Multi Agent Navigation in Unconstrained Environments using a Centralized Attention based Graphical Neural Network Controller

Yining Ma*[1,2], Qadeer Khan*[1,2] and Daniel Cremers[1,2,3]

***Abstract*— In this work, we propose a learning based neural model that provides both the longitudinal and lateral control commands to simultaneously navigate multiple vehicles. The goal is to ensure that each vehicle reaches a desired target state without colliding with any other vehicle or obstacle in an unconstrained environment. The model utilizes an attention based Graphical Neural Network paradigm that takes into consideration the state of all the surrounding vehicles to make an informed decision. This allows each vehicle to smoothly reach its destination while also evading collision with the other agents. The data and corresponding labels for training such a network is obtained using an optimization based procedure. Experimental results demonstrate that our model is powerful enough to generalize even to situations with more vehicles than in the training data. Our method also outperforms comparable graphical neural network architectures. Project page which includes the code and supplementary information can be found here: https://yininghase.github.io/multi-agent-control/**

I. INTRODUCTION

Data driven approaches to senorimotor control have seen a meteoric growth with the advent of deep learning in the last decade [1], [2], [3], [4]. Powerful neural network architectures can now be trained and deployed in real-time applications [5]. The recent success of deep learning for agent control can primarily be attributed to the following 2 factors:

1) Cheap hardware accelerators that exploit the parallel computations[6], particularly in deep neural architectures [7].
2) The availability of simulation platforms that allow benchmarking and evaluation of various vehicle control algorithms [8], [9].

[10], [11] have used such platforms to evaluate their learning based control algorithms. However, many learning based control approaches have certain limitations of their own:

1) They require collection of tremendous amounts of labeled supervised data for training which in some cases may not even be available. For e.g. recovering a vehicle from driving on a sidewalk.
2) The neural network is trained to control only one vehicle. Moreover, since the sensors are placed on the ego-vehicle, the model has partial observability of the environment. Therefore, traffic rules are used to regulate the flow of vehicles and prevent any untoward incident. Moreover, for the task of autonomous driving, the vehicle is constrained to drive only on the road.

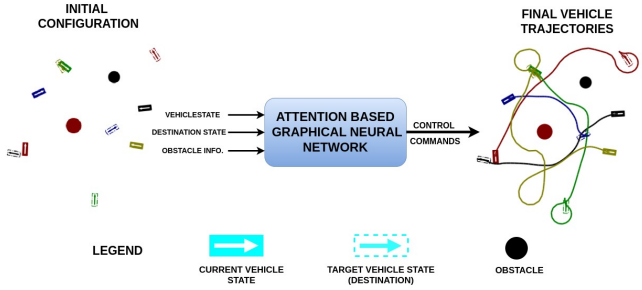


Fig. 1: Overview of multi-agent control: The initial configuration (on the left) shows five vehicles colored **black**, **red**, **green**, **blue** and **olive**. The initial starting state of the vehicles is represented by a rectangle with solid boundaries. The arrow within each rectangle depicts the corresponding orientation of that vehicle. Meanwhile, the rectangles with broken boundaries represents the desired destination/target position of each vehicle. We would like to produce the sequence of control actions such that the five vehicles safely reach their destination state without colliding with each other or the circled obstacle. These control actions are produced by the Attention Based Graphical Neural Network (A-GNN). The A-GNN receives information about the current state and desired destination state of all the five vehicles along with information about any obstacle. The network outputs the control commands for all the five vehicles together. These control commands are executed for all the five vehicles simultaneously. Each vehicle then attains a new state. This new state is then fed again to the A-GNN as the current state to predict the new steering command. This process is iteratively repeated until all the vehicles reach their corresponding destination state. The trajectory traversed by all the vehicles as a result of this iterative process is shown on the right of the figure. Some video examples of our model demonstrating this can be found on the project page: https://yininghase.github.io/multi-agent-control/#Results-of-our-Model

In this work, we present a technique to control not one but a variable number of vehicles in an unconstrained environment having obstacles. The vehicles are meant to reach their desired destination/target state without collision. Supervised labeled data is also not available. Rather, we optimize against a cost function to determine the longitudinal and lateral control labels. This optimization procedure for the task of label

* These authors contributed equally.
[1] Computer Vision Group, School of Computation, Information and Technology, Technical University of Munich. Contact: {yining.ma, qadeer.khan, cremers}@tum.de
[2] Munich Center for Machine Learning (MCML).
[3] University of Oxford.

generation is only done offline and hence does not influence the real-time operation.

Figure 1 describes this process. The Attention based Graphical Neural Network (A-GNN) is a core component which makes these control command predictions. It takes information of all obstacles, the current state of all vehicles and their desired destination/target state. The output of the A-GNN are the control commands for all the vehicles. These control commands are then executed which results in the vehicle achieving a new state. This new state along with the target/desired destination state of the all the vehicles is fed back to the A-GNN to produce the new control commands. The process is iteratively repeated until all vehicles reach their destination state. Note that the A-GNN is trained with labels obtained from offline optimization performed against a cost function.

Note that the A-GNN allows each vehicle to attend to the information of other vehicles. This allows each vehicle to make informed control decisions in order to avoid collisions among themselves while also successfully reaching their destination.

We summarize the contributions of our framework below:

1) Ability to control a variable number of vehicles to reach their desired destination states. We show that our model can even perform inference for more number of vehicles than for which it was initially trained.
2) The architecture of our A-GNN outperforms other comparable attention based GNN layers such as GAIN [12], TransformerConv [13], non-attention based architectures such as EdgeConv [14] and also the naive multi-layer percepton (MLP) method that does not utilize any edges in the graphical structure.
3) Our A-GNN can handle the presence of both dynamic vehicles and static obstacles.
4) We have released the implementation of our framework here: https://github.com/yininghase/multi-agent-control.

An application of this work, could be in large unmanned indoor warehouses where multiple agents are transporting items from one location to another. We assume that all the agents/vehicles move in the x-y plane and can be localized to determine their state. Localization in an indoor environment can be done using for eg. the approaches from [15], [16].

II. Related Work

Multi-Agent Trajectory Prediction & Control: [17], [18], [19] model pedestrian behaviour to predict their future trajectory primarily utilizing information from the past. Our model is rather concerned with control of vehicle agents using current state and destination information. [20], [21] do focus on multiple agents but for the task of leader guided formation control. Meanwhile, [22] investigates connectivity restoration for formation control. In our work, the multiple vehicles are neither guided to follow a leader nor are they in pursuit of a formation. Rather, the objective of each vehicle is to independently reach its desired destination while taking information of other vehicles into consideration. Co-operation among the vehicles in our framework only exists to the extent that each vehicle can reach its desired destination without collision with each other and the obstacles.

Attention based architectures: In the context of deep learning, the attention mechanism was primarily introduced in the discourse of Natural Language Processing [23]. However, it has gained utility in other domains too such as for the task of single vehicle control [24]. Here, attention is applied between different patches of the sensory data for the same vehicle. In contrast, our method applies attention between the different vehicles to be controlled. [12] uses an Graph Attention Isomorphism Neural Network (GAIN) architecture, where the task is to predict the trajectory of multiple vehicles using their past trajectory as input. Our method in contrast predicts the control commands using only the existing and destination state rather than the entire historical trajectory.

Reinforcement Learning [25], [26] use reinforcement learning for navigating vehicular-traffic. However, this is done by controlling the traffic signal rather than controlling the individual vehicles. Moreover, this is done only at intersections where clear rules for vehicle navigation exist. In our case, we handle an unconstrained environment where no predefined rules exist for negotiating bottlenecks. The only objective that the vehicles must ensure is that they avoid collisions while reaching the destination. [27] also use RL in simulation but for the control of humanoids rather than vehicles. [28] performs multi-agent path finding but in a discrete action space in the gridworld. This is as opposed to our approach which is in continuous action space and environment. The co-operative navigation task of [29] is similar to our task of navigating multiple agents to desired destinations. However, the agents are treated as particles while our approach considers the vehicle kinematics. Another issue with RL methods is that the training tends to be heavily sample-inefficient [30], [31] requiring far greater training sessions than methods where the target labels are already known such as in our case through optimization techniques.

BEV representation: Note that the state of the vehicles are represented in a Bird's Eye View (BEV) format. This format has extensively been used in various perception related tasks such as object detection, tracking, segmentation [32], [33] etc. It has also demonstrated to be very convenient for planning and control tasks. For e.g. [34] takes the BEV representation along with the intention of a single vehicle as input to predict its trajectory. [35] also uses the BEV representation to first train a teacher for the task of single vehicle control. Knowledge distillation is then used to train a subsequent student model that takes image data as input for control prediction. [36] uses pseudo-labeling of web images as part of a pipeline that predicts future waypoints in the BEV space. The aforementioned approaches for prediction and control are only tailored towards single vehicles. Our approach on the other hand is capable of handling multiple vehicles.

III. Framework

In this section, we first describe the architecture of our Attention based Graphical Neural Network (A-GNN) (See

Subsection III-A). Next, the process to generate the steering labels for training the A-GNN is discussed (See Subsection III-B).

A. Attention Graph Neuron Network

The A-GNN takes as input the state information of both the $N_{vehicle}$ dynamic vehicles and $N_{obstacles}$ static obstacles in the scene to predict the control commands (steering angle, φ and pedal acceleration, p) for all vehicles. Each vehicle/obstacle is indexed by $i \in [\ 1,\ N_{vehicle} + N_{obstacle}]$. The feature vector for entity i at layer l of the neural network is denoted by z_i^l. The input feature vector for each entity is given by $z_i^0 \in \mathbb{R}^8$ representing the current location (x and y), current orientation (θ), current velocity v, target position ($\hat{x}$ and $\hat{y}$), target orientation ($\hat{\theta}$) and whether or not the entity is a vehicle (0) or a circular obstacle (with radius r). Hence, with this representation $z^0_{i_{vehicle}} = [x, y, \theta, v, \hat{x}, \hat{y}, \hat{\theta}, 0]^T$. Meanwhile, $z^0_{i_{obstacle}} = [x, y, \theta, 0, x, y, \theta, r]^T$. Note that the target state of the static obstacle is the same as its current state. This is because the obstacles are stationary.

We now construct a Graph wherein each vehicle and obstacle is considered a node. Input node i is represented by the feature vector z_i^0. Note that each vehicle node in the graph needs to retrieve state information about all other entities in the environment in order to avoid collision and reach its desired destination. Therefore, we build an edge from all the other vehicle nodes and all obstacle nodes towards this vehicle node. Meanwhile, the obstacle nodes are static and do not have any incoming edge. Mathematically, for any vehicle node i its neighbors in the Graph G are $N_i = \{j | j = 0, 1, 2..., N_{vehicles} + N_{obstacle} \cap j \neq i\}$ and for any of the obstacle nodes its neighbor in the graph G is $\emptyset$.

This Graph G is then passed through a series of neural layers to eventually predict the control command for each vehicle. The control command $\in \mathbb{R}^2$ corresponds to the steering angle (φ) and pedal acceleration (p) of the vehicle. The flow of information through the A-GNN is described as follows:

The input node features are first converted to a higher dimensional latent vector: $z_i^1 = \sigma^1(W^1 \cdot z_i^0) \in \mathbb{R}^{d_1}$, where $W^1 \in \mathbb{R}^{d_1 \times 8}$ is a trainable weight matrix, while σ^1 is the non linear ReLU activation for the first layer. Next, a series of L residual graphical layers are used. In these layers information about the neighbouring entities is retrieved by each node via an attention based mechanism. The residual connection carries information from the preceding layer $l-1$ to a successive layer $l+1$. This ensures that prior information important for the network is carried forward. Mathematically, this process is described by:

for $k = 1, 2, ..., L$:

$$\begin{aligned} z_i^{2k} &= \sigma^{2k}(F_{Att}^{2k}(z_i^{2k-1}, z_{N_i}^{2k-1})) \\ z_i^{2k+1} &= \sigma^{2k+1}(F_{Att}^{2k+1}(z_i^{2k}, z_{N_i}^{2k}) + z_i^{2k-1}) \end{aligned} \tag{1}$$

where $z_i^{2k-1} \in \mathbb{R}^{d_{2k-1}}$, $z_i^{2k} \in \mathbb{R}^{d_{2k}}$ and $z_i^{2k+1} \in \mathbb{R}^{d_{2k+1}}$. To cater for the residual connection in the last equation of the for loop, note that: $d_{2k+1} = d_{2k-1}$. Meanwhile F_{Att}^{l+1} describes the *Attention* mechanism at layer $l+1$ and is defined by: ,

$$F_{Att}^{l+1}(z_i^l, z_j^l)) = W_{self}^l \cdot z_i^l + \sum_{j \in N_i} \alpha_{ij} \cdot F_{value}(z_i^l | z_i^l - z_j^l)) \tag{2}$$

where $\alpha_{ij} = \frac{F_{query}(z_i^l)^T \cdot F_{key}(z_j^l)}{\sum_{k \in N_i} F_{query}(z_i^l)^T \cdot F_{key}(z_k^l)}$ are the attention weights of the neighbours of vehicle i. Meanwhile, F_{value}, F_{query}, F_{key} are the joint trainable parameters of a U-Net inspired architecture with shared encoder weights for the attention mechanism.

The final layer F of the A-GNN outputs the steering commands for all the vehicles:

$$z_i^F = \sigma^F(W^F \cdot z_i^{2L+1}) \tag{3}$$

σ^l with $l \in \{1, ..., 2L+1\}$ is the ReLU non-linear activation. Meanwhile, σ^F is chosen to be the scaled *tanh* function to accommodate negative values of the steering angles and reverse pedal acceleration.

B. Label Generation Process

In this subsection, we describe the procedure for generating the data and corresponding control commands. This is done by first creating a motion model of each vehicle using the kinematic bicycle model [37]. Next, the optimization is run such that all vehicles reach their desired destination state without colliding with each other or any obstacles in the scene. Once we generate enough data, the A-GNN model architecture described in Subsection III-A is trained to predict the control commands for the corresponding vehicle scenarios in the dataset. Now one may rightfully ask, what is the advantage of training the A-GNN when we can already generate the control commands from optimization? The reason is that as number of cars in the environment is increased, the optimization becomes slower and may not be applicable for real time operation. The advantage of the A-GNN is that it learns to extract patterns from the data and applies them to similar situations not seen during training. In fact, in the experiments section, we show that the data is generated using optimization for a maximum of 3 vehicles in the environment. However, the A-GNN is powerful enough to even make predictions for controlling 6 vehicles in the scene.

Recall that the state of the vehicle is described by its location (x and y), orientation/angle (θ) and velocity (v). The vehicle can be controlled by adjusting the acceleration from the pedal (p) and maneuvering the steering angle (φ). Using the kinematic bicycle model, the equations of motion can be updated from time t to $t+1$ using the following:

$$\begin{aligned} x_{t+1} &= x_t + v_t \cdot \cos(\theta_t) \cdot \Delta t \\ y_{t+1} &= y_t + v_t \cdot \sin(\theta_t) \cdot \Delta t \\ \theta_{t+1} &= \theta_t + v_t \cdot \tan(\varphi_t) \cdot \gamma \cdot \Delta t \\ v_{t+1} &= \beta \cdot v_t + p \cdot \Delta t \end{aligned} \tag{4}$$

where, β and γ are tuneable parameters during optimization. We would like to use these equations of motion to determine the control commands of each vehicle for a horizon of H timesteps ahead. The cost function to be minimized during

the optimization should cater to two primary objectives: One is to guide the vehicle to the target location and the other is to prevent collision with other vehicles/obstacles. The first component of the cost ensures that the predicted vehicle state at any step in the horizon is as close to the target state as possible. This is done by penalizing the difference between current vehicle state and target vehicle state through target cost C_{tar}:

$$C_{tar} = \sum_{t=1}^{H} \sum_{i=1}^{N_{vehicle}} \|X_t^{(i)} - X_{target}^{(i)}\|_2 \cdot w_{pos} + \|\theta_t^{(i)} - \theta_{tar}^{(i)}\|\|_2 \cdot w_{orient} \tag{5}$$

For convenience $X = [x, y]^T$ represents the position vector. w_{pos} and w_{orient} are the tuneable weights.

The optimization should also ensure that a vehicle does not collide with obstacles and also stays clear of other vehicles. For an obstacle with radius r, a penalty is introduced if a vehicle is within a margin of r_{mar_obs}. The cost occurring due to collision of any vehicle with any obstacle over any of the timesteps in the horizon is described by C_{coll_obs}:

$$C_{coll\_obs} = \sum_{t=1}^{H} \sum_{i=1}^{N_{vehicle}} \sum_{j=1}^{N_{obstacle}} [\frac{1}{\|X_t^{(i)} - X^{(j)}\|_2 - r^{(j)}} - \frac{1}{r_{mar\_obs}}] \cdot \Pi_{obs}^{i,j} \cdot w_{col\_obs}$$
$$\Pi_{obs}^{i,j} = \begin{cases} 1 & (\|X_t^{(i)} - X^{(j)}\|_2 - r^{(j)} - r_{mar\_obs}) < 0) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Likewise, a penalty is also incurred if any vehicle i collides with vehicle j or is in its vicinity with a margin less than r_{mar_veh}. The cost C_{coll_veh} is given as:

$$C_{coll\_veh} = \sum_{t=1}^{H} \sum_{i=1}^{N_{vehicle}-1} \sum_{j=i+1}^{N_{vehicle}} [\frac{1}{(\|X_t^{(i)} - X_t^{(j)}\|_2} - \frac{1}{r_{mar\_veh})}] \cdot \Pi_{veh}^{i,j} \cdot w_{col\_veh}$$
$$\Pi_{veh}^{i,j} = \begin{cases} 1 & (\|X_t^{(i)} - X_t^{(j)}\|_2 - r_{mar\_veh}) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Note that the r_{mar_obs} and r_{mar_veh} are the safety margin of the obstacles and other dynamic vehicles. When an ego-vehicle enters the safety margin of other objects, the collision cost starts to penalize it in inverse proportion to its distance to the other object. Meanwhile, w_{col_obs} and w_{col_veh} are the tuneable weights.

Finally we optimize for the control commands via:

$$\min_{p,\varphi} [C_{tar} + C_{coll\_obs} + C_{coll\_veh}] \tag{8}$$

This cost function is minimized using sequential least square programming to yield the optimal control commands.

IV. EXPERIMENTS

This section provides the results of our experiments. We first give details about the data generation and model training process in Subsection IV-A. Next, the evaluation metrics and the quantitative results are discussed in Subsection IV-B. Lastly, the behaviour of the attention weights of our A-GNN are analyzed for a scenario with three vehicles in Subsection IV-C. Meanwhile, videos demonstrating the qualitative performance of our model can be visualized in the provided video on the project page: https://yininghase.github.io/multi-agent-control/#Results-of-our-Model.

A. Data Generation and Model Training

We are not aware of any open source platform for simultaneous control of multiple vehicles in unconstrained environments that also considers vehicle kinematics. Therefore, we create our own as depicted in the provided codebase: https://github.com/yininghase/multi-agent-control. The vehicles in this platform are maneuvered following the equations of motion described in Subsection III-B. For the purpose of training the model we collect data using this platform and determine the labels by minimizing the cost function in Equation 8. The data for which labels are generated contain between 1-3 vehicles and 0-4 static obstacles, for a total of around 20,961 trajectories. The start and destination states of the vehicle/obstacles are generated at random. Each trajectory is collected for 120 timesteps. Therefore, the total number of scenarios generated are 2,515,320. Note that increasing the number of vehicles and obstacles respectively beyond 3 and 4 significantly slows down the optimization for determining the optimal control values during this data and label generation process. Nevertheless, we demonstrate in Table I, that our model is still powerful enough to make inference for even more vehicles than for which it was trained.

The collected ground truth data is split into the training and validation set with a 4:1 ratio. The data for ground truth control has a horizon of 20. However, only the control command predicted at the first step of the horizon is used to train the model. At inference time, this command is executed, which causes the vehicle to attain a new state. This new state is then fed back to the model to predict the new command. The process is iteratively repeated until the vehicle reaches the target state. Although, the model is trained to predict only the first control command, this does not mean that the rest of the steps in the horizon are meaningless. Rather, a longer horizon facilitates the optimization to look further into the future and take early actions to prevent a collision which would otherwise have been inevitable. Lastly, the MSE loss is used to train the model. It penalizes the difference between the predicted and the ground truth control variables.

B. Quantitative results

We evaluate the performance of our method in an online setting on a randomly generated dataset not seen during training. The test set contains scenarios having 1-6 vehicles and 0-4 obstacles. The model needs to predict the appropriate control commands such that all the vehicles in the test

		success to goal rate ↑					collision rate ↓					step efficiency ↑
Number of. Vehicles.	Number of. Obstacles.	Our Model	GAIN [12]	TransformerConv [13]	EdgeConv [14]	No Graph	Our Model	GAIN [12]	TransformerConv [13]	EdgeConv [14]	No Graph	Our Model
1	0	**1.0000**	0.9850	0.9894	0.1890	**1.0000**	-	-	-	-	-	1.0000
1	1	**0.9677**	0.7792	0.7580	0.7610	0.8124	**4.9616E-05**	5.6631E-03	5.3226E-03	5.4148E-03	5.5137E-03	1.0000
1	2	**0.8991**	0.5751	0.5721	0.5771	0.6593	**3.3231E-04**	1.1459E-02	1.1451E-02	1.1487E-02	1.1607E-02	1.0000
1	3	**0.8127**	0.4707	0.4929	0.4902	0.5682	**6.6306E-04**	1.5154E-02	1.6024E-02	1.6851E-02	1.6859E-02	1.0000
1	4	**0.7361**	0.3582	0.3922	0.3959	0.3582	**1.2774E-03**	1.9003E-02	2.1938E-02	2.2296E-02	2.2594E-02	1.0000
2	0	**0.9984**	0.6285	0.6600	0.6436	0.5881	**1.1006E-05**	8.6178E-03	7.6941E-03	8.0362E-03	9.7184E-03	2.0239
2	1	**0.9634**	0.6251	0.6219	0.6120	0.5627	**1.6017E-04**	1.0111E-02	9.6096E-03	1.0098E-02	1.2165E-02	1.9831
2	2	**0.8676**	0.5538	0.5624	0.5665	0.5379	**4.5357E-04**	1.2926E-02	1.2796E-02	1.3069E-02	1.4399E-02	1.9396
2	3	**0.7792**	0.4838	0.5209	0.5156	0.5049	**6.7795E-04**	1.6143E-02	1.6039E-02	1.6378E-02	1.7036E-02	1.9582
2	4	**0.6781**	0.4439	0.4982	0.4862	0.4913	**1.1135E-03**	1.8801E-02	1.8711E-02	1.9202E-02	1.8788E-02	1.8513
3	0	**0.9943**	0.4227	0.4339	0.4401	0.3993	**8.8107E-05**	1.4238E-02	1.4047E-02	1.3778E-02	1.6330E-02	2.9871
3*	1*	**0.9706**	0.4149	0.4193	0.4279	0.3708	**3.0382E-04**	1.5672E-02	1.5655E-02	1.4713E-02	1.8927E-02	2.9334
3*	2*	**0.9302**	0.4102	0.4174	0.4126	0.3582	**5.9881E-04**	1.7267E-02	1.7393E-02	1.6426E-02	2.0821E-02	2.9399
3*	3*	**0.8903**	0.4023	0.4023	0.3938	0.3499	**8.9677E-04**	1.8393E-02	1.9646E-02	1.8697E-02	2.2789E-02	2.8887
3*	4*	**0.8328**	0.3818	0.3834	0.3716	0.3328	**1.6090E-03**	2.0214E-02	2.2045E-02	2.1205E-02	2.5397E-02	3.0669
4*	0*	**0.9807**	0.2850	0.2895	0.3108	0.2614	**2.7650E-04**	1.8619E-02	1.9967E-02	1.8791E-02	2.2929E-02	3.9478
4*	1*	**0.9550**	0.3088	0.3048	0.3187	0.2526	**5.7179E-04**	1.8783E-02	2.0446E-02	1.8025E-02	2.4251E-02	4.0105
4*	2*	**0.9279**	0.3031	0.2905	0.2970	0.2379	**9.4375E-04**	1.9565E-02	2.1960E-02	1.9552E-02	2.5899E-02	3.9228
4*	3*	**0.8853**	0.3077	0.2864	0.2930	0.2344	**1.4612E-03**	1.9224E-02	2.3747E-02	2.1138E-02	2.7634E-02	3.7834
5*	0*	**0.9590**	0.2243	0.1973	0.2208	0.1797	**5.8217E-04**	2.0827E-02	2.5964E-02	2.3437E-02	2.9177E-02	4.8234
5*	1*	**0.9285**	0.2361	0.2078	0.2306	0.1699	**1.0483E-03**	2.0363E-02	2.6115E-02	2.1317E-02	3.0344E-02	4.9839
5*	2*	**0.9037**	0.2559	0.2125	0.2254	0.1663	**1.4063E-03**	1.8332E-02	2.6607E-02	2.2013E-02	3.1475E-02	4.2917
6*	0*	**0.9209**	0.1967	0.1347	0.1631	0.1207	**1.1376E-03**	1.9465E-02	3.1612E-02	2.7085E-02	3.5283E-02	7.0979
6*	1*	**0.8949**	0.1947	0.1556	0.1744	0.1195	**1.5096E-03**	1.7424E-02	3.0917E-02	2.4239E-02	3.5972E-02	5.5704
6*	2*	**0.8717**	0.1697	0.1479	0.1660	0.1152	**1.8932E-03**	1.5176E-02	3.1828E-02	2.4354E-02	3.6918E-02	2.8350

TABLE I: Shows the performance of the five different models i.e. our Model, GAIN [12], TransformerConv [13], EdgeConv [14] and No Graph (MLP) when measured against the *success-to-goal rate* and *collision rate* metrics. The evaluation is done on completely unseen test data comprising of scenarios with 1-6 vehicles and 0-4 obstacles as shown by the corresponding rows. Each row in the table is evaluated on 4062 scenarios. The rows labeled with an asterisk (*) are those vehicle-obstacle combinations that were not even in the training set. The training set only comprised of 1-3 and 0-4 obstacles. Therefore, these rows are particularly interesting in demonstrating that our model has generalization capability giving good performance even in these situations. Being the best performing, the *step efficiency* metric is also given for our method in the last column. It shows that our method is generally more efficient than running the vehicles one by one.

scenarios reach their target state without colliding with others. The online performance of the model is evaluated against three criterion:

1) Success-to-goal rate: is defined by the percentage of vehicles that successfully reach their target state within a tolerance without colliding with other objects along the way. The location tolerance is set to be 1.25 meters from the vehicle centers and angle tolerance is set to 0.2 radians. Higher value for this metric is better.

2) Collision rate: is defined as the total number of collisions caused by a model divided by the total distance travelled by all vehicles. Note that this metric is meaningless when evaluation is being done for one vehicle with no obstacles. This is because no collisions are expected to occur in such a scenario. A lower value of this metric is better. Lastly, note that the inverse of this metric describes the average total distance travelled before a collision happens.

3) Step Efficiency: A naive approach to solve the problem of navigating all the vehicles to the goal while avoiding collision is to control the individual vehicles one by one while keeping other vehicles stationary. This considerably simplifies the problem and is akin to having one dynamic agent with multiple static obstacles. However, this approach is inefficient taking more steps to solve the whole problem with many vehicles in the scene. Therefore, we introduce the step efficiency metric, to show the advantage of our model on tackling all the vehicles simultaneously. It is the ratio of the lower bound of the number of steps required by running the vehicles one by one with other vehicle kept stationary divided by the number of steps used by navigating all the vehicles simultaneously using our approach. A higher value for this metric is better.

For the step efficiency metric, calculating the total number of steps by running the the vehicles one after the other is not trivial. This is because the total number of steps is influenced by the order in which the vehicles are run. Therefore, to ensure that the the step efficiency metric is not affected by the permutation order, we use the lower bound of the actual number of steps for running the vehicles one after the other. To do this, all other vehicles are ignored when one of the vehicles is run with its steps being counted. This leads to a simplified version of the original task since the vehicle being run only has to consider the presence of static obstacles. This leads to either an equal or lesser number steps taken to reach its destination. The same is done for all the other vehicles and their steps are summed to yield the lower bound of the actual number of steps taken. Since the summation is a permutation invariant function, the step efficiency metric is therefore not influenced by the vehicle order. This is implemented by simply removing the edges between vehicles in our GNN model so that all the vehicles run simultaneously rather than one-by-one to get the total steps.

Comparision with other methods: We make comparison with 3 additional Graphical Neural Network architectures: namely the attention based GAIN [12] and TransformerConv [13] and the non-attention based EdgeConv [14]. Comparision is also made with the naive Multi-Layer Perceptron (MLP) model which has a similar architecture to our model but does

not utilize any graphical edges in its structure.

Training: The training process for all these models is similar as our method described in Subsection IV-A. The difference is in the respective architectures. GAIN [12] and TransformerConv [13] do not utilize the relative information of the neighbouring edge nodes as was done for our case in Equation 2. They also do not utilize a joint trainable U-Net inspired architecture with shared encoder weights for the attention mechanism. Meanwhile, EdgeConv [14] is different from our architecture in that a node does not attend to other nodes. Meanwhile, the MLP does not use any graphical edges in its structure. Note that GAIN [12] utilizes the historical trajectory information of the road vehicles as input. We only use the current and target state and therefore do not require processing of any temporal information. Moreover, rather than having the entire road network in the map topology guiding the vehicles, our environment is unconstrained. Therefore, to adapt the GAIN into our pipeline, we replace all our graph attention layers with the graph attention isomorphism operator introduced in [12] without needing any historical trajectory information and the entire road map topology. For TransformerConv [13] and EdgeConv [14], we directly use the implementation from [38].

Evaluation: Experimental results for all the models are shown in Table I. Note that our approach consistently performs better than all the other approaches. The model even performs well when there are 4-6 vehicles in the scene. It is important to highlight here that our training data only contained up to a maximum of 3 vehicles. This demonstrates the predictive power of our A-GNN which has generalized to work on even more number of vehicles than it was trained.

We now give some observations about the performance of our A-GNN in Table I which is superior to the other three Graphical Neural Network architectures and the naive MLP model without edges. It can be seen that the collision rate of our model is much lower and the success-to-goal rate is also higher than all the others. The reason for poor performance of the other three models is that they only seem to learn to reach the target but do not learn how to avoid collision. Figure 2 shows that in pursuit of reaching their desired destination, the vehicles controlled by the other models collide with one another or with other obstacles. Hence, they have an extremely high collision rate. The reason why our model outperforms all the other models is because of the difference in architectures. Primarily we are using a U-net inspired attention mechanism with shared encoder weights along with concatenating relative information between the self-node and the neighbouring nodes when determining the attention weights.Removal of either the U-Net architecture or not concatenating the relative information reduces the performance.

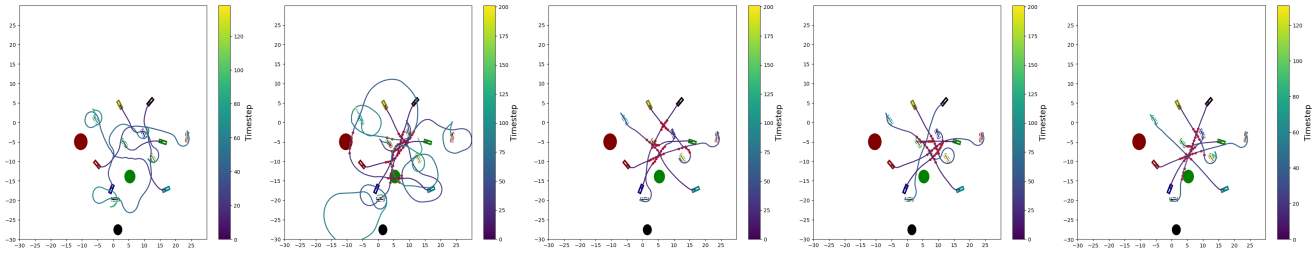
Furthermore, Figure 2 shows our model can generalize when there are 6 vehicles in the environment. Such a scenario was not present in the training dataset. The training dataset was limited to a maximum of 3 vehicles. Meanwhile, the other models show poor performance on these scenarios with plenty of collisions among the vehicles.

Note that in Table I of the main paper, the success-to-goal rate for our model is not a perfect score of 1. The reason the model fails to always reach the target destination is that it tends to behave conservatively. Rather than taking the risk of collision, it sometimes stops mid-way before other objects to avoid collision. This is because the ground truth data obtained from the optimization is not always prefect. Nevertheless, it is important to keep such failed samples in the dataset for training as they teach the model to learn to stop before other objects to avoid collision. If these failure samples are removed from the training set, then the model tends to perform worse as it starts colliding with other obstacles.

Lastly, since our model is the best performing among all others, the last column in Table I also provides the *step efficiency* metric for it. It can be observed that our method of executing the control commands for all vehicles simultaneously tends to be generally faster than running the vehicles one by one.

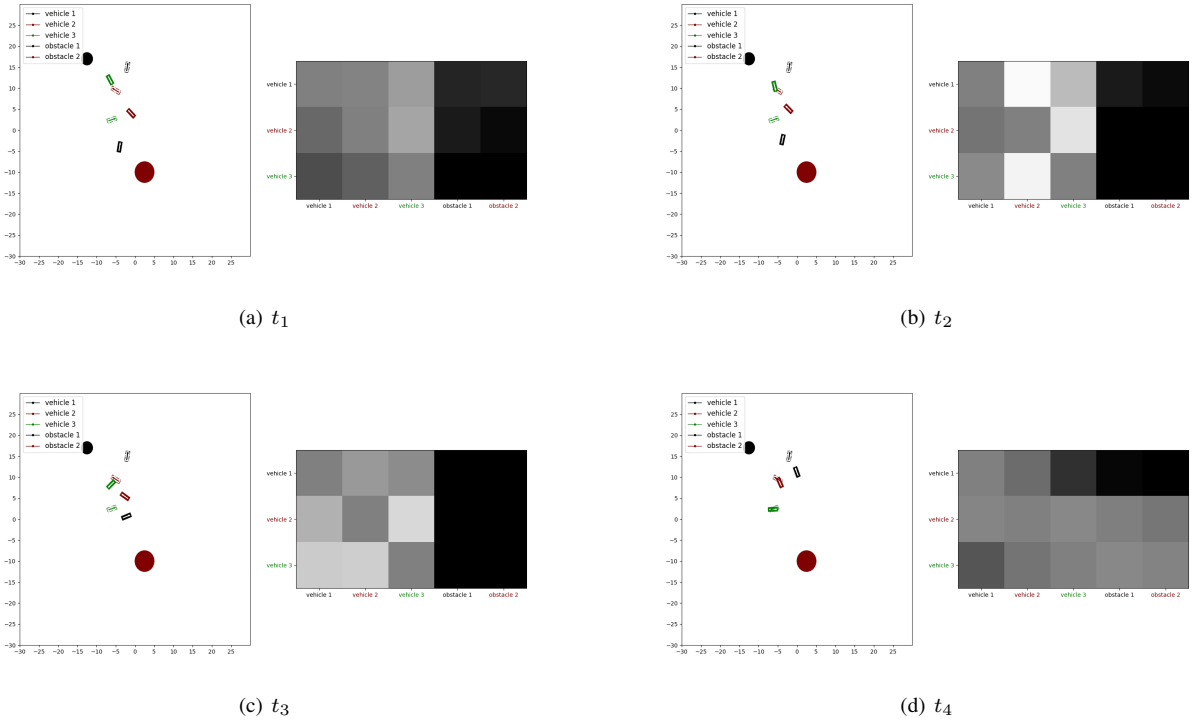
C. Attention Weights Analysis

To get some intuition about the behavior of our model, we visualize the mean of attention logits from all the graph attention layers of our model. Figure 3 shows a visualization of these attention logits as a matrix for 4 timesteps t_1, t_2, t_3 and t_4 for a scenario with 3 vehicles and 2 obstacles. The rows in the attention matrix correspond to the vehicle of interest. The columns show which vehicle/obstacle is being attended to. A lighter shade in the attention matrix depicts high attention and a darker shade represents lack of attention. Because the static obstacles are stationary and are never a hindrance for any of the 3 vehicles, their attention logits of all the vehicles on them is generally low (dark pixel on the last 2 columns) for all the 4 timesteps. At time t_1, the vehicles are relatively far away from each other, so the attention logits are relatively low. At $t = t_2$, the **red** vehicle (vehicle 2) is in the way of the **black** vehicle (vehicle 1), the **green** vehicle (vehicle 3) is in the way of **red** vehicle (vehicle 2), the **red** vehicle (vehicle 2) is in the way of **green** vehicle (vehicle 3) and all the 3 vehicles are close to each other. So each vehicle pays more attention to the other vehicles that are on their way to the target and in close proximity. These 3 corresponding pixels are therefore very bright. At $t = t_3$, the **red** vehicle (vehicle 2) decides to stop and let the **green** vehicle (vehicle 3) pass through. So the attention of the **green** vehicle (vehicle 3) on the **red** vehicle (vehicle 2) goes down. However the **red** vehicle (vehicle 2) needs to wait until the **green** vehicle (vehicle 3) passes through. So the attention of **red** vehicle (vehicle 2) on the **green** vehicle (vehicle 3) is still high. And the **black** vehicle (vehicle 1) has found its way to go around the **red** vehicle (vehicle 2), so the attention of the **black** vehicle (vehicle 1) on the **red** vehicle (vehicle 2) also goes down. Finally, all the vehicles reach their goal, and their attentions on each other reduce. This trivial qualitative example demonstrates, that our model is able to learn driving behaviour analogous to how a human would make decisions. However, more experiments and quantitative studies would be required to better understand the reasoning process behind these graphical neural architectures. Therefore, this work can



(a) Our Model (b) GAIN (c) TransformerConv (d) EdgeConv (e) No Graph

Fig. 2: Shows the trajectory traversed by the vehicles as a result of applying the control commands predicted by the five different models for a scenario containing 6 vehicles and 3 obstacles. The red dots on the trajectories show the point of collision between the vehicles. Except for our model, all other models have plenty of collisions. Corresponding videos for all the models can be found on the project page: https://yininghase.github.io/multi-agent-control/#Comparison-with-Other-Models.



(a) t_1 (b) t_2

(c) t_3 (d) t_4

Fig. 3: Predicted Trajectories of our model with Attention Logits: The color of pixel at row i column j shows the attention logits of object i on object j. The lower the logits value is, the darker is the color and vice versa. The gray value on the diagonal means that a vehicle is neutral to attending to itself. Corresponding video can be found on the project page: https://yininghase.github.io/multi-agent-control/#Attention-Mechanism-of-our-Model.

potentially be used as a starting point for interpreting the decision making capabilities of trained models before their deployment in relevant and related applications.

V. Future Work

As we discussed in Section II, concurrent Reinforcement Learning (RL) methods do not consider the vehicle kinematics and also tend to be heavily sample-inefficient requiring far more steps than our method to train the model. However, the advantage is that they can be trained without labels obtained from optimization. Therefore, to exploit this aspect, we can extend this work by using our pretrained model as an initializer to prevent the cold-start problem associated with Reinforcement Learning. This way the model can be trained by progressively adding more vehicles into the environment. This prevents the cumbersome offline optimization process.

VI. Conclusion

In this paper, we proposed an attention based graphical neural network model. It is capable of predicting the control commands for multiple agents for reaching their desired destination without collision. We demonstrated that utilizing

information of both the ego-vehicle and neighboring agents in the graphical layers helps the model generalize well. In fact, the model performs even on situations with more vehicles and obstacles than those in the training set.

Supplementary Information: For the interested reader, further details about the model architecture, run times, training details, etc can be found in the supplementary material on the project page: https://yininghase.github.io/multi-agent-control/

Acknowledgements: We thank Marc Brede for helping with the initial setup of the label generation process in the early phase of the project.

REFERENCES

[1] V. Rausch, A. Hansen, E. Solowjow, C. Liu, E. Kreuzer, and J. K. Hedrick, "Learning a deep neural net policy for end-to-end control of autonomous vehicles," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 4914–4919.

[2] H. M. Eraqi, M. N. Moustafa, and J. Honer, "End-to-end deep learning for steering autonomous vehicles considering temporal dependencies," *arXiv preprint arXiv:1710.03804*, 2017.

[3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[4] I. Sobh, L. Amin, S. Abdelkarim, K. Elmadawy, M. Saeed, O. Abdeltawab, M. Gamal, and A. El Sallab, "End-to-end multi-modal sensors fusion system for urban automated driving," *Advances in Neural Information Processing Systems (NIPS) Workshops*, 2018.

[5] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[6] J. Pérez Fernández, M. Alcázar Vargas, J. M. Velasco García, J. A. Cabrera Carrillo, and J. J. Castillo Aguilar, "Low-cost fpga-based electronic control unit for vehicle control systems," *Sensors*, vol. 19, no. 8, p. 1834, 2019.

[7] T. Wu, W. Liu, and Y. Jin, "An end-to-end solution to autonomous driving based on xilinx fpga," in *2019 International Conference on Field-Programmable Technology (ICFPT)*, 2019, pp. 427–430.

[8] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*. Springer, 2018, pp. 621–635.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[10] H. Karnan, G. Warnell, X. Xiao, and P. Stone, "Voila: Visual-observation-only imitation learning for autonomous navigation," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2497–2503.

[11] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "Transfuser: Imitation with transformer-based sensor fusion for autonomous driving," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–18, 2022.

[12] Y. Liu, X. Qi, E. A. Sisbot, and K. Oguchi, "Multi-agent trajectory prediction with graph attention isomorphism neural network," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 273–279.

[13] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked label prediction: Unified message passing model for semi-supervised classification," *arXiv preprint arXiv:2009.03509*, 2020.

[14] W. Yue, S. Yongbin, L. Ziwei, S. E. Sarma, and M. M. Bronstein, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, 2019.

[15] M. Abbas, M. Elhamshary, H. Rizk, M. Torki, and M. Youssef, "Wideep: Wifi-based accurate and robust indoor localization system using deep learning," in *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, 2019, pp. 1–10.

[16] X. Wang, L. Gao, S. Mao, and S. Pandey, "Csi-based fingerprinting for indoor localization: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 763–776, 2016.

[17] Y. Yuan, X. Weng, Y. Ou, and K. M. Kitani, "Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9813–9823.

[18] H. Tang, P. Wei, J. Li, and N. Zheng, "Evostgat: Evolving spatiotemporal graph attention networks for pedestrian trajectory prediction," *Neurocomputing*, vol. 491, pp. 333–342, 2022.

[19] B. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2375–2384.

[20] S. Moorthy and Y. H. Joo, "Distributed leader-following formation control for multiple nonholonomic mobile robots via bioinspired neurodynamic approach," *Neurocomputing*, vol. 492, pp. 308–321, 2022.

[21] S. He, R. Xu, Z. Zhao, and T. Zou, "Vision-based neural formation tracking control of multiple autonomous vehicles with visibility and performance constraints," *Neurocomputing*, vol. 492, pp. 651–663, 2022.

[22] R. Dutta, H. Kandath, S. Jayavelu, L. Xiaoli, S. Sundaram, and D. Pack, "A decentralized learning strategy to restore connectivity during multi-agent formation control," *Neurocomputing*, vol. 520, pp. 33–45, 2023.

[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[24] K. Chitta, A. Prakash, and A. Geiger, "Neat: Neural attention fields for end-to-end autonomous driving," in *International Conference on Computer Vision (ICCV)*, 2021.

[25] M. Guo, P. Wang, C.-Y. Chan, and S. Askary, "A reinforcement learning approach for intelligent traffic signal control at urban intersections," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 4242–4247.

[26] Z. Ge, "Reinforcement learning-based signal control strategies to improve travel efficiency at urban intersection," in *2020 International Conference on Urban Engineering and Management Science (ICUEMS)*, 2020, pp. 347–351.

[27] B. Haworth, G. Berseth, S. Moon, P. Faloutsos, and M. Kapadia, "Deep integration of physical humanoid control and crowd navigation," in *Proceedings of the 13th ACM SIGGRAPH Conference on Motion, Interaction and Games*, 2020, pp. 1–10.

[28] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. S. Kumar, S. Koenig, and H. Choset, "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019.

[29] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[30] N. Vithayathil Varghese and Q. H. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics*, vol. 9, no. 9, 2020.

[31] D. Li, D. Zhao, Q. Zhang, and Y. Chen, "Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]," *IEEE Computational Intelligence Magazine*, vol. 14, no. 2, pp. 83–98, 2019.

[32] L. Peng, Z. Chen, Z. Fu, P. Liang, and E. Cheng, "Bevsegformer: Bird's eye view semantic segmentation from arbitrary camera rigs," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2023, pp. 5935–5943.

[33] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.

[34] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[35] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on Robot Learning (CoRL)*, 2019.

[36] J. Zhang, R. Zhu, and E. Ohn-Bar, "Selfd: Self-learning large-scale driving policies from the web," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 17 316–17 326.

[37] D. Wang and F. Qi, "Trajectory planning for a four-wheel-steering vehicle," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, 2001.

[38] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.