localDiff}(D^s)$
3. $(G^s_I, G^s_{II}) \Leftarrow \text{Kmeans}(G^s, K)$ , where $G^s_I$ contains $G^s = 0$
4. $D^s$ is divided accordingly into $D^s_I$ and $D^s_{II}$
5. $(D^s_i) \Leftarrow \text{Kmeans}(D^s_I, K_1)$ , $i=1,2,...,K_1$
6. $(D^s_j) \Leftarrow \text{Kmeans}(D^s_{II}, K_2)$ , $j=K_1+1, K_1+2,..., K$
7. Compute $\Pi_0 = \{\mu_0, \Sigma_0, \pi_0\}$
8. OUT: $\Pi_0$
```

The changing points for the overtaking data are mainly caused by the steering wheel control of drivers. To capture the control change of drivers, the local control changing rate $g^s_i \in G^s$ is defined as follows.

$$g^s_i = \frac{a^s_{i+1} - a^s_{i-1}}{L^s_i} \tag{9}$$

$$L^s_i = \left\| s^s_{i+1} - s^s_{i-1} \right\| = (s^s_{i+1} - s^s_{i-1})^T (s^s_{i+1} - s^s_{i-1}) \tag{10}$$

where a^s_i is the steering wheel angle of the i th data point in the source dataset, s^s_i is the state vector of the i th data point, and L^s_i denotes the Euclidean distance between the 2 nearest state vector around it.

To start the initialisation algorithm, the training sets D^s, D^t and cluster parameters K_1 , K_2 should be given, where K_1 represents the number of clusters with relatively low control changing rates, while K_2 represents the number of other parts. To be consistent with the original LPA, we define $K = K_1 + K_2$. As shown in TABLE I, the whole initialisation algorithm contains two main clustering progresses. One is to cluster the source dataset in terms of the local control changing rate, and the other one is to cluster it according to the Euclidean distance between each point.

When the initial GMM parameters Π_0 are obtained, the EM algorithm will be triggered with the input of Π_0 and K . After that, the GMM model is trained using source dataset with its parameters Π as presented in Subsection A. Once the clustering work of the source dataset has been finished, the target dataset can be clustered accordingly based on the corresponding relationship between the source data points and target data points.

D. Local Procrustes Analysis

To start the LPA-based transfer learning, for each cluster $k \in [1, K]$, the training data should be pre-processed using the principal component analysis (PCA). The PCA process is shown as below, which can help to reduce the dimensionality of the data and facilitate the PA procedure [15, 22].

$$m^s = B^s_k (d^s - \omega^s_k) \tag{11}$$

$$m^t = B^t_k (d^t - \omega^t_k) \tag{12}$$

where $m^s \in M^s_k$, $m^t \in M^t_k$, $d^s \in D^s_k$ and $d^t \in D^t_k$ (d^s and d^t represents z_i in source and target domain respectively). The value $\omega^s_k = \mathbb{E}\{D^s_k\}$ and $\omega^t_k = \mathbb{E}\{D^t_k\}$ is respectively the mean vector of the local source dataset and local target dataset. B^s_k and B^t_k are local transformation matrices calculated using PCA. The detailed calculation procedure of B^s_k and B^t_k can be found in [17] and [22].

The local mapping function is supposed to be linear projections between local source manifolds and local target manifolds. The projection $f_k : M^s_k \to M^t_k$ is presented as:

$$m^t = f_k(m^s) = A_k m^s \tag{13}$$

where $A_k \in \mathbb{R}^{J \times J}$ is a transformation matrix of the k th cluster, and J is the number of main components computed by local dataset using PCA. Following [22], to learn the parameters in local transformation matrices A_k , local covariance matrices Σ^{ss}_k and Σ^{ts}_k are calculated using local manifolds M^s_k and M^t_k , so that A_k can be expressed as:

$$A_k = \Sigma^{ss^{-1}}_k \Sigma^{ts}_k \tag{14}$$

Then all of the LPA parameters calculated above could be presented as $\Theta = \{A_k, B^s_k, B^t_k, \omega^s_k, \omega^t_k\}$. Given parameters $\{\Pi, \Theta\}$, the data collected from the source driver can then be transferred to the target driver, using equations below [15].

$$h_k = \frac{\pi_k N(d^s \mid \mu_k, \Sigma_k)}{\sum_{i=1}^{K} \pi_i N(d^s \mid \mu_i, \Sigma_i)} \tag{15}$$

$$d^t = \sum_{k=1}^{K} h_k (B^{t^{-1}}_k A_k m^s + \omega^t_k) \tag{16}$$

Based on the original LPA, the initialisation algorithm for EM is modified in this paper to deal with the problem of modelling and transferring driver steering behaviour. Thus, for ease of expression, the new method will be named modified local Procrustes analysis, or MLPA for short in the rest part of this paper.

III. Experiments and Analysis

To test the performance of MLPA, the overtaking scenario shown in Figure 1 is built in PreScan [23], a software platform for simulating vehicle and traffic dynamics. According to the steering behaviour model described in Section II, we need to obtain three kinds of information, human driver information, host vehicle information and the information of the other two vehicles. All the information can be collected by the man-in-the-loop experiments using the driving simulator, Logitech G 27, shown in Figure 2.

Figure 2. The driving simulator

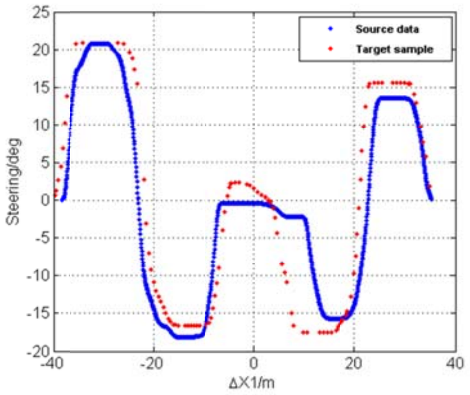

Figure 3. Example of source and target data smples

In order to obtain the information of host vehicle and other vehicles, three virtual sensing systems, GPS, Radar and Ladar are applied, which are embedded modules of PreScan. The information of the host vehicle (position and velocity) is obtained from the global positioning system (GPS). Radar and Ladar are employed here to obtain the information related to other vehicles (relative distance and velocity). For all the experiments, the initial positions of all vehicles are set the same. The speed for Vehicles 1 and 2 is set as 54 km/h. The initial relative distance between these two vehicles will keep at 35 m during the whole test.

The data collecting process is conducted with multiple drivers driving the host vehicle in the same simulation environment described above. Two drivers are selected to formulate the transfer learning problem, with one as the source driver and the other the target driver. The data related to the steering change will be sampled to form the source dataset D^s and target dataset D^t as shown in Figure 3. As mentioned before, the number of data in D^t should be much smaller than the number of data in D^s. In our experiments, 1200 data points are collected for the source driver, while less than 150 data are available for the target driver. To test the performance of MLPA three experiments are carried out. In all of the three experiments, we use $K_1 = 8$, $K_2 = 18$ and $K = 26$. These parameters are set to keep a balance between accuracy and efficiency of the algorithm as suggested by [24].

A. Experiment I: Evaluating MLPA

The first experiment is aiming to evaluate the transferring capability of MLPA. Here, 150 data samples are collected from the target driver. As there are sufficient data (1200 samples) from the source domain, the source driver model can be obtained directly using the GMM method described in Section II. A. Based on the transfer learning process presented in Section II. B, C and D, the source domain data can be transferred to the target domain without collecting enough data from the target driver. The experimental result for the transfer learning is shown as below.

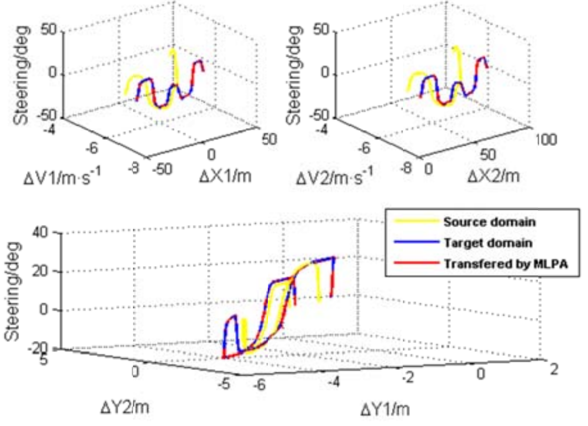

Figure 4. The result of MLPA

It can be seen from Figure 4. that the data collected from source domain are successfully transferred to the target domain. The curve of steering wheel angle transferred by MLPA has a very similar shape of the curve in the target domain. This result verifies the transferring ability of MLPA.

B. Experiment II: Comparing MLPA with GMM

To show the advantage of transfer learning, the GMM without transfer learning is compared with MLPA in this experiment. For the traditional GMM-based model, the source domain data are not involved in the training process, and the model is entirely determined by the data collected from target domain.

The number of samples collected from the target domain increases from 110 to 140, and the experiment is repeated 10 times for each number of samples to get the average prediction error. Examples of the experimental results are shown in Figures 5 to 8. The mean squared error (MSE) is used in this paper to capture the prediction error and shown in TABLE II. It is clear that with the increase number of samples, both MLPA and GMM can reduce the prediction error. However, the performance of GMM is not stable with such small number of training data and in some cases (see TABLE II) the MSE of GMM is even higher than 10. The performance of MLPA, on the other hand, is pretty stable with the MSE lower than 0.8 in all the cases. Without the help of the data from source domain, GMM has a much higher prediction error than MLPA in almost all the cases.

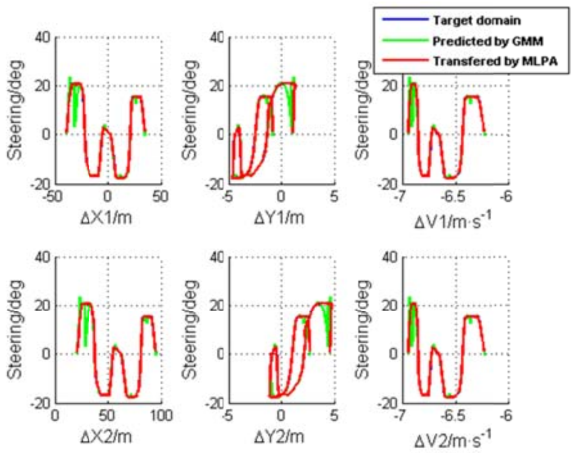

Figure 5. The result of MLPA and GMM with 110 samples

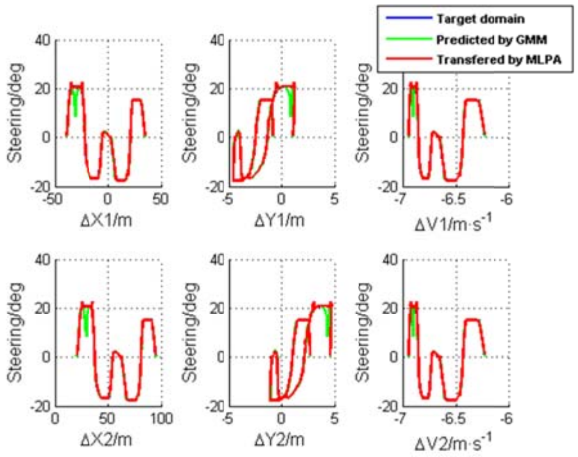

Figure 6. The result of MLPA and GMM with 120 samples

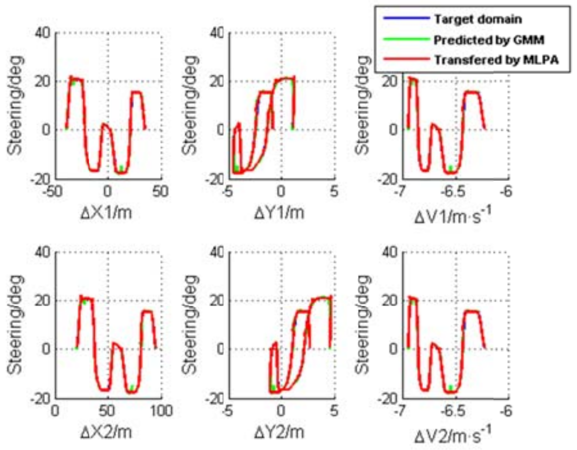

Figure 7. The result of MLPA and GMM with 130 samples

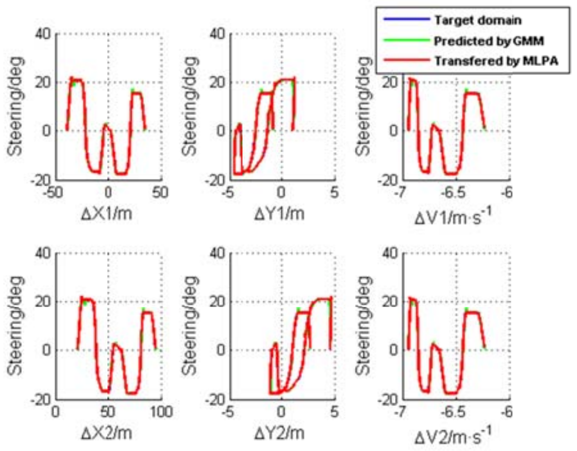

Figure 8. The result of MLPA and GMM with 140 samples

TABLE II. MSE OF MLPA AND GMM

	110 samples		120 samples		130 samples		140 samples	
	MLPA	GMM	MLPA	GMM	MLPA	GMM	MLPA	GMM
1	0.25	0.38	0.15	0.36	0.11	1.70	0.10	0.18
2	0.16	1.78	0.16	12.09	0.16	0.17	0.09	0.22
3	0.23	6.11	0.35	5.71	0.10	0.76	0.12	0.11
4	0.25	0.18	0.12	1.85	0.13	0.21	0.18	0.37
5	0.30	10.45	0.19	0.23	0.09	0.15	0.14	0.19
6	0.16	4.88	0.70	3.27	0.13	6.12	0.11	0.11
7	0.37	0.17	0.18	1.56	0.15	1.92	0.11	1.90
8	0.13	11.39	0.20	1.84	0.22	2.01	0.13	1.76
9	0.58	0.78	0.12	1.62	0.20	0.27	0.11	0.12
10	0.30	4.45	0.28	2.22	0.18	0.20	0.08	1.70
mean	0.27	4.06	0.25	3.08	0.15	1.35	0.12	0.67

C. Experiment III: Comparing MLPA with LPA

As mentioned in Section II that the main difference between MLPA and LPA is the initialisation algorithm, or specifically, the clustering algorithm. In this experiment, the performance of MLPA and LPA for learning the driver steering behaviour is compared.

130 data samples collected from the target domain are involved in the experiment. Similar to the above subsection, the experiment is repeated 10 times to get the average performance of both algorithms. To show the difference between two algorithms clearly, the curves are projected to the Steering- ΔX_1 coordinate system.

Figure 9 shows an example of comparison between two algorithms. MLPA can reproduce the curve of steering wheel angle of the target domain perfectly, while PLA does not perform well around the changing points (such as the data points between 15 to 20 m, and 20 to 30 m). It is mainly because the initialisation algorithm of MLPA can deal with the clustering problem of changing points properly by introducing the control changing rates (please refer to Section II. C.). Although the difference between MLPA and LPA is only obvious around the changing points, the MSE caused by this small difference is quite different in most cases. It can be seen from Figure 10 that the prediction error of PLA is much higher than MLPA in 9 of 10 experiments. This result proves the advantage of MLPA over PLA for modelling driver steering behaviour during overtaking.

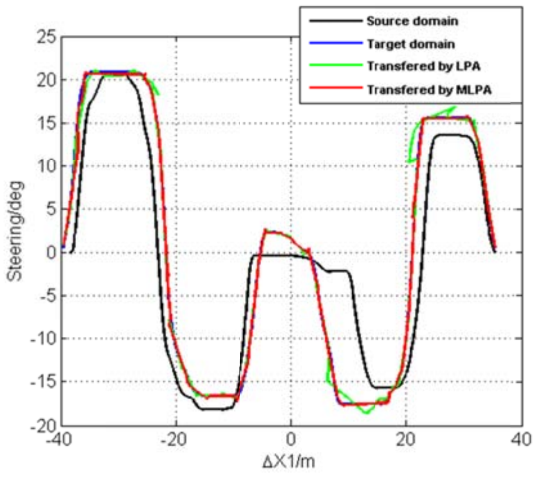

Figure 9. Thecomparison between MLPA and LPA

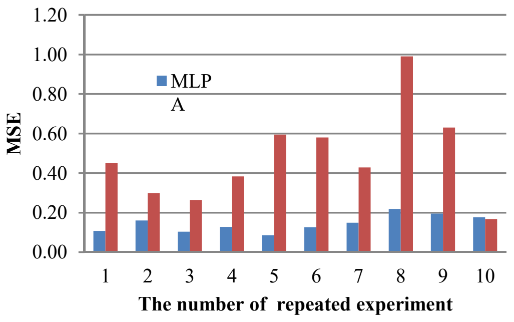

Figure 10. The MSE of MLPA and LPA

IV. Conclusions

In this paper, a novel method for driver model adaptation using transfer learning is developed for the overtaking scenario. The method is based on the modified local Procrustes analysis (MLPA), in which a new initialisation algorithm specific to driver steering behaviour is proposed to start the LPA procedure.

Based on the data collected from the human-in-the-loop experiment, the effectiveness of MLPA for transfer learning is evaluated. Then the superior performance of MLPA on dealing with small data samples is proven by a comparative study with the traditional GMM method. After that, the advantage of MLPA over LPA in dealing with the driver steering behaviour is shown. In the future work, except for the overtaking scenario, more different scenarios will be involved to test the performance of MLPA. To avoid the drawback of the exact alignment in MLPA, another knowledge transfer mechanism based on the data distribution will be considered.

References

[1] J.-H. Kim *et al.*, "Modeling of driver's collision avoidance maneuver based on controller switching model," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* vol. 35, no. 6, pp. 1131-1143, 2005.

[2] H. Okuda, N. Ikami, T. Suzuki, Y. Tazaki, and K. Takeda, "Modeling and Analysis of Driving Behavior Based on a Probability-Weighted ARX Model," *IEEE Transactions on Intelligent Transportation Systems,* vol. 14, no. 1, pp. 98-112, 2013.

[3] S. Sekizawa *et al.*, "Modeling and Recognition of Driving Behavior Based on Stochastic Switched ARX Model," *IEEE Transactions on Intelligent Transportation Systems,* vol. 8, no. 4, pp. 593-606, 2007.

[4] T. Taniguchi, S. Nagasaka, K. Hitomi, N. P. Chandrasiri, T. Bando, and K. Takenaka, "Sequence prediction of driving behavior using double articulation analyzer," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 46, no. 9, pp. 1300-1313, 2016.

[5] T. Taniguchi, K. Furusawa, H. Liu, Y. Tanaka, K. Takenaka, and T. Bando, "Determining utterance timing of a driving agent with double articulation analyzer," *IEEE Transactions on Intelligent Transportation Systems,* vol. 17, no. 3, pp. 810-821, 2016.

[6] J. Zheng, K. Suzuki, and M. Fujita, "Car-following behavior with instantaneous driver–vehicle reaction delay: A neural-network-based methodology," *Transportation research part C: emerging technologies,* vol. 36, pp. 339-351, 2013.

[7] C. Colombaroni and G. Fusco, "Artificial neural network models for car following: experimental analysis and calibration issues," *Journal of Intelligent Transportation Systems,* vol. 18, no. 1, pp. 5-16, 2014.

[8] J. Morton, T. A. Wheeler, and M. J. Kochenderfer, "Analysis of recurrent neural networks for probabilistic modeling of driver behavior," *IEEE Transactions on Intelligent Transportation Systems,* vol. 18, no. 5, pp. 1289-1298, 2017.

[9] P. Angkititrakul, C. Miyajima, and K. Takeda, "Modeling and adaptation of stochastic driver-behavior model with application to car following," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, 2011, pp. 814-819: IEEE.

[10] A. Y. Ungoren and H. Peng, "An adaptive lateral preview driver model," *Vehicle system dynamics,* vol. 43, no. 4, pp. 245-259, 2005.

[11] P. Angkititrakul, C. Miyajima, and K. Takeda, "An improved driver-behavior model with combined individual and general driving characteristics," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, 2012, pp. 426-431: IEEE.

[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering,* vol. 22, no. 10, pp. 1345-1359, 2010.

[13] D. Han, Q. Liu, and W. Fan, "A new image classification method using CNN transfer learning and web data augmentation," *Expert Systems with Applications,* vol. 95, pp. 43-56, 2018.

[14] D. Wang and T. F. Zheng, "Transfer learning for speech and language processing," in *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2015, pp. 1225-1237.

[15] N. Makondo, B. Rosman, and O. Hasegawa, "Knowledge transfer for learning robot models via local procrustes analysis," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, 2015, pp. 1075-1082: IEEE.

[16] B. Delhaisse, D. Esteban, L. Rozo, and D. Caldwell, "Transfer learning of shared latent spaces between robots with similar kinematic structure," in *2017 International Joint Conference on Neural Networks (IJCNN),* 2017, pp. 4142-4149: IEEE.

[17] B. Bocsi, L. Csató, and J. Peters, "Alignment-based transfer learning for robot models," in *2013 International Joint Conference on Neural Networks (IJCNN),* 2013, pp. 1-7: IEEE.

[18] L. Cardamone, A. Caiazzo, D. Loiacono, and P. L. Lanzi, "Transfer of driving behaviors across different racing games," in *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, 2011, pp. 227-234.

[19] S. Lefèvre, A. Carvalho, Y. Gao, H. E. Tseng, and F. Borrelli, "Driver models for personalised driving assistance," *Vehicle System Dynamics,* vol. 53, no. 12, pp. 1705-1720, 2015.

[20] C. Miyajima and K. Takeda, "Driver-Behavior Modeling Using On-Road Driving Data: A new application for behavior signal processing," *IEEE Signal Processing Magazine,* vol. 33, no. 6, pp. 14-21, 2016.

[21] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[22] C. Wang and S. Mahadevan, "Manifold alignment using procrustes analysis," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1120-1127: ACM.

[23] T. I. Prescan, *A Simulation & Verification Environment for Intelligent Vehicle Systems*. 2015.

[24] C. Lu, H. Wang, C. Lv, J. Gong, J. Xi, and D. Cao, "Learning Driver-specific Behaviour for Overtaking: A Combined Learning Framework," *IEEE Transactions on Vehicular Technology,* 2018.