Distributed Connectivity-maintenance Control of a Team of Unmanned Aerial Vehicles using Supervised Deep Learning

Muhammad Sunny Nazeer, Marco Aggravi, Paolo Robuffo Giordano, Claudio Pacchierotti

Distributed Connectivity-maintenance Control of a Team of Unmanned Aerial Vehicles using Supervised Deep Learning

Muhammad Sunny Nazeer, Marco Aggravi, Paolo Robuffo Giordano, Claudio Pacchierotti

***Abstract*— This paper presents the design of a decentralized connectivity-maintenance technique for the teleoperation of a team of multiple Unmanned Aerial Vehicles (UAV). Different from current, typical connectivity-maintenance approaches, the proposed technique uses machine learning to attain significantly more scalability in terms of number of UAVs that can be part of the robotic team. It uses Supervised Deep Learning (SDL) with Artificial Neural Networks (ANN), so that each robot can extrapolate the necessary actions for keeping the team connected in one computation step, regardless the size of the team. We compared the performance of our proposed approach vs. a state-of-the-art model-based connectivity-maintenance algorithm when managing a team composed of two, four, six, and ten aerial mobile robots. Results show that our approach can keep the computational cost almost constant as the number of drones increases, reducing it significantly with respect to model-based techniques. For example, our SDL approach needs 83% less time than a state-of-the-art model-based connectivity-maintenance algorithm when managing a team of ten drones.**

I. Introduction

Teleoperation of mobile robots can be useful in the entertainment industry, in telepresence, meteorology, surveillance, search and rescue, inspection of damaged buildings, and so on. A clear trend looking into the near future is for these mobile robots, both ground and aerial, to become smaller and more agile, which will make the use of multi-robot systems (robotic "teams") more and more feasible [1], [2]. The idea behind the use of multi-robot systems is that using more robots will lead to improved redundancy, covered area, and heterogeneity of available robotic features [3].

This paper presents and evaluates a behavior learnt from a decentralized connectivity-maintenance algorithm via a deep-learning based approach for the teleoperation of a team of multiple UAVs by a human operator. It employs Supervised Deep Learning (SDL) with Artificial Neural Networks (ANN) to maintain the connectivity of the team, so that each robot can calculate the necessary connectivity-maintenance actions in one computation step. Its main contributions are:

1) design of a deep learning ANN-based control scheme to teleoperate a multi-robot team from data generated using the Generalized-Connectivity-Maintenance (GCM) algorithm presented in [4], [5];
2) training of the learning scheme for controlling a multi-robot team composed of two, four, six, and ten UAVs;
3) experimental evaluation of the performance of the proposed machine-learning based scheme vs. the model-based GCM algorithm [4], [5].

M. S. Nazeer, M. Aggravi, P. Robuffo Giordano, and C. Pacchierotti are with CNRS, Univ Rennes, Inria, IRISA – Rennes, France.

M. S. Nazeer is now at The BioRobotics Institute, Scuola Superiore Sant'Anna – Pisa, Italy. E-mail: muhammadsunny.nazeer@santannapisa.it

With respect to [4], [5] and other model-based (or analytic) approaches for controlling multi-robot systems, we expect our approach to be more scalable regarding the number of drones it can manage. While the computation time of [4], [5] increases rapidly as the number of drones (and thus neighbors) increases, the computation time of the proposed approach is expected to remain almost constant regardless the number of involved agents. This feature allows using connectivity-maintenance techniques with an arbitrary number of agents, well beyond what is typically possible using standard machines running available model-based connectivity-maintenance algorithms. We compare the performance of our technique vs. the parent GCM algorithm managing up to ten drones.

II. Related works

Teleoperation of aerial robots. Commercial aerial systems enable the control of aerial robots through remote controllers or dedicated smartphone applications. Another approach to remotely control the motion of drones is using hand gestures detected by the on-board drone camera [6]. A similar technique uses a Kinect depth camera [7] to detect the operator's gesture and body posture, which are then used to control the motion of a Parrot drone. Another way to detecting hand gestures is using a LeapMotion. Using this tracking technology, in [8], hand position and orientation are translated into commands for the roll, pitch, and yaw of a Parrot ARDrone robot, while in [9], researchers compare the control performance of a drone when employing hand gestures vs. speech commands; and in [10], gestures are used to impart predefined motions such as "fly in a circle" or "ascend in a spiral fashion." A promising approach for people with motor disabilities is controlling aerial drones with their gaze [11]. More recently, body suits and exoskeletons have also been used to control drones [12], [13], [14].

Multi-robot formation control by a human. Enabling humans to control a multi-robot system poses several challenges. Franchi et al. [15] proposed a technique based on a "leader-follower" paradigm. A human user directly controls the leader of the robotic team through a grounded haptic interface, while the other robots follow its motion through spring-like couplings among connected pairs of neighbors. This technique enables the user to focus on the control of one robot (the leader), while the rest of the team is controlled by the autonomous algorithm. However, this approach does not enable any control on the shape/characteristics of the formation. Extensions of [15] have been presented in [4], [16], introducing the maintenance of global properties of the team such as its connectivity or rigidity. Aggravi et al. [17]

extended the distributed connectivity-maintenance algorithm of [4] by including airflow-avoidance behaviors, a consensus-based action for enabling fast displacements with minimal topology changes, an automatic decrease of the minimum connectivity level, and the automatic detection and resolution of deadlock configurations. In other approaches, the operator controls some target global property of the formation, such as its velocity of barycenter, while the robots maintain a desired shape of the formation [18], [19].

Machine Learning (ML) for aerial robots. Most of the abovementioned techniques employs model-based approaches, where the control laws are designed and analyzed analytically. While such approaches are rather effective with a predictable and guaranteed behavior (within the modeling assumptions), they often show scalability issues, as their computation cost increases very fast as the number of drones/neighbors in the team increases. ML is an interesting approach to make the control of aerial robots more flexible [20]. ML has been used in [21], [22] to attain robust control, capable of alleviating model mismatches, wind disturbances, and measurement noise during the autonomous control of a drone. One of the biggest advantage of ML-based approaches is in their ability to learn substantially sophisticated dynamics either from experiences demonstrated by another agent [23], [24] or from its operational dataset [25], without explicitly requiring an expert understanding of the underlying mechanics of the subject platform. Similarly, researchers have proposed an application of Support Vector Regression (SVR) for adaptive control law of UAV platforms in [26], the design of a position and throttle controller of a UAV based on Adaptive Neuro-Fuzzy Inference System in [27], and a concurrent learning algorithm to improve the performance of the model reference adaptive control architecture in [28]. More recently, Jardine et al. [29] uses ML to tune the gains of a model predictive control algorithm for anticipating the trajectory of a quadcopter. Further use of ML for aerial robotic applications are in autonomous navigation and path planning [30], [31], [32] as well as in obstacle avoidance [33], [34], [35], and vision-based formation control [36]. Regarding multi-robot systems, Reinforcement Learning (RL) has been very popular in the recent past. For example, Derhami and Momeni [37] achieved formation control using multi-agent fuzzy RL, John and Andersson [38] combined RL with behavior-based formation control algorithms, and Garamifard et al. [39] used RL for reactive cooperation among the team agents. More recently, Mox et al. [40] used a convolutional neural network (CNN) based on an available image dataset to place multiple agents in a network most suitable for inter-agent communication. Finally, Huang et al. [41] implemented an off-policy deep reinforcement learning algorithm (DQN) to preserve connectivity in a multi-agent environment, while Juntong et al. [42] and Minghao et al. [43] used deep RL for multi-agent formation control while adding constraints for enforcing a fixed center or shape of the swarm.

However, while deep RL control approach show good robusticity towards disturbance, they often need substantial computational time and data [23], [44] to derive the target policy [45]. In this respect, scalability is often overlooked, as the required resources tend to increase significantly. Moreover, such training approaches also require a reward function to best describe the expected control behavior, thus requiring (expert) knowledge of the task and system at hand.

III. Preliminaries

We start by summarizing the model-based GCM algorithm introduced in [4], [5]. Our objective is to train an artificial neural network able to elicit an overall connectivity-maintenance behavior similar to the GCM while supporting a (much) higher number of drones. Let us start by considering a team of N_t robots modeled as 3D-point masses, having positions $\mathbf{x}_i \in \mathbb{R}^3$, $i \in \{1, \cdots, N_t\}$ and double integrator dynamics

$$\ddot{\mathbf{x}}_i = \mathbf{u}_i,\ i \in \{1, \ldots, N_t\}, \tag{1}$$

with $\mathbf{u}_i \in \mathbb{R}^3$ representing the control action (a force) applied to robot i, and $\mathbf{v}_i = \dot{\mathbf{x}}_i$ its velocity. The connectivity-maintenance control described in [4] steers each robot in the team using a gradient-descent strategy which guarantees connectivity maintenance of the sensing/communication formation graph $\mathcal{G}$. Each node of this graph represents an agent. Local links between neighbors may be broken or created at runtime during the teleoperation as sensing/communication conditions evolve. The connectivity-maintenance algorithm can take into account several constraints, according to the nature of the task and the features of the robots involved. For example, it can account for avoiding obstacles and collisions, keeping a line-of-sight visibility free of occlusions, and retain a target distance with respect to the neighbors. The information about these constraints is encoded into the *adjacency matrix* $\mathbf{A} \in \mathbb{R}^{N_t \times N_t}$ associated to the formation graph $\mathcal{G}$ [4], from which it is possible to evaluate the *Laplacian matrix*. The second smallest eigenvalue λ_2 of this matrix provides information about the connectivity of the graph. If $\lambda_2 > 0$, the graph is connected, while if $\lambda_2 = 0$, the graph is disconnected [46].

The connectivity-maintenance control consists in a decentralized gradient descent of a potential function $V^\lambda(\lambda_2(\mathbf{x}(t))) \geq 0$ [4] of the connectivity eigenvalue $\lambda_2(\mathbf{x}(t))$, that w.l.o.g. we will call $\lambda_2(t)$. The potential $V^\lambda(\lambda_2(t))$ is then defined with a vertical asymptote at $\lambda_2^{\min} > 0$, smoothly decreasing from $\lambda_2^{\min}$ to $\lambda_2^{\max} = \lambda_2^{\min} + \Delta$, and eventually going to zero for $\lambda_2(t) \geq \lambda_2^{\max}$. The connectivity-maintenance action is defined as the gradient descent

$$\mathbf{F}_i^\lambda = -\frac{\partial V^\lambda(\lambda_2(t))}{\partial \mathbf{x}_i}. \tag{2}$$

The computation of $\mathbf{F}_i^\lambda$ made by robot i can be decentralized by considering a decentralized estimation $\hat{\lambda}_2^i(t)$ of the true value $\lambda_2(t)$, as explained in [4]. Finally, we can define the control input for robot i in (1) as

$$\mathbf{u}_i = \mathbf{F}_i^\lambda - \mathbf{B}_i \mathbf{v}_i + \mathbf{F}_i^e, \tag{3}$$

where $\mathbf{B}_i \in \mathbb{R}^{3\times 3}$ is a damping term (included to increase stability [17]), and $\mathbf{F}_i^e \in \mathbb{R}^3$ is an additional exogenous force that may act (or not) on robot i. For example, in [17], $\mathbf{F}_i^e$

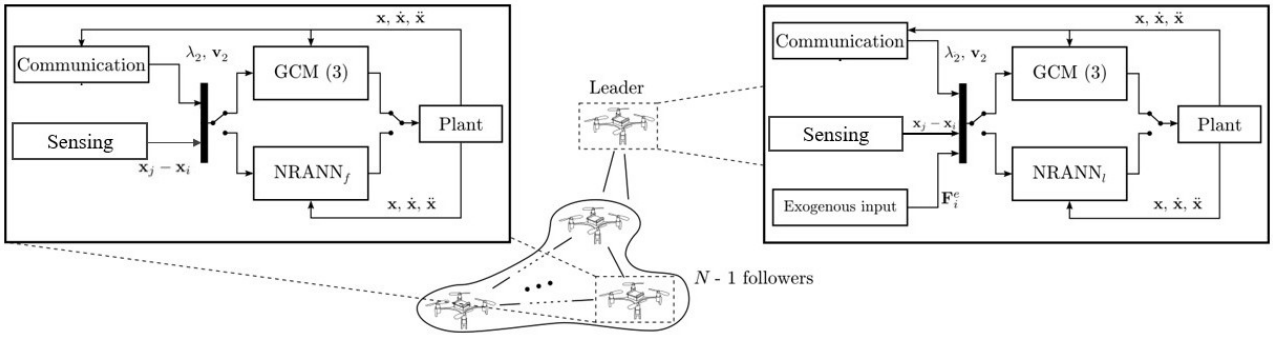


Fig. 1. Control schemes for the multi-robot team. Starting from the building blocks necessary to run the connectivity-maintenance control (communication, inter-agent and environmental sensing), we control the team motion either through the standard model-based GCM approach or the proposed ML-based NRANN one. The objective is to evaluate the pros and cons of employing GCM vs. NRANN for the control of multi-robot systems.

is imparted by a human user to one (leader) drone, so as to control its motion and, in turn, the one of the whole team. If all external forces $\mathbf{F}_i^e$ are bounded, the connectivity-maintenance action of (3) guarantees $\lambda_2(t) > \lambda_2^{\min} > 0$, $\forall t$.

Each robot i needs to perform these calculations for each one of its neighbors.

IV. AI-BASED CONNECTIVITY-MAINTENANCE

The considered model-based approach GCM already holds the information about 1) maintaining global connectivity of the network of agents and guaranteeing 2) obstacle and collision avoidance [4], [5]. Our objective is to use ML to train a policy capable of eliciting this behavior while significantly reducing its computation cost, thus enabling its use with teams composed of a higher number of robots. For example, the cost of the control algorithm presented in [4] increases with the number of neighbors of each robot. In the case of a highly connected graph, most of the robots are neighbors of each other and the computation time tends to increase drastically.

To achieve our objective, we trained different ANNs. We call an ANN trained to manage N_t robots as NRANN^{N_t}. Similarly, the GCM algorithm implemented to manage N agents is called as GCM^N. In this work, we considered teams composed of $N_t = \{2, 4, 6, 10\}$ robots. However, the proposed approach can be easily extended to manage teams composed of an arbitrary number of robots.

A. Data collection

We implemented the distributed connectivity-algorithm of [4] using Phyton. Then, we ran 567 hours of simulations, equally considering teams composed of $N_t = \{2, 4, 6, 10\}$ robots. The motion of the team was generated by providing a designated robot leader with force $\mathbf{F}_i^e$, generated in a continuous fashion in the range of $[-1, 1] * \mathbf{B}_i$, where $\mathbf{B}_i$ is the same damping factor of eq. (3). This behavior of $\mathbf{F}_i^e$ was designed starting from the user-input data collected in our previous work [17], where 12 human operators were asked to teleoperate a robot leader within a team enforcing the connectivity-maintenance algorithm of [4]. We considered a sampling time of 0.005 s, leading to a total of 408,240,000 data points. Each point carried information about *(i)* the position, *(ii)* velocity, and *(iii)* acceleration of each robot composing the team, as well as *(iv)* the team connectivity parameter λ_2, *(v)* the eigenvector associated to λ_2 for each robot [4], and *(vi)* the imparted $\mathbf{F}_i^e$.

B. Artificial Neural Network (ANN) training

We trained a set of two ANNs for each considered team composed of N_t robots. The first ANN, referred to as NRANN$_l$, is for controlling the robot leader, while the second ANN, referred to as NRANN$_f$, is for controlling all the other (followers) robots. The only difference is the presence of the exogenous force $\mathbf{F}_i^e$ in NRANN$_l$, which we refer to as $\mathbf{F}_l^e$. In Sec. VII we discuss on the opportunity of using the same network for both types of agents.

To train these networks, we used Long short-term memory (LSTM) artificial recurrent neural networks (RNN) [47]. Differently from more standard feed-forward neural networks, this type of RNN has feedback connections. Let us define the LSTM input layer with $\mathbf{I}$, using subscript l for the leader agent and $f = \{2, \ldots, N_t\}$ for the follower agents, and the dense output layer with $\mathbf{O}$, using again subscript l for the leader agent and $f = \{2, \ldots, N_t\}$ for the follower agents. Between them, we included two LSTM hidden layers of 128 nodes. Additional hyperparameters used for training include: 10% dropout layer in the second LSTM layer, *tanh* activation function in all the layers (including output layer), 0.0003 as a constant learning rate, 32 samples per batchsize, and 200 epochs. These hyperparameters were tuned based on trial-and-error. Standard Adam optimizer was used for training with mean squared error as a loss function. The input and output layers for the leader and follower networks are defined as follows,

$$\begin{aligned} \mathbf{I}_l &= \left[\mathbf{x}_2 - \mathbf{x}_1, \ldots, \mathbf{x}_{N_t} - \mathbf{x}_1, \dot{\mathbf{x}}_l, \mathbf{F}_l^e, \lambda_2, \mathbf{v}_{\lambda_2}^l\right] \in \mathbb{R}^{4(N_t+1)} \\ \mathbf{O}_l &= \mathbf{F}_1^\lambda \in \mathbb{R}^3 \end{aligned} \tag{4}$$

$$\begin{aligned} \mathbf{I}_f &= \left[\mathbf{x}_1 - \mathbf{x}_f, \ldots, \mathbf{x}_{N_t} - \mathbf{x}_f, \dot{\mathbf{x}}_f, \lambda_2, \mathbf{v}_{\lambda_2}^f\right] \in \mathbb{R}^{4N_t+1} \\ \mathbf{O}_f &= \mathbf{F}_f^\lambda \in \mathbb{R}^3 \end{aligned} \tag{5}$$

where $\mathbf{x}_i \in \mathbb{R}^3$, $i \in \{1, \cdots, N_t\}$ are the positions of the robots, $\mathbf{x}_i - \mathbf{x}_j$ the relative position from robot i to j, $\dot{\mathbf{x}}_i$ is the velocity of robot i, $\mathbf{F}_i^e \in \mathbb{R}^3$ is the exogenous input to the leader robot, λ_2 is the connectivity eigenvalue computed in a

decentralized manner by each robot, $\mathbf{v}_{\lambda_2}^l \in \mathbb{R}^{N_t}$ and $\mathbf{v}_{\lambda_2}^f \in \mathbb{R}^{N_t}$ are the full eigenvectors of the components associated to robots l and f, using 1-hop communication based on the order at which $\mathbf{x}_i - \mathbf{x}_j$ are considered, and, finally, $\mathbf{F}_l^\lambda$ and $\mathbf{F}_f^\lambda$ are the connectivity forces acting on l and f, respectively. Fig. 1 summarizes the two considered control schemes. The choice of our (recurrent) neural network architecture also takes into account the predicted forces at previous time instances.

All the dataset, collected as described in Sec. IV-A, is arranged into a flatten array of dimension $(n_{samples}, 1, n_{inputs})$. This data is fed as input to the leader and follower ANNs. The parameteric optimization of both ANNs was motivated based on the connectivity-maintenance dataset (refer to Sec. IV-A). The NRANN$_l$ produces the force $\mathbf{F}_l^\lambda$ at which the leader drone should move to respond to the commanded exogenous force while being consistent with the GCM behavior (see eq. (4)). Similarly, each NRANN$_f$ produce the force $\mathbf{F}_i^\lambda$ to drive the follower drones in a manner consistent with the GCM behavior (see eq. (5)). These forces are then imparted to the drones as accelerations, considering their mass to be 1 kg [4]. Each drone computes this output force in a decentralized manner.

V. Experimental Evaluation

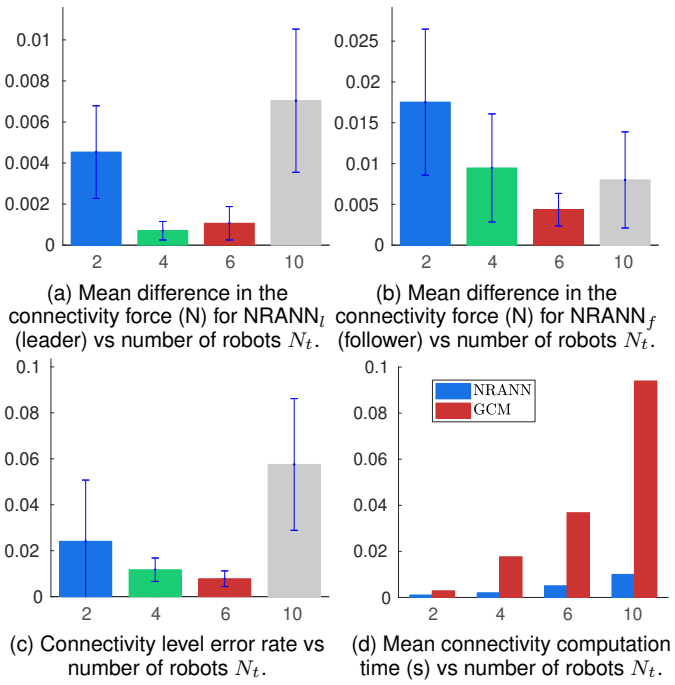
To evaluate the effectiveness of the proposed approach and compare it to the standard GCM algorithm, we tested it in a wide range of prerecorded simulations data as well as in two real world experiments.

A. Prerecorded simulations

We started by considering a set of simulated team motions. We collected 15 episodes of 3 minutes for each considered team size $N_t = \{2, 4, 6, 10\}$, yielding a total of 15×3×4=3 hours of data. During each episode, the team leader was driven by a set of pre-selected continuous exogenous forces, resulting in different patterns of motion. As in Sec. IV-A, this behavior was inspired from the user-input data collected in our previous work [4], [17] and covered a wide range of possible leader behaviors.

The robotic team was controlled either by the standard model-based GCM or by the proposed deep learning ANN-based approach (NRANN). When using the GCM, each robot implements eq. (3) to compute the connectivity control action necessary to keep the team connected, as described in Sec. III; while when using the NRANN, the leader robot implements NRANN$_l$ and the others NRANN$_f$, as described in Sec. IV. We considered a sampling time of 0.005 s and each data point carried information about the position, velocity, and acceleration of each robot, as well as the connectivity-maintenance action $\mathbf{F}_i^\lambda$, connectivity parameter λ_2, the eigenvector associated to λ_2 for each robot [4], and the imparted exogenous force $\mathbf{F}_i^e$.

We analyzed (i) the computation time, (ii) the global connectivity level λ_2, and (iii) the forces imparted to the robots, under both the GCM and the proposed NRANN approach for teams of $N_t = \{2, 4, 6, 10\}$ robots. In Sec. VI, we discuss the applicability of the results for $N_t > 10$.



(a) Mean difference in the connectivity force (N) for NRANN$_l$ (leader) vs number of robots N_t.

(b) Mean difference in the connectivity force (N) for NRANN$_f$ (follower) vs number of robots N_t.

(c) Connectivity level error rate vs number of robots N_t.

(d) Mean connectivity computation time (s) vs number of robots N_t.

Fig. 2. Experimental evaluation. We compared the standard GCM approach with the proposed NRANN control technique in terms of (a)-(b) connectivity force, (c) connectivity level, and (d) computation time.

Fig. 2a shows the mean error in the connectivity force for 2, 4, 6, and 10 robots for the leader robot, while Fig. 2b shows the same metric for the followers. This error is computed by averaging the mean absolute error on $\mathbf{F}_\lambda$ for all 15 episodes

$$e_{\mathbf{F}}^{N_t} = \frac{\sum_{i=1}^{15} \bar{F}_\lambda^i}{15},$$

where

$$\bar{F}_\lambda^i = \frac{\sum_{k=1}^{N_s} \|\mathbf{F}_\lambda(k)_{GCM} - \mathbf{F}_\lambda(k)_{NRANN}\|}{N_{steps}},$$

being i the episode at hand, N_s the number of samples in the episode, $\mathbf{F}_\lambda(k)_{GCM}$ and $\mathbf{F}_\lambda(k)_{NRANN}$ the connectivity force in the GCM and NRANN control conditions, respectively. Since we are interested in NRANN mimicking as close as possible GCM, the lower $e_{\mathbf{F}}^{N_t}$ the better. To understand whether the errors presented in Figs. 2a and 2b are satisfactory, we can evaluate them with respect to the mean and maximum levels of connectivity force registered throughout all the episodes, i.e., 0.099 N and 34.383 N, respectively. This means that the errors registered in Figs. 2a and 2b are lower than 7% and 0.02% (errors in the worst case, $N_t = 10$) of the mean and maximum connectivity forces registered during GCM conditions, respectively.

Fig. 2c shows the mean difference in the connectivity levels between GCM and NRANN. This error is computed by averaging the mean absolute error on λ_2 for all 15 episodes

$$e_\lambda^{N_t} = \frac{\sum_{i=1}^{15} \Lambda_i}{15},$$

where

$$\Lambda_i = \frac{\sum_{k=1}^{N_s} |\lambda_2(k)_{GCM} - \lambda_2(k)_{NRANN}|}{N_s}$$

i the episode at hand, N_s the number of samples in the episode, and $\lambda_2(k)_{GCM}$ and $\lambda_2(k)_{NRANN}$ the connectivity level in the GCM and NRANN control conditions, respectively. As for before, a lower e^n_λ means a closer performance between GCM and NRANN control conditions.

Finally, Fig. 2d shows the mean computation time of one control cycle for GCM and NRANN, again with 2, 4, 6, and 10 robots. This metrics is evaluated by averaging over the 15 episodes the mean computation time of one control loop for GCM and NRANN. Of course, the fastest the computation, the better it is. The connectivity was always maintained in all conditions and throughout all the trials.

B. Real world use case

We also tested our NRANN control in a real environment. An expert operator carried out a series of indoor teleoperation trials with a teams of two, three and four quadrotors. A video of two representative repetitions (for three and four drones) is available at https://youtu.be/IDHS8dMvSo0 and as supplementary material.

a) Real environment and task: We use quadrotors from MikroKopter GmbH (HiSystems GmbH, Germany), controlled using one computer. Each quadrotor runs an internal near-hovering control loop at 1 kHz; new desired accelerations from the control output are calculated on-board at 200 Hz. The PC (16 GB RAM, 4×2.80 GHz Intel Xeon CPU E5-1603 CPU, NVIDIA Quadro K2000) enables the control of Sec. IV on board the quadrotors using the Robotics Open Source Software (CNRS/LAAS, France) over Wi-Fi. The PC and the quadrotors communicate through the pocolibs middleware [48]. Robots positions are registered using an optical tracking system composed of twelve cameras, running at 100 Hz. Robots fly in a $7 \times 7 \times 3$ m room delimited by a safety net, as shown in Figs. 3 and 4. The task consisted in controlling multi-robot teams of either three and four quadrotors by moving their leader across the room and back, choosing different arbitrary patterns of motion. We carried out a total of eight trials. Four times the team was controlled using the standard model-based GCM control, four times using the proposed NRANN learning approach.

b) Subjects and operator's input: One 33 years old male expert operator took part in this experiment. He knew the connectivity control algorithm and how to control the robots. No practice trial was needed. For this experiment, the operator used an off-the-shelf joypad Gamepad F310 (Logitech, CH) for inputting the exogenous velocity to the leader robot; the left thumbstick provided x- and y-axis velocities, while the right thumbstick provided the z-axis velocity.

c) Results: Figures 3 and 4 show the robots trajectories for one representative trial with three and four quadrotors, respectively, using NRANN. Specifically, Figs. 3a and 4a show the room and its walls (left-hand side) as well as the robots trajectories at time $t = T_\mathrm{f}/2$ and $t = T_\mathrm{f}$ (central and right-hand sides). Each robot is identified with a color: red for the leader robot (robot 1), green for robot 2, blue for robot 3, and cyan for robot 4 (for Fig. 4). The circles indicate the robots starting position. Figs. 3b and 4b then show three shots of the scene, taken from a camera placed at $[\,4,\,-4,\,1.5\,]$ m. The connectivity was always maintained in all conditions and throughout all the trials.

VI. Discussion

The objective of this work is to show the feasibility of SDL-based control schemes to achieve scalable yet effective connectivity-maintenance distributed multi-robot control.

It is important to note the varying dimension of the input feature space for the leader $(4(N_t+1))$ and follower network $(4N_t+1)$ owing to N_t agents in the training environment, as also highlighted in eqs. (4) and (5). The ANN architecture is adjusted based on the desired level of accuracy and prediction time. The considered feature space is directly related to the fact that each robot only interacts with its neighbors, regardless of the *total* number of robots in the team. For example, for $N_t = 2$, the two drones are of course always connected. This is different with respect to cases with $N_t > 2$, where the connections between agents can change and evolve during the task. For this reason, we cannot simply replicate the behavior of a 2-drones team to cases with more agents. Instead, we carried out separate training routines depending on the number of drones in a team. Interestingly, since – considering connectivity – each robot only interacts with its neighbors, it is likely that an NRANN control policy trained for a relatively high N_t can work well with a much larger team, i.e., the NRANN only needs to manage the immediate neighbors of the agent, regardless the total number of agents in the team. How many neighbors each robot can have mostly depends on the type of sensors it is equipped with as well as of the type of task and environment. While there is no straightforward relationship between average number of neighbors and size of the team N_t, we can safely assume that, eventually, as the team size increases, the number of neighbors will saturate. This fact is also relevant when analyzing the results of Sec. V-A and Fig. 2. We can see from Figs. 2a–2c that NRANN approaches behave well with respect to the team connectivity force and level, showing an overall behavior similar to that of the GCM. On the other hand, as we see from Fig. 2d, the computation time for NRANN is almost invariant to the number of robots in the team, while it increases rapidly for GCM. This is of course expected. Since the GCM must compute the connectivity-maintenance action (see Sec. III) *for each neighbor*, it is expected that a larger team (and thus, more neighbors) leads to a higher computation time. On the other hand, the NRANN compute the connectivity-maintenance action in one step, regardless of the number of neighbors involved. It is worth noticing that the GCM computation time increases with the number of neighboring robots and not with the number of total robots in the team. This means that the computation time might eventually saturate for a sufficiently larger number of robots in team, as the number of neighbors does not increase indefinitely. The choice of using the proposed NRANN vs. the standard GCM should take into account all the above considerations.

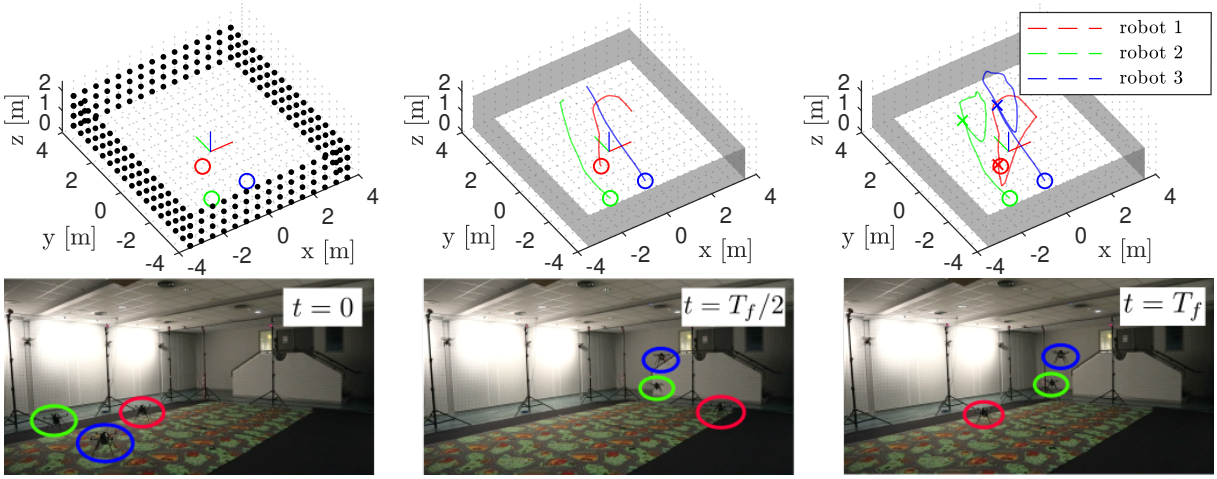


Fig. 3. Real world use case, employing three quadrotors controlled with the proposed NRANN policy. Each robot is identified with a color: red is the leader robot (robot 1), green and blue are robot 2 and robot 3, respectively. (a) from left to right: robot initial positions at time $t = 0$ and walls; robot trajectories at time $t = T_\mathrm{f}/2$; robot trajectories and positions at time $t = T_\mathrm{f}$. (b) shots of the experiment at time $t = 0$, $t = T_\mathrm{f}/2$, and $t = T_\mathrm{f} = 33$ s, from left to right, respectively, from two cameras.

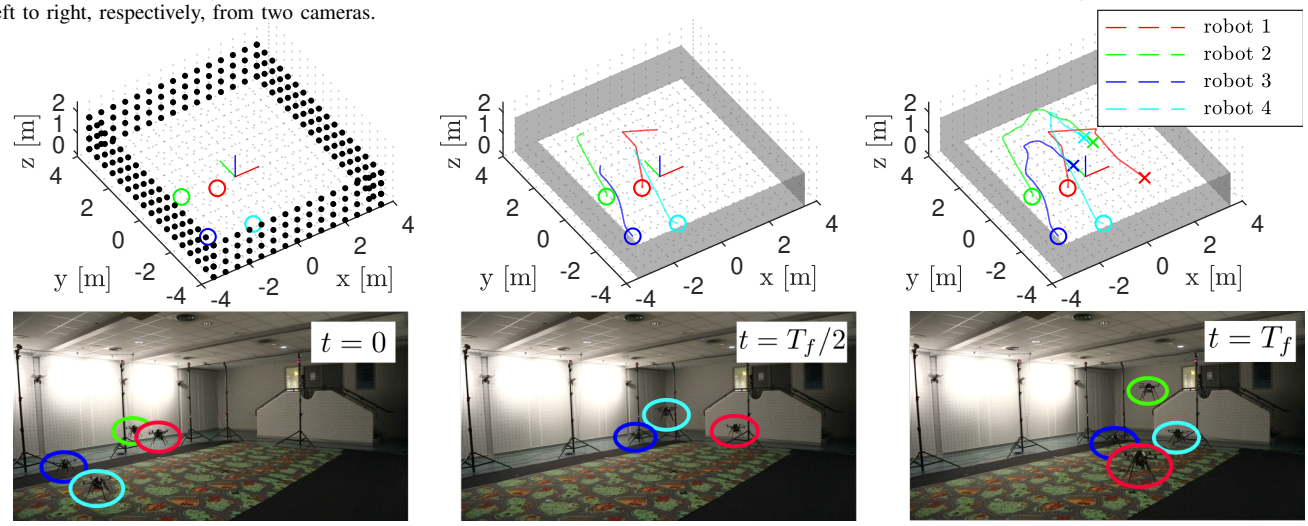


Fig. 4. Real world use case, employing four quadrotors controlled with the proposed NRANN policy. Each robot is identified with a color: red is the leader robot (robot 1), green, and blue, and cyan are robot 2, robot 3, and robot 4, respectively. (a) from left to right: robot initial positions at time $t = 0$ and walls; robot trajectories at time $t = T_\mathrm{f}/2$; robot trajectories and positions at time $t = T_\mathrm{f}$. (b) shots of the experiment at time $t = 0$, $t = T_\mathrm{f}/2$, and $t = T_\mathrm{f} = 19$ s, from left to right, respectively, from two cameras.

VII. Conclusions

We presented an SDL distributed connectivity-maintenance scheme for multi-robot systems. Starting from a standard model-based connectivity-maintenance control algorithm (GCM), we tested the viability of LSTM artificial recurrent neural networks (RNN) to run the same connectivity-maintenance behavior while achieving improvements in terms of computation time and scalability.

We trained control policies considering $N_t = \{2, 4, 6, 10\}$ robots in the team and analyzed their performance with respect to the original GCM in terms of mean connectivity force applied to the robots, connectivity level, and computation time. The proposed SDL technique was able to maintain the connectivity of the team throughout the considered simulated and real trials, achieving errors lower than 7% and 0.02% (worst case) of the mean and maximum connectivity forces registered during GCM control, respectively. At the same time, we witnessed a significant improvement in terms of computation time, with the proposed SDL techniques needing 83% less time than GCM when managing a team of ten robots (largest registered improvement). Since the GCM computes the connectivity-maintenance action for each neighbor, while the NRANN computes the same action in one step, we expect this gap to increase as the number of neighbors increase.

While the presented results are encouraging, there are however still some points to address. First, the proposed approach, although it always kept the team connected in over 3 hours of simulations, it does not *guarantee* the connectivity of the team as GCM does due to inherent black box nature of the ANN and generalization capability of the ANN under eluded scenario. This means there might exist some potential situations, which we did not encounter in our evaluation, where the SDL control may loose the connection of the team. To address this issue, we should consider Constrained ML techniques, in which restrictions and constraints can be imposed on different parts of the learning process. Another limitation is the need for a different training when changing

the number of robots in the team. As training takes (a lot of) time, it would be beneficial to train one SDL technique able to manage an arbitrary number of robots. Since each robot only interacts with its neighbors and the maximum number of neighbors is limited by the characteristics of the onboard sensors and the nature of the environment, we expect that an SDL control trained for a relatively large team of robots can work well with an arbitrarily larger team. Both of these situations present an opportunity for the future work.

ACKNOWLEDGMENT

This work was supported by the ANR-20-CHIA-0017 project "MULTISHARED".

REFERENCES

[1] R. Mendonça, M. M. Marques, F. Marques, A. Lourenco, E. Pinto, P. Santana, F. Coito, V. Lobo, and J. Barata, "A cooperative multi-robot team for the surveillance of shipwreck survivors at sea," in *Proc. IEEE/MTS OCEANS*, 2016, pp. 1–6.

[2] R. R. Murphy, K. L. Dreger, S. Newsome, J. Rodocker, E. Steimle, T. Kimura, K. Makabe, F. Matsuno, S. Tadokoro, and K. Kon, "Use of remotely operated marine vehicles at minamisanriku and rikuzentakata japan for disaster recovery," in *Proc. IEEE International symposium on safety, security, and rescue robotics*, 2011, pp. 19–25.

[3] 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.

[4] P. Robuffo Giordano, A. Franchi, C. Secchi, and H. H. Bülthoff, "A passivity-based decentralized strategy for generalized connectivity maintenance," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 299–323, 2013.

[5] C. Secchi, A. Franchi, H. H. Bülthoff, and P. Robuffo Giordano, "Bilateral Control of the Degree of Connectivity in Multiple Mobile-robot Teleoperation," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013, pp. 3645–3652.

[6] W. S. Ng and E. Sharlin, "Collocated interaction with flying robots," in *Proc. IEEE Int. Workshop on Robot and Human Interaction (ROMAN)*, 2011, pp. 143–149.

[7] A. Sanna, F. Lamberti, G. Paravati, and F. Manuri, "A Kinect-based natural interface for quadrotor control," *Entertainment Computing*, vol. 4, no. 3, pp. 179–186, 2013.

[8] A. Sarkar, K. A. Patel, R. G. Ram, and G. K. Capoor, "Gesture control of drone using a motion controller," in *Proc. Int. Conf. on Industrial Informatics and Computer Systems (CIICS)*, 2016, pp. 1–5.

[9] R. A. S. Fernández, J. L. Sanchez-Lopez, C. Sampedro, H. Bavle, M. Molina, and P. Campoy, "Natural user interfaces for human-drone multi-modal interaction," in *Proc. Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2016, pp. 1013–1022.

[10] M. Chandarana, A. Trujillo, K. Shimada, and B. D. Allen, "A natural interaction interface for uavs using intuitive gesture recognition," in *Advances in Human Factors in Robots and Unmanned Systems*, 2017, pp. 387–398.

[11] J. P. Hansen, A. Alapetite, I. S. MacKenzie, and E. Møllenbach, "The use of gaze to control drones," in *Proc. ACM Symp. on Eye Tracking Research and Applications*, 2014, pp. 27–34.

[12] C. Rognon, S. Mintchev, F. Dell'Agnola, A. Cherpillod, D. Atienza, and D. Floreano, "Flyjacket: An upper body soft exoskeleton for immersive drone control," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2362–2369, 2018.

[13] HYPERSUIT, "Hypersuit," https://web.archive.org/web/20190530103302/https://www.hypersuit.fr/, [Online; accessed May-2019].

[14] L. A. Sandru, M. F. Crainic, D. Savu, C. Moldovan, V. Dolga, and S. Preitl, "Automatic Control of a Quadcopter, AR. Drone, Using a Smart Glove," in *Proc. Int. Conf. on Control, Mechatronics and Automation*, 2016, pp. 92–98.

[15] A. Franchi, C. Secchi, H. I. Son, H. H. Bülthoff, and P. Robuffo Giordano, "Bilateral Teleoperation of Groups of Mobile Robots with Time-Varying Topology," *IEEE Trans. Robotics*, vol. 28, no. 5, pp. 1019–1033, 2012.

[16] D. Zelazo, A. Franchi, H. H. Bülthoff, and P. Robuffo Giordano, "Decentralized Rigidity Maintenance Control with Range Measurements for Multi-Robot Systems," *International Journal of Robotics Research*, vol. 34, no. 1, pp. 105–128, 2015.

[17] M. Aggravi, C. Pacchierotti, and P. R. Giordano, "Connectivity-maintenance teleoperation of a uav fleet with wearable haptic feedback," *IEEE Trans. Automation Science and Engineering*, pp. 1–20, 2020.

[18] D. Lee, A. Franchi, H. I. Son, C. Ha, H. H. Bülthoff, and P. Robuffo Giordano, "Semiautonomous haptic teleoperation control architecture of multiple unmanned aerial vehicles," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 4, pp. 1334–1345, 2013.

[19] F. Schiano, A. Franchi, D. Zelazo, and P. Robuffo Giordano, "A rigidity-based decentralized bearing formation controller for groups of quadrotor uavs," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 5099–5106.

[20] S. Y. Choi and D. Cha, "Unmanned aerial vehicles using machine learning for autonomous flight; state-of-the-art," *Advanced Robotics*, vol. 33, no. 6, pp. 265–277, 2019.

[21] M. Ö. Efe, "Neural Network Assisted Computationally Simple $PI^{\lambda}D^{\mu}$ Control of a Quadrotor UAV," *IEEE Trans. Industrial Informatics*, vol. 7, no. 2, pp. 354–361, May 2011.

[22] S. Wang, B. Li, and Q. Geng, "Research of RBF neural network PID control algorithm for longitudinal channel control of small UAV," in *2013 10th IEEE International Conference on Control and Automation (ICCA)*, June 2013, pp. 1824–1827.

[23] M. S. Nazeer, D. Bianchi, G. Campinoti, C. Laschi, and E. Falotico, "Policy adaptation using an online regressing network in a soft robotic arm," in *2023 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2023, pp. 1–7.

[24] M. S. Nazeer, C. Laschi, and E. Falotico, "Soft dagger: Sample-efficient imitation learning for control of soft robots," *Sensors*, vol. 23, no. 19, p. 8278, 2023.

[25] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, "Learning quadrotor dynamics using neural network for flight control," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 4653–4660.

[26] J. Shin, H. J. Kim, and Y. Kim", "Adaptive support vector regression for UAV flight control," *Neural Networks*, vol. 24, no. 1, pp. 109 – 120, 2011.

[27] S. Kurnaz, O. Cetin, and O. Kaynak, "Adaptive neuro-fuzzy inference system based autonomous flight control of unmanned air vehicles," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1229 – 1234, 2010.

[28] G. V. Chowdhary and E. N. Johnson, "Theory and flight-test validation of a concurrent-learning adaptive controller," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 2, pp. 592–607, 2011.

[29] "Parameter tuning for prediction-based quadcopter trajectory planning using learning automata," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 2341 – 2346, 2017, 20th IFAC World Congress.

[30] J. Junell, E.-J. Van Kampen, C. De Visser, and Q. Chu, "Reinforcement learning applied to a quadrotor guidance law in autonomous flight," 01 2015.

[31] E. M. Kan, M. H. Lim, Y. S. Ong, A. H. Tan, and S. P. Yeo, "Extreme learning machine terrain-based navigation for unmanned aerial vehicles," 03 2013, pp. 469–477.

[32] A. Loquercio, A. I. Maqueda, C. R. del-Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, April 2018.

[33] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-Guided Policy Search," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 528–535.

[34] S. Yang, S. Konam, C. Ma, S. Rosenthal, M. Veloso, and S. Scherer, "Obstacle avoidance through deep networks based intermediate perception," 2017.

[35] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.

[36] F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning vision-based flight in drone swarms by imitation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4523–4530, 2019.

[37] V. Derhami and Y. Momeni, "Applying Reinforcement Learning in Formation Control of Agents," in *Intelligent Distributed Computing IX*. Springer, 2016, pp. 297–307.

[38] R. John and O. Andersson, "Intelligent formation control using reinforcement learning," vol. Linkoping University, pp. 1–62, 01 2017.

[39] A. Geramifard, J. Redding, and J. P. How, "Intelligent cooperative control architecture: A framework for performance improvement using safe learning," *Journal of Intelligent & Robotic Systems*, vol. 72, no. 1, pp. 83–103, Oct 2013.
[40] D. Mox, V. Kumar, and A. Ribeiro, "Learning connectivity-maximizing network configurations," 2021.
[41] W. Huang, Y. Wang, and X. Yi, "Deep q-learning to preserve connectivity in multi-robot systems," 2017, pp. 45–50.
[42] J. Lin, X. Yang, P. Zheng, and H. Cheng, "End-to-end decentralized multi-robot navigation in unknown complex environments via deep reinforcement learning," 2019.
[43] M. Li, Y. Jie, Y. Kong, and H. Cheng, "Decentralized global connectivity maintenance for multi-robot navigation: A reinforcement learning approach," 2021.
[44] R. Szadkowski, M. S. Nazeer, M. Cianchetti, E. Falotico, and J. Faigl, "Bootstrapping the dynamic gait controller of the soft robot arm," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2669–2675.
[45] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[46] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.
[47] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.
[48] LAAS/CNRS, "Pocolibs middleware," https://www.openrobots.org/wiki/pocolibs, 2020, [Online; accessed Sep-2020].