Neural Network Based Algorithm for Multi-UAV Coverage Path Planning

Neural Network Based Algorithm for Multi-UAV Coverage Path Planning

Giovanni Sanna[1], Simone Godio[1] and Giorgio Guglieri[2]

Keywords: Neural Network, Artificial Intelligence, UAV, Supervised Imitation Learning, Urban Coverage Path Planning.

***Abstract* - This paper proposes a method to tackle the Coverage Path Planning (CPP) problem for a fleet of AI-driven UAVs while accounting for congestion, collision avoidance, and efficiency of the path. The algorithm relies on a mixed-use of decentralized Artificial Neural Networks (ANN) and the A* pathfinder. Each UAV has elementary cognitive skills to sense information about the nearby environment which are then fed as an input to the network. The neural network creates a correlation between the current state of a UAV and the best action to take at each time step. The exploration strategy is stored in a labeled database for the single-UAV case; the neural network learns and replicates it in the multi-UAV case, being able to generalize the acquired skills over new maps not contained in the database. The training session imitates human priors in a multi-class classification, which completely bypasses common drawbacks such as the need for large databases or high computational resources. The case study focuses on complex urban areas, for which the grid resolution of the traditional approaches can't model the problem with sufficient accuracy.**

I. Introduction

Multi-robot Coverage Path Planning (mCPP) is a well-known problem that interests researchers all over the world. The growing popularity of Unmanned Aerial Systems (UAS) sheds light on new fields of application: among them, the aerial CPP in an urban environment has very unique features for which the use of traditional coverage algorithms could prove problematic. Coverage Path Planning is defined as: [16]"*Coverage Path Planning is the task of determining a path that passes overall points of an area or volume of interest while avoiding obstacles.*" This task is integral to many robotic applications, such as vacuum cleaning robots, painter robots, demining robots, lawnmowers, automated harvesters, window cleaners, and inspection of complex structures[16]. For UAV applications, a slight deviation from the traditional CPP definition leads to a complete change of paradigm. In fact, because of their very nature, UAVs do not need to physically pass over through all the points in the area: to explore the scene, it is sufficient for the point to be within the frame of the camera. Consider one single UAV, equipped with a camera pointing downwards to capture the scene underneath. The camera has a square cropped footprint $S \times S$, as shown in Fig[1]. If H [m] is the height from the ground, and FOV is the camera Field Of View [deg], it's possible to obtain the side S [m] of the camera footprint, by calculating:

$$S = 2H \cdot tan\left(\frac{FOV}{2}\right) \tag{1}$$

In [17][16][15][11], accurate description of mCPP state of the art analysis is reported in line with the established scientific research field. In order to plan a path, most approaches use space discretization over an occupancy grid, which allows the creation of known optimal trajectories e.g. the *"back and forth"* or *"spiral"*, while avoiding image overlapping between consequential scans. Those traditional approaches align the camera FOV with the cells of the occupancy grid, as shown in Fig[2.(a)]: this makes

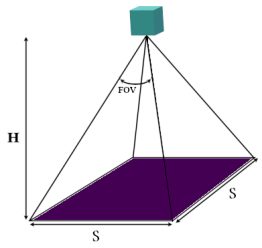

Fig. 1. UAV's camera cone of visibility: in purple, according to the Field Of View, the camera ground footprint; in teal, a single UAV.

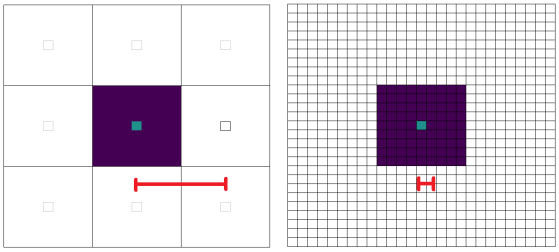

Fig. 2. Top View: (a) Traditional approaches' space discretization (b) Our space discretization.

[1]Giovanni Sanna and Simone Godio are with the Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Torino, Italy, giovanni.sanna@studenti.polito.it, simone.godio@polito.it

[2]Giorgio Guglieri is with the Department of Mechanical and Aerospace Engineering, Politecnico di Torino, PIC4SeR, CNR-IEIIT, Torino, Italy, giorgio.guglieri@polito.it

it possible to treat UAV applications just like any other terrestrial robot. This approach is useful in surroundings with low obstacle density and in applications such [14] surveillance, [6] smart farming, [1] photogrammetry, disaster management and [7] wildfire tracking.[16]The offline mCPP literature has a plenty of elegant solutions: e.g. in [8] the multi-agent decision making is delegated to a dynamic cost-map, while other architectures include [4]wavefront algorithm, [9]fractal trajectories, [13]harmony search, [2]genetic algorithm, and [3]chaotic ant colony. In these applications, the displacement between two adjacent cells is equal to *S*, as shown in Fig[2.(a)]. However, traditional methods are not useful in urban applications with high density of complex-shaped obstacles. Too low resolution would miss urban complexity, too high resolution would require low flight height from terrain and non-optimal trajectories in urban environments with large, obstacle-free, explorable areas.

This work presents a Multi-UAV CPP algorithm that uses a compatible space resolution with real urban maps. The tighten grid Fig[2.(b)] allows planning smoother and precise trajectories that can follow urban shapes. The core of the algorithm uses a pre-trained Artificial Neural Network (ANN) to decide what action to take at each-timestep. Over the years, the use of ANNs on the coverage path planning problem has been explored both in the trained and non-trained versions. For those applications, the training stage is usually faced with Multi-Agent Reinforcement Learning (MARL). There are several MARL architecture that can be implemented, each one with its pros and cons. The basic idea behind decentralized architectures is: each agent (UAV) senses its state from the environment (e.g. presence of obstacles). Based on its state, it takes actions (e.g. move right) and interacts with the environment (the map): the agent receives positive rewards if the action had positive consequences or vice-versa. After a long training, the agent learns a policy to map states to actions and maximize the expected discounted return. This process is not trivial at all. In 2019, Google DeepMind showed MARL incredible capabilities by creating [5]AlphaStar, an AI-driven player of the game StarCraft II, which defeated several professional players. The action space was so huge that each agent would need an equivalent experience of up to 200 years of real-time plays to reach useful performances. In one step of the training, the AI was trained on Human priors. This technique, called Imitation Learning, is usually implemented with [10]Reinforcement Learning. In this paper, we present Imitation Learning capabilities used in conjunction with Supervised Learning, where a set of labeled experiences are used for CPP purposes.

II. Assumptions

In the proposed work, the simulation is discrete in time and space. The space discretization uses an occupancy grid over a matrix. Each cell of the matrix can be in only one of the following states: unexplored, explored, obstacle. Since maps are inspired by real scenarios, real obstacles are approximated due to the nature of the matrix map: if even part of the obstacle lies inside a cell, then the whole cell is considered as an obstacle cell, as shown in Fig 3. The obstacle cells' distribution and shape are both known a priori. The path is planned offline as obstacles are fixed. The fleet consists of a variable number of agents with the same technical capabilities - each agent occupies always one and only one cell in the map, and the correspondent cell is set as an obstacle cell. At each time-step, each agent explores all nearby cells in the line of sight with its camera FOV: the camera, always perpendicular to the ground, has a squared footprint with side $S \in \mathbb{R}$. Furthermore, without loss of generality, it is assumed that the movement of the agent is defined with four directions, specifically Forward, Right, Down, Backward. Since no camera rotation action is allowed, the UAVs are always heading up. The camera FOV takes 9×9 cells and is always centered in the UAV. Compared to the traditional approach, where each action would lead to a displacement equal to S, in this work the displacement at each time-step is equal to $S/9$, equivalent to 1 $cell$, as shown in Fig 2. The height of flight from terrain is fixed so that the problem can be treated in two dimensions. Finally, it is assumed that all unexplored cells can be reached from at least one agent.

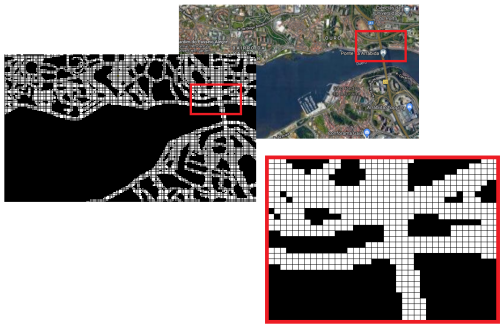

Fig. 3. Case study 4 - an example of our grid resolution inspired by real urban cases. From top to bottom images are shown, in order: satellite view; relative map over the occupancy grid; a zoomed portion of the map: the portion is the one inside the rectangle.

III. Algorithm Overview

The proposed approach features the following:

- environment partitioning using K-Means.
- use of a neural network for short-term planning.
- use of an A*-like algorithm for longer-term planning.

The method is validated in simulation. Let $A \in \mathbb{N}$ be the number of agents: each agent is denoted with a_i with $i = 1, ..., A$. The main map, m_0, is a $h \times w$ matrix. Each cell of m_0 can be in only one of those three states:

- explored
- unexplored
- obstacle

Unexplored cells turn into explored cells when being visited by at least one agent. The purpose of the algorithm is to

plan coordinate trajectories in such manner that unexplored cells are visited by at least one UAV. An unexplored cell turns its state into explored when it is contained inside the camera FOV of a UAV (hence an area of 9×9 cells centered on a UAV) and when no obstacle cells stand between the UAV and the cell. Each UAV occupies always one and only one cell of the map - when it takes one of the four actions, it moves to an adjacent cell according to the choice. To avoid collisions, each agent senses other agents as obstacle cells. Other types of obstacles (e.g. buildings) are fixed, cause they are a unique feature of the environment. To divide the exploration workload, each UAV is unequivocally assigned to an exploration area, and the planning takes place so that each UAV prioritizes the visit in its corresponding area. The environment, hence the set of unexplored cells, is split into exploration zones by means of the K-means algorithm, as shown in Fig 4, following criteria explained in Section IV.

The algorithm can be thought in three steps:

Step 1: Area division – unexplored cells of m_0 are subcopied into maps m_i with $i = 1..A$. This divides the target explorable area - as shown in Fig 4;

Step 2: Allocation – each UAV is assigned to a specific area: its task is to completely explore its area;

Step 3: Simulation – Exploration via Artificial Intelligence, Section V, and Explorative A*, Section VI.

Each UAV has a local view on the map to decide which action to take. The local view is a squared 19x19 grid centered in the agent. In this sense, the $camera\ field\ of\ view \subset local\ view$. If the local view contains unexplored cells, then decision making is driven by the Artificial Neural Network, Section V, else, the UAV enlarges its local view until it finds an unexplored cell using A*/Explorative A*, Section VI.

IV. Area decomposition

The target explorable space is decomposed into subregions of exploration. This allows achieving a better UAVs spatial distribution while enhancing coverage performance. The decomposition is done with an unsupervised algorithm called K-means. The K-means clustering algorithm attempts to split a given anonymous data set (such as unexplored points of the map) into a fixed number K of clusters. At the end of the algorithm, each cell of the map, initially anonymous, will have a unique label that identifies its belonging to the corresponding exploration zone. The algorithm relies on two elements: points and centroids. A number of $P \in \mathbb{N}$ of points $x_1, x_2, .., x_P$ are randomly spread over all unexplored cells. A number of $K \in \mathbb{N}$ centroids $c_1, c_2, .., c_K$ are placed in each UAV's starting position. Since each agent will be assigned to one subregion, the number of clusters is equal to the number of UAVs, $K = A$. The iteration can be summarized as shown in the following pseudo-code [1]:

Notice that the function $D(x_i, c_j) : \mathbb{R}^4 \to \mathbb{R}$ computes the Euclidean Distance between the positions of the *i-th* point and the *j-th* centroid, given in terms of row and column of the correspondent position in the matrix

Algorithm 1 K-means clustering pseudo-code

```
Place points x_1, ..., x_P over unexplored cells, randomly
Place centroids c_1, ..., c_K in each UAV's starting position
while not converge do
  for each point x_i, i = 1, ..., P do
    find nearest centroid c_j {arg min D(x_i, c_j)}
    assign the point x_i to cluster c_j
  end for
  for each centroid c_j, j = 1, ..., K do
    count the number N of its assigned points x_a,
    compute mean position of all the assigned point x_a,
    move c_j to that new mean position:
    c_j = (1/N) · Σ x_a
  end for
  Stop when none of the cluster assignment change
end while
from c_1, ..., c_K, calculate Voronoi Diagram and divide
the target area. =0
```

map. Once the final position of the centroids has been determined, the area is divided by using Voronoi Diagrams to create sub-areas. Each sub-area is then assigned to the correspondent UAV, as shown in Figure 4 and, importantly, each UAV can sense or view only unexplored points of its assigned submap. Moreover, once a UAV has fully explored its assigned area, its sight capabilities are enhanced so that it can sense unexplored cells of the other UAVs area. Therefore, the collaboration between parts is encouraged and exploration results are improved.

V. The Neural Network

A trained Artificial Neural Network (ANN) is the core of decision-making. It acts as the "UAV's brain" and decides what action to take. The input of the ANN is the locally sensed state of a UAV. The sensed state is then propagated forward through all the hidden layers until it reaches the output layer, where one of the four actions (in general, classes) is selected. Despite the number of agents A being variable, only one ANN is used. This means that all agents share the same weight set hence the same way of making decisions: at each time step, the ANN performs several forwarding propagations up to A, the number of agents. Since each agent has a limited view of nearby cells, the ANN is not always used, as described below. The discriminating factor of this choice is the state. The state is a feature vector of 376 units (or neurons) fed as an input to the network. The first 360 entries contain information about the local view of the UAV. This local view is a 19x19 matrix, reshaped into a column vector. Since the local view is always centered on the UAV, the central cell does not contain relevant information. By counting the number of unexplored cells in the local view, the agent decides whether to use ANN or the A*/Explorative A*. If the number of unexplored cells inside the local view is zero, the agent does not have enough information, and any action

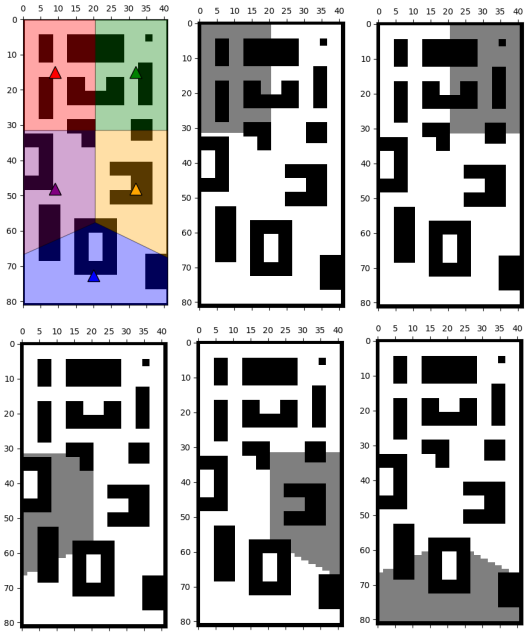

Fig. 4. Environment partition example - Case study map - Five agents. In the upper left map, triangles show the final centroids' position of the K-means clustering algorithm and the relative Voronoi Diagram. In the other maps, black cells indicate obstacles, white cells indicate explored cells and gray cells indicate unexplored cells.

taken will not lead to a strategy, therefore Explorative A* is chosen, Section VI. Each cell of the local view is fed in the correspondent neuron of the input layer following the conversion:

- +1, if the cell is unexplored;
- 0, if the cell is explored;
- -1, if the cell is an obstacle.

Consequently, the attraction contribution of each nearby cell is sorted in ascending order. As to the remaining units of the state, 8 neurons are used in to remember the previous two actions at time-step $t-1$ and $t-2$. Four units for the respective four actions. Only the corresponding neuron is activated; all the others are set to 0. This information is a key element for the ANN approach. The memory of the past gives a kind of logical inertia for which trajectories result in increased smoothness and linearity. The remaining 8 neurons are used as detectors for obstacle cells and unexplored cells in the main four directions: forward, right, left, backward. The detectors sense in a range of 25 cells for obstacle cells and 40 cells for unexplored cells, obviously greater than the local view and multi-hot-encoded. This leads the UAV to prioritize certain types of exploration. The complete architecture is shown in Fig[5]. The training uses a dropout layer with a deactivation level of 30%: this is fundamental to avoid training data over-fitting.

The training process is fully guided by the *Adaptive Moment Estimation* algorithm, an optimizer that leads the state of art in neural network training. The loss function used is the categorical cross-entropy, which compares the

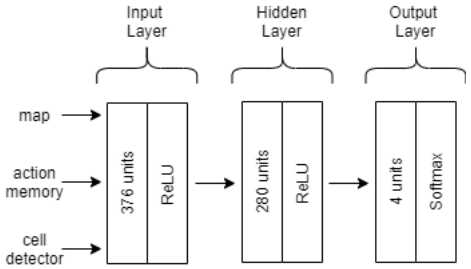

Fig. 5. Artificial Neural Network architecture.

predicted probability distribution with the target probability distribution:

$$Loss = -\frac{1}{E}\sum_{e=1}^{E}\sum_{c=1}^{C} t_{c,e}\log \hat{y_{c,e}} \tag{2}$$

, where E is the number of sample experiences, C the number of output classes, $t_{c,e} \in [0,1]$ is the target prediction and $\hat{y_{c,e}} \in [0,1]$ is the real neural network's prediction. The training session revises all the network's trainable parameters, such as weight and biases, in order to minimise the loss function. Moreover, the trained network is also able to predict the best actions to take in a never-seen situation. One drawback of using neural networks is the need for large databases with labeled samples, often difficult to find. Our method overcomes this problem by creating the database from scratch with relatively little effort. The creation of the database exploits a framework thanks to which is possible to manually guide a single UAV in a selected environment. This process can be run by both an expert human or an explicit algorithm. In this work, entries are created based on human priors: at each time-step, the inserted action is paired with the corresponding actual UAV state and registered in the database. The result is a structured database with a number of E entries, each one containing a state-action pair. Once the database is created, it is possible to create new data from the existing entries: this process is called data augmentation. Each entry can be mirrored vertically, horizontally, and diagonally, creating a new database four times larger than the starting one. By rotating the initial state of $\pi/2$ it is possible to repeat the mirroring process. The final result leads to a net eight-time augmented data-set. Data augmentation often leads to an incongruous situation: if several entries with different labels correspond to the same state, learning is undermined. Database cleaning removes both redundancies and incongruous states, resulting in increased consistency and training accuracy. The learning curve is shown in Fig[6].

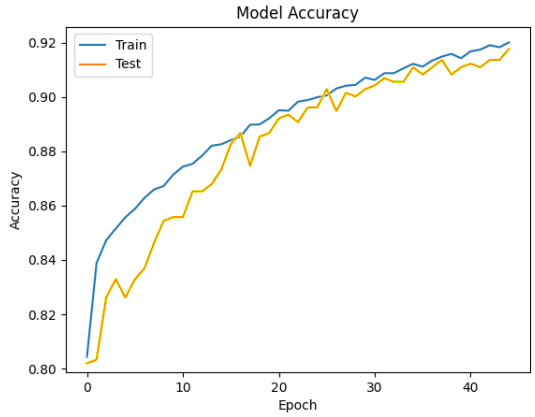

Fig. 6. Training session - Neural network learning curve - increasing accuracy over each training epoch.

The learning process reached a 92.01% of accuracy over a database with $54^{'}531$ labeled entries. After establishing the quality of the training process, a key element of the UAVs' performance relies upon the strategy used by the human expert to create the database. Three guidelines drive the strategy:

- Minimize the number of time-steps.
- Reduce the number of turning maneuvers.
- Avoid going twice through the same cell.

The number of turning maneuvers is a relevant metric evaluation since the UAV's energetic payload is limited, as claimed in[12]. Finally, not using the same cell twice should reduce the number of redundant movements. Taking into account all these elements while manually driving a multi-agent system in a complex environment is unsustainable for a human. For this reason, the experiences are collected in a single-agent and single sub-area simulation. The human priors technique is applicable only with a limited action-space: for this reason, only four directions of movement are available.

VI. Explorative A*

The local sight of each UAV often leads to situations where no unexplored point lies inside their view, in which case the neural network approach is not useful. This issue occurs especially towards the end of the exploration when few cells are missing. In these cases, an explicit pathfinder is used. The algorithm is articulated in three main steps:

Step 1: Goal cell – enlarge the local view in order to detect the nearest unexplored point and set it as the goal.

Step 2: Explorative A* – check if Explorative A* convergences from the UAV position toward the goal.

Step 3: A* – If Explorative A* did not converge, then try to use the classic version of A*.

The A* algorithm is one of the most established and efficient pathfinders which uses heuristics to guide its search. The high speed of convergence is guaranteed by construction, as a "best solutions-first" policy is used,

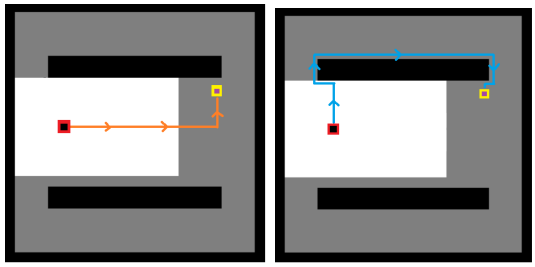

Fig. 7. Traditional A* and Explorative A* in comparison, same situation. The UAV is located in the red square; the goal cell to reach is represented in yellow. The left figure shows the path with Traditional A*, which is the shortest path; the right figure shows the path with Explorative A*, which account both the length of the path and the exploration of new cells (in grey).

which suites perfectly the high interconnection of urban environments. Compared to the classic A*, where the aim is to find the path that minimizes the number of movements, Explorative A* is an A*-like algorithm created ad-hoc for this paper purpose that searches for the shortest path that both reaches the goal and maximize the exploration rate. A practical example is shown in Fig[7] but it is not discussed in detail. Although dynamic versions of A* exist, this work uses a static version. If a UAV encounters an obstacle while following a planned path, that obstacle can only be another agent. In those cases, the UAV hovers for a time-step and waits for the other agent to move. Several measures are coded to improve efficiency. If an agent is following its A*/Explorative A* planned path towards the goal and another agent explores its goal cell, then a new path is planned toward a new goal cell. Moreover, if a UAV encounters unexplored cells during the planned path such that at least one unexplored cell lies inside its local FOV, then A*/Explorative A* is deactivated and Neural Network decision-making activates.

VII. Simulations

The evaluation metrics offer a clear and objective basis to judge the quality of the results. Six parameters are considered:

$Mean\ Moves\ Num.\ (MMN)$	$\sum_i M_i \frac{1}{A}$
$Trajectory\ Efficiency\ (TE)$	$\sum_{i,t} \frac{E_{i,t}}{9} \cdot \frac{1}{A \cdot MMN}\ \%$
$Mean\ Num.\ Turn.(MNT)$	$\sum_{i,t} T_{i,t} \cdot k_\theta \frac{1}{A \cdot MMN}\%$
$Mean\ UAVs\ Dist.^*\ (MUD)$	$\sum_{i,j,t} D(a_{i,t}, a_{j,t}) \frac{2 \cdot T_r^{-1}}{A^2 - A}$
$Mean\ Dist.\ Unexp.^*(MDU)$	$\sum_{i,j,t} min(D(a_{i,t}, u_{j,t})) \frac{T_r^{-1}}{U_t}$

- Mean Moves Number (MMN): the smaller MMN, the more successful the overall strategy is: the parameter is calculated dividing the sum over the number of total moves M_i of the $i-th$ UAV by the UAV number A.
- Trajectory Efficiency (TE): considers the exploration quality: since each UAV can explore up to 9 cells per

action, the exploration rate $E_{i,t}$ of the $i-th$ UAV at timestep t is divided by 9. The overall sum of E_i, t is then divided by both the number of agents A and the MMN.

• Mean Number of Turns (MNT): this metric is considered since the energetic limitations must be accounted for: turnovers are an important factor in energy dissipation. Therefore, the sum of all turnovers $T_{i,t}$ at time t by the agent $i-th$, is multiplied by a factor k_θ, which can be equal to 1 if the turnover is 180° or equal to 0.5 if the turnover is 90°. The overall sum is divided by the number of agents A and by the Mean Moves Number.

• Mean UAVs Distance (MUD^*): indicates how distributed are the UAVs. The symbol * indicates that the quantity is calculated excluding initial and final time-steps. Empirically, this condition is that one in which the percentage of exploration is contained between 30% and 70%. The function $D(a_i, a_j)$ is the Euclidean distance between the agent a_i and the agent a_j. The positions are given in terms of the correspondent row and column of the UAV position in the matrix map. The sum of distances is then divided by the number of timesteps a regime T_r and by the number of all possible combinations of the relative distances between the UAVs. For A UAVs, the number of possible pairs is $(A)(A-1)/2$.

• Mean Distance from Unexplored cells (MDU^*): considers the overall coverage- a regime. For each unexplored cell of the map u_j the Euclidean distance between the cell u_j and the nearest UAV a_i is considered. The overall sum of these distances is then divided by the number of timesteps a regime T_r and the number of U_t at timestep t.

The first batch of results is presented for a simple geometry case study to appreciate a visual representation of the trajectories. The case study shown in Figure 8 carries out a complete coverage path planning with a number of agents/UAVs equal to 5. In Table I below, evaluation metrics are implemented for the case study map, a grid 82×42, with an increasing number of agents from 3 to 6:

TABLE I

CASE STUDY 1 - RESULTS - INDOOR CASE - FIG 8

$Num.\ of\ UAVs$	3	4	5	6
MMN	183	168	133	126
TE	41.1%	32.8%	32.4%	27.1%
MNT	6.73%	7.21%	6.31%	5.88%
MUD	22.25	31.48	38.54	35.31
MDU	19.55	14.04	13.39	13.61

As can be seen from Table I, the higher the number of UAVs that simultaneously explore the map, the lower the number of movements that lead to complete exploration. On the other hand, the increased number of agents leads to a performance reduction in terms of trajectory efficiency, and UAVs tend more frequently to pass over cells that have been already explored. The weighted number of turning maneuvers appears to be independent of the UAV's number: this is the consequence of sharing a single neural network, hence the same way of acting. For tiny maps as this Case Study 1, both Mean UAVs' Distance (MDU) and Mean Distance from Unexplored cells (MUD) are dependent on the UAVs starting position. However, good explorations have both high MDU and low MUD, to have UAVs located in strategic positions.

The following three case studies are based on real cases occupancy grids. Case Study 2 is inspired by a view of Turin's Porta Nuova, Italy, as shown in Figure and in the table below.

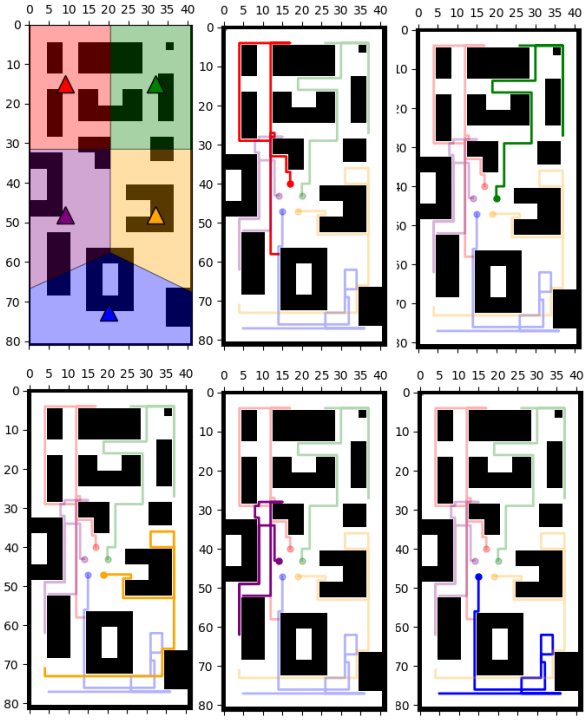

Fig. 8. Trajectories in the map Case Study I. The upper central image shows the overall UAVs' track, according to the area division shown in Fig 4. In the other images, each singular UAV trajectory is highlighted.

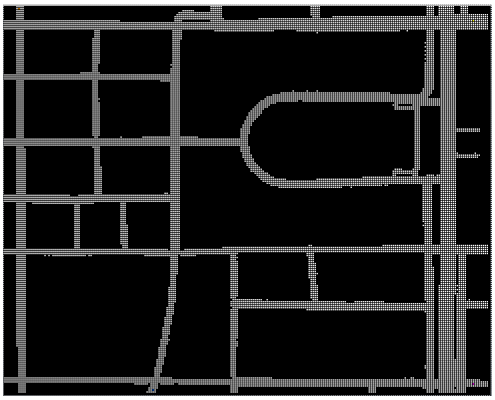

Fig. 9. Occupancy grid - Turin, Porta Nuova - dim.: 247×195 cells.

TABLE II

CASE STUDY 2- RESULTS - TURIN, PORTA NUOVA - FIG 9

$Num.\ of\ UAVs$	3	4	5	6
MMN	1019	854	827	591
TE	37.5%	33.5%	27.5%	32.1%
MNT	4.01%	4.51%	5.27%	4.01%
MUD	153.10	118.96	132.97	120.33
MDU	72.61	54.18	50.37	48.76

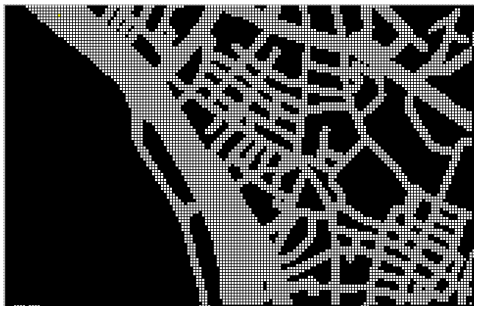

Fig. 10. Occupancy grid - Genova, Porto Antico - dim.: 104×161 cells.

TABLE III

CASE STUDY 3- RESULTS - GENOVA, PORTO ANTICO FIG 10

$Num.\ of\ UAVs$	3	4	5	6
MMN	1012	674	553	482
TE	23.6%	26.3%	26.4%	24.23%
MNT	21.2%	20.8%	19.39%	20.40%
MUD	53.76	69.48	62.99	65.73
MDU	41.91	25.33	26.96	22.09

Fig. 11. Occupancy grid - Porto, Douro River - 133×199 cells.

TABLE IV

CASE STUDY 4- RESULTS - PORTO, DOURO RIVER - FIG 11

$Num.\ of\ UAVs$	3	4	5	6
MMN	1537	1157	911	791
TE	21.7%	21.6 %	21.9%	20.8%
MNT	19.43%	18.56%	17.56%	18.05%
MUD	131.11	98.27	95.34	90.13
MDU	46.01	38.01	38.45	32.71

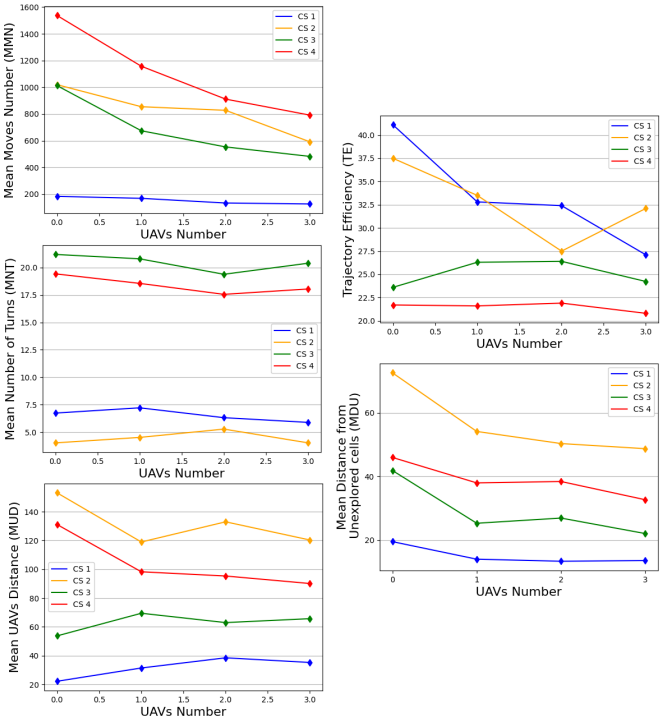

Fig. 12. An overall visual representation of the five evaluation metrics for each of the four Case Studies (CS).

VIII. CONCLUSIONS AND FUTURE WORKS

In this paper, we present an innovative approach to solve the Coverage Path Planning problem with a fleet of UAVs. In the context of machine learning and neural networks, the CPP problem has usually been tackled with Reinforcement Learning techniques. If, on the one hand, this approach can model the problem successfully, on the other hand, the

training session requires a huge amount of computational resources. Supervised Learning can be used in decision-making applications as well, and not limited to regression and classification problems. To create a labeled database, it is necessary to acquire experience from an expert system - this work solves the problem by imitating human priors. For decision-making applications, the supervised imitation learning usage is confined to simplified problems, where the action space is limited and a human can actually guide the solution. Although the experience collection was done offline and for the single-agent case, the final application uses multiple UAVs on real urban occupancy grids, thanks to the generalization abilities.

Future works foresee the improvement of the environment partition with the K-means algorithm, enhanced motion model, and increased collaborations between agents. Moreover, deeper neural networks should be explored, in conjunction with convolutional layers and different network architectures. Finally, particular attention is paid to the selection of the state vector, whose selection is decisive in the success or failure of the entire work.

REFERENCES

[1] A. Cesetti et Al. *A Visual Global Positioning System for Unmanned Aerial Vehicles Used in Photogrammetric Applications*. Springer, 2011.

[2] A. Sonmez et al. *Optimal path planning for UAVs using Genetic Algorithm*. IEEE, 2015.

[3] D. Zhang et al. *UAV Path Planning Based on Chaos Ant Colony Algorithm*. ResearchGate, 2015.

[4] L. Nam et al. *An Approach for Coverage Path Planning for UAVs*. Comput. Electron. Agric, 2016.

[5] O. Vinyals et al. *Grandmaster level in StarCraft II using multi-agent reinforcement learning*. nature, 2019.

[6] P. Lottes et al. *UAV-Based Crop and Weed Classification for Smart Farming*. IEEE, 2017.

[7] Pham H.X. et al. *A Distributed Control Framework for A Team of Unmanned Aerial.* IEEE, 2017.

[8] S. Godio et al. *A Bioinspired Neural Network-Based Approach for Cooperative Coverage Planning of UAVs*. information, 2021.

[9] S. Sadat et al. *Fractal Trajectories for Online Non-Uniform Aerial Coverage*. IEEE, 2015.

[10] W. H. Guss et al. *NeurIPS 2019 Competition: The MineRL Competition on Sample Efficient Reinforcement Learning using Human Priors*. ResearchGate, 2021.

[11] X. Zhou et al. *Survey on path and view planning for UAVs*. ScienceDirect, 2019.

[12] Y. Choi et al. *Energy-Constrained Multi-UAV Coverage Path Planning for an Aerial Imagery Mission Using Column Generation*. Springer, 2020.

[13] J. Valente et al. *Aerial coverage optimization in precision agriculture management: A musical harmony inspired approach*. Comput. Electron. Agric, 2013.

[14] N. Basilico; S. Carpin. *Deploying teams of heterogeneous UAVs in cooperative two-level surveillance missions*. IEEE, 2015.

[15] E. Galceranm M. Carreras. *A Survey on Coverage Path Planning for Robotics*. ScienceDirect, 2019.

[16] H. Choset. *Coverage for robotics - A survey of recent results*. Springer, 2001.

[17] Lisane B. Brisolara Tauã M. Cabreira and Ferreira Paulo R. Jr. *Survey on Coverage Path Planning with Unmanned Aerial Vehicles*. ResearchGate, 2019.

ACKNOWLEDGEMENTS

This work is also based on the MSc thesis work of Giovanni Sanna - Aerospace Engineering student at Politecnico di Torino, Italy.

The Ph.D. fellowship of Simone Godio is funded by the Leonardo Company, Italy. The research program is shared with PIC4SeR—Politecnico di Torino Interdepartmental Centre for Service Robotics.