MA-DV^{2}F: A Multi-Agent Navigation Framework Using Dynamic Velocity Vector Field

Yining Ma, Qadeer Khan, and Daniel Cremers

***Abstract*—In this paper, we propose MA-DV^{2}F: Multi-Agent Dynamic Velocity Vector Field. It is a framework for simultaneously controlling a group of vehicles in challenging environments. DV^{2}F is generated for each vehicle independently and provides a map of reference orientation and speed that a vehicle must attain at any point on the navigation grid such that it safely reaches its target. The field is dynamically updated depending on the speed and proximity of the ego-vehicle to other agents. This dynamic adaptation of the velocity vector field allows prevention of imminent collisions. Experimental results show that MA-DV^{2}F outperforms concurrent methods in terms of safety, computational efficiency and accuracy in reaching the target when scaling to a large number of vehicles.**

***Index Terms*—Path planning for multiple mobile robots or agents, autonomous vehicle navigation, autonomous agents.**

I. Introduction

THE task of multi-agent navigation has attracted widespread attention in recent years due to myriad applications in areas such as search and rescue missions [1], area exploration [2], pickup and delivery services [3], warehouses [4], self-driving cars [5] etc. The task of multi-agent path finding/navigation involves simultaneously directing a group of vehicles from their initial position to their desired destination while avoiding collisions with other agents. The task is known to be NP-hard even in the discrete setting [6]. An ideal algorithm must find the optimal solution in limited time. This leads to contradictory goals since determining the optimal solution requires searching a larger solution space, which necessitates more time. In structured environments such as indoor spaces, prior knowledge and understating of the layout impose constraints that can reduce the solution search space. In unstructured environments, there are no such constraints. This allows an algorithm the flexibility to find a solution. However, since the search space is much larger, there is no guarantee that the solution found is optimal. The problem is further exacerbated when the search space is continuous and agents are non-holomonic vehicles. The constraints arising from the vehicle kinematics add to the complexity.

There have been various techniques and heuristics attempting to find (near-)optimal trajectories for multiple agents. The methods can be divided into two primary categories: 1) Learning based data driven methods [7], [8] and 2) Search/optimization based methods [9], [10]. Learning based algorithms involve training a neural network on data, with the understanding that the network will generalize at inference time. The training data should encompass all the situations that the model is expected to encounter at test time. This necessitates a large amount of training samples. The greatest challenge with large training samples is determining the corresponding supervised labels for training; that might be too tedious to obtain. In contrast to supervised learning, an alternate would be to train using reinforcement learning (RL) [11], where the model explores the environment and acquires rewards or penalties depending on the actions taken. The model then exploits this experience to execute the correct control actions at test time. However, RL algorithms tend to be more sample inefficient than supervised methods. In contrast, optimization [12] or search based [13] methods involves simultaneously optimizing trajectories for multiple vehicles. As the number of vehicles are added, the complexity of the optimization/search becomes intractable making it infeasible to scale to a large number of vehicles [14], [15].

In this paper, we propose Multi-Agent Dynamic Velocity Vector Field (MA-DV^{2}F), which generates vectors for the orientation and reference speed for every vehicle on the map. The vehicles then just need to follow in the direction of their respective velocity vector fields to successfully reach their destinations. The vector field for each vehicle is generated independently and can be adapted dynamically depending on the vehicle's proximity to other agents (neighbouring vehicles or obstacles). Decoupling reduces the complexity of adding vehicles & allows for parallel generation of DV^{2}F of each vehicle, thereby increasing the throughput. An added benefit of our approach is that the generated DV^{2}F can be used to train a learning based graphical neural network (GNN) in a self-supervised manner. This self-supervised learning based approach neither requires tedious labeling of data nor necessitates sample inefficient environment exploration. We test our framework under challenging collision prone environments. A scenario is regarded to be challenging if trajectories of different vehicles considered independently from other agents intersect at multiple places at the same time. This would necessitate a collision avoidance maneuver for safe navigation towards the target.

Received 9 November 2024; accepted 17 March 2025. Date of publication 10 April 2025; date of current version 30 April 2025. This article was recommended for publication by Associate Editor M. Alhafnawi and Editor M. A. Hsieh upon evaluation of the reviewers' comments. This work was supported by the ERC Advanced Grant SIMULACRON. *(Corresponding author: Yining Ma.)*

The authors are with the Chair of Computer Vision & Artificial Intelligence, School of Computation, Information and Technology, Technical University of Munich, Germany, and also with the Munich Center for Machine Learning, Germany (e-mail: yining.ma@tum.de; qadeer.khan@tum.de; cremers@tum.de).

Project page for this work can be found here: https://yininghase.github.io/MA-DV2F/.

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3559830, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3559830

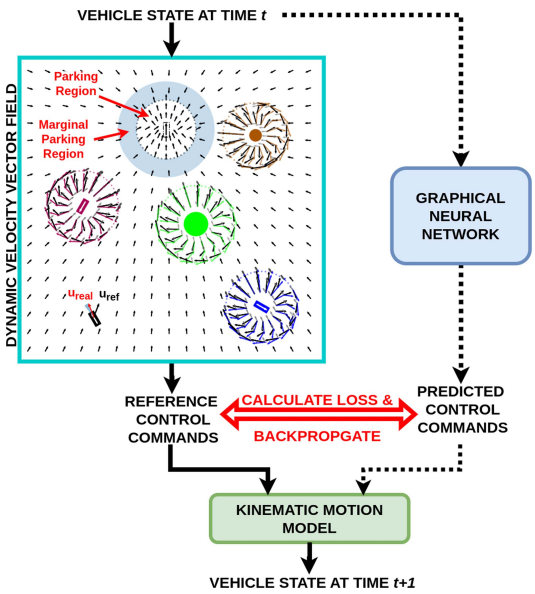


Fig. 1. Shows the pipeline for MA-DV^{2}F. The state of all vehicles at time t is used to create the dynamic velocity vector field (DV^{2}F) from which the reference control commands are determined. These commands are, in turn, used to determine the next state at time $t+1$ using the kinematic motion model. Note that the reference control commands can optionally be used to train a GNN in a self-supervised manner (indicated by dotted arrows). Meanwhile, the DV^{2}F is shown for the black ego-vehicle with 2 other vehicles (blue & maroon) and 2 obstacles (green & brown) in the scene. The target state for the black ego-vehicle is shown by the dotted rectangle with a solid arrow at the top of the map. The dotted circle around the target state is the parking region. The black arrows indicate the ideal reference orientation as unit vectors which the ego-vehicle should attain at each different position on the map. The colored dotted circles around other vehicles and obstacles show the collision avoiding regions for the black ego-vehicle. Each black arrow in these regions is composed of an attractive target approaching component (gray arrow) and a repulsive collision avoidance component (colored arrow). Note that due to kinematic constraints, the reference orientation ($\mathbf{u_{ref}}$) might not be attainable for the ego-vehicle at its existing location. The shaded wedge in front of the ego-vehicle shows the region of reachable orientation at the next time step. The real orientation ($\mathbf{u_{real}}$) is therefore the physically attainable orientation by the ego-vehicle that is closest to the reference. An example of the dynamic velocity vector field is shown in the project page: https://yininghase.github.io/MA-DV2F/#VD. Note that DV^{2}F for the neighboring blue & maroon vehicles is likewise created separately (not shown in this figure).

Fig. 1 shows the pipeline for both the MA-DV^{2}F (left branch, solid arrows) and optional training of the self-supervised GNN counterpart (right branch, dotted arrows). The input is a continuous state representation of all the multiple vehicles and the outputs are the corresponding continuous control variables. The vehicles are non-holonomic with rotation radius determined by the kinematic model.

We summarize the contribution of this letter as follows:

- Our proposed MA-DV^{2}F outperforms other concurrent learning and search based approaches for the task of multi agent navigation in challenging, collision prone environments.
- Even the self-supervised learning based counterpart of MA-DV^{2}F scales better than other learning and search based methods.
- MA-DV^{2}F can determine the solutions orders of magnitude faster than other SOTA search based approaches.
- We release the complete code of MA-DV^{2}F on the project page here: https://yininghase.github.io/MA-DV2F/.

The project page also contains additional supplementary information, such as videos better depicting the operation of our method in comparison with other approaches, details of the challenging scenarios, dynamics of the velocity field at different regions on the navigation grid, etc.

II. Related Work

In [16] proposed using Artificial Potential Fields (APF) for trajectory planning. An agent is subjected to an attractive potential force towards the target which serves as the sink and a repulsive potential force away from obstacles [17], [18]. However, a common problem with such methods is their propensity to get stuck in local minima [19], [20] when the attractive force from the target is cancelled out by the repulsive force arising from an another agent for e.g. when the ego-vehicle is symmetrically aligned with other agents to the target [21] and thus leading to a bottleneck situation. We break such bottlenecks by enforcing the vehicles to move in the clockwise direction.

In [13] proposed a two level tree based search algorithm for multi-agent path finding. However, the tree may grow exponentially, making the search inefficient. This is because multi-agent path planning methods on trees and graphs are known to be NP-hard [22] since the search space grows exponentially as the number of agents rise [13]. Nevertheless, [14] used [13] to generate expert data for training a GNN model that can scale up to more vehicles than trained on. [23] uses RL and proposes a local reward function to encourage environmental exploration. However, the need for exploration tends to make the learning sample-inefficient [24], [25] particularly when compared with imitation learning approaches [26]. [27] rather combines RL for single agent path planning with imitation learning to learn actions that can influence other agents. All approaches described above work either on a discrete grid, discrete action space, assume holomonic robots or their combination.

CL-MAPF [9] uses a Body Conflict Tree to describe agent collision scenarios as spatiotemporal constraints. It then applies a Hybrid-State A* Search algorithm to generate paths satisfying both kinematic and spatiotemporal constraints of the vehicles. However, under challenging test scenarios with vehicles crowding together, the algorithm takes long to search for a solution and can easily time out. To find feasible solution for large-scale multi-vehicle trajectory planning, CSDO [10] first searches for a coarse initial guess using a large search step. Then, the Decentralized Quadratic Programming is implemented to refine this guess for minor collisions. GCBF+ [8] based on GCBF [7] aims to provide safety guarantees utilizing control barrier functions (CBF). A Graphical Neural Network is trained to learn agent control policy.

III. Framework

A. Problem Description

We aim to solve the task of multi-agent navigation in unconstrained environments. Given N_{veh} dynamic vehicles and N_{obs} static obstacles in the scene, the task is to ensure each vehicle reaches its desired destination while avoiding collision with other agents. The state vector for ego-vehicle i

$(1 \leq i \leq N_{veh})$ at current time t can be represented as $\mathbf{s}_t^{(i)} = [x_t, y_t, \theta_t, v_t, x_{tar}, y_{tar}, \theta_{tar}]^T$, where x_t and y_t shows the position, θ_t, the orientation and v_t the speed at current time t. Meanwhile, x_{tar} and y_{tar} are co-ordinates of the target position, and θ_{tar} is the target orientation. Each ego-vehicle is controlled by a control vector $\mathbf{c}_t^{(i_{veh})} = [p_t, \varphi_t]^T$, where $p_t \in [-P, P]$ and $\varphi_t \in [-\Phi, \Phi]$ are the pedal acceleration and steering angle, limited in magnitude by P and Φ respectively. The obstacle k $(1 \leq k \leq N_{obs})$ is represented by a state vector $\mathbf{s}^k = [x, y, r]^T$, where x and y is the position, and r is the radius of the circle circumscribing all vertices of the obstacle. The kinematics of the vehicles are modeled using the bicycle model [28].

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

It describes how the equations of motion can be updated in time increments of Δt assuming no slip condition, valid under low or moderate vehicle speed v when making turns. Note that β and γ are the tuneable hyper-parameters modeling the environment friction and vehicle wheelbase, respectively.

We create a velocity vector field for each vehicle, which in turn can dynamically generate reference control variables. Generation of the velocity field can be divided into two stages:

- Estimation of reference orientation of ego-vehicle for every position in the map (See Subsection III-B).
- Estimation of reference speed for the corresponding positions. (See Subsection III-C).

The orientation that a vehicle should possess at any position on the map is referred to as the *Reference Orientation* and is represented as a unit vector. The black arrows in Fig. 1 show the reference orientation for the black ego vehicle at different positions in the map. It can be seen that the arrows attract the ego-vehicle towards its target while repelling it away from the other vehicles and obstacles. Note that the current orientation of the ego-vehicle is not aligned with the reference orientation at its existing position. Therefore, we ought to find the control variables that align the ego-vehicle with the reference orientation. Apart from the reference orientation at each position in the map, the ego-vehicle also has a reference speed which should be attained by the control variables. Lastly, note that apart from the black vehicle, a separate reference orientation map is also created for the blue and maroon vehicles present in Fig. 1 (not shown in the figure).

Hence, our task of multi-agent navigation simplifies to finding the reference orientation maps and corresponding reference speed for each vehicle independently. In the subsequent subsections, we describe how the reference orientation and speed are estimated.

B. Estimation of Reference Orientation

We first define some frequently used functions and vectors before we begin discussion of reference orientation estimation. The vector function $\mathbf{f_{uni}}(\mathbf{a})$ is the function which takes a non-zero vector $\mathbf{a}$ as input and divides by its magnitude to convert it into a unit vector.

Other scalar functions include $f_{sgn}(a)$ and $f_{pos}(a)$ which both output 1 if the scalar input a is positive. However, $f_{sgn}(a)$ outputs -1, while $f_{pos}(a)$ outputs 0 when a is negative. We now define the vector from the next position $(x_{t+1}^{(i)}, y_{t+1}^{(i)})$ of ego vehicle i to

- its target position $(x_{tar}^{(i)}, y_{tar}^{(i)})$ as $\mathbf{X}_{tar}^{(i)} = [x_{tar}^{(i)} - x_{t+1}^{(i)}, y_{tar}^{(i)} - y_{t+1}^{(i)}]^T$
- the position $(x_{obs}^{(k)}, y_{obs}^{(k)})$ of the static obstacle k as $\mathbf{X}_{obs_k}^{(i)} = [x_{obs}^{(k)} - x_{t+1}^{(i)}, y_{obs}^{(k)} - y_{t+1}^{(i)}]^T$
- next position $(x_{t+1}^{(j)}, y_{t+1}^{(j)})$ of another vehicle j as $\mathbf{X}_{veh_j}^{(i)} = [x_{t+1}^{(j)} - x_{t+1}^{(i)}, y_{t+1}^{(j)} - y_{t+1}^{(i)}]^T$

The unit vector along the Z-axis is $\mathbf{Z} = [0, 0, 1]^T$, while the unit orientation vector of the ego vehicle at the current and target states is given by $\mathbf{U}_t^{(i)} = [\cos(\theta_t^{(i)}), \sin(\theta_t^{(i)})]^T$ and $\mathbf{U}_{tar}^{(i)} = [\cos(\theta_{tar}^{(i)}), \sin(\theta_{tar}^{(i)})]^T$ respectively.

For the ego vehicle i, the target reaching component of the reference orientation $\mathbf{u}_{tar}^{(i)}$ is defined as:

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

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

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

where r_p is the parking threshold (black dotted circle in Fig. 1) and $0.5 \cdot v_d^2$ is the marginal parking threshold (shaded blue region in Fig. 1). v_d is the default reference speed, estimation of which is explained in Subsection III-C. ϵ_p is the threshold above which an additional term is introduced when the vehicle is within the parking threshold.

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. In this case, the direction of the orientation is flipped using $f_{sgn}({\mathbf{X}_{tar}^{(i)}}^T . \mathbf{U}_t^{(i)})$. This is to prevent the vehicle from moving in circles. A detailed explanation of this is given in the supplementary file on the project page: https://yininghase.github.io/MA-DV2F/supplementary.pdf.

Equation (2) also shows for the condition when the distance $\|\mathbf{X}_{tar}^{(i)}\|_2$ falls below the parking threshold r_p, and is closer to the target. The ego vehicle should be guided to not only reach the position of the target ($\mathbf{X}_{tar}^{(i)} = 0$) but also be aligned with the target orientation ($\mathbf{U}_{tar}^{(i)} = \mathbf{U}_t^{(i)}$). The balancing act between these two conditions is handled by the $\lambda_{tar}^{(i)}$ term in (2). Like previously, $f_{sgn}({\mathbf{X}_{tar}^{(i)}}^T \cdot \mathbf{U}_{tar}^{(i)})$ in $\lambda_{tar}^{(i)}$ makes sure the reference and target orientations are in the same directions

when parking. The $f_{pos}(\|\mathbf{X}_{tar}^{(i)}\|_2 - \epsilon_p)$ term in $\lambda_{tar}^{(i)}$ is designed to expedite the parking behavior when the vehicle position is exactly on the left or the right side of the target and the vehicle orientation is parallel to the target orientation.

Besides reaching the target, the ego vehicle i should also avoid collision on its way to the target. This is realized by the collision avoiding component $\mathbf{u}_{coll}^{(i)}$ which comprises of collision avoidance between the ego vehicle i and either static obstacle k ($\mathbf{u}_{obs_k}^{(i)}$) or with another vehicle j ($\mathbf{u}_{veh_j}^{(i)}$).

The equation for determining ($\mathbf{u}_{obs_k}^{(i)}$) is given by:

$$\mathbf{u}_{obs_k}^{(i)} = \begin{cases} \mathbf{f_{uni}}(\mathbf{X}_{obs_k}^{(i)}) \cdot \alpha_{obs_k}^{(i)} + \mathbf{R}_{obs_k}^{(i)} \cdot \beta_{obs_k}^{(i)} & \alpha_{obs_k}^{(i)} \leq 0 \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$\alpha_{obs_k}^{(i)} = \|\mathbf{X}_{obs_k}^{(i)}\|_2 - r_{obs}^{(k)} - r_{veh} - (r_c + |v_t^{(i)}|)$$

$$\beta_{obs_k}^{(i)} = f_{pos}({\mathbf{X}_{tar}^{(i)}}^T \cdot \mathbf{X}_{obs_k}^{(i)}) \cdot (\|\mathbf{X}_{obs_k}^{(i)}\|_2 - r_{obs}^{(k)})$$

$$\mathbf{R}_{obs_k}^{(i)} = \mathbf{f_{uni}}(\mathbf{Z} \times \mathbf{X}_{obs_k}^{(i)}) \quad (3)$$

where $r_{obs}^{(k)}$ is the radius of the static obstacle k, r_{veh} is the radius of the smallest circle enclosing the ego-vehicles, r_c is the static component, while $|v_t^{(i)}|$ is the speed-based dynamic safety margin for collision avoidance between ego vehicle i and obstacle k. Higher the vehicle speed $|v_t^{(i)}|$, larger is this margin $r_c + |v_t^{(i)}|$. When the distance between the ego vehicle i and static object k ($\|\mathbf{X}_{obs_k}^{(i)}\|_2 - r_{obs}^{(k)} - r_{veh}$) is below the collision avoidance margin $r_c + |v_t^{(i)}|$, then $\alpha_{obs_k}^{(i)} \leq 0$, and the reference orientation is modified to prevent collision with the static obstacle. Under this condition, the first term $\mathbf{f_{uni}}(\mathbf{X}_{obs_k}^{(i)}) \cdot \alpha_{obs_k}^{(i)}$ will guide the vehicle to drive away from the static obstacle. However, driving away is not enough as this might cause a bottleneck in cases where the obstacle is symmetrically collinear between the ego-vehicle and its target. We would additionally like the ego-vehicle to drive around the obstacle to reach the target for which the term $\mathbf{R}_{obs_k}^{(i)} \cdot \beta_{obs_k}^{(i)}$ is relevant. If the ego-vehicle is between the obstacle and target then $\beta_{obs_k}^{(i)} = 0$ (since $f_{pos}({\mathbf{X}_{tar}^{(i)}}^T \cdot \mathbf{X}_{obs_k}^{(i)}) = 0$) and hence there is no need for the vehicle to drive around the obstacle. However, if that is not the case, then an additional component is added whose direction is given by $\mathbf{R}_{obs_k}^{(i)}$ and magnitude ($\beta_{obs_k}^{(i)}$) is proportional to how far the vehicle is away from the obstacle. $\mathbf{R}_{obs_k}^{(i)}$ is given as the cross product between $\mathbf{Z}$ and $\mathbf{X}_{obs_k}^{(i)}$ with a zero being appended to the third dimension of $\mathbf{X}_{obs_k}^{(i)}$ which originally lies in the 2D space. The vector resulting from the cross-product is perpendicular to $\mathbf{X}_{obs_k}^{(i)}$ which causes the vehicles to move in the clockwise direction around the obstacle as shown in Fig. 2.

Likewise, the component for avoiding collision between the ego vehicle i and another vehicle j ($\mathbf{u}_{veh_j}^{(i)}$) is similar to (3) except that the static obstacle radius $r_{obs}^{(k)}$ will be replaced by the other vehicle's radius r_{veh} in the $\alpha_{veh_j}^{(i)}$ term. Secondly, the speed based dynamic margin is $|v_t^{(j)}| + |v_t^{(i)}|$ rather than just $|v_t^{(i)}|$.

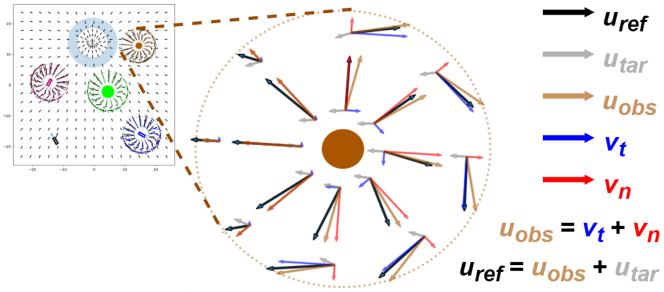


Fig. 2. Shows the constituents of the reference orientation vector $\mathbf{u}_{ref}$ near an obstacle. It comprises of a target approaching ($\mathbf{u}_{tar}$) and the collision avoiding ($\mathbf{u}_{obs_k}$) components. $\mathbf{u}_{obs_k}$ includes $\mathbf{v}_n = \mathbf{f_{uni}}(\mathbf{X}_{obs_k}^{(i)}) \cdot \alpha_{obs_k}^{(i)}$ guiding the vehicle to drive away from the obstacle and $\mathbf{v}_t = \mathbf{R}_{obs_k}^{(i)} \cdot \beta_{obs_k}^{(i)}$ leading the vehicle to go around the obstacle. When the ego-vehicle is between the obstacle and target, it is not necessary for it to go around the obstacle and thus $\mathbf{v}_t = \mathbf{0}$. The formulation to calculate $\mathbf{u}_{obs_k}$ is described in (3).

The overall collision avoiding component $\mathbf{u}_{coll}^{(i)}$ is given by:

$$\mathbf{u}_{coll}^{(i)} = \sum_{k=1}^{N_{obs}} \mathbf{u}_{obs_k}^{(i)} + \sum_{j=1, j\neq i}^{N_{veh}} \mathbf{u}_{veh_j}^{(i)} \quad (4)$$

Finally, the ideal reference orientation vector $\hat{\mathbf{u}}_{t+1}^{(i)}$ ($\mathbf{u_{ref}}$ in Fig. 1) is:

$$\hat{\mathbf{u}}_{t+1}^{(i)} = \mathbf{f_{uni}}(\mathbf{u}_{tar}^{(i)} + \mathbf{u}_{coll}^{(i)}) \quad (5)$$

From this, the corresponding ideal reference orientation angle $\hat{\theta}_{t+1}^{(i)}$ for ego vehicle i can be calculated by applying $arctan2$ to $\hat{\mathbf{u}}_{t+1}^{(i)}$. However, kinematic constraints arising from the motion model ((1)) may prevent the vehicle from immediately attaining the ideal reference orientation $\hat{\theta}_{t+1}^{(i)}$ in the next time step. Therefore, we instead use $\theta_{t+1}^{(i)}$ referred to as the *real* orientation. It is the reachable orientation closest to $\hat{\theta}_{t+1}^{(i)}$. The unit vector corresponding to this real reference orientation angle $\theta_{t+1}^{(i)}$ for ego vehicle i is $\mathbf{u}_{t+1}^{(i)}$ ($\mathbf{u_{real}}$ in Fig. 1) $= [\cos(\theta_{t+1}^{(i)}), \sin(\theta_{t+1}^{(i)})]^T$.

C. Estimation of Reference Speed

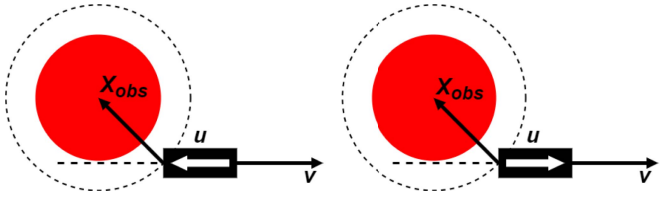
The reference speed $v_{t+1}^{(i)}$ is chosen after determination of the reference orientation, which depends on the situation the vehicle is in. Fig. 3 shows a scenario wherein a vehicle is at the same location next to an obstacle but with opposite orientations. This determines if the velocity should move the car forward or backward.

We use the logical *or* ($\vee$) and logical *and* ($\wedge$) operators [29] to describe the criteria for collision avoidance between ego vehicle i and static obstacle k under such situations as:

$$F_{obs_k}^{(i)} = (\alpha_{obs_k}^{(i)} + \epsilon_c \leq 0) \wedge (\gamma_{obs_k}^{(i)} > 0)$$

$$B_{obs_k}^{(i)} = (\alpha_{obs_k}^{(i)} + \epsilon_c \leq 0) \wedge (\gamma_{obs_k}^{(i)} < 0)$$

$$\gamma_{obs_k}^{(i)} = {\mathbf{u}_{t+1}^{(i)}}^T \cdot \mathbf{X}_{obs_k}^{(i)} \quad (6)$$



(a) Forbidden Forward (b) Forbidden Backward

Fig. 3. Shows two different scenarios when the black ego vehicle is in a collision avoiding region. In (a), the ego vehicle is facing the obstacle, i.e. $\mathbf{u}^T \cdot \mathbf{X}_{obs} > 0$, in which case the vehicle should be forbidden to move forwards. In (b), the ego vehicle is oriented away from the obstacle, i.e. $\mathbf{u}^T \cdot \mathbf{X}_{obs} < 0$, in which case the vehicle should be prevented from moving backward.

where $\alpha^{(i)}_{obs_k}$ is the same as defined in (3), ϵ_c is an additional tolerance for the collision checking region. $F^{(i)}_{obs_k}$ equalling $true$ indicates the ego vehicle i is forbidden to drive forward towards obstacle k. This happens when the angle between the reference orientation ($\mathbf{u}^{(\mathbf{i})}_{\mathbf{t+1}}$) and the vector from the ego vehicle to the obstacle ($\mathbf{X}^{(i)}_{obs_k}$) is less than $90°$ i.e. $\gamma^{(i)}_{obs_k} = {\mathbf{u}^{(i)}_{t+1}}^T \cdot \mathbf{X}^{(i)}_{obs_k} > 0$. Likewise, $B^{(i)}_{obs_k}$ equaling $true$ happens when $\gamma^{(i)}_{obs_k} = {\mathbf{u}^{(i)}_{t+1}}^T \cdot \mathbf{X}^{(i)}_{obs_k} < 0$ indicating that the ego vehicle i is forbidden to drive backward.

Similarly for the case of ego vehicle i and another vehicle j, the conditions for preventing the ego-vehicle from moving forward $F^{(i)}_{veh_j}$ or backward $B^{(i)}_{veh_j}$ are the same as described in (6) except that $\mathbf{X}^{(i)}_{veh_j}$ replaces $\mathbf{X}^{(i)}_{obs_k}$.

Summarising the results above, the magnitude and sign of the velocity depends on the combination of these boolean variables i.e. $F^{(i)}_{obs_k}$, $F^{(i)}_{veh_j}$, $B^{(i)}_{obs_k}$ and $B^{(i)}_{veh_j}$ as follows:

$$F^{(i)}_{coll} = \left(\bigvee_{k=1}^{N_{obs}} F^{(i)}_{obs_k}\right) \vee \left(\bigvee_{j=1, j\neq i}^{N_{veh}} F^{(i)}_{veh_j}\right)$$

$$B^{(i)}_{coll} = \left(\bigvee_{k=1}^{N_{obs}} B^{(i)}_{obs_k}\right) \vee \left(\bigvee_{j=1, j\neq i}^{N_{veh}} B^{(i)}_{veh_j}\right)$$

$$v^{(i)}_{coll} = \begin{cases} v_d & B^{(i)}_{coll} \wedge (\neg F^{(i)}_{coll}) \\ -v_d & (\neg B^{(i)}_{coll}) \wedge F^{(i)}_{coll} \\ 0 & B^{(i)}_{coll} \wedge F^{(i)}_{coll} \\ v^{(i)}_{tar} & \text{otherwise} \end{cases} \tag{7}$$

The first 3 conditions check whether or not there is a potential for collision. The velocity is $+v_d$, when the ego-vehicle is prevented from moving backwards ($B^{(i)}_{coll}$) but allowed to move forward ($\neg F^{(i)}_{coll}$) and $-v_d$ when vice versa. The reference velocity is zero when the ego-vehicle is prevented from moving both forward ($F^{(i)}_{coll}$) and backward ($B^{(i)}_{coll}$). In all other cases, the velocity is $v^{(i)}_{tar}$ defined as:

$$v^{(i)}_{tar} = \begin{cases} \xi^{(i)}_p \cdot \lambda^{(i)}_p \cdot v_d & \|\mathbf{X}^{(i)}_{tar}\|_2 \leq r_p \\ v_d \cdot f_{sgn}({\mathbf{u}^{(i)}_{t+1}}^T \cdot \hat{\mathbf{u}}^{(i)}_{t+1}) & \text{otherwise} \end{cases}$$

$$\lambda^{(i)}_p = \begin{cases} \bar{\lambda}^{(i)}_p & (\|\mathbf{X}^{(i)}_{tar}\|_2 < \epsilon_p) \wedge (|\theta^{(i)}_{tar} - \theta^{(i)}_{t+1}| < \epsilon_o) \\ \sqrt{\bar{\lambda}^{(i)}_p} & \text{otherwise} \end{cases}$$

$$\bar{\lambda}^{(i)}_p = \text{minimum}\left(\frac{\|\mathbf{X}^{(i)}_{tar}\|_2}{r_p} + \frac{|\theta^{(i)}_{tar} - \theta^{(i)}_{t+1}|}{v_d}, 1\right)$$

$$\xi^{(i)}_p = \begin{cases} 1 & {\mathbf{u}^{(i)}_{t+1}}^T \cdot \mathbf{X}^{(i)}_{tar} > 0.25 \\ -1 & {\mathbf{u}^{(i)}_{t+1}}^T \cdot \mathbf{X}^{(i)}_{tar} < -0.25 \\ f_{sgn}(\mathbf{u}^{(i)}_t) & \text{otherwise} \end{cases} \tag{8}$$

Where ϵ_p and ϵ_o are the acceptable position and orientation tolerances for deciding to stop the vehicle at the target. When the ego vehicle is within the parking radius, it reduces its speed as it gets closer to the target position and orientation. This is achieved by the multiplier $\lambda^{(i)}_p$, which is proportional to $\bar{\lambda}^{(i)}_p$ when the vehicle state is very close to the target state and $\sqrt{\bar{\lambda}^{(i)}_p}$ when farther away from the tolerance. The square root accelerates the vehicle approaching its target when there is still some distance between the current state and target state, i.e. $\neg((\|\mathbf{X}^{(i)}_{tar}\|_2 < \epsilon_p) \wedge (|\theta^{(i)}_{tar} - \theta^{(i)}_{t+1}| < \epsilon_o))$. When the vehicle is very close to the target state, i.e. $(\|\mathbf{X}^{(i)}_{tar}\|_2 < \epsilon_p) \wedge (|\theta^{(i)}_{tar} - \theta^{(i)}_{t+1}| < \epsilon_o)$, the ratio $\bar{\lambda}^{(i)}_p$ prevents the vehicle from shaking forward and backward. For $\bar{\lambda}^{(i)}_p$, the first term: $\frac{1}{r_p} \cdot \|\mathbf{X}^{(i)}_{tar}\|_2$, decreases linearly with the ego vehicle distance to its target, and the second term, i.e. $\frac{1}{v_d} \cdot |\theta^{(i)}_{tar} - \theta^{(i)}_{t+1}|$, reduces linearly with the angle difference between the reference $\theta^{(i)}_{t+1}$ and target $\theta^{(i)}_{tar}$ orientation angles. However, we do not allow the reference parking speed $v^{(i)}_{tar}$ to be any higher than the default reference speed v_d. So, $\bar{\lambda}^{(i)}_p$ is clipped to a maximum of 1. $\xi^{(i)}_p$ controls whether the ego vehicle moves forward or backward by checking ${\mathbf{u}^{(i)}_{t+1}}^T \cdot \mathbf{X}^{(i)}_{tar}$. Originally, the vehicle should move forward when facing the target, i.e. ${\mathbf{u}^{(i)}_{t+1}}^T \cdot \mathbf{X}^{(i)}_{tar} > 0$, and move backward when backing towards the target, i.e. ${\mathbf{u}^{(i)}_{t+1}}^T \cdot \mathbf{X}^{(i)}_{tar} < 0$. However, to prevent the vehicle from changing direction at high frequency within short traveling distances, we set a margin allowing the vehicle to keep its previous direction when $|{\mathbf{u}^{(i)}_{t+1}}^T \cdot \mathbf{X}^{(i)}_{tar}| \leq 0.25$.

When the ego vehicle i is out of the parking area of radius r_p, the reference speed $v^{(i)}_{t+1}$ takes the forn $v_d \cdot f_{sgn}({\mathbf{u}^{(i)}_{t+1}}^T \cdot \hat{\mathbf{u}}^{(i)}_{t+1})$, where the reference speed $v^{(i)}_{t+1}$ takes the default value v_d, but changes to negative, i.e. $-v_d$, when ${\mathbf{u}^{(i)}_{t+1}}^T \cdot \hat{\mathbf{u}}^{(i)}_{t+1} < 0$.

The final ideal reference speed $\hat{v}^{(i)}_{t+1}$ for ego vehicle i is:

$$\hat{v}^{(i)}_{t+1} = v^{(i)}_{coll} \tag{9}$$

Similar to the reference orientation, the ideal reference speed $\hat{v}^{(i)}_{t+1}$ may not be achievable due to the limitation of the maximum pedal command. The real reference speed $v^{(i)}_{t+1}$ is therefore the reachable speed value closest to $\hat{v}^{(i)}_{t+1}$.

TABLE I
SHOWS THE *SUCCESS* RATE METRIC (HIGHER IS BETTER) FOR DIFFERENT MODELS, I.E. OUR MA-DV^{2}F, OUR SELF-SUPERVISED GNN MODEL, CSDO [10], CL-MAPF [9], GCBF+ [8] AND THE SUPERVISED GNN MODEL [12]

Number of Vehicles	Number of Obstacles	MA-DV^{2}F (Ours)	Self-Supervised Model (Ours)	CSDO [10]	CL-MAPF [9]	GCBF+ [8]	GCBF+ (only position)	Supervised Model [12]
success rate ↑								
10	0	**1.0000**	0.9929	0.9693	0.4290	0.0613	0.9021	0.7285
20	0	**1.0000**	0.9895	0.9400	0.4963	0.0563	0.8169	0.2041
30	0	**1.0000**	0.9793	0.8820	0.4977	0.0458	0.7550	0.0477
40	0	**1.0000**	0.9760	0.7063	0.5632	0.0451	0.7080	0.0137
50	0	**1.0000**	0.9686	0.6523	0.5992	0.0426	0.6618	0.0045
10	25	**0.9952**	0.9458	0.9682	0.5640	0.0509	0.7192	0.3734
20	25	**0.9902**	0.9208	0.9663	0.5997	0.0393	0.6479	0.0756
30	25	**0.9844**	*0.8998*	0.8456	0.6320	0.0385	0.5754	0.0196
40	25	**0.9772**	*0.8723*	0.6723	0.6531	0.0330	0.5211	0.0068
50	25	**0.9704**	0.8504	0.5897	0.6749	0.0290	0.4789	0.0028

As GCBF+ does not consider vehicle orientation when reaching the target, we additionally evaluate the Success rate by only checking vehicle position and disregarding the orientation. For each row, the bold number show the highest success rate among all algorithms.

Calculation of Reference Steering Angle and Reference Pedal: Given the reference orientation and velocity, the Vehicle Kinematic (1) can be inverted to determine the reference steering angle $\varphi_t^{(i)}$ and pedal acceleration $p_t^{(i)}$ for controlling the vehicle at time t :

$$\varphi_t^{(i)} = arctan2\left(\frac{(\theta_{t+1}^{(i)} - \theta_t^{(i)})}{v_t^{(i)} \cdot \gamma \cdot \Delta t}\right)$$
$$p_t^{(i)} = \frac{v_{t+1}^{(i)} - \beta \cdot v_t^{(i)}}{\Delta t} \tag{10}$$

These reference control commands can either be directly used to control the vehicles or as labels for training the GNN model in a self-supervised manner using the architecture of [12].

IV. EXPERIMENTS

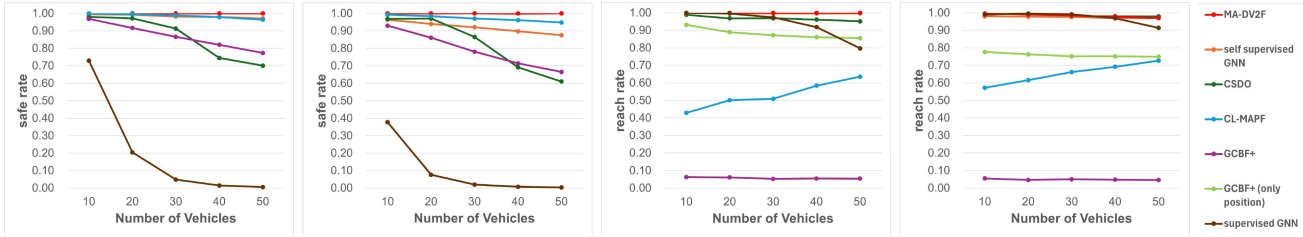
A. Algorithm Evaluation

To assess the performance of our MA-DV^{2}F approach and its learning based counterpart (self-supervised GNN model), we compare with other recent learning and search/optimization based algorithms in challenging, collision prone test cases. The recent learning based approaches include [12] which is a GNN model trained in a supervised manner and an extension of [7] i.e. GCBF+ [8], also a GNN model incorporating safety constraints using Control Barrier Functions (CBF). Meanwhile, the two search/optimization based algorithms used for comparison include CL-MAPF [9] and CSDO [10]. The test dataset comprises of challenging, collision prone scenarios with the number of vehicles ranging from 10 to 50 and static obstacles chosen between 0 and 25. Note that all the static obstacles in the test cases are fixed to be circles with fixed radii because of the limitation of the CL-MAPF and CSDO environment setting and because it circumscribes an obstacle of arbitrary shape, allowing for safer behaviour. For the GCBF+, we use the DubinsCar model because it has vehicle kinematics that are most similar to ours. As GCBF+ has an entirely different working domain with different map and agent sizes, we scale our test cases when running GCBF+ under the rule that the ratio of the map to agent size remains equal across all the algorithms. Details regarding the generation of test samples are given in the supplementary file on the project page.

Table I shows the evaluation results of the different methods across the various vehicle-obstacle combinations described earlier against the *success rate* metric [8]. It measures the percentage of vehicles that successfully reach their targets within a tolerance without colliding with other objects along the way. The results show that our MA-DV^{2}F outperforms other algorithms achieving almost 100% success rate across all vehicle-obstacle combinations. Even the self-supervised GNN model performs far better than other algorithms when scaling the number of vehicles. Only the CSDO algorithm has slightly better results than our GNN model when the number of agents is low (20). However, CSDO's performance drops drastically as the number of agents are increased under these challenging conditions. Note that the GCBF+ pipeline only checks whether the agents reach their target positions but ignores the target orientations as shown in the project page: https://yininghase.github.io/MA-DV2F/#MC which explains why it has such poor performance. Therefore, we additionally evaluate GCBF+ by ignoring the orientation and only considering the position. Results of which are shown in the second last column. Even then, the GCBF+, does not match the performance of MA-DV^{2}F.

B. Discussion

Investigating failure Causes: Note that the *Success rate* metric measures a model's ability to not only reach its destination but also avoid collisions. Therefore, for challenging test cases, a navigation algorithm may fail due to two main reasons: the algorithms behave too aggressively by driving the vehicles to their targets, albeit at the cost of colliding with other agents or behaves too conservatively to ensure safety of the vehicles resulting in some vehicles getting stuck mid-way without reaching their targets. Therefore, to disambiguate between the two causes, we additionally evaluate all algorithms against the *Reach rate* and *Safe rate* metrics as proposed in [8]. *Reach rate* measures the percentage of vehicles that reach their targets within a tolerance



(a) safe rate with no obstacles (b) safe rate with 25 obstacles (c) reach rate with no obstacles (d) reach rate with 25 obstacles (e) Legend

Fig. 4. Shows the *Safe* and *Reach* rate metrics (Higher is better) for the different models, i.e. our MA-DV^{2}F, our self-supervised GNN model, supervised GNN model [12], CSDO [10], CL-MAPF [9] and GCBF+ [8].

disregarding any collisions along the way. Meanwhile, the *Safe rate* calculates the percentage of vehicles that do not collide with any other agents irrespective of whether or not they reach their targets. Fig. 4 shows the performance of the different methods against the *Reach Rate* and *Safe Rate* metrics. It can be seen that the supervised GNN model behaves rather aggressively reaching the target in majority of the cases albeit at the cost of multiple collisions leading to high *Reach* but low *Safe* rates. In contrast, CL-MAPF [9] takes a greedy strategy in its optimization/search pipeline. It can quickly find the path for some vehicles which are easy to plan. But then it is unable to find paths for other vehicles and gets stuck in this partial solution. This greedy strategy lead to a low reach rate, but high safe rate since no collisions happen among vehicles that do not move.

GCBF+ has a higher safe rate than reach rate. This is not because the vehicles fail to reach their target, but rather because they reach the target at an incorrect orientation. This is aligned with the intuition described in Section IV-A and is further corroborated by the fact that when orientation is ignored in the evaluation and only position is considered, the reach rate jumps drastically. Nevertheless, it is still much lower than both our MA-DV^{2}F and its GNN self-supervised counterpart. They are the only 2 methods that maintain a consistently high performance against both metrics across all vehicle-obstacle combinations.

Preventing Bottlenecks: A common problem with other planning algorithms is that vehicles tend to crowd within a small sub-region in the map. This leads to these algorithms either being unable to find an optimal path, resulting in all vehicles becoming stationary at their place (low reach rate), or finding sub-optimal paths, resulting in more collisions among vehicles (low safe rate). To prevent such bottlenecks in our MA-DV^{2}F model, (3) causes all the vehicles to drive in the clockwise direction when encountering other agents. This leads to a roundabout driving behavior which breaks the bottleneck and can be visualized on the project page: https://yininghase.github.io/MA-DV2F/#RE. Due to this, the vehicles are capable of eventually reaching their targets,by making a detour around the other agents, resulting in both a high *Reach* and *Safe* rate. Our trained GNN model also adapts this behaviour.

Intermediate Success Rate: One might wonder if the MA-DV^{2}F method outperforms every other method, what is the advantage of its self-supervised GNN model counterpart? One reason is that MA-DV^{2}F behaves conservatively in some simple test cases even though there is no collision risk nearby, causing

TABLE II
SHOWS THE TOTAL RUNTIME IN SECONDS (LOWER IS BETTER) FOR THE 1000 TEST CASES FOR EACH VEHICLE-OBSTALCE COMBINATION OF THE DIFFERENT MODELS, I.E. OUR MA-DV^{2}F, CSDO [10] AND CL-MAPF [9]

Number of Vehicles	Number of Obstacles	MA-DV^{2}F (Ours)	CSDO [10]	CL-MAPF [9]
total runtime for 1000 test cases ↓				
10	0	***4.152***	195.919	125643.851
20	0	***5.848***	1044.360	219251.537
30	0	***7.885***	6179.536	328471.949
40	0	***8.492***	12342.014	376417.421
50	0	***10.512***	18306.779	395962.103
10	25	***10.322***	352.096	89838.921
20	25	***12.260***	2213.838	163001.473
30	25	***13.733***	9148.955	190907.288
40	25	***15.562***	14350.735	224962.016
50	25	***18.405***	16598.068	253495.187

the vehicles to take a longer time to finish the journey. This is because the speed is limited to v_d. On the other hand, the self-supervised GNN counterpart is free from this restriction. It can move faster when it is far away from its target, and there is less risk of collision with other objects. The figures on the project page: https://yininghase.github.io/MA-DV2F/#ISR show the difference in the success rate between the self-supervised GNN model and MA-DV^{2}F as a function of time. At the beginning, when the vehicles tend to be far away from their targets, the GNN allows high velocity for vehicles, thereby causing some vehicles to reach their target faster leading to a success rate greater than that for MA-DV^{2}F. This can also seen in the example on the project page: https://yininghase.github.io/MA-DV2F/#MC, the vehicles driven by a GNN can drive faster towards the targets than that by MA-DV^{2}F. However, maintaining a high velocity also leads to a higher risk of collision when encountering challenging situations as time progresses and the success rate of MA-DV^{2}F will gradually catch up and eventually exceed the GNN.

Runtime Analysis: We compare the runtime of MA-DV^{2}F with concurrent search based methods (CSDO and CL-MAPF). TABLE II shows the total runtime of all the 1000 test cases for each vehicle-obstalce combinations. All methods were evaluated on a machine with an Intel Core i7-10750H CPU and GeForce RTX 2070 GPU. As can be seen, our MA-DV^{2}F, is orders of magnitude faster than its peers, since it has the ability to run the different scenarios within a batch in parallel. Meanwhile, CSDO and CL-MAPF are search/optimization-based algorithms that need to solve each test case one-by-one. Note that CL-MAPF

would continue its search until a solution is found, which is why the maximum allowed runtime needs to be clipped, otherwise the runtime would be even larger. Note that the evaluations in the experiments were done for only up to 50 vehicles since other algorithms are either extremely slow or have drastically reduced performance when scaling. However, our method is capable of scaling to a far greater number of vehicles than just 50, as can be observed on the project page for scenarios with 100, 125 and 250 vehicle-obstacle combinations: https://yininghase.github.io/MA-DV2F/#LE.

Lastly, note that the runtime analysis is not done for the learning based methods since, it is dependent on the GPU rather than the algorithm itself. Therefore, for the same model architecture, the runtime will be the same for all learning based algorithms.

Limitations: If the vehicles are densely packed or their targets are within the safety margins of one another, then due to their non-holonomic behavior there might not be enough space for them to navigate without risking a collision. In such a scenario, the vehicles will act conservatively and hesitate to proceed so as to avoid collisions, leading to fewer vehicles reaching the target. A similar outcome is observed, if some vehicles start behaving unexpectedly, wherein the safety margin would need to be increased to mitigate the risk of collision, albeit at the expense of reaching the target. More details on sensitivity analysis of the effect of change in the static component of the safety margin (r_c) and visualization of the limitations are in the supplementary file and the project page.

V. Conclusion

This work introduced MA-DV^{2}F, an efficient algorithm for multi-vehicle navigation using dynamic velocity vector fields. Experimental results showed that our approach seamlessly scales with the number of vehicles. When compared with other concurrent learning and search based methods, MA-DV^{2}F has a higher success rate, lower collision metrics and higher computational efficiency.

Acknowledgment

The authors thank Ang Li for the discussions and his feedback on the setup of the experiments.

References

[1] X. Cao, M. Li, Y. Tao, and P. Lu, "HMA-SAR: Multi-agent search and rescue for unknown located dynamic targets in completely unknown environments," *IEEE Robot. Automat. Lett.*, vol. 9, no. 6, pp. 5567–5574, Jun. 2024.

[2] L. Bartolomei, L. Teixeira, and M. Chli, "Fast multi-UAV decentralized exploration of forests," *IEEE Robot. Automat. Lett.*, vol. 8, no. 9, pp. 5576–5583, Sep. 2023.

[3] F. Kudo and K. Cai, "A TSP-based online algorithm for multi-task multi-agent pickup and delivery," *IEEE Robot. Automat. Lett.*, vol. 8, no. 9, pp. 5910–5917, Sep. 2023.

[4] B. Li and H. Ma, "Double-deck multi-agent pickup and delivery: Multi-robot rearrangement in large-scale warehouses," *IEEE Robot. Automat. Lett.*, vol. 8, no. 6, pp. 3701–3708, Jun. 2023.

[5] D. Zhu, Q. Khan, and D. Cremers, "Multi-vehicle trajectory prediction and control at intersections using state and intention information," *Neurocomputing*, vol. 574, 2024, Art. no. 127220.

[6] B. Nebel, "On the computational complexity of multi-agent pathfinding on directed graphs," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2020, pp. 212–216.

[7] S. Zhang, K. Garg, and C. Fan, "Neural graph control barrier functions guided distributed collision-avoidance multi-agent control," in *Proc. Conf. Robot Learn.*, 2023, pp. 2373–2392.

[8] S. Zhang, O. So, K. Garg, and C. Fan, "GCBF+: A neural graph control barrier function framework for distributed safe multi-agent control," *IEEE Trans. Robot.*, vol. 41, pp. 1533–1552, 2025.

[9] L. Wen, Y. Liu, and H. Li, "CL-MAPF: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints," *Robot. Auton. Syst.*, vol. 150, 2022, Art. no. 103997.

[10] Y. Yang, S. Xu, X. Yan, J. Jiang, J. Wang, and H. Huang, "CSDO: Enhancing efficiency and success in large-scale multi-vehicle trajectory planning," *IEEE Robot. Automat. Lett.*, early access, Aug. 2024, doi: 10.1109/LRA.2024.3440091.

[11] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *A Bradford Book*, Cambridge, MA, USA, 2018.

[12] Y. Ma, Q. Khan, and D. Cremers, "Multi agent navigation in unconstrained environments using a centralized attention based graphical neural network controller," in *Proc. IEEE 26th Int. Conf. Intell. Transp. Syst.*, 2023, pp. 2893–2900.

[13] G. Sharon et al., "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, 2015.

[14] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 11785–11792.

[15] Y. Ma, A. Li, Q. Khan, and D. Cremers, "Enhancing the performance of multi-vehicle navigation in unstructured environments using hard sample mining," 2024, *arXiv:2409.05119*.

[16] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. 1985 IEEE Int. Conf. Robot. Automat.*, 1985, vol. 2, pp. 500–505.

[17] M. Reda, A. Onsy, A. Y. Haikal, and A. Ghanbari, "Path planning algorithms in the autonomous driving system: A comprehensive review," *Robot. Auton. Syst.*, vol. 174, 2024, Art. no. 104630.

[18] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," *Eng. Sci. Technol., Int. J.*, vol. 40, 2023, Art. no. 101343.

[19] Á. Madridano, A. Al-Kaff, D. Martín, and A. De La Escalera, "Trajectory planning for multi-robot systems: Methods and applications," *Expert Syst. Appl.*, vol. 173, 2021, Art. no. 114660.

[20] J. Hou et al., "Large-scale vehicle platooning: Advances and challenges in scheduling and planning techniques," *Engineering*, vol. 28, pp. 26–48, 2023.

[21] I. Iswanto et al., "Artificial potential field algorithm implementation for quadrotor path planning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 8, pp. 575–585, 2019.

[22] J. Yu and S. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proc. AAAI Conf. Artif. Intell.*, 2013, vol. 27, no. 1, pp. 1443–1449.

[23] H.-T. Wai, Z. Yang, Z. Wang, and M. Hong, "Multi-agent reinforcement learning via double averaging primal-dual optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/5a378f8490c8d6af8647a753812f6e31-Paper.pdf

[24] D. Li, D. Zhao, Q. Zhang, and Y. Chen, "Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]," *IEEE Comput. Intell. Mag.*, vol. 14, no. 2, pp. 83–98, May 2019.

[25] N. V. Varghese and Q. H. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics*, vol. 9, no. 9, 2020, Art. no. 1363.

[26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.

[27] G. Sartoretti et al., "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019.

[28] D. Wang and F. Qi, "Trajectory planning for a four-wheel-steering vehicle," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2001, pp. 3320–3325.

[29] M. M. Mano, *Digital Logic and Computer Design*. Tamil Nadu, India: Pearson Education, 2017.