Vehicle Activity Recognition Using Mapped QTC Trajectories

Keywords: Vehicle Activity Recognition, Qualitative Trajectory Calculus, Trajectory Texture, Transfer Learning, Deep Convolutional Neural Networks.

Abstract: The automated analysis of interacting objects or vehicles has many uses, including autonomous driving and security surveillance. In this paper we present a novel method for vehicle activity recognition using Deep Convolutional Neural Network (DCNN). We use Qualitative Trajectory Calculus (QTC) to represent the relative motion between pair of vehicles, and encode their interactions as a trajectory of QTC states. We then use one-hot vectors to map the trajectory into 2D matrix which conserves the essential position information of each QTC state in the sequence. Specifically, we project QTC sequences into a two dimensional image texture, and subsequently our method adapt layers trained on the ImageNet dataset and transfer this knowledge to the activity recognition task. We have evaluated our method using two different datasets, and shown that it out-performs state-of-the-art methods, achieving an error rate of no more than 1.16%. Our motivation originates from an interest in automated analysis of vehicle movement for the collision avoidance application, and we present a dataset of vehicle-obstacle interaction, collected from simulator-based experiments.

1 INTRODUCTION

The vehicle activity recognition task aims to identify the actions of one or more vehicles from a sequence of observations. The pair-wise interaction between vehicle-vehicle or vehicle-obstacle is the building block of large group interactions. In dynamic traffic a vehicle will share the same roads with different objects (e.g. vehicles or obstacles), on which unexpected interactions can occur anytime and anywhere. Quite often vehicular collisions are a result of a lack of situational awareness, and not being able to identify situations before they become dangerous. Therefore, understanding the relative interaction between the vehicle and it is surrounding is crucial for recognizing the behavior that the vehicle is in (or about to enter) and to avoid any potential collisions (Ohn-Bar and Trivedi, 2016).

Previous research concerned with vehicle activity analysis has been mainly focused on quantitative methods which use sequences of real-valued features (trajectories) (Xu et al., 2017; Khosroshahi et al., 2016; Lin et al., 2013; Lin et al., 2010; Ni et al., 2009; Zhou et al., 2008). However, increasing attention has been given to the use of qualitative methods, which use symbolic rather than real-value features, with applications such as, vehicle interaction and human behavior analysis (AlZoubi et al., 2017), and human-robot interaction (Hanheide et al., 2012). Qualitative methods have shown better performance for vehicles activity analysis (AlZoubi et al., 2017). There are several motivations for using qualitative methods such as: qualitative representations are typically more compact and computationally efficient than quantitative methods, and humans naturally communicate and reason in qualitative ways rather than by using quantitative measurements, particularly when describing interactions and behaviors (Dodge et al., 2012).

In the context of activity recognition a few previous studies made their efforts on encoding the trajectory of moving object in a compact and powerful representation (e.g. two dimensional matrix (Lin et al., 2013) or trajectory texture image (Shi et al., 2015)). These features were used to train the classifier for the activity recognition task. These methods have shown to be successful in different application domains varies between human activity recognition (Shi et al., 2015; Shi et al., 2017) and pair-wise vehicles activity recognition (Lin et al., 2013). Moreover, representing trajectories in two dimensional matrix can be advantageous when paired with methods of image recognition (e.g. deep learning). Recently, increasing attention has been given to the use of methods based on deep features for object classification from image. These methods have shown its strong power in features (e.g. shape and texture of objects) learning.

In this paper, we present our method for vehicle pair-activity classification (recognition), based on QTC and DCNN. First, we use QTC to represent the relative motion between pairs of objects (vehicle-vehicle or vehicle-obstacle), and encode their interactions as a trajectory of QTC states (or characters). Then, we use one-hot vectors to represent QTC sequences as a two dimensional matrix (or im-

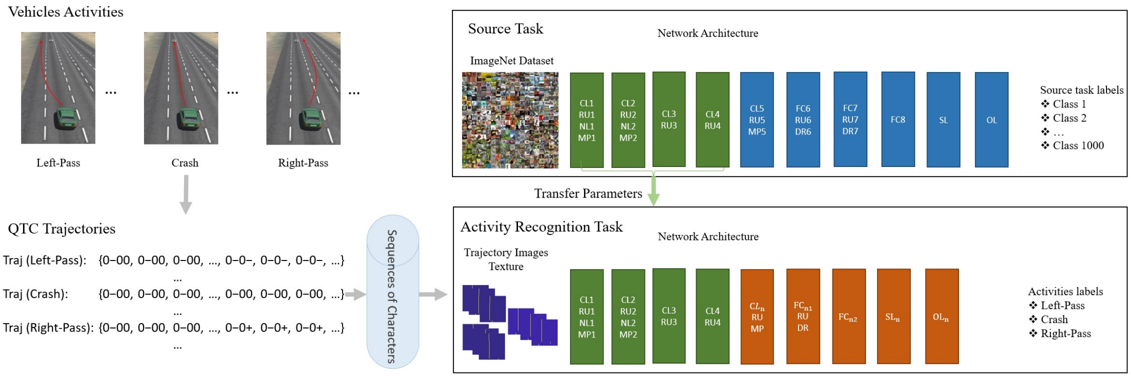


Figure 1: Our Proposed Method.

age texture), and subsequently our method adapt layers from an already trained DCNN on the ImageNet dataset and transfer this knowledge to the vehicle pair-activity recognition task. Where each activity will have a unique image signature (or texture). In addition, we have developed a dataset of vehicle-obstacle interactions, which we use in our evaluations. We accordingly present a detailed comparative work against state-of-the-art quantitative and qualitative methods, using different datasets (including our own). Furthermore, our results show that our proposed method outperforms current methods for pair-wise vehicles trajectory classification in different challenging applications. Figure 1 shows an overview of the main components of our method. Our novel approach for recognizing pair-wise vehicle activity uses, for the first time, QTC with DCNN.

Our work is primarily motivated by our interest in the automated recognition of vehicles activities. However, predicting a complete trajectory that the vehicle is in (or about to enter) helps avoiding any potential collisions. In order to gain traction as a mainstream analysis technique, we present a novel driver model for predicting a vehicle's future trajectory from partially observed one. The key contributions of our work are as follows: (1) we propose a new CNN suited for vehicle pair-activity classification, based on a modified version of AlexNet (Krizhevsky et al., 2012). (2) a novel method for pair-wise vehicle activity recognition is presented, where we integrate a driver model to estimate likely trajectories, and predict a vehicle's future activity. (3) we show experimentally that our proposed CNN out-performs existing state-of-the-art methods (such as (AlZoubi et al., 2017; Lin et al., 2013; Lin et al., 2010; Ni et al., 2009; Zhou et al., 2008)). We also introduce our own, new, dataset of vehicle-obstacle activities, which is ground-truthed and consists of 554 scenarios of vehicles (complete and incomplete scenarios) for three different types of interactions. The dataset is publicly available for other researchers studying vehicle activity, and pair-activity analysis in general.

2 Related Work

Qualitative Trajectory Calculus (QTC): Qualitative spatial-temporal reasoning is an approach for dealing with knowledge on which human perception of relative interactions is based, using symbolic representations of relevant information rather than real-valued measurements (Van de Weghe, 2004). Activity analysis methods based QTC representation have shown to be used successfully and outperform quantitative methods in different application domains including human activity analysis and vehicles interaction recognition (AlZoubi et al., 2017). Given the positions of two moving objects (k and l), the QTC represents the relative motion in four features:

- Distance Feature:
  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.
- Speed Feature:
  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).
- Side Feature:
  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.
- Angle Feature:
  S_6: The respective angles between the velocity vectors of the objects and vector L: "-" if $\theta_1 < \theta_2$,

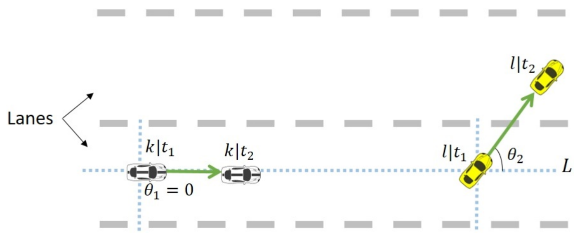


Figure 2: Example of QTC relations between two moving vehicles: $QTC_C = (-,+,0,+)$.

“+” if $\theta_1 > \theta_2$ and “0” if $\theta_1 = \theta_2$

where S_i represents the qualitative relations in QTC. figure 2 shows the concept of qualitative relations in QTC for two disjoint objects (vehicles); k and l. Three main calculi were defined (Van de Weghe, 2004): 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.

Vehicle Activity Analysis: In regards to activity analysis many previous works made their efforts on studying trajectory-based activity analysis, and a review is provided by (Ahmed et al., 2018). Generally speaking, activity analysis approaches can be divided into three categories: single-role activities (Xu et al., 2015); pair-activities (AlZoubi et al., 2017); and group-activities (Lin et al., 2013). Most of these approaches were motivated by either an interest in human behavior recognition (AlZoubi et al., 2017; Lin et al., 2013; Zhou et al., 2008), human-robot interaction (Hanheide et al., 2012), animal behavior clustering (AlZoubi et al., 2017), or vehicles interaction recognition (AlZoubi et al., 2017; Xu et al., 2017; Khosroshahi et al., 2016; Lin et al., 2013). Pair-activities approaches are most closely related to our work in this paper.

Previous works studying vehicles activity analysis for understanding traffic behaviors (Lin et al., 2013) and autonomous driving applications (Xiong et al., 2018) mainly used quantitative features. (Lin et al., 2013) presented a heatmap-based algorithm for vehicle pair activity recognition. The method represents vehicle trajectory as an activity map (or 2D matrix) and uses a Surface-Fitting (SF) method to classify the vehicles activities. More recently, methods based on qualitative features were proposed for traffic activity recognition (AlZoubi et al., 2017). A Normalized Weighted Sequence Alignment (NWSA) method was developed to calculate the similarity between QTC sequences. The method was evaluated on three different datasets (vehicles interactions, human activities and animal behaviors), and shown that it is out-performs state-of-the-art quantitative methods ((Lin et al., 2013; Lin et al., 2010; Ni et al., 2009; Zhou et al., 2008)). Those methods are most closely related to our own work. We therefore use these algorithms, vehicles-interaction dataset (Lin et al., 2013), and ground truth, as a benchmark for evaluating our own recognition method (Section 4.1).

The spatial-temporal representation of motion information is crucial to activity recognition. Recently, different techniques have been presented to encode the sequence of real-valued features in a compact way. Shi et al. (Shi et al., 2017) proposed a long-term motion descriptor called sequential deep trajectory descriptor (sDTD). The method first extracts the simplified dense trajectories of single object and then converts these trajectories into 2D images. Chavoshi et al. (Chavoshi et al., 2015) proposed a visualization technique, sequence signature (SESI), to transform the simplest variant of QTC (QTC_B) movement patterns of moving point objects (MPOs) into a 2D indexed rasterized space. The method in (Lin et al., 2013) represents trajectories of pair-vehicles as a series of heat sources; then, a thermal diffusion process creates an activity map (or 2D matrix). These methods either encode trajectories of a single object, or rely only on traditional similarity methods (e.g. Euclidean distance) which can not optimally deal with varying lengths trajectories and compound behaviors. The qualitative (AlZoubi et al., 2017) and the quantitative (Lin et al., 2013) methods have been shown experimentally as the most effective methods for representing vehicle pair-wise activity. We thus adopt it as a benchmark methods, against which we evaluate our own work.

Deep Learning Models: Recently, convolutional neural networks have shown outstanding object recognition performance especially for the large-scale visual recognition tasks (Russakovsky et al., 2015). It has shown a strong power in feature learning and the ability to learn discriminative and robust object features (e.g. shapes and textures) from images (Oquab et al., 2014). CNN models for the object classification problem have been developed such as AlexNet (Krizhevsky et al., 2012) and GoogLeNet (Szegedy et al., 2015), which were designed in the context of the “Large Scale Visual Recognition Challenge” (ILSVRC) (Russakovsky et al., 2015) for the ImageNet dataset (Deng et al., 2009). A review of deep neural network architectures and their applications for object recognition is provided by (Liu et al., 2017). Here we give an overview of AlexNet, which we adapt for our vehicle interaction recognition task.

The AlextNet model is a DCNN trained on approximately 1.2 million labeled images, and it contains 1,000 different categories from the ILSVRC dataset (Russakovsky et al., 2015). Where each image in this dataset contains single object located in the centre, occupies significant portion of the image, and limited background clutter. AlextNet model takes the entire image as an input and predict the object class label. The architecture of this network comprises about 650,000 neurons and 60 million parameters. It includes five convolutional layers (CL), two normalisation layers (NL), three max-pooling layers (MP), three fully-connected layers (FL), and a linear layer with softmax activation (SL) in the output layer (OL). The dropout regularization (DR) method (Srivastava et al., 2014) is used to reduce overfitting in the fully connected layers and Rectified Linear Units (ReLU) is applied for the activation of the layers. We have tuned and evaluated the performance of this powerful architecture of DCNN for the vehicle interactions recognition task.

3 Proposed Method

Our proposed method comprises of four main components:

- We represent vehicles' relative movements using sequences of QTC states.
- A method to represent QTC sequences into a 2D matrix (image texture) using one-hot vector representation is introduced.
- We propose a novel and updated CNN model, utilizing AlexNet and the ImageNet dataset, for vehicles pair-activity recognition task. This results in our new network *TrajNet*.
- For predicting a complete trajectory from partially observed one, a human driver model is proposed.

Figure 1 shows the main components of our recognition method. Our method extracts QTC trajectories from multiple consecutive observations (x,y positions of moving vehicles) and then project them onto 2D matrices. This results in a "trajectory texture" image which can effectively characterize the relative motion between pairs of vehicles during a time interval $[t_1, t_N]$. Along with the updated structure of *TrajNet*, we perform transfer learning using a set of "trajectory texture" images, for different vehicles activities. As we will show in the experimental results, our method generalizes across different data contexts, complete and predicted trajectories, and enables us to consistently out-perform state-of-the-art methods. We also present our model for predicting vehicles trajectories from incomplete ones.

3.1 Vehicle Activity Representation Using QTC

We extract QTC features from x, y data points (2D positions) to represent the relative motion between two vehicles, and encode their interactions as a trajectory of QTC states. For our experiments we have used the common QTC variant (QTC_C) which provides a qualitative representation of the two-dimensional movement of a pair of vehicles.

Definition: Given two interacting vehicles (or vehicle-obstacle) with their x, y coordinates (centroid positions), we define:
$V1_i = \{(x_1, y_1), ..., (x_t, y_t), ..., (x_N, y_N)\}$,
$V2_i = \{(x'_1, y'_1), ..., (x'_t, y'_t), ..., (x'_N, y'_N)\}$,
where (x_t, y_t) is the centroid of the first interacting vehicle at time t and (x'_t, y'_t) is the centroid of the second. The pair-wise trajectory is then defined as a sequence of corresponding 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; and N is the number of observations in Tv_i.

3.2 Mapping QTC Trajectory into Image Texture

The QTC_C trajectory generated in Section 3.1 is a one-dimensional sequence of successive QTC_C states. Comparing to the text data, there is no space between QTC_C states and there is no term of word. Therefore, we translate QTC_C trajectories into sequences of characters in order to apply the same representation technique for text data without losing position information of each QTC_C state in the sequence. Then, we represent this sequence into numerical values so that it could be used as an input for our CNN presented in Section 3.3.

Our method first represents the QTC_C states using the characters Cr: cr_1, cr_2, ..., cr_{81}. Then, it maps the one-dimensional sequence of characters (or QTC_C trajectory) into a two-dimensional matrix (image texture) 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 ($TrajNet$) to learn different vehicles activities.

Definition: Given a set of trajectories $\zeta = \{Tv_1, ..., Tv_i, ...Tv_n\}$ where n is the number of trajectories in ζ. We convert each QTC trajectories in ζ into sequences of characters $\zeta_C = \{Cv_1, ..., Cv_i, ...Cv_n\}$. Then, we represent each sequence Cv_i in ζ_C into an image texture I_i using one-hot vector representation, where the columns represent Cr (or QTC_C states)

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 $\zeta$
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 $\zeta$
Define: a 2D matrix ($I_i$) with size ($N \times 81$) for each sequence in $\zeta_C$, 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 $I$
```

and the rows indicate the present of the character (or QTC_C state) at the given time-stamp. Algorithm 1 describes representing QTC_C trajectories into image texture. For example, in the movement between two vehicles in figure 6(a): one vehicle is passing by the other vehicle (or obstacle) into 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 in an image texture I_i using our Algorithm 1.

3.3 Vehicle Activity Recognition

Our trajectory representation (section 3.2) results in two-dimensional numerical matrices (or images). Where each activity is represented in a unique image texture which can be used as an input to train our proposed CNN. Our vehicle trajectory images representation can be challenging, where trajectory encoding have variations based on the differences in vehicles' behaviors —although the overall activity is the same. This variation has motivated us to use a CNN based approach for our classification problem.

Gaining inspiration from AlexNet we aim to take advantage of its structure; which is robust in classifying many images with complex structures and features. However, directly learning the parameters of this network using relatively small images texture of QTC trajectories dataset is not effective. We adapt the architecture of the CNN model (AlexNet) by replacing and fine-tuning the last convolutional layer ($CL5$), the last three fully connected layers ($FL6$, $FL7$ and $FL8$), softmax (SL) and the output layer (OL) which result in our new network *TrajNet* (figure 1). We replace the last $CL5$ layer with a smaller layer CL_n where we now use 81 convolutional kernels. This was followed by ReLU and max pooling layers (same parameters as in (Krizhevsky et al., 2012)). We then include one FL_{n1}, with 81 nodes. This replaces the last two fully connected layers ($FL6$ and $FL7$), of 4096 nodes each. The reduced number of nodes is correlated to the reduction in higher level features in our trajectory texture images (as opposed to (Deng et al., 2009)), enabling more tightly coupled responses. After our new FL_{n1} we include a ReLU and dropout layer (50%). According to the number of vehicle activities (a) defined in the dataset, we add a final new fully connected layer (FL_{n2}) for a classes, a softmax layer (SL_n), and a classification output layer (OL_n). Where the output of the last fully-connected layer is fed to an a-way softmax (or normalized exponential function) which produces a distribution over the a class labels. Then, we develop training and testing procedures based on our images texture I. Each image texture (I_i) is used as input for the data layer $data$ for the network. Finally, we set the network parameters as follow: the iteration number set to 10^4; the initial learn rate 10^{-4}; and mini batch size 4. These configurations are to ensure that the parameters are fine tuned for the activity recognition task. The other network parameters are set as default values (Krizhevsky et al., 2012).

3.4 Trajectory Prediction Model

Trajectory prediction is an important precursor to capture the full vehicle scenario and as a pre-processing step for vehicle activity classification. To be able to do this human driver models are typically used to predict vehicles trajectories given a specific situation.

We propose a Feed Forward Neural Network (FFNN) as a driver model to predict complete vehicles trajectories using partially observed (or incomplete trajectories). Figure 3 shows our FFNN architecture. It consists of 9 hidden layers, and each hidden layer has z nodes, were $\{z \in \mathbf{Z} : \mathbf{Z} = [10, 10, 20, 20, 50, 20, 20, 20, 15]\}$. Given a hidden layer H^i with z nodes, $z \widehat{=} i^{th}$ element in $\mathbf{Z}$. This configuration was set empirically, and was well suited to capture the complex driving behaviors a human can make. Our FFNN passes information in one direction, first through the input layer, then modified in hidden

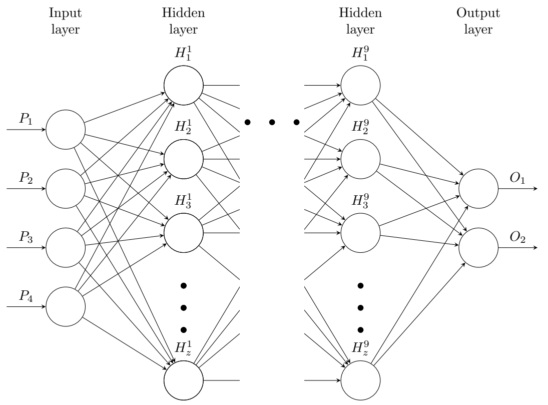


Figure 3: Driver model neural network architecture, showing layers and associated nodes. With inputs, outputs and hidden states, P, O and H, respectively. Note that we use 9 hidden layers, each with z hidden states.

layers and finally passed to the output layers. Layers are comprised of nodes, these nodes have weights associated with each input to the node. To generate an output for a node the sum of the weighted inputs and a bias value are calculated and then passed through an activation function, here we use a hyperbolic tangent function. It can be seen as a way to approximate a function, where the values of the node weights and biases are learned through training. Here training data includes the inputs with their correct or desired outputs/targets. 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. Relative changes in heading angle and translation of the ego-vehicle between times $(t-1)$ and t are calculated as follow:

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

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

Where $\theta(t)$ is the heading angle and $\nabla(t)$ is the magnitude of the change in the ego-vehicle's position. Both features ($\theta(t)$ and $\nabla(t)$) were used as primary data to train our FFNN. To remove noise, from small changes in driver heading angles, we average the heading angle and translation over the previous $0.5s$.

From this we can define the inputs to our FFNN as:

$$\Theta(t) = {}^1\!/_5 \sum_{i=(t-5)}^{t} \theta(i), \tag{3}$$

$$\mathcal{R}(t) = {}^1\!/_5 \sum_{i=(t-5)}^{t} \nabla(i), \tag{4}$$

$$\beta(t) = \sum_{j=1}^{t} \Big(\nabla(j) \sin\big(\sum_{i=1}^{j} \theta(i)\big)\Big), \tag{5}$$

$$\lambda(t) = \sqrt{(x_t - x_t')^2 + (y_t - y_t')^2}. \tag{6}$$

Where $\lambda(t)$ is the distance between the ego-vehicle and the object and $\beta(t)$ represent the lateral shift from the centre of the road. This was included to bound the driver model from going off the road when avoiding the obstacle. Θ and $\mathcal{R}$ are directly related the vehicle movement, β relates the vehicle position to the road, and λ relates the vehicle to the obstacle. These inputs are shown 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 $[O_1, O_2] = [\theta(t+1), \nabla(t+1)]$. The predicted trajectory is determined iteratively, at each time-step; this is described in Algorithm 2.

Algorithm 2 Vehicle Trajectory Prediction

```
▷ A distance of $\varepsilon$ 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 (3)-(6), respectively;
        Outputs: $[\theta(t+1), \nabla(t+1)]$;
        Update Trajectory:
            $\begin{cases} x_{t+1} = \nabla(t+1)\cos(\sum_{i=1}^{t+1}\theta(i)) + x_t, \\ y_{t+1} = \nabla(t+1)\sin(\sum_{i=1}^{t+1}\theta(i)) + y_t. \end{cases}$
        Increment time-step: $t = t + 1$;
    end while
end if
```

4 Experiments

We have performed comparative experiments in order to evaluate the effectiveness of our method, using two publicly available datasets. These datasets represent different application domains, namely, vehicle traffic movement from surveillance cameras (Lin et al., 2013) and vehicle-obstacle interaction for collision avoidance application (VOI, 2018).

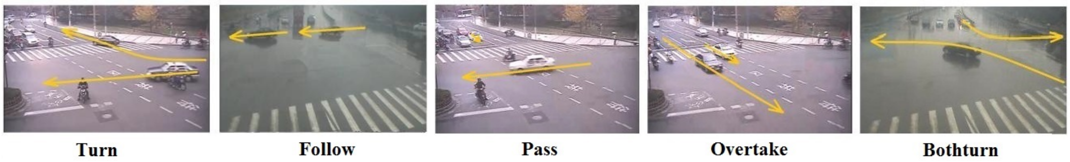


Figure 4: Sample of Traffic Dataset (Lin et al., 2013).

- We first directly compare the performance of our classification method (described in Section 3.3) against *state-of-the-art* quantitative and qualitative methods (AlZoubi et al., 2017; Lin et al., 2013; Lin et al., 2010; Ni et al., 2009; Zhou et al., 2008) using the traffic dataset presented in (Lin et al., 2013).
- Then, we evaluate the performance of our supervised method on our dataset for vehicle-obstacle interaction (VOI, 2018) using complete and predict trajectories.

All experiments were run on an Intel Core i7 desktop, CPU@3.40GHz with 16.0GB RAM.

4.1 Experiment 1: Classification of Vehicle Activities

The state of the art pair-activity classification methods (AlZoubi et al., 2017; Lin et al., 2013) were shown to outperform a number of other methods (Zhou et al., 2008; Ni et al., 2009; Lin et al., 2010), using the traffic dataset presented in (Lin et al., 2013). We therefore use these algorithms, traffic dataset, and ground truth, as a benchmark for evaluating our own classification method.

4.1.1 The Traffic Dataset

The traffic dataset was extracted from 20 surveillance videos. Five different vehicles-activities, $\{$*Turn*, *Follow*, *Pass*, *BothTurn*, and *Overtake*$\}$ are represented and corresponding annotations are provided. In total there are 175 clips; 35 clips for each activity. The dataset includes x, y coordinates for the centroid of each vehicle in each frame, a time-stamp t, and each clip contains 20 frames. Figure 4 shows example frames from the dataset, and Table 1 shows the definitions of each activity in the dataset. To best of our knowledge this is the only pair-wise traffic surveillance dataset publicly available.

4.1.2 Results for Vehicle Activity Classification

We used the provided x, y coordinate pairs for each vehicle as inputs, and constructed corresponding QTC_C trajectories for each video clip. Then, we constructed corresponding image texture (I_i) for each QTC_C trajectory using Algorithm 1. Figure 5 shows images texture for five different interactions for the samples in figure 4.

Table 1: Definition of vehicles pair-activities.

Activity	Description
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.

To determine the classification rates using our method, we used 5-fold cross validation. On each iteration, we split the images texture (I) into training and testing sets at ratio of 80% to 20%, for each class. The training sets were used to parametrize our network ($TrajNet$). The test images texture were then classified by our trained $TrajNet$. Our results are shown in Table 2, which includes comparative results obtained by (AlZoubi et al., 2017), (Lin et al., 2013), and three other algorithms (Zhou et al., 2008; Ni et al., 2009; Lin et al., 2010).The average error (AVG Error) is calculated as the total number of incorrect classifications (compared with the ground truth labeling) divided by the total number of activity sequences in the test set. Table 2 shows that our method outperforms the other five, and is able to classify the dataset with errors rate 1.16% and with a standard deviation 0.015.

4.2 Experiment 2: Classification of Vehicle-Obstacle Interaction

We are primarily interested in the supervised recognition of the relative interaction between the ego-vehicle and its surroundings, which can help in the decision making and predicting any potential colli-

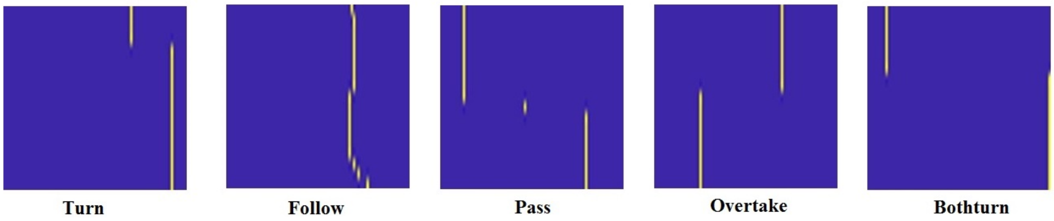


Figure 5: Images Texture Representation of Traffic Dataset.

Table 2: Misclassification error for Different Algorithms on the Traffic Dataset.

Type	*TrajNet*	*NWSA* (Al-Zoubi et al., 2017)	*Heat-Map* (Lin et al., 2013)	*WF-SVM* (Zhou et al., 2008)	*LC-SVM* (Ni et al., 2009)	*GRAD* (Lin et al., 2010)
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%
AVG Error	**1.16%**	**3.44%**	**4.58%**	**16.98%**	**27.24%**	**16.48**

Table 3: Definition of Vehicle-Obstacle Interactions.

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.

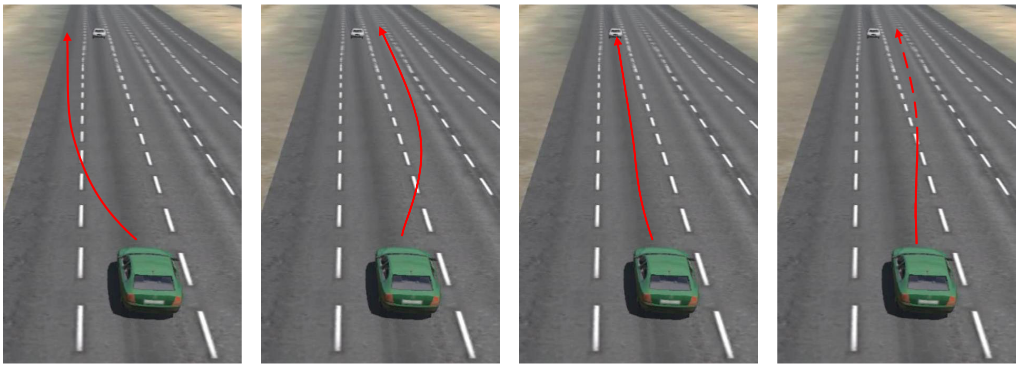
sions. We have constructed an extensive and expert-annotated dataset of vehicle-obstacle interactions, collected in a simulation environment. This presents a new challenging datasets, and therefore a new classification problem. In this section we evaluate the effectiveness and the accuracy of our supervised recognition method and our driver model for trajectory prediction.

4.2.1 Setup

For our pair-wise vehicle-obstacle interactions and our trajectory prediction model we collected data through simulation. As we have focused on near collision scenarios, there is not much available data, and real testing would not be possible for the crash scenario. Data was gathered using a simulation environment developed in Virtual Battlespace 3 (VBS3), with the Logitech G29 Driving Force Racing Wheel and pedals. Here a model of Dubai highway was used. We consider a six lane road with an obstacle in the center lane. The experiment consisted of 40 participants, all of varying ages, genders and driving experiences. Participants were asked to use their driving experience to avoid the obstacle. A Škoda Octavia was used in all trails, and with maximum speed 50KPH. We recorded the obstacle and ego-vehicle's coordinates (the x, y center position), velocity, heading angle, and distance from each other. The generated trajectories were recorded at 10Hz.

4.2.2 Simulator Dataset

We have developed a new dataset for vehicle-obstacle interaction recognition task (VOI, 2018). The dataset includes three subsets: the first (SS_1) contains of 122 vehicle-obstacle trajectories of about 600 meters each (43,660 samples) for training our trajectory prediction model. The second subset (SS_2) contains of 277 complete trajectories of three different scenarios (67 crash, 106 left-pass, and 104 right-pass trajectories); which we consider to be ground truth to evaluate the accuracy of our recognition method and our trajectory prediction model. Where the distance between the vehicle and the obstacle (length of the trajectory) is 50 meters. Here each scenario was observed and manually labelled by two experts. Table 3 shows the definitions of each scenario in the dataset. Figure 6 (a)-(c) shows samples of three different complete trajectories for Left-Pass, Right-Pass and Crash, respectively. Finally, the third subset (SS_3) contains of 277 incomplete trajectories. This subset is derived from the complete trajectories (SS_2) and used to evaluate our recognition method and our trajectory prediction



(a) Left-Pass (b) Right-Pass (c) Crash (d) Driver Model

Figure 6: Examples of scenarios from VBS3 with trajectory overlaid in red (a-c). The ego-vehicle is the green car and the obstacle is the white car. (d) shows an example of the driver model, where the solid red line is the real trajectory and the dashed line is the predicted trajectory.

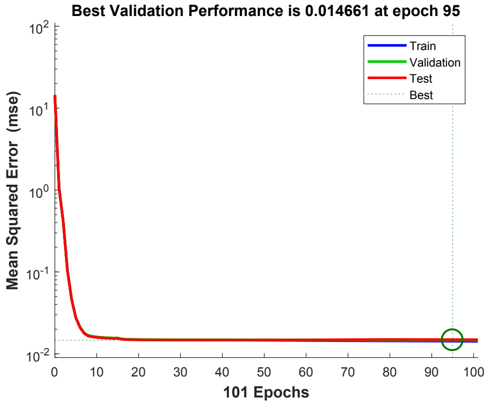


Figure 7: Training performance of driver model, and number of training iterations.

model. For trajectory prediction we considered a distance of $\varepsilon = 25$ meters between the ego-vehicle and the obstacle. This distance represented 50% of the complete trajectories. Figure 6 (d) shows an example of the incomplete trajectory for Right-Pass scenario, where the solid red line is the real trajectory (25 meters) and the dashed line is the predicted trajectory. Similarly, the third subset contains 67 crash, 106 left-pass, and 104 right-pass of incomplete trajectories with length 25 meters each.

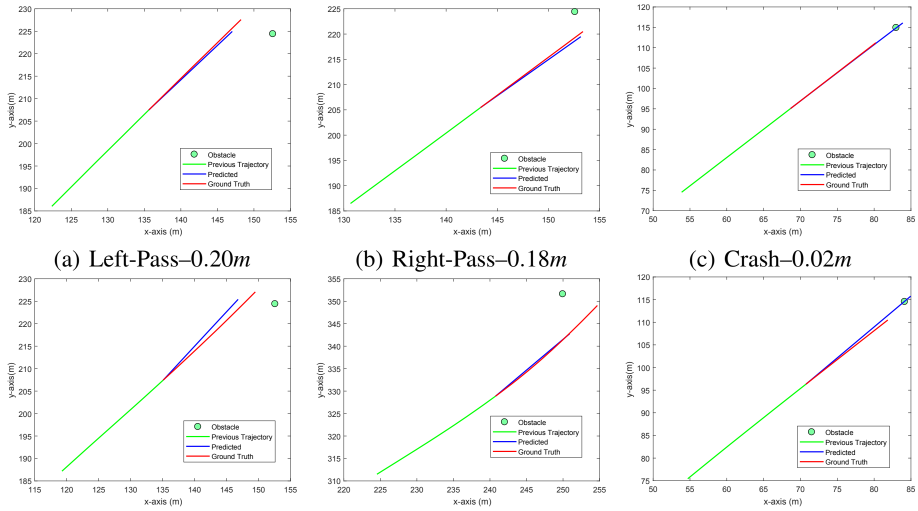
4.2.3 Evaluating Trajectory Prediction Model

We evaluated the accuracy of our driver model for trajectory predictions. First, we train our FFNN (Section 3.4) using $SS1$, containing 43,660 samples. We split the samples in SS_1 into training, validation and test subsets, 70%, 15% and 15%, respectively. The training subset is used to calculate network weights, the validation subset to obtain unbiased network parameters for training, and the test subset is used to evaluate performance. Figure 7 shows the performance of our network.

We evaluated the performance of our trajectory prediction algorithm on SS_3, consisting of 277 incomplete trajectories; we considered complete trajectories (SS_2) as the ground truth. Our test set contains 67 crash, 106 left-pass, and 104 right-pass trajectories. Three examples from each scenario of the generated trajectories are shown in figure 8. Here the trajectory starts closest to the origin and travels diagonally, from left to right. Human generated trajectories are in green and red (where red is the ground truth), the predicted trajectories are in blue. We calculate the trajectory prediction error using the Modified Hausdorff Distance (MHD) (Dubuisson and Jain, 1994). We selected the MHD because its value increases monotonically as the amount of difference between the two sets of edge points increases, and it is robust to outlier points. Given the predicted trajectory $\mathbf{T_v}$ and the ground-truth trajectory as $\mathbf{T_v}^{gt}$, we calculate our error measure as:

$$\mathcal{MHD} = min(d(\mathbf{T_v}, \mathbf{T_v}^{gt}), d(\mathbf{T_v}^{gt}, \mathbf{T_v})). \qquad (7)$$

where $d(*)$ is the average minimum Euclidean distances between points of predicted and ground-truth trajectories. The error across the entire test set is shown in figure 9. Here the red line represents the mean error and the bottom and top edges of the box indicate the 25^{th} and 75^{th} percentiles, respectively. The whiskers extend to cover 99.3% of the data and the red pluses represent outliers. Across all scenarios an average error of 0.4m was observed, this amount of error is tolerable because the overall characteristic of the trajectory is still captured, also, a typical highway lane can be up to 3m, therefore variations can still be within lane.



(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

Figure 8: Sample trajectories with associated errors, $\mathcal{MHD}$. Two sets of example trajectories (top and bottom rows), covering all three scenarios. (a)-(c) and (d)-(f) represent left-pass, right-pass, and, crash scenarios, respectively.

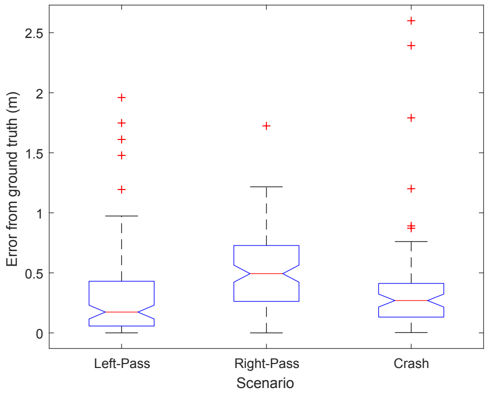


Figure 9: Errors ($\mathcal{MHD}$) from scenarios, across $SS3$.

4.2.4 Results for Vehicle-Obstacle Interaction Classification

We evaluated our recognition method on both complete (SS_2) and incomplete trajectories (SS_3). First, we used the x, y coordinate pairs for both the vehicle and the obstacle from SS_2 as inputs, and constructed corresponding QTC_C trajectories for each scenario. Then, we constructed the corresponding image texture (I_i) for each QTC_C trajectory using Algorithm 1. To determine the classification rates using our method on SS_2 dataset, we used 5-fold cross validation. On each iteration, we split the images texture in SS_2 into training and testing sets at ratio of 80% to 20% respectively, for each class. The training sets were used to parameterise our network ($TrajNet$). The test data was then classified by our trained $TrajNet$. Our results are shown in Table 4, which includes results of complete trajectories (*Complete Traj*). The average error (AVG Error) is calculated as the total number of incorrect classifications (compared with the ground truth labeling) divided by the total number of activity sequences in the test set.

For incomplete trajectories, we first trained and parametrized our FFNN (figure 3) using SS_1. Then, we used our driver model (Section 3.4) to predict the complete trajectory for each scenario which resulted in a new subset (SS_4). Similarly, we used the $x; y$ coordinate pairs for both the vehicle and the obstacle from SS_4 as inputs, and constructed QTC_C trajectories and their corresponding images texture (I) for all scenarios. Figure 6 (d) shows a sample of predicted trajectory. Again, to determine the classification rates using our method on SS_4, we used 5-fold cross validation. On each iteration, we split the images texture in SS_4 into training and testing sets at ratio of 80% to 20%, for each class. The training sets were used to parameterise our network ($TrajNet$). The test data was then classified by our trained $TrajNet$. Our results are shown in Table 4, which includes results of predicted trajectories (*Predicted Traj*). The results show that our method has high classification accuracy for both complete and predicted trajectories. The total computation time of predicting and recognizing the scenario that the vehicle is about to enter is 28 millisecond.

5 Conclusion and Discussion

In this paper we have presented a new method for predicting and classifying pair-activities of vehicles

Table 4: Misclassification error for Complete and Predicted Trajectories of Vehicle-Obstacle Interaction Dataset.

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%**

using a new deep learning framework. Our method uses the QTC representation, and we have constructed corresponding image textures for each QTC trajectory using one-hot vector. Our trajectory representation, successfully encodes different types of vehicles activities, and is used as an input for $TrajNet$. $TrajNet$ offers a compact network for classifying pair-wise vehicle interactions. We also demonstrate how we efficiently used limited amount of dataset to train $TrajNet$, and achieved high accuracy classification rates across different and challenging datasets.

We have conducted direct comparisons against the state-of-the-art qualitative (AlZoubi et al., 2017) and quantitative (Lin et al., 2013) methods, which have itself been shown to outperform other recent methods. We have shown that our classification method outperforms that developed by (Lin et al., 2013) and (AlZoubi et al., 2017); for the classification of traffic data, we achieved 1.16% error rate, compared to 3.44%, 4.58%, 16.98%, 27.24%, and 16.48% of (AlZoubi et al., 2017), (Lin et al., 2013), (Zhou et al., 2008), (Ni et al., 2009) and (Lin et al., 2010), respectively.

We have also presented our vehicle-obstacle interaction dataset for complete and incomplete scenarios, which provides a detailed and useful resource for researchers studying vehicle-obstacle behaviors, and is publicly available for download. We evaluated our classification method on this dataset, we achieved 0.0% and 0.3% for both complete and predicted scenarios datasets, respectively. This again demonstrates the effectiveness of our activity recognition method. To predict a full scenario from partial-observed one we have presented a FFNN. We evaluated our trajectory prediction method on the same vehicle-obstacle dataset and we achieved average error of 0.4m.

Encouraged by our results, we plan to extend our work by integrating our vehicles activity recognition method with our ongoing project of autonomous vehicle system to provide valuable information about the type of the scenario the vehicle is in (or about to enter) to increase the safety and to help in decision making processes.

REFERENCES

(2018). Voidataset.

Ahmed, S. A., Dogra, D. P., Kar, S., and Roy, P. P. (2018). Trajectory-based surveillance analysis: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*.

AlZoubi, A., Al-Diri, B., Pike, T., Kleinhappel, T., and Dickinson, P. (2017). Pair-activity analysis from video using qualitative trajectory calculus. *IEEE Transactions on Circuits and Systems for Video Technology*.

Chavoshi, S. H., De Baets, B., Neutens, T., Delafontaine, M., De Tré, G., and de Weghe, N. V. (2015). Movement pattern analysis based on sequence signatures. *ISPRS International Journal of Geo-Information*, 4(3):1605–1626.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Dodge, S., Laube, P., and Weibel, R. (2012). Movement similarity assessment using symbolic representation of trajectories. *International Journal of Geographical Information Science*, 26(9):1563–1588.

Dubuisson, M.-P. and Jain, A. K. (1994). A modified hausdorff distance for object matching. In *Proceedings of 12th international conference on pattern recognition*, pages 566–568. IEEE.

Hanheide, M., Peters, A., and Bellotto, N. (2012). Analysis of human-robot spatial behaviour applying a qualitative trajectory calculus. In *RO-MAN, 2012 IEEE*, pages 689–694. IEEE.

Khosroshahi, A., Ohn-Bar, E., and Trivedi, M. M. (2016). Surround vehicles trajectory analysis with recurrent neural networks. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 2267–2272. IEEE.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lin, W., Chu, H., Wu, J., Sheng, B., and Chen, Z. (2013). A heat-map-based algorithm for recognizing group activities in videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(11):1980–1992.

Lin, W., Sun, M.-T., Poovendran, R., and Zhang, Z. (2010). 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.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26.

Ni, B., Yan, S., and Kassim, A. (2009). Recognizing human group activities with localized causalities. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1470–1477. IEEE.

Ohn-Bar, E. and Trivedi, M. M. (2016). Looking at humans in the age of self-driving and highly automated vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):90–104.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1717–1724. IEEE.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Shi, Y., Tian, Y., Wang, Y., and Huang, T. (2017). Sequential deep trajectory descriptor for action recognition with three-stream cnn. *IEEE Transactions on Multimedia*, 19(7):1510–1520.

Shi, Y., Zeng, W., Huang, T., and Wang, Y. (2015). Learning deep trajectory descriptor for action recognition in videos using deep neural networks. In *Multimedia and Expo (ICME), 2015 IEEE International Conference on*, pages 1–6. IEEE.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Van de Weghe, N. (2004). *Representing and reasoning about moving objects: A qualitative approach*. PhD thesis, Ghent University.

Xiong, X., Chen, L., and Liang, J. (2018). A new framework of vehicle collision prediction by combining svm and hmm. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):699–710.

Xu, D., He, X., Zhao, H., Cui, J., Zha, H., Guillemard, F., Geronimi, S., and Aioun, F. (2017). Ego-centric traffic behavior understanding through multi-level vehicle trajectory analysis. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 211–218. IEEE.

Xu, H., Zhou, Y., Lin, W., and Zha, H. (2015). Unsupervised trajectory clustering via adaptive multi-kernel-based shrinkage. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4328–4336.

Zhou, Y., Yan, S., and Huang, T. S. (2008). Pair-activity classification by bi-trajectories analysis. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.