Behaviour Detection of Vehicles Using Graph Convolutional Networks with Attention

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Electronics and Communication Engineering by Research

by

Sravan Mylavarapu
201531197
saisravan.m@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
FEBRUARY 2021

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled " Behaviour Detection of Vehicles Using Graph Convolutional Networks with Attention " by Sravan Mylavarapu, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Dr. Anoop Namboodiri, Co.Adviser: Dr. Madhava Krishna,

To

My Family

for their unconditional love and support

Acknowledgments

I would like to thank all the people who have supported me and motivated me throughout my journey at IIIT.

First of all, I would like to express my sincere gratitude to my guide, Dr.Anoop Namboodiri for introducing me to research, Co-guide, Dr.Madhava Krishna for his constant support, and assisting me in the field of robotics. I want to thank Dr.Ravindra Balaraman for his support and guidance. Research has taught me to be patient, consistent and dedicated throughout my life and I consider it an important life lesson. The constant guidance provided by my guide helped me to develop both professionally and personally.

I am also grateful to my colleagues Mahtab Sandhu, Priyesh Vijayan and Gulshan Mittal for helping me in the initial stages and for taking part in the brain storming discussions and ideas they have been sharing with me throughout this journey. I would like to specially thank, Sarthak Sharma who was the first person apart from my guide to introduce me to robotics and inculcate interest in it.

I would also like to thank my friends Akhil Ralla, Sai Teja Reddy Moolamalla, Sudheer Achary, Shanmukh Alle, Siddharth Laguvarapu and Rama Rohit Reddy Gangula for making my college life more cherishable. Last but not the least, I express my deepest gratitude toward my parents, Murthy Mylavarapu and Sailaja Mylavarapu, for their unconditional love and support.

Abstract

Autonomous driving has evolved with the advent of deep learning and increased computational capacity in the past two decades. While many tasks have evolved that help autonomous navigation, understanding on-road vehicle behaviour from a continuous sequence of sensor data (camera or radar) is an important and useful task. Determining state of other vehicles and their motion helps in decision making for the ego-vehicle (Self/Our Vehicle). Such systems can be a part of *driver assistance* systems or even *path planners* for vehicles to avoid obstacles. Instances of such decision making could be, deciding to apply brakes incase an external agent is moving into our lane or coming opposite to us. Another instance would be keeping distance from an aggressive driver who is changing lanes and overtaking frequently could be helpful and safer. Specifically, our proposed methods classify the behaviours of on-road vehicles into one of the following categories-{Moving Away, Moving Towards Us, Parked/stationary, Lane Change, Overtake}.

Many current methods leverage 3D depth information using expensive equipment, such as Lidar and complex algorithms. In this thesis, we propose a simpler pipeline for understanding vehicle behaviour from a monocular (single camera) image sequence or video. This will help in easier installation of the equipment and helps in *Driver Assistance systems*. A simple video along with Camera parameters, scene semantics (Detected Objects in the images) are used to get information about the objects of interest (vehicles) and other static objects like lanes/poles in the scene.

We consider detecting objects across frames as the pre-processing pipe-line and propose two main methods, 1.Spatio-temporal MRGCN (Chapter:3) and 2.Relational-Attentive GCN (Chapter:4) for behaviour detection of vehicles. We divide our process of identifying behaviours into learning positional information of vehicles first, and then observing them across time to make a prediction. The positional information is encoded by a Multi-Relational Graph Convolutional Network (MR-GCN) in both the methods. Temporal information is encoded by recurrent networks in Method 1, while it is formulated inside the graph in Method 2. We also showcase how *Attention* is an important aspect in both our methods through our experiments. The proposed frameworks can classify a variety of vehicle behaviours to high fidelity on datasets that are diverse and include European, Chinese and Indian on-road scenes. The framework also provides for seamless transfer of models across datasets without entailing re-annotation, retraining and even fine-tuning. We provide comparative performance gain over baselines and detail a variety of ablations to showcase the efficacy of the framework.

Contents

Chapter Page

1 Introduction . . . 1
  1.1 Graph Convolutional Networks . . . 3
  1.2 Recurrent Networks . . . 3
  1.3 Overview of Proposed Methods . . . 4
    1.3.1 Spatio Temporal MRGCN . . . 4
    1.3.2 Relational-Attentive GCN . . . 5
  1.4 Contribution of the thesis . . . 5

2 Related Work . . . 6
  2.1 Vehicle behaviour Understanding . . . 6
  2.2 Action Recognition and Driver Prediction . . . 7
  2.3 Lane Change and Recurrent Networks . . . 7
  2.4 Graph Based Formulations . . . 8
  2.5 Graph Neural Networks . . . 8
  2.6 Summary . . . 9

3 Spatio-Temporal MRGCN . . . 10
  3.1 Overview . . . 10
  3.2 Pre-Processing . . . 11
    3.2.1 Vehicle and Landmark Detection . . . 11
    3.2.2 Bird's Eye-View Generation . . . 12
  3.3 Scene Graph Generation . . . 14
  3.4 Proposed Architecture . . . 15
    3.4.1 MR-GCN . . . 15
    3.4.2 LSTM . . . 16
    3.4.3 Attention . . . 17
  3.5 Experiments and Analysis . . . 19
    3.5.1 Datasets . . . 19
      3.5.1.1 Apollo-Scape Dataset . . . 19
      3.5.1.2 KITTI Dataset . . . 19
      3.5.1.3 Indian Dataset . . . 19
    3.5.2 Quantitative Results . . . 20
      3.5.2.1 Ablation Study on Model . . . 20
      3.5.2.2 Baselines Comparison . . . 21
      3.5.2.3 Transfer Learning . . . 21

3.5.2.4 Ablation Study on LandMarks 22
3.5.3 Implementation Details 23
3.5.4 Qualitative Analysis 23
3.6 Summary 24

4 Relational Attentive GCN 25
4.1 Overview 25
4.2 Temporal Interaction Graph Generation 25
4.3 Proposed Architecture 26
4.3.1 Multi-Relational Graph Convolution Networks 26
4.3.2 Relation-Attentive MRGCN 27
4.4 Experiment and Analysis 28
4.4.1 Dataset 28
4.4.2 Baseline Comparisons 29
4.4.2.1 Adapting St-Rnn 29
4.4.3 Quantitative Results 30
4.4.4 Relational-Attentive MRGCN (Rel-Att-MRGCN) 30
4.4.5 Transfer Learning 30
4.4.6 Implementation Details 31
4.4.7 Qualitative Analysis 32
4.5 Summary 33

5 Conclusions and Future Work 35

6 Appendix 37
6.1 Overview 37
6.2 Gating 37
6.2.1 Difference between Gating and Attention 39
6.3 Temporal Edge 39

Bibliography 42

List of Figures

Figure Page

1.1 Intuitive Explanation. The figure depicts how objects (vehicle) behaviour can be modelled as a graph. The evolution pattern of this graph helps us to classify the behaviour. Here the car (red) moves ahead of a lane marking (yellow). This change in relation helps us identify that the car is moving ahead. 2

1.2 Simple and relational Graphs. (a) denotes a simple graph with all edges of same nature. (b) shows a multi-relational graph where the color of edge denotes its nature. For e.g, all orange colored edges are of the same type/relation, similarly all blue colored are of one type/relation. 3

1.3 LSTM. A simple block representation of LSTM with the corresponding notations. 4

3.1 This illustrates the overall pipeline of our framework. The input (a) of the system consists of consecutive frames of a monocular video from cameras mounted on the cars. We use traditional object tracking pipelines, as shown in (b) to extract features. These extracted features/tracklets are then transferred into the bird's-eye view as in (d). Using these pre-processed features in bird's-eye view, we generate spatial scene graphs as illustrated with examples in (e). MR-GCNs are used to encode spatial-relations between objects from this scene graph. Subsequently, this spatial information from MR-GCNs over different frames are passed to an LSTM to model the temporal evolution of relations and predict the different vehicle behaviours as in (g). The two cars in (g) in Red and Green depict *Moving Away from Us* and *Moving towards Us* respectively. 10

3.2 Instance segmentation. Top row is the original image. Second row shows detected vehicles. 11

3.3 Semantic segmentation. Top row is the original image, bottom contains the segmented output. 12

3.4 Bird's Eye-View Example. (a) shows detected cars in red and green and lane markings in yellow. (b) shows corresponding Bird's eye view we want to achieve with objects as a single point on the road. 13

3.5 Bird's Eye View generic case. Generic case of a camera capturing images. The camera is tilted at an angle α, at a height h from ground. η denotes the normal to the ground surface. 13

3.6 Scene Graph generation. In the figure, (a) denotes Image frames at t=1, t=5 and t=10 with detected objects (vehicles and lane-markings). (b) shows Bird's Eye View for each time-step. Generated scene graphs at each time step are shown in (c). The colored edges in the graphs denote different relations between objects as described in the legend. 14

3.7 Multi-Head attention Explanation. Attention is applied for each node independently across time-steps. In the figure, (a) denotes the obtained LSTM embeddings. These are passed through self-attention layers (b), which generate attention vectors Z as shown in (c). Each box in (c) denotes output from a single head, and each head has 10 vectors for 10 time-steps. These are concatenated across heads as shown in (d). t=1 is colored in red, t=2 in green and t=10 in orange. In (d), we see concatenated attention vectors with the same color-coding depicting concatenation is done among vectors of equal time-step. These are back projected to required dimension to obtain (e). 18

3.8 Results on KITTI dataset. Accurate classification of Vehicle behaviour can be seen based on the color over the vehicle. Red denotes *Moving Away*, Green denotes *Moving towards Us* 23

3.9 Results on Indian dataset. We can see Vehicle class agnosticism of our Pipeline as non standard Vehicles such as bus and oil-tanker are correctly classified. 24

3.10 Results on Apollo Dataset. Figures depict complex Vehicles maneuvers such as overtaking, which can be seen in (a), the car on the right represented in magenta color and lane change which can be seen in image (b) and (c) with yellow on the Apollo dataset. 24

4.1 **Temporal Interaction Graph Generation:** The figure shows the detailed process of generating a temporal interaction graph after tracking objects in the scene. The top row contains image frames at t=0, t=5, t=10 time step with tracked objects. The bottom row shows the corresponding bird's eye view. In the scene, a car and two lane-markings are tracked at each time step. The thick edges in the bird's eye view denote the spatial relations between objects, and color coding for the spatial relations are shown in legend beside. The red, blue edges between car, L1 in first and the final frame, denote the spatial relations between them. The car moves forward with respect to L1 and has no relational change with L2. These temporal relations are represented with dotted lines in the graph shown as Temporal Interactions at the right. Corresponding color coding is shown in legend *Temporal interactions*. 26

4.2 This illustrates the overall pipeline of our framework. The input (a) to the pipeline is monocular image frames captured from a camera mounted on a vehicle. Similar to Ch.3, various object tracking pipelines are used to detect and track objects as shown in (b). Objects are tracked in each frame. Extracted tracklets for each object are projected to 3D space using Bird's eye-view Generator at each time step (c). We leverage the spatial relations obtained from bird's eye view and generate *Interaction Graphs* that are passed through a Rel-Att-GCN (d). (d) captures relational information to classify objects in the scene. (e) shows two vehicles in red and green classified as *Moving-away from us* and *Moving towards us* respectively. 27

4.3 Attention Map with rows as class-labels and columns as relations. Higher scores for a particular relation shows higher dependence of the class on that particular relation. Abbreviations for class labels are, MVA : moving away from us, MTU: moving towards us, PRK: parked, LCL: lane change left to right, LCR: lane change right to left, OVT: overtaking. 31

4.4 Results on Apollo and Honda datasets. Figure shows multiple scenarios depicting various behaviours such as overtaking and lane changing. (a), (b) are samples from Apollo scapes dataset while (c) and (d) are from Honda dataset. In (a), (c) and (d) we observe the model accurately classifying lane change, both left to right and right to left. (b) depicts the case of overtaking with the car in magenta overtaking the car moving forward in red. 33

4.5 Results on KITTI and Indian datasets. Figure depicts accurate behaviour classification of standard and non-standard vehicles. (a) and (b) show cases from KITTI dataset with standard vehicles. Predictions on Petrol Tanker and Bus in (c) depicts object class agnosticism and (d) denotes complex behaviours like lane changes. 34

6.1 Figure shows the Gating mechanism for a single node surrounded by neighbors with different relations between them. (a) shows the various relations between the center blue node and its neighbors. (b) shows the way gating is applied by passing information from neighbors through a GATE (shaded boxes) based on the type of relation. The gating value helps in determining the importance of information obtained from a neighbor node. 38

6.2 Figure shows an example graph used for Temporal Edge Method. (a), (b), (c) show Spatial graphs at time steps t-1, t, t+1 respectively. These graphs hold spatial relations between nodes (blue, red, black and green edges) individually. The golden-yellow edges from t-1 to t and t to t+1 depict the temporal edges. Such a single graph for T time steps is created and passed to the network. 40

List of Tables

Table Page

3.1 Confusion matrix for model trained on Apollo dataset. We have used abbreviations for label names, MVA : moving away from us, MTU: moving towards us, PRK: parked, LCL: lane change left to right, LCR: lane change right to left, OVT: overtaking and GT : Ground truth. 20

3.2 Baselines and model variations, trained on Apollo. Values provided are accuracies. Abbreviations for model architectures (rows) are, G : MR-GCN, L : LSTM, MA : Multi-Head Attention, SA : Single-Head Attention. Nomenclature in the architecture names reflect the components in it. Column abbreviations are MVA : moving away from us, MTU: moving towards us, PRK: parked, LCL: lane change left to right, LCR: lane change right to left and OVT: overtaking . 21

3.3 Transfer learning. All values are accuracies. The 2 columns under Apollo denote the model trained on APOLLO and tested on KITTI and Indian datasets. Last two columns are results with models trained on their respective datasets; i.e., trained on KITTI and tested on KITTI, trained on Indian and tested on Indian data. 22

3.4 Ablation study on landmarks. Comparison between models trained with 50% of landmarks and 75% and 100% of landmarks. Values provided are accuracies. 22

4.1 Baseline comparisons on Apollo Dataset. Columns 2 to 7 denote the classes we predict. First column denotes the methods used. Columns for each method denote the class-wise accuracies in percentage. 29

4.2 Transfer learning results. We provide results of transfer learning on all datasets. We train on Apollo Scapes dataset and test on Honda, KITTI and Indian. The second row denotes the datasets and third row denotes the method used. Accuracies of each class are depicted from 4th row onwards . Both the pre-trained models, MRGCN-LSTM and REL-ATT-GCN are tested on all three datasets and all accuracies reported are in percentages. 32

4.3 Test results on all datasets. We provide accuracies for different models trained and tested on multiple datasets. The top row denotes the dataset used for training, and testing on the same with a validation split. The second row denotes the method used. The 2 Columns under each dataset denote the class wise accuracies for both the models , MRGCN-LSTM and REL-ATT-GCN on that dataset. All the accuracies are in percentage. 32

6.1 Comparison is based on Apollo Dataset. Comparison between Gating and Rel-Attention mechanisms. Values are accuracies in percentage. 39

Chapter 1

Introduction

Autonomous vehicles have emerged since the mid of the twentieth century. With the rise of computational power and portable equipment like radars and depth cameras, it has become a significant field of research and study in the last twenty years. A rise in the industrial market and a possibility for autonomous vehicles has been observed with the advent of computer vision, robotics and deep learning. The field of autonomous navigation has many tasks and behaviour understanding of other vehicles is one of the crucial tasks as it helps in taking decisions for the driver control. Many pre-existing methods fall into two major categories. *Modular pipelines*, leveraging information from the environment such as rpm, acceleration, steering-angle from hardware equipment and predicting the required output. *Imitation learning*, an end-end process where images are directly fed to the algorithms that detect objects of interest and predict outputs. The outputs are fed to the controls of the vehicles directly. Our work falls into the recently proposed third paradigm where low level understanding of the scene is obtained using deep networks by mapping them to an intermediate representation. These representations are used by various algorithms to predict the corresponding outputs. Our problem confines to understanding vehicle behaviours in a given scene using such intermediate representations with a learning based approach.

Scene Understanding in terms of vehicles is a vast research-area and laden with many applications from autonomous driving to traffic violations. Categorizing on-road vehicle behaviours into one of the following categories: parked, moving away, moving towards us, lane change from left, lane change from right, overtaking helps in motion detection. Instances of decision making in autonomous driving could be avoiding and maintaining distance from aggressive drivers who generally tend to overtake, change lanes continuously. Another instance could be a vehicle cutting into our lane and our decision being opposite to it, i.e., vehicle cutting in from right to left would indicate us to move from left to right to avoid collision. Our model is generalizable and could be modeled in other segments such as activity-recognition of humans, pose estimation etc (not part of the current work).

Considering each object to be a single point in the scene, in both the frameworks (Ch.3, 4) we decompose the traffic scene into a graph. This graph is used by our models to predict behaviours. Intuitively, a car's temporal evolution with landmarks like lanes/poles help us, humans, to identify its behaviour. As seen in the Fig.1.1 the change of car's spatial/positional relation with lane marking tells us

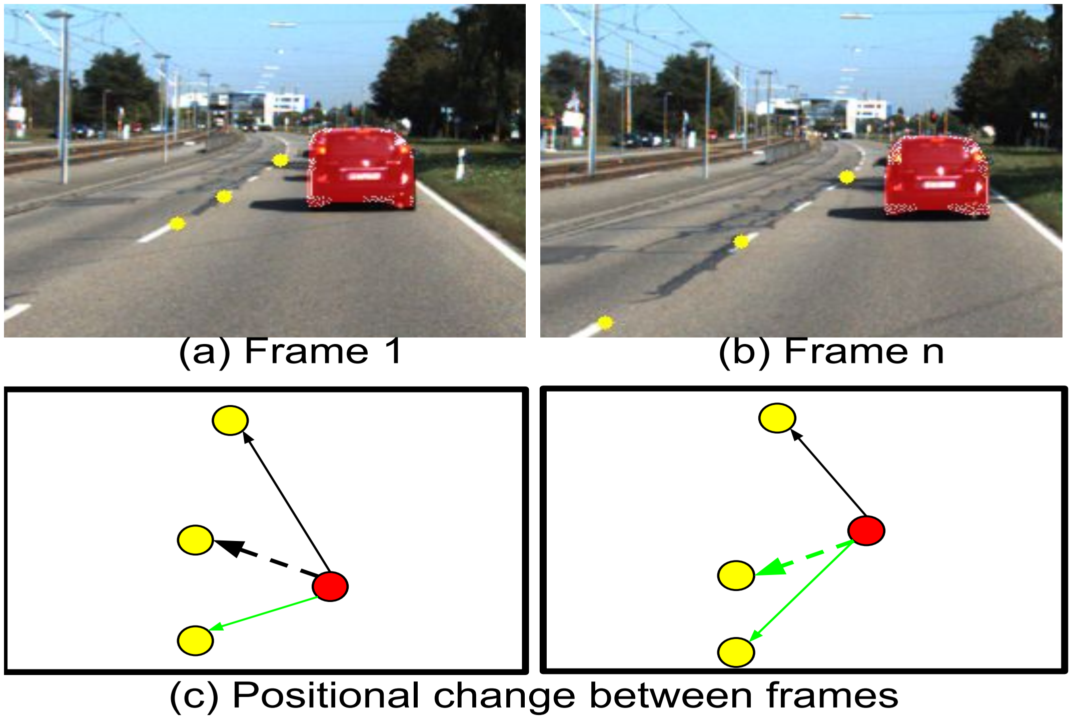


Figure 1.1 Intuitive Explanation. The figure depicts how objects (vehicle) behaviour can be modelled as a graph. The evolution pattern of this graph helps us to classify the behaviour. Here the car (red) moves ahead of a lane marking (yellow). This change in relation helps us identify that the car is moving ahead.

that the car is moving away from us. The figure depicts how objects (vehicle) behaviour can be modeled as a graph. The evolution pattern of this scene graph helps us to classify behaviours. Here the car (red) moves ahead of a lane marking (yellow) . The change in spatial-relation helps us identify that the car is moving ahead. In a similar vein, an object's relational change with respect to static objects decides its lane change behaviour while its changing relations with another moving car in the scene governs its overtaking behaviour.

Many existing methods (ref Ch.2) use LIDAR, facial expressions of the driver with additional internal camera, to detect the intention/motion of the driver/vehicle. In our case, we explicitly need the camera intrinsic parameter and camera height (from the ground) as the only additional information apart from the monocular videos. Capturing behaviour of moving vehicles is a complex task due to multiple possibilities involved. Hence to overcome this, we assume all objects to be on a planar surface. First, we detect and track objects of interest (vehicles) and landmarks (lanes or poles) in the video frames in image space using existing pre-processing techniques. Tracked objects are then projected onto a planar surface, which are then used to generate a graph with nodes as objects in the scene. To detect vehicle behaviours, we use a model consisting of two main components, *Graph Convolutional Networks* and *Recurrent Networks* in our frameworks.

1.1 Graph Convolutional Networks

Graph Convolution Networks (GCN)[1] have shown promising results in learning graph-structured data. Hence, we show that modeling data using graph based method like ours is better than those lacking graph structure modeling. Apart from the graphical structure modeling, possibility to train the model with varying number of objects is a major motivation to use GCNs unlike many other existing methods where number of objects are constant. GCNs are capable of capturing information from neighbors, which help in classifying behaviours. As seen in Fig.1.1 (c), information from lane-marking (yellow) that the car (red) has moved forward, helps in classifying the behaviour of the car (red).

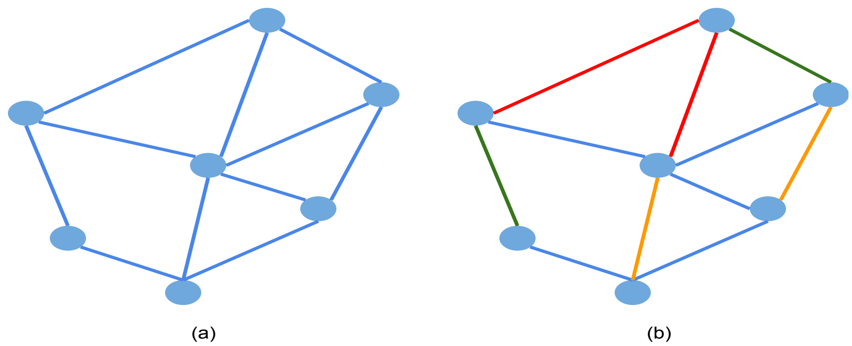


Figure 1.2 Simple and relational Graphs. (a) denotes a simple graph with all edges of same nature. (b) shows a multi-relational graph where the color of edge denotes its nature. For e.g, all orange colored edges are of the same type/relation, similarly all blue colored are of one type/relation.

In general, GCNs are used to model graph data where edges in the graph are all similar. In our case, we quantize the edges between objects into multiple relations between them and hence require a Multi-Relational Graph Convolutional Network (MRGCN)[2]. Edges in our graph have a particular relational value. MRGCN helps capture spatial relations between objects for each relation independently. These embeddings obtained from MRGCN are then fed to a recurrent network(red Ch.3) .

Fig.1.2 above shows two graphs similar in structure but varying in terms of the number of relations. Fig.1.2 (a) contains edges of a single type through out while (b) consists of edges with multiple relations between objects. Different colors in (b) depict the type of relations in the graph.

1.2 Recurrent Networks

Recurrent networks (Long Short Term Memory-LSTM) [3] have been promising in learning temporal information and are highly effective. As positional information is being captured by the MRGCN, temporal aspect of the relations need to be captured across frames. For this, we use LSTMs as shown in Fig.1.3, due to their ability to handle exploding and vanishing gradients. We do not use GRU (Gated recurrent units) [4] even though they are comparatively faster than LSTM as the dimensions to recurrent units are very small and training/testing time is negligibly effected in our case.

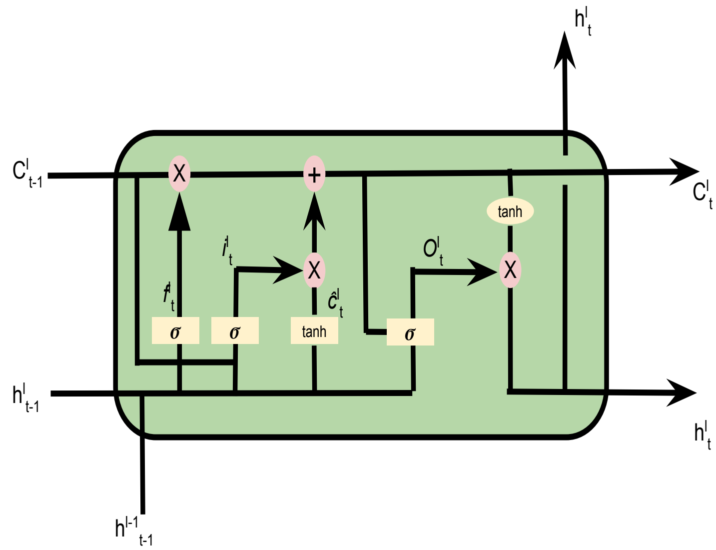


Figure 1.3 LSTM. A simple block representation of LSTM with the corresponding notations.

1.3 Overview of Proposed Methods

We propose two main methods, 1.Spatio-Temporal MRGCN (Ch.3) and Relational-Attentive GCN (Ch.4). In both cases, the objective is to classify behaviour of vehicles. We show how method 1 outperforms other existing methods and later show how method 2 is a better formulation over 1 and provide results for both.

1.3.1 Spatio Temporal MRGCN

Spatio Temporal Graph Convolutional Networks have been used in modeling skeletal graph structure to effectively predict action of a human. This confines to human joints alone with a restriction on the number of nodes. We incorporate MRGCN and LSTM as two individual modules which helps in dealing with any number of nodes in the scene.

This formulation is a complete end-end model wherein given a video sequence, we predict vehicle behaviours (except ego-centric). A graph is generated containing objects tracked in the video frames. A single graph exists for each time step t with relations between objects as one of the following {top-left, bottom-left, top-right, bottom-right}. Spatial relations per each time-step are encoded by the MR-GCN. Temporal aspect of such encodings is modelled through recurrent networks like LSTM. A key component for this framework is detecting change in relation between objects. Hence, to improve encodings obtained from the LSTM, we weight them using self-attention [5] for predicting the behaviour of vehicles.

1.3.2 Relational-Attentive GCN

In contrast to the above method, this method is not end-end. Two major changes in the framework are 1. The creation of a single graph instead of multiple graphs and 2. Attention is incorporated in the MR-GCN itself, unlike the previous case where it was applied at the end. The relations here would be $\{$Moved Ahead, Moved Backward, Moved left-to-right, Moved right-to-left$\}$. We have removed the recurrent networks as in previous method and incorporated the temporal and spatial information together within a single graph instead of learning them individually for each time step. This requires lesser parameters to learn and more definitive information is fed into the MR-GCN.

This graph when fed to the MR-GCN predicts the behaviours of vehicles. Even in this architecture, attention plays a key role by helping to eliminate redundancy (ref Ch.4). We proposed a novel Relational-Attentive mechanism inside the MR-GCN similar to the Graph Attention Networks [6] but over relations instead. This formulation is more accurate, and dividing the pipeline into smaller components helps in identifying flaws in case of wrong predictions.

1.4 Contribution of the thesis

Specifically the thesis contributes as follows

- We propose a novel yet simple scheme for spatial behaviour encoding from a sequence of single camera observations, making use of straight forward projective geometry techniques. We showcase using Multi-relation GCN (MR-GCN) as an effective way of modeling Spatio-temporal relations in on-road scenes. Though Graph Convolutional Networks (GCNs) have gained traction recently in the context of modeling relational aspects of a scene, to the best of our knowledge this is perhaps the first such reporting of using an MR-GCN in an end-to-end framework for on-road temporal-scene modeling.

- Critically we show incorporating an attention function [5] in both the frameworks leads to high performance which the MRGCN without attention function is unable to replicate.

- We reproduce the performance benefits of our frameworks over a variety of datasets, namely KITTI, Apollo-scape, Honda-H3D and Indian datasets to showcase its versatility. We achieve an improvement of at least 10% for simple classes (Moving ahead, Parked) and 20% for complex classes (Overtake and lane-changes) over baselines in both our architectures (ref Table.4.1). We show seamless transfer across multiple datasets preventing any entailment for annotation, retraining and fine-tuning. Extensive experiments on various on-road participants (not restricted to cars) such as in Indian roads along with multiple ablation studies is another feature of this effort.

Chapter 2

Related Work

Vehicle behaviour understanding is a well known task, but as mentioned, is constrained to have costlier equipment and generally used for driver assistance. To the best of our knowledge, there has not been much prior art that deals with classifying the behaviour of objects in outdoor scenes by just leveraging monocular images. There have been works involving complex setup for such tasks using LIDAR and face-cameras. We present works in the field of robotics, vision and action recognition segregating them based on the method and task.

2.1 Vehicle behaviour Understanding

Few of the closest works are [7, 8, 9, 10]. In [9, 10] vehicles behaviour is classified using a probabilistic model. The classification is with respect to the ego-vehicle (Self/Our Vehicle from which outside world is captured), for example, a car is only to be classified as overtaking if it overtakes the ego vehicle. Similarly, a car is classified as lane-change only if the ego-vehicle changes lane. Whereas in our proposed method, we classify vehicle behaviours with respect to a global frame, i.e., we are able to identify and classify overtaking and other behaviours of all the objects present in the visible scene. [7] uses probabilistic methods along with stereo input and determines the direction a car would be taking at an intersection. Our framework is able to classify more complex behaviours based only on monocular images using a simpler setup. [11] classifies road topology, which assists the driver similar to ours, but we use cues from static objects on road and classify object's behaviour rather than just a simple classification of the road-surface.

Methods [12, 13, 14] predict trajectories based on the object's previous interactions and trajectory. These methods are concerned with predicting future motion rather than classifying maneuvers.

Geng et al. [15] follows a hard rule-based approach for the task of maneuver prediction in contrast to our learnable approach, which is more generic and applicable to other fields not pertaining to robotics alone.

In Chen et al.[8], traffic scenes are summarized into textual output. Based on the objects in the scene, a specific understanding of the agents is given as input. Though similar to our approach here, it generates a textual output for the complete scene.

2.2 Action Recognition and Driver Prediction

The works [16], [17], [18] are close to our proposed method but uses various external sensors such as GPS, face camera and vehicle dynamics, and the classification is for the ego-vehicle alone.

Incase of Jain et al. [16], the authors have used dual camera setup wherein there exists an internal camera and external camera. The external camera captures the road scene while the internal camera captures facial expressions of the driver. In a case of intersection or a possible left/right turn, they predict the action the driver would take based on the facial movement and the information from the outside camera. They have used a hidden markov model for predicting the driver decision.

Specifically, St-RNN [17] lacks graphical structure but is extremely close to ours as it uses multiple relations between objects. While all other methods that have used graph based neural networks have confined to single relation between objects, the authors have proposed multiple relations between objects. St-RNN [17] proposed mainly for activity recognition, also includes other domains such as human skeletal motion modeling. In terms of activity recognition, it uses static objects and humans to encode relations between them in a video sequence. They predict the behavior of the humans based on the relations between humans and objects over time. They use recurrent networks for their purpose where a LSTM exists for each relation. The only difference being the lack of graph based network like ours, to encode relational information between objects. Though driver anticipation is a part of the work, it classifies only ego-vehicle behaviour and uses external sources like face-cameras.

2.3 Lane Change and Recurrent Networks

[19, 20] estimate the nature of a vehicle, whether it is keeping lane or changing lane. While [19] uses HMMs, [20] uses SVM with bayesian networks to predict the probability of lane-change. Both use many other parameters like steering angle, velocity etc and classifies ego-vehicle alone. [21, 22] both predict lane changes with the help of radars and multiple cameras placed around the ego-vehicle. These don't use rich available features from the images but depend on dynamic inputs from the vehicle during run-time and classify whether there could be a lane change or not. [21] uses information from multiple inputs, captures features and classifies using an SVM at the end.

Scheel et al.[23], also uses attention on top of recurrent networks similar to ours, to predict lane changes of other vehicles in the scene. But the method still uses many additional features from the surroundings like speed of vehicles, their angle etc. Tracking all objects and their velocities, it learns a recurrent based network along with attention to detect lane changes for other vehicles. Also, the number of vehicles in the scene is fixed in the whole network with the nearest vehicles in all directions. Our approach is more diverse and allows addition of more classes apart from just lane changing. We can also include classes such as *Merging into lane, Exiting the road* etc with no constraint on the nodes.

2.4 Graph Based Formulations

Graph-RQI [24] identifies driver behaviours of other vehicles in a video sequence from road-agent trajectories. Their work detects driving traits, such as aggressive or conservative driving by processing a set of monocular videos. These inter-agent interactions are represented using unweighted and undirected traffic graphs and use topological graphs across time to classify the behaviour. A vehicle is aggressive if the topological graph for it changes too frequently with other vehicles in the scene. A vehicle is conservative if the graph is merely constant across time. This is similar to our case where we check for relational changes. Our approach detects objects and also static landmarks across frames and generates graphs, similar to it to classify agents at the final step, difference being their Eigen vector based method for classification.

In Li et al.[25], the authors create a graph based formulation for activity recognition by using image features and relations between objects. They generate a graph with nodes denoting verbs or roles/nouns, with their edges showing relation between the roles and these verbs. They use recurrent networks on this graph based structures with each layer of the graph predicting the roles/verbs in the next layer with the help of recurrent networks. The final understanding of the image is the root of the graph in the form of a text and is the understanding of the complete image. Though similar to our approach in terms of creating relations, it generates classification for the whole scene in the form of a textual output.

2.5 Graph Neural Networks

Scene understanding has been traditionally formulated as a graph problem such as [26, 27]. Graphs have been extensively used for scene understanding, and most of them were solved using belief propagation [28], [29]. Though a clear notion of graphical structure was present, did not include learning based approaches. With the success of deep learning approaches into structured data prediction, problems of scene understanding like semantic segmentation and motion segmentation have been successful. There has been relatively sparse literature on scene understanding in the non-euclidean domains like video and behaviour understanding.

Recently there have been multiple successful attempts at modeling learning algorithms for graph data and specifically, [1, 30, 31, 32, 33] have proposed algorithms to generalize deep learning to unstructured data. With the success of these methods on the graph classification task, multiple recent works have extended the methods to address multiple 3D problems like Shape segmentation [34], 3D correspondence [35], CNN on surfaces [36].

Classification in terms of activity and ego-centric vehicles have become popular with the advent of GCNs [1]. Wu et al.[37], close to our approach deals with classifying action as a whole and individually but leverages image features and only deals with classifying activity recognition for humans. It generates graphs individually for each object and assumes other objects as neighbors in it. For an example video sequence, the authors have tracked humans across frames and generated graphs for each detected human. We handle multiple objects by incorporating relations between them rather than creating graphs for each object, which is a better and clean formulation.

Closest to our approach using GCNs is Li et al.[38], which classifies only ego vehicle behaviour, unlike our method that classifies all objects in the scene. While [37] tracks objects through the frames, [38] only detects objects independently across frames. Reason for such independent detection is their task of ego-vehicle classification. In their graphs, ego-vehicle is the only consistent node through all the frames. In their method, the authors have created individual graphs at each time step with the originally detected objects. Based on this information, they pass the graphs through a GCN and LSTM for classification of the ego-vehicle behaviour. This method is the only other method that has used GCNs, LSTMs and static landmarks as similar to our approach. [39] (STAG network) similar to [38] in terms of extracting features detects collision in a given video sequence. It extracts image features using object detection techniques and generates relational and independent features similar to [38]. It rather uses local and non-local operators[40] for detection whereas Li et al.[38] generates independent graphs and leverages GCNs for detection.

In the Computer Vision community, event recognition for stationary surveillance video is well researched over standard datasets [41], [42]. In the robotics community, motion detection methods such as [43, 44, 45, 46, 47] segment objects through frames using CNN based features into categories and are not concerned with classifying object behaviour. While [48, 49] have included semantics along with motion bear some resemblance, it aims to segment vehicles and generate a 2D mesh around the objects. The proposed methods go beyond the traditional task of motion segmentation and recognize complex behaviours of objects.

2.6 Summary

In this chapter, we have listed out the existing work in the field of vision and robotics that have some similarities with our work and have shown the difference between their and our approaches. We provide quantitative results and show that our formulation is effective in the chapters 3, 4 along with few other works in this chapter.

Chapter 3

Spatio-Temporal MRGCN

Given a video sequence, we aim to classify all vehicles in the scene into one of the following behaviours: {Moving Away from Us, Moving Towards us, Lane-Change left-right, Lane Change right-left, Overtake}. In this chapter, we provide our architecture along with experimental results on multiple datasets for our task.

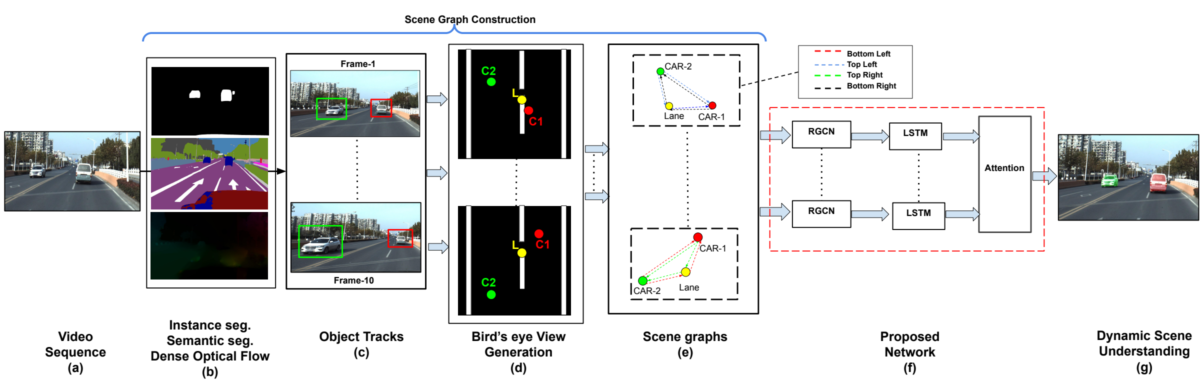
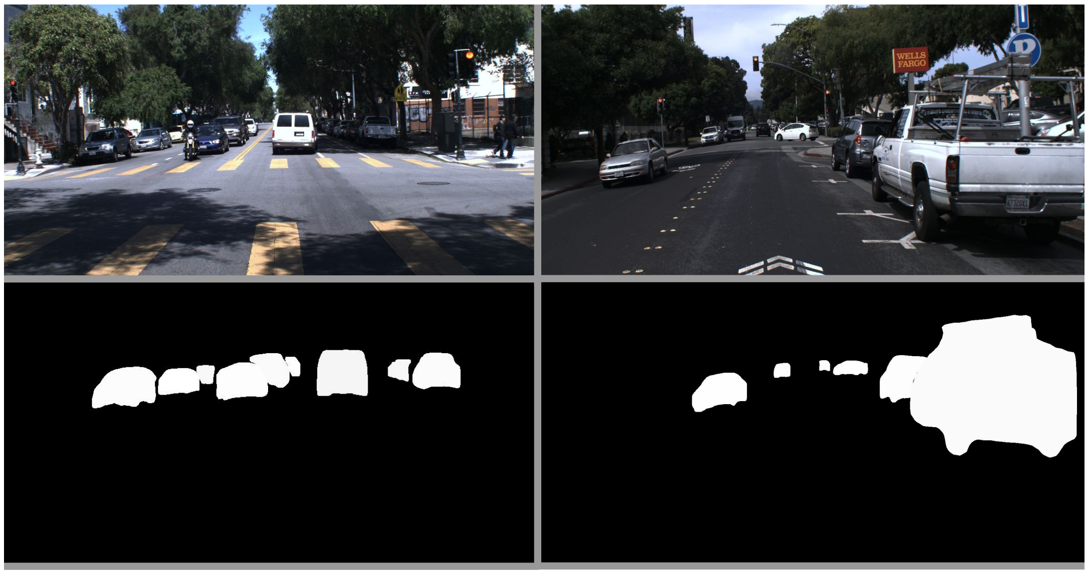


Figure 3.1 This illustrates the overall pipeline of our framework. The input (a) of the system consists of consecutive frames of a monocular video from cameras mounted on the cars. We use traditional object tracking pipelines, as shown in (b) to extract features. These extracted features/tracklets are then transferred into the bird's-eye view as in (d). Using these pre-processed features in bird's-eye view, we generate spatial scene graphs as illustrated with examples in (e). MR-GCNs are used to encode spatial-relations between objects from this scene graph. Subsequently, this spatial information from MR-GCNs over different frames are passed to an LSTM to model the temporal evolution of relations and predict the different vehicle behaviours as in (g). The two cars in (g) in Red and Green depict *Moving Away from Us* and *Moving towards Us* respectively.

3.1 Overview

From given video frames, tracks for all objects are obtained using pre-processing methods (3.2) in image space, which are used as input to the *Bird's eye View Generation*(3.2.2) that generates tracks in 3D representation. These tracks are used for generation of Scene graphs in 3.3. Generated graphs are processed by our network for predicting vehicle behaviours. The overall pipeline can be seen in Fig.3.1



(a) (b)

Figure 3.2 Instance segmentation. Top row is the original image. Second row shows detected vehicles.

3.2 Pre-Processing

For a complete scene understanding, we require non-static objects as well as static landmarks. While it is naive and obvious to model only vehicles in a video for scene understanding, we found that additionally modeling lane markings as landmarks can facilitate in obtaining rich relational information about the vehicles.

This includes detection of vehicles and landmarks (lane markings) and tracking them through a window of T frames. These are general heuristics and we have used existing state of the art detection, tracking techniques for our purpose. At the end of the pre-processing we obtain individual tracks for each object through T time-steps.

3.2.1 Vehicle and Landmark Detection

The three major components required for dynamic feature extraction are **instance segmentation** [50], **semantic segmentation** [51] and **per-pixel optical flow** [52] in a video frame. *Mask-RCNN* [50] is used to compute instance segmentation of moving objects in the scene. An example of instance segmentation can be seen in Fig.3.2.

In case of static objects like poles and lane markings, which act as landmarks, we follow the method of *Inplace-ABN* [51] and compute the semantic understating of the scene. Few images from our datasets are depicted in Fig.3.3. The corresponding lane detections can be seen in the image in *white* on the road. The optical flow is computed between two consecutive frames in the video using *python dense optical*

Figure 3.3 Semantic segmentation. Top row is the original image, bottom contains the segmented output.

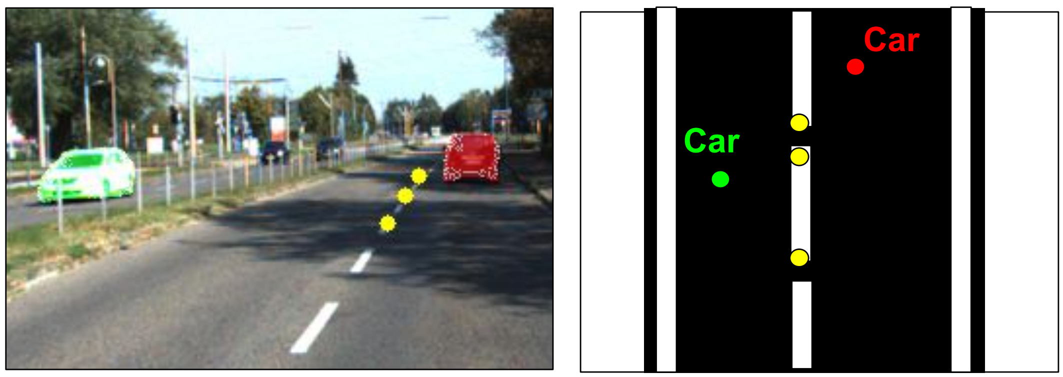
flow [52]. Each of the objects in the scene is tracked by combining optical flow information with features from the instance and semantic segmentation.

The obtained tracks for cars $C_T \in N_1 * T * d$ and $L_T \in N_2 * T * d$ where N_1 and N_2 denotes the number of tracked vehicles and lane-markings respectively for T frames and d denotes the dimension of image-co-ordinates, which is 2. All the tracked objects are in image space.

3.2.2 Bird's Eye-View Generation

As the tracks for objects, static and non-static are in image space, we found that moving those into 3D space would better capture the relative positioning of different objects. Our aim is to create a graph with all objects on a single plane similar to the one shown in Fig.3.4.

For the conversion from image to bird's-eye view/ Top-View, we employ Eqn.3.1 as described in [53] for object localization in bird's-eye view. [53] requires only camera height and instrinsic matrix of the camera (K) that are simpler to obtain rather than general techniques like LIDAR for depth. For the creation of a graph, we assume objects to be on a 2D planar surface. In our case, since camera is mounted on top of the ego-centric car, the road ahead in the image is the planar surface. For this, all objects in image space are assigned a reference point. For lanes, it is the center point of lane marking, for vehicles it is the center of the bounding box that intersects with the road. An example of a top-view can be seen in Fig.3.4(b).



(a) Image (b) Bird's Eye View

Figure 3.4 Bird's Eye-View Example. (a) shows detected cars in red and green and lane markings in yellow. (b) shows corresponding Bird's eye view we want to achieve with objects as a single point on the road.

Let $b = (x, y, 1)$ be the reference point in homogeneous coordinates of the image. We transfer these homogeneous co-ordinates b to a Bird's-eye view as follows.

$$B = (B_x, B_y, B_z) = \frac{-hK^{-1}b}{\eta^T K^{-1}b} \tag{3.1}$$

Here K is the camera intrinsic calibration matrix, η is the surface normal, and h is the camera height from the ground plane and B is the 3D co-ordinate (in the bird's-eye view) of the point b in the image. The visualization of the above formulation can be seen in Fig.3.5.

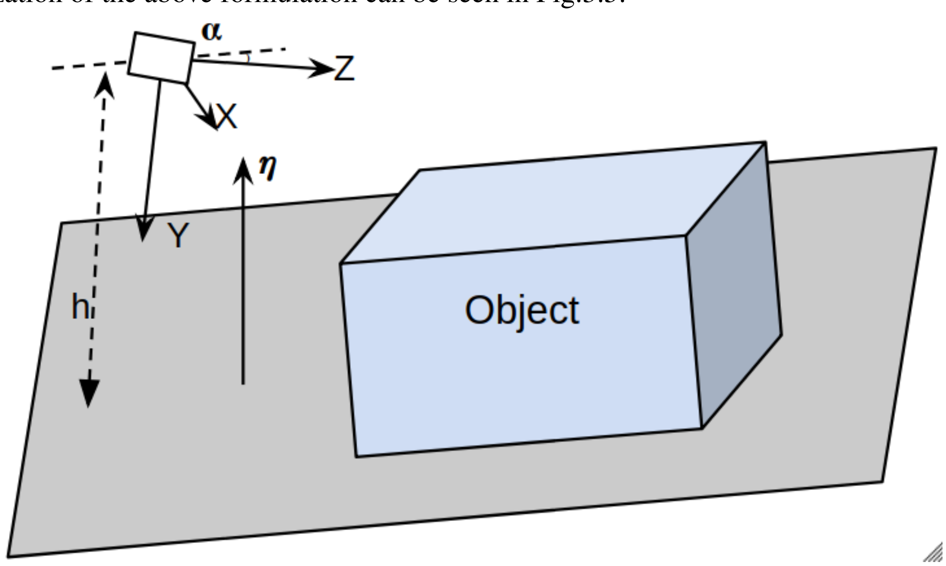


Figure 3.5 Bird's Eye View generic case. Generic case of a camera capturing images. The camera is tilted at an angle α, at a height h from ground. η denotes the normal to the ground surface.

Fig.3.5 shows a generic case of an object on a surace and a camera on top that is tilted with respect to the road at an angle α. h, η are height of camera from ground and normal to the surface of the road respectively as explained earlier.

3.3 Scene Graph Generation

We denote *Spatial Scene graph* as the graph generated from the above obtained 3D tracks per time-step t. The spatial information in a video frame at time t captured as a scene graph S^t, is based on the relative position of different objects in the bird's-eye view. The spatial relation, $S^t_{i,j}$ between a subject, i and an object, j at time, t is the quadrant in which the object lies with respect to the subject, i.e., $S^t_{i,j} \in \mathcal{R}_s$, where $\mathcal{R}_s = \{$top left, top right, bottom left and bottom right$\}$. An example of a spatial scene graph and Spatial-relations is illustrated in Fig.3.6. In the figure, (a) denotes the image frames, (b) shows the bird's eye-view and (c), the generated scene graphs. In (a), the two cars in red, orange in *frame-1* are behind the lane in yellow and in the *final frame* move ahead of the lane marking. The relational change between car (orange) and lane (yellow) can be observed by looking at the change in color of the edges between them in the graphs.

Note that at each time step t, an independent graph is generated while the number of nodes (or objects tracked) is constant in all T graphs.

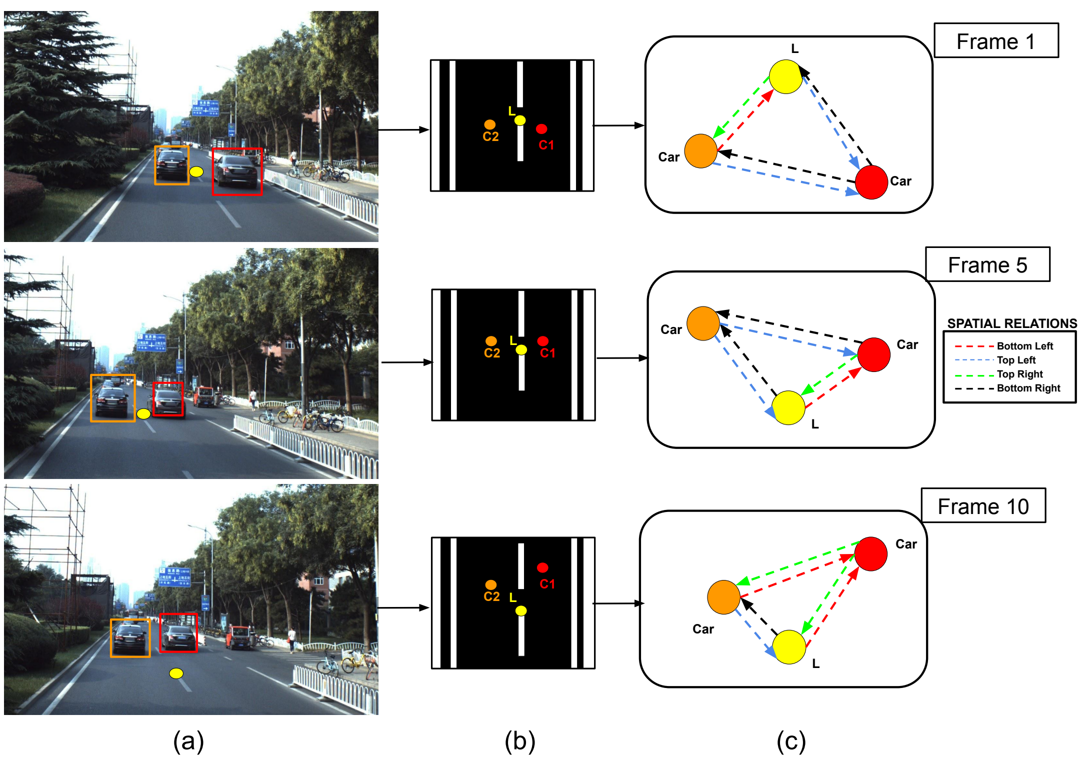


Figure 3.6 Scene Graph generation. In the figure, (a) denotes Image frames at t=1, t=5 and t=10 with detected objects (vehicles and lane-markings). (b) shows Bird's Eye View for each time-step. Generated scene graphs at each time step are shown in (c). The colored edges in the graphs denote different relations between objects as described in the legend.

3.4 Proposed Architecture

We obtain Spatio-temporal representations of vehicles i.e., relations between objects and their evolution with time into a representation. This is done by modeling the temporal dynamics of their spatial relations with other vehicles and landmarks over time. The Spatio-temporal representation is obtained by stacking three different neural network layers. Each network satisfies it's own purpose in predicting the behaviours. First, we use a Multi-Relational Graph Convolutional Network (MR-GCN) [2] layer to encode the spatial information of each object (node in graph) with respect to its surrounding objects at time t into an embedding E^t.

Then, for all nodes, it's spatial embedding from all time steps, $\{E^1, E^2, ..., E^T\}$ is fed into a Long Short-term Memory layer (LSTM) [3] to encode the temporal evolution of the embeddings. The Spatio-temporal embeddings, $\{C^1, C^2, ..., C^T\}$ obtained at each timestep from the LSTM is then passed through a (multi-head) self-attention layer to select relevant Spatio-temporal information necessary to make behaviour predictions. The details of the different stages in the pipeline follow in order below. The overall pipeline is best illustrated in Fig.3.1.

3.4.1 MR-GCN

The first part of the architecure is the Multi-relational Graph convolutional Networks (MRGCN) [2], originally proposed for knowledge graphs that handles multi-relational interactions between objects. The reason for choosing MRGCN is due to it's ability to handle multiple relations unlike the GCNs [1] which assume unique nature of edges between all nodes. We generalize graph convolution first, and then apply it for individual Scene graphs independently.

General Formulation of MRGCN: MRGCNs, similar to GCNs, perform graph-convolution for each relation r independently. Finally, aggregate information from all relations for each node, which is output of a layer of the MRGCN.

Given any Multi-Relational Graph G with set of relations R, $G = (V, S)$ with vertex set V and Edge set S, where $S_{i,j} \in R$ is an edge between node i and j. Feature for node i under relation r in l^{th} layer of MRGCN is defined as follows:

$$h_r^l[i] = \sum_{j \in \mathcal{N}_r[i]} \frac{1}{c_r[i]} W_r^l h^{l-1}[j] \tag{3.2}$$

where, $\mathcal{N}_r[i]$ denotes set of neighbour nodes for v_i under relation r, $\mathcal{N}_r[i] = \{j \in V \mid S_{j,i} = r\}$ and $c_r[i] = |\mathcal{N}_r[i]|$ is a normalization factor. Here $W_r^l \in \mathcal{R}^{d' * d}$ is the weights associated with relation r in the l^{th} layer of MR-GCN; d', d are dimensions of $l - 1$ and l th layers of MRGCN.

Neighborhood information aggregated from all the relations are then combined by a simple summation to obtain the node representation as follows:

$$h^l[i] = ReLU(W_s^l h^{l-1}[i] + \sum_{r \in R_d} h_r^l[i]) \tag{3.3}$$

where the first terms correspond to the node information (self-loop) and W_s^l is the weight associated with self-loop.

Applying for each Scene graph: Hence, in our case we apply Eqn.3.2, 3.3 for each scene graph G^t independently where, $G^t = (V, S^t)$ with vertex set V and Edge set S^t where $S^t_{i,j} \in R_s$ is an edge between node i and j. Thus the equations are as below:

$$h^l_{r,t}[i] = \sum_{j \in \mathcal{N}_{r,t}[i]} \frac{1}{c_{r,t}[i]} W^l_r h^{l-1}_t[j] \tag{3.4}$$

where, $\mathcal{N}_{r,t}[i]$ denotes set of neighbour nodes for v_i under relation r in graph G^t ie; $\mathcal{N}_{r,t}[i] = \{j \in V \mid S^t_{j,i} = r\}$ and $c_{r,t}[i] = |\mathcal{N}_{r,t}[i]|$.

$$h^l_t[i] = ReLU(W^l_s h^{l-1}_t[i] + \sum_{r \in R_d} h^l_{r,t}[i]) \tag{3.5}$$

where, $h^l_t[i]$ denotes the output from l^{th} layer of MRGCN for node i (in graph G^t).

To account for the nature of the entity (static or non-static), we learn entity embeddings, $\mathcal{E}_i \in \mathbb{R}^{|O|*d}$ for node v_i where $O = \{Vehicles,\ Lane\ Markings\}$. The input to the first layer of the MRGCN $h^0_t[i]$ is the embedding $\mathcal{E}_i$ based on nature of node i.

The MRGCN layer defined in above equations provides a relational representation for each node in terms of its immediate neighbors. Multi-hop representations for nodes can be obtained by stacking multiple MRGCN layers, i.e., stacking k layers of MRGCN provides a k-hop relational representation, $h^k[i]$. In essence, MRGCN processes the spatial information from each time-step t and outputs **a k-hop spatial embedding**, $E^t = h^k_t$.

3.4.2 LSTM

Once the spatial relations are obtained from the MRGCN, to capture the temporal dynamics of spatial relations between objects, we use LSTM to obtain Spatio-temporal embeddings, C. At each time-step t, the LSTM takes in the current spatial embedding of the objects, E^t and outputs a Spatio-temporal embedding of the objects, C^t conditioned on the states describing the Spatio-temporal evolution information obtained until the previous time-step as defined in Eqn.3.6.

$$C^t, H^t = LSTM(E^t, C^{t-1}), where \tag{3.6}$$
$$C^t = f^t * C^{t-1} + i^t * \tilde{C}^t, \tag{3.7}$$
$$H^t = o^t * tanh(C^t) \tag{3.8}$$
$$where, i^t = \sigma(w_i[H^{t-1}, E^t] + b_i) \tag{3.9}$$
$$f^t = \sigma(w_f[H^{t-1}, E^t] + b_f) \tag{3.10}$$
$$o^t = \sigma(w_o[H^{t-1}, E^t] + b_o) \tag{3.11}$$
$$\tilde{C}^t = tanh(w_c[H^{t-1}, E^t] + b_c) \tag{3.12}$$

where, i^t, f^t, o^t represent input, forget and output gates of the LSTM respectively. w_c, w_i, w_f and w_o denoting the weights and b_c, b_i, b_f and b_o being the bias for their respective gates. H^{t-1} is the hidden state of LSTM in time-step $t-1$.

3.4.3 Attention

The third component in the set of networks is the attention. We observe that capturing a change in relation is the core step. A change in relation between two objects could occur in any time-step $t \in \{0, 1, ..., T\}$. Hence, we improve the temporal evolution of spatial relations by weighting them. Specifically, here we use Multi-Head Self-Attention described in [5] on the LSTM embeddings. Attention helps us weight the LSTM embeddings by giving more weight to those capturing important information removing redundancy.

The Self-Attention layer, originally proposed in the Natural language processing (NLP) literature, obtains a contextual representation of a data at a time, t in terms of data from other time-steps. The current data is referred to as the query, Q, and the data at other time-steps is called value, V, and is associated with a key, K. All three matrices Q, K and V are learned parameters of same dimension. Q*K generates scores for each time-step w.r.t the other time-steps depicting their correlation, and multiplying finally with V gives the weighted embeddings. Overall, it gives weight of each time-step with other time-steps. Attention function outputs a weighted sum of value based on the context as defined in Eqn.3.13.

$$Attention_m(\,Q,\, K,\, V\,) = softmax(\frac{Q\,K^T}{\sqrt{d_k}})V \tag{3.13}$$

where, d_k is the dimension of the key vectors. Herein our task, at every time-step, the query is based on the current LSTM embedding, and the key and value are representations based on the LSTM embeddings from other time-steps. For a particular time-step t, the attention function looks out for temporal correlations between the LSTM embeddings at the current time-step and the rest. *Softmax* helps in determining the weight that needs to be given to the LSTM embeddings and V captures information from all time steps. Hence, this weighting from the *softmax* function helps in determining important

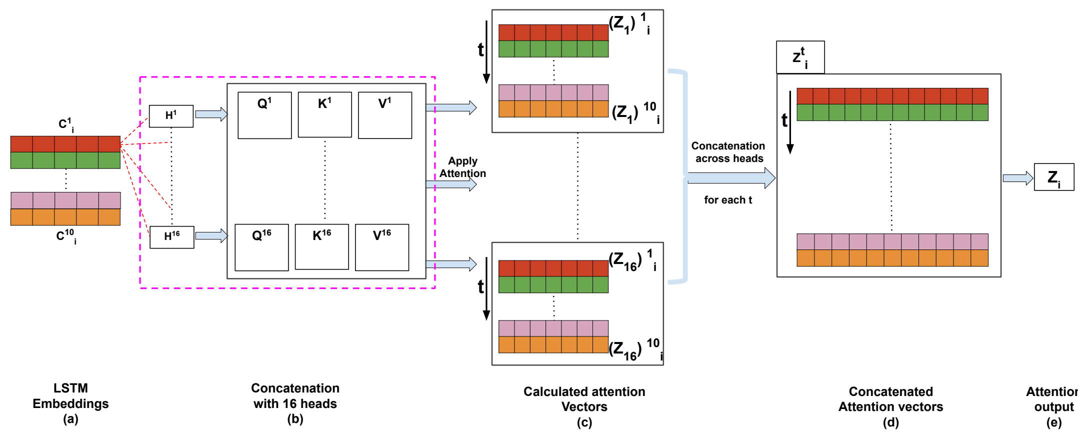


Figure 3.7 Multi-Head attention Explanation. Attention is applied for each node independently across time-steps. In the figure, (a) denotes the obtained LSTM embeddings. These are passed through self-attention layers (b), which generate attention vectors Z as shown in (c). Each box in (c) denotes output from a single head, and each head has 10 vectors for 10 time-steps. These are concatenated across heads as shown in (d). t=1 is colored in red, t=2 in green and t=10 in orange. In (d), we see concatenated attention vectors with the same color-coding depicting concatenation is done among vectors of equal time-step. These are back projected to required dimension to obtain (e).

information. High scores are given to those time-steps with relevant information. Herein, we use the multi-head attention, which aggregates (concat) information from multiple attention functions, each of which may potentially capture different contextualized temporal correlations. A multi-head attention with M heads is defined below:

$$Z^t = \mathop{\Big\|}_{m=1}^{m=M} Attention_m(C^t\,W_m^Q, C^t\,W_m^K, C^t\,W_m^V) \tag{3.14}$$

where $\|$ represents concatenation, $W_m^Q \in \mathbb{R}^{d\times d_k}$, $W_m^K \in \mathbb{R}^{d\times d_k}$ and $W_m^V \in \mathbb{R}^{d\times d_v}$ are parameter projection matrices for m^{th} attention head. $Z^t \in \mathbb{R}^{n\times Md_v}$ is the concatenated attention vectors for t^{th} time-step from M heads .

A clear understanding as to how the attention function is applied in our case across time-steps is depicted in Fig.3.7. Fig.3.7(a) denotes the T embeddings obtained for each time-step from LSTM. These are passed through self-attention (b) to generate attention vectors (c). Attention vectors from each time-step are concatenated across heads to generate Z^t, shown in (d). These are projected back to same dimension as that of (a) to form (e).

Information from all the time-steps, Z, are average pooled along the time-axis and label predictions are made with them. All the three components in the behaviour prediction stage, are trained end-end by minimizing a cross-entropy loss defined over the true labels, Y and the predicted labels, $\hat{Y}$. The final prediction components are described below.

$$\begin{aligned} &U = Pool_{AVG}(Z) \\ &\hat{Y} = UW_l \\ &min\;\; CrossEntropyLoss\,(Y, \hat{Y}) \end{aligned} \tag{3.15}$$

where, W_l projects U to number of classes.

3.5 Experiments and Analysis

We evaluate our vehicle behaviour understating framework on multiple publicly available datasets captured from moving cameras.

3.5.1 Datasets

Our Method takes in monocular videos to understand the vehicle behaviour. We evaluate our framework on three datasets, two publicly available datasets namely, KITTI [54] and Apollo-scapes [55] and our proprietary dataset, Indian Dataset which is captured in a more challenging cluttered environment. For all the three datasets we collected human annotated labels for the vehicle maneuvers.

3.5.1.1 Apollo-Scape Dataset

Apollo-scape harbors a wide range of driving scenarios/maneuvers such as *overtaking*, *Lane change*. It is a dataset by the Baidu team from China. The dataset includes mainly wide 4-lane roads in cities as well as main highways with 4-lane roads. All the images are recorded from a camera mounted on top of an SUV. We selected this dataset as our primary dataset because it contains considerable instances of complex behaviours such as *overtaking*, *Lane change* which are negligible in the other datasets.

3.5.1.2 KITTI Dataset

This dataset is constrained to a city and regions around the city along with few highways. We pick sequences in *tracking* dataset, 4, 5, 10 for our purpose. We have asked manual annotators to label the behaviour of the car by looking at the video at each frame of the video sequence. Though KITTI does not have too many sequences as compared to Apollo-scape, KITTI consists of wider roads with higher visibility, better image quality and decent traffic conditions unlike slightly occluded and congested ones as in Apollo-scapes.

3.5.1.3 Indian Dataset

Although all the datasets specified above address most of the problems for autonomous navigation, they lack in terms of variations of the vehicles on the road. To account and infer the behaviour of the objects with high variation in object types, we capture a dataset in the Indian driving conditions. This dataset contains Non-Standard vehicles such as Buses, petrol tankers and mini-vans which are not previously modeled by the framework. This dataset helps us show that our framework is independent of the vehicle nature.

GT \ Predicted	MVA	MTU	PRK	LCL	LCR	OVT	total
MVA	**691**	1	13	16	4	83	**808**
MTU	1	**212**	21	0	2	0	**236**
PRK	8	39	**1330**	19	7	10	**1413**
LCL	10	0	6	**137**	0	8	**161**
LCR	6	0	5	3	**112**	3	**129**
OVT	18	0	1	0	0	**53**	**72**

Table 3.1 Confusion matrix for model trained on Apollo dataset. We have used abbreviations for label names, MVA : moving away from us, MTU: moving towards us, PRK: parked, LCL: lane change left to right, LCR: lane change right to left, OVT: overtaking and GT : Ground truth.

3.5.2 Quantitative Results

We show that our framework is capable of learning complex behaviours such as overtake and lane changes. We provide multiple experimental values on all the datasets along with transfer learning results. In-depth analysis of our model's performance is available in the form of confusion matrix, ref Table. 3.1. The values in Table. 3.1 are count of the instances for each class, with columns being predicted count and rows showing the actual class. Values provided as *accuracies* in any table are *recall* values as we are more concerned about how good our model performs on a particular class independently. For the sake of simpler and cleaner figures and tables, we have used abbreviations for label-names. MVA : moving away from us, MTU: moving towards us, PRK: parked, LCL: lane change left to right, LCR: lane change right to left, OVT: overtaking in both the tables 3.1, 3.2.

3.5.2.1 Ablation Study on Model

To study the importance of each component of our system, we perform an ablation study on the model. We observe the importance of encoding spatial information along with temporal aspects by comparing variations of our network with each other, as described in table 3.2. First, we use MR-GCN followed by a simple LSTM (G + L), which provides below-par results but performs comparatively better than MR-GCN followed by a single-head attention (G + SA). Adding a single head Attention to MR-GCN and LSTM (G + L + SA), helps to weight the Spatio-temporal encodings as described in 3.4.3 providing a huge improvement in accuracies. The final model with multi-head attention (G+L+MA) attains improvement, specially for $overtake$ showing it's complex behaviour. Overall, each component in the network provides a meaningful representation based on the information encoded in it.

Classes	MVA	MTU	PRK	LCL	LCR	OVT
Architecture						
G + L + MA	**85**	**89**	**94**	**84**	**86**	**72**
G + L + SA	84	75	95	51	65	51
G + L	78	72	74	37	49	41
G + SA	70	18	54	38	14	40
L	37	35	34	6	24	15
L + MA	56	41	46	8	27	13

Table 3.2 Baselines and model variations, trained on Apollo. Values provided are accuracies. Abbreviations for model architectures (rows) are, G : MR-GCN, L : LSTM, MA : Multi-Head Attention, SA : Single-Head Attention. Nomenclature in the architecture names reflect the components in it. Column abbreviations are MVA : moving away from us, MTU: moving towards us, PRK: parked, LCL: lane change left to right, LCR: lane change right to left and OVT: overtaking

3.5.2.2 Baselines Comparison

We observe the importance of spatial information encoded as a graph by comparing the performance of our method (G+L+MA) against a simple LSTM (L) and a LSTM followed by Attention (L + MA) models, which encode relational information with a 3D location based positional features; ref Table. 3.2. To train these models (L and L + MA), we give 3D locations of objects obtained from the bird's eye View (3.2.2) as a direct input. For each object in the video, we create a feature vector consisting of it's distance and angle with all other nodes for T time-steps. The distance is a simple Euclidean distance. To account for the feature of the object (lane-markings/Vehicles), we create a 2 dimensional one-hot vector $\{1,0\}$ representing Vehicles and $\{0,1\}$ representing static objects. To create a feature vector for i^{th} object, we find distances and angles with all other nodes for T time-steps in the scene. In both the cases, the above features are fed to the LSTM at each time-step. We pool the output from LSTM for L and output from Attention layer for L + MA, along time dimension and project to number of classes by using a dense layer.

Table.3.2 shows clear improvement in accuracy for models having a MR-GCN to encode the spatial information when compared to the simple recurrent models (L and L + MA). Our method outperforms the baselines by a good margin for all classes.

3.5.2.3 Transfer Learning

As embeddings obtained from MR-GCN are agnostic to the visual scene, they are dependent on the scene graph obtained from the visual data (images). To show this, we trained the model only on Apollo dataset and tested it on Indian and KITTI. While testing, we removed the lane change and overtaking behaviour as it is not present in KITTI and Indian dataset. In Table.3.3 we observe that results are better with model trained on Apollo dataset as compared to models trained on their respective datasets, this could be attributed to the presence of more complex behaviours (overtaking, lane change) in Apollo

Trained on	Apollo		KITTI	Indian
Tested on	KITTI	Indian	KITTI	Indian
Moving away from us	99	99	85	85
Moving towards us	98	93	86	74
Parked	99	99	89	84

Table 3.3 Transfer learning. All values are accuracies. The 2 columns under Apollo denote the model trained on APOLLO and tested on KITTI and Indian datasets. Last two columns are results with models trained on their respective datasets; i.e., trained on KITTI and tested on KITTI, trained on Indian and tested on Indian data.

dataset and shear difference in the size of datasets (Apollo has 4K frames as compared to 651 and 722 frames as in Indian and KITTI datasets).

All the values in the table are accuracies. The first column is the classes, second, third being the results of testing on KITTI and Indian with the model trained on Apollo-scape. The last two columns are also tested on KITTI and Indian but with the model trained on the same datasets respectively.

3.5.2.4 Ablation Study on LandMarks

Landmarks are the stationary points (lane marking, poles, etc.), which along with other vehicles, are used by MR-GCN to encode a spatially-aware embedding for a node. We carry out an ablation study to understand the effects of how the number of landmarks available in sequence affects the performance of our pipeline. We train three models, one with 50 percent landmarks available, another with 75 percent available and the last one with all the landmarks available. The results mentioned in above experiments are tabulated in Table. 3.4 under their respective rows *ours (50%)*, *ours (75%)*, *ours (full)*. Behaviours which are directly dependent on landmarks (lane change, moving ahead and back) show a difference of atleast 10% when comparing the full model and the model trained with 75% landmarks and a difference of atleast 15% when compared with a model trained with 50% landmarks.

Tested on	Apollo-Scape		
Method	ours (50%)	ours (75%)	ours (full)
Moving away	72	76	**85.3**
Moving towards us	67	75	**89.5**
Parked	97	97	**94.8**
LC left - right	66	74	**84.1**
LC right - left	71	72	**86.4**
Overtaking	68	65	**72.3**

Table 3.4 Ablation study on landmarks. Comparison between models trained with 50% of landmarks and 75% and 100% of landmarks. Values provided are accuracies.

The observed results make it evident that the performance improvement is achievable with the utilization of more information in the form of increased landmarks.

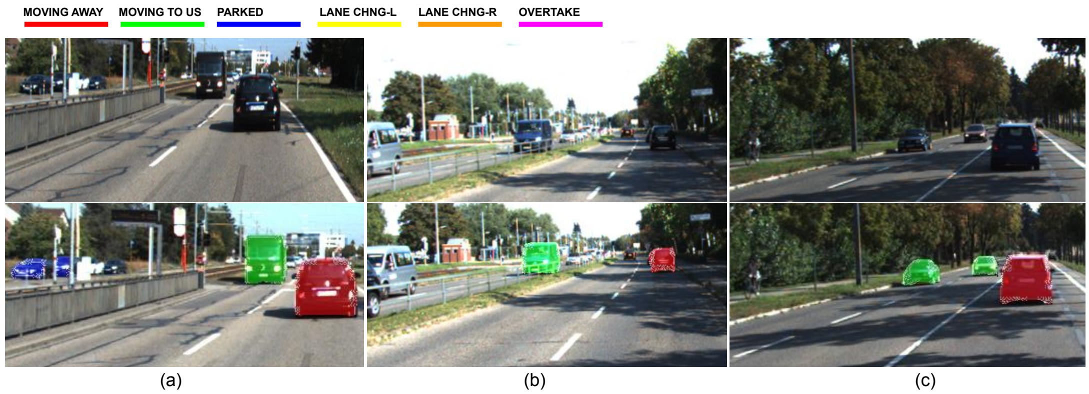


Figure 3.8 Results on KITTI dataset. Accurate classification of Vehicle behaviour can be seen based on the color over the vehicle. Red denotes *Moving Away*, Green denotes *Moving towards Us*

3.5.3 Implementation Details

All the graphs are constructed for a length of $T = 10$ time-steps on all datasets. Each of our dataset is split into train and test as 70 and 30 percentage respectively. We empirically found that using two layers for MR-GCN provides us with the best results, with first and second layers being 128, 32 respectively. Input and Output dimensions of LSTM are both equal to 32. In the Multi-Head Attention layer, we choose 16 heads and 1024 as the dimension of the feed-forward network. To train on Indian and KITTI datasets, we use a smaller version of the network with the dimension of first and second layers being 64 and 32. The LSTM dimensions are also reduced to 16 in case of Indian and KITTI. The output dimension of Attention is 32, equal to input for Apollo while is 16 for other two datasets. We pool the output from the attention layer and apply a dense layer to project to number of classes. Single head and multi-head and no attention almost have similar run time. All models are trained and tested on a single NVidia Geforce Gtx 1080 gpu.

3.5.4 Qualitative Analysis

The following color coding is employed across all qualitative results. Blue denotes vehicles that are *parked*, red is indicative of vehicles that are *moving away*, and green the vehicles that are *moving towards us*. Yellow indicates *lane changing: left to right*, orange indicates *lane changing: right to left* behaviour with magenta denoting *overtaking*. Complex behaviours such as overtaking and changing lanes can be observed in Fig.3.10, which shows qualitative results obtained on Apollo dataset. Fig.3.10 (a) depicts a car (right most car) in the midst of overtaking (seen with magenta color). Fig.3.10 (b) and 3.10 (c) present us with cases of lane change. In Fig.3.10 (b) we see a van changing lane, along with a moving car, In Fig.3.10 (c) we observe a bus changing lane from left to right along with two cars parked on the right side.

Object class agnosticism can be observed in Fig.3.9 (Indian dataset), where behaviour of several non-standard vehicles are accurately predicted. In Fig.3.9 (a) we can see a car on the left side of the road

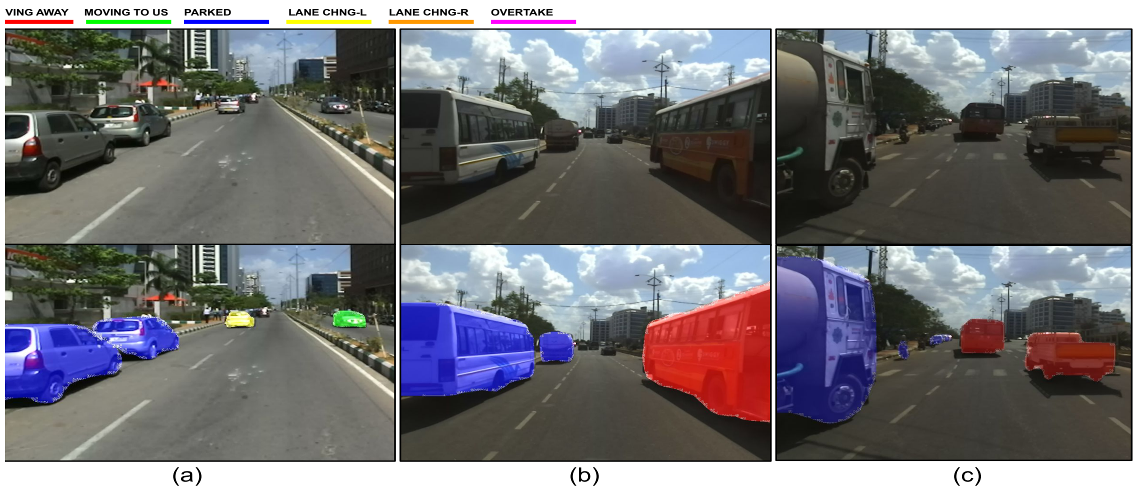


Figure 3.9 Results on Indian dataset. We can see Vehicle class agnosticism of our Pipeline as non standard Vehicles such as bus and oil-tanker are correctly classified.

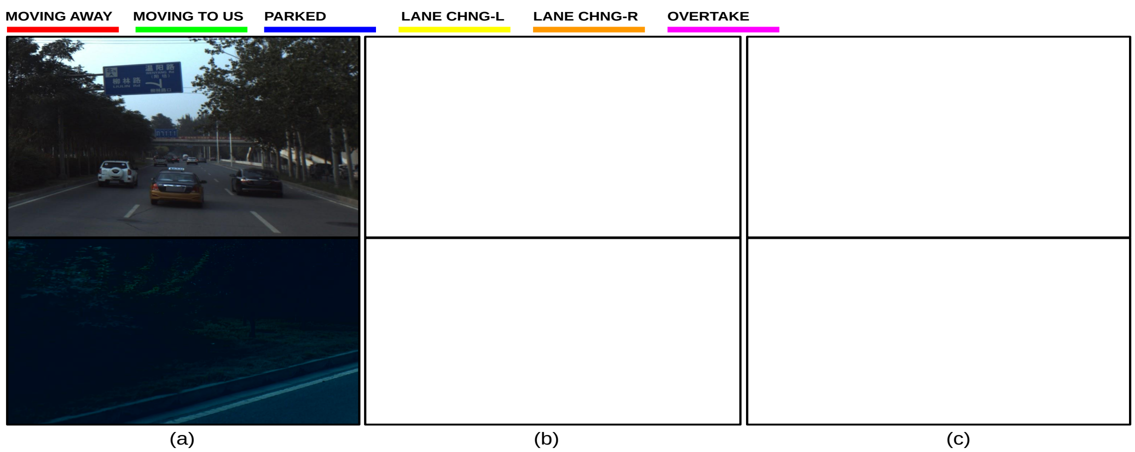


Figure 3.10 Results on Apollo Dataset. Figures depict complex Vehicles maneuvers such as overtaking, which can be seen in (a), the car on the right represented in magenta color and lane change which can be seen in image (b) and (c) with yellow on the Apollo dataset.

(predicted *changing lane*) coming onto the road along with parked cars and one car coming towards us. In Fig.3.9 (b) truck and bus are *parked* on left side of the road while another bus is accurately classified as *moving forward* and not *overtaking*. Note that behaviour is classified as overtaking only if a moving vehicle overtakes (moves ahead of) another moving Vehicle. Fig.3.9 (c) provides a case with a semi-truck and bus moving away from us, along with a truck parked. Fig.3.8 shows qualitative results obtained on KITTI sequences 4, 5, 10.

3.6 Summary

In this chapter we have shown the complete pipeline of our method for behaviour detection into the following categories : {Moving Away from Us, Moving towards Us, Parked, Lane-Change (left-right), Lane-Change(right-left), Overtake}. Videos are pre-processed to obtain object tracks. These are converted to 3D co-ordinates. Graphs generated using these tracks are passed through our network consisting of MRGCN, LSTM and Attention to predict the final behaviours.

Chapter 5

Conclusions and Future Work

In this chapter we summarize the work done in the thesis and provide possible future directions in behaviour prediction.

As stated, behavioural prediction can be helpful in making decisions for autonomous driving. Current methods such as Li et al [38] consider classification of only ego-centric vehicle and Structural-RNN [17] does not have graphical structure. Our method can be adapted to predict the ego-centric behaviour if the 3D co-ordinate tracks of ego-car are available. We can use SLAM methods like ORB-SLAM [58] to find trajectory of the ego-vehicle which have been extremely accurate on well known datasets like KITTI. In practical applications, we can get it through a GPS to make it real-time and introduce just one more node in our graph.

Apart from detection, we could extend our model to predict trajectories using regression instead of classification. There have been multiple methods (ref Ch.2) that confine to predicting trajectories of the ego-vehicle or predicting the actual 3D co-ordinates of other vehicles. But predicting behavioural change is a much complex task as it requires a longer range of prediction (generally 10 time-steps). This can be possibly done incase the 3D tracks of all objects are available and are accurate upto few mm. Datasets such as **Honda** and **KITTI** have been releasing new data where all vehicles are tracked using LIDAR. In such cases, a simple GCN can be used in our *Spatio-Temporal Graph Modelling* (Ch.3) with the LSTM predicting trajectories for each time step instead of the final time-step. The behaviour can be further estimated with the currently available architectures.

Our model transfers seamlessly across datasets and could be configured to various applications apart from those presented above. Since it is independent of the appearance of the object, it could be used to model any data that involves multiple objects with different nature. Specifically, it can be used for *activity-recognition of humans* or *pose-estimation in case of sports athletes*. In case of pose-estimation, landmarks can be a racket/bat (in the hands of the athlete) and tracked body-joints (of the athlete) can be the non-static objects. Relations between the racket and human joints determine the current pose. This will be a more complex and a challenging task to solve due to numerous possible poses a human can exhibit.

Having proposed both the approaches, we hope our architectures could be helpful in tackling detection and prediction tasks in multiple domains and not just pertaining to robotics and vision.

Chapter 6

Appendix

In this chapter, we provide other experiments and observations to guide the direction of research for further study which helps in reducing repetitive work. Through the course of our experiments, we majorly found two methods that are both effective and intuitive. Hence we present the advantages and disadvantages of both methods in a precise manner.

6.1 Overview

The two main methods are *Gating* and *Temporal Edge* described in sec.6.2 and 6.3. While *Gating* is similar to our approach in Ch.4, where we use attention to weight relations, gating weights neighbors instead. *Tedge* (Temporal Edge) is a special case where we generate graphs at each time step similar to Ch.3. But here we connect edges of a node across time steps. Detailed explanation of each method is provided in further sections.

6.2 Gating

As stated, gating is similar to attention except that it applies weight across neighbours of a node. Advantage here is that we remove major noise inside the RGCN itself and is extremely simple to implement.

Similar to Ch.4, Temporal Interaction Graph $G = (V, E)$ is defined with vertex set V and Edge set E, where $E_{i,j} \in R_d$ is an edge between node i and j.

We define gates for each relation r in layer l based on a gate matrix $W^l_{r,g}$. The value of gate would be in $[0, 1]$ denoting its importance. These gate values are multiplied by the original information thereby weighting it. Each layer consists of its own set of gates allowing weighting the information at multiple levels independently.

$$g^l_{u,v} = Sigmoid\ (h^l[u]\ .\ W^l_{r,g}) \tag{6.1}$$

where, $g^l_{u,v}$ is the gate value between two nodes u and v with edge $E_{v,u} = r$. $W^l_{r,g}$ is the gate-matrix associated with relation r in layer l. $h^l[u]$ is the embedding obtained from MRGCN layer for node u, as defined in Ch.4.

The i^{th} node feature in relation r obtained from a graph convolution over relation, r in l^{th} layer is defined as follows:

$$h^l_r[i] = ReLU(\sum_{j \in \mathcal{N}_r[i]} g^l_{i,j} \frac{1}{c_r[i]} W^l_r h^{l-1}[j]) \tag{6.2}$$

where, $g^l_{i,j}$ is the gating value between node i and j, $\mathcal{N}_r[i]$ denotes set of neighbour nodes for v_i under relation r, $\mathcal{N}_r[i] = \{j \in V \mid E_{j,i} = r\}$ and $c_r[i] = |\mathcal{N}_r[i]|$ is a normalization factor. Here $W^l_r \in \mathcal{R}^{d' * d}$ is the weights associated with relation r in the l^{th} layer of MR-GCN; d', d are dimensions of $l-1$ th and l th layers of MRGCN.

Neighborhood information aggregated from all the relations are then combined by a simple summation to obtain the node representation as follows:

$$h^l[i] = ReLU(W^l_s h^{l-1}[i] + \sum_{r \in R_d} h^l_r[i]) \tag{6.3}$$

where the first terms correspond to the node information (self-loop) and W_s is the weight associated with self-loop.

An example of how gating is performed can be seen in the Fig.6.1. The information from neighbor nodes is passed through the corresponding gates as shown in (b). This helps in weighting important information and reducing redundancy. Colors of the edges denote edge type and the rectangular boxes denote the gates for each relation.

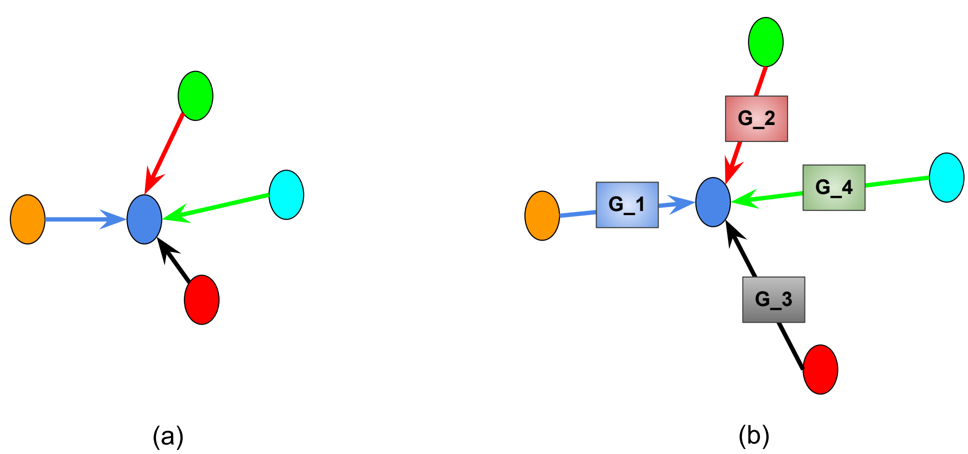


Figure 6.1 Figure shows the Gating mechanism for a single node surrounded by neighbors with different relations between them. (a) shows the various relations between the center blue node and its neighbors. (b) shows the way gating is applied by passing information from neighbors through a GATE (shaded boxes) based on the type of relation. The gating value helps in determining the importance of information obtained from a neighbor node.

6.2.1 Difference between Gating and Attention

Though attention and gating look similar, the weighting approach is different in both. In attention, weighting is applied individually for each node across all relations, $r \in R_d$. For gating, we apply gating mechanism for each node except that a matrix is learned for each relation that helps in weighting *neighbors* of a node. That is, neighbouring node information is aggregated after passing through a *gate* based on their relation, hence the name gating.

We have applied gating for our data and have produced results on Apollo-Scape. Table 6.1 shows a similar performance with respect to our relational-attention method. Though it is on par with our relational attention in other classes, it falls behind by 5% in *Overtake*.

Note : We have tried only few improvements like skip connections for gating method. Other complex mechanisms may yield better results. We leave this for future scope and present only possible directions here.

Method Used	Moving away from us	Moving towards us	Parked	Lane change L-R	Lane change R-L	Over take
Gated-GCN	95	99	97	98	98	84
Rel-Att-GCN	**95**	**99**	**98**	**97**	**96**	**89**

Table 6.1 Comparison is based on Apollo Dataset. Comparison between Gating and Rel-Attention mechanisms. Values are accuracies in percentage.

6.3 Temporal Edge

This method is similar to our MRGCN-LSTM (Ch.3). We generate independent graphs at each time step. But to incorporate temporal information, instead of using a recurrent network, we add a special kind of edge between every two time steps. This is the *temporal edge* as we name it and is the edge between node i_t and i_{t+1}. That is, this edge is between the same node i, at two time steps. Hence a single complete graph for the scene is formed with such connections across time steps as shown in Fig.6.2. This graph is passed through a MRGCN for final prediction.

Similar to Ch.3, we define a Graph $G = (V, E)$ with vertex set V and Edge set E, where $E_{i,j} \in R_s$ is an edge between node i and j. Here, R_s = { *Top left*, *Bottom left*, *Top right*, *Bottom right*, *temporal* }. *temporal* edge is the one that helps in capturing information across time steps. This edge does not have a complementary edge like the { *Top-left* and *Bottom-right* }, { *Top-right* and *Bottom-left* }. Temporal edge exists for every object in the scene (node in graph) originating at time t and ends at time $t + 1$ of the same node.

A simple graph convolution is performed on the obtained graph. The difference is that we have a single graph G containing *spatial graphs* from each time step, which are connected through temporal edges. Graph convolution is performed as described in Eqn.4.1 and 4.2 to obtain final node features

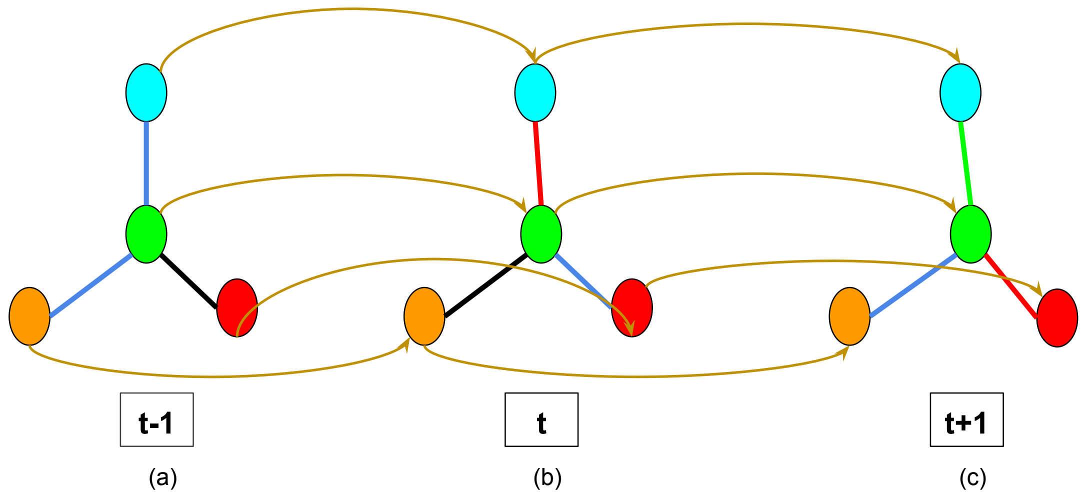


Figure 6.2 Figure shows an example graph used for Temporal Edge Method. (a), (b), (c) show Spatial graphs at time steps t-1, t, t+1 respectively. These graphs hold spatial relations between nodes (blue, red, black and green edges) individually. The golden-yellow edges from t-1 to t and t to t+1 depict the temporal edges. Such a single graph for T time steps is created and passed to the network.

$h^l[i]$ for each node i. Loss is calculated and propagated only through the last time step predictions T as it includes information propagated from all time steps through temporal edges.

Temporal Edge does not work very well in our case, i.e., it produces sub par results even for simpler classes. We observe this to be caused by our fully connected graphs. As general GCNs are not good at handling noise and this noise is carried all along T time steps making the graph huge and unwanted information flowing across nodes.

Note: Gating was published in the ACL 2019 by *Yale University* [59]. A similar formulation as Temporal Edge was published in AAAI by *Honk kong University* [60]. Further advancements in these methods could be found in the papers.

Related Publications

1. S. Mylavarapu, M. Sandhu, P. Vijayan, K. M. Krishna, B. Ravindran,and A. Namboodiri, "Towards accurate vehicle behaviour classification with multi-relational graph convolutional networks", Intelligent Vehicle Symposium (IV)-2020.

2. S. Mylavarapu, M. Sandhu, P. Vijayan, K. M. Krishna, B. Ravindran,and A. Namboodiri, "Understanding Dynamic Scenes using Graph Convolution Networks", International Conference on Intelligent Robotics and Systems (IROS)-2020.

Bibliography

[1] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016.

[2] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[3] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[4] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[6] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2017.

[7] Andreas Geiger, Martin Lauer, Christian Wojek, Christoph Stiller, and Raquel Urtasun. 3d traffic scene understanding from movable platforms. *TPAMI*, 2014.

[8] Chao-Yeh Chen, Wongun Choi, and Manmohan Chandraker. Atomic scenes for scalable traffic scene recognition in monocular videos. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016.

[9] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. *CoRR*, abs/1803.07549, 2018.

[10] Sayanan Sivaraman and Mohan Manubhai Trivedi. Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1773–1795, 2013.

[11] Andreas Ess, Tobias Mueller, Helmut Grabner, and Luc Van Gool. Segmentation-based urban traffic scene understanding. In *BMVC*, 2009.

[12] Shashank Srikanth, Junaid Ahmed Ansari, Karnik Ram R, Sarthak Sharma, Krishna Murthy J., and Madhava Krishna K. Infer: Intermediate representations for future prediction, 2019.

[13] Jens Schulz, Constantin Hubmann, Julian Löchner, and Darius Burschka. Interaction-aware probabilistic behavior prediction in urban environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3999–4006. IEEE, 2018.

[14] Reza Sabzevari and Davide Scaramuzza. Multi-body motion estimation from monocular vehicle-mounted cameras. *IEEE Transactions on Robotics*, 32(3):638–651, 2016.

[15] Xinli Geng, Huawei Liang, Biao Yu, Pan Zhao, Liuwei He, and Rulin Huang. A scenario-adaptive driving behavior prediction approach to urban autonomous driving. *Applied Sciences*, 7(4):426, 2017.

[16] Ashesh Jain, Hema S Koppula, Bharad Raghavan, Shane Soh, and Ashutosh Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3182–3190, 2015.

[17] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 5308–5317, 2016.

[18] Athma Narayanan, Isht Dwivedi, and Behzad Dariush. Dynamic traffic scene classification with space-time coherence, 2019.

[19] H. Berndt, J. Emmert, and K. Dietmayer. Continuous driver intention recognition with hidden markov models. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 1189–1194, 2008.

[20] P. Kumar, M. Perrollaz, S. Lefèvre, and C. Laugier. Learning-based approach for online lane change intention prediction. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 797–802, 2013.

[21] B. Morris, A. Doshi, and M. Trivedi. Lane change intent prediction for driver assistance: On-road design and evaluation. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 895–901, 2011.

[22] M. M. Trivedi, T. Gandhi, and J. McCall. Looking-in and looking-out of a vehicle: selected investigations in computer vision based enhanced vehicle safety. In *IEEE International Conference on Vehicular Electronics and Safety, 2005.*, pages 29–64, 2005.

[23] Oliver Scheel, Naveen Shankar Nagaraja, Loren Schwarz, Nassir Navab, and Federico Tombari. Attention-based lane change prediction, 2019.

[24] Rohan Chandra, Uttaran Bhattacharya, Trisha Mittal, Xiaoyu Li, Aniket Bera, and Dinesh Manocha. Graphrqi: Classifying driver behaviors using graph spectrums, 2019.

[25] Ruiyu Li, Makarand Tapaswi, Renjie Liao, Jiaya Jia, Raquel Urtasun, and Sanja Fidler. Situation recognition with graph neural networks supplementary material.

[26] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1):67–92, 1973.

[27] David Marr and Herbert Keith Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proc. R. Soc. Lond. B*, 200(1140):269–294, 1978.

[28] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International journal of computer vision*, 61(1):55–79, 2005.

[29] Leonid Sigal, Michael Isard, Horst Haussecker, and Michael J Black. Loose-limbed people: Estimating 3d human pose and motion using non-parametric belief propagation. *International journal of computer vision*, 98(1):15–48, 2012.

[30] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2224–2232. Curran Associates, Inc., 2015.

[31] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

[32] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015.

[33] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016.

[34] Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *CVPR*, pages 6584–6592, 2017.

[35] Or Litany, Tal Remez, Emanuele Rodola, Alex Bronstein, and Michael Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5660–5668. IEEE, 2017.

[36] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Transactions on Graphics (TOG)*, 36(4):71, 2017.

[37] Jianchao Wu, Limin Wang, Li Wang, Jie Guo, and Gangshan Wu. Learning actor relation graphs for group activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9964–9974, 2019.

[38] Chengxi Li, Yue Meng, Stanley H Chan, and Yi-Ting Chen. Learning 3d-aware egocentric spatial-temporal interaction via graph convolutional networks. *arXiv preprint arXiv:1909.09272*, 2019.

[39] Roei Herzig, Elad Levi, Huijuan Xu, Hang Gao, Eli Brosh, Xiaolong Wang, Amir Globerson, and Trevor Darrell. Spatio-temporal action graph networks, 2018.

[40] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks, 2017.

[41] Xiaoyang Wang and Qiang Ji. Video event recognition with deep hierarchical context model. In *CVPR*, 2015.

[42] Anthony Hoogs et. al Sangmin Oh. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR*, 2011.

[43] Tao Chen and Shijian Lu. Object-level motion detection from moving cameras. *IEEE Transactions on Circuits and Systems for Video Technology*, 2016.

[44] Nolang Fanani, Matthias Ochs, Alina Sturck, and Rudolf Mester. Cnn-based multi-frame imo detection from a monocular camera. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 957–964. IEEE, 2018.

[45] Rahul Kumar Namdev, Abhijit Kundu, K Madhava Krishna, and CV Jawahar. Motion segmentation of multiple objects from a freely moving monocular camera. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4092–4099. IEEE, 2012.

[46] René Vidal and Shankar Sastry. Optimal segmentation of dynamic scenes from two perspective views. In *CVPR*, volume 2, 2003.

[47] Abhijit Kundu, K.M. Krishna, and J. Sivaswamy. Moving object detection by multi-view geometric techniques from a single camera mounted robot. In *IROS*, 2009.

[48] N Dinesh Reddy, Prateek Singhal, Visesh Chari, and K Madhava Krishna. Dynamic body vslam with semantic constraints. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1897–1904. IEEE, 2015.

[49] Johan Vertens, Abhinav Valada, and Wolfram Burgard. Smsnet: Semantic motion segmentation using deep convolutional neural networks. In *IROS*, 2017.

[50] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[51] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5639–5647, 2018.

[52] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *CVPR*, 2017.

[53] Shiyu Song and Manmohan Chandraker. Robust scale estimation in real-time monocular SFM for autonomous driving. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 1566–1573. IEEE Computer Society, 2014.

[54] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[55] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 954–960, 2018.

[56] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[57] Abhishek Patil, Srikanth Malla, Haiming Gang, and Yi-Ting Chen. The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. In *International Conference on Robotics and Automation*, 2019.

[58] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[59] Yinchuan Xu and Junlin Yang. Look again at the syntax: Relational graph convolutional network for gendered ambiguous pronoun resolution, 2019.

[60] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition, 2018.