



**HAL**  
open science

# Multi-UAVs end-to-end Distributed Trajectory Generation over Point Cloud Data

Antonio Marino, Claudio Pacchierotti, Paolo Robuffo Giordano

► **To cite this version:**

Antonio Marino, Claudio Pacchierotti, Paolo Robuffo Giordano. Multi-UAVs end-to-end Distributed Trajectory Generation over Point Cloud Data. *IEEE Robotics and Automation Letters*, 2024, 9 (9), pp.7629-7636. 10.1109/LRA.2024.3426374 . hal-04552877v2

**HAL Id: hal-04552877**

**<https://hal.science/hal-04552877v2>**

Submitted on 28 Jun 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

# Multi-UAVs end-to-end Distributed Trajectory Generation over Point Cloud Data

Antonio Marino, Claudio Pacchierotti, Paolo Robuffo Giordano

**Abstract**—This paper introduces an end-to-end trajectory planning algorithm tailored for multi-UAV systems that generates collision-free trajectories in environments populated with both static and dynamic obstacles, leveraging point cloud data. Our approach consists of a 2-branch neural network fed with sensing and localization data, able to communicate intermediate learned features among the agents. One network branch crafts an initial collision-free trajectory estimate, while the other devises a neural collision constraint for subsequent optimization, ensuring trajectory continuity and adherence to physical actuation limits. Extensive simulations in challenging cluttered environments, involving up to 25 robots and 25% obstacle density, show a collision avoidance success rate in the range of 100 – 85%. Finally, we introduce a saliency map computation method acting on the point cloud data, offering qualitative insights into our methodology.

**Index Terms**—distributed control, graph neural network, trajectory generation

## I. INTRODUCTION

The domain of multi-agent Unmanned Aerial Vehicle (UAV) trajectory planning has garnered significant attention, given its diverse range of applications [1], [2], [3]. In these settings, planning algorithms play a pivotal role in calculating trajectories that are both safe and directed towards a defined goal. These algorithms have to consider the dynamic state of the environment and the presence of neighbouring agents. In practical applications involving drones, trajectory planning is crucial to navigate around obstacles and accommodate limitations in the drone’s actuators [4], [5], [6].

Despite possessing more theoretical guarantees, centralized approaches are often less appealing than decentralized counterparts due to their computationally intensive nature and the dependence on full-state information of each robot at every algorithm iteration which renders them impractical for real-world execution [7]. In contrast, decentralized planning not only demonstrates enhanced scalability but also provides robustness against potential failures in a centralized architecture [8].

For more effective coordinated planning, the planning algorithm must consider not only local sensing data but also the planning decisions of a few neighbouring team members [9].

Thus, communication emerges as a critical element in realizing distributed solutions for multi-agent systems. Within this context, one of the focuses of this article is to integrate local sensing and communication strategies to achieve end-to-end distributed multi-UAV motion planning.

For UAV navigation, learning-based control from point clouds or images offers computational advantages [10], [11], [12]. Recently, distributed learning methods for multi-agent scenarios have emerged, using reinforcement learning [13], [14], [15] and Graph Neural Networks (GNNs) [16], [17]. GNNs, by exploiting the communication graph, excel in encoding distributed policies, as demonstrated by Blumenkamp et al. [18], who showed multi-robot coordination in narrow passages. However, learning-based control needs additional safety guarantees, as seen in GLAS [19], which uses a local safety function to ensure stability and safety. Most results apply to static environments without considering real robot dynamics. A recent work [20] combines safety control strategies with learning over a graph of agents and obstacles, using a GNN-based network to predict control estimates and a constraint function for Control Barrier Function (CBF) optimization from LiDAR data.

However, all these methods are susceptible to trapping robots in local minima as they compute only a local control action. To explicitly tackle this problem, a contribution of this work is to propose a trajectory planning algorithm that takes into account spatio-temporal predictions and can avoid trapping the robots in deadlocks. Planning for multi-drone coordination inherently poses a high-dimensional challenge, even when safety is the sole requirement. Moreover, planning algorithms often necessitate access to map data for obstacle information, as noted in prior work [21]. This map should be updated online to tackle environmental changes which in large-scale environments can become cumbersome, bringing a computational bottleneck. Furthermore, optimization-based approaches may need to relax collision constraints as team density increases. Potentially, this relaxation leads to collisions, as discussed in [22] which reports real-time motion planning for a swarm of up to 20 drones. The recent literature [23], [24], [25], [26] has presented notable results for robust multi-drone large-space travelling that take into account physical size, actuator limitations, alongside tracking disturbances, communication delay, and asynchronous communication. However, these algorithms typically rely on perfect sensing and tracking of obstacles at all times or a pre-available obstacle map, which, generally, is unavailable in real-world scenarios. Furthermore, these algorithms require tuning the map grid size to find collision-free paths and, despite implementing a decentralized approach, each drone must communicate with all other drones in the team, which

Manuscript received: February 26, 2024; Revised: April 23, 2024; Accepted: June 15, 2024.

This paper was recommended for publication by Giuseppe Loianno upon evaluation of the Associate Editor and Reviewers’ comments.

A. Marino is with Univ Rennes, CNRS, Inria, IRISA – Rennes, France. E-mail: antonio.marino@irisa.fr

C. Pacchierotti and P. Robuffo Giordano are with CNRS, Univ Rennes, Inria, IRISA – Rennes, France. E-mail: {claudio.pacchierotti,prg}@irisa.fr

This work was supported by the ANR-20-CHIA-0017 project “MULTI-SHARED”

Digital Object Identifier (DOI): see top of this page.

reduces scalability.

In this paper, we present a novel data-driven approach for distributed trajectory generation and safety collision avoidance under LiDAR-based observations. Our main contributions are:

- A decentralized and asynchronous end-to-end trajectory planning method using attention-based GNNs to learn a decentralized policy from point cloud data. Unlike existing methods requiring prior knowledge of the environment, our approach allows each drone to plan its trajectory directly from sensed data.
- An optimization process that generates collision-free trajectories by predicting collision constraints from point cloud data and inter-drone communication. This method allows real-time updates and adjustments, ensuring higher reliability and safety in dynamic, uncertain environments.
- Leveraging the point cloud saliency map computed by a variation of VisBackProp [27], we propose a qualitative analysis of the predicted trajectories and how the point cloud contributes to it.

## II. PROBLEM STATEMENT

Consider a distributed trajectory generation problem for a set of UAVs  $V := \{1, \dots, N\}$  modelled at time  $t$  in  $SE(3)$  with second-order dynamics. The UAV team operates within a three-dimensional workspace containing static and dynamic obstacles  $\mathcal{O}_t$ . Each drone's objective is to navigate the environment toward its target without colliding with other agents. A team mission is the collective traversing of the space such that every drone satisfies its objective. We make the following assumptions

**Assumption 1.** *Each drone is controlled via a trajectory-tracking algorithm and equipped with a 3D range sensor, such as a LiDAR, with 360° field of view.*

**Assumption 2.** *At time  $t$ , we assume perfect state estimation, i.e. the quaternion orientation  $\mathbf{quat}_i(t)$  and the triplets of position  $\mathbf{r}_i(t)$ , velocity  $\mathbf{v}_i(t)$  in world frame are available to agent  $i$ .*

**Assumption 3.** *Each drone can communicate with a limited number of neighbours at a given frequency without communication loss*

The observation data for a drone  $i$  is denoted by  $pc_i \in R^{m \times 3}$ , which includes the relative positions of the  $m$  "hit" points in the scanned environment. The trajectory generated must comply with the following constraints:

- **collision avoidance:** each pair of drones and drone-obstacle maintains a safety distance of  $2d$  throughout the whole trajectory, where  $d > 0$  is the radius of a sphere containing the physical body of the agents.
- **limited sensing and communication:** each agent has a limited sensing and communication range radius  $R$ . We define the neighbours of agent  $i$  as  $\mathcal{N}_i = \{j \in V \mid \|\mathbf{r}_i - \mathbf{r}_j\|_2 \leq R, j \neq i\}$ ; therefore, the agents can only sense other agents or obstacles inside a sphere of radius  $R$  originating on the agent.

- **physical limitations:** The generated trajectory must satisfy the drone's velocity and acceleration limits and must be smooth and continuous relative to the actual state.

## III. METHOD

This section presents the proposed policy as a neural network that fulfils the above requirements. We start by presenting the trajectory encoding used by the proposed approach. Then, we present the privileged expert used for training and how we process the input features. Subsequently, we describe the neural network architecture depicted in Fig. 1 and deployed on the drone composed of input features processing, communicated variables aggregation and the optimization layer to cope with physical limitations. Finally, we describe the algorithm to compute the saliency map, used to analyze the neural network behaviour in the results.

### A. Trajectory Representation

The trajectory used by each drone is a polynomial-based curve, e.g., B-Splines, which allows us to impose velocity and higher derivatives limitation through linear constraints on the trajectory Control Points (CP). In fact, the convex hull of the curve CP leads to the representation of the outer trajectory polyhedron and, by imposing constraints on the CP, we ensure they are satisfied throughout the trajectory.

In this paper, we adopted a MINVO basis [28] expression for the curve. MINVO bases are recently introduced in the literature and, compared to B-spline or Bernstein bases, form a simplex  $\mathcal{M}$  enclosing the given polynomial curve with minimum volume by construction. Hence, these bases lead to less conservative polyhedron representations. Given a polynomial order, we can pass from MINVO to B-Spline CP and vice-versa by a linear transformation. For these reasons, it is ideal to generate and apply constraints on MINVO CP for trajectory descriptions. In particular, we define two linear mappings,  $h_v(\cdot)$  and  $h_a(\cdot)$ , to pass from MINVO CPs position to their velocity and acceleration, respectively.

### B. Privileged Expert

Our trajectory planner is trained via privileged learning [29]. Specifically, we generate a dataset from an expert controller: MADER [23]. MADER employs a decentralized and asynchronous approach to generate feasible and safe trajectories, leveraging a combination of path planning and Quadratic Programming (QP) optimization. We can consider this algorithm as a privileged expert for two reasons: first, MADER exploits comprehensive knowledge of the physical dimensions of obstacles and drones, as well as the trajectories of both, which is a piece of information typically unavailable in real-world scenarios; second, MADER ensures to find feasible trajectories only for unlimited time budget available in simulation. The optimization proposed by MADER focuses on determining the separation planes between the drone and potential obstacles throughout the trajectory, facilitating motion within a safe region. Moreover, the same reasoning is applied to trajectories committed by the other drones, leveraging the trajectory outer

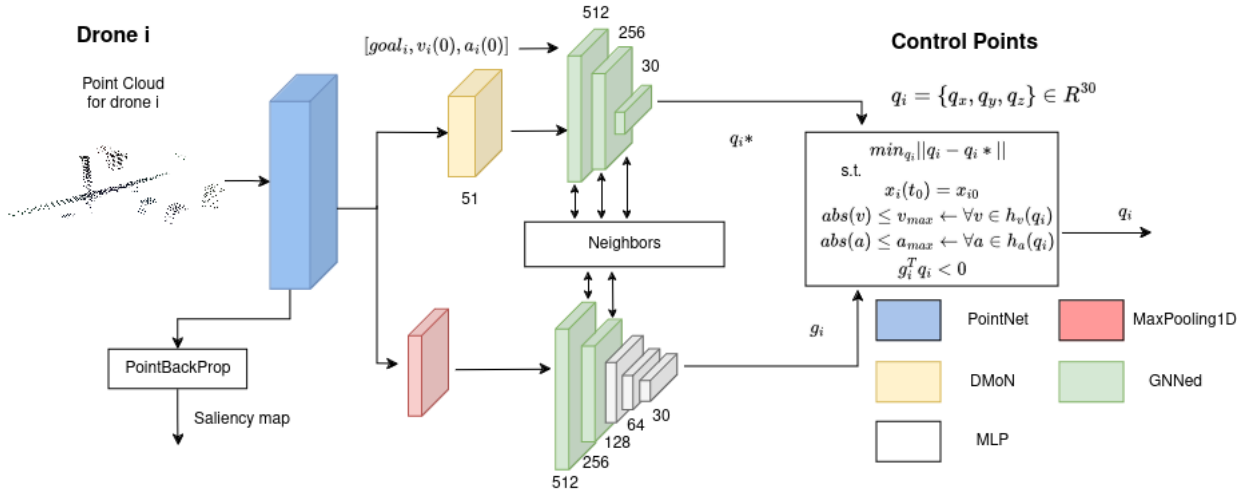


Fig. 1. neural network architecture deployed on each drone.

simplex as a cluttered area. However, the growth of the decision variables is proportional to the number of obstacles and other drones within the decision horizon. This renders the optimization more difficult to solve within a limited time budget, leading to potential infeasibility and hazardous halts. However, in simulation where full obstacle information is available and no time constraints are present, we can fully exploit MADER to generate a dataset of safe trajectories.

In our approach, to construct a dataset, we chose MADER over centralized alternatives as its features of asynchrony and decentralization align well with the objectives of our trajectory planner. Moreover, learning over an already decentralised expert policy eases the training process. After the training process, at execution time, the learned policy does not rely on any privileged information and synthesizes the trajectory from sensor inputs only.

### C. Neural Network Input Features

Every drone  $i$  in the team shares the same neural network architecture that accepts as input raw point clouds normalized in a unitary sphere and transformed into the world frame using  $quat_i$ . Additionally, the proposed neural network uses also the current velocity,  $v_i$ , and the current desired acceleration,  $a_i$ , normalized by the maximum velocity  $v_{max}$  and the maximum acceleration  $a_{max}$  allowed. We also include the goal location ( $goal_i$ ) in the drone  $i$  frame, projected on the sensing sphere with radius  $R$  if the distance to the goal is greater than  $R$ . Then, the  $goal_i$  is divided by  $R$  to have coordinates in the range  $[-1, 1]$ . Drone  $i$  can communicate intermediate neural network features with drones in its neighbourhood  $\mathcal{N}_i$ . We enclose the drone initial conditions in  $x_{i0} = [0_3, v_i, a_i]$  for a generated trajectory starting from the ego location,  $0_3 = [0, 0, 0]$ . In the final deployment, we add to the learnt control point  $q$  the current drone location  $r_i(t)$  to track the trajectory.

### D. Neural Network Structure

The neural network consists of two main branches that process the point cloud  $pc_i$ , combine it with localization

data  $[goal_i, v_i, a_i]$ , and communicate features to compute a trajectory guess vector  $q_i^*$  (described by MINVO CPs) and a vector of collision coefficients  $g_i$  for drone  $i$ . The latter is used to define a linear combination of CPs composing  $q_i$ , thereby classifying its safety. Specifically,

**Definition III.1.** Given  $pc := [pc_1^T \dots pc_N^T]^T$ , the joined point clouds from  $N$  drones, for drone  $i$ , the scalar value  $g_i^T q_i$  is  $g_i^T q_i < 0$  if and only if the control points vector  $q_i$  belongs to the safe set  $\mathcal{S}_i$ , where  $\mathcal{S}_i = \{q_i \mid \|p_i - p_j\|_2 > 2d, p_i \in \mathcal{M}(q_i), p_j \in pc\}$  and  $\mathcal{M}(q_i)$  is the outer simplex of  $q_i$ .

The term  $g_i^T q_i$  can be used as a linear constraint to generate a safe trajectory. The collision coefficients  $g_i$  serve to fortify the learned policies against collisions but also to ensure their applicability and reliability in unseen scenarios where the expert policy is unavailable or not included in the dataset. We formulate a QP layer as the final stage within the neural network that optimizes  $q_i$  to closely align with the initial guess  $q_i^*$ , while concurrently ensuring adherence to predefined maximum drone velocity ( $v_{max}$ ) and acceleration ( $a_{max}$ ) constraints. This combination not only refines trajectory predictions but also strengthens the system's resilience in navigating complex environments. The QP formulation is the following:

$$\begin{aligned} & \min_{q_i} \|q_i - q_i^*\|_2^2 \\ & \text{s.t.} \\ & x_i(t_0) = x_{i0} \\ & abs(v) \leq v_{max} \leftarrow \forall v \in h_v(q_i) \\ & abs(a) \leq a_{max} \leftarrow \forall a \in h_a(q_i) \\ & g_i^T q_i \leq 0 \end{aligned} \quad (1)$$

The point cloud is processed by a PointNet layer [30] comprising three filters of 64, 128, 256, respectively. PointNet employs sequences of point transformations through a compact transformation network and 1D-CNN with a unitary kernel size to generate spatially-permutation invariant features. The output of this layer,  $m \times 256$  with  $m$  points, splits to serve distinct purposes within the collision coefficients branch and the trajectory



generation branch. To generate  $g_i$ , i.e., assigning a safety score to trajectories crossing the environment, we require global features extracted from the point cloud. Consequently, the PointNet output is transformed into a 256-dimensional vector through global max pooling over the features.

Conversely, within the trajectory generation branch, our approach involves further clustering of the point cloud based on the features learned by PointNet. Clusters prove advantageous in classifying spatial regions that hold crucial information for trajectory generation. Clustering was already adopted in the design of PointNet++, which consists of spatial clustering coupled with smaller PointNet modules. We adopted DMoN [31], a clustering methodology based on a graph neural network, approximating spectral modularity maximization to recover high-quality features and spatial-based clusters. Unlike PointNet++, DMoN's approach does not solely rely on spatial displacement and does not require a nested architecture of PointNets. The adjacency matrix required by DMoN is represented as a binary sparse map encoding the spatial proximity of the points. This strategic combination of PointNet and DMoN very well captures intricate spatial structures for trajectory generation tasks, as we show in the results of Sec. V. We chose a number of clusters (51) that, together with the localization data, form a 64-dimensional vector for the next layer.

Both branches exchange the local features with the neighbouring agents using a message-aware graph attention network (MAGAT) [16]. Assuming that each drone  $i$  can communicate with its set of neighbours  $\mathcal{N}_i$ , the communication graph can be represented by a binary sparse matrix  $\mathbf{A}$ . We let  $\mathbf{A} \in \mathbb{R}^{N \times N}$  be the adjacency matrix of the communication graph. Given a graph signal  $\mathbf{x} \in \mathbb{R}^{N \times F}$  distributed over the drones, we can define a graph filter as

$$H_{\mathbf{A}}(\mathbf{x}) = \sum_{k=0}^K \mathbf{A}^k \mathbf{x} \mathbf{H}_k, \quad (2)$$

which combines the elements of  $\mathbf{x}$  over the adjacency matrix of the communication graph and applies the graph filter weights  $\mathbf{H}_k \in \mathbb{R}^{F \times F'}$ . The quantity  $K \geq 1$  represents the filter length, which implies repeated 1-hop communications over the graph. Therefore, the filter can be executed distributively over the graph. The filter in eq. (2) transforms the graph signal from an  $F$ -features space into a signal of an  $F'$ -features space. MAGAT enhances this framework by incorporating an attention mechanism to weigh the relative importance of drone features. Specifically:

$$H_{\mathbf{A}\mathcal{E}}(\mathbf{x}) = \sum_{k=0}^K (\mathcal{E} \circ \mathbf{A})^k \mathbf{x} \mathbf{H}_k, \quad (3)$$

$$\mathcal{E}_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{x}_i^T \mathbf{W} \mathbf{x}_j))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(\mathbf{x}_i^T \mathbf{W} \mathbf{x}_k))}$$

where  $\mathcal{E}$  is the matrix of attention weights. Considering the need to communicate graph signals over the network, the size of  $F$  affects network congestion. To address this, we extend the scheme by introducing an encoding-decoding mechanism to compress the graph signal before communication and reconstruct its original dimension afterwards. This involves point-

wise learnable encoding and decoding functions, denoted as  $e_{\theta}(\cdot) : \mathbb{R}^F \rightarrow \mathbb{R}^G$  and  $d_{\theta}(\cdot) : \mathbb{R}^G \rightarrow \mathbb{R}^F$ , respectively:

$$H_{\mathbf{A}e\mathbf{d}}(\mathbf{x}) = \sum_{k=0}^K d_{\theta}((\mathcal{E} \circ \mathbf{A})^k e_{\theta}(\mathbf{x})) \mathbf{H}_k. \quad (4)$$

Here,  $G \ll F$  reduces the number of communicated variables, with  $e_{\theta}$  and  $d_{\theta}$  functions implemented as MLP with ReLU activation functions. The resulting GNNed layer,

$$\mathbf{x} = \text{ReLU}(H_{\mathbf{A}e\mathbf{d}}(\mathbf{x})) \quad (5)$$

is employed 3 times in the trajectory generation branch (with filter dimensions 512, 256, 30) and twice in the collision branch (with filter dimensions 512, 256), totalling  $L = 5$  layers. Set parameters include  $K = 1$  and  $G = 5$ . We can use the unit-delay communication model [32] and communicate unit-time delayed signals to compute the graph filters in (4) in one shot, by sending  $[\mathbf{x}_i(t), \dots, \mathbf{A}_i^{K-1} \mathbf{x}(t - K - 1)]$  for each agent. This model allows releasing a new output at each communication iteration but at the cost of more communicated variables. Therefore, each drone communicates  $LGK$  variables, i.e. 25 in our implementation.

### E. Point Cloud Saliency Map

A saliency or attention map, denoted as  $s = [0, 1]^m$ , serves as a feature map unveiling the relevance of input data in the decision-making process. This map is instrumental in inspecting and interpreting neural networks, particularly under distribution shifts. Existing studies on point cloud saliency maps [33], [34] predominantly rely on gradient-based methods or incorporate additional neural network modules. As highlighted by the authors of VisBackProp [27], for autonomous navigation, it is convenient to dispose of a gradient-free method that can be evaluated online. Addressing this gap, we applied the same reasoning of VisBackProp to the PointNet layer in our architecture, proposing a variation of VisBackProp that can handle point cloud. We focus on the PointNet layer in our architecture, as it processes the point cloud and is dynamically shaped during training to enhance the representation of points for trajectory generation. The 1D-CNNs composing the PointNet draw a direct parallelism with VisBackProp that is applied to 2D-CNNs. For a PointNet made of  $P$  sequences of feature transformations and 1D-CNN, we save the average map  $\bar{h}_i$  computed over the features generated after each sequence. Starting from  $\bar{h}_P$ , we loop over the list of  $\bar{h}_i$  by multiplying the actual element with previous element of the list and normalize between  $[0, 1]$  after each multiplication. The algorithm is summarized in Algorithm 1.

## IV. TRAINING

Our model training dataset  $\mathcal{D}$  encompasses UAV around 16k trajectories under the control of the expert controller within a simulated environment. The simulation environment incorporates a 10% obstacle density organized in a forest-like setting and confined within a sphere of 10m radius centred in the world origin. The obstacles are partitioned equally into static and dynamic elements, with the dynamic obstacles

**Algorithm 1** PointBackProp

---

**Require:** point cloud ( $pt \in \mathbb{R}^{m \times 3}$ ), PointNet  
 $h_1, h_2, \dots, h_P \leftarrow$  get the  $P$  intermediate features computed by PointNet  
 $\bar{h}_1, \bar{h}_2, \dots, \bar{h}_P \leftarrow$  compute average of the features  
 $s := \bar{h}_P \triangleright$  saliency map  $s = [0, 1]^m$   
**for**  $i = P-1 \dots 1$  **do**  
     $s := \bar{h}_i \cdot s$   
    Normalize  $s$  between  $\{0, 1\}$   
**end for**  
compose point cloud values  $\{s, pt\}$

---

**Algorithm 2** Training Algorithm

---

**Require:**  $\mathcal{D}$  batch size  $b$   
**for** point cloud in  $\mathcal{D}$  **do**  
    Combine point clouds of the  $N$  drones  
    Generate random trajectories  $\hat{q}$   
    Solve QP to constraint  $\hat{q}$  with the initial conditions and physical limitations  
    add  $\hat{q}$  to  $\mathcal{S}_i$  according to the Definition III.1  
    Append trajectories and collision indexes to the dataset  
 $\hat{\mathcal{D}} \leftarrow \hat{\mathcal{D}} \cup \{\hat{q}\}$   
**end for**  
calculate adjacency matrix of drones  $A \triangleright$  This step is only needed in centralized training  
calculate adjacency of point clouds  $A_{pt}$   
**for**  $i = 1 \dots$  epochs **do**  
    collect  $b$  batches  $\{pt, v, a, quat, goal, q\} \leftarrow \mathcal{D}$   
    compute  $q^*, g \leftarrow$  model  
    **if**  $i > epochs_i$  **then**  
        prepare QP in eq (1) with constraint  $g^T q^* < 0$   
        Solve QP  $\rightarrow q$   
        **if** QP feasible **then**  
             $q \rightarrow q^*$   
        **end if**  
    **end if**  
    extract  $\{\hat{q}\}$  in  $\hat{\mathcal{D}}$   
    Compute constraint  $L_g(g, \hat{q})$  using eq (7)  
    Compute loss  $L_q(q, q^*)$  in eq (6)  
    Update model weights  
**end for**

---

following a three-dimensional trefoil knot motion spanning a width of 1m. For each mission, we randomly placed 8 drones around the surface of the 10-m sphere, with each drone target located on the opposite side of the sphere surface, forcing the traversal of the sphere centre. The sensing/communication radius is fixed at  $R = 4$  m. At a sampling rate of 15Hz, we capture point cloud data, localization information, and current control trajectory in MINVO control points, starting from the current drone location. The trajectories are saved as clamped spline with 10 equally spaced internal knots for each axis starting from the current timestamp  $t_0$  to  $t_f$ . Therefore, the trajectories are defined solely by 10 CPs for each axis for segment polynomials of degree 3. This helps the training process by focusing on learning the control points.

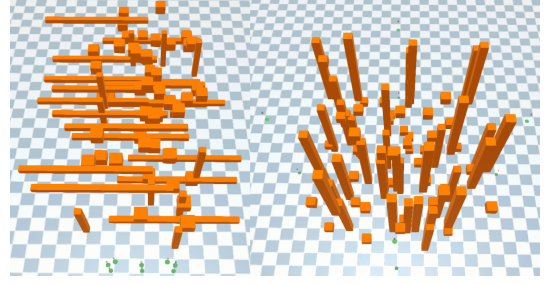


Fig. 2. Experimental environments: dynamic corridor (right) and dynamic forest (left).

In addition, we generate 560k random trajectories  $\hat{q}$  offline, originating from the initial robot location, and categorize them as successful or unsuccessful based on definition III.1 within the joint point cloud space, i.e. successful if the trajectory does not collide with obstacles or agent. Moreover, we introduce a diminishing safety distance  $d$  along the trajectory, ranging from the drone's actual size at the trajectory's initiation to 0 at its termination. This approach optimizes the safe set by progressively reducing conservatism, thereby prioritizing the safety of the trajectory's initial location:

$$L_q = \sum_{i \in \mathcal{V}} \frac{1}{30} \|q_i - q_i^*\|_2^2 \quad q_i \in \mathcal{D}, \quad (6)$$

$$L_g = \sum_{i \in \mathcal{V}} \sum_{\hat{q}_i \in \mathcal{S}_i} [g_i^T \hat{q}_i]^+ \sum_{\hat{q}_i \notin \mathcal{S}_i} [-g_i^T \hat{q}_i]^+, \quad (7)$$

where  $[\cdot]^+ = \text{Softplus}(\cdot)$  stands for a continuous ReLU function ensuring strict constraint satisfaction. We split the dataset into training, validation, and testing at ratios of 0.6, 0.3, and 0.1 respectively. We solve the imitation learning problem by using behavioural cloning with ADAM optimizer [35] employing a learning rate  $1e-3$  and forgetting factors 0.9 and 0.999. We trained for 200 epochs of which in the first  $50 = epochs_i$  the loss of eq. (6) is computed on the trajectory guess, without solving the QP. This solution helps to ease the learning and predict a good initial guess. The rest of the 150 epochs use the QP optimization as last layer as explain in Sec. III. If the QP is unfeasible, we used the initial guess to train our neural network. We exploit OptNet [36] to realize a differentiable quadratic programming layer and use classic backpropagation methods during training. We summarize the training process in the Algorithm 2.

## V. EXPERIMENTS

All simulations and training were conducted on a machine running Ubuntu 22.04. with Intel Core i7-9750H @ 2.60GHz CPU, Nvidia RTX 2080Ti and 32G RAM. The communication between the drones is employed through ROS with a communication rate of 100Hz while a new trajectory is calculated when a new point cloud is sensed at 15Hz. As specified in the assumption, we do not consider communication loss but, being the trajectory computation at a lower rate, we can guarantee a communication delay margin of 0.056 s. We first provide an ablation study and analysis results for a general quadrotor model with a perfect controller tracking trajectories with a

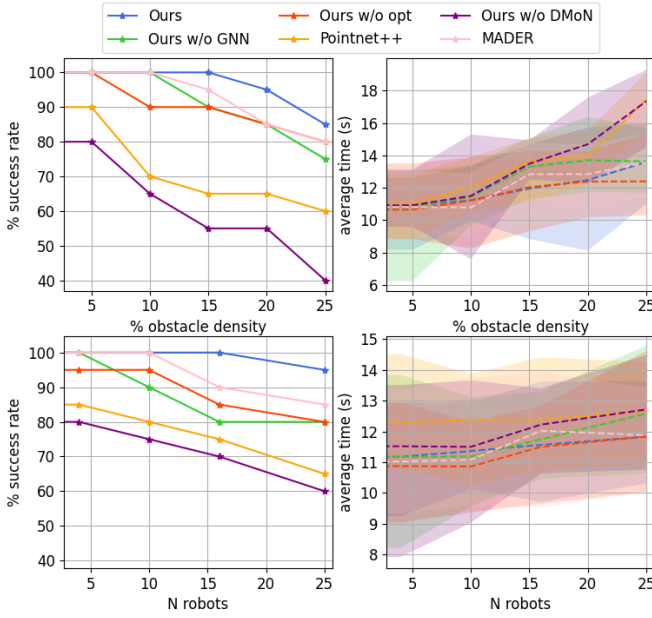


Fig. 3. Success rate and safety rate increasing obstacles and agents in the range of obstacle density [5% – 25%] and number of robots between [5 – 25] for our approach, ours without GNN, ours without collision constraint, ours with Pointnet++, ours without DMoN and MADER algorithm.

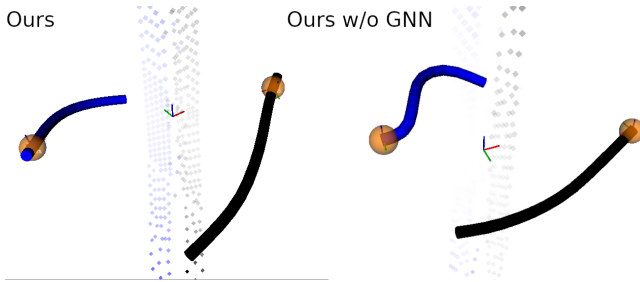


Fig. 4. Saliency map in a scenario with two drones for our approach and ours without GNN. The two drones (black and blue) sense each other and a pillar through their point cloud while moving toward the target. The saliency map is displaced by the alpha channel of the points.

maximum twist equal to 3.5 m/s in any directions and maximum acceleration equal to  $[20, 20, 9.6]$  m/s<sup>2</sup>. We assume the quadrotor size to be confined in a sphere of radius  $d = 0.15$ m. Moreover, we chose  $t_f = 1$ s which offers a good compromise between knots resolution and prediction horizon. In this setting, the obstacle reaches a maximum velocity of 2 m/s. Then, we provide experimental results using Crazyflies drones in a physical simulation with maximum velocity  $[1.0, 1.0, 1.0]$  m/s and maximum acceleration  $[2.0, 2.0, 2.0]$  m/s<sup>2</sup>. We leveraged a simulator built in Unity+Mujoco and software in the loop (SITL)<sup>1</sup> to mimic the real drone behaviour. In this setting, we reduce the obstacle maximum velocity to 1 m/s and  $t_f = 3$ s because of the drone's reduced velocity.

We consider two experimental conditions: a variable number of agents in a range of  $[4, 25]$ , with dynamic and static obstacle density of 10% of the space; a fixed number of agents to 8 and increasing obstacle density from 5% to 25%. In both

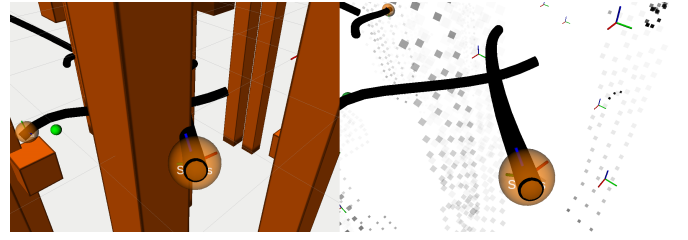


Fig. 5. Saliency map (on the left) of collision case using our approach for a trajectory starting from the drone and traversing the obstacle.

conditions, we evaluate the average success rate and average travel time for the agents to reach their respective targets. We tested the approach in two settings shown in Fig. 2: the forest-like used during training and a corridor-like environment  $8\text{m} \times 20\text{m}$  including also horizontal pillars. In the second setting, the drones are required to traverse the corridor from end to end. For each experimental condition (number of drones in team, obstacle density), we carried out 50 repetitions of the travelling mission, equally distributed in the two scenarios. We consider a mission to be successful if all the agents reach their target without colliding. We proceeded with an ablation study to evaluate our approach:

- *ours w/o GNN*: we replaced GNN layers with normal MLP, to evaluate the impact of communication over the predictions.
- *ours w/o opt*: we did not use the predicted constraint  $g$  to test the impact of this introduced feature.
- *Pointnet++*: we replaced our solution of Pointnet-DMoN with Pointnet++ and max pooling.
- *ours w/o DMoN*: we removed DMoN clustering and used max pooling to predict the trajectory guess, as for the constraint prediction branch.

Additionally, for comparison, we also consider MADER, providing full obstacle trajectories but a limited time budget of 0.35s for each optimization according to the original paper to guarantee a real-time execution. Our approach reaches an average computational time of 1.3ms with a maximum of 1500 points sensed. The computational time in real scenarios can change based on the LiDAR sensor resolution and the solver used for the last optimization layer which might be different from OptNet as, after the training, there is no need for the optimization to be differentiable. Note that, the sensing frequency dominates the computational time.

## A. Results

We present an analysis of the average success rate and average travel time on successful trajectories, as illustrated in Fig. 3. Our proposed neural network consistently achieves a remarkable success rate of 100% for obstacle densities up to 15% and 16 robots, gradually decreasing to 85% for the highest obstacle density of 25%. Notably, the effectiveness of our model is closely tied to point cloud processing, as evidenced by the success rates of *Pointnet++* and *ours w/o DMoN*, ranging from 90% to 40%. The clustering capabilities of *Pointnet++* contribute to an average 8.3% higher success

<sup>1</sup>[https://gitlab.inria.fr/amarino/crazyswarm2\\_unity\\_sim](https://gitlab.inria.fr/amarino/crazyswarm2_unity_sim)

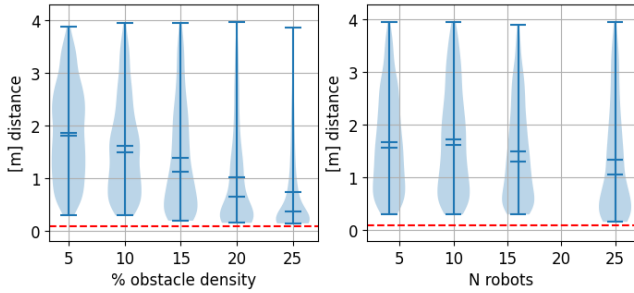


Fig. 6. Distance from points sensed in a range of 4 m for flight tests with Crazyflie drones. The red dashed lines denote the physical safe distance.

rate compared to *ours w/o DMoN*, emphasizing its impact on the environment processing.

When the model does not exploit the learnt collision constraint  $g$ , it encounters difficulties in finding collision-free trajectories, especially in high-constraint spaces populated by both cooperative and non-cooperative agents where the success rate diminishes to 80% for an obstacle density of 25% and 25 robots. The integration of GNN enhances the model’s resilience to collisions and traversal time as the number of drones increases, showcasing the advantages of cooperative trajectory predictions. Our approach shows advantages also compared to MADER with a high density of obstacles and robots primarily because when the optimization in MADER does not find a feasible trajectory the algorithm keeps using the previously computed trajectory. When the drone reaches the end of the trajectory, it breaks remaining exposed to obstacle collision until the optimization becomes feasible again because of the dynamical changes in the environment. Notably, *ours w/o opt* and MADER reach the same success rate of 85% with 25% obstacle density which we speculate is due to the drone coordination through GNN.

Our approach exhibits the lowest average travel time of 11.8s with 25 robots even if MADER and *ours w/o opt* find faster trajectories to traverse the space with fewer drones (5 – 10) and travel times comparable to *ours* as obstacle density increases. Moreover *ours w/o opt* has the lowest travel time when the obstacle increases with  $12.4s \pm 2.1$ , at the expense of less safe trajectories. In contrast, *Pointnet++* and *ours w/o DMoN* have similar travel times, approximately 17.3s with a high variance of 2s, facing drone deadlock situations or predicting longer paths to reach the goal.

Our approach reaches a similar average time to MADER as we used it to generate the dataset. However, we note that MADER does not generate global optimal time trajectories due to its asynchronous communication strategy, aiding the first drone committing the trajectory. As a result, subsequent drones must adapt to avoid collisions, resulting in non-optimal trajectories.

We use Algorithm 1 to generate saliency maps, which help interpret the network’s behaviour and provide insights into the point cloud’s contribution to trajectory prediction. The saliency map allows us to qualitatively assess how well the neural network uses the sensing data for decision-making. We expect that the relevance of each point in the cloud is

influenced by its distance from the drone, as the network predicts spatio-temporal commands and potential collisions around the drone, independent of the goal location. Initially, we focus our analysis on a scenario involving two drones and a static obstacle, as illustrated in Fig. 4. To highlight the contribution of different observations, we assign distinct colours (blue and black) to points sensed by individual drones, with the alpha channel representing saliency values—more transparent points indicate less impact on predictions. It is evident from the saliency map that each drone perceives the obstacle from only one side. As expected, points closer to the drones are more transparent since the first part of the trajectory is predefined by the continuity with the current motion. Compared to *ours w/o GNN*, the saliency map for our approach shows a balanced distribution of points between the two drones, resulting in more uniform trajectories across the environment. In contrast, when deploying *ours w/o GNN*, we note an unbalanced use of the drone point cloud with the black drone relying more heavily on its sensed points than the blue drone, leading to distinct behaviours. This discrepancy indicates more potential conflicts between the drones due to differences in the perceptions of the environment, while GNN helps to reach coherent observations across different viewpoints. Additionally, saliency maps serve as valuable tools for analyzing failure cases. Figure 5 showcases a collision scenario with *ours*. The trajectory intersects a pillar, with its points appearing transparent in the saliency map, suggesting that the network is “blind” to the obstacle, although perceived through the point cloud. Conversely, other obstacles are clearly visible and incorporated into the trajectory planning.

### B. Physical Simulation

In a physical simulator, We tested our approach with Crazyflie drones. We recorded a mission for each experimental scenario: obstacle density ranging from 5% to 25% and numbers of agents ranging from 4 to 25. Moreover, as for the non-physical simulation, the missions were conducted for the two scenarios of a dynamic forest and a dynamic corridor. Figure 6 illustrates the distribution of accumulated drone-to-obstacle distances during the flights. Notably, no collisions were observed, as all recorded distances remained above the safety threshold of 0.1 meters.

## VI. CONCLUSION

This work introduces a decentralized end-to-end trajectory planner that addresses static obstacles, dynamic obstacles, and other agents. By learning a collision constraint alongside the trajectory, we ensure the safety and dynamic feasibility of the trajectories in a QP framework. We extensively validate our approach in simulation against variations of our approach and demonstrate robustness to team scalability and varying obstacle densities in the environment. Additionally, we derived an algorithm to compute the saliency map of the point cloud, which we used as a tool to interpret and understand both failure and success cases. Future work could explore how to exploit the saliency map to enhance prediction performance and consider alternative learning frameworks that do not



rely on privileged experts. Furthermore, future research will examine additional robustness metrics under conditions such as sensor noise, tracking deviations, dynamic environmental changes, and hardware experiments.

## REFERENCES

- [1] J. Gu, T. Su, Q. Wang, X. Du, and M. Guizani, "Multiple moving targets surveillance based on a cooperative network for multi-uav," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 82–89, 2018.
- [2] P. Peng, W. Dong, G. Chen, and X. Zhu, "Obstacle avoidance of resilient uav swarm formation with active sensing system in the dense environment," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 529–10 535.
- [3] Y. Gao, Y. Wang, X. Zhong, T. Yang, M. Wang, Z. Xu, Y. Wang, Y. Lin, C. Xu, and F. Gao, "Meeting-merging-mission: A multi-robot coordinate framework for large-scale communication-limited exploration," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 700–13 707.
- [4] A. Alcántara, J. Capitán, R. Cunha, and A. Ollero, "Optimal trajectory planning for cinematography with multiple unmanned aerial vehicles," *Robotics and Autonomous Systems*, vol. 140, p. 103778, 2021.
- [5] H.-J. Kim and H.-S. Ahn, "Realization of swarm formation flying and optimal trajectory generation for multi-drone performance show," in *2016 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2016, pp. 850–855.
- [6] C. Zhao, J. Liu, M. Sheng, W. Teng, Y. Zheng, and J. Li, "Multi-uav trajectory planning for energy-efficient content coverage: A decentralized learning-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3193–3207, 2021.
- [7] Y. Chen, U. Rosolia, and A. D. Ames, "Decentralized task and path planning for multi-robot systems," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4337–4344, 2021.
- [8] J. Cortés and M. Egerstedt, "Coordinated control of multi-robot systems: A survey," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 6, pp. 495–503, 2017.
- [9] —, "Coordinated control of multi-robot systems: A survey," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 6, pp. 495–503, 2017.
- [10] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.
- [11] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64-mw dnn-based visual navigation engine for autonomous nano-drones," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8357–8371, 2019.
- [12] P. Miera, H. Szolc, and T. Kryjak, "Lidar-based drone navigation with reinforcement learning," *arXiv preprint arXiv:2307.14313*, 2023.
- [13] S. El-Ferik, M. Maaruf, F. Al-Sunni, A. A. Saif, and M. M. Al Dhaifallah, "Reinforcement learning-based control strategy for multi-agent systems subjected to actuator cyberattacks during affine formation maneuvers," *IEEE Access*, 2023.
- [14] X. Liu, Y. Liu, and Y. Chen, "Reinforcement learning in multiple-uav networks: Deployment and movement design," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8036–8049, 2019.
- [15] S. Batra, Z. Huang, A. Petrenko, T. Kumar, A. Molchanov, and G. S. Sukhatme, "Decentralized control of quadrotor swarms with end-to-end deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 576–586.
- [16] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-aware graph attention networks for large-scale multi-robot path planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5533–5540, 2021.
- [17] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 11 785–11 792.
- [18] J. Blumenkamp, S. Morad, J. Gielis, Q. Li, and A. Prorok, "A framework for real-world multi-robot systems running decentralized gnn-based policies," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8772–8778.
- [19] B. Riviere, W. Hönig, Y. Yue, and S.-J. Chung, "Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4249–4256, 2020.
- [20] S. Zhang, K. Garg, and C. Fan, "Neural graph control barrier functions guided distributed collision-avoidance multi-agent control," in *Conference on Robot Learning*. PMLR, 2023, pp. 2373–2392.
- [21] B. Sabetghadam, R. Cunha, and A. Pascoal, "A distributed algorithm for real-time multi-drone collision-free trajectory replanning," *Sensors*, vol. 22, no. 5, p. 1855, 2022.
- [22] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.
- [23] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multiagent and dynamic environments," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 463–476, 2021.
- [24] K. Kondo, R. Figueroa, J. Rached, J. Tordesillas, P. C. Lusk, and J. P. How, "Robust mader: Decentralized multiagent trajectory planner robust to communication delay in dynamic environments," *IEEE Robotics and Automation Letters*, 2023.
- [25] Z. Wang, C. Xu, and F. Gao, "Robust trajectory planning for spatial-temporal multi-drone coordination in large scenes," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 12 182–12 188.
- [26] J. Park, D. Kim, G. C. Kim, D. Oh, and H. J. Kim, "Online distributed trajectory planning for quadrotor swarm with feasibility guarantee using linear safe corridor," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4869–4876, 2022.
- [27] M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. J. Ackel, U. Muller, P. Yeres, and K. Zieba, "Visualbackprop: Efficient visualization of cnns for autonomous driving," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4701–4708, 2018.
- [28] J. Tordesillas and J. P. How, "Minvo basis: Finding simplexes with minimum volume enclosing polynomial curves," *Computer-Aided Design*, vol. 151, p. 103341, 2022.
- [29] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on Robot Learning*. PMLR, 2020, pp. 66–75.
- [30] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2017, pp. 77–85.
- [31] A. Tsitsulin, J. Palowitch, B. Perozzi, and E. Müller, "Graph clustering with graph neural networks," *Journal of Machine Learning Research*, vol. 24, no. 127, pp. 1–21, 2023.
- [32] F. Gama, Q. Li, E. Tolstaya, A. Prorok, and A. Ribeiro, "Synthesizing decentralized controllers with graph neural networks and imitation learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1932–1946, 2022.
- [33] Z. Jiang, L. Ding, G. K. Tam, C. Song, F. W. Li, and B. Yang, "C2spoint: A classification-to-saliency network for point cloud saliency detection," *Computers & Graphics*, vol. 115, pp. 274–284, 2023.
- [34] T. Zheng, C. Chen, J. Yuan, B. Li, and K. Ren, "Pointcloud saliency maps," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1598–1606.
- [35] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
- [36] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 136–145.