Partial Swarm SLAM for Intelligent Navigation

Jawad Yasin, Huma Mahboob, Suvi Jokinen, Mohammadhashem Haghbayan, Muhammad Mehboob Yasin and Juha Plosila

June 4, 2022

Partial Swarm SLAM for Intelligent Navigation*

Jawad N. Yasin[1,2][0000-0002-2663-9019], Huma Mahboob[1][0000-0003-4507-1403], Suvi Jokinen[1][0000-0002-2806-8619], Hashem Haghbayan[1][0000-0001-6583-4418], Muhammad Mehboob Yasin[3][0000-0003-0013-743X], and Juha Plosila[1][0000-0003-4018-5495]

[1] Autonomous Systems Laboratory, Department of Future Technologies, University of Turku, Vesilinnantie 5, 20500 Turku, Finland
{janaya, ssjoki, mohhag, juplos}@utu.fi, m.huma.mahboob@gmail.com
[2] ABB Oy - Finland
[3] Department of Computer Networks, College of Computer Sciences & Information Technology, King Faisal University, Hofuf, Saudi Arabia
mmyasin@kfu.edu.sa

Abstract. The focus of this work is to present a novel methodology utilizing the classical SLAM technique and integrating with the swarm agents for localizing, guiding, and retrieving the agents towards the optimal path while using only necessary tracker-based information between the agents. While navigating in an unknown environment with no-prior map information, upon encountering large obstacles (out of the field of view detection range of the onboard sensors, the swarm is divided into sub-swarms. This is done while dropping tracking points at every turn. Similarly, the time stamps for every turn taken and the gap width available between obstacles are recorded. Once an agent from any sub-swarm category reaches the destination, the agent broadcasts these tracker points to the rest of the swarm agents. Utilizing this broadcasted key information, the rest of the agents are able to navigate toward the destination without having to find the path. With the help of simulation examples, it is shown that the proposed technique is efficient over other similar randomized turn-based techniques.

Keywords: Swarm robotics · Distributed systems · Exploration schemes · SLAM

1 Introduction

Swarm robotics can be defined as the study of how a large group of agents or robots can be controlled in a such a way to achieve an overall desired behavior or shape to perform a set of tasks. This overall emergence of the behavior is due to the interactions of the agents with other agents within the swarm as well as the objects in the environment [1]. A swarm of robots can be utilized for

* This work has been supported by the Academy of Finland-funded research projects (AURORA: 330493, ADAFI: 335512).

a wide range of tasks ranging from search and rescue to mapping to military purposes [2,3]. That is due to the ability of the agents within the swarm to self localize, self-organize, communicate with other agents, as well as the flexibility and scalibility of the overall swarm making the utilization of swarm of robots ideal for such unknown environments [4]. Similarly, for a swarm to navigate autonomously, in any environment introduces several research challenges, such as keeping or maintaining the formation, collision avoidance, localizing, inter-agent communication, path finding [5]. Among other approaches for localizing, the agents in the swarm can utilize simultaneous localization and mapping (SLAM) to autonomously self localize and navigate in unknown environments with no prior map information [6]. SLAM is a known and fundamental technique in the navigation of autonomous robots. However, the focus of the studies and development in SLAM has been mostly from the perspective of individual robots and which leaves a gap for the development of SLAM with multiple robots or a swarm as a whole [7].

The existing multi-robot SLAM techniques focus on either collective production of maps or centralized map merging (due to limited onboard computational resources). Recently, an approach was introduced where the robots produce individual maps by utilizing different exploration methodologies and these maps were later integrated on a remote platform [8,9]. Moreover, SLAM, from computational perspective, is intensive and similarly for transferring the data between robots will also require large amount of data to be transferred and processed [10]. However, none of the existing methodologies address the issue of utilizing the SLAM technique to facilitate the agents of the swarm to collaborate by exchanging minimum information required to direct the other agents towards a common goal. In order to develop an effective and efficient SLAM technique for swarm, there are several questions that need to be addressed, ranging from the inter-agent communication of the swarm to exploration of the environment to utilization of the acquired data and sharing of the necessary information between the agents.

In this article, we propose an algorithm in which only the necessary information is shared between the agents, i.e., reducing the overhead for communication, and subsequently is computationally light for the agents. Agents note the coordinates of the position where they disperse and the main swarm is divided into sub-swarms. This point of separation is noted by respective agent as it navigates in a different direction. Upon approaching an available gap between the obstacles, the agent note the coordinates of this point as well, the time it took to travel from the first point, the velocity with which the agent navigated, and the shape/width of the gap found between the obstacles. This process is continued until one of the agents of the sub-swarm finds a clear route to the destination. At this point, that agent broadcasts the recorded information to rest of the sub-swarms for them to follow the tracker points and reach the destination.

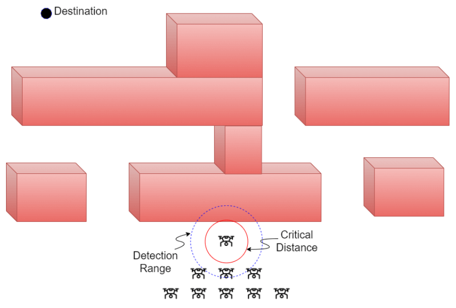
The novelty of the proposed algorithm is as follows:

1. while navigating towards the destination, upon encountering obstacle(s) large enough to have their edges out of bounds from the sensor's range, the agents

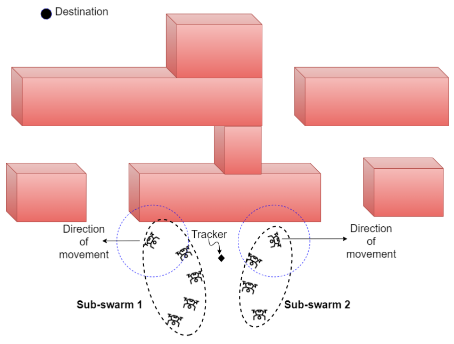
are dispersed in different directions and individual agents keep a record of their movement

2. only the necessary information is shared between the agents rather than sharing the whole acquired maps with other agents, i.e., reducing the overhead of the communication between the agents
3. when the agents disperse to find a route for reaching the destination, an agent upon finding an opening towards the destination only keeps a track of certain features and shares only that information with other agents for them to perform targeted pathfinding and navigate towards the destination in an efficient manner

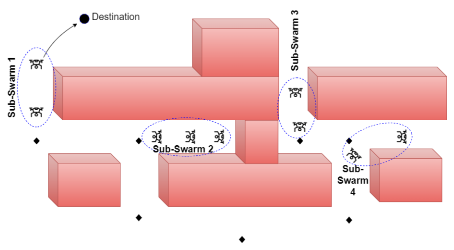
The rest of the paper is organized as follows. Motivation is provided in Section 2. Section 3 describes the proposed approach. Simulation results are provided in Section 4. Finally, the concluding remarks, discussion and some future work in given in Section 5.



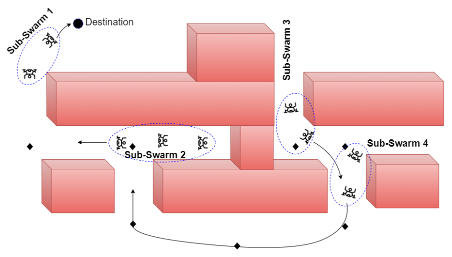
(a) Initial Setup, Obstacle in range



(b) Swarm divided into sub-swarms



(c) Swarm further divided into sub-swarms. Sub-swarm 1 finds path to destination



(d) All sub-swarms start tracking back utilizing the Tracker points

Fig. 1: Illustration of the Partial Swarm SLAM technique. (a) shows the swarm approaching a large obstacle with edges not visible in the ranging sensor's range. (b) Swarm gets divided into sub-swarms, and sub-swarm 1 and 2, start navigating in different directions to find the route. (c) All the sub-swarms have place pheromone trackers (coordinates) while navigating through the maze of unknown environment. (d) Sub-swarm 1, upon finding path to destination, broadcasts its placed pheromone trackers to the rest of the swarm agents.

2 Motivation

While navigating in an unknown environment, with no prior map information, autonomous agents have to perform avoidance maneuvers by analyzing the information at hand, i.e., by utilizing the onboard sensors for observing the surroundings, evaluating the situation, and choosing the continuation trajectory as necessary [11]. However, in situations, where the encountered obstacle(s) are large and while utilizing the onboard ranging sensors, the agents cannot detect either edge of the obstacle, arguably the best course of action is to make a calculated guess (by keeping the direction of the destination under consideration) and turn or deviate accordingly [cite our paper]. In such a scenario, since the final destination is known, the agent draws a tangent from its own coordinates towards the destination and chooses the direction to turn accordingly. In such a manner, based on the information at hand, without any knowledge of the map, the agent takes the best/optimal decision. However, such an approach can also lead to a much larger deviation leading to longer mission time, battery drainage, or even local minima.

In order to tackle such a situation, we propose a new technique of agent dispersion, inspired by the ant pheromone technique, where the ants leave pheromones to direct and guide other ants in the group to follow the path to take [12]. It is achieved by dividing the swarm in half, in either direction for routing finding purposes. They keep a track of the markers and turns they take accordingly. Every time, a similar situation is encountered, the sub-swarm gets divided further and starts exploring. Once, an agent finds the route to the goal/destination, it broadcasts the required tracker-based information to the rest of the swarm. Based on this information, the agents start backtracking to where they chose a different path from that specific agent and simply follow the trackers provided to them to reach their goal for mission completion, as shown in Figure 1.

3 Proposed Approach

For simulating the agents, the kinematics model of a differential drive robot is used. The differential drive robot works on the principle of the difference between the velocity of the left and the right wheels. This difference determines the heading of the robot. Kinematic model of a differential drive robot:

$$\begin{aligned} \dot{x} &= vcos\theta, \\ \dot{y} &= vsin\theta, \\ \dot{\theta} &= \frac{v_{\Delta}}{W} \end{aligned} \tag{1}$$

where $\dot{x}$ and $\dot{y}$ are the x and y positions of the robot, v is the velocity, $\dot{\theta}$ is the heading angle of the robot.

To calculate the turning curve of the robot, the following equation is used:

$$\Delta V = \frac{v_r - v_l}{W} \quad (2)$$

where ΔV is the difference between the left and the right wheel speed, v_r and v_l are the right and the left speeds respectively, and W is the width of the robot.

Algorithm 1 Global Routine

```
procedure NAVIGATION & DETECTION
    $B_{Formation} \leftarrow False$;
    $Destination \leftarrow False$;
    if $Self.ID == 1$ then
        $\alpha \leftarrow Self$;
        $\alpha_{Alive} \leftarrow False$;
    else
        $\alpha \leftarrow Leader(Self)$;
        $\alpha_{Alive} \leftarrow True$;
    end if
    while True do
        $D$ = Scan();
        if $D <$ ReactionRange then
            $D_o$, $A_o$ ← Calculate obstacle distance and angles at which the edges lie;
            if $D_o < \gamma$ && $A_o < \Gamma$ then
                Collision Avoidance($D_o$, $A_o$);
                if $Destination == L.o.S.$ then
                    $Self.Destination \leftarrow True$
                    Broadcast Tracker points;
                end if
            end if
            if $B_{Formation}$ && $Destination$ then
                Traceback($PoB, t, G_w, T_D$);
            else
                Navigate();
            end if
        end if
    end while
end procedure
```

The top-level pseudo code of the agents is given Algorithm 1. In the initial setup the agents are assigned their respective IDs. Every agent in the swarm executes this top-level algorithmic routine locally. In the beginning, the Boolean variables $B_{Formation}$ and $Destination$ are initialized, whose roles are to notify the global routine if the swarm has been divided into sub-swarms and if the agent has reached the destination respectively (Lines 2-3). Then the algorithm checks if the global leader has been declared and if the leader-follower connection has been set up. If not, then the global leader is declared (α) and the followers are connected to their respective and immediate leaders accordingly. As the global leader does not have any leader, therefore, $\alpha Alive$ (My leader is Alive) is set to False. And for the followers, this flag is set to True (Lines 4-9). After this, the agents start scanning their surroundings while navigating toward the destination (lines 11-12). As soon as an object is detected, it is checked if the distance to the object (D) is less than the defined *ReactionRange* (line 13). If the detected obstacle lies within the *ReactionRange*, the distance (D_o) along with the angles at which edge(s) of the detected obstacle(s) lie are calculated (Line 14). If the calculated distance lies within the defined deviation range (γ) and the angles at which the edge(s) have been detected also lie within the defined ranges, indicating that continuing the current trajectory will lead to a potential collision, the control is transferred to Collision Avoidance algorithm (Lines 15-16). Every agent checks if the destination is in its line of sight ($L.o.S.$), if this is true, the agent then sets the destination flag to *True* for itself and broadcasts the tracker points to the rest of the agents (Lines 17-19). The algorithm then checks if the swarm was divided into sub-swarms ($B_{Formation}$ Flag indicates the break of formation) during any phase of the mission or while performing avoidance maneuvers and if the agent has reached the destination (implying that the route to the destination is now complete). In this case, the Traceback() function is called to provide the pheromone tracker points to other agents to allow them to simply follow and navigate towards the destination (Line 22-23). Otherwise, the agents continue the navigation process until the goal is achieved.

3.1 Collision Avoidance

Algorithm 2 Collision Avoidance

```
    procedure CollisionAvoidance($D_o$, $A_o$)
2:      if $A_o$ != NULL then                                   ▷ Obstacle edge(s) detected
            $\beta$ = detect_edges($D_o$);
4:          if $\beta > 2$ then
                $\zeta \leftarrow$ calculated gap between obstacles;
6:              if $\zeta > R_c$ then
                    Align agent to pass through;
8:                  if $B_{Formation}$ then
                        $T_D$[][] = Turn direction and number of turn;
10:                     $G_w$ = Gap width;
                    end if
12:             else                                            ▷ out of bounds Obstacle
                    $B_{Formation}$ = True;
14:                 Break Formation();
                end if
16:         end if
        else                                                    ▷ No edges detected
18:         Break Formation();
            $B_{Formation}$ = True;
20:     end if
    end procedure
```

In the collision avoidance phase in our proposed algorithm, it is first checked if there were any edges detected of the detected obstacle(s). If A_o (angles at which the edges have been detected) has real values, then it means that the edges have been detected and the obstacle is not large enough to be out of bounds from the ranging sensor's view (Line 2). Then the number of edges are detected (β) in order to check how many obstacles have been detected that can cause a potential collision if the current trajectory is continued (Line 3). If the detected edges are more than 2, it indicates multiple obstacles detected, and then the algorithm calculates the gap (ζ) that is available between detected obstacles (Lines 4-5). After calculating the available gap between the detected obstacles, it is checked if the gap is sufficient enough for the agents to pass through, i.e., the gap bigger than R_c. R_c is calculated based on the dimensions of the agent plus a defined safe distance that is to be allowed on either side of the agent, Eq. 1:

$$R_c = \delta + \tau \tag{3}$$

where $R_[c$ is the collision radius, and δ, and τ are the width and the minimum safe distance allowed from either side of the agent respectively.

Then agent(s) is aligned to pass through the available gap (Lines 6-7). Further, it is checked if the swarm has already broken down the initial formation to create sub-swarms for route finding, then the direction of the turn the agent is taking and the width of the gap which the agent is navigating through are noted as part of pheromone tracker pointers (Lines 8-10). These pheromone tracker pointers are later used to direct other agents to find the route. Whereas, if the available gaps are not wide enough for the agent to pass through or if there were no edges detected (Line 2), then in both cases, it is treated as a single obstacle out of bounds case, the $B_{Formation}$ flag is set to *True*, and the control is transferred to BreakFormation() algorithm (Lines 12-19).

3.2 Formation Breaking and Path Finding Mode

Algorithm 3 Break Formation

```
    procedure BreakFormation
2:      $TangentLine$ = Calculate tangent to destination;
        $PoB$ = current coordinates;
4:      $t$ = time;
        $T_D[][]$ = turn directions and number of turns;
6:      $G_w$ = Gap width;
        $G_{1i}$, $G_{2j}$ = Create sub-swarms();
8:      $G_{1i}$,$G_{2j}$ $\leftarrow$ Temporary sub-swarm leaders;
        Short term path planning(TangentLine);
10: end procedure
```

Algorithm 3, Break Formation, starts by drawing a tangent line from the agents' coordinates to the destination for directional purposes (Line 2). Agents utilize onboard positioning systems (GPS) to determine the direction to the destination. Then the PoB (point of break) is noted, i.e., the current position (Line 3). The time stamps are noted starting from PoB onwards, a stamp for every turn taken (Line 4). This helps as a cross-check for tracing back the route for every agent, by allowing the agent to verify if at approximately the same amount of time it has arrived at the similar position. This is further cross-checked with the gap width (G_w) (Line 6), also noted in Algorithm 2. The agent then notes the direction in which it is turning and further notes the number of the turns it has taken (Line 5). Then the swarm is divided into sub-swarms, by pooling the agents alternatingly into G_{1i} and G_{2j} (Line 7). The sub-swarms are assigned their respective temporary or sub-swarm leaders for continuing the mission and leading the rest of the follower agents (Line 8). After this, short term path planning is done by the sub-swarm leader by utilizing the calculated $TangentLine$ to resume navigating in the direction of the destination after bypassing the current obstacle (Line 9).

3.3 Trace Back Utilizing Tracker Points

Algorithm 4 Trace Back

```
   procedure TRACEBACK($PoB, t, G_w, T_D$)
2:     if $Self.Destination$ != $True$ then
           Rcv(Tracker points(PoB));
4:         Set (speed);
           if $Tracker[i]$ then
6:             cross-check $t$, $G_w$, $T_D$;
               Navigate;
8:         end if
```

Algorithm 4 shows the pseudo-code for the *Traceback* function. the algorithm starts by checking if the agent performing the check has already reached the destination (Line 2). If the *Destination* flag is *False*, the agent receives the tracker points, and sets its speed accordingly by reading the speed of the broadcasting agent (Lines 3-4). Upon reaching a tracker point ($Traker[i]$), the agent cross-checks the provided data by checking the time it took for navigating between $Tracker[i]$ and $Tracker[i-1]$, the shape/width of the gap at the current position (G_w), and takes the respective turn to continue navigation towards the destination (Lines 5-7).

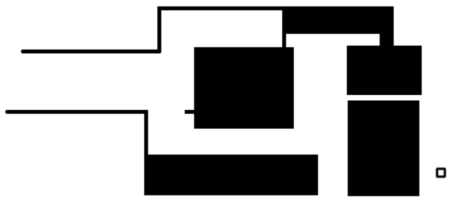
4 Simulation Results

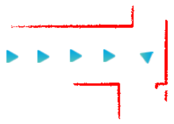
The initial conditions defined for our work are as follows:

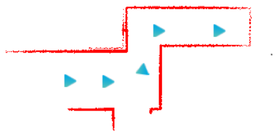
1. all agents obtain their position vectors utilizing their onboard localization techniques
2. the communication channel is lossless
3. the agents can communicate and broadcast their tracker-based information to other members of the swarm

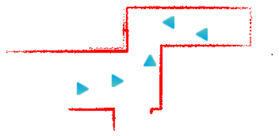
For the simulation environment, we used python graphics. For the ranging sensor, we simulated the output of the LiDAR sensor in a 2-dimensional scale. The scaling of the environment closer to the real-world movement is performed by converting the distance traveled and the speed with which the agents are navigating from meters and meters per second to pixels and pixels per second.

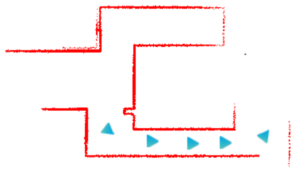
Figure 2(a) shows the floor map of the environment used for testing the proposed methodology. The destination mark is shown by the hollow square in


(a) The floor map of the environment used for verification


(b) Swarm comes across a large obstacle, i.e., out of bounds


(c) With no-prior map information, the swarm utilizes the priority-based turning and turns left


(d) Swarm is turning back after reaching a dead end, here the congestion is also faced


(e) Swarm keeps navigating in the other direction

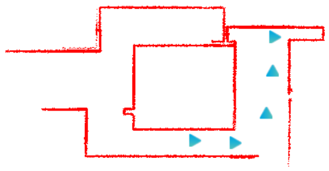
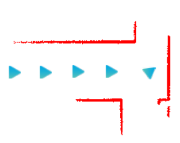
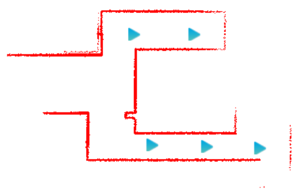
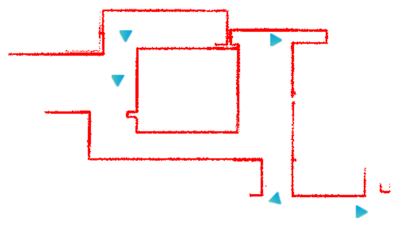

(f) Swarm once again faces the issue of reaching a dead end, rerouting the whole swarm, congestion and wastage of energy resources

Fig. 2: Simulation Results: Priority-based turn

the right corner. Figure 2(b) the agent encountering the large obstacle and taking a priority-based left turn. The rest of the swarm follows in a similar manner. As shown in Figure 2(c), the swarm reaches a dead end. In this situation, the agents have to turn back and go to the point from where the swarm took the left turn. This not only drains more energy resources but is also time taking and results in congestion (Figure 2(d)). After the swarm reroutes back to the initial position, from where it turned left, the agents navigate in the other direction, where the swarm faces a similar situation again, Figure 2(e). Figure 2(f) shows the swarm facing another dead end. Upon tracking back, the swarm will finally find the path to the destination, in the considered simulation setup.


(a) Scene 1, swarm starts navigating and encounters a larger obstacle


(b) Scene 2, swarm is divided into sub-swarms and simultaneously navigating the environment to find the path


(c) Scene 3, one agent finds the destination, and other agents in the sub-swarms have started navigating towards that point

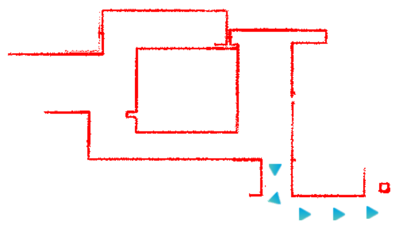

(d) Scene 4, all the agents of the sub-swarms have successfully reached the desired coordinates

Fig. 3: Simulation Results: Partial Swarm SLAM

The proposed technique is tested in the same simulation setup. As shown in Figure 3(a), the agents start navigating while exploring the unknown environment simultaneously. Upon encountering an obstacle, the swarm divides into two sub-swarms and keeps on navigating in the pursuit of finding the path to the destination, as shown in Figure 3(b). Figure 3(c), shows the swarm divided into further sub-swarms. Here, it can be seen that an agent of the sub-swarm found the unobstructed path to the destination. And the other agents in the sub-swarms have started navigating back by backtracking the tracker points. Figure 2(d) shows the final scene, where the agents (sub-swarms) have to regroup into the initial swarm setup and reached the destination.

As it is evident from the comparative simulation results, utilizing the proposed approach the time to mission completion can be significantly reduced in environments with no-prior map information available. In the considered setup, utilizing the priority-based turn technique, the longest distance any agent had to travel was approximately 1.5 times the longest distance any agent traveled utilizing the proposed Partial Swarm SLAM technique. The distances covered by individual agents are provided in Table 1. If only the traveled distance is considered, while employing the traditional method, the overall distance covered by the swarm was approx. 1.6 times more as compared to the distance covered by the swarm using the Partial Swarm SLAM method.

Table 1: Total distance (in meters) travelled by agents, comparison between Partial Swarm SLAM and traditional method

Agent No.	PS-SLAM	Traditional
1	310m	660m
2	310m	660m
3	490m	660m
4	480m	660m
5	480m	660m

5 Conclusion

In this paper, we developed an algorithm for utilizing the SLAM technique partially in order to guide dispersed agents of the swarm towards the destination. The agents in the sub-swarms, keep a track of the vital information about their movements, such as tracker points (the coordinates) at which the swarm/sub-swarm gets divided into further sub-swarms, the timestamps between two adjacent tracker points, the velocity at which the agent traveled between the tracker points, the which direction the agent turned towards, the shape or the gap available between the obstacles/objects. This key information is then shared by the agent that finds a clear route to the destination. The agents in the rest of the swarm then utilized this information to quickly localize themselves while navigating and tracing back towards the tracker points left by the broadcasting agent. The simulation results provide sufficient proof that the proposed methodology works reliably in the simulated environments. The results evidently show that this technique helps navigating the swarm in an unknown environment with no-prior map information and without any communication with central servers for path-finding purposes. The chances for the swarm agents to reach a local minimum by utilizing this technique are minimized considerably in comparison to priority-based or randomized turning methods.

In the future, we plan to include detailed comparative and analytical results with other techniques, to show the efficiency of the proposed technique. Furthermore, the proposed approach will be further developed by injecting noise in the setup by including the IMU drift and limited range communication between the agents to analyze the efficiency of the proposed approach in-depth.

The chain linked limited range communication between the agents will restrict the extent to which the agents can disperse. It will also be interesting to analyze, which tracking back utilizing the broadcasted trackers, if an agent or a sub-swarm has to deviate, for instance, due to any newly added obstacle, and how computationally expensive the re-localization process will be.

References

1. Yasin, J.N., Mohamed, S.A.S., Haghbayan, M.H., Heikkonen, J., Tenhunen, H., Yasin, M.M., Plosila, J.: Energy-efficient formation morphing for collision avoidance in a swarm of drones. IEEE Access **8**, 170681–170695 (2020). https://doi.org/10.1109/ACCESS.2020.3024953
2. Shakhatreh, H., Sawalmeh, A.H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N.S., Khreishah, A., Guizani, M.: Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. IEEE Access **7**, 48572–48634 (2019). https://doi.org/10.1109/ACCESS.2019.2909530
3. Ladd, G., Bland, G.: Non-Military Applications for Small UAS Platforms. https://doi.org/10.2514/6.2009-2046, https://arc.aiaa.org/doi/abs/10.2514/6.2009-2046
4. Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence **7**(1), 1–41 (2013)
5. Yasin, J.N., Haghbayan, M.H., Yasin, M.M., Plosila, J.: Swarm formation morphing for congestion-aware collision avoidance. Heliyon **7**(8), e07840 (2021)
6. Martínez, D., Pallejà, T., Moreno, J., Tresanchez, M., Teixidó, M., Font, D., Pardo, A., Marco, S., Palacín, J.: A mobile robot agent for gas leak source detection. In: Bajo Perez, J., Corchado Rodríguez, J.M., Mathieu, P., Campbell, A., Ortega, A., Adam, E., Navarro, E.M., Ahrndt, S., Moreno, M.N., Julián, V. (eds.) Trends in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection. pp. 19–25. Springer International Publishing, Cham (2014)
7. Kegeleirs, M., Grisetti, G., Birattari, M.: Swarm slam: Challenges and perspectives. Frontiers in Robotics and AI **8**, 23 (2021). https://doi.org/10.3389/frobt.2021.618268, https://www.frontiersin.org/article/10.3389/frobt.2021.618268
8. Park, S., Kim, H.: Dagmap: Multi-drone slam via a dag-based distributed ledger. Drones **6**(2) (2022). https://doi.org/10.3390/drones6020034
9. Kegeleirs, M., Garzón Ramos, D., Birattari, M.: Random walk exploration for swarm mapping. In: Althoefer, K., Konstantinova, J., Zhang, K. (eds.) Towards Autonomous Robotic Systems. pp. 211–222. Springer International Publishing, Cham (2019)
10. Mattar, E.A.: Mobile robot feature-based slam behavior learning, and navigation in complex spaces. In: Hurtado, E.G. (ed.) Applications of Mobile Robots, chap. 4. IntechOpen, Rijeka (2018). https://doi.org/10.5772/intechopen.81195
11. Yasin, J.N., Mahboob, H., Haghbayan, M.H., Yasin, M.M., Plosila, J.: Energy-efficient navigation of an autonomous swarm with adaptive consciousness. Remote Sensing **13**(6) (2021). https://doi.org/10.3390/rs13061059, https://www.mdpi.com/2072-4292/13/6/1059
12. Sumpter, D.J., Beekman, M.: From nonlinearity to optimality: pheromone trail foraging by ants. Animal behaviour **66**(2), 273–280 (2003)