Vehicle Activity Recognition Using DCNN

Alaa AlZoubi[1] and David Nam[2]

[1] School of Computing, The University of Buckingham, Buckingham, United Kingdom
alaa.alzoubi@buckingham.ac.uk
[2] Propelmee Ltd, Milton Keynes, United Kingdom
david@propelmee.com

Abstract. This paper presents a novel Deep Convolutional Neural Network (DCNN) method for vehicle activity classification. We extend our previous approach to be able to classify a larger number of vehicle trajectories in a single network. We also highlight the flexibility of our approach in integrating further scenarios to our classifier. Firstly, a spatiotemporal calculus method is used to encode the relative movement between vehicles as a trajectory of QTC states. We then map the encoded trajectory to a 2D matrix using the one-hot vector mapping, this preserves the important positional data and order for each QTC state. To do this we associate the QTC sequences with pixels to form a 2D image texture. Afterwards, we adapted trained CNN architecture into our vehicles activity recognition task. Two separate types of driving data sets are used to evaluate our method. We demonstrate that the proposed method out-performs existing techniques. Along with the proposed approach we created a new dataset of vehicles interactions. Although the focus of this paper is on the automated analysis of vehicle interactions, the proposed technique is general and can be applied for pairwise analysis for moving objects.

Keywords: Vehicle Activity Classification · Spatiotemporal Calculus · Trajectory Texture · Transfer Learning · Deep Convolutional Neural Networks.

1 Introduction

Vehicular collisions are typically the result of a deficient situational understanding of the surrounding scene, and therefore being unable to identify and act upon dangerous situations, before there are consequences. Being able to understand the relative interaction between a vehicle and its surroundings is imperative in defining the situation a vehicle is in, or about to enter, and hence to plan accordingly and do appropriate decision making [16]. The objective of vehicle activity recognition is to classify a set of actions (from one of more vehicles) for a sequence of observations. During dynamic traffic scenarios complex interactions can occur, either between vehicles themselves, or even a possible broken

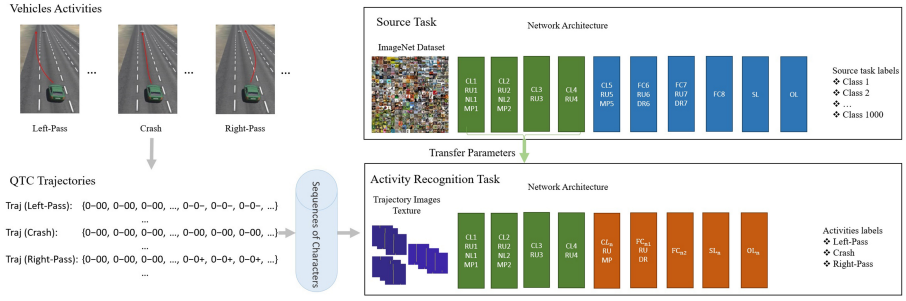


Fig. 1. Overview of presented approach. Representation of trajectories using QTC (left). Customised CNN for activity classification (right) [4].

or stalled vehicles; in this situation we refer to the stationary vehicle as an obstacle. Larger group interactions can then be constructed from the pair-wise interactions —vehicle-vehicle or vehicle-obstacle.

There are two main approaches for trajectory analysis, quantitative and qualitative approaches. Traditionally there has been significant research within the quantitative approach; which uses sequences of real-valued features (trajectories) [25, 10, 12, 13, 15, 27], however, recent interest has grown around qualitative approaches. Qualitative methods use a symbolic representation of scenarios of moving objects, and have shown superior performance for vehicle activity analysis when compared with quantitative methods [2]. It has found applications in vehicle interaction and human behavior analysis [2], and human-robot interaction [9]. Along with improved performance there are also other benefits for a qualitative approach, such as having a more compact representation (and therefore is more computationally efficient). Qualitative representations are also a more natural way of describing interactions [7]; where it is more flexible to variations in trajectories, but still captures the overall behaviour.

Within the context of activity recognition having a compact and informative representation for encoding trajectories, of moving objects, can be beneficial. [12] encodes trajectories within a two dimensional matrix, while [20] encodes them in a trajectory texture image. Both approaches were shown to be successful in different application domains, human activity recognition [20, 19] and pair-wise vehicles activity recognition [12]. Based on this the 2D representations are then used to train a classifier to perform the activity recognition task. Also, with the recent success of deep learning methods for image classification, representing trajectories within a 2D image enable the capabilities of previously built and trained networks. In particular neural networks are well suited for learning features based on the shape and texture of an object.

In this series, we present our novel approach for vehicle pair-activity recognition and classification, based on QTC and DCNN. Our method consists of two stages, firstly we employ QTC as a means to, compactly, represent the relative motion between pairs of objects (vehicle-vehicle or vehicle-obstacle). We then

encode their interactions as a trajectory of QTC states. To convert this representation as a 2D image we use one-hot vectors to convert QTC sequences to a two dimensional matrix (or image texture). The second stage of our approach is activity classification, using the more encoded trajectory texture image. To do this we adapt, an already trained, DCNN (trained on the ImageNet dataset for image classification). The motivation for this was to use the already trained layers as a starting point to transfer knowledge to the vehicle pair-activity recognition task. A unique image signature (or texture) is produced for each activity. We evaluate our method using a dataset of vehicle-obstacle interactions, which we have captured ourselves. We also present a detailed comparison against state-of-the-art quantitative and qualitative methods, using different datasets (including our own). Moreover, results demonstrate that our proposed approach gives higher performance when compared with current methods for pair-wise vehicles trajectory classification. An overview of the method and main contributions of our work is shown in Figure 1 .

This article is an extension to our work previously presented [4]. We have evaluated our approach on a new dataset which combines vehicles activity trajectories from different data sources. We have extended the categories of our activity classifier, while still achieving high performance. Further detailed evaluations on this dataset are given in Section 4.4; where our approach is now able to accurately distinguish between eight different pairwise activities, without compromising on accuracy. This allows for a more general method, which can be used for applications with a wide range of scenarios. To highlight the main contributions made within this paper, we demonstrate, for the first time, a novel approach for classifying pair-wise vehicle activities using QTC with DCNNs.

Our work is primarily motivated by our interest in the automated recognition vehicle's scenario or activity, however, being able to predict a vehicle's future trajectory can help in avoiding any potential collisions. To be able to gain traction and boost usability as a main-stream analysis method, we present a novel driver model for predicting a vehicle's future trajectory, from its partially observed prior trajectory. Main novel aspects of this work are as follows: proposing a new CNN tuned for pair-wise vehicle activity recognition, (using a modified version of AlexNet [11]). The second contribution is a novel method for vehicle activity classification, based on a vehicle's potential future scenario; to achieve this we utilise a driver model to produce likely trajectories. Lastly, through experimentation we demonstrate that our proposed CNN out-performs current approaches (which includes [2, 12, 13, 15, 27]). Additionally, we include a new, open-source, dataset of pairwise vehicle-obstacle interactions, (with associated ground-truth). It consists of 554 vehicle scenarios (complete and incomplete scenarios) for three different types of interactions.

2 BACKGROUND

In this section we give a brief overview of the state-of-the-art in trajectory analysis, qualitative trajectory calculus, and deep learning.

2.1 Trajectory Analysis Techniques

In order to do trajectory analysis a spatial-temporal representation of motion information has to be defined. Currently there are numerous techniques for trajectory representation; in order to encode a sequence of continuous states in an efficient and fast way. A long-term motion descriptor given the name, sequential deep trajectory descriptor (sDTD), was presented in [19]. Their technique initially determines the simplified dense trajectories of a single object and then converts these trajectories into 2D images. Chavoshi et al. [5] presented a visualization technique, sequence signature (SESI), to convert a basic variation of QTC (QTC_B) movement patterns of moving point objects into a 2D indexed rasterized matrix. The approach in [12] represents trajectories of pair-vehicles as a series of heat sources, where a thermal diffusion process creates an activity map as a 2D matrix. The above techniques either encode trajectories of a single object, or utilise traditional similarity methods (e.g. Euclidean distance) which unable to cope with varying lengths trajectories and compound behaviors. The vehicle activity analysis tackled in this paper requires a more general technique, invariant to trajectory length and compound behaviors. It has been demonstrated that quantitative [12] and qualitative [2] methods are the most efficient for encoding vehicle pair-wise activity. We thus utilise both as the standard against which we evaluate our own work.

Approaches for trajectory analysis are grouped into three categories: single-role activities [26], pair-wise-activities [2], and group-activities [12]. Typically these techniques were focused on specific types of behaviours: human behavior recognition [2, 12, 27], human-robot interaction [9], animal behavior clustering [2], or vehicles interaction recognition [2, 25, 10, 12]. Quantitative features were mainly used in [12] for modelling traffic behaviours, and the same was done for autonomous driving applications in [24]. [12] proposed an algorithm for vehicle pair activity recognition using a heat map. There vehicle trajectories were represented as an activity map and then a Surface-Fitting method was used to group the resulting vehicle activities.

An approach using qualitative features for traffic activity recognition was proposed in [2]. There a Normalized Weighted Sequence Alignment technique was introduced, to calculate the similarity between QTC sequences. They evaluated their approach using three different datasets: human generated trajectories, trajectories from animals, and trajectories from vehicle interactions. It was demonstrated that their techniques provided improved performance over other state-of-the-art quantitative methods [12, 13, 15, 27]. The latter are more closely associated with the work proposed in this paper. Hence, we use these approaches to compare our method. We also do a full comparison using the vehicle-interaction dataset [12], provided with ground truth, to give a complete evaluation of our recognition and classification approach (Section 4.1). Within this work we will focus on pair-wise vehicle-to-vehicle and vehicle-to-obstacle interactions. For a deeper understanding of trajectory analysis we refer to reader to [1]; where further review is provided.

2.2 Trajectory Representation

We have now established that for robust and efficient trajectory analysis being able to represent trajectories, in a compact and meaningful way, is essential. Qualitative spatial-temporal reasoning is a class of techniques which processes trajectory information in a similar manner to the human perception system, in terms of relative interactions. It does so by using symbolic representations, of specific classes of interactions, rather than numeric measurements [23]. For activity analysis, approaches using QTC representations have demonstrated a higher level of robustness and performance, when contrasted against quantitative methods; in different application areas, including: human activity analysis and recognition of vehicle interaction [2].

Four features are used as a base unit for describing more complex pair-wise trajectory information. Given the coordinates of two objects, k and l, their interactions can be described using QTC as follows:

1. **Distance Features**
   - S_1: distance of k with respect to l: "-" indicates decrease, "+" indicates increase, "0" indicates no change.
   - S_2: distance of l with respect to k.

2. **Speed Features**
   - S_3: Relative speed of k with respect to l at time t (which dually represents the relative speed of l with respect to k).

3. **Side Features**
   - S_4: Displacement of k with respect to the reference line L connecting the objects: "-" if it moves to the left, "+" if it moves to the right, "0" if it moves along L or not moving at all.
   - S_5: Displacement of l with respect to L.

4. **Angular Features**
   - S_6: The respective angles between the velocity vectors of the objects and vector L: "-" if $\theta_1 < \theta_2$, "+" if $\theta_1 > \theta_2$ and "0" if $\theta_1 = \theta_2$

here S_i represents the qualitative relations in QTC. Figure 2 demonstrates the concept of qualitative relations within QTC for two disjoint objects, (k and l). Three main calculi were defined [23]: QTC_B, QTC_C and QTC_{Full}. Where QTC_C (S_1, S_2, S_4 and S_5) and the combination of the four codes results in $3^4 = 81$ different states.

2.3 Deep Neural Networks

The increasing popularity of convolutional neural networks have demonstrated their applicability for object recognition; in particular when it comes to very large-scale visual recognition probelms [18]. Much of its success is attributed to

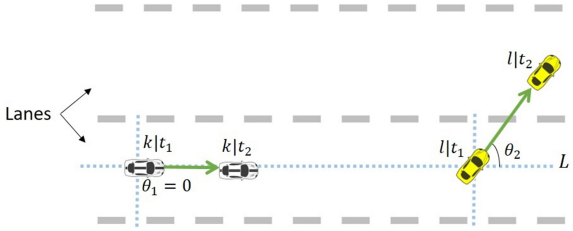


Fig. 2. Diagrammatic representation showing QTC relations between two moving vehicles: $QTC_C = (-, +, 0, +)$ [4].

the ability for CNNs to learn base image features and then combine them to form, discriminative and robust, object features (such as shapes and textures) [17]. Some successful CNNs for image classification are presented in AlexNet [11] and GoogLeNet [22], which were both designed within the context of the "Large Scale Visual Recognition Challenge" (ILSVRC) [18] for the ImageNet dataset [6]. For a further in-depth analysis we refer the reader to [14], which provides a review of deep neural network architectures and their applications for object recognition.

Within our proposed approach for activity recognition we uslise AlexNet [11]. This model provided a good foundation for activity recognition, as it was trained on approximately 1.2 million labeled images, consisting of 1,000 different categories from the ILSVRC dataset [18]; as a point to note, each image in this dataset has one centrally located object, which occupies a significant portion of the image and there is also limited background clutter. This has allowed the network to learn a wide a robust set of base features. AlextNet uses the entirety of the image as an input and produces probabilities for each class. In terms of the network structure there are 650,000 neurons and 60 million parameters in AlexNet. Its architecture is made up of two normalisation layers, three fully-connected layers, three max-pooling layers, five convolutional layers, and a linear layer with softmax activation, to produce the probabilistic classification outputs. Dropout regularization [21] is employed as a means to reduce overfitting in the fully connected layers. Rectified Linear Units (ReLU) are also used as the activation of the layers, and to provide non-linearity to the system.

A novel aspect of this work involved customizing AlexNet for the activity recognition task, and evaluating the performance of our improved DCNN architecture on multiple datasets.

3 THEORETICAL FORMULATION

In Figure 1 we overview the key steps and contributions for our vehicle activity recognition method; where the right side demonstrated our novelty in our deep learning framework. Within our first step QTC trajectories are computed from consecutive observations and then projected onto 2D matrices, (here observations consist of the 'x' and 'y' coordinates). At this point the resulting 2D matrices (or

trajectory texture images) now encode the relative motions' of pairs of vehicles at consecutive time frames, $[t_1, t_N]$.

One of the main novelties arising from this work is given in *TrajNet*; a new deep learning network, which uses transfer learning, using trajectory texture images, to effectively learn the features presented in varying vehicle activities. Additionally, we demonstrate the use of a Neural Network (NN) model for predicting a vehicle's future trajectory, for our set of scenarios, from an incomplete one. In our experimental section we demonstrate that our method is generalizable across multiple scenarios and data types, is compatible with complete and predicted trajectories. Moreover, we also demonstrate that the proposed method, consistently, out-performs other state-of-the-art approaches.

We summarize the key novel contributions of this work as follows: Representing specific scenarios using sequences of QTC states, for a pair of vehicles' relative movements. We introduce a technique to represent QTC sequences as an image texture, based on the one-hot vector encoding. A novel CNN model is proposed for vehicle activity recognition; it utilizes AlexNet and the ImageNet dataset, but improves upon the base network, for vehicles pair-activity recognition task. This results in our new network *TrajNet*. For predicting a complete trajectory from a partially observed one, a NN based human driver model is presented.

3.1 Encoding Pair-wise Vehicle Interactions

Representing Vehicle Behaviour with QTC Given x, y, in the top-down coordinate space, as the two-dimensional position of the centroid of a vehicle, we employ QTC for representing the behaviour of a pair of interacting vehicles. Here we consider interacting vehicles to be in close enough proximity that its presence could potentially influence the actions and decision making of the other vehicle. Changes in their relative positions, (hence their interactions) are expressed as a sequence of QTC states. We use the common QTC variant (QTC_C) within our approach. The common QTC variant is represents the vehicle pair's relative two-dimensional movement qualitatively.

Definition: Given two interacting vehicles, or vehicle/obstacle pair, interacting —in this case the presence of an obstacle in the path of a vehicle will cause an action, likely to avoid it, to be taken —with one another and their centroid x, y coordinates, we define:

$$V1_i = \{(x_1, y_1), ..., (x_t, y_t), ..., (x_N, y_N)\}, \tag{1}$$
$$V2_i = \{(x'_1, y'_1), ..., (x'_t, y'_t), ..., (x'_N, y'_N)\}, \tag{2}$$

here (x_t, y_t) and (x'_t, y'_t) are the centroids of the first interacting vehicle and second centroids of the interacting vehicles, at time t, respectively. The pair-wise trajectory is therefore represented as a sequence of ordered QTC_C states: $Tv_i = \{S_1, ..., S_t, ..., S_N\}$, where S_t is the QTC_C state representation of the

relative movement of the two vehicles (x_t, y_t) and (x'_t, y'_t) at time t in trajectory Tv_i. N is the number of observations in Tv_i.

Transforming Sequential QTC States to a 2D Image Texture The time varying sequence of QTC_C states, described in Section 3.1, is a one-dimensional succession of QTC_C states. It can be seen as analogous to both vehicles trajectories. When drawing a comparison with text information, there are limitations; such as where are no spaces between QTC_C states and there are no concepts of words. In order to be able to decode a higher level of information from these succession of states, we translate QTC_C trajectories into sequences of characters; with the aim of applying the same representation technique for text data without losing location information of each QTC_C state in the sequence. We then represent this sequence numerically, so that it is able to be used as an input for our CNN. Further discussion on our contribution to this is covered in Section 3.2.

To be able to achieve this we first represent the QTC_C states using the a

Algorithm 1 Image representation of QTC trajectory.

Input: set of trajectories $\zeta = \{Tv_1, ..., Tv_i, ...Tv_n\}$ where n is the number of trajectories in ζ
Input: QTC_C states Cr: $cr_1(- \ - \ - \ -), ..., cr_{81}(+ \ + \ + \ +)$
Output: n 2D matrices (images I) of movement pattern
Extract: sequences of characters $\zeta_C = \{Cv_1, ..., Cv_i, ...Cv_n\}$ from QTC_C trajectories ζ
Define: a 2D matrix (I_i) with size $(N \times 81)$ for each sequence in ζ_C, where 81 is the number of characters in Cr and N is the length of Cv_i
Initialise: set all the elements of I_i into zero
Update: each matrix in I:
for $i = 1$ to n **do**
    for $j = 1$ to N **do**
        $I_i(j, Cv_i(j)) = 1$
    end for
end for
return I

symbolic representation Cr: cr_1, cr_2, ..., cr_{81}. We then map our representation, the one-dimensional sequence of characters (or QTC_C trajectory), onto a two-dimensional matrix (image texture). This is done using one-hot-vector representation, to efficiently evaluate the similarity of relative movement. This results in images texture which we use as an input to train our network ($TrajNet$), with the goal of being able to distinguish between different vehicle behaviours.

Definition: Given a set of trajectories $\zeta = \{Tv_1, ..., Tv_i, ...Tv_n\}$ where n is the total number of trajectories in ζ, we convert each QTC trajectory; calculated from ζ, to form a sequences of characters $\zeta_C = \{Cv_1, ..., Cv_i, ...Cv_n\}$. Following

this, we represent each sequence Cv_i in ζ_C as an image texture I_i using one-hot vector representation. Here the columns represent Cr (or QTC_C states) and the rows indicate the presence of a unique character (or QTC_C state) at a specific time-stamp. Algorithm 1 describes mapping QTC_C trajectories into image texture. For example, the relative motion between two vehicles, as seen in figure 7(a), one vehicle is passing by the other vehicle (or obstacle) towards the left during the time interval t_1 to t_e. This interaction is described using QTC_C: $(0\ -\ 0\ 0, 0\ -\ 0\ 0, ..., 0\ -\ 0\ -, 0\ -\ 0\ -, ...)_{t_1-t_e}$ or $(cr_{32}\ cr_{32}\ ...\ cr_{31}\ cr_{31}\ ...)_{t_1-t_e}$. This trajectory can be represented as an image texture I_i using our Algorithm 1.

3.2 CNN based Activity Classification

Upon creating our two-dimensional image representation of QTC sequences, from the pair-wise vehicle trajectories, we are able to train our proposed CNN, using these images as inputs. Classification of the generated image textures is still a challenging task; hence our motivation for transfer learning. Given a scenario between two vehicles, despite the overall activity between two vehicles being the same, there can be countless variations in the trajectories and the resulting activity image. These variations can be due to multiple factors: variations in environmental conditions, influences from other actors, and the different driving styles of people. Considering all these factors, the types of variations are not tractable, and to be able to efficiently model this we adopted a CNN based approach for our activity recognition algorithm.

We decided to use AlexNet; as it is a proven and well established classification network. Its structure is utilised, which has shown to be able to classify many images with complex structures and features, as a base layer for our approach. Transfer learning is done on this network, as completely learning the parameters of this network from start would be unrealistic, do to the relatively small images texture of QTC trajectories available in our dataset (where AlexNet was trained on the order of a million images). As a result of AlexNet being trained on such a large number of images for a general classification task, the earlier layers are well tuned for extracting a wide array of basic features (edges, lines, etc.). We take advantage of this by only replacing and fine-tuning the final convolutional layer ($CL5$), the last three fully connected layers ($FL6$, $FL7$ and $FL8$), softmax (SL) and the output layer (OL), this results in our new network *TrajNet* (Figure 1).

For the final convolutional layer we uses a smaller layer CL_n, which consists of 81 convolutional kernels. This was preceded by ReLU and max pooling layers (using the same parameters as in [11]). We then incorporated one fully connected FC_{n1}, with 81 nodes. This is used in place of the the last two fully connected layers ($FL6$ and $FL7$), of 4096 nodes each. The number of nodes used in the final layers is correlated with the reduction in higher level features in our trajectory texture images (as opposed to [6]). This gives more tightly coupled responses. After our new FC_{n1} we include a ReLU and dropout layer (50%). Based on the number of vehicle activities (a) defined in the dataset, we add a final new fully connected layer (FC_{n2}) to match the a classes. A softmax layer (SL_n), and a classification output layer (OL_n) were also added to reflect the number of classes.

The output of the final fully-connected layer is passed to an a-way softmax (or normalized exponential function) which produces a probability distribution over the a class labels. The following are training and testing procedures using our image texture I. Each image texture (I_i) is used as input for the data layer *data* for the network. The network parameters were initialised as follows: iteration number = 10^4, initial learn rate = 10^{-4} and mini batch size = 4. These parameters were chosen empirically and were based on fine tuning the later layers for the activity recognition task. The other network parameters were set according to [11].

3.3 Extending Vehicle Activity Classification with Predictive Trajectory Modelling

In this section we introduce a novel approach to predicting a vehicle's future trajectory, given its and its pair's states. Ideally, the earlier we are able to identify which scenario a vehicle is in the more time there is to take appropriate action. This step would act as a prerequisite for pair-wise vehicle activity classification, in order to better identify scenario sooner.

To be able to predict the future pathway a vehicle may take, within the context of a specific scenario, we propose a Feed Forward Neural Network (FFNN). Details of our driver model, to predict full vehicle trajectories using partially observed (or incomplete trajectories), are shown in Figure 3. The proposed model architecture is made up of 9 hidden layers; where each hidden layer has z, such that $\{z \in \mathbf{Z} : \mathbf{Z} = [10, 10, 20, 20, 50, 20, 20, 20, 15]\}$. Given a hidden layer H^i with z nodes, that layer is defined such that $z \widehat{=} i^{th}$ element in $\mathbf{Z}$. This configuration was determined empirically, and was well suited to model the complex and intricate decisions a human driver a human can make. Our trajectory prediction model is unidirectional, where inputs are fed sequentially first through the input layer, then processed in hidden layers and finally passed to the output layers. Each layer is made up of nodes, where these nodes have weights and biases associated with each input to the node. To determine the output for a given node the sum of the weighted inputs, along with addition of the bias value, are calculated and then passed through an activation function; here we use a hyperbolic tangent function. The FFNN can be conceptualised as a way to approximate a function, where the values of each node's weight and bias are learned through training. For training data we use prior (incomplete) trajectories with their correct or desired output trajectory. Training is done through Levenberg-Marquardt backpropagation with a mean squared normalized error loss function.

Definition: Given x, y and x', y' as centroid positions of the ego-vehicle and obstacle, respectively, we calculate the relative changes in the ego-vehicle's heading angle and translation of the ego-vehicle between times $(t-1)$ and t

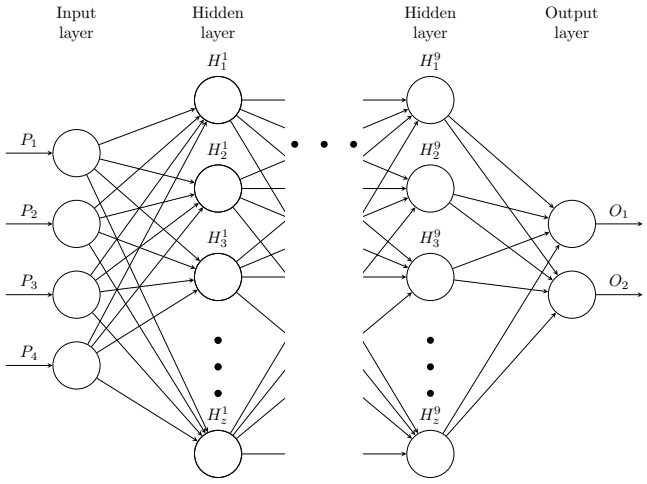


Fig. 3. Network architecture for trajectory prediction model. Layers with their associated nodes are shown. The inputs, outputs and hidden states, are labelled P, O and H, respectively. Note that we use 9 hidden layers, each with z hidden states [4].

follows:

$$\theta(t) = \tan^{-1}(\frac{(y_t - y_{t-1})}{(x_t - x_{t-1})}, \tag{3}$$

$$\nabla(t) = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}, \tag{4}$$

Here $\theta(t)$ is the ego-vehicle heading angle and $\nabla(t)$ is the magnitude of the change in the ego-vehicle's motion. Both features ($\theta(t)$ and $\nabla(t)$) were used as prior information in the training of our FFNN. To account for the effects of noise, (mainly associated with tiny fluctuations in the ego-vehicle's heading angles, caused by the driver), we calculate a moving average of the heading angle and translation over the previous $0.5s$. With this prior information we can define the inputs to our FFNN as:

$$\Theta(t) = {}^1\!/_5 \sum_{i=(t-5)}^{t} \theta(i), \tag{5}$$

$$\mathcal{R}(t) = {}^1\!/_5 \sum_{i=(t-5)}^{t} \nabla(i), \tag{6}$$

$$\beta(t) = \sum_{j=1}^{t} \Big(\nabla(j) \sin\big(\sum_{i=1}^{j} \theta(i)\big)\Big), \tag{7}$$

$$\lambda(t) = \sqrt{(x_t - x_t')^2 + (y_t - y_t')^2}. \tag{8}$$

Where $\lambda(t)$ is the distance between the ego-vehicle and the object. To help bound the driver model from going off the road when avoiding the obstacle we introduced $\beta(t)$, which is the lateral shift from the centre of the road. Θ and $\mathcal{R}$ are directly related the ego-vehicle movement, β relates the vehicle location to the road, and λ corresponds to the vehicle and obstacle interaction. The inputs are further visualised in figure 3, where $[P_1, P_2, P_3, P_4] \equiv [\Theta(t), \mathcal{R}(t), \beta(t), \lambda(t)]$ and outputs are $[O_1, O_2] \equiv [\theta(t+1), \nabla(t+1)]$. To determine the future trajectory the algorithm in run iteratively, producing a future motion at each time-step. This is described in Algorithm 2.

Fig. 4. Example pair-wise manoeuvres for traffic dataset used in [4], based on work presented in [12].

Algorithm 2 Vehicle trajectory prediction.

▷ A distance of ε meters between the vehicle and the obstacle was chosen to start prediction.

if $(\lambda(t) < \varepsilon)$ **then**
  while $((x_t < x'_t) \wedge (y_t < y'_t))$ **do**
    Inputs: $[\Theta(t), \mathcal{R}(t), \beta(t), \lambda(t)]$, using (5)-(8), respectively;
    Outputs: $[\theta(t+1), \nabla(t+1)]$;

$$\begin{cases} x_{t+1} = \nabla(t+1)\cos(\sum_{i=1}^{t+1}\theta(i)) + x_t, \\ y_{t+1} = \nabla(t+1)\sin(\sum_{i=1}^{t+1}\theta(i)) + y_t. \end{cases}$$

    Increment time-step: $t = t + 1$;
  end while
end if

4 EVALUATION

Within our experimental section we explore the performance of our vehicle activity recognition approach by doing multiple comparisons, on different datasets and against other methods. We contrast the performance of our classification approach (Section 3.2) with state-of-the-art quantitative and qualitative approaches to activity classification [2, 12, 13, 15, 27]. We also evaluate our driver model by comparing with our manually obtained ground truth. We demonstrate

Table 1. Definitions of vehicle manoeuvres as described in [4].

Activity	**Definition**
Turn	One vehicle moves straight and another vehicle in another lane turns right.
Follow	One vehicle followed by another vehicle on the same lane.
Pass	One vehicle passes the crossroad and another vehicle in the other direction waits for green light.
Bothturn	Two vehicles move in opposite directions and turn right at same time.
Overtake	A vehicle is overtaken by another vehicle.

the generality of our approach and explain how to can be applied to other scenarios. We also show that gives the best performance, compared with the other approaches; making it a suitable choice for time and safety critical applications.

4.1 Datasets and Experimental Set-up

In this section we explain our evaluation methodology, firstly we present our experimental approach and data used, after which we present a detailed evaluation of our approaches. We performed our tests on two publicly available datasets. The two different datasets represent different application domains, motion and interaction from vehicle traffic (obtained from surveillance cameras [12]) and vehicle-obstacle interactions, for modelling more scarce scenario of a potential collision [3]. Two state-of-the-art pair-wise activity recognition approaches [2, 12] have been demonstrated to give superior performance (on the [12] dataset) when compared with a number of other methods [27, 15, 13]. For consistency this has motivated up to compare our approach using these algorithms, the traffic dataset, and ground truth, as a base for evaluating our own activity classification technique. [12] is also, to best of our knowledge, the only pair-wise traffic surveillance dataset publicly available. All experiments were conducted on an Intel Core i7 desktop, CPU@3.40GHz with 16.0GB RAM.

Traffic Motion Dataset The traffic dataset was obtained by extracting trajectories from 20 surveillance videos; see [12] for more details. Five unique vehicle activities, *Turn*, *Follow*, *Pass*, *BothTurn*, and *Overtake*, are defined and their corresponding annotations are provided. 175 clips are presented in total, each activity having 35 clips. Here clips are composed of segments with 20 frames each The dataset provides x, y coordinates associated with the centroid of each vehicle, for each frame and time-stamp t. Figure 4 gives representative frames from the dataset. Table 1 details the definitions of each activity in the dataset.

Vehicle Obstacle Dataset For our second dataset [3] we focused on close proximity manoeuvring for vehicle/obstacle interactions. These types of scenar-

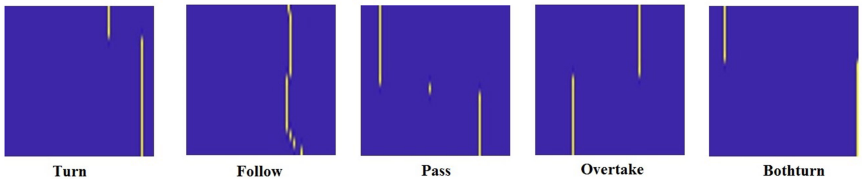


Fig. 5. Our image texture representation of QTC encode trajectories [4]. Scenarios from traffic dataset [12].

Table 2. Description of pair-wise vehicle activities in vehicle/obstacle dataset [3].

Scenario	**Description**
Left-Pass	The ego-vehicle successfully passes the object one the left.
Right-Pass	The ego-vehicle successfully passes the object one the right.
Crash	The ego-vehicle and the obstacle collide.

ios can be potentially very dangerous for the vehicles involved, hence, they happen rarely and there is not much available data —real testing would also not be a viable option for our crash scenario. To address this we developed our own data through a simulation environment, developed using Virtual Battlespace 3 (VBS3), with the Logitech G29 Driving Force Racing Wheel and pedals. For realism, our simulated tests were carried out on a realistic model of a Dubai highway. A six lane highway was used, with the obstacle placed in the center lane. A total of 40 participants were used in our trials, all of varying ages, genders and driving experiences. Participants were encouraged to drive naturally and to use their driving experience to avoid the obstacle. Within the simulation environment a Škoda Octavia was used, and a maximum speed of $50km/h$ was adhered to. We label the manoeuvres manually into three groups: pass left, pass right, and crash. The Cartesian centroid positions for the obstacle and ego-vehicle's was recorded, along with their velocity, yaw angle, and Euclidean distance from each other. Data was recorded at 10Hz. This data was used for pair-wise vehicle-obstacle activity classification but also for our trajectory prediction model.

Within our simulation trails, drivers completed a full manoeuvre, until they had passed the obstacle. However, we part of the motivation of this work, our approach is focused around recognising events early within the manoeuvre. Hence, our new dataset for vehicle-obstacle interaction recognition task [3], was be partitioned into three subsets, to demonstrate different aspects of our method.

- Our first subset (SS_1) consists of 122 vehicle-obstacle trajectories of about 600 meters each (43,660 samples). We used this group to train our driver trajectory prediction model.
- Our second group (SS_2) contains complete trajectories. The activity breakdown within this group is as follows: 67 crash, 106 left-pass, and 104 right-pass trajectories. We consider this group to be our ground truth, and used it to evaluate the accuracy of our recognition method and our trajectory

prediction model. The initial distance between the driven vehicle and the obstacle (i.e. the total distance travelled in each trial) was 50 meters. Here two experts labelled each trial as one of the three activities. We summarise the definitions of each scenario in Table 2. Examples of each scenario are shown in Figure 7; where the three scenarios are given from left to right, Left-Pass, Right-Pass and Crash, respectively.

– The third subset (SS_3) consisted of 277 incomplete trajectories. This group is taken as partial trajectories from this trajectories derived in SS_2. We use these partially observed trajectories to evaluate our recognition method, for predicting the events in advance; with the use of our trajectory prediction model. This subset is composed of: 67 crash, 106 left-pass, and 104 right-pass incomplete trajectories, of 25 meters in length each.

When using SS_3 we selected an observed distance of $\varepsilon = 25$ meters between the ego-vehicle and the obstacle. This distance represented 50% of the full distance of the manoeuvre. Figure 7 (d) gives an example of an incomplete trajectory for Right-Pass scenario. Here the solid red line is the ground truth trajectory (taken by the human participant, and is a distance of 25 meters) and the dashed line is the predicted trajectory.

4.2 Validation of Driver Model

To be able to classify a scenario earlier within the manoeuvre, and with a higher degree of accuracy, we utilised our driver model to complete the remainder of a manoeuvre. It is therefore imperative to demonstrate the accuracy of the driver model. Firstly, we trained our driver model (Section 3.3) using the SS_1 dataset, (which consisted of 43,660 data points). We separated SS_1 into training, validation and test sets, in a 70%, 15% and 15% split, respectively. The training subset is used by our FFNN to learn the network parameters (weights and biases), the purpose of the validation subset was to obtain, unbiased, network parameters for the training process. Lastly the test subset was used to evaluate the network performance, to ensure correct functioning. Figure 8 demonstrates that our network parameters were learned correctly.

We used SS_3, to perform a more in-depth analysis of our driver model. This dataset was also not seen during network training. It consisted of 277 incomplete trajectories, where SS_2 (the completed version) is considered as the ground truth. SS_3 consists of 67 crash, 106 left-pass, and 104 right-pass scenarios. Figure 9 shows samples of the input trajectory, predicted trajectory using our approach, and the ground truth trajectory, for our three scenarios. Our trajectories are positioned so that they begin closer to the origin and move from left to right, with increasing 'x' and 'y' values. The trajectories, created by the trail participants, are in green and red (where the red section is the ground truth), and the trajectories from our driver model are in blue. We measure the error from our FFNN based driver model using the Modified Hausdorff Distance (MHD) [8]. The selection of our error metric MHD, was motivated by its property of increasing monotonically as the amount of difference between the two sets of edge

points increases. It is robust to outlier points. Given the driver model generated trajectory as $\mathbf{T_v}$ and the ground-truth trajectory as $\mathbf{T_v}^{gt}$, we determine our error measure as follows:

$$\mathcal{MHD} = min(d(\mathbf{T_v}, \mathbf{T_v}^{gt}), d(\mathbf{T_v}^{gt}, \mathbf{T_v})). \tag{9}$$

Here $d(*)$ is the average minimum Euclidean distances between points of predicted and ground-truth trajectories. The error across SS_3 is shown in figure 6. Here the red line shows the average error and the bottom and top edges of the box give the 25^{th} and 75^{th} percentiles, respectively. The whiskers of each box extend to cover 99.3% of the data. Red pluses represent outliers. Across all three manoeuvres a mean error of 0.4m was seen, this amount of error is negligible for activity recognition, because the overall characteristic of the trajectory is still captured. Moreover, a standard highway lane can be over 3m, therefore variations will still be within lane.

Table 3. Comparison of proposed approach with state-of-the-art algorithms on the traffic dataset [12] .

Type	*TrajNet*	*NWSA* [2]	*Heat-Map* [12]	*WF-SVM* [27]	*LC-SVM* [15]	*GRAD* [13]
Turn	2.9%	2.9%	2.9%	2.0%	16.9%	10.7%
Follow	0.0%	5.7%	11.4 %	22.9%	38.1%	15.4%
Pass	0.0%	0.0%	0.0%	11.7%	17.6%	15.5%
Bothturn	0.0%	2.9%	2.9%	1.2%	2.9%	4.2%
Overtake	2.9%	5.7%	5.7%	47.1%	61.7%	36.6%
Average Error	**1.16%**	**3.44%**	**4.58%**	**16.98%**	**27.24%**	**16.48**

4.3 Results on Vehicle Activity Recognition

The main objective of this work is to be able to perform supervised classification of pair-wise vehicle activity, between the ego-vehicle and its surroundings. Having prior information about the type of event a vehicle is in, or about to enter, can have benefits for path planning and decision making. As part of this work we have collected and labelled a comprehensive dataset of high speed vehicle object interactions, in simulation. With this new dataset of complex scenarios unique classification problems arise. Here we will examine the performance and accuracy of our algorithm and provide insight into its results.

Results on Traffic Mostion Dataset: We first evaluate the performance of our algorithm on the traffic dataset [12]. We utilised the Cartesian coordinate

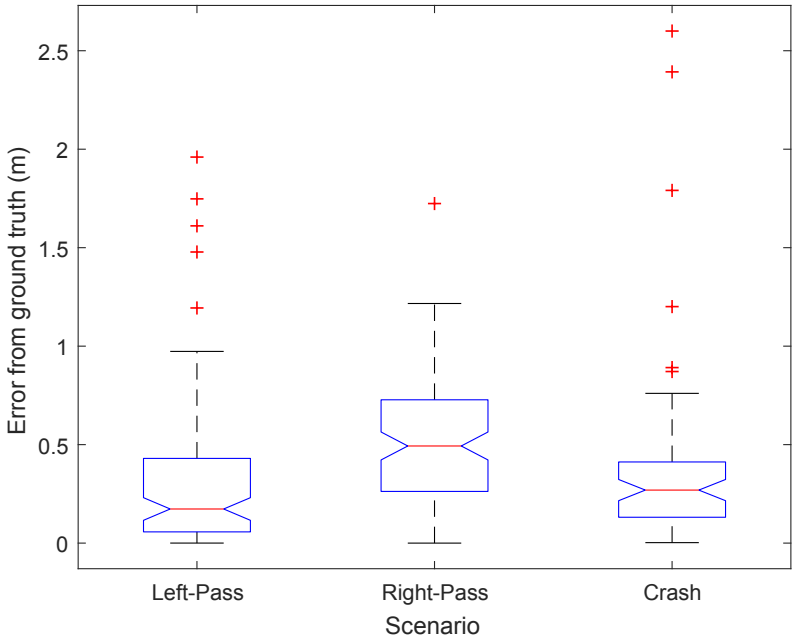
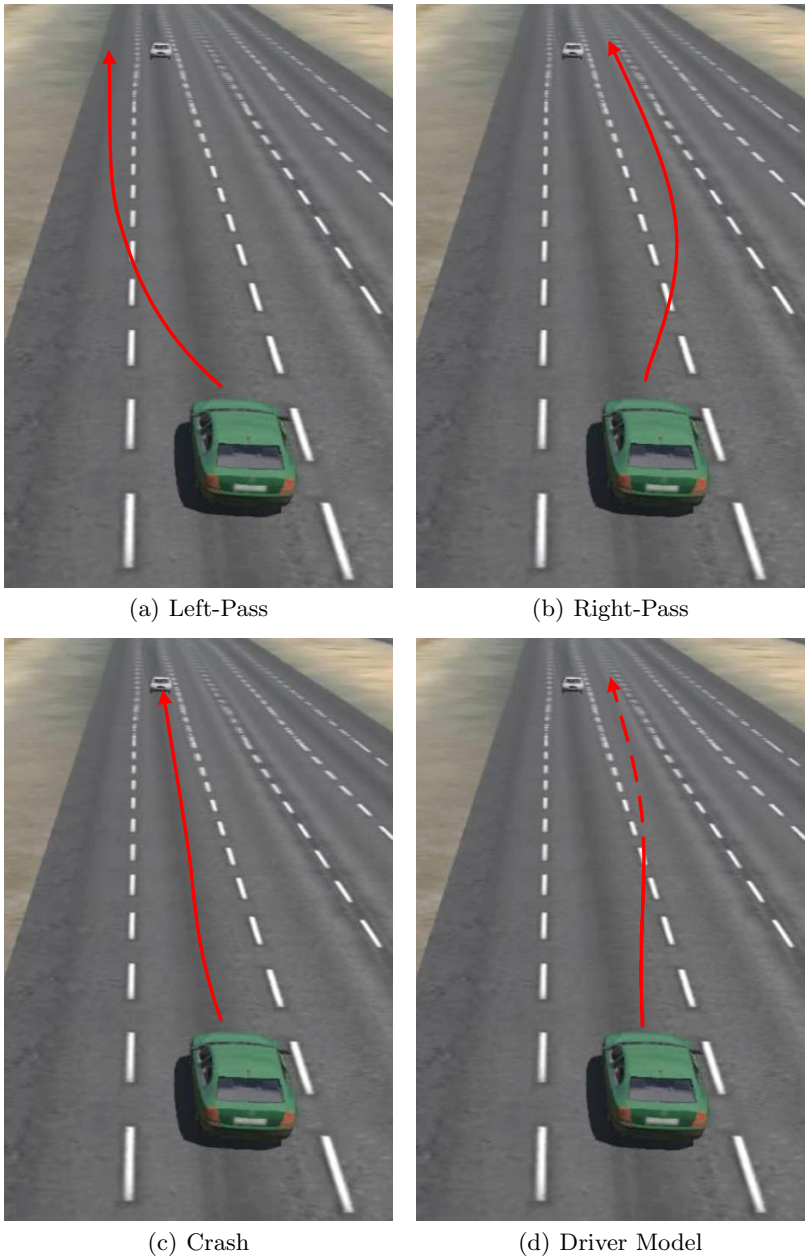


Fig. 6. Errors ($\mathcal{MHD}$) from scenarios, across $SS3$ [4].

pairs, given for each vehicle, as inputs to our algorithm. For each clip the coordinates were then processed into their corresponding QTC_C trajectories. The final pre-processing stage involved constructing our image texture (I_i) for corresponding QTC_C trajectories (done with Algorithm 1). Figure 5 gives example images texture for five different interactions along with the samples in figure 4. In order thoroughly evaluate our approach we perform 5-fold cross validation. For each fold we separate the images texture (I) into training and testing sets, with a 80% to 20% split for each class, respectively. The training sets were used to determine the weights of our network ($TrajNet$). The unseen test images texture were then classified by our trained $TrajNet$. Results of this are shown in Table 3. We also incorporate comparative results obtained from [2], [12], and three other approaches [27, 15, 13]. The average error (AVG Error) is calculated as the ratio between the total number of incorrect classifications (compared with the ground truth labels) and the total number of activity sequences within the test set. We show that our approach gives better performance when compared with the five state-of-the-art approaches (Table 3). It is able to classify the dataset with errors rate 1.16% and with a standard deviation 0.015.

Results on Vehicle Obstacle Dataset: Our second set of experiments for activity recognition algorithm was done on our simulation based dataset [3]. We test our classification approach using SS_2 for complete trajectories and SS_3 for partial trajectories. The pair-wise centroid positions of each vehicle x, y in



(a) Left-Pass (b) Right-Pass

(c) Crash (d) Driver Model

Fig. 7. Representation of manoeuvres (shown through VBS3) displayed in red (a-c). Here the green car represents an ego-vehicle and the white car a possible obstacle. Results of applying our driver model are shown in (d), where the solid red line is the previously driven path and the dashed line is the predicted trajectory [4].

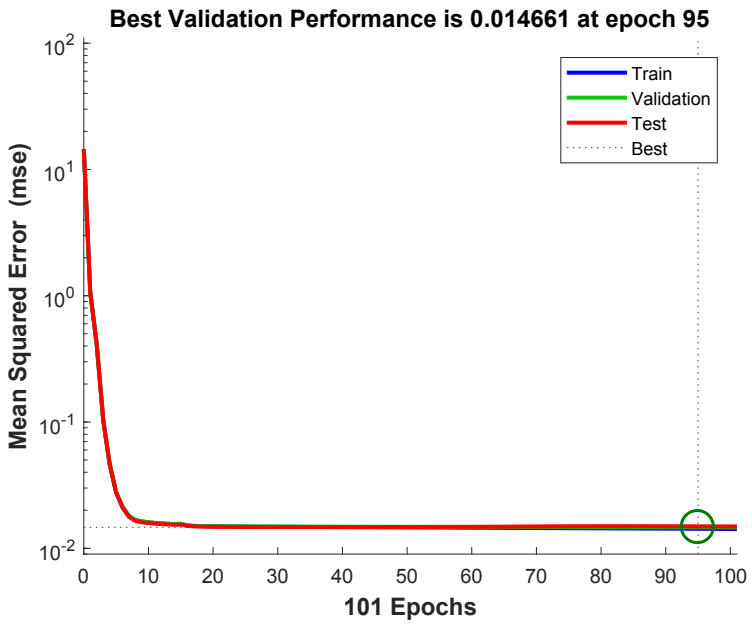
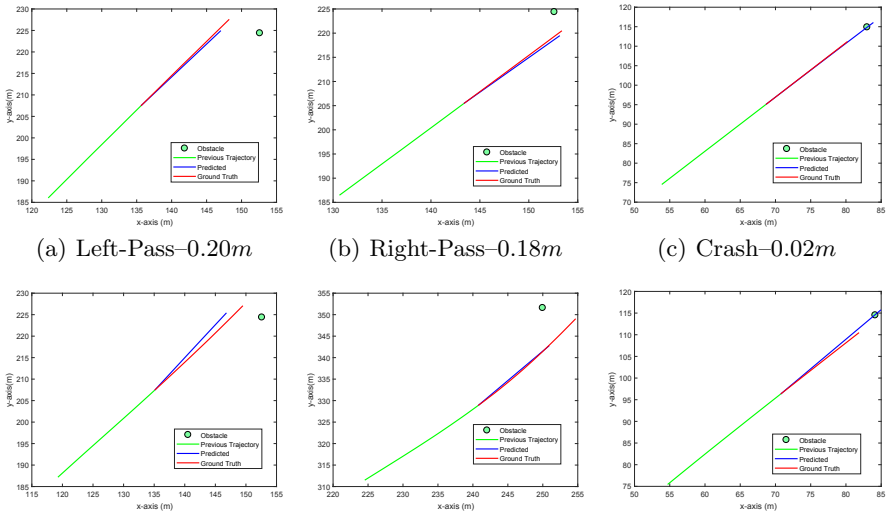


Fig. 8. Results of training for FFNN based driver model. Note that the error rates for the training and testing set are similar [4].

SS_2 were given as inputs. This was preprocessed and encoded to their QTC_C trajectory representation for each scenario. The next step in preprocessing was to transform the trajectory representations to image texture (I_i). One image was created for each QTC_C trajectory, based on Algorithm 1. To evaluate our classification approach we used 5-fold cross validation on the SS_2 dataset. For each fold we separated the images texture, generated from SS_2, into two sets, representing 80% to 20% of the dataset, for training and testing respectively. The network weights were trained using the training subset $TrajNet$, and then tested on the test subset. This approach showed the robustness of our technique, and helps to reduce any bias the network may have received from a training set. Results of the complete manoeuvre classification are given in Table 4. The network trained on full trajectory manoeuvres is represented as "*Complete Traj*". Here the error ratio (AVG Error) is defined as the total number of false classifications, as obtained from the ground truth labelling, divided by the total number of sequences in the test subset.

While we have demonstrated the efficacy of our classification method on full manoeuvres, we will now evaluate results on partial trajectories; in order to make pre-emptive decisions. The first step for evaluating the incomplete trajectories was to train our FFNN based driver prediction model(figure 3), using SS_1. Using our fully trained driver model, we now perform a prediction step (on the SS_3 dataset) to generate future trajectories and a complete manoeuvre. We refer to this augmented dataset as SS_4.



(a) Left-Pass–0.20m (b) Right-Pass–0.18m (c) Crash–0.02m

(d) Left-Pass–0.69m (e) Right-Pass–0.14m (f) Crash–0.29m

Fig. 9. Example trajectories for simulation dataset [3]. Error rates are based on, $\mathcal{MHD}$. Two examples from each scenario are given (top and bottom rows), covering all three scenarios. (a)-(c) and (d)-(f) represent left-pass, right-pass, and, crash scenarios, respectively [4].

Table 4. Classification error for full and partially observed trajectories, SS_2 and SS_3, respectively [4].

Type	**Complete Traj** (SS_2)	**Predicted Traj** (SS_4)
Crash	0.0%	0.0%
Left-Pass	0.0%	0.0%
Right-Pass	0.0%	1.0%
AVG Error	**0.0%**	**0.3%**

Similarly to evaluating the full trajectories, we utilise the x, y centroid pairs in the Cartesian space, for both the vehicle and the obstacle from SS_4, as inputs to our classifier. We then generate QTC_C trajectories and associated images texture (I) for all runs. A sample trajectory is shown in Figure 7 (d). We calculated the classification accuracy of our algorithm on SS_4, using a 5-fold cross validation. For each fold, the images texture in SS_4 were separated into training and testing subsets, 80% to 20% respectively. The network weights were optimised for $TrajNet$ using the training subset and test subset was classified by the optimised network $TrajNet$. Results of our testing on incomplete trajectories are presented in Table 4. We differentiate classifications from our network trained on incomplete trajectories by labelling it "*Predicted Traj*". It can be seen that our approach still delivers a high level of performance, despite not being given

the complete trajectory. This can be seen a strong motivator for coupling this approach with potential decision making and path planning algorithms.

4.4 Results on Datasets From Different Sources

To demonstrate the robustness and generality of our method, we conducted similar classification experiment using a challenging dataset which combines the two datasets ([3] and [12]). This results in a new dataset which contains 452 scenarios for eight different activities, *Turn*, *Follow*, *Pass*, *BothTurn*, *Overtake*, *Crash*, *Left-Pass* and *Right-Pass*. This is to show that our single network is able to accurately distinguish between an even larger number of driving tasks. Again, we calculated the classification accuracy of our algorithm using a 5-fold cross validation with training and testing subsets, 80% to 20% respectively. The network weights were optimised for $TrajNet$ using the training subset and test subset was classified by our generic $TrajNet$.

Table 5. Five-fold cross validation results on datasets with eight different activities from different sources.

Fold	Classified	Correct	Error Rate	Train Time
Fold 1	90	89	0.01	9 min 50 sec
Fold 2	90	89	0.01	9 min 58 sec
Fold 3	90	90	0.0	10 min 13 sec
Fold 4	91	90	0.01	9 min 36 sec
Fold 5	91	89	0.02	9 min 23 sec

The results on this dataset show that our method is able to classify the data coming from different sources with average errors rate 0.01. Table 5 shows five-fold cross-validation results for the classification challenge training data of combined datasets with the training time of each fold. The computational efficiency of the algorithm, which includes trajectory predication and activity classification for one scenario was on the order of 28 milliseconds.

5 CONCLUSION

A new approach based on a deep neural networks for vehicles activity recognition was presented. We have built upon our pervious approach by extending the number of trajectories classes our network is able to classify. This further demonstrates the capabilities of our approach to interpret multiple types of vehicle interactions.We utilise a QTC representation to capture invariant interaction features. We then detailed steps on constructing a corresponding image textures, for a more interpretable representation of the QTC trajectories; based on a one-hot vector encoding. Our technique is able to efficiently capture multiple

scenarios of vehicles interactions. We describe the reasoning behind our architecture and how we efficiently used a limited amount of data to train TrajNet, while not compromising on a high level of classification accuracy.

The method has been tested on two different challenging datasets: traffic motion and vehicles interaction datasets. The experimental results showed that the method is robust and performs better than existing methods with 1.16% error rate, as opposed to 3.44%, 4.58%, 16.98%, 27.24%, and 16.48% of [2], [12], [27], [15] and [13], respectively.

A separate contribution is our vehicle-obstacle interaction dataset (VOIDataset), which includes complete and incomplete trajectories. VOIDataset is publicly available for download. Our method obtained high accuracy with error rates 0.0% and 0.3%, for complete and incomplete scenarios in VOIDataset, respectively. Through combining the two datasets we showed the robustness of our activity classification method.

To be able to pre-empt a future scenario from a partial observed example we introduced a FFNN method for trajectory predication. The trajectory prediction model was also evaluated on VOIDataset and the average error was 0.4m.

This approach can have potential impacts in future driving technologies, whether in safety related applications or improving path planning. As a step towards future work we would aim to tackle more complex manoeuvres, involving multiple vehicles and infrastructure.

References

1. Ahmed, S.A., Dogra, D.P., Kar, S., Roy, P.P.: Trajectory-based surveillance analysis: A survey. IEEE Transactions on Circuits and Systems for Video Technology (2018)
2. AlZoubi, A., Al-Diri, B., Pike, T., Kleinhappel, T., Dickinson, P.: Pair-activity analysis from video using qualitative trajectory calculus. IEEE Transactions on Circuits and Systems for Video Technology (2017)
3. AlZoubi, A., Nam, D.: Vehicle Obstacle Interaction Dataset (VOIDataset). `https://figshare.com/articles/Vehicle_Obstacle_Interaction_Dataset_VOIDataset_/6270233` (2018)
4. AlZoubi., A., Nam., D.: Vehicle activity recognition using mapped qtc trajectories. In: Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,. pp. 27–38. INSTICC, SciTePress (2019). https://doi.org/10.5220/0007307600270038
5. Chavoshi, S.H., De Baets, B., Neutens, T., Delafontaine, M., De Tré, G., de Weghe, N.V.: Movement pattern analysis based on sequence signatures. ISPRS International Journal of Geo-Information **4**(3), 1605–1626 (2015)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. pp. 248–255. IEEE (2009)
7. Dodge, S., Laube, P., Weibel, R.: Movement similarity assessment using symbolic representation of trajectories. International Journal of Geographical Information Science **26**(9), 1563–1588 (2012)

8. Dubuisson, M.P., Jain, A.K.: A modified hausdorff distance for object matching. In: Proceedings of 12th international conference on pattern recognition. pp. 566–568. IEEE (1994)
9. Hanheide, M., Peters, A., Bellotto, N.: Analysis of human-robot spatial behaviour applying a qualitative trajectory calculus. In: RO-MAN, 2012 IEEE. pp. 689–694. IEEE (2012)
10. Khosroshahi, A., Ohn-Bar, E., Trivedi, M.M.: Surround vehicles trajectory analysis with recurrent neural networks. In: Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on. pp. 2267–2272. IEEE (2016)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
12. Lin, W., Chu, H., Wu, J., Sheng, B., Chen, Z.: A heat-map-based algorithm for recognizing group activities in videos. IEEE Transactions on Circuits and Systems for Video Technology **23**(11), 1980–1992 (2013)
13. Lin, W., Sun, M.T., Poovendran, R., Zhang, Z.: Group event detection with a varying number of group members for video surveillance. IEEE Transactions on Circuits and Systems for Video Technology **20**(8), 1057–1067 (2010)
14. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F.E.: A survey of deep neural network architectures and their applications. Neurocomputing **234**, 11–26 (2017)
15. Ni, B., Yan, S., Kassim, A.: Recognizing human group activities with localized causalities. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. pp. 1470–1477. IEEE (2009)
16. Ohn-Bar, E., Trivedi, M.M.: Looking at humans in the age of self-driving and highly automated vehicles. IEEE Transactions on Intelligent Vehicles **1**(1), 90–104 (2016)
17. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. pp. 1717–1724. IEEE (2014)
18. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3), 211–252 (2015)
19. Shi, Y., Tian, Y., Wang, Y., Huang, T.: Sequential deep trajectory descriptor for action recognition with three-stream cnn. IEEE Transactions on Multimedia **19**(7), 1510–1520 (2017)
20. Shi, Y., Zeng, W., Huang, T., Wang, Y.: Learning deep trajectory descriptor for action recognition in videos using deep neural networks. In: Multimedia and Expo (ICME), 2015 IEEE International Conference on. pp. 1–6. IEEE (2015)
21. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
22. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
23. Van de Weghe, N.: Representing and reasoning about moving objects: A qualitative approach. Ph.D. thesis, Ghent University (2004)
24. Xiong, X., Chen, L., Liang, J.: A new framework of vehicle collision prediction by combining svm and hmm. IEEE Transactions on Intelligent Transportation Systems **19**(3), 699–710 (March 2018). https://doi.org/10.1109/TITS.2017.2699191

25. Xu, D., He, X., Zhao, H., Cui, J., Zha, H., Guillemard, F., Geronimi, S., Aioun, F.: Ego-centric traffic behavior understanding through multi-level vehicle trajectory analysis. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on. pp. 211–218. IEEE (2017)
26. Xu, H., Zhou, Y., Lin, W., Zha, H.: Unsupervised trajectory clustering via adaptive multi-kernel-based shrinkage. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4328–4336 (2015)
27. Zhou, Y., Yan, S., Huang, T.S.: Pair-activity classification by bi-trajectories analysis. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. pp. 1–8. IEEE (2008)