VI. Supplementary

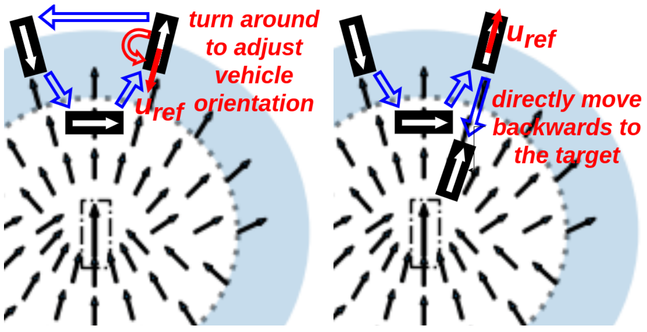
A. Further Explanation of MA-DV^{2}F

In this Subsection, we further elaborate some the design choices for Equation 2 used to determine the reference orientation. The Equation is repeated here again for clarity:

$$\mathbf{u}_{tar}^{(i)} = \begin{cases} \mathbf{f_{uni}}(\mathbf{X}_{tar}^{(i)}) \cdot \xi_{tar}^{(i)} & \|\mathbf{X}_{tar}^{(i)}\|_2 > r_p \\ \mathbf{f_{uni}}(\mathbf{U}_{tar}^{(i)} + \lambda_{tar}^{(i)} \cdot \mathbf{f_{uni}}(\mathbf{X}_{tar}^{(i)})) & \text{otherwise} \end{cases}$$

$$\lambda_{tar}^{(i)} = (\frac{\|\mathbf{X}_{tar}^{(i)}\|_2}{r_p} + f_{pos}(\|\mathbf{X}_{tar}^{(i)}\|_2 - \epsilon_p)) \cdot f_{sgn}(\mathbf{X}_{tar}^{(i)\,T} \cdot \mathbf{U}_{tar}^{(i)})$$

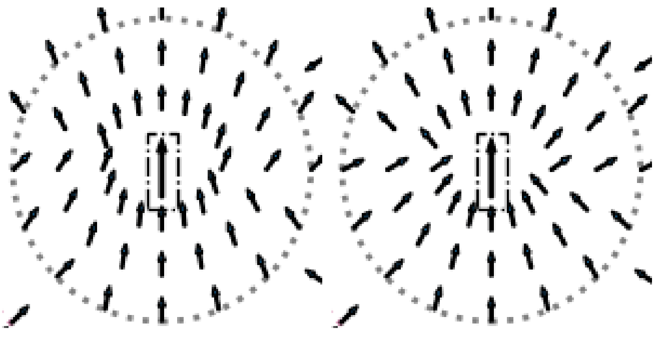
$$\xi_{tar}^{(i)} = \begin{cases} 1 & \|\mathbf{X}_{tar}^{(i)}\|_2 \geq 0.5 \cdot v_d^2 + r_p \\ f_{sgn}(\mathbf{X}_{tar}^{(i)\,T} \cdot \mathbf{U}_t^{(i)}) & \text{otherwise} \end{cases}$$


turn around to adjust vehicle orientation
u_{ref}
u_{ref}
directly move backwards to the target

(a) Keep Reference Orientation (b) Reverse Reference Orientation

Fig. 5: Shows the two different circumstances of the ego-vehicle overshooting its target and entering the marginal parking region (shaded blue region). In Subfigure (a), The reference orientation vector $\mathbf{u}_{ref} = \mathbf{f_{uni}}(\mathbf{X}_{tar}^{(i)})$ points towards the target, while the current vehicle orientation is opposite to the reference orientation. Thus, the vehicle needs to turn around again to match the reference orientation. In Subfigure (b), the reference orientation vector $\mathbf{u}_{ref} = -\mathbf{f_{uni}}(\mathbf{X}_{tar}^{(i)})$ is flipped based on the current vehicle orientation. In this case, the vehicle does not need to turn around but directly move backwards to the target.

Circular Motion Prevention: Equation 2 shows that when the ego-vehicle i is far away from the parking threshold, the reference orientation is in the direction of $\mathbf{X}_{tar}^{(i)}$. However, as the ego-vehicle approaches the target and the velocity is high, then the vehicle might overshoot the target and enter the shaded marginal parking threshold region causing the current ego-vehicle orientation $\mathbf{U}_t^{(i)}$ to be opposite to $\mathbf{X}_{tar}^{(i)}$ as seen in the left of Figure 5. This will make the vehicle go in circles in an attempt to align itself with the reference orientation. A better alternative would be switch the reference orientation direction so that it is aligned with the ego-vehicle orientation and drive the car in reverse as shown in the right of Figure 5. This is done by dynamically switching the reference orientation depending on the sign of the dot product: $f_{sgn}(\mathbf{X}_{tar}^{(i)\,T} .\mathbf{U}_t^{(i)})$.



(a) Without Refining Item (b) With Refining Item

Fig. 6: Shows the reference orientations near the target with/without the refining item $f_{pos}(\|\mathbf{X}_{tar}^{(i)}\|_2 - \epsilon_p)$ in $\lambda_{tar}^{(i)}$ in Equation 2. Subfigure (a) shows the situation without the refining item. The reference direction arrows in the left side or right side of the target position are more parallel to the target direction. This will lead to the vehicle shaking forwards and backwards there rather than approaching the target. Therefore, the refining item $f_{pos}(\|\mathbf{X}_{tar}^{(i)}\|_2 - \epsilon_p)$ is added to $\lambda_{tar}^{(i)}$. The new reference direction arrows shown in Subfigure (b) have more biases towards the target position, which can accelerate the parking process.

Parking behaviour Refinement: In Equation 3, an additional term: $f_{pos}(\|\mathbf{V}_{tar}^{(i)}\|_2 - \epsilon_p)$ was introduced in $\lambda_{tar}^{(i)}$ to refine the parking behavior when the vehicle position is exactly on either the left or the right side of the target. As can be seen in Figure 6, this additional refinement term causes the reference orientation to be more biased towards the target position within the parking region. The ego vehicle aligns itself better to the reference orientation there, while simultaneously allowing it to reach the final target quicker. Removing this term would make the reference orientations rather parallel to the final target causing the ego vehicle to oscillate forward and backwards around the target leading to a longer parking time. This is particularly true when the ego vehicle position is exactly to the left or to the right side of the target.

B. Self-supervised Training of GNN Model

The reference steering angles $\varphi_t^{(i)}$ and reference pedal acceleration $p_t^{(i)}$ mentioned in Section III cannot only be used to control the vehicles directly, but also as labels to supervise training of the GNN model. This approach can therefore additionally be used to train a learning based Graphical Neural Network (GNN) controller in a self-supervised manner using these control labels and network architecture given in [12]. However, [12] requires running an optimization based procedure offline to collect enough training data with ground truth labels. This is a computationally expensive and slow process when the number of agents are large. In contrast, our self-supervised learning method is capable of directly training the model online during the simulation process without collecting ground truth labels beforehand. Note that in [12],

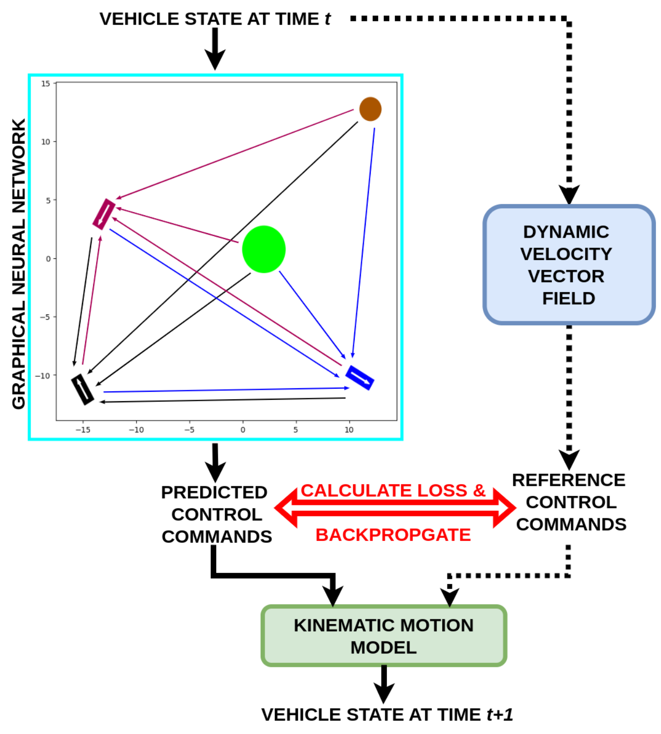

VEHICLE STATE AT TIME t
GRAPHICAL NEURAL NETWORK
DYNAMIC VELOCITY VECTOR FIELD
PREDICTED CONTROL COMMANDS
CALCULATE LOSS & BACKPROPGATE
REFERENCE CONTROL COMMANDS
KINEMATIC MOTION MODEL
VEHICLE STATE AT TIME t+1

Fig. 7: Shows the pipeline of the self-supervised learning part. The model is trained by running simulation on different training cases without existing labels. For the simulation with T steps in total, at each step of the simulation, the DV^{2}F are calculated for all the vehicles. For each vehicle, its DV^{2}F gives the reference control variables as training labels. At the same time, the state of the vehicles and the obstacles will be inputted to the training model, obtaining the predicted control variables from the model. Then, the loss is calculated based on the reference and predicted control variables and back-propagated to update the weights of the model. The predicted control variables will also be passed to the vehicle kinetic equation to update the vehicle states for the next simulation step. This process iterates until finishing this simulation turn.

all the vehicle nodes in the GNN model will have incoming edges from any other vehicle or obstacle nodes. In this case, the number of edges in the graph will grow quadratic to the number of agents(neighbouring vehicles/obstacles), leading to a heavy computational burden when when scaling to more agents. Thus, we remove edges between the neighboring agent j and the ego vehicle i if the distance $\|X_j^{(i)}\|$ between them is greater than the threshold given by:

$$D_j^{(i)} = \begin{cases} r_{obs}^{(j)} + r_{veh} + |v_t^{(i)}| + 2 \cdot r_c & j \text{ is an obstacle} \\ 2 \cdot r_{veh} + |v_t^{(i)}| + |v_t^{(j)}| + 2 \cdot r_c & j \text{ is a vehicle} \end{cases} \tag{11}$$

Training Pipeline: The self-supervised learning pipeline is shown in as Figure 7. At the start of training, different training samples with multiple agents placed at random positions are generated as the start points of the DV^{2}F simulations. Each sample only contains the states of the vehicles and static obstacles without any control labels. At each step during the simulation, the reference control variables of each vehicle is determined online using the DV^{2}F according to the states of the vehicles and obstacles at that time. The loss is calculated on the fly based on the reference control variables and the predicted control variables by the model. This loss is then back-propagated to update the model immediately at the current step. Unlike [12] which solves the individual optimization problem for each case one-by-one, our dynamic velocity vector field determines reference control variables in a closed-form solution. Thus, we can run one simulation with multiple cases as a batch running in parallel, and then change to another batch for the next turn of simulation until finishing all the training cases as an epoch.

Loss Function: The dynamic velocity vector field gives a low reference speed limited by $|v_d|$. However, if the vehicle is still far away from its target and has low risk to colliding with other agents, the speed limit of $|v_d|$ can be removed allowing the vehicle to move faster to its target. To this end, we first define a vehicle state cost to evaluate vehicle control. Assume the current vehicle position $(x_t^{(i)}, y_t^{(i)})$, orientation $\theta_t^{(i)}$ and speed $v_t^{(i)}$ are fixed, for the given vehicle control variables $\varphi_t^{(i)}$ and $p_t^{(i)}$, we first apply the vehicle kinematic equation to obtain $x_{t+2}^{(i)}$, $y_{t+2}^{(i)}$, $\theta_{t+1}^{(i)}$ and $v_{t+1}^{(i)}$. We redefine $\|\mathbf{X}_{tar}^{(i)}\|$, $\|\mathbf{X}_{obs_j}^{(i)}\|$ and $\|\mathbf{X}_{veh_j}^{(i)}\|$ for the time $t+2$. The vehicle state cost $C(\varphi_t^{(i)}, p_t^{(i)})$ is then calculated as follows:

$$\begin{aligned} C &= C_{tar} + C_{obs} + C_{veh} \\ C_{tar} &= \|\mathbf{X}_{tar}^{(i)}\|_2 \\ C_{obs} &= \sum_{j=1}^{N_{obs}} f_{max}^2(-\alpha_{obs_j}^{(i)}, 0) + f_{max}(-\alpha_{obs_j}^{(i)}, 0) \\ C_{veh} &= \sum_{j=1, j\neq i}^{N_{veh}} f_{max}^2(-\alpha_{veh_j}^{(i)}, 0) + f_{max}(-\alpha_{veh_j}^{(i)}, 0) \end{aligned} \tag{12}$$

where $\alpha_{obs_j}^{(i)} = \|\mathbf{X}_{obs_j}^{(i)}\|_2 - r_{obs}^{(j)} - r_{veh} - (r_c + |v_t^{(i)}|)$ and $\alpha_{veh_j}^{(i)} = \|\mathbf{X}_{veh_j}^{(i)}\|_2 - 2 \cdot r_{veh} - (r_c + |v_t^{(i)}|)$ are defined same as in Equation 3 mentioned in Section III-B, C_{tar} measures the distance of the ego vehicle i to its target, and C_{obs} and C_{veh} evaluate the collision risk of the ego vehicle. The gradient of this vehicle state cost should alone be enough to guide the GNN model in learning to reach the target while avoiding collisions. However, in practice, the GNN does not converge to the optimal due to high non-linearities in Equation 12. Therefore, we combine this equation with the labels obtained from the dynamic velocity vector field which expedites the model training.

The overall loss function used in this pipeline is defined as follows:

$$\begin{aligned}
L &= L_{steer} + L_{pedal} \\
L_{steer} &= (\varphi_t^{(i)} - \tilde{\varphi}_t^{(i)})^2 \\
L_{pedal} &= \begin{cases} \Delta C + f_{pos}(\Delta C) \cdot (\Delta C)^2 & \alpha_{tar}^{(i)} \wedge \beta_{tar}^{(i)} \\ (p_t^{(i)} - \tilde{p}_t^{(i)})^2 & \text{otherwise} \end{cases} \\
\Delta C &= C(\varphi_t^{(i)}, \tilde{p}_t^{(i)}) - C(\varphi_t^{(i)}, p_t^{(i)}) \\
\alpha_{tar}^{(i)} &= \|\mathbf{X}_{tar}^{(i)}\|_2 - |\tilde{v}_{t+1}^{(i)}| - r_p > 0 \\
\beta_{tar}^{(i)} &= (|\tilde{v}_{t+1}^{(i)}| > v_d) \wedge (|v_{t+1}^{(i)}| = v_d) \wedge (\tilde{v}_{t+1}^{(i)} \cdot v_{t+1}^{(i)} > 0)
\end{aligned} \tag{13}$$

where the $\varphi_t^{(i)}$, $p_t^{(i)}$ and $v_{t+1}^{(i)}$ are the reference steering angle, reference pedal and calculated reference speed from dynamic velocity vector field, the $\tilde{\varphi}_t^{(i)}$, $\tilde{p}_t^{(i)}$ and $\tilde{v}_{t+1}^{(i)}$ are the corresponding values predicted by the GNN model. During training, the steering angle $\tilde{\varphi}_t^{(i)}$ is fully supervised by reference steering angle $\varphi_t^{(i)}$. However, the pedal acceleration loss comprises one of two parts depending on the condition of the vehicle. If the vehicle is far way from the parking region, i.e. $\|\mathbf{X}_{tar}^{(i)}\|_2 - |\tilde{v}_{t+1}| - r_p > 0$, and the default reference speed v_d limits the predicted speed, i.e. $(|\tilde{v}_{t+1}| > v_d) \wedge (|v_{t+1}| = v_d) \wedge (\tilde{v}_{t+1} \cdot v_{t+1} > 0)$, then the pedal loss is supervised by the relative vehicle state cost $C(\varphi_t^{(i)}, \tilde{p}_t^{(i)}) - C(\varphi_t^{(i)}, p_t^{(i)})$. This allows the vehicle to speed up when no other agents are nearby and slow down when getting close to other objects or its target. Otherwise, the pedal acceleration loss is supervised by the reference pedal $\tilde{p}_t^{(i)}$. Note that we use the reference steering angle $\varphi_t^{(i)}$ rather than the predicted steering angle $\tilde{\varphi}_t^{(i)}$ to calculate the vehicle state cost.

C. Training & Test Data Generation

The advantage of our self-supervised approach is that the data for training the GNN model can be generated on the fly, since it does not require supervised labels. However, for the purpose of reproducibilty of our experiments, we generate date beforehand. One common option is to generate the samples by choosing a vehicle's starting and target positions randomly on the navigation grid. However, the risk with this approach is that the dataset might be heavily skewed in favour of one scenario and may not capture other types of situations that the vehicle is expected to encounter. In contrast, our training regime is developed to handle these diverse situations. These situations can primarily be segregated into 3 modes: *collision, parking and normal driving*. In the collision mode, the vehicles are placed such that they have a high probability of collision. This is done by first randomly choosing a point on the grid, defined as a "collision center". The starting and target position of at least two vehicles are placed on opposite sides of this collision center with slight random deviation. Hence, the model will be pushed to learn a collision avoidance maneuver. Meanwhile, in the parking mode, the target position for each vehicle is sampled near its start position (within a distance of $10m$). Note that in both the collision and parking modes, the position of the vehicles are also chosen randomly but with certain constraints i.e. starting and target positions being on the opposite sides of collision center (collision mode) or in close proximity (parking mode). In the *normal driving* mode, the starting and target positions are chosen randomly without any of the constraints described earlier. Lastly, for all modes the following additional conditions are to be fulfilled: No two target or two staring positions of vehicles can overlap. Likewise, if there are static obstacles, the starting/target positions cannot overlap with it. The starting position can be placed within an obstacle's circle of influence but not a target position, since then the vehicle will struggle attaining equilibrium. The target position will attract the ego-vehicle whereas the obstacle influence will repel it. Our training dataset contains training cases from 1 vehicle 0 obstacles to 5 vehicles 8 obstacles, each with 3000 samples that in turn contains 1000 samples from each of the 3 modes. Each of these 3000 scenarios serve as the starting point for the simulation that is run for $T = 200$ timesteps during training, in order for the vehicles to reach their target positions. As mentioned in Section IV-A, we also generate test scenarios range from 10 vehicles - 0 obstacles to 50 vehicles - 25 obstacles, each with 1000 samples. However, our test dataset is generated only using the collision mode.

Multiple training samples can be grouped as a batch and run simultaneously. However, this training simulation will always end with all the vehicles in the training batch approaching their targets and parking. To prevent the model from overfitting on parking behavior, we adopt a asynchronous simulation during the training. Specifically, a training batch if further divided into multiple mini-batches. These mini-batches are in different simulation time steps. In this case, within a training batch, there are always some training samples just starting, some driving the vehicles on the way and the rest parking the vehicles. Besides, for each time step t during the training simulation, we also perturbs the vehicle state using a zero-mean Gaussian noise as a data augmentation. The standard variance decreases linearly along the simulation time t: $\boldsymbol{\sigma}_t = \frac{T-t}{T} \cdot [\sigma_x, \sigma_y, \sigma_\varphi, \sigma_v]^T$. The validation dataset reuse the training samples without random perturbation. The simulation length during validation is reduced to $T = 1$.

To train the neural network, we use the Adam optimizer with an initial learning rate of 0.01 and weight decay of 10^{-6}. The learning rate is reduced by factor of 0.2 from its previous value, if the validation loss does not reduce for 15 epochs. The number of training epochs is set to be 500 but training is prematurely stopped, if there is no decrease in the validation loss for 50 epochs.