International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 12, No.1 (Oct-2022)

https://dx.doi.org/10.12785/ijcds/120180

A Decentralized Coordination Algorithm for Patrolling and Target Tracking in Internet of Robotic Things (IoRT) using Dynamic Waypoints and Self-triggered Communication

Janardan Kumar Verma[1] **and Virender Ranga**[2]

[1]*Department of Computer Engineering, National Institute of Technology, Kurukshetra, Haryana, India*
[2]*Department of Information Technology, Delhi Technological University, Delhi, India*

Received 22 Jan. 2021, Revised 15 Jul. 2022, Accepted 23 Jul. 2022, Published 31 Oct. 2022

Abstract: The Internet of Robotic Things (IoRT) deals with autonomous objects such as robots and sensors together. It can be used for efficient patrolling and target tracking; however, mobile sensors (robots) must coordinate to decide their optimal actions. This paper proposes a decentralized coordination strategy that can enable multiple robots to perform both patrolling and target tracking in the deployed scenario. The basic idea is that the whole site is divided into local zones, and a single robot is deployed in each zone. All robots are logically divided into two groups, namely category $C1$ and category $C2$ robots. A dynamic waypoint generation algorithm is proposed to assist category $C1$ robots in the perimeter patrol. It produces the waypoints such that certain locations can be prioritized and intruders cannot predict the patrolling trajectory. Category $C2$ robots are responsible for area patrolling within the zone. Here, we also propose a strategy such that the places with a high probability of unusual acts can be visited frequently. We use the distributed Extended Kalman filter (EKF) to estimate and predict the position of the targets. Each robot has a self-triggered communication mechanism to share the necessary information with the neighbors, such as the estimated position of the intruder, EKF parameters, asking for help, etc. We have also developed an Internet of Thing (IoT) based web application to monitor and control the robots. In this app, robots subscribe to the server for necessary information and commands and publish their position, patrolled area, intruders' position, battery status, etc., to the server for real-time monitoring and control. The proposed solution is validated through simulations in the Robot Operating System (ROS) and Gazebo. The results show the patrolling and target tracking performance using idleness and error in the target's estimated position, respectively, as metric.

Keywords: Cooperative target tracking, Cooperative patrolling, Internet of Robotic Things Coordination, Multi-Robot System, Multi-Robot Coordination.

1. Introduction

Nowadays, the expedite advancements in Multi-Robot Systems (MRS) are motivated by robotics' progress and cooperative behavior present in various living beings such as flocks of birds, fish swarm, etc. An increasing interest in the MRS coordination [1] research domain for monitoring and surveillance applications [2] such as mapping and exploration, monitoring wildfire and environment, security, and disaster management has emerged. The multi-robot systems employ robot groups that communicate and execute various tasks need to coordinate. The mobility aspect of mobile robots enhances their inherent sensor coverage and renders supervision in areas where the deployed static nodes do not provide adequate coverage. Therefore, the combination of static sensors and MRS i.e. IoRT [3],[4] offers additional benefits, enhanced efficiency, and broader application domains. Deploying networked MRS [5], [6] and static sensor networks [7], [8] are two popular ways for tracking mobile targets. The robots detect and track the moving target by sharing the target's information with neighbor robots. These robots estimate the target's position as the tracking destination and regulate their motion towards it.

Along with target tracking, this paper focuses on engaging multi-robots in the patrolling problem also. In patrolling problems, robots monitor the environment for a longer extent of time. They move and sense the environment for any unusual activity. In some cases, they can collect data from the deployed sensors in the environment and transmit the captured data to the base station for further actions. Integrating of patrolling and tracking can provide various benefits such as: when the intruder is detected by patrolling team there is no need of additional communication and team to track, it can enhance the response time of the action taken on the intruder, and it eliminates the need for separate detection system to send alert when intruder enters the zone. Combining tracking and patrolling is widely applicable in civilian, military environments such as surveillance around a sensitive site, house, livestock farms, for security at airport, etc.

E-mail address: janardan18@gmail.com, virenderranga@dtu.ac.in

In our proposed work, initially, robots operate in a patrolling mode where multiple robots patrol a large premise, subdivided into locally protected zones that are assigned to each robot (as shown in Figure 1). Robots cannot enter the zones of other robots unless they are requested (through help message) for helping. When the intruder is detected and needed to be tracked, the corresponding robot immediately switches to the tracking mode. After completing their tracking task, they switch back to the patrolling mode. In this proposed solution, we have tried to minimize the idleness time of sensitive places and maximize the target's observation time with minimum estimation error and communication cost.

2. Related Work

Several multi-robot coordination algorithms for patrolling and target tracking have been proposed. The detailed information of such algorithms is presented in [9], [10], [11]. Chang et al. [12] developed a patrolling algorithm in a wireless sensor network (WSN). In the first step, the tours are planned in the algorithm. The next step consists of sending data mules on these precomputed tours, and in the last step, the speed of the dispatched nodes is regulated to fulfill the revisit constraints of the points of interest.

Authors in [13] proposed a patrolling algorithm that accounts for the robots joining and leaving the linear perimeter with dynamic length. The work shown in [14] focuses on the movement of robots along with their segments. The robots coordinate locally with their neighbors to reciprocate the fluctuations in the team size, speed, and perimeter length. A velocity controller for the robots following their tours is presented in [15]. The main focus is to reduce the uncertainty, which increases with different rates in an environment. In [16], the tours for other sensing locations are determined to meet the idleness constraint. Furthermore, the periodicity property of these tours is investigated. Nigam et al. [17] proposed a method to convert the problem of idleness minimization into the short horizon control law. This method finds the locations that the robot should visit. The patrolling problem using multiple robots can be categorized into two subparts: 1. Planning paths for multi-robots in the environment, 2. Coordinating and regulating the movement of the robots in these planned paths. Minimizing idleness is the optimization criteria in patrolling problems. Pasqualetti et al. [18], proposed algorithms to compute minimal idleness segments for the robots in a given cyclic graph or tree and chain. Authors in [19] have calculated a tour for the environment consisting of sensor locations, including various priorities. The control law is also developed for coordinating the movement of robots in the precalculated tour with the main focus on reducing weighted idleness.

Generally, target tracking approaches are based on approximating the target's position and coordinating the robots' or sensors to estimate the target position and move towards it. Various proposed works use Distributed Kalman Filter (DKF) with slight variations to evaluate the target's location [20], [21]. In such works, the information in the DKF expresses numerous measurements along with the covariance matrices. This information form in DKF can be signified to summarize each sensor's measurements and covariance matrix. In [20], a consensus algorithm is proposed to implement the Kalman filter in a distributed way and estimate the sums. Consensus algorithms provide various ad hoc applications for Kalman filters in a distributed manner. Consensus Kalman Filter (CKF) is proposed in [22] to optimize the consensus matrix, and Kalman gains design parameters. Speranzon et al. [23] presented a distributed adaptive algorithm to analyze the effect of packet losses and measurement noises. The adaptive weights are computed at each node locally to reduce the estimation error variance. The proposed algorithm also considers dispersing conditions for the weights to ensure the whole network's estimation stability. Some approaches need the formation control algorithms after estimating the target's position so that robots can congregate towards the approximate target's position. Hu et al. [24] implemented a vision-guided control strategy on two robot fishes for target tracking. In [25] and [26], path following and sliding-mode method are employed for formation control, respectively.

In our work, the proposed algorithm can handle tracking and patrolling simultaneously. Moreover, the waypoint generation (for patrolling purpose) can handle prioritized patrolling (keeping lower idleness of sensitive locations) and the patrolling trajectories are probabilistic thus making it difficult to be predicted by the intruder. This is largely neglected in the presented works. We have also developed self-triggered communication stagey to reduce the communication cost, which is not considered in the above works. For target position and estimation, we have used EKF, however our modified EKF is distributed and the EKF parameters are shared through communication messages (only at triggering conditions) only with the neighbor robots. Thus, minimizes the complexity and communication cost of distributed EKF.

3. Problem Description

A large site is considered where premises are divided into locally protected zones. Each robot can patrol in its local area of the site, ensuring comprehensive control over its most remote parts. In some cases, the robot can leave (for a fixed time interval) its local area to assist other robots. Each robot can operate in two modes patrolling and tracking. When the target needs to be followed, the robots can switch to tracking mode and then back to patrolling automatically.

The problem is to cooperatively patrol the entire perimeter of the site by the robots whose local area has one or more common edges with the site's perimeter. Robots which do not share any edge of their local area with the perimeter are responsible for area (not the perimeter) patrolling of the site. Whenever the presence of an unusual object or intruder (target to be tracked) is detected, all the robots should track the target cooperatively and share the target's live location. Robots update this to the security administrator for taking further actions. The main purpose of this proposed coordination approach is to minimize idleness time (significantly higher priority locations) and maximize the observation time of intruders (target) with the least communication cost and movement

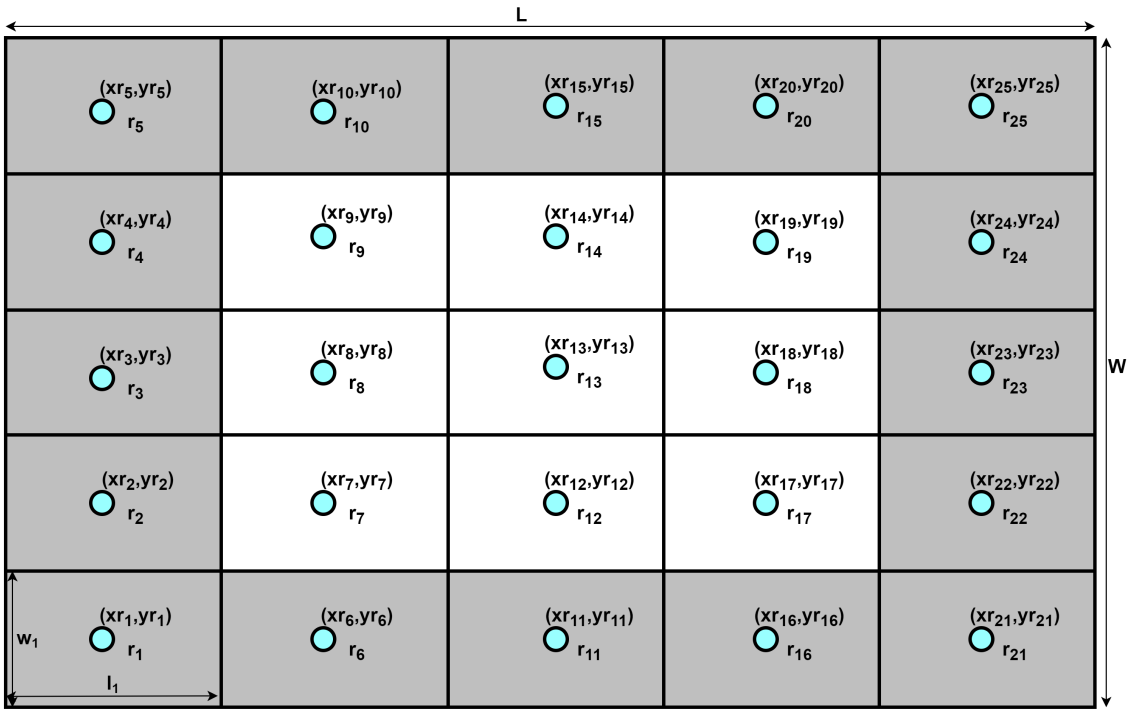

Figure 1. Problem scenario

of the robots. Coordination becomes more challenging when the target leaves the region of a robot and enters into the area of another robot.

For simplicity, the local area of each robot is represented as a rectangle or square. We divide the site of interest (S) into N number of local regions and deploy one robot into each local area of the site (S). $R = (r_1, r_2, r_3 \ldots . r_N)$ is the set of robots deployed on the site. The local area of the robot r_i can be represented by LA_i.

$$LA_i = ((xr_i, yr_i), (w_i, l_i)) \tag{1}$$

Here (xr_i, yr_i) is the center and (w_i, l_i) is the width and length of the local area. Robots are equipped with necessary sensors such as laser, camera, lidar for detecting the target and capable of communicating with monitoring station and neighbor robots. At time t the position and maximum speed of a robot can be denoted by $r_i(t) = (x_i^t, y_i^t, \theta_i^t)$ and $V_{max} r_i$ respectively. Once the target is in the sensing range of the robot, its position can be estimated using EKF. The estimated location of the target T at time t is represented by $T_t = (x_t, y_t)$.

Once the intruder has been detected inside the site's perimeter afterward, the target (intruder) can be tracked until it remains within the perimeter or neutralized. Between the time of first detection and neutralization, if any robot does not detect the target, then its predicted location can be used for tracking for a maximum time period of P_t seconds, without getting any new measurement. The predicted location of the target is denoted by $T_t^p = (x_t^p, y_t^p)$. We assume that robots can localize themselves, and the maximum speed of the target cannot be more than the maximum speed of robots $(V_{max} T < V_{max} r_i)$, i.

If two robots have one or more common edges between their local zones, such robots are defined as the neighbor of each other. For rectangular zones, any robot can have a maximum of four and a minimum of two neighbors. Due to the decentralized nature of the coordination approach, each robot maintains its neighbor list Nr_i.

$$Nr_i = [(r_j, LA_j)]\, j \tag{2}$$

Here j is the id of the neighbor robot. The neighbor list is a set containing pair of all the neighbors r_j, and their local area LA_j. Local zones assigned to each robot are fixed. Therefore, the neighbors of each robot are unchanged. Though any change in the neighbors can be communicated to the corresponding robots through the central monitoring server. The detection of target T at time t by the robot r_i can be defined as O_i^t :

$$O_i^t = \begin{cases} 1 & \text{if } T \in FOV \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

We have also developed a web app for monitoring and control of IoRT. Each robot can send its position and location of any unusual activity or target to the IoT server. The administrator can view live location and send commands to robots over the internet from any device capable of running web apps. Each robot has a self-triggered communication mechanism to inform (when the target is intended to enter the region of some robot) its neighbors about the presence of the target to be tracked.

4. Proposed Approach

This section presents our proposed coordination algorithm for patrolling and target tracking in an environment where the site is divided into local zones. We logically categories the robots into two types, based on their local

area (zone). First category robots are denoted by $C1$. The local boundary of these robots is part of the entire perimeter. The second category is $C2$. Robots of this category do not share any portion of their provincial border with the perimeter. Each robot patrols in its assigned area. $C1$ robots are primarily responsible for perimeter patrol only, though sometimes they may visit other parts of their zone also. $C2$ robots are only accountable for area patrolling within their local zone. By default, all robots are in patrol mode. The tracking mode is activated once the robot detects the object of interest or receives a tracking command from another robot.

For target tracking purposes, all the robots are equivalent. Each one can perform target tracking whenever an intruder or target of interest is detected. Robot switches to tracking mode and estimates the position of the target. Then the object can be continuously tracked and followed while it remains in its local area. Once the object leaves from the local zone of the robot, it switches back to patrolling mode. If the object is intended to enter the local area of some other robot, then prior information can be given to the corresponding robot to keep track of the target.

A. Self-triggered Communication Policy

We use three kinds of messages to coordinate the patrolling and tracking tasks. The communication is triggered whenever any of the following three events occur: 1. the target appears near to the local boundary of any robot, 2. the target enters in the local zone of a robot, 3. when an outer robot switch back to the tracking mode. These three events correspond to three types of messages, first two responsible for tracking, and the last is used for patrolling purposes. The $type1$ and $type2$ messages can only be sent by a robot who has a target in its sensing range or it has sufficiently accurate prediction (using EKF) of the target's position. This also means that it had the target in the sensing range just a little before. If the estimated position of the target is within the local zone of the robot, then $type1$ message can be sent to all the neighbors. Otherwise, $type2$ message can be sent only to the robot in whose local zone target has entered. In other words, the message can be sent to the robot, satisfying both equations (4) and (5).

When a robot receives $type1$ message, it computes the orthogonal distance of the target from all of its edges. If the target is found near a threshold value, the robot starts moving move towards the corresponding edge. Upon receiving $type2$ message, the robot can start moving towards the target to get it within the sensing range. The $type3$ message can only be sent by $C1$ robots. Whenever an outer robot goes in tracking mode, it sends $type3$ msg to its $C1$ category neighbors (always two in case of local rectangular zones). Therefore, $type3$ messages can only be received by $C1$ robots. Upon receiving this kind of message, the robot will assist the sender by patrolling some portion of the sender's perimeter. This message is used to ask assistance from neighbors whenever some outer robot ($C1$) switches to tracking mode.

$$Tx_i = \begin{cases} \text{true} & \text{if } \left(xr_i - \frac{l_i}{2} < x_t \leq xr_i + \frac{l_i}{2}\right) \\ \text{false} & \text{otherwise} \end{cases} \tag{4}$$

$$Ty_i = \begin{cases} \text{true} & \text{if } \left(yr_i - \frac{w_i}{2} < y_t \leq yr_i + \frac{w_i}{2}\right) \\ \text{false} & \textit{otherwise} \end{cases} \tag{5}$$

B. Dynamic Waypoint Generation

The purpose of waypoints is to guide the robot for patrolling. Dynamic waypoint generation makes it difficult or impossible to predict the patrolling pattern by intruders. Both categories ($C1$ and $C2$) of robots differ in how they generate waypoints, $C1$ robots need to have their waypoint near the perimeter, as shown as a dark shaded region in Figure 2(a). Sometimes, they also need to visit the inner area of their local zones (shown as a light shaded region in Figure 2(a)). However, each robot of this category majorly patrols near the perimeter (part of the entire perimeter). $C2$ robots apply two criteria for waypoint generation. First, uniformly distribute the waypoint within the local zone such that the entire local zone can be covered with minimum idealess time. Second, give extra emphasis near the edges, which are common with a $C1$ robot. Suppose $C2$ robot has neighbors of category $C1$ then near the corresponding edges. In that case, more waypoints should be generated because that side of the local zone has a high probability of intruders being spotted, as shown in Figure 2(a). If there are no such neighbors, then waypoints should be uniform throughout the zone, as shown in Figure 2(c). There are four $(\alpha, w\_dis_{min}, w\_dis_{max}, c_{rate})$ parameters for waypoints generation algorithm. Using these parameters, we can control how far the next waypoint be and how fast robot want to converge towards the destination. The lower value of c_{rate} means the slow convergence towards the destination, and α controls the arbitrariness in the direction or number of direction a waypoint can be generated. A significant value means more alternate paths. The random distance (r_{dis}) of the next waypoint can be minimum w_dis_{min} and maximum w_dis_{max} from the current position. Every time the waypoint is generated at a random distance from the current position (step 3).

Then we find (using equation (7)) α number of equally spaced points on the circumference of circle with center (st_x, st_y) and radius r_{dis}.

$$\theta = \left\{ n\frac{2\pi}{\alpha} \right\}, n0, 1, 2 \ldots \alpha - 1 \tag{6}$$

Where θ is set of equally spaced angles in radian corresponding to each equally spaced point on circumference and n is an integer ($0n < \alpha$).

$$PO = \{(st_x + r_{dis}\cos\delta,\ \ st_y + r_{dis}\sin\delta)\ \ \ \delta\theta\} \tag{7}$$

Where PO is the set of all equally spaced points on the circumference. Now we need to calculate distance of each point in PO from the destination and find the point which

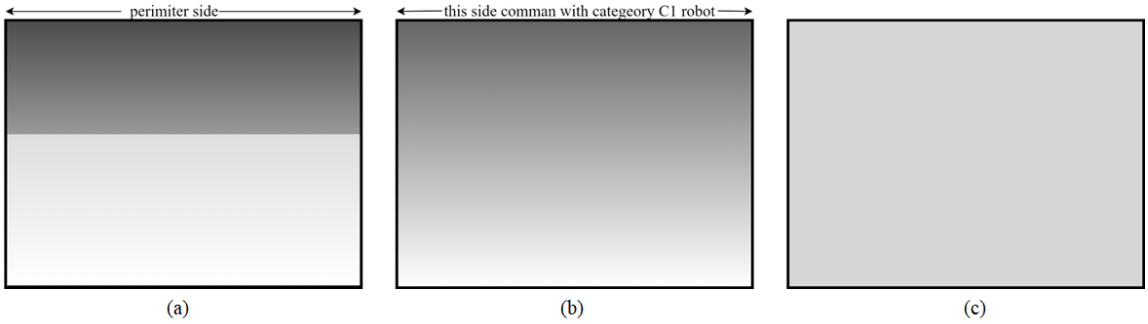

Figure 2. Varying density of waypoint generation

Algorithm- 1:

1. ***generate _waypoints***$(start,\ destination)$:
2. $define\ \alpha, w\_dis_{min}, w\_dis_{max}, c_{rate}$
3. $get\ random\ distance\ r_{dis}\ where\ (w_{dis_{min}} \leq r_{dis} \leq w_{dis_{max}})$
4. $PO = find\ \alpha\ points\ using\ (6) and\ (7)$
5. ***for each*** $point$ ***in*** PO
6. $find\ its\ distance\ from\ destination$
7. $calculate\ weight\ and\ probability\ of\ the\ point\ using\ (9)\ and\ (10)$
8. ***end for***
9. $PO_u = select\ a\ random\ point\ as\ next\ waypoint\ using\ (11) and\ (12)$
10. $start = PO_u$
11. ***end***

has minimum distance from it.

$$b = MIN\left(\left\{\sqrt{(PO_{n,x} - dst_x)^2 + \left(PO_{n,y} - dst_y\right)^2}\right\} \quad \forall n \in \{0, 1, 2 \ldots, \alpha - 1\}\right) \tag{8}$$

Where $PO_{n,x}$ is x coordinate of n^{th} point in the set PO, similarly $PO_{n,y}$ is n^{th} y coordinate. Here z is the index of point which has minimum distance to the destination point (dst_x, dst_y).

$$W = \left\{e^{(c_{rate} * (\cos(\theta_i - \theta_z) - 1))}\right\} \quad \forall i \in \{0, 1, 2, \ldots \alpha - 1\} \tag{9}$$

Where W is the set of weights assigned to each points of set PO. θ_z is angle of the point which has minimum distance to the destination. Now we need to calculate the probabilities of each point. After that, a probability is assigned to each point in PO based on its distance from the destination and c_{rate} (step 5-8). Points closer to the destination gets a higher probability.

$$P = \left\{\frac{W_i}{\sum_{u=0}^{\alpha} W_u}\right\} \quad \forall i \in \{0, 1, 2, \ldots \alpha - 1\} \tag{10}$$

Where P is the set of probabilities for each point under consideration, W_i is the weight of i_{th} point from set W.

$$E = [E_0,\ E_1,\ E_2 \ldots . E_\alpha] \tag{11}$$

Where $E_0 = 0$, $E_j = \sum_{i=0}^{j-1} P_i \quad \forall j \in \{0, 1, 2, \ldots \alpha\}$ Now generate a random number rd between zero and one. After that find the index u such that equation (12) is satisfied.

$$E_u rd < E_{u+1} \tag{12}$$

Where $u\alpha - 1$. Therefore, the next generated waypoint is PO_u, points having higher weight in W have high chance to be selected however others can also be selected.

Value of c_{rate} plays a vital role in this regard, large c_{rate} means the difference between the probability, assigned to points close to the endpoint, and points farther can be significant. Therefore, the chance of selecting a waypoint close to the endpoint may be higher. However, for smaller values of c_{rate} this difference can be lesser. Thus, the chance of choosing a waypoint close to the endpoint cannot be as high.

C. Coordination Algorithm

Robots need to coordinate their actions to perform an efficient patrolling and tracking. The complete coordination algorithm is presented in *Algorithm* – 2. The default mode of all the robots deployed in the vicinity is patrolling. Each robot depending on its category generates dynamic waypoints (Wl_i). Procedure shown in *Algorithm* – 1 is the same for all the robots to generate waypoints. However, the density of waypoints generated at different places inside the LA_i differs for each category of robot. if there is no intruder detected on the site, robots can continue their patrolling (step 3-9). Whenever an intruder is detected, the corresponding robot switches in the tracking mode and starts following the intruder; all other robots keep patrolling in their respective local zones. The extended Kalman Filter (EKF) is used for the estimation and prediction of the target position (equation (13) - (17)).

Prediction :

$$X^{'} = DX + \beta \tag{13}$$

$$P^{'} = DPD^T + \gamma \tag{14}$$

Update :

$$K = P^{'} H^T (HP^{'} H^T + R)^{-1} \tag{15}$$

$$X = X^{'} + K * (z - HX^{'}) \tag{16}$$

$$P = (I - KH)P^{'} \tag{17}$$

Where X is the state of the target, K is Kalman gain matrix, γ is noise covariance matrix, P is process covariance matrix, $P^{'}$ is predicted process covariance matrix, z is the measurement vector, H is measurement function, and I is the identity matrix equal to the size of state vector X.

Algorithm- 2:

Inputs

LA_i: Local zone of Robot i,

Nr_i: Neighbours of Robot i,

Wl_i: Dynamic List of waypoints to follow,

P_t: Prediction Timer (how long to predict)

F: Frequency of sending type1 messages

1. mode = Patroling;
2. **Repeat** (at each time step t):
3. **if** (mode = Patroling)
4. **if** (Wl_i list is empty):
5. Wl_i = generate waypoints using Algorithm 1
6. **end if**
7. move to next waypoint (w_x, w_y) in the Wl_i
8. $Wl_i = Wl_i - (w_x, w_y)$
9. **end if**
10. **if** (mode = Patroling and category = C1)
11. **if** (help message received)
12. update Wl_i to patrol some area of other robot
13. **end if**
14. **if** (stop_help message received)
15. update Wl_i to patrol own area only
16. **end if**
17. **end if**
18. **if** (type1 message received)
19. compute target distance from all edges
20. **if** (target is very near to any edge)
21. move towards the edge nearest to target
22. **end if**
23. **end if**
24. **if** (type2 message received)
25. move towards target
26. update EKF using Int_message.EKF_data
27. start P_t
28. **if** (mode = Patroling and category = C1)
29. Send help msg to Nr_i
30. **end if**
31. mode = Tracking
32. **end if**
33. **if** (target in sensing range)
34. detect and estimate target position
35. update EKF using measurment
36. start P_t
37. **if** (estimated position is within LA_i)
38. move towards target;
39. **if** (mode = Patroling and category = C1)
40. Send help msg to Nr_i
41. **end if**
42. mode = Tracking
43. **if** (target is near edge)
44. send type1 msg to Nr_i once in F secs
45. **end if**
46. **end if**
47. **else**
48. if mode = tracking: start patroling timer
49. R_j = estimated target position is within LA_j
50. send type2 msg to R_j
51. **end else**
52. **end if**
53. **else**
54. **if** (P_t is valid)
55. predict target position
56. **if** (predicted position is within LA_i)
57. move towards target
58. **if** (target is near edge)
59. send type1 msg to Nr_i once in F secs
60. **end if**
61. **end if**
62. **else**
63. if mode = tracking: start patroling timer
64. R_j = predicted position is within LA_j
65. send type2 msg to R_j
66. **end else**
67. **end if**
68. **end else**
69. **if** (patroling timer expired)
70. **if** (mode = Tracking and category = C1)
71. Send stop_help msg to Nr_i
72. **end if**
73. mode = Patroling
74. **end if**
75. **End Repeat**

$$D = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \beta = \begin{pmatrix} \frac{a_x \Delta t^2}{2} \\ \frac{a_y \Delta t^2}{2} \\ a_x \Delta t \\ a_y \Delta t \end{pmatrix}$$

Here β is random acceleration vector and $a_x \sim N(0, \sigma_{ax}^2)$ and $a_y \sim N(0, \sigma_{ay}^2)$ are random acceleration in x and y direction.

Whenever a $C1$ category robot switches from patrolling to tracking mode, it must send a help message to its neighbors (step 31-34, 43-46) to patrol its perimeter while it's in tracking mode. While tracking the target when the target/intruder reaches very near (predefined threshold) to any edge of the local zone, the robot sends a $type1$ message to neighbors (step 47-49, 63-65). The purpose of this message is to alert the neighbors that intruded may enter their local zone. Therefore, neighbors should move towards the corresponding edge from where intruders can come (step 20-25). When a robot detects the target in another robot's local zone, it sends a $type2$ message, and receiving robot moves towards the location, observed in the message (step 51-55, 67-71). It also starts its patrolling timer if the robot is in tracking mode. This means that the intruder is out of its local zone. Therefore, this robot will return to patrolling mode after waiting a certain amount of time, i.e., when the patrolling timer expires. Whenever such patrolling timer expires for a $C1$ robot, it sends a $stop\_help$ message to its neighbors (step 74-77), upon receiving this message, neighbors stop patrolling the perimeter of other robots (step 15-17).

We have used a central server, which maintains the global information of our IoRT system. It has information about robots. If the target gets lost, then the server can estimate its approximate location and direct the nearby robot to search.

D. Algorithm Analysis

In the tracking part of proposed algorithm, EKF is well known for the estimation and prediction of dynamic state of a linear and non-linear system. The acceleration of the target is modelled as noise ($a_x \sim N(0, \sigma_{ax}^2)$, $a_y \sim N(0, \sigma_{ay}^2)$). We have used timer P_t to ensure the accuracy of predicted state.

Scalability: The algorithm is well scalable due to its decentralized (all robots other than neighbors do not need to take part in any communication) nature.

Communication cost: Number of messages has been greatly reduced due to self-triggered policy. The number of type1 messages depends on F and Nr_i. Which can be reduced (by increasing F) further at cost of tracking quality.

$$No.of\ type1\ msgs\ \propto \frac{(Nr_i)}{F}$$

The number of $type2$ messages is directly proportional to the number of times target moves from the region of one robot to another. $help$ and $stop_{help}$ messages are only sent by outer robots when target passes through their region, which is less likely to happen once the target is inside the perimeter. Therefore, such messages are needed when a new target first enters in the site.

In the patrolling part of the proposed algorithm, $(\alpha, w\_dis_{min}, w\_dis_{max}, c_{rate})$ provide the required control over waypoint generation. Random number rd used in equation (12) ensures that patrolling path is unpredictable (any point can be selected as next waypoint even if it's not at minimum distance from end point). Distance between the waypoints, randomness in their direction can be controlled by c_{rate} and α.

The algorithm will always find the next waypoint close to the end point because we look in every direction (equation (7)).

$$\theta = \left\{ n\frac{2\pi}{\alpha} \right\}, n0, 1, 2 \ldots \alpha - 1 \tag{18}$$

After few iterations the waypoints will converge near the end point. This is ensured by weights assigned as per following:

$$Weights = \left\{ e^{(c_{rate} * (\cos(\theta_i - \theta_z) - 1))} \right\}$$

θ_z is the angle of point which has minimum distance to the end point. θ_i is angle of all other points.

5. Simulation and Results

We have implemented and simulated our proposed coordination algorithm with Gazebo, ROS, and Matlab. The robots (Pioneer-3AT [27]) are differential drive-based, and each robot is equipped with a camera to detect the target, having a sensing range $6m$ and fov $1.5rad$. Each robot has a maximum linear speed of $1.4m/s$, and angular velocity of $0.6rad/s$. Each robot has assigned a local zone of $30 * 30m^2$. For most of the experiments site area is $120 * 150m^2$, which means twenty robots (grid of $4 * 5$) need to be deployed to cover the entire area. We have also simulated nine robots with $3 * 3$ grid to verify case when $C2$ robot has four neighbors of type $C1$. Each robot has a peer to peer connection with all (maximum four) its neighbors.

Gazebo[28] is a 3D robotics simulator, it can simulate the robot dynamics, environment, and targets. Figure 3, shows Gazebo environment with twenty robots and moving targets. Targets are simulated as blue color ball; they are continuously moving on random (as shown in Figure 7) and circular path. For some time, they remain inside the perimeter then move out and after some time some of them again enter the site. The simulator is capable of generating sensor data (such as image, IMU data, location etc.) and motion, with due consideration of gravity, friction etc. ROS can interact with Gazebo thus it can receive and send commands to entities (robots and target) present in the Gazebo environment. we have developed a ROS node which is deployed on each robot, this node is responsible for communication with IoT server using MQTT protocol. Matlab can also interact directly with ROS, and through ROS it can communicate with Gazebo.

The area of the local zone can be decided based on the sensor range, required idealness in patrolling, and accuracy in tracking. If the local zone is large, then the

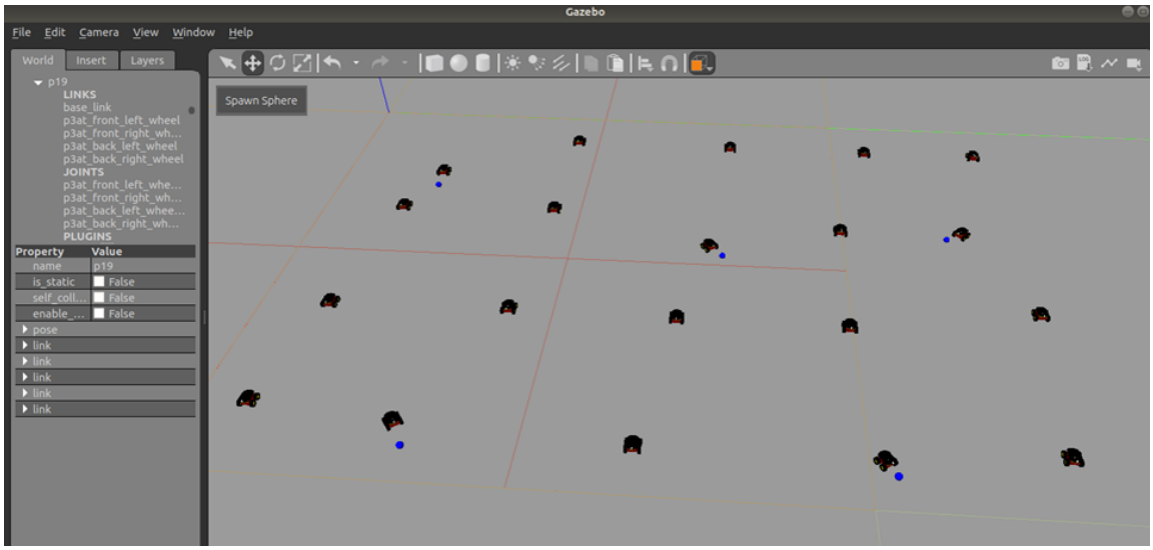

Figure 3. Gazebo Simulation Environment

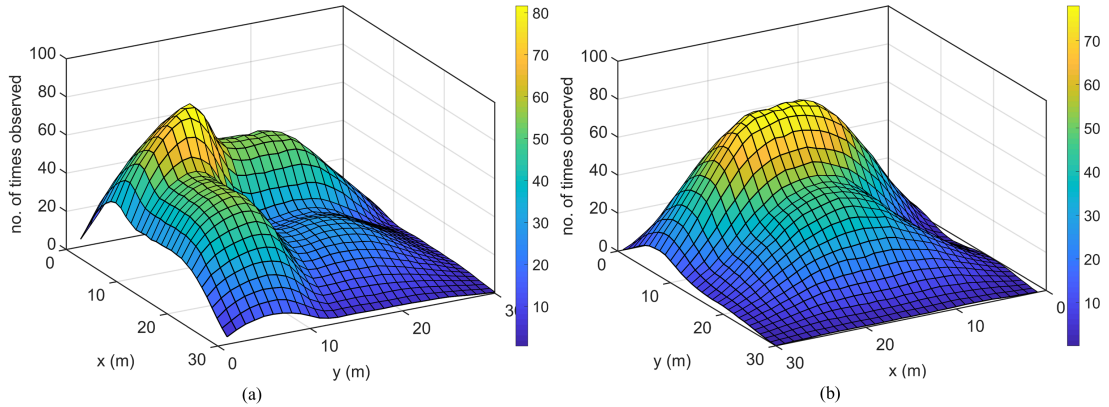

Figure 4. Observation count of each point (within the local zone) (a) of C1 with two open edges (b) of C1 with one open edges

idealness may increase. However, it does not affect the tracking quality if triggering condition (threshold distance) for sending $type1$ msg is also adjusted accordingly. Larger the zone area larger the threshold should be. However, it can increase the communication cost. In our scenario, we do not need the idealness of each point in the local zone to be equal or fair. In this case, robots require to visit some places frequently as compare with other places. For category $C1$ robots, need to visit points near the site's perimeter frequently, and category $C2$ robots need to visit those places near the local zone of $C1$ robots, i.e., frequent visits near the edges command with $C1$ robots.

Figure 4(a) shows how many times different points in the local zone are observed (during one patrol cycle) by the robot of category $C1$ and two edges of its local zone from the perimeter. Figure 4(b) shows the same for the robot of category C1 but only one edge of its local zone forms the perimeter. Each point within the local zone is considered as an area of $1 * 1m^2$. As expected in both cases robots are frequently patrolling the area near the perimeter. The peaks indicate the highest visited points. In Figure 4(a), there are points along the line around $y = 4$ *to* 12, and another line around $x = 4$ *to* 12. These points are near the perimeter therefore visited at high frequency compared to other points in the zone. However, in Figure 4(b), only one edge forms perimeter, only points along the line around $y = 4$ *to* 15 are visited the most. This satisfies our requirement, the waypoints are dynamic so robot does not move from fixed places, and places that need more attention are observed frequently.

The local zone of robots of category $C2$ does not have any perimeter. However, they can have a varying number of neighbors depends on the position of their local zone. Here we are only concerned about neighbors of category $C1$. As can be seen in Figure 1, r_7 has two such neighbours (r_2 *and* r_6), r_{12} has one such neighbour (r_{11}) etc. Therefore there can be at most four cases in the case of the local rectangular zone; having four (only exists if nine robots are deployed in a $3 * 3$ grid) or two (such as r_{19}) or one (such as r_{14}) or none (such as r_{13}) of such neighbours. Figure 5 shows the number of times different points in the local zone are observed (during one patrol cycle) by the robot of category $C2$. Figure 5(a) presents the case when there are four neighbors of category $C1$ that means that there are high chances that intruder may come from any of the four edges. Therefore, the robot visits near the four edges frequently, it can be seen in Figure 5(a). however, it leads to four peaks because corner places

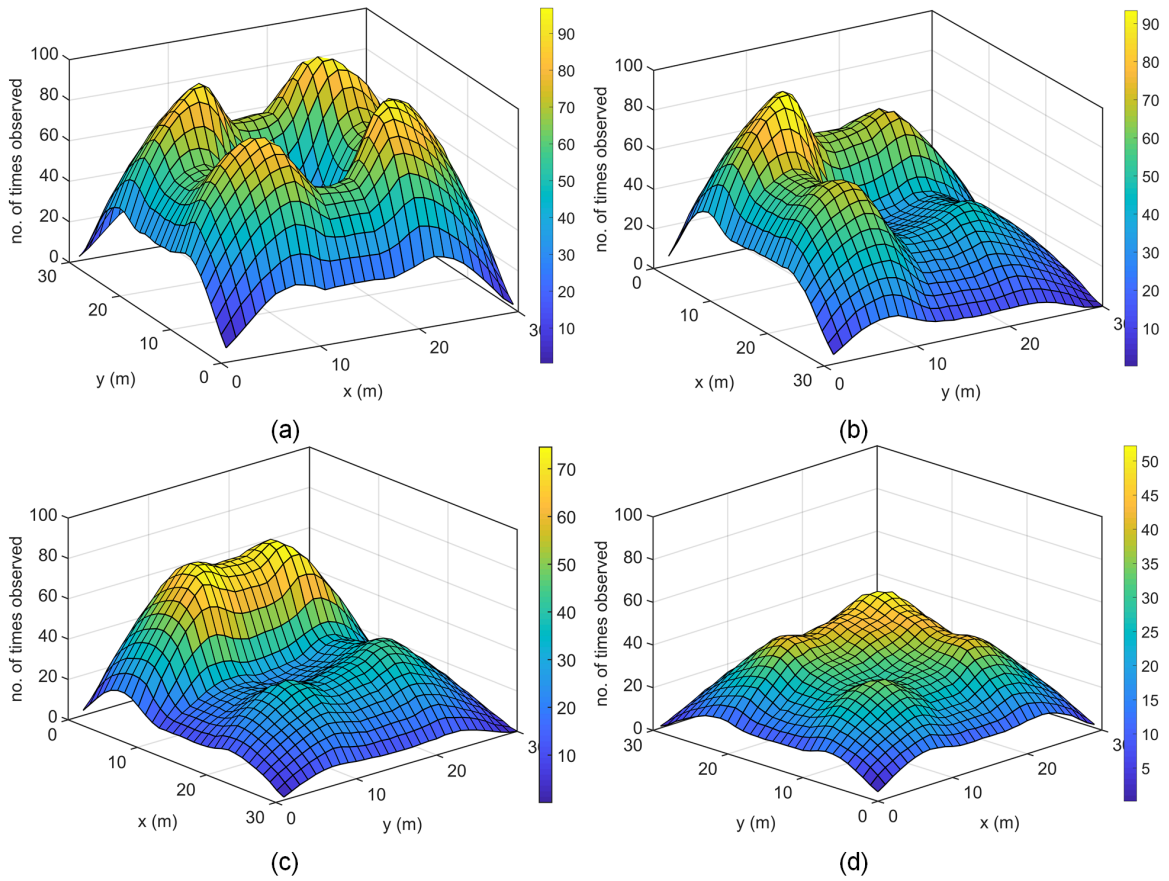

Figure 5. Observation count of each unit point (within the local zone) of C2 robots (a) with four neighbors (b) with two neighbors (c) with one neighbor (d) with no neighbor

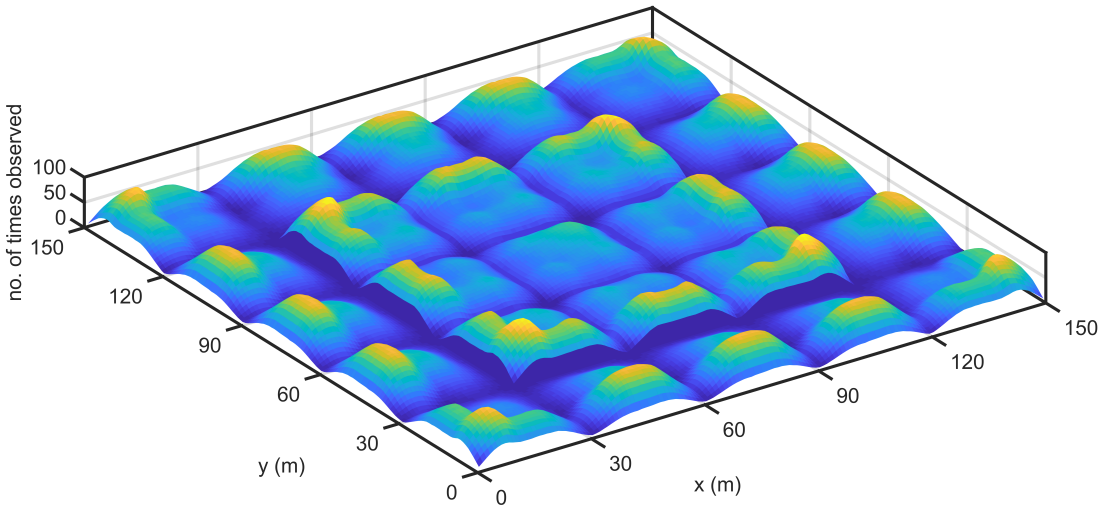

Figure 6. Observation count (each unit point) of entire site with twenty-five robots

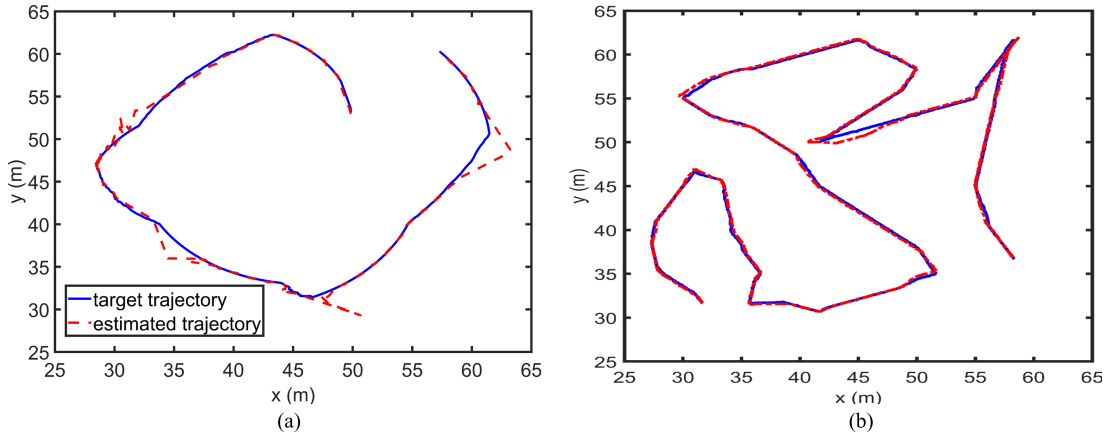

Figure 7. Tracking performance with the target moving on a random path (a) type1 message only (b) both type1 and type2 messages

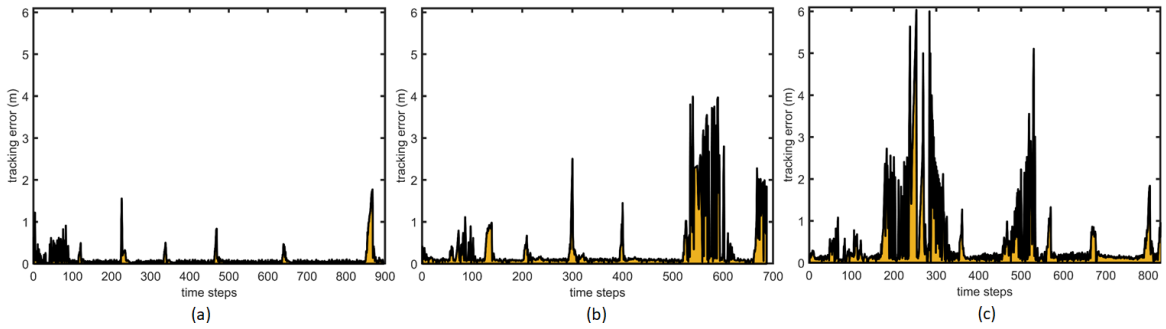

Figure 8. Tracking error with the target moving at (a) $v_T = 0.9m/s$ (b) $v_T = 1.2m/s$ (c) $v_T = 1.4m/s$

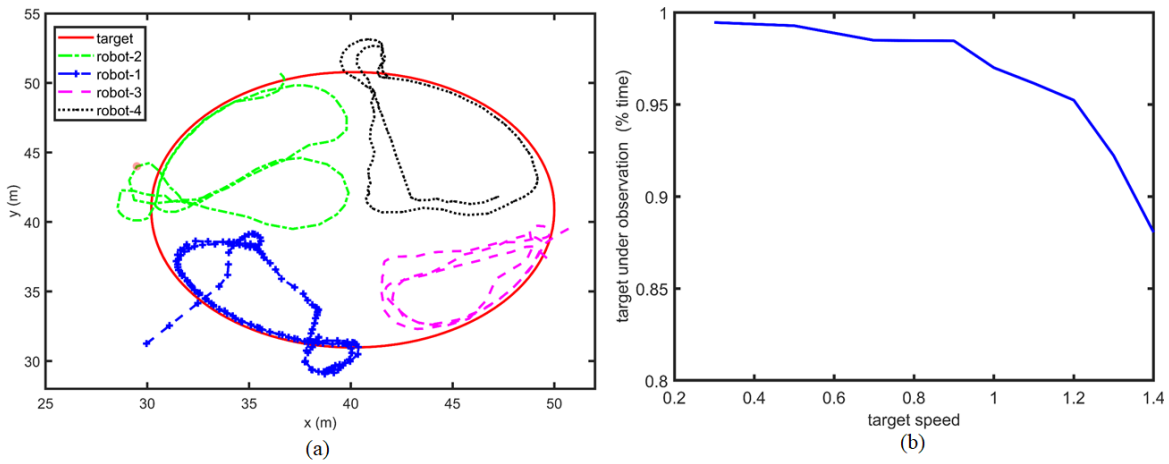

Figure 9. (a) Path followed by robots while tracking the target (b) Effect of observation time with the increasing speed of the target

act as an exchange point for two edges. Thus, visiting frequency of corner points is approximately thirty percent higher than points along the edges. Figure 5(b) shows the result for a robot that has two such neighbors. Therefore, it visits points along with the two of its borders frequently. It seems similar to Figure 4(a); however, the difference is here, the remaining (other than near the edges) area is visited more frequently as compared to Figure 4(a).

Robots are having only one of such neighbors need to worry about only one of its edge. Figure 5(c) shows the result for such a case. One side is visited frequently at other places visiting frequency is around fifty percent lesser. Figure 5(d) is the case when there is no such neighbor. Therefore, the robot is only responsible for area patrolling without special attention near the edges. Therefore, curve is almost flat, except for some places with small peaks near the corners.

Simulation has been performed with extreme settings such as target moving with complex and sharp turns, speed of target very close to the maximum speed of robots, target moving from one zone to another very frequently, to verify that the algorithm can perform coor-

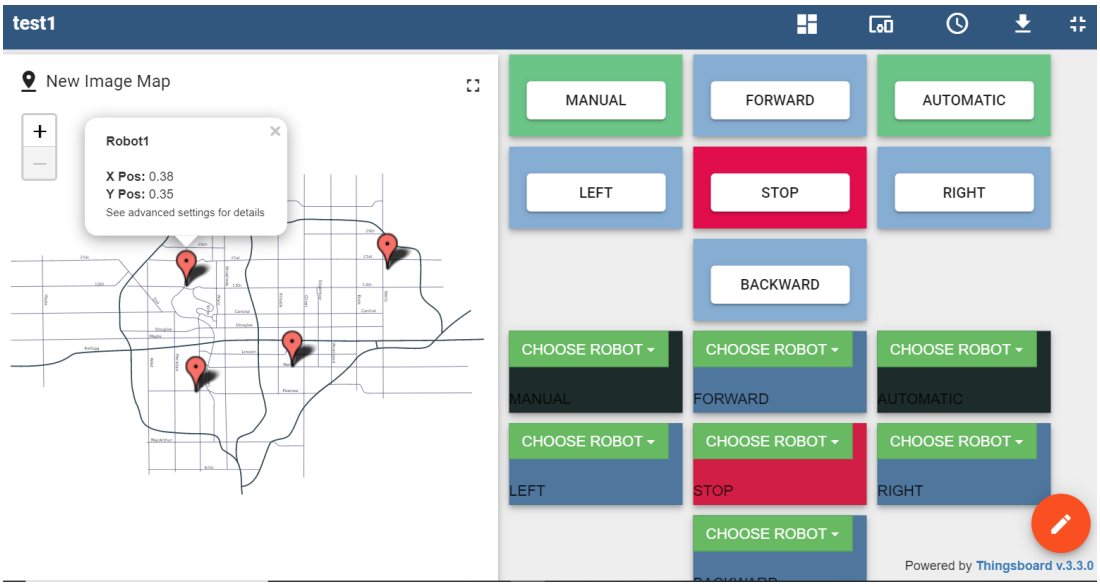

Figure 10. Snap of web-app using IoT server

dinated tracking in extreme situations.

Figure 6, shows the result of simulation over forty iterations with twenty-five robots. It can be seen in Figure 6 that all outer robots patrol in such a way that entire perimeter has a high (yellowish peaks) observation count. $C2$ robots have also formed a wall like defense to detect any intruder, which may have not be detected by $C1$ robots. Edges having less observation count are in turn have higher count by neighbor robot.

In Figure 7, tracking performance is shown; the difference between estimated and actual trajectory shows the deviation from real values. Results are shown in Figure 7(a) shows poor performance (because it only uses $type1$ message) as compared to Figure 7(b) (using both $type1$ and $type2$ message). This shows the importance of $type2$ messages. Whenever intruder/target move from the local zone of one robot to other, the chances of error in the estimated position of the target increases. This is because the robot needs to hand over the target to another robot when it reaches the end of its local zone. Figure 7(b) target takes sharp turns, and crossing the local zones is frequently still. The performance is quite good. It can be seen that the performance decreases when the target crosses the grid lines (switches the zone). Sometimes there can be a higher error in estimation due to a sudden change in target speed, direction.

Figure 8 shows an error in the estimated position of the target with respect to the time with varying target speeds. As the speed of the target is increasing from $0.9m/s$ to $1.4m/s$, the maximum error (at the crossing area) in the estimation is also increasing. At a speed of $0.9m/s$ the maximum error is around $1.9m$, at speed of $1.2m/s$ it is around $4m$, and at speed of $1.4m/s$ (this the maximum speed of robots), it's about $6m$. Along with the increment in the maximum error, the persistence time of the error is also increasing. At a lower speed, the error persists for a shorter time spawn. Accurate estimation of target position becomes more complex at a higher rate. However, the error while the target is within the local zone (not crossing it) is relatively low, as shown in Figure 8(a). The error is around $0.1-0.2m$ only. Even when the target speed is equal to the maximum speed of the robot, the error is below $0.5m$ most of the time and the robots were able to successfully track and follow the intruder.

Figure 9(a) shows the path of the target and path of the four (from whom local zone target passes) robots while tracking. It can be observed here that the robots are successfully coordinating while remaining in their respective local zones. Here, the local zone of all four robots is $40 * 40m$. Figure 9(b) shows how the target observation time decreases with an increase in speed. It explains why the error in the estimation increases, as shown in Figure 8.

Figure 10 shows a screenshot of the web app connected to the IoT server. The left part shows the current location in the environment and other related information of each robot. The right part shows the control menus of the robots. The algorithm allows robots to coordinate for both patrolling and tracking simultaneously in the application. When an intruder is present in a robot's local zone, it performs monitoring while other robots (except neighbors) keep patrolling and vice versa.

In future we would like to address tracking of multiple targets, which is a challenging problem and it becomes harder when patrolling is performed simultaneously. How to select and assign robots when there are multiple targets? Another challenge arises when the speed of the intruder is higher than the robots in that case how to communicate and pre plan the robots which can possibly detect the intruder, it may need a prediction algorithm to trigger the robots which are already deployed near to the trajectory (future) of the intruder.

6. Conclusion

In this paper, we proposed a decentralized coordination algorithm that can perform both patrolling and target tracking in the environment where the site is divided into local zones, and each robot has assigned one zone. Patrolling is based on dynamic waypoint generation; therefore, routes taken by the robot are sufficiently distinct at the same time, it provides higher visiting frequency (or lower idleness) at certain locations as per the requirements. Robots are able to assist each other by extending and shrinking the local perimeter to be patrolled. To initiate and complete this assistance process, exactly four messages (by category C1 only) are required. Any intruder, if detected by any robot, can be tracked and followed by the robot. The information regarding the position and EKF parameters are shared with the neighbors only when the target has crossed the threshold. This, in turn, alerts the neighbors. The results show that the error in estimating the target's position at a higher speed is around 0.5 m. It increases only when the target switches from one local zone to other. Robots are successfully able to track, follow, and share the required information with other neighbors. At any point, the remote user can use a web application to monitor the position and status of robots and intruders (if present), and each robot can also be controlled by it. While patrolling, the exchange position from where a robot passes several times while moving from one side of the local zone to another is visited more than the required frequency, this can be improved in the future. The detection and tracking of an intruder can also be improved by using static sensors in the environment.

Acknowledgment

This work is supported by the University Grant Commission, Government of India, and Nvidia GPU Grant.

References

[1] J. K. Verma and V. Ranga, "Multi-Robot Coordination Analysis, Taxonomy, Challenges and Future Scope," *Journal of Intelligent Robotic Systems*, vol. 102, no. 1, p. 10, may 2021.

[2] B. Rao, H. Durrant-Whyte, and J. Sheen, "A Fully Decentralized Multi-Sensor System For Tracking and Surveillance," *The International Journal of Robotics Research*, vol. 12, no. 1, pp. 20–44, feb 1993.

[3] P. Simoens, M. Dragone, and A. Saffiotti, "The Internet of Robotic Things: A review of concept, added value and applications," *International Journal of Advanced Robotic Systems*, vol. 15, no. 1, pp. 1–11, 2018.

[4] P. P. Ray, "Internet of Robotic Things: Concept, Technologies, and Challenges," *IEEE Access*, vol. 4, pp. 9489–9500, 2016.

[5] M.-F. Ge, Z.-H. Guan, B. Hu, D.-X. He, and R.-Q. Liao, "Distributed controller–estimator for target tracking of networked robotic systems under sampled interaction," *Automatica*, vol. 69, pp. 410–417, jul 2016.

[6] X.-Y. Yao, H.-F. Ding, and M.-F. Ge, "Task-space tracking control of multi-robot systems with disturbances and uncertainties rejection capability," *Nonlinear Dynamics*, vol. 92, no. 4, pp. 1649–1664, jun 2018.

[7] M. Z. A. Bhuiyan, G. Wang, and A. V. Vasilakos, "Local Area Prediction-Based Mobile Target Tracking in Wireless Sensor Networks," *IEEE Transactions on Computers*, vol. 64, no. 7, pp. 1968–1982, 2015.

[8] M. Elhoseny and A. E. Hassanien, "Mobile Object Tracking in Wide Environments Using WSNs," in *Studies in Systems, Decision and Control*, 2019, vol. 165, pp. 3–28.

[9] A. Ez-Zaidi and S. Rakrak, "A Comparative Study of Target Tracking Approaches in Wireless Sensor Networks," *Journal of Sensors*, vol. 2016, no. i, pp. 1–11, 2016.

[10] D. Portugal and R. Rocha, "A Survey on Multi-robot Patrolling Algorithms," in *IFIP Advances in Information and Communication Technology*, 2011, vol. 349 AICT, pp. 139–146.

[11] A. Khan, B. Rinner, and A. Cavallaro, "Cooperative Robots to Observe Moving Targets: Review," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 187–198, jan 2018.

[12] C.-Y. Chang, G. Chen, G.-J. Yu, T.-L. Wang, and T.-C. Wang, "TCWTP: Time-Constrained Weighted Targets Patrolling Mechanism in Wireless Mobile Sensor Networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 6, pp. 901–914, jun 2015.

[13] D. Kingston, R. W. Beard, and R. S. Holt, "Decentralized perimeter surveillance using a team of UAVs," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1394–1404, 2008.

[14] J. J. Acevedo, B. C. Arrue, I. Maza, and A. Ollero, "Cooperative perimeter surveillance with a team of mobile robots under communication constraints," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 5067–5072.

[15] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 410–426, apr 2012.

[16] J. L. Fargeas, B. Hyun, P. Kabamba, and A. Girard, "Persistent visitation under revisit constraints," in *2013 International Conference on Unmanned Aircraft Systems, ICUAS 2013 - Conference Proceedings*, vol. 28, no. 2, apr 2013, pp. 952–957.

[17] N. Nigam, S. Bieniawski, I. Kroo, and J. Vian, "Control of multiple UAVs for persistent surveillance: Algorithm and flight test results," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 5, pp. 1236–1251, 2012.

[18] F. Pasqualetti, A. Franchi, and F. Bullo, "On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms," *IEEE Transactions on Robotics*, vol. 28, no. 3, pp. 592–606, 2012.

[19] F. Pasqualetti, J. W. Durham, and F. Bullo, "Cooperative patrolling via weighted tours: Performance analysis and distributed algorithms," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1181–1188, 2012.

[20] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proceedings of the IEEE Conference on Decision and Control*, 2007, pp. 5492–5498.

[21] R. Rahman, M. Alanyali, and V. Saligrama, "Distributed tracking in multihop sensor networks with communication delays," *IEEE Transactions on Signal Processing*, vol. 55, no. 9, pp. 4656–4668, sep 2007.

[22] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "Distributed Kalman filtering based on consensus strategies," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 622–633, may 2008.

[23] A. Speranzon, C. Fischione, B. Johansson, and K. H. Johansson, "Adaptive distributed estimation over wireless sensor networks with packet losses," in *Proceedings of the IEEE Conference on Decision and Control*, 2007, pp. 5472–5477.

[24] Y. Hu, W. Zhao, and L. Wang, "Vision-based target tracking and collision avoidance for two autonomous robotic fish," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 5, pp. 1401–1410, 2009.

[25] J. Ghommam and F. Mnif, "Coordinated path-following control for a group of underactuated surface vessels," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 3951–3963, 2009.

[26] M. Defoort, T. Floquet, A. Kökösy, and W. Perruquetti, "Sliding-mode formation control for cooperative autonomous mobile robots," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 11, pp. 3944–3953, 2008.

[27] "Pioneer-3AT, http://wiki.ros.org/Robots/AMR˙Pioneer˙Compatible (accessed Nov. 05, 2021)."

[28] "Gazebo, http://gazebosim.org/ (accessed Nov. 05, 2021)."

Janardan Kumar Verma Janardan Kumar Verma received M.Tech degree in Computer Engineering in 2014 from National Institute of Technology, Kurukshetra, India. Currently he is pursuing Ph.D. at the Department of Computer Engineering, National Institute of Technology, Kurukshetra, India. His current research interests include Mobile Sensor Networks, Multi-robot System, Autonomous vehicles, and Artificial Intelligence.

Virender Ranga Virender Ranga has received his Ph.D. degree in 2016 from Computer Engineering Department of National Institute of Technology, Kurukshetra, Haryana, India. He worked as Assistant Professor in the Computer Engineering Department, National Institute of Technology Kurukshetra, India since March 20, 2008 to October 25, 2021. Currently, Dr. Virender Ranga is working as Associate Professor in IT Department of Delhi Technological University, Delhi. He has been published more than 100 research papers in various SCI/SCIE/SCOPUS Journals and reputed International Conferences in the area of Computer Communications and Computer Security. He has been conferred by Young Faculty Award in 2016 for his excellent contributions in the field of Computer Communications. He is a member of editorial board of various reputed journals like Journal of Applied Computer Science Artificial Intelligence, International Journal of Advances in Computer Science and Information Technology (IJACSIT), Circulation in Computer Science (CCS), International Journal of Bio Based and Modern Engineering (IJBBME) and International Journal of Wireless Networks and Broadband Technologies. Currently, he is acting as Ambassador of Asia Region for Bentham Science. He is an active reviewer of many reputed journals of IEEE Transactions, Springer, Elsevier, Talyor Francis, Wiley and InderScience. His research area includes Wireless Sensor Adhoc Networks Security, IoT security, FANET security, SDN security, IoRT etc.