Proceedings of 2024 IEEE 12th International Conference on Intelligent Systems (IS)

RRT*-Based Leader-Follower Trajectory Planning and Tracking in Multi-Agent Systems

Catalina Agachi
Department of Computer Science
University College London, UK
Email: catalina.agachi.19@ucl.ac.uk

Farshad Arvin
Department of Computer Science
Durham University, UK
Email: farshad.arvin@durham.ac.uk

Junyan Hu
Department of Computer Science
Durham University, UK
Email: junyan.hu@durham.ac.uk

***Abstract*—Coordination of multi-agent systems has received significant attention during the past few years owing to its wide real-world applications, such as cooperative exploration, aircraft formation, and autonomous vehicle platooning. To address this issue, this research presents a novel method for multi-agent systems to navigate through environments with obstacles. The system consists of a group of agents with a leader-follower structure, where the leader aids in guiding the agents toward the target location and the followers are steered to maintain a flexible formation. To achieve cooperation, the agents communicate within a connected and undirected network, exchanging information within a specific radius. The leader's path is generated using the RRT* algorithm, which serves as a reference for the followers. A control law utilizes consensus and APF is then implemented, ensuring coordinated motion while maintaining safe distances among agents and between agents and obstacles. Finally, the effectiveness of the developed two-layer coordination strategy is verified by simulations.**

***Keywords*—Path planning; cooperative control; multi-robot systems; swarm robotics.**

I. Introduction

Multi-agent systems (MASs) include multiple intelligent agents that interact in a shared environment to achieve collective goals, which have been widely implemented in some real-world applications, e.g., vehicle platooning [1], [2], area coverage [3]–[5], search and rescue [6], robot and bee colony interactions [7], etc. Due to the existence of multiple cooperating agents, MASs may provide a more efficient approach to problem-solving, as they can handle situations that are difficult to be solved by a single agent. Besides, they can collectively work towards achieving a common goal, thus resulting in high efficiency and robustness. There are various challenges in the research area of MASs. One of the main challenges is the limited communication among agents. Another challenge is decentralized decision-making, where each agent acts independently based on its local perception and its own goals. Therefore, control protocols have to ensure that the behaviors and actions taken by the agents can lead to reaching the global goal.

Motivated by natural behaviors, such as bird flocks, fish schools, and bee colonies, distributed control aims to coordinate a team of robots to achieve desired collective movements through local information. Many studies were conducted to develop control protocols for MASs. One of them has been conducted by Couzin et al. [8] and it demonstrated that by using controllers that are based only on state exchange from the close neighbors, a swarm of agents can have a global impact in the speed and direction of motion of the whole group towards the final destination. Another study [9] proposed a discrete-time model for MASs that are traveling in the same plane and have the same speed, where the communication topology of the systems changes at every point in time, and where there is a leader with a fixed heading. Having a dynamic communication topology leads to the set of neighbors of each agents constantly changing. The research proves that the headings of all agents must converge if all the agents are linked directly or indirectly to their leader. A similar study [10] looked at the dynamics of MASs that have the same velocity, use decentralized nearest-neighbors rules, and exchange their information on networks that change over time. Additionally, nonsmooth analysis was used to accommodate for the unexpected network topology change. Besides, it was proved that convergence to a common velocity vector is guaranteed as long as the communication network of the agents stays connected. However, in the aforementioned studies, the movement of the whole group in a constraint environment (e.g., containing obstacles) was not addressed.

In the context of MASs, efficient path planning is crucial for achieving coordinated agent motion. To navigate an MAS safely in an unknown environment, reliable and efficient planning algorithms will need to be considered in the protocol design. For a single agent path finding problem, Rapidly-Exploring Random Tree (RRT) [11] has been widely applied to many robotic applications. RRT tends to expand mostly in unexplored areas, and although the number of edges its generated final path contains is minimal, the tree always stays connected. Various improvements of the algorithm have been published throughout the years. The most well-known one being RRT*, which introduces two new concepts: near neighbor search and rewiring tree operations [12]. The near neighbor operation finds the nearest neighbour for a randomly generated configuration in an area within a given radius. Then the tree is rewired in that area so that all nodes in that area are connected by the shortest paths [13]. The RRT* algorithm can achieve an optimal solution if enough iterations are performed, and the generated path is smoother than the one generated by

979-8-3503-5098-2/24/$31.00 ©2024 IEEE

RRT. Another improved version of RRT uses two trees that get extended with every iteration, and then using the starting point and the goal point as focus, and the trajectory length as the long axis, an ellipse interval sampling is constructed [14]. The algorithm aims to shorten the convergence time of both RRT and RRT*. An interesting issue of path planning strategies that RRT also has is the sharp change in direction, and a solution has been published that smooths the path generated [15]. However, how to use RRT-based navigation solutions properly in a MAS remains an open question.

Inspired by the aforementioned research advancements and limitations, this paper plans to develop a novel coordination framework via path planning and consensus approach. The MAS is firstly divided into leaders and followers. The leader's path is generated using the RRT* approach, which aims to obtain a safe and efficient path in the obstacle environment. Then, followers connected via a communication network are steered to keep a flexible formation using local information. To avoid potential collision among the agents, the artificial potential field (APF) approach is combined with the consensus protocol. To the best observation of the authors, such a unified framework that integrates path planning, consensus, and collision avoidance, has not been developed in the literature.

II. Preliminaries

A. Motion Planning

The motion planning issue is a computationally demanding task that comprises of determining a list of feasible configurations that enable a robot to travel from a start to a goal position while avoiding collision with all obstacles. This planning issue can be formulated as proposed by Karaman *et al.* [16] [12]. The dynamics of a robot can be expressed as $\dot{x}(t) = f(x(t), u(t))$, where $x(t) \in \mathcal{X}$ and $u(t) \in \mathcal{U}$, with $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{U} \subset \mathbb{R}^m$ represent the robot's position and the designed control input, respectively. The obstacle region is denoted as $\mathcal{X}_{obs}$, the obstacle-free space as $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$, the initial state as $x_{init} \subset \mathcal{X}$, and the goal state as $x_{goal} \subset \mathcal{X}$. The motion planning issue consists of finding a designed effort $u : [0, \mathcal{T}] \rightarrow \mathcal{U}$ that produces a reliable path $x(t) \in \mathcal{X}_{free}$ for $t \in [0, \mathcal{T}]$ starting from an initial position $x(0) = x_{init}$ to the goal position $x(T) \in \mathcal{X}_{goal}$. To obtain an optimal path, the obtained optimal waypoints should minimize a given cost, $c(x)$, where each acceptable trajectory $x : [0, \mathcal{T}] \rightarrow \mathcal{X}$ is mapped to a positive value.

B. Graph Theory

To describe the communication of the robots, graph theory can be used [17]. A graph is a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_1, ..., v_n\}$ is the set of vertices, $\mathcal{E}$ is the set of edges or arcs, and n is the total number of vertices in the graph. A vertex or a node represents a robot, and the edges or arcs represent connections between them. $\mathcal{E}$ contains all edges between nodes, an edge $e_i = (v_i, v_j)$ represents an edge from v_i to v_j and it is usually represented as an arrow with a tail at v_i and the head at v_j, it also means that v_j can receive information from v_i. Two nodes v_i and v_j are considered networked if there is a direct path from one node to another, and the shortest path length between v_i and v_j is defined as the distance between them. If all nodes v_i and v_j in graph $\mathcal{G}$ are connected, then $\mathcal{G}$ is defined as strongly connected. In the case of undirected graphs, a direct path from v_i to v_j indicates the existence of a direct path from v_i to v_j. An adjacency matrix $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$ is a square matrix of size n, and its elements are defined as

$$a_{ij} = \begin{cases} 1 & \forall (v_i, v_j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

C. Control Protocol Design

As mentioned above, the communication of a group of robots can be represented by a graph, however, in order to explain their dynamics, each node i of the graph representing a robot needs to correspond to a dynamic state vector $x_i(t)$ [18]. The dynamics of each node or robot can be expressed as

$$\dot{x_i} = f_i(x_i, u_i) \tag{2}$$

where u_i is the control input and $f_i(\cdot)$ is a flow function [17]. Control protocols involve designing distributed algorithms that coordinate the actions of individuals so that a common goal is reached by the system. And they are based on the ability of each node in a graph to communicate with another and obtain information. The performance of a control protocol can be affected by the communication graph topology as each agent's control law relies solely on data that is received from its local neighbors. A common objective of cooperative controller design is consensus, which means driving all the nodes to the same value, also known as a consensus value.

III. Main Results

The MAS considered in this study consists of a group of $\mathcal{N}$ agents. To aid the group of agents in navigating, a leader is introduced, resulting in a two-layer system. The communication network of the agents is considered to be connected and undirected. Each agent in the group has the capability to share information with neighboring agents within a specific radius. The group must navigate through an environment that has obstacles, and each agent is responsible for avoiding them. Additionally, agents must avoid colliding with one another while traveling through this environment. Agents are represented as objects with a center and a radius of 0, while objects are represented as objects with a center and a positive radius. This method integrates two primary algorithms, which are explained as follows. Firstly, the RRT* strategy is used to produce the path of the leader. Secondly, a control protocol based on consensus and collision avoidance is described. The section concludes with the explanation of the two-layer dynamical system consisting of the leader and the followers.

A. Motion Planning of the Leader

The following part gives a detailed explanation of the RRT* algorithm, which forms a key component in finding the path of the leader.

The RRT* algorithm is a path finding algorithm, it works by creating and maintaining a tree structure, $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, that consists of a set of nodes $\mathcal{V}$ which represent states belonging to $\mathcal{X}_{free}$, and directed edges $\mathcal{E}$ that connect them $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. To understand how the RRT* algorithms solves the optimal path planning issue, we first need to explain a set of the procedures the algorithm uses [19].

- $Sample$: Randomly samples a state $z_{rand} \in \mathcal{X}_{free}$ from the obstacle-free region.
- $Distance$: Returns the optimal trajectory cost between two states assuming there are no obstacles. The Euclidean distance is used for the calculation of the cost.
- $Nearest$: Returns the closest node in the tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ to a given state $z_{rand} \in \mathcal{X}$ based on the $Distance$ function.
- $Near$: Returns the vertices in $\mathcal{V}$ that are within the confines of a ball of volume $\gamma((\log n)/n)^d$ around $z_{rand} \in \mathcal{Z}$, where γ is a constant value, and n is given as input to the function.
- $ObstacleFree$: Checks if a path $x : [0, \mathcal{T}]$ is in the obstacle-free region X_{free} for all $t = 0$ to $t = \mathcal{T}$.
- $Steer$: Solves for a control input $u : [0, \mathcal{T}]$ that drives the system from $x(0) = z_{rand}$ to $x(\mathcal{T}) = z_{nearest}$ along the path $x : [0, \mathcal{T}] \to \mathcal{X}$.
- $InsertNode$: Adds a new node z_{new} to $\mathcal{V}$ in the tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ and creates an edge to an existing tree node $z_{current}$ as the parent of z_{new}. It also assigns a cost to z_{new} which is the sum of its parent's cost and the cost of the trajectory belongs to the new edge.

The RRT* algorithm which is outlined in Alg. 1 begins from an empty tree and creates a single node for the initialization. Afterward, it proceeds to build and improve the tree over $\mathcal{N}$ iterations. During each iteration, the algorithm samples a random state z_{rand} from the obstacle-free space $\mathcal{X}_{free}$, and finds a trajectory x_{new} that extends the nearest node in the tree $z_{nearest}$ towards the sample. If the trajectory does not cross the obstacle region $\mathcal{X}_{obs}$, the RRT* evaluates the cost regarding arriving potential parent nodes in the neighborhood of z_{new}. The cost is calculated as the sum of the cost required to reach the potential parent node and the cost of the trajectory to z_{new}. The node with the lowest cost is chosen as the parent for the new node that is being included in the tree. The ReWire procedure is then applied to nearby nodes z_{near} to determine whether it is possible to reduce the total cost by making z_{new} their parent. If this is the case, the algorithm rewires the tree accordingly and proceeds with the next iteration.

When given the start coordinate and goal coordinate of the leader, the RRT* algorithm generates a sequence of x and y coordinates that can be connected to form the final path. The leader will then use the consensus protocol described in the following sections to reach each of these coordinates one by one.

Algorithm 1 $\mathcal{T} = (\mathcal{V}, \mathcal{E}) \leftarrow \text{RRT}^*(z_{init})$

$\mathcal{T} \leftarrow InitializeTree()$;
$\mathcal{T} \leftarrow InsertNode(\emptyset, z_{init}, \mathcal{T})$;
for $i = 1$ *to* $i = N$ **do**
  $z_{rand} \leftarrow Sample(i)$;
  $z_{nearest} \leftarrow Nearest(\mathcal{T}, z_{rand})$;
  $(x_{new}, u_{new}, \mathcal{T}_{new}) \leftarrow Steer(z_{nearest}, z_{rand})$;
  if $ObstacleFree(x_{new})$ **then**
    $\mathcal{Z}_{near} \leftarrow Near(\mathcal{T}, z_{new}, |\mathcal{V}|)$;
    $z_{min} \leftarrow ChooseParent(\mathcal{Z}_{near}, z_{nearest}, z_{new}, x_{new})$;
    $\mathcal{T} \leftarrow InsertNode(z_{min}, z_{new}, \mathcal{T})$;
    $\mathcal{T} \leftarrow ReWire(\mathcal{T}, \mathcal{Z}_{near}, z_{min}, z_{new})$;
  end if
end for
return $\mathcal{T}$

B. Consensus of the Followers

Consensus protocols achieve the rendezvous of all robots to a desired state. For an efficient protocol, collision avoidance must also be integrated. The following subsections provide detailed explanations of both the convergence process and the integration of collision avoidance techniques.

1) Convergence: The dynamics of an agent that is part of a group moving in a 2-dimensional space can be represented by

$$\dot{x_i}(t) = u_{a,i}(t) \tag{3}$$

where the rate of change of the agent's position with respect to time $\dot{x_i}(t)$ is determined by a control input $u_{a,i}(t)$, and is studied in the context of continuous-time models.

In a fixed topology network $\mathcal{G}$ with an adjacency matrix $\mathcal{A}$ representing a MAS, a consensus protocol can be defined as follows

$$u_{a,i} = kC(s)\sum_{j=1}^{N} a_{ij}(x_i - x_j) + a_{i0}(x_i - x_0) \tag{4}$$

where $i \in \{1, 2, \ldots, N\}$, $C(s)$ is an Strict Negative Imaginary (SNI) transfer function with $C(0) > 0$, and k is a positive control gain, and x_0 is the position of the leader.

Motivated by [20], we then have the following theorem to ensure that the consensus of the followers can be achieved.

Theorem 1: Let N mobile robots be networked via the graph $\mathcal{G}$. $C(s)$ is an SNI transfer function with $C(0) > 0$. Thus, there is a bounded $k^\star > 0$ so that for any $k \in (0, k^\star]$, the mobile robots reach consensus via the cooperative protocol (4).

Proof: We use the notations $\mathbf{E}(s)$, $\mathbf{X}(s)$, $\mathbf{R}(s)$ to describe the Laplace transform of the vector signals $e(t) = [x_0 - x_1(t), x_0 - x_2(t), \cdots, x_0 - x_N(t)]^\top = [e_1(t), e_2(t), \cdots, e_N(t)]^\top$, $x(t) = [x_1(t), x_2(t), \cdots, x_N(t)]^\top$ and $r(t) = \mathbf{1}_N x_0$. Let the consensus position error be

$e(t) \triangleq r(t) - x(t)$. We can easily obtain the equation below by considering the property of $C(s)$

$$\mathbf{E}(s) = \left[I + (\mathcal{L} \otimes \frac{k}{s} C(s))\right]^{-1} \mathbf{R}(s). \tag{5}$$

Following Lemma 3 in [20], the distributed consensus strategy is asymptotically stable when the SNI controller is applied for all $k \in (0, \infty)$. The steady-state consensus error is then given by

$$\begin{aligned} e_{ss} &= \lim_{t\to\infty} e(t) = \lim_{s\to 0} s\mathbf{E}(s) \qquad (6a)\\ &= \lim_{s\to 0} s\left[I_N + \left(\mathcal{L} \otimes \frac{k}{s} C(s)\right)\right]^{-1} \mathbf{R}(s) \\ &= \lim_{s\to 0} s\left[sI_N + (\mathcal{L} \otimes kC(s))\right]^{-1} (s\mathbf{R}(s)) \\ &= \left[\mathcal{L} \otimes (kC(0))\right]^{-1} \left(\lim_{s\to 0} sI_N\right) \left(\lim_{s\to 0} s\mathbf{R}(s)\right) \\ &= [0\ 0\ \cdots\ 0]^\top \qquad (6b) \end{aligned}$$

since $C(0) > 0$, $\mathcal{L} > 0$ and $\lim_{t\to\infty} r(t) = \mathbf{1}_N x_0 = r_{ss}$, where r_{ss} is the steady-state value of the leader's position. The asymptotic stability property of the position reaching error, i.e., $\lim_{t\to\infty} e(t) = 0$ implies $x(t) \to \mathbf{1}_N r_{ss}$ as $t \to \infty$. This completes the proof. ■

The interconnection topology between the robots changes over time however. The changes occur in non-overlapping, contiguous time intervals $[t_i, t_{i+1})$, $i = 0, 1, ...,$ and there are infinite such intervals starting at time $t_0 = 0$. The neighbour set for each robot and the Laplacian associated with the switching interconnection graph are time-varying, but remain constant over each time interval $[t_i, t_{i+1})$. Therefore, a dynamic graph $\mathcal{G}_{s(t)}$ should be considered where $s(t) : \mathbb{R} \to \mathcal{J}$ and $\mathcal{J}$ is a set of indices $\mathcal{J} = \{1, 2, ..., m\}$.

2) Collision Avoidance: While traveling in a space containing obstacles, if a collision avoidance protocol is not used, robots might collide with these obstacles. Furthermore, robots might collide with each other as well. Considering the set of N robots where $x_i \in \mathbb{R}^2, i \in \{1, 2, ..., N\}$ represents the position of the i^{th} robot, and the set of M obstacles where $w_k \in \mathbb{R}^2, k \in \{1, 2, ..., M\}$ represents the position of the k^{th} obstacle, then the APF approach can be integrated with the control law to achieve collision avoidance.

The collision avoidance between robots $u_{b,i}$ which is expressed as

$$u_{b,i} = \sum_{j=1}^{N} \mu_b b_{ij} a_{ij} \left(\frac{1}{\|x_i - x_j\|} - \frac{1}{d_b}\right) \frac{x_i - x_j}{\|x_i - x_j\|^2}, \tag{7}$$

and the collision avoidance between robots and obstacles $u_{c,i}$ which is expressed as

$$u_{c,i} = \sum_{k=1}^{M} \mu_c c_{ik} \left(\frac{1}{\|x_i - w_k\|} - \frac{1}{d_c}\right) \frac{x_i - x_k}{\|x_i - x_k\|^2}. \tag{8}$$

Consider d_b to represent the minimum distance of repulsive-region between any two robots so that they don't collide, and d_c to represent the minimum distance of repulsive-region between any robot and obstacle, the distance to be calculated from the center of each object. The value of distance d_c needs to be larger than the minimum distance between any two obstacles. This minimum distance is measured from the centers of the two obstacles while accounting for their radii being subtracted. Given the set of N robots and the set of M obstacles, $\forall i, j \in \{1, 2, ..., N\}$ and $\forall k \in \{1, 2, ..., M\}$, b_{ij} and c_{ik} can be defined as follows

$$b_{ij} = \begin{cases} 1\ \forall \|x_i - x_j\| \le d_b, \\ 0 \text{ otherwise}, \end{cases} \tag{9}$$

and

$$c_{ik} = \begin{cases} 1\ \forall \|x_i - w_k\| \le d_c, \\ 0 \text{ otherwise}. \end{cases} \tag{10}$$

Finally, both μ_b and μ_c represent constant gains, and by choosing $\mu_b, \mu_c > 0$, the protocol is guaranteed to drive all robots without any collisions [21].

C. The Combined Two-Layer Multi-Agent Systems

The MAS considered consists of two layers: the leader, and the followers. The followers compose the multi-agent system, and the leader aids the group in guiding them to the final target location. The leader finds an almost optimal path by using the RRT* algorithm, and also doesn't have any communication with any of the followers. The followers however, can communicate between them and also can communicate with the leader as long as they are withing a certain radius ρ from the given follower or the leader. The motion of each agent with the combined input can be given by

$$u_i = u_{a,i} + u_{b,i} + u_{c,i}. \tag{11}$$

Although all robots and the leader follow the control low described above, subtle variations exist in the law applicable to each of them, as elaborated in the subsequent sections. It is worth noting that the only aspect that remains consistent for all entities is the approach to obstacle collision avoidance.

For the leader, the control law reduces only to consensus u_a and collision avoidance with obstacles u_c as the leader is considered to have a virtual presence only, and therefore it has no physical form, so there is no need to check whether it collides with other robots. For the consensus law, instead of considering all robots within a radius ρ, the leader uses only its target location which is part of the list of coordinates along the final path generated by applying the RRT* algorithm. When the first target coordinates from the list are reached by the leader, the target location changes to the coordinates of the next point in the list. To ensure that the leader moves at a reasonable speed and that the followers are able to follow the leader and not remain behind, a new constant γ is introduced, where $\gamma \in \mathbb{R}$ is within the following range $(0, 1)$. Therefore the control law used by the leader is defined as

$$u_0 = \gamma(u_{a,0} + u_{c,0}) \tag{12}$$

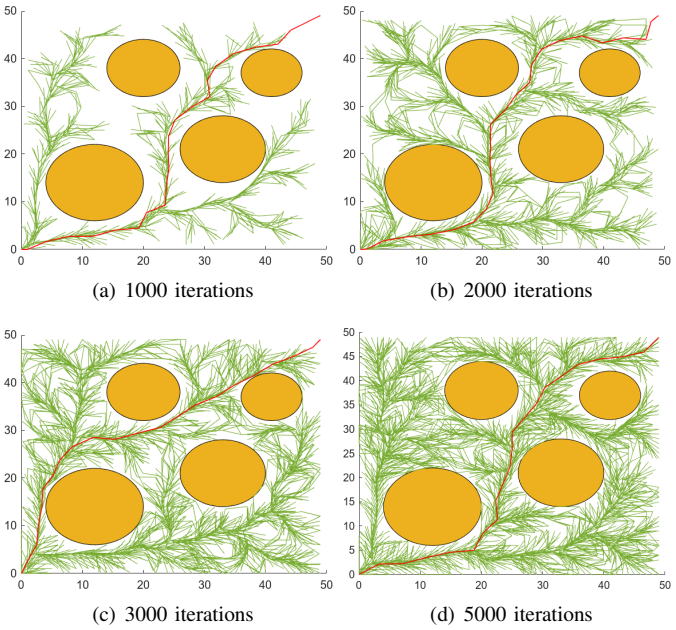
In summary, the proposed method for the MAS involves a leader and followers. The leader uses the RRT* approach to obtain an optimal path without communication, while the followers communicate with each other and the leader within a radius. Each agent's motion is governed by consensus for coordination, collision avoidance between robots, and collision avoidance between robots and obstacles. The leader follows a list of target coordinates, while the followers use the general control law with neighbors within a radius. The method combines path planning, consensus, and collision avoidance to guide the followers while ensuring safety and efficiency in navigation.

IV. Evaluation

To be able to put the new method in practice, we have created an environment where motion of the MAS will be simulated. The environment is a 2-dimensional space with x and y coordinate limits from 0 to 50. The space contains 4 obstacles of various sizes that were placed in such way to show the capacity of the approach to avoid these obstacles and still reach the target location in a reasonable time.

In Fig. 1, the paths generated by the RRT* algorithm can be observed. The most important parameters when using the RRT* algorithm are ϵ, the number of nodes n, and the radius. Epsilon represents the step size or the maximum distance that a new node can extend from its parent node in the tree. It determines the granularity of the search space exploration. A smaller epsilon value leads to a finer exploration, potentially resulting in a more precise and detailed path, therefore we have decided to use $\epsilon = 1$. The number of nodes refers to the total count of nodes in the RRT* tree. Each node represents a configuration or state in the search space. A higher number of nodes generally implies a more extensive exploration of the space, which can lead to a better estimation of the optimal waypoints. Various values for the number of nodes have been tested and the results can be seen in Fig. 1. From these figures, the most optimal path was generated when $n = 5000$. Finally, the radius parameter defines the region around a new node within which other existing nodes are considered for potential connection or rewiring. It determines the extent of the local search for finding nearby nodes. A larger radius allows for greater flexibility in connecting nodes, facilitating the discovery of alternative paths and potentially improving the quality of the final path. We considered the most suitable value for the radius to be 4. The paths in red in Fig. 1 represent the paths generated by the RRT* method, and the path in sub figure (d) will be used for testing the method.

As already mentioned in the previous paragraph, the leader will follow the path calculated by the RRT* strategy in Fig. 1 (d). For this simulation a MAS consisting of 4 agents will be used. The initial positions of the agents are randomly generated within a range $(0, 8)$ for both x and y. The agents can exchange information with other agents within a radius $\rho = 6$, and the distance to be maintained between the robots and between the robots and obstacles is $d = 2$. For the



(a) 1000 iterations (b) 2000 iterations

(c) 3000 iterations (d) 5000 iterations

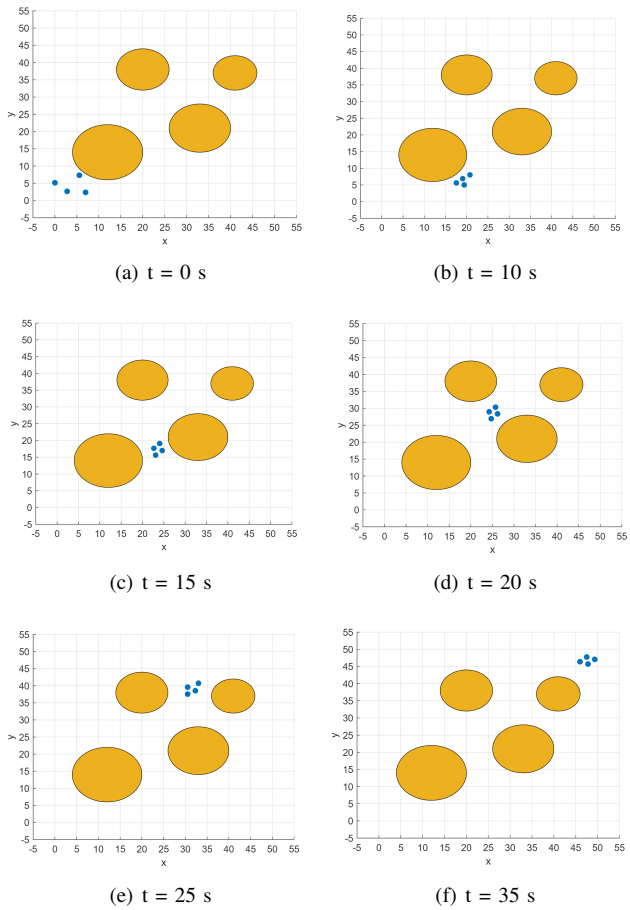
Fig. 1. The path from coordinate (0,0) to coordinate (49,49) generated by the RRT* algorithm using (a) 1000 iterations, (b) 2000 iterations, (c) 3000 iterations, and (d) 5000 iterations.

motion of the leader, $\gamma = 0.2$ has been used. For the collision avoidance algorithm both μ_b and μ_c are equal to 80. The results of applying the method can be seen in Fig. 2. The agents manage to travel from the starting location to the target location, while also maintaining a constant distance between them and avoiding any obstacles. The algorithm takes 35 s to achieve this. Depending on the size of the MAS, the time taken will vary. As well, depending on the settings of the robotic platforms and the working environment, the parameters of values such as ρ, the distance to be maintained between agents and obstacles, and γ should be adjusted to further improve the performance.

Fig. 3 illustrated the consensus error of the trajectory of all agents. The tracking error represents the difference between the target location and the actual trajectory followed by the agents. It is easy to observe that all agents converge to a common value while maintaining a distance between them. This demonstrates that the system successfully adapts and corrects its motion to minimize deviations, resulting in accurate tracking and convergence to the desired target location.

V. Conclusions

In this work, we proposed a unified approach for MAS to navigate through environments with obstacles. The integration of a leader in the group of agents resulted in a two-layer system, therefore improving the navigation capabilities of the agents. To generate the path for the leader, the RRT* algorithm was used, which provided an almost optimal trajectory. The leader's path served as a reference for the followers, helping coordination and consensus among the agents. The commu-



(a) t = 0 s (b) t = 10 s

(c) t = 15 s (d) t = 20 s

(e) t = 25 s (f) t = 35 s

Fig. 2. A MAS comprised of 4 robots traveling from coordinate (0,0) to coordinate (49,49), showing the progress of the system at various points in time.

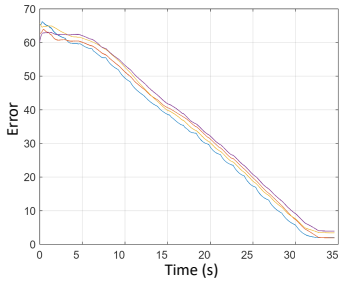


Fig. 3. Tracking error (distance between agent location and target location over time).

nication network among the agents was considered connected and undirected, enabling information sharing within a specific radius. Through simulations, the feasibility of the developed approach in guiding the MAS through complex environments was demonstrated.

ACKNOWLEDGEMENT

This work was supported by EU H2020-FET-OPEN RoboRoyale project [grant number 964492].

REFERENCES

[1] Z. Qiang, L. Dai, B. Chen, and Y. Xia, "Distributed model predictive control for heterogeneous vehicle platoon with inter-vehicular spacing constraints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 3, pp. 3339–3351, 2022.

[2] S. Xie, J. Hu, Z. Ding, and F. Arvin, "Cooperative adaptive cruise control for connected autonomous vehicles using spring damping energy model," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 2974–2987, 2023.

[3] F. Rekabi-Bana, J. Hu, T. Krajník, and F. Arvin, "Unified robust path planning and optimal trajectory generation for efficient 3D area coverage of quadrotor UAVs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 3, pp. 2492–2507, 2024.

[4] E. J. Rodríguez-Seda, X. Xu, J. M. Olm, A. Dòria-Cerezo, and Y. Diaz-Mercado, "Self-triggered coverage control for mobile sensors," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 223–238, 2022.

[5] K. Champagnie, F. Arvin, and J. Hu, "Decentralized multi-agent coverage path planning with greedy entropy maximization," in *2024 IEEE International Conference on Industrial Technology*, 2024, pp. 1–6.

[6] K. Wu, J. Hu, Z. Li, Z. Ding, and F. Arvin, "Distributed collision-free bearing coordination of multi-uav systems with actuator faults and time delays," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[7] F. Rekabi-Bana, M. Stefanec, J. Ulrich *et al.*, "Mechatronic design for multi robots-insect swarms interactions," in *2023 IEEE International Conference on Mechatronics*, 2023, pp. 1–6.

[8] I. Couzin, J. Krause, N. Franks, and S. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, pp. 513–6, 03 2005.

[9] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[10] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 863–868, 2007.

[11] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotic Research*, vol. 30, pp. 846–894, 06 2011.

[13] I. Noreen, A. Khan, and Z. Habib, "A comparison of RRT, RRT* and RRT*-smart path planning algorithms," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 10, p. 20, 2016.

[14] J. Chen and J. Yu, "An improved path planning algorithm for UAV based on RRT," in *2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering*, 2021, pp. 895–898.

[15] K. R. Jayasree, P. R. Jayasree, and A. Vivek, "Smoothed rrt techniques for trajectory planning," in *2017 International Conference on Technological Advancements in Power and Energy*, 2017, pp. 1–8.

[16] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1478–1483.

[17] F. Lewis, H. Zhang, K. Movric, and A. Das, *Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches*, 2014.

[18] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[19] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 7681–7687.

[20] J. Hu, B. Lennox, and F. Arvin, "Robust formation control for networked robotic systems using negative imaginary dynamics," *Automatica*, vol. 140, p. 110235, 2022.

[21] H. Tnunay, Z. Li, C. Wang, and Z. Ding, "Distributed collision-free coverage control of mobile robots with consensus-based approach," in *IEEE International Conference on Control & Automation*, 2017, pp. 678–683.