ROBOT ASSISTED REAL WORLD IMPLEMENTATIONS OF SENSOR DEPLOYMENT ALGORITHMS

BY

Mohammed Azharuddin

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER NETWORKS

APRIL 2016

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by MOHAMMED AZHARUDDIN under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE IN COMPUTER NETWORKS.

Thesis Committee

Dr. Uthman Baroudi (Adviser)

Dr. Marwan Abu-Amara (Member)

Dr. Sami El Ferik. (Member)

Dr. Ahmad Almulhem (Department Chairman)

Dr. Salam A. Zummo (Dean of Graduate Studies)

22/5/16

Date

Dedicated

To my loving parents, wife, lovely Hajira and cute Hamza

ACKNOWLEDGEMENTS

"Read! In the name of your Rabb (Cherisher and Sustainer) Who created— created man, out of a leech-like clot: Read! And your Rabb is Most Bountiful Who has taught (the use of) pen. He has taught man that which he knew not."(Qur'an, 96:1-5).

In the Name of Allah, Most Gracious, Most Merciful. Praise be to him, the Cherisher and Sustainer of the worlds. I am grateful to Allah for bestowing me with knowledge, sound health and best people around me to achieve the goals I always dreamt of. May peace and blessings be upon prophet Mohammad (PBUH) and his family.

Firstly, I would like to express my sincere gratitude to my advisor Dr. Uthman Baroudi for the continuous support of my M.Sc. Study and related research, for his patience, motivation, and immense knowledge. His knowledge, expertise, problem solving skills and humbleness inspires me as a role model. His guidance helped me in all the times of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my thesis.

Besides my advisor, I would like to thank thesis committee: Dr. Marwan Abu-Amara and Dr. Sami El Ferik for their insightful comments and encouragement, but also for the hard question which incited me to widen my research from various perspectives.

I would like to acknowledge WISELAB for providing access to all the state of the art technology and research facilities. I am thankful to Dr. Uthman Baroudi for such a great initiative of managing and providing students with such a resource.

I thank my colleagues Jamal, Shaboti and Enam for the support and the sleepless nights we were working together to complete the research, and for all the fun we have had in the last two years.

Last but not the least, I would like to thank my family: my parents and my wife for supporting me and trusting me during tough times of my life. Their deepest love and respect cannot be expressed in words or in writing. May Allah bless them with the best health and make me their supporting pillar as they were to me when needed. Finally, special love to my cute little once Hajira and Hamza.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Full Name : Mohammed Azharuddin

Thesis Title : ROBOT ASSISTED REAL WORLD IMPLEMENTATIONS OF SENSOR DEPLOYMENT ALGORITHMS

Major Field : COMPUTER NETWORKS

Date of Degree : APRIL, 2016

Existing researches in the field of sensor deployment technologies are purely based on simulations & emulations and lacks the real characteristics of the Wireless Sensor Robot Network (WSRN) like heterogeneity of environment, dynamic obstacle distribution, non-availability of GPS, robot's obstacle & sensor detection capabilities etc. This raises a serious doubt on the correctness and reliability of the existing deployment techniques. To overcome these crucial challenges, we designed five novel algorithms, three for single robot scenarios namely ***Basic-SCAN, Opportunistic-SCAN (Opp-SCAN)*** and ***Focused Coverage SCAN (F-SCAN)*** and two for multi-robot scenarios namely ***Cooperative Deployment Mode (CDM)*** and ***Divide & Conquer (D&C)***.

In the first part of the thesis we present Basic SCAN & Opportunistic SCAN algorithms which are used to perform the initial deployment task by a robot. The robot enters the ROI from a known starting point in an unknown environment and executes either Basic SCAN or

Opportunistic SCAN to deploy the sensors. The major difference between the two algorithms is the way in which the deployment takes place. In the section part of thesis, we propose an ***F - Coverage*** version of both Basic and Opportunistic SCAN algorithms which guarantees the added performance and coverage on top of deployment algorithms. Finally, in the third part of the thesis, we propose two novel methods of multi robot team based sensor deployment termed as ***Divide & Conquer (D&C)*** mode and ***Cooperative Deployment (CDM)***. The D&C mode is much simpler with less complexity when compared to CDM which is much intelligent and require more cooperation among the robots for carrying out the task.

This work has been carried at three different levels. In the first phase, we implemented our algorithms on ***MATLAB*** simulation environment and prove that proposed model outperform existing deployment techniques backed by solid mathematical models. In the second phase, we extended our work by implementing them on ***WEBOTS*** (a near to real world robotics tool) and achieved consistent and outperforming results. Finally, we proved our claims by cross-compiling our algorithms and deploying them on real E-puck robots and executing it on the real test bed scenario.

In order to evaluate the performance, we considered key factors like coverage, robot distance, completion time and message overhead and prove that SCAN based deployment outperforms the back tracking deployment in both single and multi-robot scenarios. In terms of coverage, the BASIC SCAN algorithm performs better than all other algorithms, but at the cost of higher total distance travelled. When it comes to distance and message overhead, the Opportunistic SCAN algorithm performs better than all others. In order to ensure the consistency of the proposed model, we have rigorously tested the algorithms in numerous scenarios with dynamically changing obstacle distributions with an optimum confidence interval. The major differentiating

aspect and contribution of this work was to propose novel and standard sensor deployment framework, especially in the areas where there is no GPS and where human reachability is not possible.

ملخص الرسالة

الاسم الكامل: محمد ازهار الدين

عنوان الرسالة: تنفيذ حقيقي لخوارزميات نشر المجسات بإستخدام الربوتات

التخصص: هندسة شبكات

تاريخ الدرجة العلمية: 2016

إن الابحاث الحاليه في مجال نشر المجسات فقط تعتمد على المحاكاه وتفتقر لبعض الخصائص الطبيعيه لشبكة المجسات والربوتات اللاسلكيه مثل التباين في البيئه ووجود المعوقات وعدم توفر خدمة تحديد المواقع (GPS). كل هذا يثير العديد من التساؤلات حول صحة العديد من التقنيات الموجوده حاليا. للتغلب على مثل هذه الصعوبات, نقترح بهذه الدراسه خمسه خوارزميات ثلاثه منهم بإستخدام ربوت واحد واثنين بإستخدام اكثر من ربوت. في الجزء الاول من هذه الرساله سنعرض خوزازمية نشر المجسات الأساسيه (Basic SCAN) وخوارزمية نشر المجسات الانتهازيه (Opportunistic SCAN) واللتين يستخدما اوليا لنشر المجسات بإستخدام ربوت منفرد. يدخل الربوت المنطقه الاساسيه من نقطة بداية معروفه في بيئه غير معروفه وينفذ إما الخوازميه الاساسيه او الانتهازيه لنشر المجسات. الفرق الرئيسي بين هذه الخوارزميات هي في طريقة نشر المجسات. في هذا الجزء من الرساله نقترح استخدام التغطيه المركزه (F- Coverage) لكل من الخوارزميات الاساسيه والانتهازيه والتي تضمن اداء وتغطيه وسع. اخيرا, في الجزء الثالث من الرساله نقترح طريقتين جديدتين لنشر المجسات بإستخدام اكثر من ربوت, الاولي طريقة فرق تسد (Divide & Conquer) والطريقه الثانيه طريقة النشر التعاوني. الطريقه الاولى اكثر سهولة واقل تعقيدا مقارنة بالطريقه الثانيه والتي هي اكثر ذكاء وتتطلب التعاون بين الربوتات.

هذا الشغل تم تنفيذه على ثلاثة مراحل. في المرحله الاولى تم تفيذ الخوارزميات على لغة البرمجه ماتلاب وتم اثبات ان الطرق المقترحه تتغلب على الطرق الحاليه وتم تدعيمها بإثبات رياضي. في المرحله الثانيه تم تطوير العمل بتنفيذه بإستخدام وبيوت (Webots) والتي تحاكي الواقع اكثر وكانت النتائج متناسقه ومتغلبه على الطرق الاخرى. في الاخير تم تنفيذ الطرق المقترحه على ربوتات حقيقه تسمى Epuck.

من اجل تقييم الاداء, اخذنا في الاعتبار عوامل مهمه مثل التغطيه, المسافه, الوقت والرسائل لاثبات ان الطريقه الاساسيه افضل من طريقه نشر الربوتات بإستخدام التتبع الخلفي في كل من طريقة ربوت واحد او اكثر من ربوت. بالنسبة للتغطيه, الطريقه

الاساسيه تغلبت على الطرق الاخرى ولكن التكلفه من ناحية المسافه المقطوعه. بالنسبة للرسائل والمسافة المقطوعه, الطريقه الانتهازيه تغلبت على الطرق الاخرى. من اجل ضمان التناسق بين الطرق المقترحه, تم تنفيذ الخوارزميات بالعديد من السيناريوهات وتغير طريقة نشر المعوقات. الميزه الاهم في هذا الشغل هي اقتراح اطار عام للأخذ بالاعتبار السيناريوهات التي لا يوجد فيها نظام تحديد المواقع والتي يصعب ايضا وصول الانسان اليها.

CHAPTER 1

INTRODUCTION

In recent times, the usage of sensor networks has drastically increased in each and every aspect of human lives and there is plethora of successful applications currently deployed on such networks. Due to the tremendous development in this field, it has a great potential to be applied across multiple domains for technological improvements.

A sensor is a device that usually has very limited capacity, size and communication range, which are used to detect or measures a physical property and records, indicates, or otherwise responds to certain events [1]. Sensors can be both static and autonomous depending on the application domain [2]. Hybrid sensor networks [3], [4], [5] can be more energy efficient where static sensors are only responsible for sensing data from the environment and autonomous devices are responsible for the relocation [6] of those sensors. Here autonomous devices refer to the actuators or the robotic agents who are resource rich in terms of battery power, movement features etc. On the other hand, static wireless sensors are very tiny in shape and resource constraint. Thus, these small sensors deplete energy quickly if they have to perform movement, maintenance and repair by themselves, which is not feasible for real life application scenario. Different versions and configurations of hybrid sensor networks have been formulated to carry out the desired outcome with optimum results.

1.1. Wireless Ad Hoc and Sensor Networks

Wireless sensor network (WSN) is a large-scale distribution of wireless nodes having a very limited source of energy, storage, and processing efficiency. One of the most significant aspects of WSN is its capability to create a collaborative network without prior infrastructure setup i.e. The mobile nodes may or may not have the information of the whole network before joining in the network for information processing.

Each and every node of a network can act as a sender, receiver or forwarder. This way, a sensor can communicate with all other sensors through multi-hop forwarding of information across the network, acting as an infrastructure-less network behaving dynamically based on the optimum network parameters like best path, minimum energy consumption, minimum message overhead etc. Figure 1.1 illustrates a sample of real world examples of WSN which consist of disparate application domains, integrated over a centralized network.

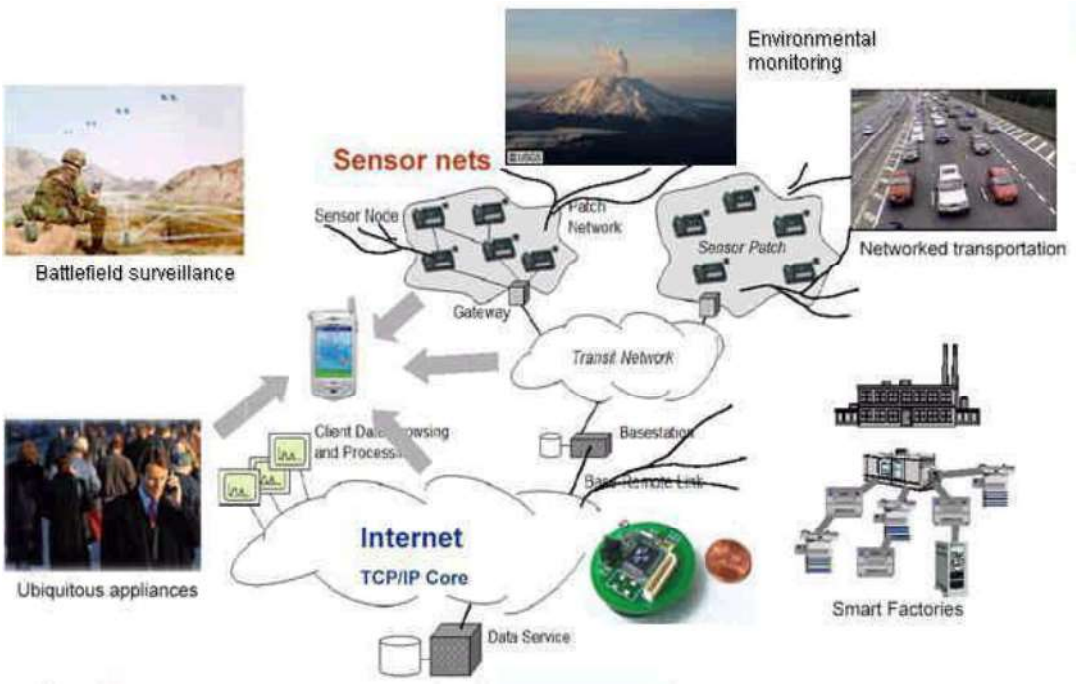

Figure 1.1. An example of real world, heterogeneous and distributed wireless sensor network (WSN).

These networks can be applied in numerous real time and mission critical applications, some examples of such networks are:

- Event monitoring & detection [7]
- Health monitoring [8]
- Temperature sensing
- Humidity sensing.
- Oil and gas exploration.
- Toxic gas emission detection. [9]
- Traffic control.
- Manufacturing and plant automation.
- Military surveillance etc. [10] to provide critical surveillance and data capture in a region of interest (ROI). Figure 1.2 demonstrates a sample full featured WSN based battle field with a mobile actuator and devices.

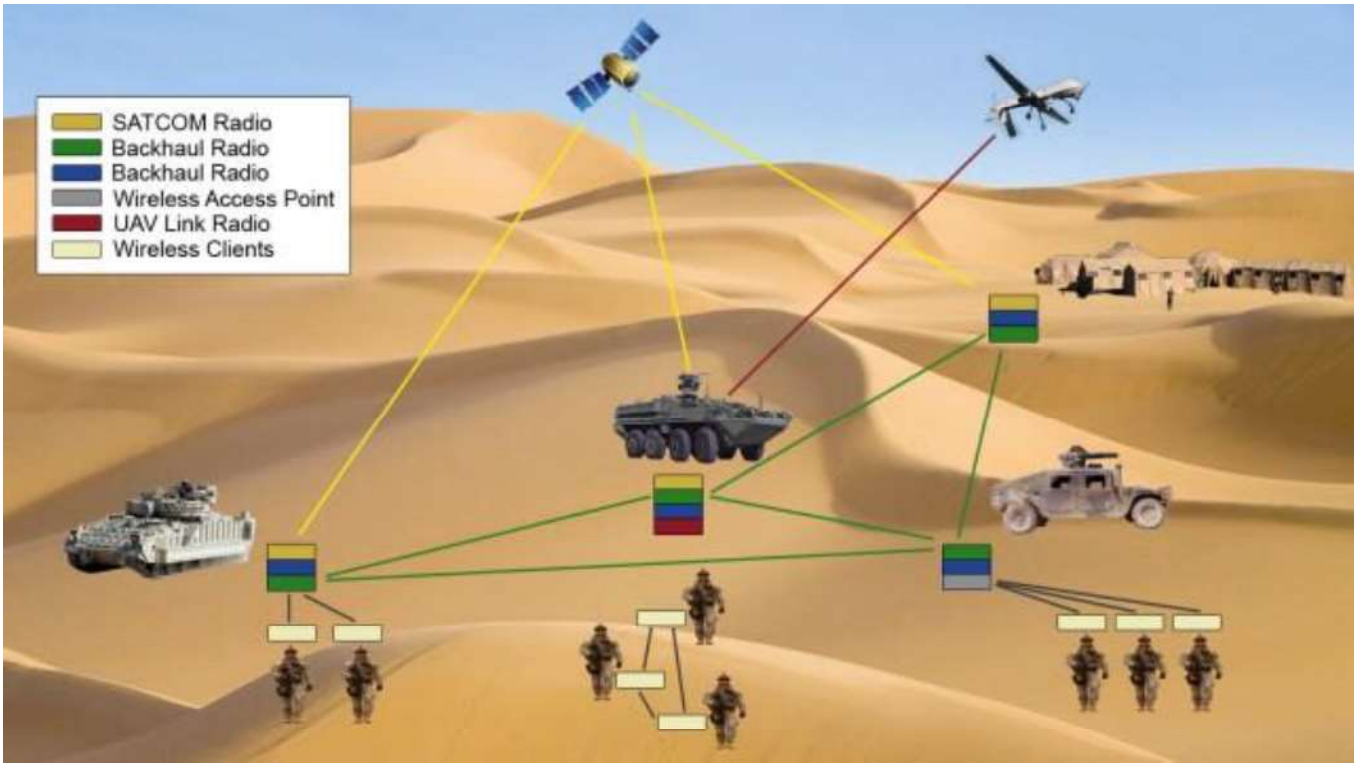

Figure 1.2. Wireless sensor network application domain in military communication showing modern battlefield.[10]

Another interesting research area is the usage of mobile robots in mission critical applications like search and rescue operations which are inherently dangerous and often impossible for humans to carry out. The main requirement of such application is to be reliable, robust, secure and with optimized performance.

Wireless Sensor Robot Network (WSRN) on the other hand is a special type of WSN in which resource-rich, mobile robots are responsible for deploying and managing the resource constrained, stationary sensors in which actions are collaboratively executed by the robotic agents on the basis of the information received by the deployed, standalone sensing units [11]. These kinds of networks have important deployment scope in mission critical applications like

search and rescue operations are used for disaster relief plans where human accessibility is impossible. In other words, A WSRN could be thought of as a distributed control system that needs to timely react to sensor information with an appropriate action [11] . Figure 1.3 shows an example of underwater WSRN [2] [7] in which nodes are networked optically, acoustically and using radio.

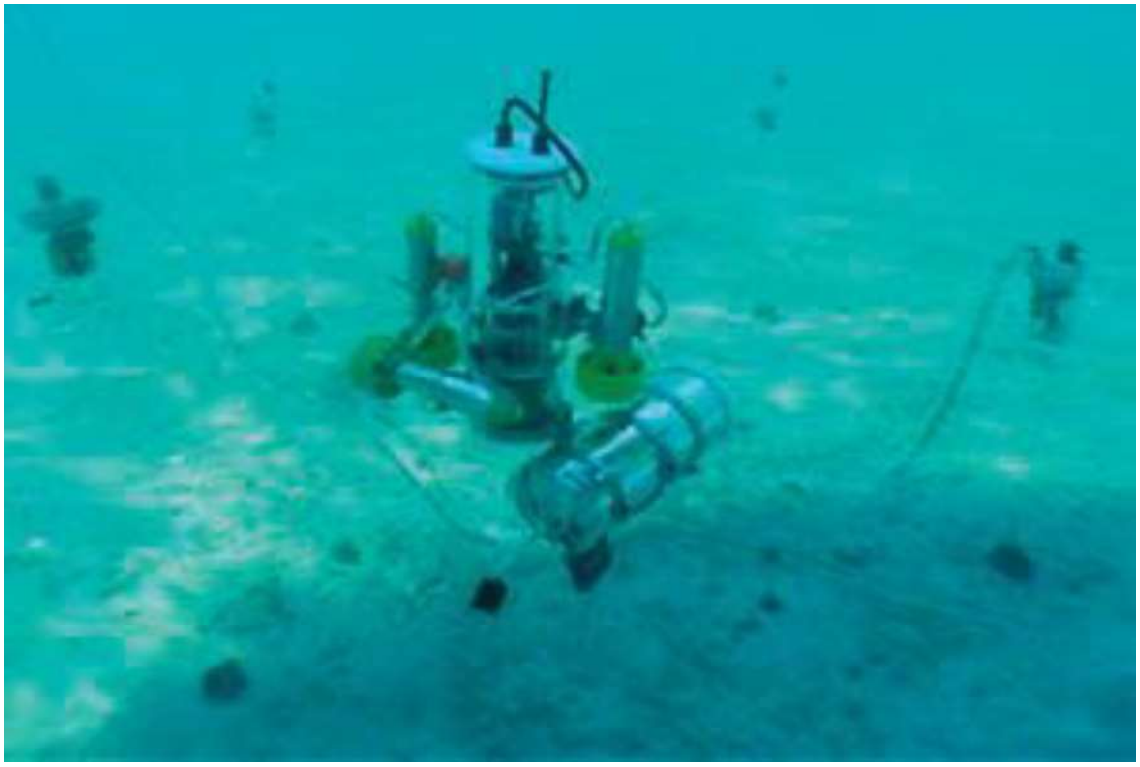

Figure 1.3. A sensor network deployed in Moorea, French Polynesia. Copyright © DRL, MIT. [12]

Some of the pivotal issues in WSRN, which are considered to be an open area of research, but not limited to sensor deployment are real-time inter-robot task allocation, distributed sensor-robot coordination mechanisms [13], Remote monitoring middleware [14] and quality of service (QoS) preservation [14]. This work is focused towards the sensor deployment in a region of interest (ROI), which is first and foremost aspect of WSRN.

1.2. MOTIVATION

The main motivation of this work is to design and implement a robot assisted sensor deployment algorithms which should be robust, reliable, GPS-less and should be based on real world implementation. Our work is motivated by seven major observations from different contemporary research in sensor deployment area, details of which are provided in section 1.2.1 through 1.2.7.

1.2.1. REAL WORLD IMPLEMENTATION

Numerous existing researches in the field of sensor deployment technologies are purely based on simulations & emulations and lacks the real characteristics of the Wireless Sensor Robot Network (WSRN) like heterogeneity of environment, dynamic obstacle distribution, non-availability of GPS, robot's obstacle & sensor detection capabilities etc. This raises a serious doubt on the correctness and reliability of the algorithms. These crucial issues motivated us to design a novel and standard development framework based on a real robot.

1.2.2. WHERE HUMAN REACHABILITY IS IMPOSSIBLE:

There are places where human reachability can be hazardous due to adverse environmental condition, toxic gas explosion hazards from industries, unused and abandoned places where toxic chemical properties may remain for a longer period of time. These places can be explored using robotic agents to avoid adverse effects on humans. Thus, one of the major requirements for this thesis was to propose a novel technique which can be implemented in robots to accomplish such tasks efficiently.

1.2.3. GPS-LESS ENVIRONMENT:

From literature survey of existing researches we observed that most of the sensor deployment techniques are based on the usage of GPS location services. Although GPS is most widely used for object tracking, location determination, geographic information collection etc., It is hard to access inside the buildings. Thus, if we want to deploy sensors inside buildings using mobile robots, the task becomes quite impossible. Additionally, the location calculation would add extra overhead for the robots.

1.2.4. MAXIMIZED COVERAGE:

Coverage in wireless sensor networks is usually defined as a measure of how well and for how long the sensors are able to observe the physical space. Coverage and connectivity are two of the most fundamental issues in WSNs, which have a great impact on the performance of WSNs. Hence an outmost consideration has to be given to this important aspect. Additionally, the maximum coverage is desirable from every deployment algorithm otherwise the uncovered areas remain abandoned and no information can be collected from those locations. The task becomes more sophisticated when the desired deployment area comes with obstacles inside of it.

1.2.5. SCALABILITY:

Scalable algorithm has to be executed on the robot so that it can provide similar performance for increased areas and numbers of obstacles. The issue of scalability has to be considered by adding opportunistic behavior on the model. Scalability is a must have requirement as the robots do not know the obstacle position before facing them.

1.2.6. ROBOT DEADLOCK PREVENTION:

As per definition, Deadlock is a problem that can exist when a group of processes compete for access to fixed resources. Deadlock can exist if and only if four conditions hold simultaneously:

- **Mutual exclusion:** at least one resource must be held in a non-sharable mode.
- **Hold and Wait:** there must be a process holding one resource and waiting for another.
- **No preemption:** resources cannot be preempted.
- **Circular wait:** there must exist a set of processes [P_1,P_2,P_3… P_n] such that P_1 is waiting for P_2,P_2 for P_3, and so on and P_n waits for P_1….

There are three ways to deal with deadlock, they are:

- **Deadlock Prevention & Avoidance:** Ensure that the system will never enter a deadlock state
- **Deadlock Detection & Recovery:** Detect that a deadlock has occurred and recover
- **Deadlock Ignorance:** Pretend that deadlocks will never occur.

The first option is the best one to resolve deadlock, and to do that we must ensure that at least one of the condition among Mutual exclusion, hold & wait, No preemption and circular wait does not happen in the application.

To tackle any dead end while the robot moves. In [11], the author introduced a Backtrack-based algorithm for robots which keeps track of pointer and uses that when any obstacle is faced. There are several situations when this back pointer is removed by other robots. This pointer removal can create dead end of the current robot that owned the back pointer previously. A technique has to be designed which ensures that this sort of scenario never appears by using some kind of opportunistic approach in which the robot tries to find any available route before going back to the start position on its route when any obstacle is faced.

1.2.7. MULTI-ROBOT COORDINATION:

The problem of sensor deployment becomes more complex when it comes to multi-robot deployment. Numerous existing research works have targeted this area but very few presented an end-to-end solution to this problem. We have extended our single robot algorithms and propose two novel approaches of sensor deployment with a team of robots which provide end-to-end deployment procedure.

All the above stated issues were closely studied and resolved by designing and implementing three novel robot assisted sensor deployment algorithms on real robots. These algorithms are based on similar robot deployment strategy, but differ in the way they carry out the deployment tasks. These algorithms could be a good fit in different application scenarios.

1.3. THESIS CONTRIBUTIONS:

In this work, we propose a novel solution to the problem of wireless sensor deployment in hazardous areas where human reachability is impossible. Below are four major contributions of the thesis:

- Design and propose five novel robot assisted sensor deployment algorithms, three of them for single robot scenarios and two for multi robot scenarios.
- Propose a novel and real world development framework for robot assisted sensor deployment in GPS-less and Obstacle unaware region of interest (ROI).
- Implementation of Scalable Coordinated Actuator Network (SCAN) and Back Tracking Deployment (BTD) sensor deployment using a real E-puck robot.

- An extensive work on a list of various real world challenges during the experiment, including the environmental conditions, obstacle and boundary problems etc. And their solutions.
- Application of algorithms in multi robot scenarios and their outcomes.
- Performance evaluation of algorithms in terms of coverage, robot distance travelled, message overhead & execution time and to provide recommendations for various critical application scenarios.

CHAPTER 2

BACKGROUND

This chapter is geared towards describing the comprehensive related work done for this thesis. For the better categorization of related work we have divided this chapter into two sections, in the first section we present complete background of robot assisted sensor deployment algorithms and a broad classification of various research works in this field. In the second section we present related work for robot assisted real world implementations. We also present the major advantages and contributions of our work compared to the previous works.

2.1. Related work for existing WSN deployment algorithms

The task of sensor deployment in a target field can be broadly classified into four categories, they are self-deployment, deterministic deployment, random deployment and incremental deployment. [16]

The random deployment technique is used when the environment to be covered is completely unknown and the QoS for coverage guarantee is not of key importance. The results of random deployment are almost always not effective and results in inefficient network coverage. Deterministic deployment is used in known environment and which is physical easily accessible [17]. Generally, such deployments are preplanned and the exact locations of nodes are known before the execution of the algorithm. Incremental deployment techniques are the most recent and has great research potential. In incremental deployment, the nodes are incremental deployed depending on the various QoS parameters of an application [18]. This kind of networks are

generally scalable, resilient and reliable as they have capabilities to easily be restored [19]. This work is aimed at designing algorithms which are based on incremental deployment strategy, but with an added hybrid flavor of random deployment.

Sensor deployment algorithms can be further classified into five different categories, they are:

- *Virtual Force Based Approach,* [17]
- *Movement Assisted Approach,* [17]
- *Computational Geometry Based, and* [20]
- *Pattern Based Approach and* [21]

The virtual force based approach the network is modeled based on the physics concept of attraction and repulsion depending on the distance between the two actuators. The threshold distance between the nodes would decide the uniform deployment of sensors in the network. [20] [21] [22] [23] Are some of the early approaches based on virtual force deployment.

Movement assisted sensor network deployment technique has nodes / actuators capable of moving around the region and have mobile capabilities. These movements are applied for sensor placement. The basic requirement for movement assisted deployment technique is to handle the obstacle avoidance and deadlock prevention conditions.

In Computational geometry technique the basic problem is to design and implement an algorithm in terms of basic mathematical points, lines, equations and geometrical figures. Two of the most commonly used data types in such technique is Voronoi Diagram and Delaunay Triangulation [24]

Finally, in Pattern based approaches are used to design the deployment technique as patterns like Triangle, Rectangle, Diamond, hexagon etc. This problem becomes coverage optimization problem based on tilling and tessellations. [25]

2.1.1. Sensor Deployment Algorithms: Taxonomy

The figure represents and hierarchical form of taxonomy of the major sensor deployment algorithm.

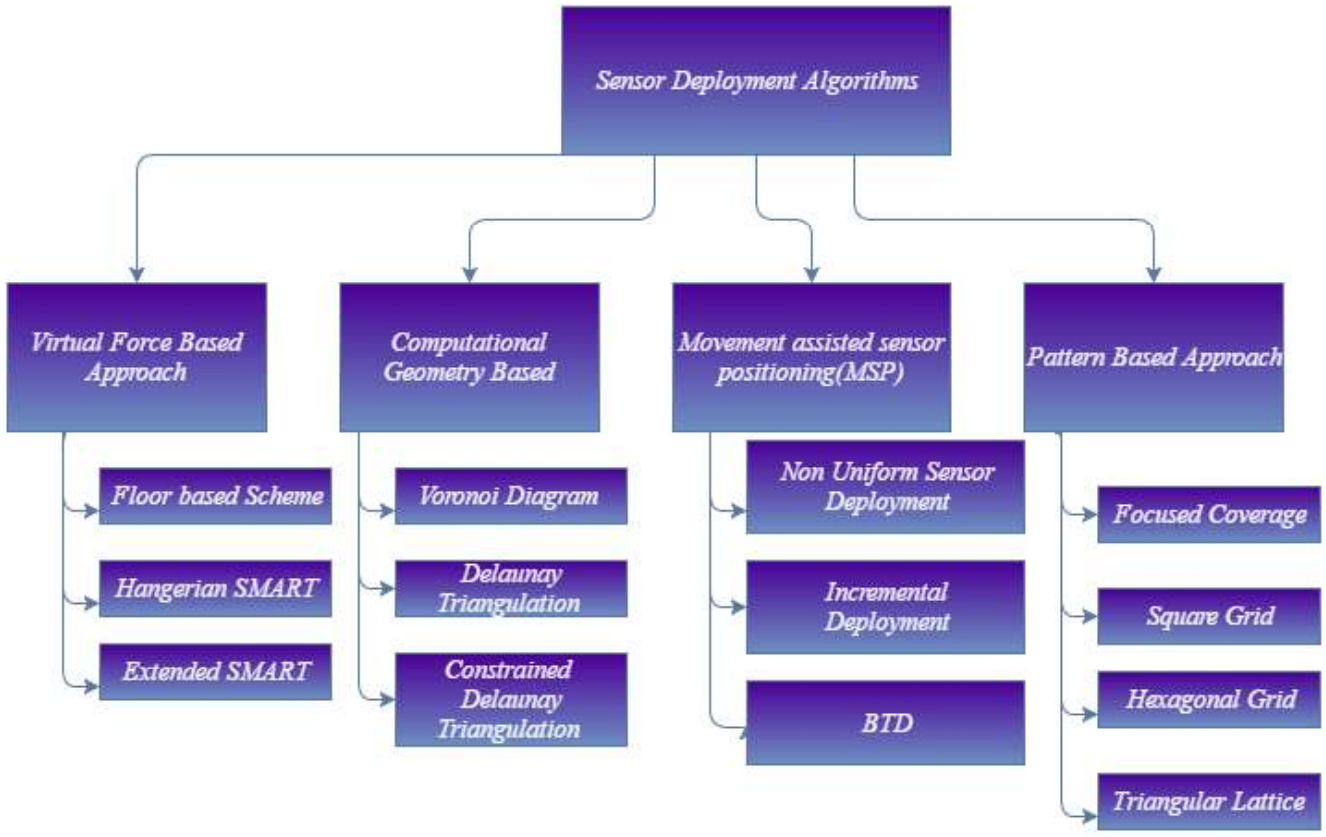

Figure 2.1: Hierarchical taxonomy of major sensor deployment algorithms.

Ref #	Deployment Technique	Algorithm Proposed	Distributed / Centralized	Issues Handled	Drawbacks
[18]	Incremental Deployment	Largest Empty Circle	Centralized	Network Lifetime, Energy Consumption	Obstacle Adaptability
[19]	Potential Field Approach	Clustering & Merging	Distributed & Centralized	Coverage, Sensor Movement	Obstacle Adaptability, Connectivity
[26]	Sensor Relocation	Local Detection Diagram(LDD)	Distributed	Coverage, Sensor Movement	Obstacle Adaptability, Connectivity
[20]	Virtual Force	Floor based Scheme	Distributed	Coverage, Sensor Movement, Connectivity	Computational Overhead
[27]	Coverage Optimization	Particle Swarm Optimization(PSO)	Centralized	Coverage	Obstacle Adaptability, Connectivity
[28]	Robot Deployment	Node Placement, Spiral Placement	Centralized	Coverage, Obstacle Adaptability,	Connectivity
[21]	Virtual Force, Graph Theory	Delaunay triangle graph (RDTG)	Centralized	Connectivity, Coverage	Obstacle Adaptability
[29]	Grid based Delaunay Triangulation	DT Score	Centralized	Coverage, Obstacle Adaptability,	n/w lifetime
[30]	Relocation, Virtual Force	------	Distributed	Coverage, Connectivity	NA
[31]	Non Uniform Sensor	Movement assisted sensor	Centralized	n/w Lifetime	Connectivity and

Ref #	Deployment Technique	Algorithm Proposed	Distributed / Centralized	Issues Handled	Drawbacks
	Deployment	positioning(MSP)			Coverage
[32]	Optimization Problem	----	Centralized	Power Consumption, Fault Tolerance	Coverage and Connectivity
[33]	Intelligent Mobile Sensor	Voronoi Diagram	Distributed	Energy Efficiency, Sensor Movement	Obstacle adaptability
[34]	Incremental Deployment	-	Distributed	Coverage	NA
[35]	Electrostatic Problem	-	NA	Connectivity	NA
[36]	Mobility for Improving Coverage	-	Distributed	Coverage	NA
[37]	Mobility for Improving Coverage	-	-	Coverage,	NA
[22]	Virtual Force	Hangerian, SMART, Extended SMART	Centralized	Coverage, Sensor Movement	Obstacle adaptability
[38]	Sensor	Grid Quorum	Distributed	Fault Tolerance	NA

Ref #	Deployment Technique	Algorithm Proposed	Distributed / Centralized	Issues Handled	Drawbacks
	Relocation				
[39]	Non Uniform Sensor Distribution	MAND	Centralized	Energy Efficiency	Coverage
[40]	Node Failure Detection	MST, SRN	Distributed	Fault Tolerance, Power Consumption, Sensor Movement	Coverage
[41]	Double Mobility	Event Driven, Dominating Set Maintenance	Distributed	Coverage, n/w Lifetime	NA
[42]	Localization without GPS enabled Sensor	MontoCorlo	Distributed	Localization	Coverage
[43]	Node Discovery	Machine Learning	Distributed	NA	Coverage
[44]	GPS less Localization	-	Distributed	Coverage	NA
[23]	Movement Assisted, Virtual Force	-	-	Coverage	Obstacle adaptability
[45]	Cellular Learning	-	Distributed	Coverage	Connectivity

Ref #	Deployment Technique	Algorithm Proposed	Distributed / Centralized	Issues Handled	Drawbacks
	Automata				
[46]	Landmark Node	-	-	Connectivity	Coverage
[47]	Dynamic Coverage		Distributed	Coverage	NA
[48]	Constrained Coverage	-	Distributed	Coverage, Connectivity	NA
[49]	Focused Coverage	GA, GRC	Distributed	Coverage	NA
[50]	Fluid Dynamics	-	-	Coverage	NA

Table 2.1. A generic classification of Sensor Deployment Algorithms.

2.2. Related work for SCAN deployment Algorithms:

The challenging task of sensor deployment in an environment where human intervention is impossible completely relies on the technique used by robotic swarms to carry out this task. This task of placing senor nodes by mobile actuators has been considered as a major strategic research work in WSRN. [51] Thus, algorithms which are executed on the robots plays a crucial role in the success or failure of technique. The sensor deployment algorithms are broadly classified into two major types, they are BLANKET coverage and FOCUSED coverage. The vast majority of research works focus on BLANKET coverage technique, but very recently some novel research has been produced under a new category called FOCUSED coverage [49]. The objective of BLANKET coverage is to lay the sensors in a way that considers the entire ROI as a target for

increasing network connectivity without regards to any particular point of interest, whereas, FOCUSED coverage provides monitoring near and around a point of interest (POI) and its vicinity. This class of algorithms, where sensors are deployed to monitor around a strategic area called Point of Interest (POI). This area around this POI relatively has higher priority than area away from POI. F-Coverage algorithms uses the concept of mathematical tessellation.

A tessellation of a flat surface is the tiling of a plane using one or more geometric shapes, called tiles, with no overlaps and no gaps. In mathematics, tessellations can be generalized to higher dimensions and a variety of geometries. One of the F-coverage deployment techniques can be applied using Equilateral triangle tessellation (TT) is a planar graph composed of congruent equilateral triangles. Figure 2.2 shows a sample ROI mapped with hexagonal tessellation and the point F represents the POI.

The goal is to develop a carrier-based sensor placement algorithm that yields a sensor network surrounding F in hexagonal layers and with an equilateral triangle tessellation (TT) layout of nodal separation$\sqrt{3}R_s$.

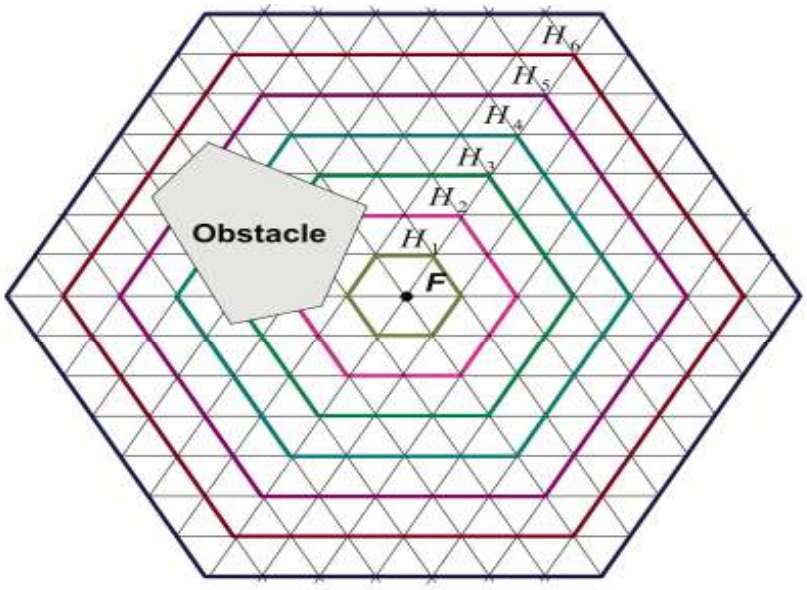

Figure 2.2: F-coverage problem envisioned as a vertex coverage problem

In this work we propose a hybrid class of algorithms, which combines the features of both the techniques and provides better coverage and much efficient results.

Back-Tracking based sensor deployment [52] by a robot team is one of the most novel sensor deployment techniques and was one of the only works which provided a methodology other than the typical SLAM based technique. The authors provided the localization and deployment process in a single step rather than in multiple steps. As per BTD algorithm, the entire ROI is mapped to a graph and a sensor is placed at each vertex of the graph. To carry out this task, a robot carries an unlimited number of sensors which practically is not feasible.

A team of robots starts placing the sensors independently, asynchronously and without any coordination. The trajectory of each robot is predefined by the rank of its direction. Once a robot drops a sensor at a point it stores a vector $<\boldsymbol{Position, Robot_{ID,} Sequence\#, Color, Back_{PTR>}}$ which represents in the respective sequence, the ***position*** of the sensor at which it was dropped, the ID of the robot which placed the sensor, ***Sequence number*** of the sensor in incremental order, ***Color*** of the node which can be represented as either black or white. A node declares itself as ***"white"*** if there is at least one adjacent node empty otherwise it declares itself as ***"black",*** finally the $back_{PTR}$ pointer is an indicator of the lowest ID node with white color in the robots backward trajectory.

There are various shortcomings of this work, but three of the most important of them are

1. The unrealistic assumption that a robot can carry an infinite number of sensors for placement.
2. Robot deadlock condition if it loses the back pointer. This is a very common issue in case of multi robot implementation if both the robot tries to reach a white node and due to

improper coordination, if the back pointer is lost then the entire application ends up in deadlock.

3. No real world implementation and evaluation of this work. Our work is a major enhancement over BTD with a significant addition to it and is backed with solid proof of better performance.

Yuanteng and MattW proposed a novel approach called ***STARS: Static Relays for Multi-Robot Real-time search and monitoring*** [53] in which robots compute in offline mode the required number of reference points where anchor nodes are required and number of relay nodes required for the information to be transferred. The relay nodes are responsible for transferring the captured data to base station. A robot makes movement based on the information obtained at the base station from the relay nodes. Each robot would be responsible for a set of sensing nodes and the relay nodes. The **STAR** algorithm divides the main task of area coverage into a set of four smaller sub-tasks Sensing Point Identification, Relay Point Identification, Assignment of Precedence and finally the tour generation. This novel work has three major drawbacks. First and foremost, STARS algorithm requires the prior knowledge of the obstacle location and distribution, this is practically a major shortcoming of any environment localization mapping algorithms, as in environments which human reachability is infeasible, it's impossible to know the position of obstacles. Secondly, the number of static nodes carried by a robot will be limited and would be limited to cover the entire region. Finally, the propose solution is neither fault aware of unexpected behavior, nor is reliable to be executed in practical scenarios.

In [54] authors proposed a protocol named Carrier-Based focused formation (CBFCF) which is based on the concept of focused coverage algorithm. The algorithm consists of two major tasks, reactive advertising routine (RAR) and iterative sensor placements (ISP). These tasks are

executed in parallel manner. The RAR executes on each sensor based on the requirement, and publishes the empty neighbors around a node, whereas, ISP runs in de-centralized manner on both sensors and robots in multiple iterations. The major significance of this work is that it was the first and innovative work based on F-coverage algorithms, also it is a purely localized approach and is an obstacle aware technique.

In [55] and [56] authors focused on mobile sensor networks with a significant luxurious wastage of network equipment's and locomotive power and hence are not much of usage.

One of the most significant researches in the field of real robot implementation was done by [57]. This work was done as a part of the DARPA software for Distributed Robotics (SDR) program. In this work, researchers have developed a multi-robot system for deploying a team of 80 robots, for achieving the task of localization and sensor deployment. The overall task was divided into two phases, the first phase was to explore the region of interest (ROI) using the SLAM (Simultaneous localization And Mapping) of robot's local estimation and global estimations. On the other hand, the second phase involves sensor deployment in which robots perform pre-planning step of determining the desired sensor deployment position which are further used for actual sensor deployment by the of chain of robots which are led by a leader robot. This work was an early real world implementation of robots which proved to be a very costly and resource rich technique and involves huge computation for localization technique. Moreover, this technique has segregated the task of localization and deployment. Our work targets the same problem of sensor deployment, but the problem is achieved using the robot's adaptive capability to simultaneously perform localization as well as deployment at the same time. Another crucial difference between the previous works and our work is the solution to the deadlock problems which occurs if the back pointers are lost.

In [22] the authors have proposed a SCAN based technique of sensor deployment methods. In this work authors proposed Hungarian-algorithm-based centralized and optimal solution. The basic version of their work was named as SMART which was enhanced with a newer version named Extended SMART. The basic behind this work is to apply the classic Hungarian method to sensor deployment technique to achieve movement-assisted sensor deployment using mobile robots.

The basic idea behind this work is to divide the sensor network into an $n \times n$ 2D mesh of clusters, each cluster is denoted by a square area and controlled by a cluster head. The role of the cluster can be switched between different nodes and the task of cluster head is to communicate with the cluster head of another cluster. In this work, the authors have performed an extensive simulation and proved that their algorithms provide optimal network coverage and a decent sensor deployment technique.

2.3. Related work for SLAM & Multi-Robot implementations:

Simultaneous Localization and Mapping (SLAM) is a technique by which a mobile robot is released in a completely unaware area assigned with the task of incrementally building a map of this environment by continuously mapping its data with the information currently obtained. The commencing work on SLAM was successfully conducted in the year 1995 by [58]. This work was highly appreciated and was a kick-off for consecutive works by [59] [60] in which essential theory on SLAM convergence and initial results were developed. Several original works by SLAM done by [61] [62] [63] [64] and [65] targeting the experimental implementation both indoor, outdoor and under water environments. The SLAM problem can be solved in various ways one of them is Probabilistic SLAM in which the probabilistic distribution equation. [66] [67] Proposed approaches for probabilistic SLAM techniques. Extended Kalman Filter SLAM

(EKF-SLAM) is another solution for solving the problem of localization [68] proposed a solution for EKF-SLAM problem.

The FastSLAM algorithm proposed by [69] was a new approach to solving the SLAM problem using recursive probabilistic theory. SLAM algorithm is still studied extensively and a relatively recent work by [58] has tackled similar problem of robot assisted localization and sensor deployment using the SLAM localization technique. SLAM has a wide and successful implementations in indoor, outdoor, a real and under water implementations. [70] [71] [72] Are some of the successful indoor implementations, [73] is an outdoor implementation, [74] for aerial implementation, and subareas in [63] [75] [76].

Another approach for sensor deployment was achieved using an autonomous agent named AVATAR [77]. In this paper the authors have implemented complex sensor network deployment method using autonomous flying robots. This work presents a detailed analysis of deployment algorithm and experimental studies supported by data collected from various trials. They designed a helicopter for sensor deployment in a grass field marked as 7 x 7 grids. They had their work compared in both manual and autonomous modes. [77]. This work was outcome of integration of three projects from three different labs (CSIRO, USC and Dartmouth). This outcome of this work was to come up with a desired network topology and 1. Autonomous sensor deployment using aerial robots 2. Seamless integration Ground-Ground & Ground-Air equipment's and 3. Finally, Capture the results and utilize it to repair the network connectivity.

A similar work to AVATAR was previously done by [78] [79] in which two flying robots were used to complete tasks allocated to them. Some other works include [80] [81] which were also couple of early works for aerial robotics. AVATAR was an extension of this work.

Another significant work in this area is [82] in which authors have proposed information gathering system using mobile actuators and WSN in underground conditions which resembles post-earth quake conditions. This work has used the concept of RSSI technique to deploy and maintain the sensor network.

MARVIN [83] was another project which aims at providing the solution to monitoring of a region of interest (ROI) using the UAV (unmanned aerial vehicles). The authors proposed a technique and provided a framework for localization and coordination of UAV's, static sensors and mobile nodes in GPS denied environment. The result of this experiment suggests that there were a lot of aberration in the communication among the nodes.

[84] Is another early and novel work on real world implementation of asynchronous and heterogeneous controlling of mobile robots. The outcome of this work was a control architecture named ATLANTIS, which combines a reactive control mechanism with a traditional planning system.

Very recently [85] has been presented which is a significant work providing a framework of cooperative multi-robot navigation, exploration, mapping and object detection with robot operating system (ROS) called WAMbot. This work provides a large-scale mapping of the interior region of 500 x 500m global maps using approximately 20 robots in real-time. Yet again, the framework works in two steps, one for localization another on deployment.

Batalin and Sukhatme proposed the Least Recently Visited (LRV) algorithm [86], this algorithm targets at problem of coverage, exploration and sensor deployment, where already deployed sensors give recommendations to robots for the direction to continue sensor placement. The

algorithm produces full sensing coverage in a long run; but it uses excessive movements to explore the ROI, and does not even offer termination. The main concept behind this work is to have two properties. 1) LRV algorithm is supposed to be complete on graphs and 2) LRV algorithm is optimal on trees.

Chang et al. Designed a novel algorithm for sensor deployment in unpredictable regions using the concept of the spiral movement of robots [87] [28]. This work proves to be of good contribution as it focused on deployment as well as the energy efficiency of the robots and sensors. They presented a Snakelike Deployment (SLD) algorithm. However, despite the claim made, this algorithm cannot guarantee full area coverage. This motivates us to develop an algorithm that does guarantees coverage, terminates, and remains efficient in terms of robot movements.

In other related work [88], Santpal S. And Krishnendu proposed two algorithms for efficient placement of sensors in a sensor field. The main focus of their work was to come up with an algorithm which would optimize the process of placing the minimum number of sensors with maximum coverage. They designed a sensor detection model based on probabilistic mathematics according to which the probability of detection of a target by a sensor has exponential relation with the distance between the targets to the sensor.

Howard et al. proposed an incremental self-deployment strategy where a robot deploys sensors in an unknown environment one at a time and incrementally retrieves the next sensor location information from a central controller based on previously deployed sensors. [34] This approach is very expensive and non-robust as the entire processing for the sensor position is done by the

centralized controllers which would be highly computationally expensive. Also, if the central controller goes down then the entire ROI would be disconnected.

Robot assisted sensor relocation is another important aspect which needs to be considered when it comes to WSRN. Sensor relocation refers to the task of replacing failed sensors with some other potentially unimportant sensors or redundant once without impacting or with minimal impact of existing network topology. [51] [38] [89] [90] [91] are some of the various papers which designed protocols for mobile sensor networks (MSN) but only three works till date has designed and solved the problem of robot assisted sensor relocation and they are [92] [93] [94]. The deciding factor in the aforementioned research work to be operational is to divide the wireless network into active and passive modes.

Upon failure of active nodes, the passive nodes can be replaced by the failed active nodes. One of the key factors in sensor relocation algorithms is to minimize the total cost of coverage repair task and keep a balance between network coverage and the cost of network restoration. This requirement of maintaining a balance between coverage and cost gave a new direction to a newer problem in [95] called Carrier-Based Coverage Repair (CBCR).

Very recently, Sharifi et al. [96] Presented an approach for recharging a failed sensor using actor fleet. The concept behind this work is that, a sensor notifies its energy and position status which includes their ID, location, current power, power ratio and interest coefficient to nearby anchor point, based on which the anchor takes appropriate action to help recharge the needy node.

Our current format of work does not consider the task of network maintenance in scope. However, we believe that the network maintenance and recovery could be an excellent future extension to the existing work.

CHAPTER 3

PROPOSED MODEL

Our proposed model overcomes the key challenges in the field of robot-assisted sensor deployment in hazardous areas. The key difference between this work and previous works is the exploitation of non-GPS-based algorithms. This chapter is divided into three parts, in the first part we present the design and architecture of the Basic SCAN, Opportunistic SCAN and Focused Coverage algorithms. We then present the second part of the chapter, in which we present the basic concepts like sensor deployment, coverage, mathematical model and a list of assumptions which are essential to understand, before presenting the experimental setup & performance evaluation. In the final section of this chapter, we describe some application scenarios with detailed explanations of the behavior of algorithms in such circumstances.

3.1. Basic SCAN, Opportunistic SCAN & F-Coverage algorithms

As mentioned in earlier chapter, the major contribution of this thesis was to propose five novel algorithms, three of them for single robot scenarios and two of them for multi robot scenarios. In this section we will introduce three novel algorithms for single robot, namely BASIC SCAN, OPPOTUNISTIC SCAN and Focused-Coverage (F-Coverage) SCAN. The other two algorithms for multi robot scenarios would be discussed in chapter 7.

We will begin with the first part of this chapter in which we will present the complete design and architecture of BASIC, OPPORTUNISTIC SCAN & F-Coverage algorithms. These algorithms

were designed to ensure reliable sensor deployment, which overcomes key challenges in the field of robot-assisted sensor deployment in hazardous areas.

The Basic SCAN & Opportunistic SCAN algorithms are used to perform the initial deployment task by a robot. The robot enters the ROI from a known starting point in an unknown environment and executes either Basic SCAN or Opportunistic SCAN to deploy the sensors. The major difference between the two algorithms is the way in which the deployment takes place.

We also propose F-Coverage version of both Basic and Opportunistic SCAN algorithms which guarantees the added performance and coverage on top of deployment algorithms.

3.1.1. DESIGN OF BASIC SCAN & BASIC SCAN WITH F-COVERAGE

This section provides brief overview of both basic and Opportunistic SCAN algorithms, followed by a detailed section where we would present the complete architecture of the algorithms. Firstly, we begin by explaining the basic SCAN algorithm with an actual scenario, followed by detailing out the F-Coverage version of Basic SCAN algorithm. Figure 3.1 shows a theoretical view of a sample ROI where the sensor deployment is to be carried out. For the reason of simple illustration, we have placed three obstacles in ROI. However, the algorithms are designed such that they are scalable to complex obstacle distribution.

3.1.1.1. BASIC SCAN Algorithm:

The Basic SCAN algorithm is the first novel algorithm we have proposed. This is the most basic version of deployment algorithms, ensuring optimal sensor deployment. However, it requires much more computing resources and completion time when compared to the Opportunistic version.

Let us consider a region of interest of dimension 200 x 200 m. A robot completely unaware of the environment enters the ROI from position START (0, 0) and reaches till (200, 0). When it reaches (200, 0), it detects the boundary using its built-in camera/acoustic signals. As soon as the mobile robot detects any obstacle or boundary, it get back to its starting location along its backward direction. It also adds/subtracts (adding is performed on the forward path, subtraction on the reverse path) the sensing range every time it comes back to restart its new path exploration for deployment using Equation (1).

$$N_s = O_s \pm S_r \quad \text{------- (1)}$$

Where Ns denotes the new starting location, Os refers to the old starting location, and Sr is the current sensing range.

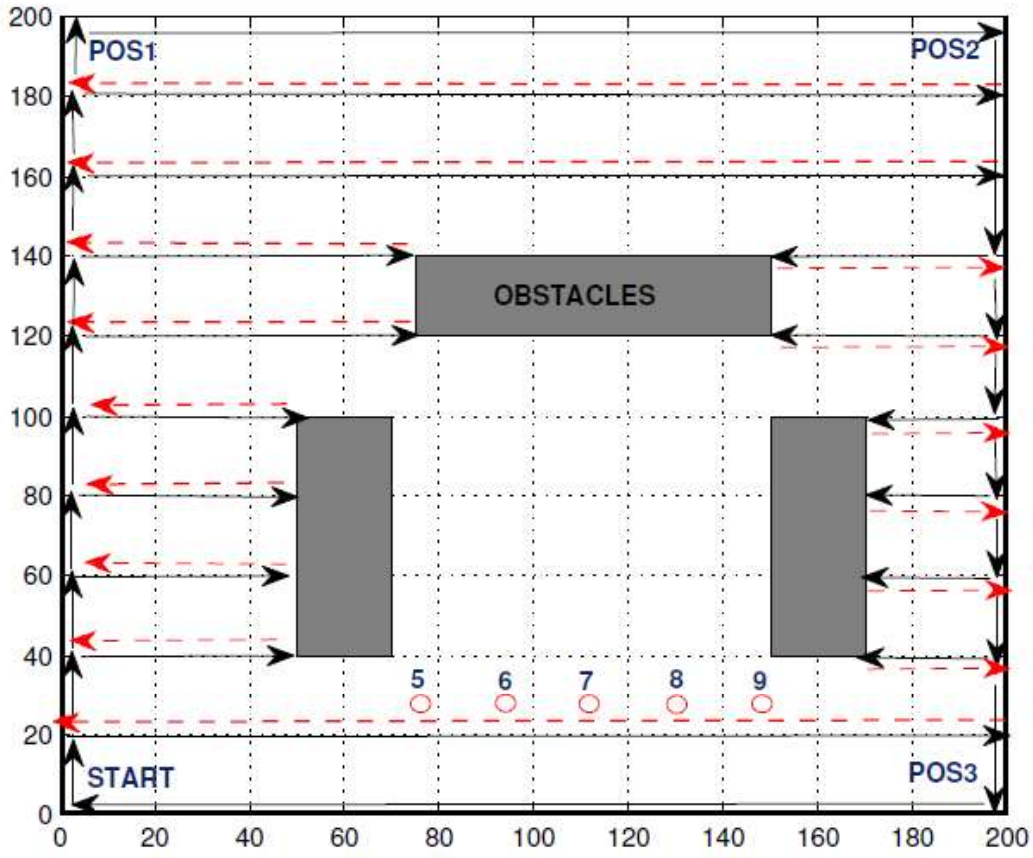

Figure 3.2. Basic SCAN deployment by a robot. The solid black arrow refers to the robots' forward path, and the dashed red arrow refers to the backward path. Red circles represent nodes with empty neighbors. (Theoretical view)

This simple scheme described by Equation (1) is repeated until the robot reaches the corner at the (0,200). Now, the robot R adds another parameter for checking already placed sensors in the field while moving from (0,200) to (200,200).

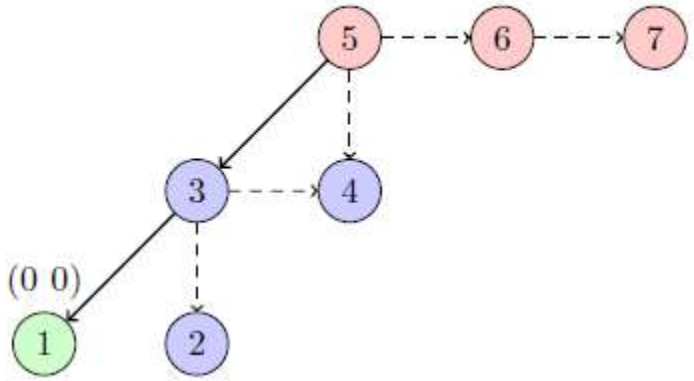

Figure 3.3. Initially, nodes 5, 6, and 7 have empty neighbors. Only the message forwarded from 5 is shown for simplicity. A similar approach is used for nodes 6 and 7 to forward the information to node 1 located at the origin (0, 0).

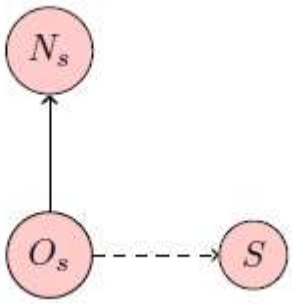

Figure 3.4. Os adds Sr when going up and reaches a new starting location, Ns. S denotes the stopping criteria (when a boundary/obstacle is encountered by the robot).

In Figure 3.2., it is evident that the robot can determine the presence of any sensor without going all the way to that specific co-ordinate. It can sense obstacles using a laser beam/acoustic signals within a distance S_r, thus reducing the total path travelled by the robot. So, when the robot moves from (0,200) to (200,200), it does not place any sensor in the field as the sensors are already placed in its way. Now, the robot moves from (200,200) to (200,0) by decreasing its step size by S_r and places sensors where there is none. It also assigns sensor ID for each placed sensors which are increasing monotonically. The robot will reach to the third location (200,0), by doing this. When the robot reaches a point where it has sensors/obstacles/walls around its four directions, it pauses for a moment and moves forward or backward depending on the situation. In this scenario, the robot R moves to (200,0) and it gets back from there to the original location (0,0). So, robot follows the forward movement in four directions, and the sensor placement direction is always on the right side with respect to that direction (Figures 3.2 and 3.3).

Basic-SCAN terminates when the robot visits all the four corners at least once. Figure. 3.4 shows that the robot has visited all four co-ordinates, namely, (0, 0)–(0,200)–(200,200) – (200, 0). So, the robot will terminate running SCAN and stay at C3 (Command & Control Center).

3.1.1.2. BASIC SCAN WITH F-COVERAGE

After the basic scan completes there would be several areas in the ROI where the sensor coverage cannot be guaranteed, especially in the cases where the obstacle distribution is very complex or the obstacle have irregular shapes and sizes. In such scenarios, the simple basic SCAN alone cannot guarantee the best results, there comes the need of F-Coverage based

deployment. For example: in the **Figure 3.6** there is an uncovered region in the center of ROI where the basic SCAN algorithm failed to cover. Robot runs F-Coverage algorithm to know the additional prospective locations where sensors can be placed. We noticed that in **Figure 3.4**, sensors with IDs 5, 6, 7, 8, and 9 have empty places on all of its sides. So, all of these nodes send the beacon message using multi-hop communication back to location (0, 0). For example, in **Figure** 3.2, if node 5 has neighbors 3, 4, and 6 then it sends this message to 3 first, and 3 sends to 1. Similarly, 6 sends to 4, 4 to 2 and 2 to 1.

When robot receives all these information from 5, 6, and 7, it again prioritizes according to sensor ids. The sensor with the lowest ID gets the message first as a next forwarder. Then robot moves to location of neighbor 5 (these locations were initially calculated by the robot while deploying and kept in the memory) using the reverse path and resumes placing sensors using SCAN. Figure 3.8 shows the deployment scenario after the F-Coverage algorithm completes execution. It is clear from the figure that the complete ROI has been deployed with sensors and the coverage is guaranteed. Hence, we can conclude that when F-Coverage algorithm is applied over the SCAN algorithm the result is the best case sensor deployment.

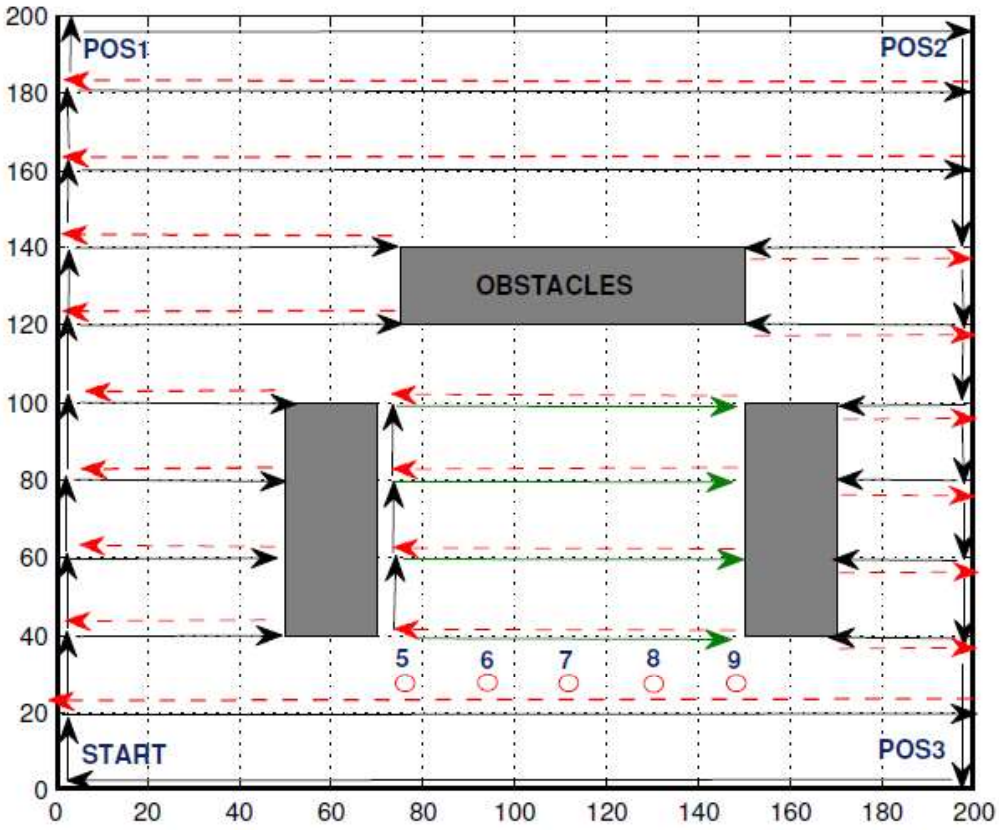

Figure 3.5. Basic SCAN with FC deployment by the robot. Five sensors (5, 6, 7, 8, and 9) with empty neighbors are selected using SCAN-FC on the same path. The sensor with the lowest ID is responsible for event notification to the control center. No additional message exchange is required in this scenario.

3.1.2. DESIGN OF OPPORTUNISTIC SCAN & OPP-SCAN WITH F-COVERAGE

This section provides design details with examples for both Opportunistic SCAN algorithm and F-Coverage version of Opportunistic SCAN algorithms. Firstly we begin by explaining the Opportunistic SCAN algorithm with an actual real scenario followed by explanation of the F-Coverage version of Opportunistic SCAN algorithm.

3.1.2.1. OPPORTUNISTIC SCAN ALGORITHM:

In this section we present an overview of Opportunistic SCAN algorithm which is an enhancement over the Basic version. We have proven that the design of opportunistic form of sensor deployment is an enhancement over major sensor deployment algorithms and gives optimal results with minimal computing resources and lesser execution time. Figure 3.6 provides an overview of Opportunistic SCAN algorithm. In Figure 3.6 the robot moves quite similarly like Basic SCAN except it does not go back when any obstacle and boundary is faced. The robot starts its tour from START (0, 0) and reaches POS3 (200, 0). This time, the robot does not go back to START (0, 0), instead, it looks for any other available route and finds one on the top. Here comes the difference between Basic-SCAN and Opportunistic SCAN in terms of switching the deployment direction. As, robot keeps the status of the last deployment direction, it always turns opposite direction from earlier whenever it has to change the normal route. Figure 3.7 shows the hierarchical form of the sensor deployment.

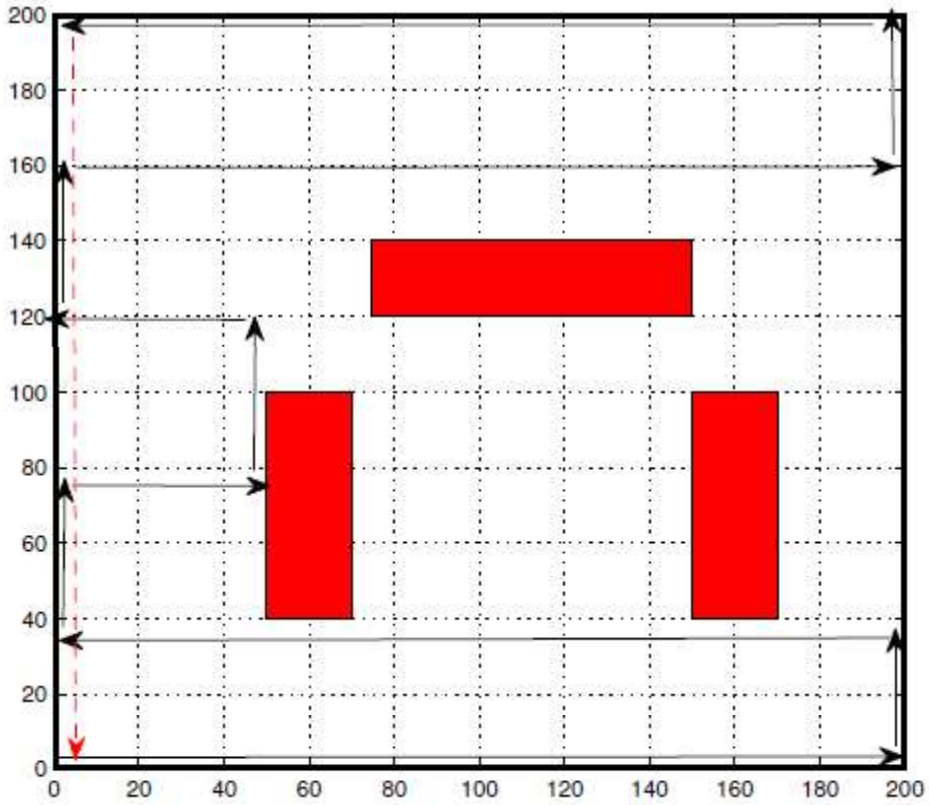

Figure 3.6. Opportunistic SCAN deployment by robot, shows less coverage compared to basic SCAN algorithm implementation for this particular obstacle distribution. (Theoretical view)

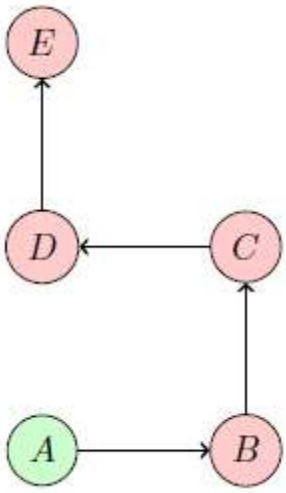

Figure 3.7. Illustration of deployment direction change for the robot. Adding Sr to B and D while going up reaches the new starting locations C and E.

Figure 3.8 show that the robot follows the path START-POS3-POS2-POS1. Now, the robot needs to reach START, as it has already visited four corners once. When returning to START, if any empty place is found, sensors are deployed in those locations as well. At the end of the robot's first tour, an empty area remains, as when using basic SCAN.

3.1.2.2. OPPORTUNISTIC SCAN WITH F-COVERAGE

Similar to the Basic scan deployment strategy, after the Opportunistic scan completes, there would be several areas in the ROI where the sensor coverage cannot be guaranteed, especially in the cases where the obstacle distribution is very complex or the obstacle have irregular shapes and sizes. In such scenario the F-Coverage based deployment is used after Opp-SCAN. In the **Figure 3.8** there are several areas where there are holes, Nodes 5, 6, 7, and 8 again follow Focused Coverage and send information about the empty neighborhood to START. Then, the robot moves to 5 and starts to deploy again. As mentioned earlier, the initial Opportunistic SCAN deploys sensors via robots to some places and misses out the regions which fall inside the intersection of several obstacles.

Figure 3.7 shows the initial distribution of sensors across the ROI. In **Figure 3.8** we shows a gray square denotes the hiatus for the robot after the second round. The robot waits there for some predefined threshold time to look for any prospective hole. If no more notifications about holes are received, it returns to START.

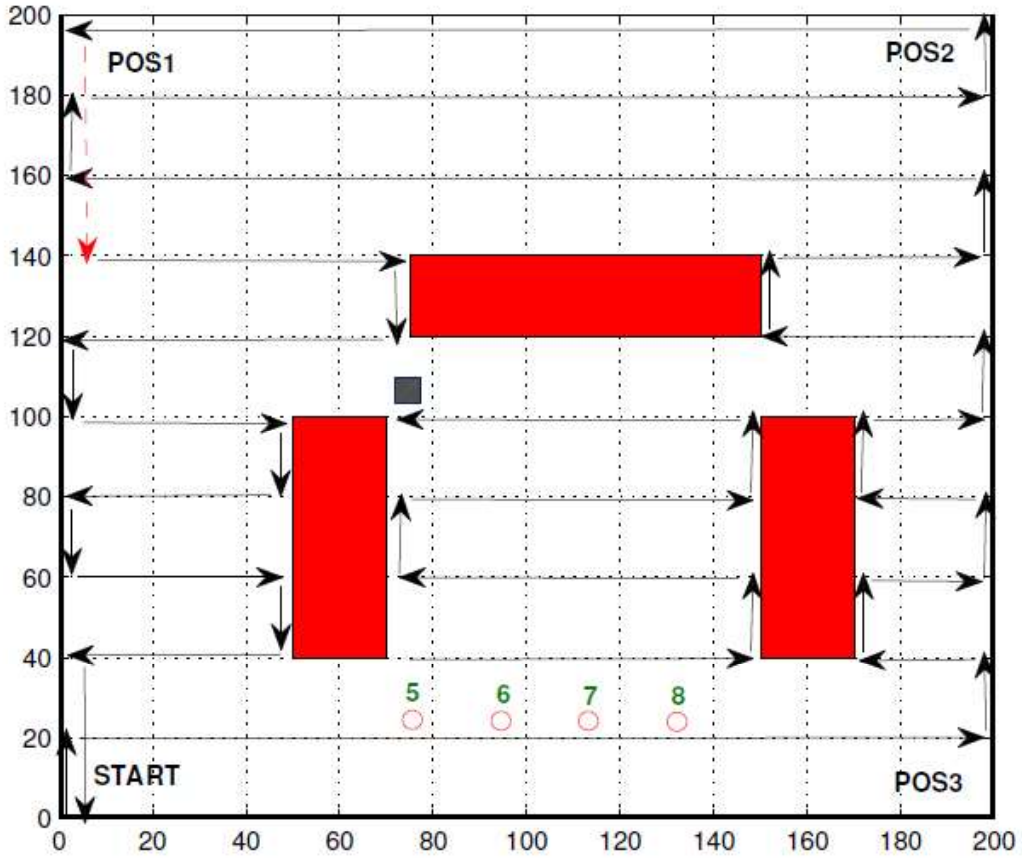

Figure 3.8. Opportunistic SCAN with FC deployment by robot. Nodes 5,6,7,8 did not have neighbor to all of its four sides. Next deployment starts from node 5 by the robot.

3.1.3. Architecture & Pseudo Code – BASIC & OPPORTUNISTIC SCAN:

In this section, we present the steps and pseudo code of both the basic SCAN and Opportunistic SCAN algorithms. First, we present the variables' identity in Table 3.1 before going through the algorithms in detail. SCAN and SCAN-FC both algorithms are used for initial deployment of sensors via a single robot.

Variable / Function	Description
Offset	A single step size
TR_{sensor}	Robots' sensing range
O_dir	Direction of obstacle
cp	Current position
co	Coordinate
ops	Operation type
BF	Boundary flag
m_dir	Moving direction
go_back	Move back backward path
move_right	Move 90 degrees right
store_coordinate	Save the geometric locations

Table 3.1. Information about parameters used in Algorithm. 1, 2, 3, and 4

Algorithm. 1 presents the basic SCAN algorithm and Table 3.1. Refers to the various parameters used across all the algorithms.

- ***Offset:*** variable refers to the distance between two consecutive sensors which would be used for the robot to identify the next placement of sensor from its current position.
- ***TR_{sensor}:*** refers to the robots sensing capability, the prerequisite for the algorithms to be operational is that the TR_{sensor} value should be at least equal to or greater than the offset.
- ***m_dir:*** has the moving directional value right.
- ***maxX and maxY:*** are the maximum coordinates of the ROI where the deployment is to be carried out. As mentioned in earlier chapters, the only required knowledge the robot haves is its initial coordinates and the maximum ending coordinates. maxX and maxY are these ending coordinates where robot need to stop the sensor deployment algorithms. This co-ordinates can be referred to as the boundary of the region of interest.
- ***o_dir:*** refers to offset direction, co refers to the coordinate, cp indicates the current position, and

- ***BF***: indicates the boundary flag.

ALGORITHM 1: Skeleton of ***SCAN*** algorithm for initial sensor deployment executed exclusively at the robot processing devices.

Input: Start position (sp) and current position (cp) of the robot. maxX and maxY can be value representing the maximum coordinates of the ROI.

Output: Maximum area of the ROI is covered by the sensors deployed by the robot.

```
1:   offset = TRsensor
2:       m_dir = right
3:       BF = 0
4:       maxX = 200, maxY = 200;
5:       if (o dir = NORTH) then
6:                PLACESENSOR (maxY, cp, m_dir, offset)
7:       end
8:       if (o dir = SOUTH) then
9:                PLACESENSOR (maxY, cp, m_dir,-offset)
10:      end
11:      if (o dir = EAST) then
12:               PLACESENSOR (maxX, cp, m_dir, offset)
13:      end
14:      if (o dir = WEST) then
15:               PLACESENSOR (maxX, cp, m_dir,-offset)
16:      end
17:      if (o dir = NORTH=SOUTH) then
18:               co = Y
19:      end
20:      if (o dir = WEST=EAST) then
21:               co = X
22:      end
23:      if (o dir = NORTH=EAST) then
24:               ops = >=
25:      end
26:      if (o dir = WEST=SOUTH) then
27:               ops = <=
28:      end
29:      repeat
30:               repeat
31:                        PLACESENSOR;
32:                        forward to m_dir
33:               until obstacle = 0 or BF = 0;
34:      until cp.coopsmaxP.co;
```

35: *go_back ;*
36: *add offset ;*
37: *BF = 0*

The above presented algorithm is a basic pseudo code of SCAN algorithms. This is the core logic behind the robot's capability of carrying out autonomous deployment. The algorithm can be considered as a finite state machine (***FSM***) where the robot switches between various states based on its current condition. The algorithm starts from an initial coordinates (0,0) and the robot moves horizontally incrementing its movement by offset value and placing one sensor at a time. It continues this movement till it encounters either an obstacle or a boundary. Once either of the conditions happens to be true, then the robot makes a certain angular turn (180^0 degrees in case of Basic SCAN and 90^0 in case of Opp-SCAN) and then moves vertically for another offset value and again make a turn. This steps are continued throughout the deployment process until the robot either reaches the maximum coordinates or ends up in certain timeout condition.

This form of algorithmic presentation is a very basic form of what is actually implemented, as there were numerous critically issues related to environment and system integration which aren't represented in the algorithm, however they are clearly presented in the implementation part of this work.

ALGORITHM 2: Skeleton of ***PLACESENSOR*** algorithm for dropping a sensor at robot current position and storing its details.

Input: Maximum Coordinates (maxP) in any direction, Current position (cp) of the robot, Robot current direction (m_dir) and offset.

Output: Robot drops a sensor at its current position represented by (cpX) and (cpY).

```
1:    PLACESENSOR (maxP, cp, m_dir, offset)
2:    cpX = cp[0] + offset;
3:    cpY = cp[1] + offset;
4:    maxX = 200, maxY = 200;
5:    if (cpX < maxX) AND (cpY < maxY) then
6:          Drop sensor at (cpX, cpY ).
7:          Store cpX, cpY, m_dir in the robots memory
8:    End
```

Algorithm 2 shows the techniques used by the robot to place a sensor at a particular location of ROI. The algorithm take maximum coordinates, current robot position and current robot's direction as input and adds offset values to the current position and drops a sensor at the current position + offset location. This is a simple representation of the sensor dropping technique which would be executed on the robots processor.

ALGORITHM 3: Skeleton of the ***OPP-SCAN*** algorithm for initial sensor deployment as executed by the robot.

Input: Start position (sp), corner count set to zero, and current position (cp) of the robot

Output: Maximum area of the ROI is covered by the sensors deployed by the robot.

```
1: repeat
2:      if ! (boundary || obstacle) then
3:              moveright;
4:              placesensor;
5:              storecoordinate;
6:              if (boundary = max) then
7:                      corner = corner + 1;
8:              else
9:                      corner =corner;
10:             end
11:             else
12:                     if (boundary || obstacle) then
13:                             add offset;
14:                             if (boundary || obstacle) then
15:                                     move up;
16:                                     if dir = right then
17:                                             dir=left;
18:                                     else
19:                                             dir=left;
20:                                             add offset;
21:                                             if! (boundary || obstacle) then
22:                                                     move dir;
23:                                                     place sensor;
24:                                                     store coordinate;
25:                                                     if (boundary = max) then
26:                                                             corner = corner +1;
27:                                                     else
28:                                                             corner=corner;
29:                                                     end
30:                                             else
31:                                                     wait;
32:                                             end
33:                                     end
34:                             else
35:                                     wait;
36:                             end
37:                     else
38:                             wait;
39:                     end
40:             end
41:     until (corner! = 4);
```

Algorithm 3 represents the procedure for Opportunistic SCAN which runs until ***corner*** value becomes 4, i.e., the corners of the square shaped ROI.

- ***boundary*** and ***obstacle:*** variables checks for any available obstacles or boundaries along the robots' way.

- ***move_right:*** function turns the robot to move in the right direction with respect to its current position.
- ***place_sensor:*** function places a sensor when the conditions are checked true.
- ***store_coordinate:*** function gathers the coordinates where the ***place_sensor*** function places sensor successfully.
- ***Corner:*** is incremented when the robot reaches at (0, 0),(200,0),(200,200), and (0,200) locations.
- ***Dir:*** is changed from its previous value every time the robot moves forward using the ***move_up*** function.

3.1.4. F-coverage based decentralized critical region exploration:

In this section we provide the details of F-Coverage algorithm applied over the Basic SCAN algorithm. After the robot successfully completes the execution of SCAN algorithm and when the robot gets back to the start position, we ideally assume that the whole region is deployed with sensors. But there may be some obstacles which can create holes in the network.

To ensure that there are no holes left in the network, sensors run another algorithm which is mentioned in Algorithm 4. All the sensors run ***FIND_HOLES*** to check any available empty neighbor. The sensors send this information to the robot with its location and ID through intermediate neighbors. Sensor selects the next sensor as a transmitter from its neighbors using the minimum ID to reach the robot. This step can be easily performed inside each sensor using the exchanged information among the neighbor sensors with their coordinates and IDs. For example, referring to Figure 3.4, node 53 will receive responses from three neighbors (nodes 52,

54, and 10). Using the directional antennas the node 53 can easily figure out that there is no message from the top side indicating that there is no sensor (i.e., there is a hole). Then, node 53 will pick the node with the smallest ID among its neighbors as a next forwarder, i.e., node 10, to the command and control center.

ALGORITHM 4: Critical region exploration algorithm executed by the sensors after robot's first tour

Input: Sensor coordinates;

Output: deploy sensor in the empty neighborhood;

```
1:    for all the sensors do
2:        check holes within range;
3:        if hole found then
4:            inform robot with calculated coordinate and ID;
5:            PLACESENSOR;
6:        else
7:            no holes found
8:        end
9:    end
```

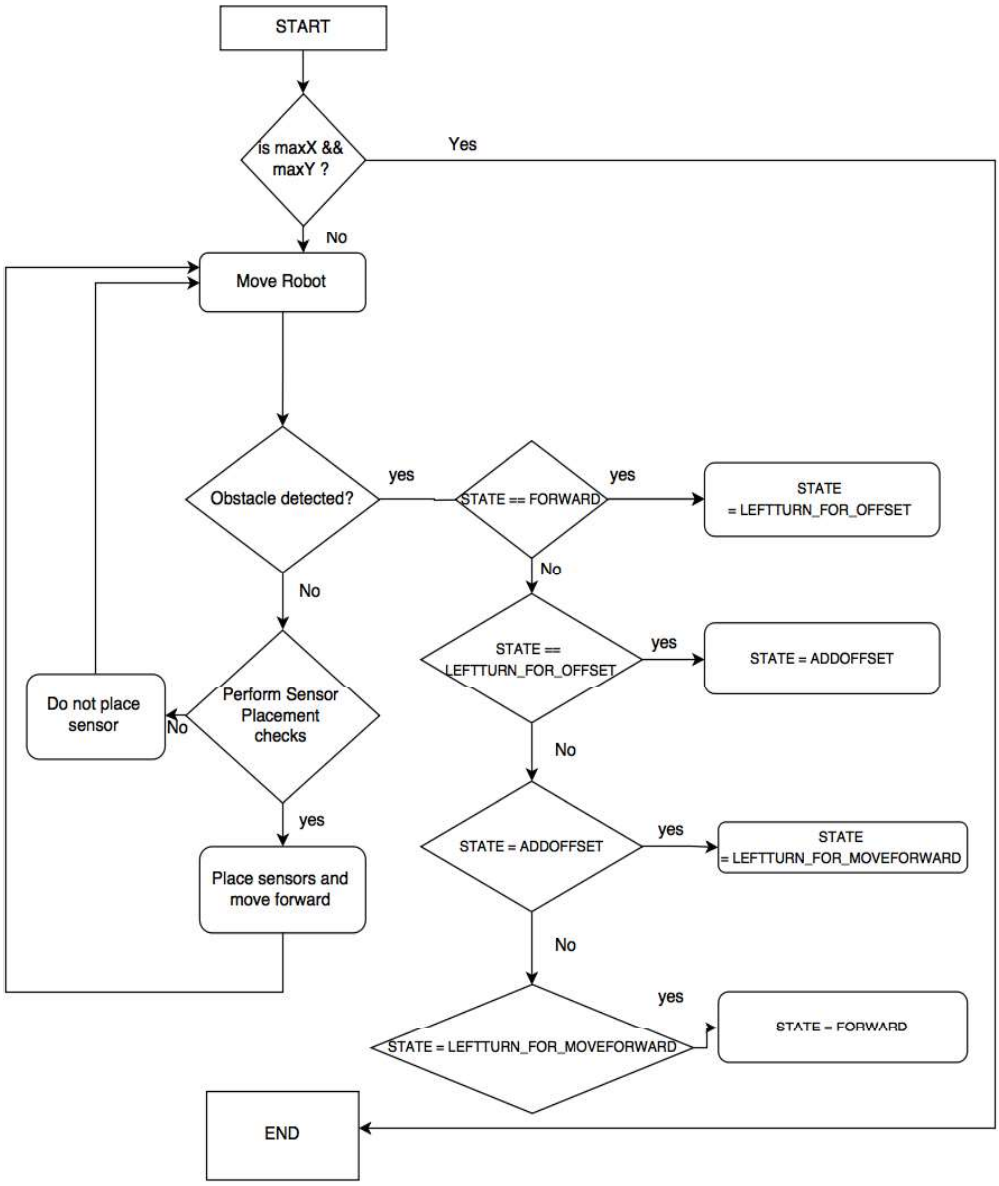

Figure 3.9: A basic flow chart of SCAN algorithms for sensor deployment, an overview of various possible decisions and states a robot can take.

3.1.5. Theoretical Analysis

This section provides theoretical analysis of the SCAN & OPP-SCAN algorithms. The analysis will be based on the best, worst and average case of performance parameters.

Lemma 3.1: Both SCAN and Opportunistic SCAN ensure coverage guarantee for the intended ROI (region of interest).

Proof: Let us consider that our intended ROI requires s points for sensor placement excluding the obstacle regions. on the other hand, robot moves from the origin to reach four corners to satisfy the initial stopping criteria and stores the travelled points in its own stack, m. Value of m has to be larger or equal to s to ensure that the ROI is covered with sensor deployment. As robot may visit the same grid point's s more than once. In case of basic SCAN, m holds the set of points twice,

m = 2{p1, p2, p3… p_n}. On the other hand, Opportunistic SCAN holds the coordinates once except in certain situations where the only option to explore the new region are to go back to its previous path. So, the unique set of points are to be considered while comparing with the number of points to be explored. Equation 2 provides the general relationship of the total travel distance for both the SCAN and OPP-SCAN algorithms.

$$T_d = \sum_{i=1}^{m} D_i + \sum_{j=1}^{n} D_j + \sum_{k=1}^{p} D_k \qquad (2)$$

Td denotes the total travel distance, D_i counts the values of the distance for the initial m deployed sensors, and D_j and D_k count the distance from the FIND_HOLES procedure. So, the total number of deployed sensors, T_n, can be determined using Equation (3).

$$T_n = m + n + p \qquad (3)$$

Thus, we can conclude the following relationship.

$$T_n \geq s \qquad (4)$$

s and m are equal when there is no obstacles (i.e., n and p both have null values). We need additional points to cover using FIND_HOLES when the obstacle distribution is random. Robots sensing range has to decrease to achieve the goal. Thus, the initial intended number of points (s) does not match in those cases. So, the total number of points deployed becomes equal or greater than the initial calculated points (s). Hence, the below relation holds.

$$m + n + p \geq s \qquad (5)$$

Lemma 3.2: If the width and height of the ROI are m and n respectively, then the maximum travel distance for SCAN and OPP-SCAN becomes $2 \times \frac{n(m+1)}{l+m}$ and $\frac{n(m+1)}{l+m}$ respectively, where *l* is the sensing range.

Proof: Let, the width and height of the ROI be m and n units. The robot has maximum sensing range of l units. If the robot's deployment direction is towards height (n), then basic SCAN travels 2n distance before moving towards width (m). So, the robot must move $m/l + 1$ times towards width with distance 2n and an additional length of m.

$$T_{scan}(\max) = 2n \times \left(\frac{m}{l+1}\right) + m \qquad (6)$$

In the above equation, typically $m/l \gg 1$ results significant decrease of travel distance with the increment of *l*. On the other hand OPP-SCAN provides 50% reduction in travel distance compared to basic SCAN.

$$T_{opp\text{-}scan}(\max) = n \times \left(m/l + 1\right) + m \qquad (7)$$

For example, if the width and height of the ROI are both 8 units and the sensing range is 2 units, then we can calculate the distance for both basic SCAN and Opportunistic SCAN.

$T_{scan}(\max) = (2 \times 8) \times \left(8/2 + 1\right) + 8$

$= 16 \times 5 + 8$

$= 88$

$T_{opp\text{-}scan}$ (max) $= 8 \times \left({}^{8}/_{2} + 1\right) + 8$

$= 8 \times 5 + 8$

$= 48$

Lemma 3.3: Execution efficiency: The time complexity of sensor placement of SCAN is O(n) whereas for BTD its O(n3). SCAN algorithm is designed to outperform in-terms of both execution and overall completion time.

Proof: SCAN algorithm performs the task of placing a sensor at a particular location by simply looping for a single time to check if the sensor is already placed in near proximity of its current position. So, this looping would be of a single magnitude of O (n).

Whereas, in case of BTD, the robot does two activities. One for placing the sensor at a particular location and the second is to loop around each and every placed sensor to perform the coloring task. i.e. if all the neighbors of a particular sensors are alive then it changes it color to black. Thus the complexity becomes O(n3).

$\sum_{i=1}^{n}\{[Sensor\ Placement\ Check\}\sum_{j=1}^{n}\{[Neighbour\ detection\ Check\}\sum_{k=1}^{n}\{[Changing\ the\ color\ of\ the\ sensors\}$

$= O(n3) = \frac{n(n+1)(2n+1)}{6}$

3.2. Basic Concepts, Mathematical Model & Assumptions:

Now that we have clearly described the design and architecture of all algorithms, it is good time to cover some of the basic concepts of WSN like sensor deployment, Coverage, mathematical model etc. which would be applied to our algorithms. The aim of this section is to set a solid background before going into the implementation & performance evaluation and also to ease the understanding of the reader.

3.2.1. Deployment of Sensors

All deployment algorithms considers the entire hazardous area to be the region of interest (ROI) which is further divided into equal sized virtual grids. The first phase, which we call deployment, targets the optimized placement of sensors considering the unpredictable locations and dimensions of any obstacles within the ROI. We assume that the robots have sufficient number of sensors to deploy the count of which defers from application to application and mainly depends on the size of ROI. In general, the number of sensors a robot should carry be such that, they can cover the entire ROI. Initially, each robot is responsible for sensor deployment in its specified virtual grid in the ROI. Robots lay sensory units independently and asynchronously, which is governed by an intelligent direction ranking feature of the proposed algorithm. Once a sensor is laid down, it starts communicating with its one hop neighbor exchanging the information about their IDs and co-ordinates.

In ideal scenario, every deployed sensor is expected to communicate with all of its four neighbors. If a sensor detects holes in all of its neighborhoods, then it uses its adaptive transmission capability to reach to the node at the next nearest available node at different tier. In the worst case if no sensor is reachable, then node should communicate directly with the control

center located near to the ROI. Figure 3.10 depicts the sensor communication in best, adaptive and worst case scenarios.

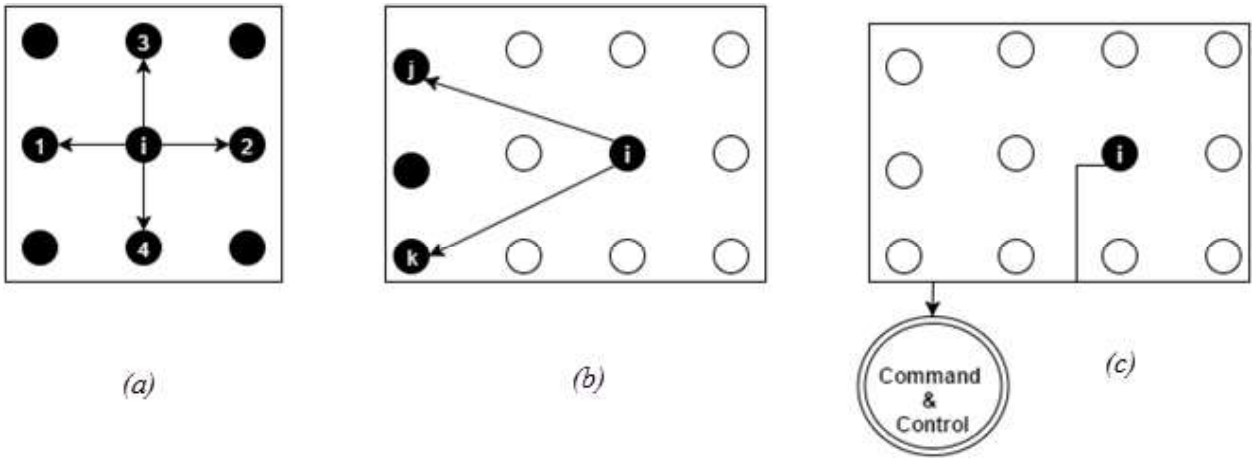

Figure 3.9 Graphical view of sensor communication in Best, Adaptive and Worst case scenarios. The best case (a), adaptive case (b) and control center in worst case scenario (c).

3.2.2. Coverage

Coverage in wireless sensor networks is usually defined as a measure of how well and for how long the sensors are able to observe the physical space. [98] Coverage and connectivity are two of the most fundamental issues in WSNs, which have a great impact on the performance of WSN's. Hence an out most consideration has to be given to this important aspect.

To obtain maximum coverage, we propose a variant of Focused coverage (F-Coverage) an overview of which was provided in earlier sections. This algorithm would be executed on top of initial SCAN by the robot to further detect nodes with empty neighbor set. The nodes which have at least one neighbor location as empty communicate with beacon message via multi hop communication to the source robot. This is done using the minimum ID selection from the

neighbor set of each step. So, the message is eventually propagated to the start location. Additionally, there is one command and control center (C3) responsible for collecting the required information from the robot.

3.2.3. Mathematical model of SCAN algorithms

Consider a square area A that needs to be surveyed and simultaneously deploy maximum possible number of sensors NS_{max} $\Delta x = \Delta y = S$. A number of obstacles (NO) do exist and there is no a priori knowledge of the positions of these obstacles.

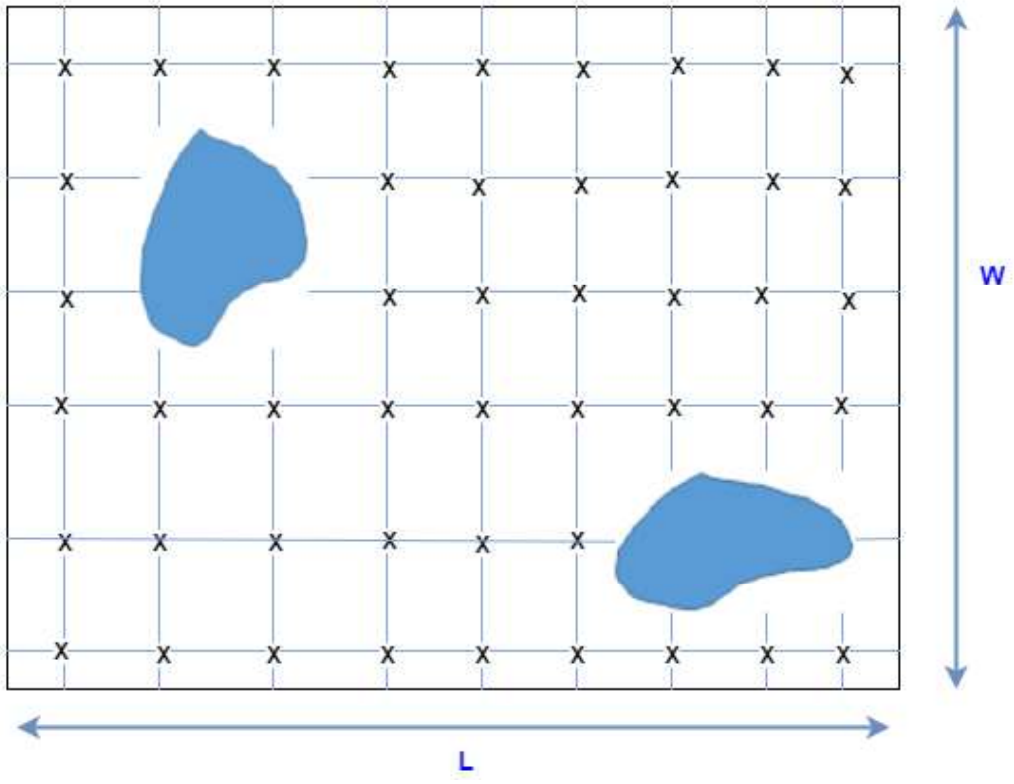

Figure 3.10 Example of a sample region of interest with obstacles with unknown locations, sizes and shapes.

Let ***A*** has a dimension of ***L by W***; both are multiple of ***S***. In case of no obstacle, ***N by K*** sensor can be deployed.

$$\text{Where } N = {}^{L}/_{\Delta x},$$

$$K = {}^{W}/_{\Delta y}.$$

We denote this as D_{max} (The maximum possible number of deployed sensors). Now, in case of obstacles, the question is, what is the maximum number of sensors (D^{o}_{max}) that can be deployed over the remaining area A_R. So our problem can be formulated as follows.

$$\max\{N | A_R = A - \sum_{i=1}^{m} O_i\}$$

Where O_i is the area of the i^{th} obstacle. The problem becomes very difficult. There is no existing method, which can compute exactly the number of sensors regardless of obstacle shape or orientation.

One of the desirable properties of sensor deployment is to acquire a perfect grid like deployment to achieve optimized coverage and connectivity. Figure 3.2 shows an example of desired sensor deployment pattern and Figure 3.12 shows the path taken by the robot to achieve the deployment.

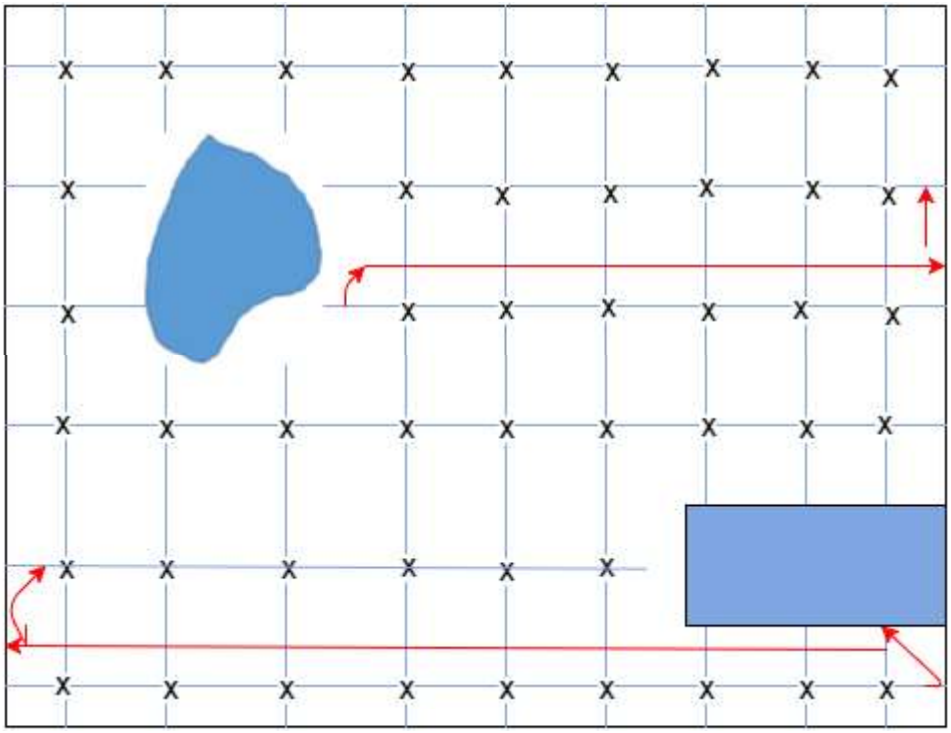

Figure 3.11. The path taken by the robot to achieve the grid deployment.

The robot initiates movement from the point (0,0) till it reaches the point (MaxX,0), it then turns to the left followed by adding an offset. It faces and obstacle before completion of offset value, so the robot turns left and move further. However, at this point the robot doesn't deploy the sensors as it has not completed the offset movement. It keeps track of the moved offset and adds up the remaining offset on the other end. This way the robot achieves a perfect grid like deployment.

Let N_{offset} be the row wise inter sensor distance for the SCAN algorithm to consider while deploying the sensors. As the robot reaches the maxX co-ordinates it faces an obstacle and

covers only partial offset value say $N_{\Delta offset}$ and returns to the other side. This time the robot moves the remaining offset value

$$N_{remainingOffset} = N_{Offset} - N_{\Delta Offset}$$

Example: Considering offset value to be N_{offset} = 20 mm and the robot faces obstacle by moving

$N_{\Delta Offset}$ = 8.5. So, the robot on the other end moves the remaining offset of

$N_{remainingOffset}$ = 20 – 8.5 = 11.5. This simple technique is iterated throughout the deployment process, achieving a perfect grid like structure.

3.2.4. Perfect Grid Deployment:

In this section we present a theoretical model for the sensor deployment to proof that the result of sensor placement can be mapped to 2-Dimensional array. This is important for the robot to have a map of the entire environment in its memory. Figure 3.13 shows a ROI post completion of the first round of sensor deployment. As soon as the robot starts deploying the sensor, it stores the coordinates where the sensor is being dropped and consecutively marks the corresponding index of the matrix as ***"1"***. In this way, when the robot completes the tour of the environment, it would have a matrix as shown in figure 3.14. The ***"0"*** represents holes in the network, whereas a 1 represent a presence of the sensor.

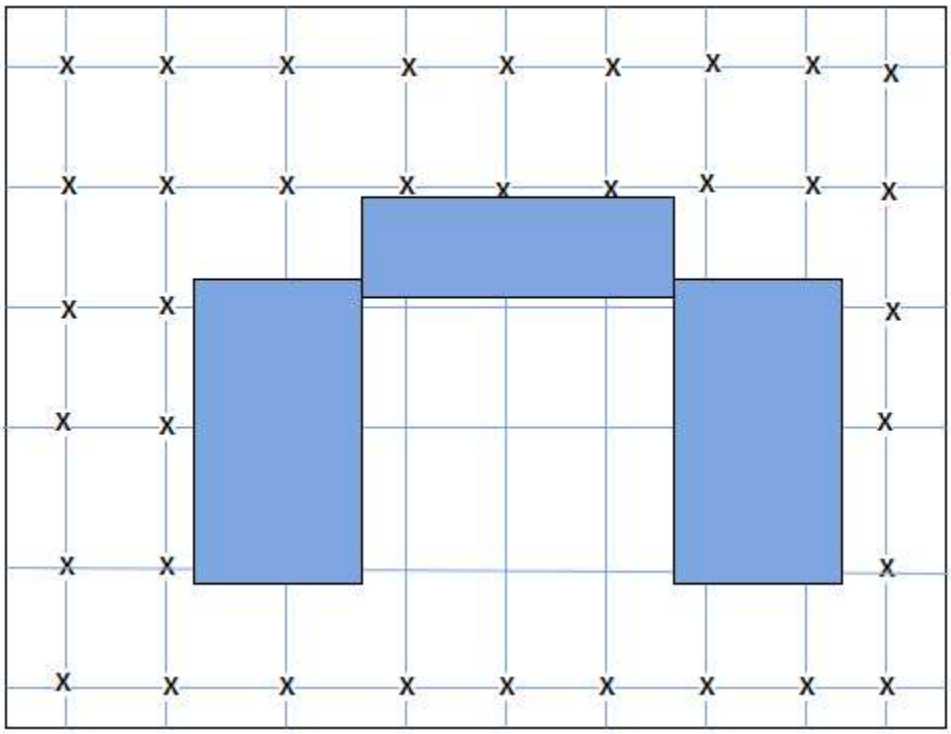

Figure 3.12. Shows an ROI post completion of first round of sensor deployment. The central region of ROI is uncovered and needs F-Coverage executes and completes the deployment.

Using the matrix, the robot can easily figure out the contiguous uncovered area which has to be covered using the Focused-Coverage algorithm. The robot now scans through the matrix and retrieves the minimum coordinates X_{42} i.e. (4,2) in this case and reaches that location and continues the sensor placement till the entire matrix becomes a unit matrix except where there are obstacles. Figure (3.14) and (3.16) shows the resulting ROI and matrix after the robot completes the F-Coverage algorithm and ensures maximal coverage. The items $X_{22,}X_{32,}X_{42,}$ $X_{26,}X_{36,}X_{46,}X_{47,}X_{27,}$ $X_{37,}X_{47}$ and X_{57} remains zero as they are being covered by the obstacles.

$$\begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} & X_{04} & X_{05} & X_{06} & X_{07} & X_{08} \\ X_{10} & X_{11} & X_{12} & X_{13} & X_{14} & X_{15} & X_{16} & X_{17} & X_{18} \\ X_{20} & X_{21} & X_{22} & X_{23} & X_{24} & X_{25} & X_{26} & X_{27} & X_{28} \\ X_{30} & X_{31} & X_{32} & X_{33} & X_{34} & X_{35} & X_{36} & X_{37} & X_{38} \\ X_{40} & X_{41} & X_{42} & X_{43} & X_{44} & X_{45} & X_{46} & X_{47} & X_{48} \\ X_{50} & X_{51} & X_{52} & X_{53} & X_{54} & X_{55} & X_{56} & X_{57} & X_{58} \\ X_{60} & X_{61} & X_{62} & X_{63} & X_{64} & X_{65} & X_{66} & X_{67} & X_{68} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Figure 3.13.Mapping the ROI to a 2-D array where a sensor is represented by one and hole is represented by zero.

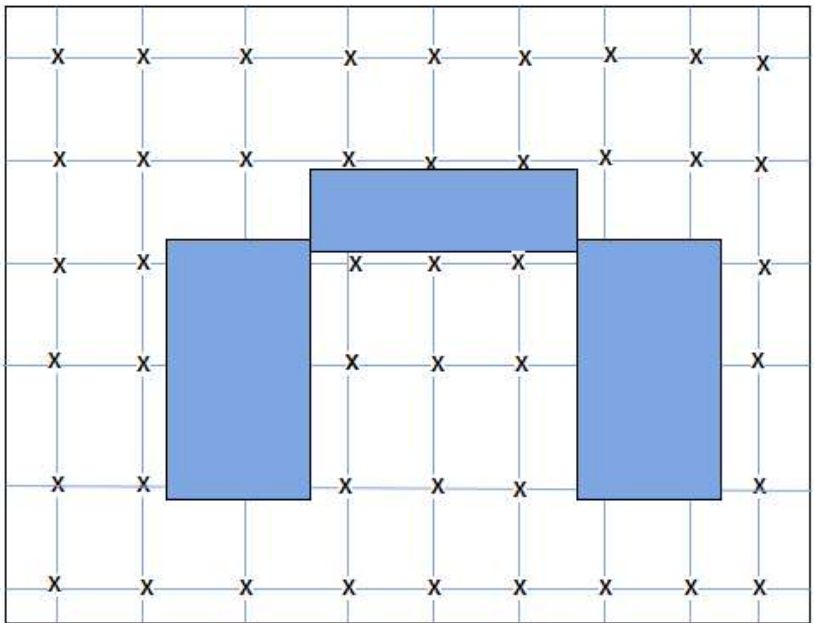

Figure 3.14. Shows a ROI post execution of F-Coverage algorithm which ensures maximal coverage.

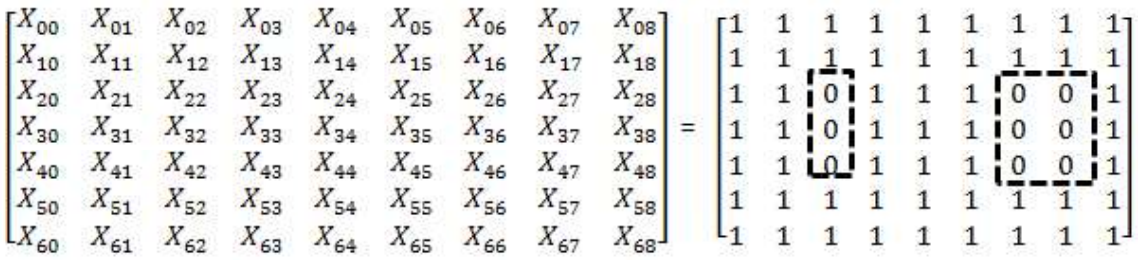

Figure 3.15. 2-D array post execution of F-Coverage algorithm where a sensor is represented by one and hole is represented by zero.

3.2.5. Assumptions

Before concluding this chapter and diving deep into the performance evaluation we have enlisted several assumptions as follows:

- Sensors sensing range can be adjusted depending on the position and their placement.
- In general cases, we assume that the obstacles are not joined with each other and do not create any circular or closed loops where robots cannot enter.
- The robot has a directional antenna and can sense obstacles using a laser beam/acoustic signal. The robot keeps track of its moving co-ordinates in its local memory to ensure and avoid exceeding predefined boundary region.
- The robot is aware of its initial starting location. It also knows the maximum sensing range of the sensors which is needed to calculate the robot's step length.
- The terrain of the ROI is assumed to be flat with obstacles.
- Obstacle distribution in ROI would be randomized and may appear at any location.
- Sensors can identify the directions of received signals using angle of arrival technique.

3.3. APPLICATION SCENARIOS, CASE STUDIES & ASSUMPTIONS

In this section we present various real world application scenarios and sample case studies. The purpose of this section is to provide a brief overview of technical aspects of algorithms, it is very important to relate them to real life examples. We have selected three application scenarios, in the first scenario we consider a ROI in which the obstacles are closer to the boundary. The reason of selecting this scenario is to demonstrate the fact that if all the obstacles are close to the boundary region, the robot will face them and go back to the boundary position and add the sensing range of the sensors to move forward to the next suitable position for the sensors to be deployed. In this way robot may travel all the way from the starting point and get back without even entering the main deployment region.

In the second scenario, we consider a ROI in which Obstacle position is slanted within the region, the purpose of demonstrating this particular scenarios was to show the behavior of SCAN based algorithm in complex obstacle scenarios. Finally, in the last one, we present a sample scenario for post deployment maintenance phase.

Let us consider a building in which a certain activities are to be monitored for which we need to be deploy static sensors all around the place. To do that, we will use mobile robots who carry the task of sensor deployment. Sensors will collect data from the environment and send them back to the command and control center using multihop communication. The robot initially enters the ROI from an origin (0,0) and receives a command from the control center to explore the building. Let, the command and control center referenced as C3, robot as R, and a set of sensors as S_n. At first, the robot R gets the starting co-ordinates, maximum boundary, and available sensor information as its initial configuration. Let, the sensing range of the sensors is $S_{r,}$ We explore more complex scenarios in the later sections. As a step size of the robot to move

forward. The entrance is denoted as * for the robot. R enters the building from * (co-ordinate is provided for this) and moves up adding S_r. Then it moves right and deploys sensors by assigning a sensor ID (sensor ID's are assigned starting from 1) to each sensor. The robot keeps the previous position of the placed sensors in its local memory while moving to the next position for further validation and tracking. The robot faces one obstacle O1 after moving some distance, D_n. Then the total distance travelled by the robot considering all four corner visits is denoted by Equation. 8.

$$\sum_{c=1}^{4} \sum_{1}^{n} D_n = n \times S_r, \forall n \in R \quad \ldots\ldots\ldots\ldots\ldots\ldots 8$$

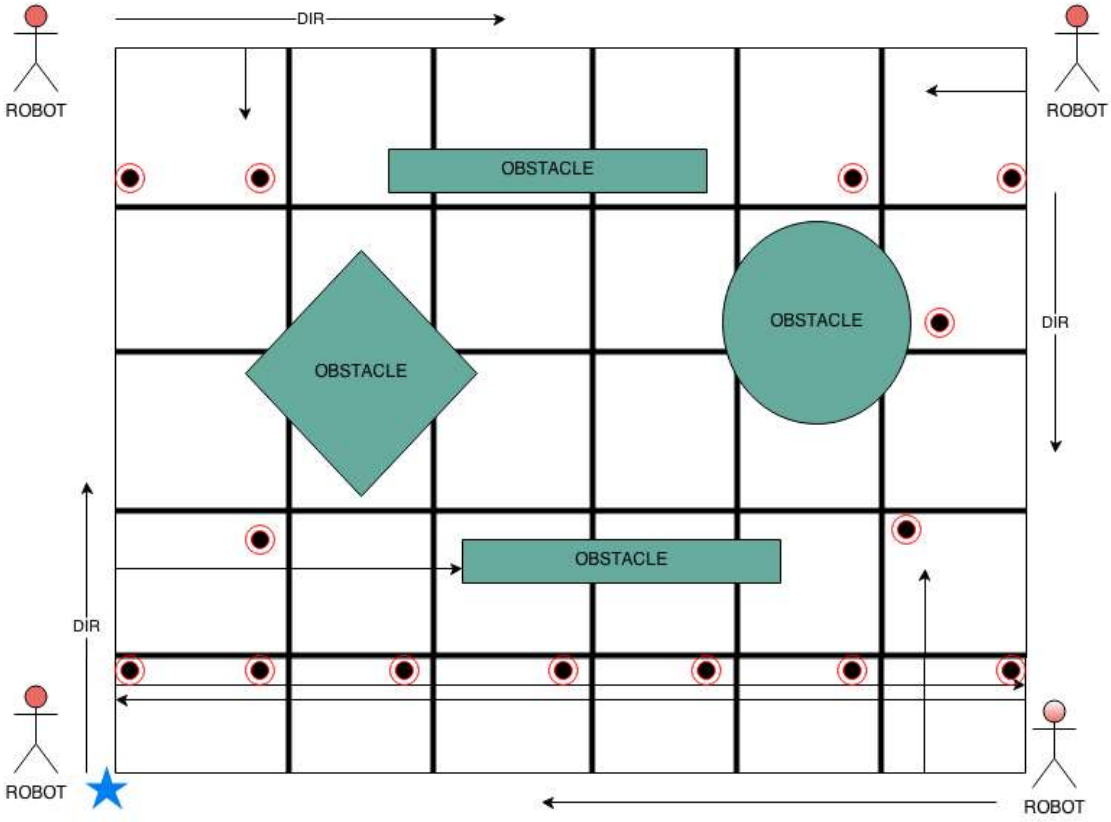

Figure 3.16. An example of an application scenario, assuming the indoor view of a building with obstacles at different locations. The process only reveals the initial SCAN algorithm deployment status. Focused coverage algorithm is not applied in this example.

3.3.1. Case Study 1: Obstacles are closer to the boundary

In Figure. 3.18, it is quite impossible for the robot to enter into the intended area for sensor deployment. If all of the obstacles are located close to the boundary region, the robot will face them and go back to the boundary position and add the sensing range of the sensors to move forward to the next suitable position for the sensors to be deployed. In this way robot may travel all the way from the starting point and get back without even entering the main deployment region. In this scenario two possible cases may arise. Firstly, if the distance (d) between the boundary and the obstacle becomes less than the sensing range then robot will come back to the starting point without placing any sensors. So, there will be no single sensors in the field which will further call for robots to perform re-deployment (initially no sensor is placed, thus no execution of focused coverage algorithm). This issue is solved by again, sending the robot to the field with a decreased sensing range. This process is repeated until a sensor is placed in the intended region. Once a single sensor is placed, it can obtain information about empty neighborhood, which are not yet covered by the robot. Secondly, when distance (d) is more than the sensing range, it falls under normal conditions that ensures deployment of the sensors.

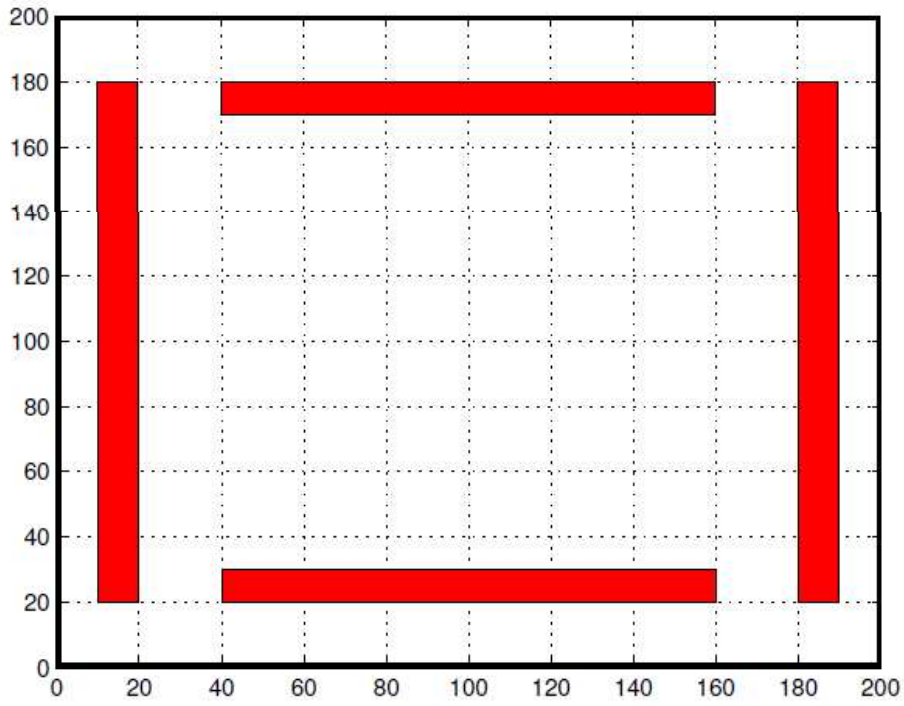

Figure 3.17. Illustration of the abnormal scenario when robot needs to adjust its sensing range after reaching the first corner of the deployment region. The example explains a theoretical view when all the obstacles are very close to the boundary. The distance $d < S_r$, where d is the distance between obstacle surface and boundary region.

3.3.2. Case Study 2: Obstacle position is slanted within the region

As SCAN ensures both horizontal and vertical coordinate accumulation and placement, the ideal placement of obstacles would also be horizontal and vertical. Figure. 3.19. Shows another scenario where obstacles are not placed ideally with respect to the robot movement. Both the obstacles are slanted, thus the Basic SCAN algorithm cannot access between the obstacles. This case is solved by the Opportunistic SCAN algorithm, which is shown in Figure. 3.6. Using the red dashed arrow. Basic SCAN can get back whenever it finds any obstacle (Protective

algorithm). Opportunistic SCAN wants to use maximum availability of routes and does the exploration in an efficient way.

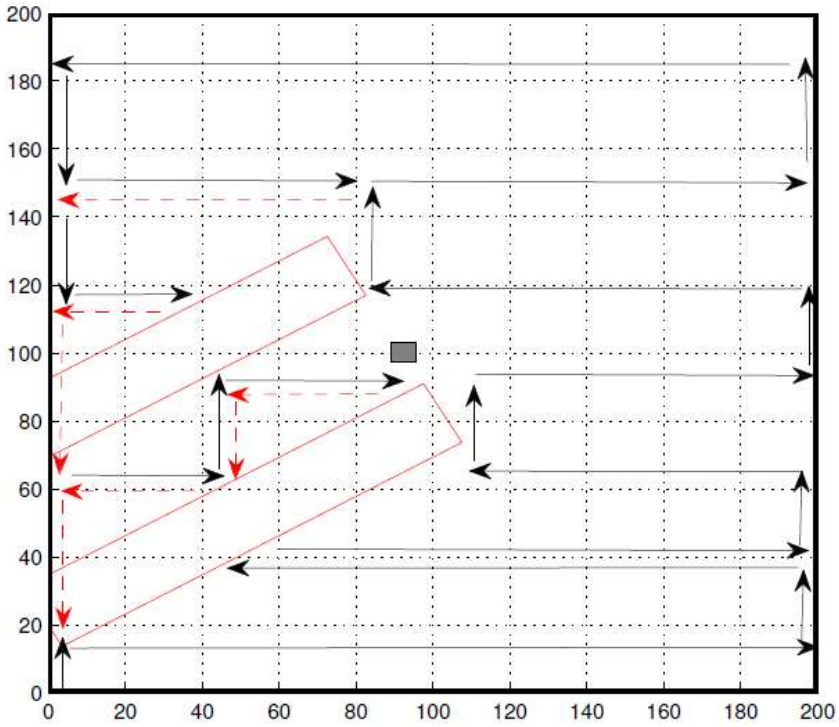

Figure 3.18. Obstacle position is slanted within the region. Both obstacles are parallel to each other, thus it's not possible for the basic-SCAN to enter and cover the whole region in between. Opportunistic-SCAN overcomes this issue.

To be more specific, if the robot comes to the location (0,60) then it will follow the black arrow on the positive x-axis. Now, basic SCAN would make the robot go back directly to position (0,60) on the first encounter of the slanted obstacle. But, Opportunistic SCAN moves the robot till the gray rectangle, where it puts a sensor and waits for some threshold time to look for any available path. As it does not find any, it returns to (0,60) and resumes for further new deployment.

3.3.3. Case Study 3: Maintenance phase:

Let us consider that some sensors are placed using Opportunistic SCAN with focused coverage via phase one and two. Figure 3.20. Describes one such scenario where 51 sensors are placed using phase one (SCAN) and the rest are deployed using phase two (Opp-SCAN). It can be noted from the figure that sensor nodes with id 9, 10, 11 each has empty neighbors. Thus, they will be responsible for alert notification initiator for the command and control center. Additionally, the nodes are placed on the same line previously. The node with the smallest ID will be responsible for sending the empty neighbor notification message to the command and control center.

In this particular case, theoretically node 9 sends the alert message, which in turn notifies the robot to deploy sensors in the empty spaces from phase one. So, next sensor placement starts from position 52, and SCAN is performed to again treat that as the new origin.

Now, let's assume all the nodes with ID 9, 10, 11 are damaged due to some unavoidable reason. So, the corresponding neighbors from both lower and upper rows will sense this empty event. Nodes 52, 53, 54 and 6, 7, 8 will sense empty neighborhood using acoustic sensors / laser beams which are already equipped with them. The question is which nodes are going to take care of this situation? As, node 6 has the minimum ID among all the six nodes, its message will reach first to the control center. If we consider another scenario where nodes from more than one consecutive row fail, then the above process cannot maintain this failure.

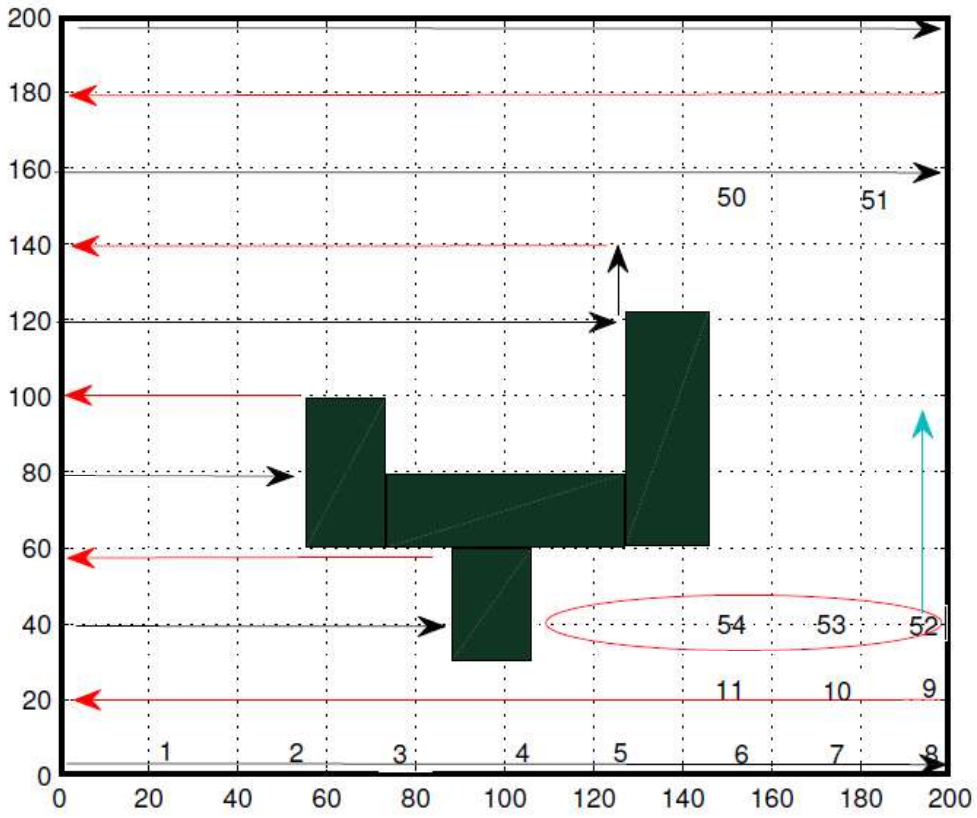

Figure 3.19. Obstacles are shown using green rectangles. Sensor placement is considered using Opportunistic SCAN-FC algorithm. Red eclipse shows the region from where Focused Coverage algorithm starts second phase deployment.

For example, if nodes 6, 7, 8 and 9, 10, 11 are down, then nodes with ID 52,53 and 54 cannot communicate with control center using focused coverage algorithm where the message is sent from one node to another using descending order of their corresponding ID's assigned by the robot. In this case, node 52 has to send messages to its above neighboring nodes. (Shown in green arrow). If this method is used, then the message transfer would take longer time compared to the focused coverage method. We can reduce this longer path by increasing the sensing range of the source node. In this case, node 52 will increase its range to find available neighbor node

who's ID is smaller than itself. So, we can say, sensing range can be increased or longer path has to be chosen to manage multilevel failures.

CHAPTER 4

REAL WORLD CHALLENGES & SOLUTION STRATEGIES

The main achievement of this work was to successfully implement the proposed algorithms on real robots. During the implementation we faced numerous real world challenges which were tackled by certain solution strategies. The chapter is dedicated towards enlisting the various challenges encountered and their respective details.

4.1. GPS-less environment:

As discussed earlier, one of crucial challenge with any deployment algorithms is to have a localization mechanism, especially in the region where there is no GPS availability. We tackle this crucial problem by using a real E-puck robot entering a region of interest (ROI) having no prior knowledge of the environment, except for its original coordinates. The robot runs SCAN algorithms, calculating its current position at each step using the embedded ODO meter.

Below example further details out robots mechanism of calculating its current position without a GPS.

Let us assume that, as per an application scenario, we need to deploy sensors at a distance of every ***2 meters*** using an E-puck robot with a speed of ***300 units*** and having a wheel radius of ***20.55 mm.***

Parameter	values
Inter sensor distance	2 meters
Speed of E-puck robot	300 units
Wheel radius of E-Puck	20.55 mm
Speed of E-puck in radians	0.00628 rad/s

Table 4.1. Robot parameters and their respective values for a standard E-Puck robot.

1 unit = 0.00628 rad/s

This implies, 300 units = 1.884 rad/s.

Converting angular to linear velocity using V = r X $\tilde{\omega}$.

Where ***'r'*** is radius of a standard E-puck robot, which is 20.55 mm.

Hence, V = 0.038622 m/s.

Using the above equations, a robot can calculate the current distance covered using the ODO meter and deploy the sensor only if it has covered a minimum distance of offset value. An important aspect to be considered here is that of a possible error in ODO meter reading. To fix this ODO meter error issue, robot would maintain a matrix of already deployed sensors and after each and every placement of sensor, the robot ensures that the inter-sensor distance is equivalent to the matrix mapping. In this way, the outcome of algorithm would ensure equi-distance and exact sensor deployment.

4.2.Dynamic obstacle distribution:

To solve this major environmental challenge, we utilized 8 IR sensors which are embedded within the E - puck robot. These sensors are capable of easily sensing the proximity of any obstacle / boundary, as soon as it encounters one and it makes a 90' turn on its right or left based on its current moving direction.

These 8 IR sensors are represented by *PS0, PS1, PS 2... PS7*. As per the basic requirements of SCAN & BTD algorithms, the robot is required to make a turn of exact 90 degrees. This was a tough task to achieve in the real world, as the E-puck robot had an inherent minor deviation in its differential wheel, while making a turn. Nevertheless, we corrected this by using a deviation variable Δx which is relatively calculated based on the robots initial angle while making a turn.

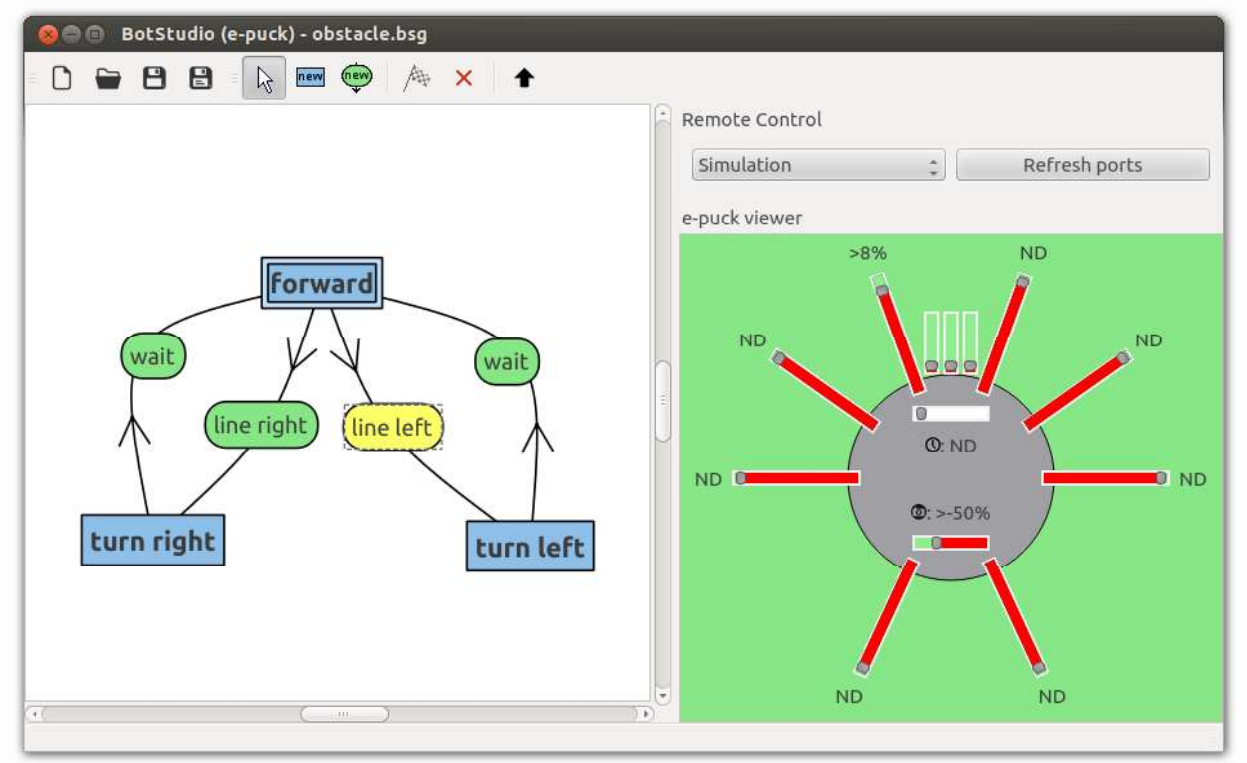

Figure 4.1. Right side of the figure shows an overview of an E-puck robot equipped with 8 IR sensors and left side shows a sample flow chart for handling obstacle sensing condition.

4.3.Uneven surface with terrains and mountains:

To solve this problem we have considered various shapes of obstacles like terrains, rectangle, square, disperse ROI etc. Additionally, we have designed algorithm which can executed on any robust robot which can operate smoothly in technically harsh temperature and humid conditions.

4.4.Tackling deadlock conditions:

In this section we prove the deadlock free nature of proposed algorithms: we show that both collisions between robots and deadlocks are avoided, and also that any execution of the algorithm eventually ends in a uniform deployment configuration.

This point can proved in two stages, they are:

- First, enlist the various **STATES** in which robot can go into, and
- Various scenarios where deadlock might occur and prove that our algorithm handles all such scenarios.

A robot at any time has a **STATE** which determines the current operations & rules the robot would follow. Upon startup, the execution of the algorithm is logically divided in four phases.

STATE 0 (DEPLOYING):	The robot is currently busy with deployment of sensors in the ROI.
STATE 1 (COLLECTING):	The robot has completed the deployment and waiting for a multi-hop messages from the sensors to continue the F-Coverage algorithm.
STATE 2 (TRAVELLING)	Upon receiving an F-Coverage command from any sensor, the robot initiates it movement towards the requested sensor to start focused coverage.

STATE 3 (SLEEPING):	The robot after completion of deployment or to conserve the energy or after waiting for certain duration of, can put itself on sleep.

Table 4.2. Four STATES in which robot at any time can be.

Note that, due to asynchrony, at any point in time different robots could be in different state, meaning one robot might be in deploying phase, whereas other robot might be collecting message to start focused deployment. In particular, all robots that wake up for the first time on the left or bottom borders, enter the **DEPLOYING** phase. In special cases some robots might be assigned the task of only focused coverage, such robots can directly begin with **COLLECTING** state.

We have now clearly defined the various states in which a robot can be at any point of execution. Now, we will briefly describe the basic concepts behind deadlock and how such scenarios be tackled.

As per definition, Deadlock is a problem that can exist when a group of processes compete for access to fixed resources. Deadlock can exist if and only if four conditions hold simultaneously:

- **Mutual exclusion:** at least one resource must be held in a non-sharable mode.
- **Hold and Wait:** there must be a process holding one resource and waiting for another.
- **No preemption:** resources cannot be preempted.
- **Circular wait:** there must exist a set of processes [p1, p2… pn] such that p1 is waiting for p2, p2 for p3, and so on and pn waits for p1….

There are three ways to deal with deadlock, they are:

- **Deadlock Prevention & Avoidance:** Ensure that the system will never enter a deadlock state

- **Deadlock Detection & Recovery:** Detect that a deadlock has occurred and recover
- **Deadlock Ignorance:** Pretend that deadlocks will never occur.

The first option is the best one to resolve deadlock, and to do that, we must ensure that at least one of the condition among Mutual exclusion, hold & wait, No preemption and circular wait does not happens in the application.

Now that we have clearly defined both the robot ***STATES*** and deadlock conditions, we now prove that there would not be any deadlock. To do so, we need to present four Lemmas i.e.

1. No robot can enter deadlock during **DEPLOYING** state.
2. No robot can enter deadlock during **COLLECTING**.
3. No robot enters deadlock during Focused coverage based deployment i.e. during **TRAVELING** state towards the point of interest
4. No robot can enter deadlock if any of the robot enters **SLEEPING** state.

- ***Lemma 1: No robot can enter deadlock during DEPLOYING state.***

  Proof: As stated earlier, a robot enters a ROI from the known point entering the first state i.e. **STATE-0 DEPLOYMENT**, it deploys the sensors in incremental fashion and at each step it drops a sensor and updates its memory with the deployment matrix. Thus, each robot is responsible of carrying out its own deployment independently and hence ruling out the case of deadlock. In this case we have applied the concept of **non-mutual exclusion i.e.** there is no resource which can result in deadlock.

- ***Lemma 2: No robot can enter deadlock during STATE-1 COLLECTING state.***

  Proof: After completion of the initial phase of deployment, all robots go into **STATE-1** ***COLLECTING*** phase in which they map already deployed sensors to a virtual matrix in their memory and waits for notification from sensors to initiate the focused coverage algorithm. Suppose there are ***"n"*** robots $\boldsymbol{R_1, R2, R3}_{....}\boldsymbol{R_n}$ and ***"i"*** nodes $\boldsymbol{N_1, N_2, N_{3.....}N_i}$ of the form (x^i, y^i), the assignment would follow the below equation (9).

$$[R_1, R_2, R_3, R_4 \ldots . R_n] = [N_1(x^1, y^1), N_2(x^2, y^2), N_3(x^3, y^3), N_4(x^4, y^4) \ \ldots \ N_i(x^i, y^i)] \quad \ldots\ldots (9)$$

When a robot receives notifications from all the ***"i"*** nodes, it evaluates and arrange all the nodes in the increasing order of their sequence and approaches to that particular node which it is responsible of. For example, if 50, 100 and 105 are the three nodes, which are to be covered in focused coverage phase by the robots R_1, R2, R3 and R_4. As per the above explained rule, the robot R_1 would be assigned node 50, R_2 assigned to 100 and R_3 assigned to 105 and robot R_4 would remain in ***COLLECTING*** phase, as it has not been assigned to any node. Hence, by following the above approach of synchronizing the assignment of node during COLLECTING phase and by applying the concept of ***non-Hold &Wait***, deadlocks are avoided.

- ***Lemma 3: No robot enters deadlock during Focused coverage deployment i.e. during STATE-2 TRAVELING state towards the point of interest***

Proof: **STATE-2 TRAVELING** state begins after each robot is assigned its respective point of interest from where they are to carry out F-Coverage algorithm. Each robot travels towards its assigned point (x^i, y^i) without any interference with any other robot. If by any chance, a case has been encountered where two or more robots are assigned the same reference points in which the robot with shortest distance would reach the point first and all other robots would go into COLLECTING phase. This process ensures that at any point of time only one robot is assigned to one point of interest. This process can be described with below pseudo code.

1: **For** all *i* robots **do**
2: $R_i = N_i(x^i, y^i)$
3: **end**
4: **if** $N_i(x^i, y^i)$ *is assigned to robot* > 0 **then**
5: (x^i, y^i) is assigned to robot with least T_d
6: **else**
7: $R_i = N_i(x^i, y^i)$
8: **end**

- ***Lemma 4: No robot can enter deadlock if any of the robot enters SLEEPING state.***

The robot enters **STATE 3 - SLEEPING** state in either of the two scenarios.

1. After completion of deployment.
2. To conserve the energy after waiting for certain duration of time without any communication.

In the first case, the robots completes the deployment with optimal network coverage and it notifies the Command & Control center and goes into SLEEP until it is waked-up for network restoration. In second case, the robot waits for a certain configurable timeout without being in communication, after which the robot notifies the Command & Control and goes into sleep. In both the scenarios there is no chance of deadlock.

Therefore, with above Lemmas proved, no deadlocks can occur during any execution of algorithms and ensure finite time completion.

State	Deadlock condition	Prevention technique
STATE 0 (DEPLOYING):	The robot is currently busy with deployment of sensors in the ROI.	**MUTUAL EXCLUSION.**
STATE 1 (COLLECTING):	The robot has completed the deployment and waiting for multi-hop messages from the sensors to continue the F-Coverage algorithm.	**NON-HOLD & WAIT**
STATE 2 (TRAVELLING)	Upon receiving an F-Coverage command from any sensor, the robot initiates it movement towards the request sensor to start focused coverage.	**NON-HOLD & WAIT**
STATE 3 (SLEEPING):	The robot after completion of deployment or to conserve the energy or after waiting for certain duration of	**NON CIRCULAR WAIT**

State	Deadlock condition	Prevention technique
	time without any communication can put itself on sleep.	

Table 4.3. Deadlock tackling techniques in multi-robot scenarios.

4.5.Multi-Robot coordination:

To solve this problem, we have deployed multiple E-puck robots in a single region of interest (ROI) and used divide and conquer methodology. We prove that the deployment results would definitely improve in multi robot coordination scenarios when compared to a single robot.

Each robot would be responsible for sensor deployment in its own region and all the robots communicate directly to command and control center. The main purpose of command and control system is to provide the robots with the start coordinates of deployment and also to provide the focal point to continue the F-Coverage deployment. The algorithms have been designed in a way to minimize the dependency of C&C center to maximum, however its inevitable dependency still exists in two of the aforementioned scenarios. A high degree of care and a backup plan should be in place to avoid single point of failure of command and control center.

CHAPTER 5

SIMULATION & REAL WORLD EXPERIMENTS

This chapter presents the implementation of SCAN algorithms and experimental setup at all the three environments i.e. on MATLAB, Webots and real robot levels. We initially conducted simulation experiments on MATLAB tool and then cross validated the results on WEBOTS tool and finally implemented our algorithms on real E-Puck robot. The first part of this chapter explains the mathematical simulated MATLAB implementation, in the second section we present the implementation on Webots robotics tool and in the final part we explain the implementation using a real E-Puck robot.

5.1. MATLAB Based Simulation:

Initially we conducted experiment on MATLAB. The desired area where the sensors have to be placed can be of random geometry; however we have set as a 200 x 200 sq. m. rectangle on which various static obstacles were placed. The sensing range of each sensor was uniformly set to Sr, and the robot uses this as its hop length. Sr was adjustable in a range of 10-50 meters. To explore the effectiveness and correctness of our algorithms, we tested them in random obstacle distributions in a sample of ten various scenarios and each scenario was executed fifty times to attain an optimum confidence interval. Table 5.1. presents concise description of the parameters used in our simulation.

Simulation Parameter	Value
Area (Sq.m.)	400 sq.m.
No. of Robots	1
Unit grid lengths	10 meter
Robot's sensing range	Adjustable
Sensing range	Between 10-50 meter
Obstacle Type	Rectangle
No. of obstacles	3
Confidence Interval	10

Table 5.1. Simulation parameters used in MATLAB experiments.

5.1.1. Basic SCAN and basic SCAN with Focused Coverage

This section provides implementation details with examples for both basic SCAN and basic SCAN with Focused Coverage. The mechanism of Basic SCAN was explained earlier in chapter 4 and Figure 5.1 shows the actual implementation of the basic SCAN algorithm in the MATLAB environment. Basic-SCAN terminates when the robot visits all the four corners at least once. Figure. 5.1 shows that the robot has visited all four co-ordinates, namely, (0,0)–(0,200)–(200,200)– (200,0). So, the robot will terminate running SCAN and stay at C3.

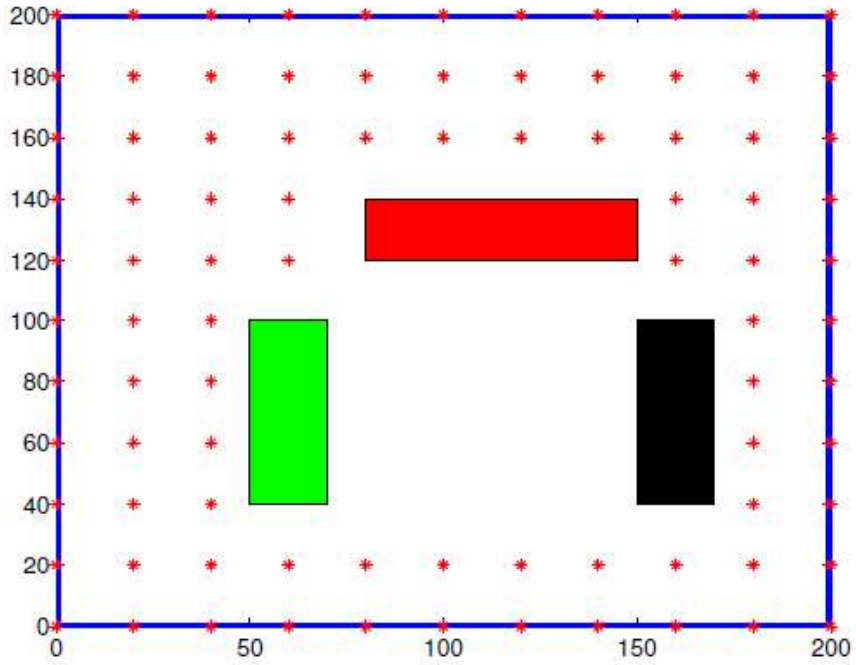

Figure 5.1. Sensor deployment using basic SCAN. The red, black and green rectangles represent obstacles. Red stars denote the sensors deployed by the robots. There is a blank space in the region, which will be addressed by the Focused Coverage algorithm. (Actual simulation view)

To increase the coverage the sensors now run Focused SCAN (SCAN-FC) to know the additional prospective locations where sensors can be placed. Figure 5.2 shows the actual simulation of the Basic SCAN FC algorithm.

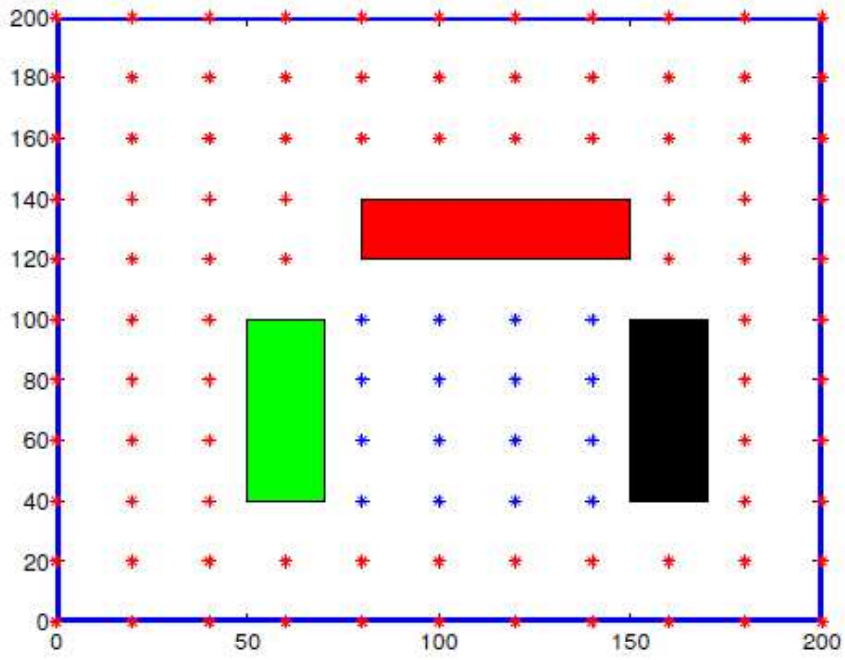

Figure 5.2. SCAN-FC sensor deployment. The blue stars represent the deployed sensors using focused SCAN. Blue sensors are placed after the initial SCAN algorithm is completed. (Actual simulation view)

5.1.2. Opportunistic SCAN and Opportunistic SCAN with Focused Coverage

This section provides the simulation overview for both Opportunistic SCAN (OPP-SCAN) and Opportunistic SCAN with Focused Coverage (OPP-SCANFC). In Fig. 4.6. We explained the detailed functioning of Opp-SCAN algorithm which is executed on MATLAB and Figure 5.3 shows the original simulation output. In Figure 5.3 the robot moves quite similarly like Basic SCAN except it does not go back when any obstacle and boundary is faced. The robot starts its tour from START (0,0) and reaches POS3(200,0). This time, the robot does not go back to START (0,0), instead, it looks for any other available route and finds one on the top.

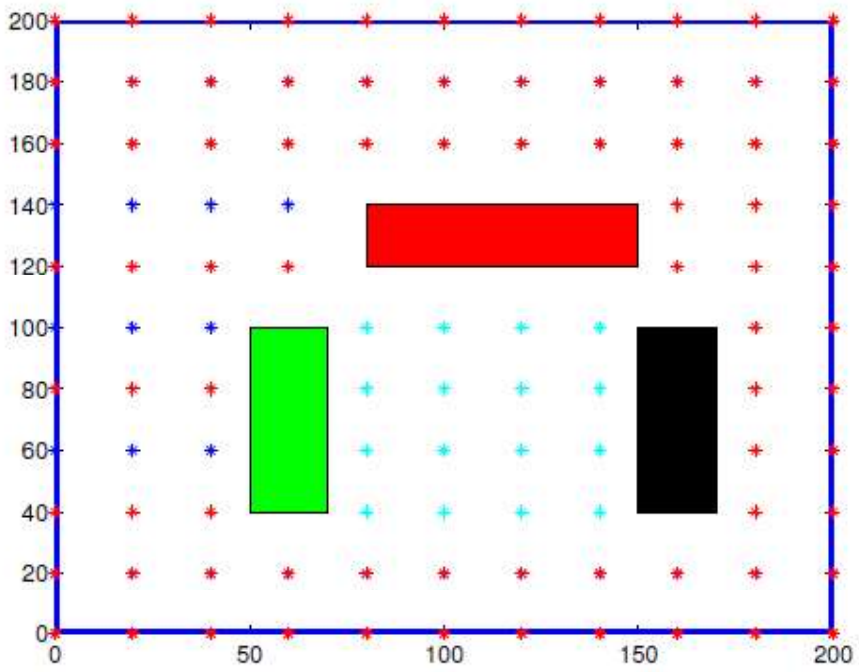

Figure 5.3. OP-SCAN-FC sensor deployment uses both Opportunistic SCAN and Focused coverage to ensure 100% coverage. (Actual simulation view)

5.2.WEBOTS – A REAL WORLD ROBOTICS TOOL

This chapter is dedicated towards providing the system architecture, design methodology and capabilities of WEBOTS tool. The first part of this chapter present the technical details of WEBOTS tool followed by the system architecture & components involved and finally the design methodology used in this work.

5.2.1. Overview of WEBOTS:

Webots is a powerful and industry standard mobile robotics tool used to program, control and implement various algorithms and deploy & run directly on real robots. It has global usage by more than twelve hundred universities, R & D centers and companies. It has rich features to

easily implement complex algorithms on simulation environments and convert the implementation into package which can directly be deployed on real robots. Thus, providing a great means to bridge the gap between the simulation and real world implementations. [97]

5.2.2. Key features of the WEBOTS tool:

- **Wireless Mobile actuator or robot prototyping:**

This feature provides numerous inbuilt and ready available robots like E-Puck, Khepera, Nao robot etc. Additionally, there is no limit on number of robots to be added in an environment allowing us to include as disparate robots in a single experiment and integrate them. Figure 5.4 shows a sample E-Puck robot highlighting the various key features available.

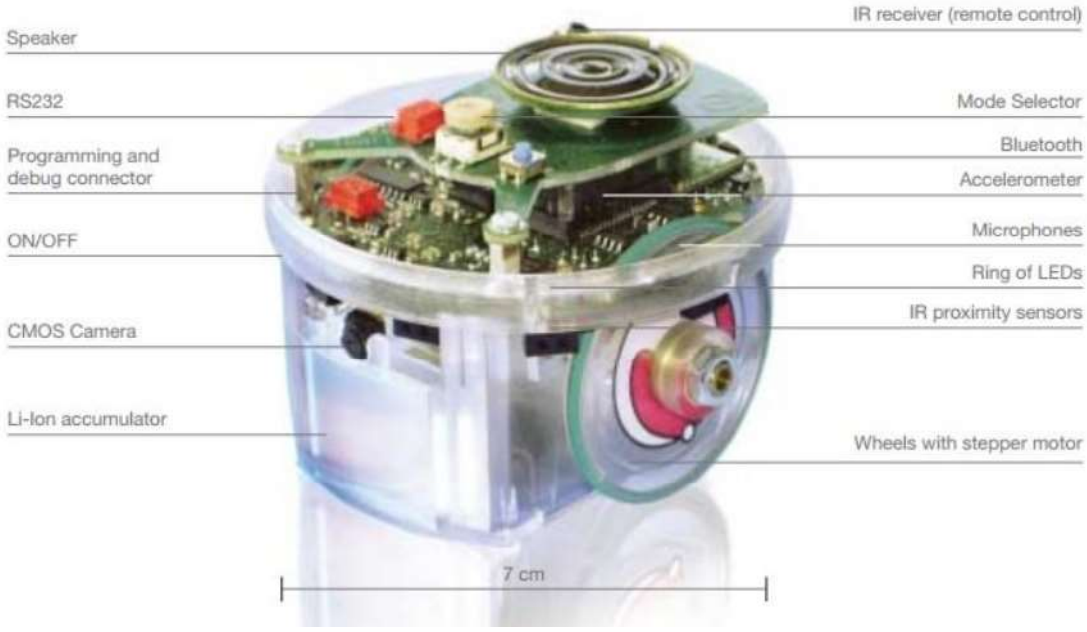

Figure 5.4. A sample of real E-puck mobile robot highlighting the key features of the robot. [97]

Figure 5.5. A sample team of E-puck robots. Image from (webuser.unicas.it)

- **Powerful API's and Robot libraries:**

The tool provides out-of-the-box support for 200+ powerful API's and support various programming languages like C, C++, Java, Python, MATLAB etc. to name few. It supports both object oriented as well as structure oriented programming paradigm. As part of this work, we have used C, C++, Python and JAVA programming languages to implement the algorithms. In case of single robot scenario we have used C programming language, but in case of multi-robot scenarios we have used various combinations of robots and programming languages like C, Java, C++ & Python.

Webots provides large collection of robot models like E-puck, Nao, Khepera, Boe-Bot, Pioneer, Katana, Shrimp, and HOAP-2 etc. This drastically reduces the implementation, testing and invest time as we can firstly implement the algorithms on the simulation environments and later move it to real robots.

- **ROS Interface (Robot Operating System):**

One of the most distinguishing feature of this tool its integration support with all major ROS stacks like OpenSLAM, Inverse kinematics, OpenCV, OpenSLAM, OpenRAVE , GMapping etc. This is done by using the roscpp (C++) or the rospy (Python) controller interfaces. This integration is however out of the scope of current work. However, as part of extension to this current work, we foresee a great work outcome if we integrate our current work with ROS related interfaces and environments.

- **Controllable environment:**

Webots provides a distinguishing feature called "**Supervisor Controller**" which supports all the features of Command & Control center allowing us to control the entire experimental environment like robotic communication, trajectory controlling, run time modification of robot properties & values, add & remove the objects etc.

- **Interactive 3D Simulation & State-of-the-art 3D graphics:**

Provide rich and interactive user interface with 3-Dimensional view of the environment. This feature very important to test the behavior, robustness, reliability and correctness of the algorithms before deploying it on the real robots. Additionally, this features allows us to create sample scenarios with various environmental conditions, obstacle distributions, robot collision scenarios and actual WSN formation.

- **Integration with Physic plugin programs:**

Webots provide excellent feature to simulate physics behaviors like sensors & actuators behavior, dynamics, uniform and non-uniform frictions etc. This is a feature of foremost importance especially when it comes to implementing a WSN based actuator networks.

- **Inter-Robot Communication, Sensor equipment's, Camera and Display devices:**

Webots is equipped with range of sensor devices like distance sensor, light sensor, cameras, LIDARs, GPS, gyro, accelerometer, compass, bumpers, position sensors, force-feedback sensor, etc. In the current work, we made use of distance and camera sensors. Distance sensor is the most important one as in any deployment algorithm, the most important task is to detect an obstacle which can be easily done using a distance sensor.

Additionally, we can make use of highly accurate and realistic camera devices on-board robots. This camera can be integrated with the programmed controller to capture the environmental information and process it. The tool provides enhanced features like adjusting the resolution and field of view, setup stereo-vision, spherical projections, white noise, pan-tilt systems, etc.

Lastly, we can utilize existing display devices like LCD & television and integrate it with robots program to display the robot condition, draw its trajectory, notify and modify some robot variable etc.

Above mentioned features are few of numerous features available in the tool. To summarize, Webots is a powerful and feature rich software and is a best fit to carry out all the activities of this work.

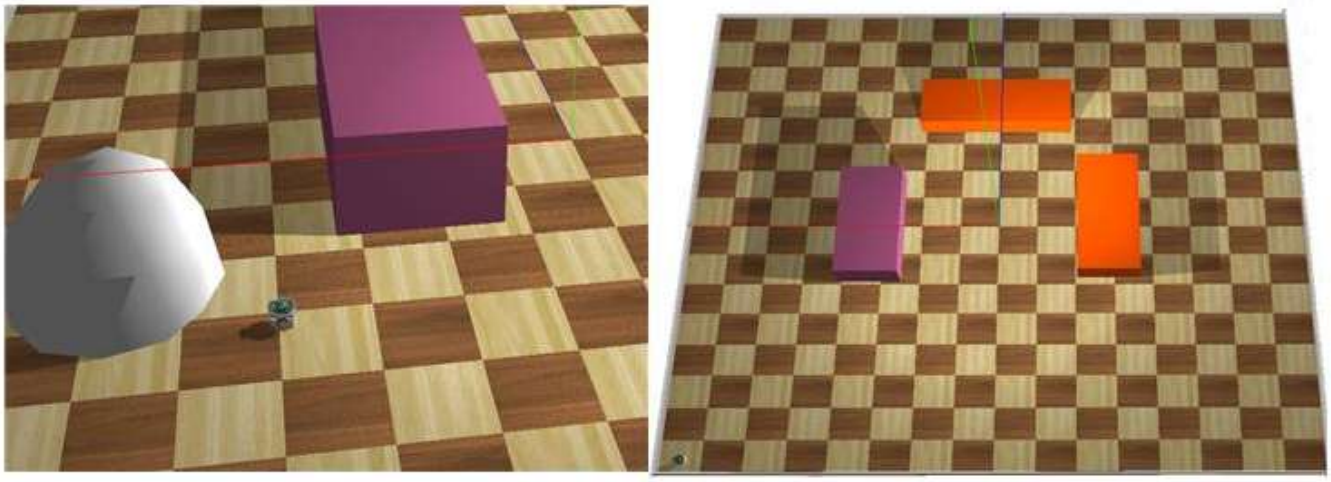

Figure 5.6. A sample Webots world with 3 obstacles and an E-puck robot.

5.2.3. SYSTEM ARCHITECTURE USING WEBOTS:

In this section we present comprehensive details of the system architecture of our proposed model and provide a mapping of model with Webots tool. The overall system architecture consists of four components, they are a world, a robot controller, a supervisor controller and an executor. We provide details of each component below.

- **World:**

A world, in Webots is referred to the real environment which in our case can refer to a region of interest (ROI) consisting of various obstacles, terrains and unexpected conditions. World is the main entity within which we can include various objects including robots and sensors.

All the physical properties of the environment, robots, objects etc. are defined in world. In other words, a world is the superset of the entire implementation.

- **Robot controller:**

A software program which runs on individual mobile robot responsible of realizing the actual implementation of sensor deployment algorithms. This is the component where the robot actually

initializes all the 8 IR sensors and keeps sensing the environment for boundaries / obstacles. Webots provides various APIs for programming logic to fulfill the requirements.

- **Supervisor controller:**

Acting as a command and control center for the entire setup, responsible of communicating with all the robots and capture the received data. Supervisor would also be responsible of initializing the deployment process by informing the robot about the initial coordinates of the robots. It also takes care of initializing the F-Coverage deployment post completion of Basic SCAN algorithm

- **Executor:**

Takes care of running all the programs which basically is a classic program compiler used to decode and compile the program. In our work we have used C-programming compiler and Java compiler. Figure 5.7 depicts the system architecture of our experiment.

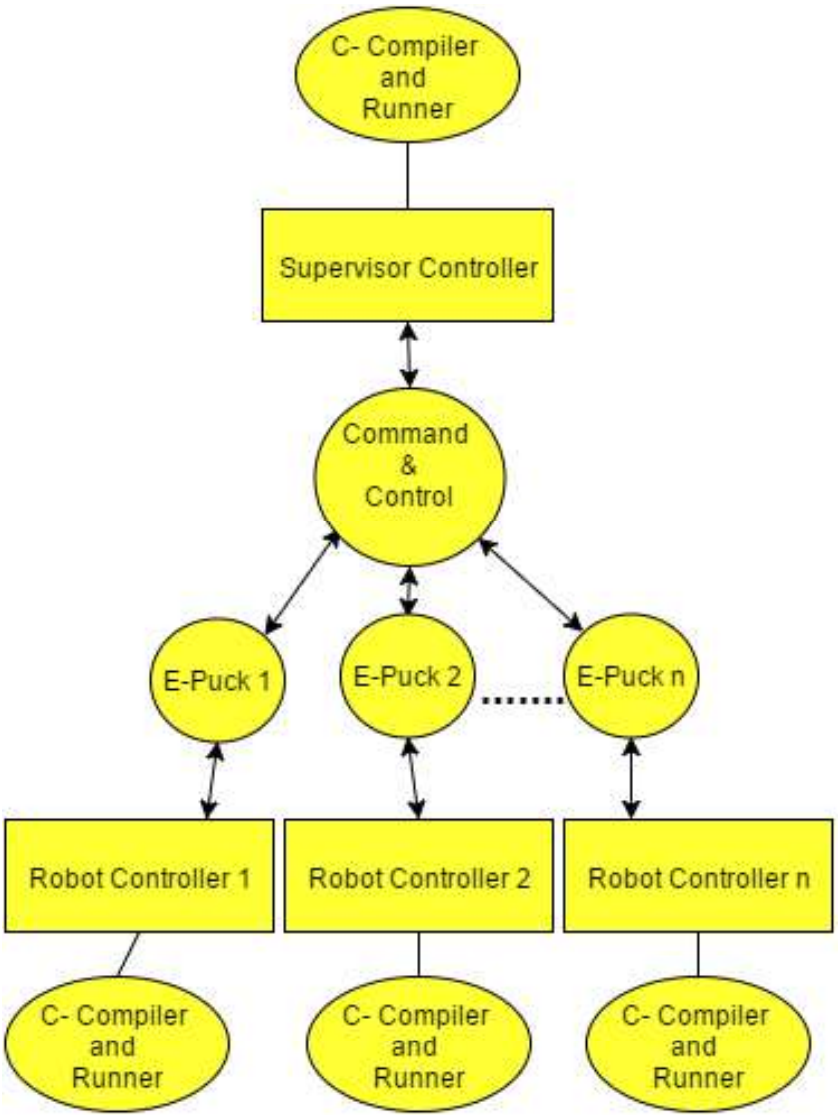

Figure 5.7. Describes the system architecture of Webots environment which includes multiple robots, their controllers and a supervisor.

5.2.4. WEBOTS based Experimental Setup:

To implement the designed algorithms we use Webots tool which provides end-to-end features to program debug, execute and cross-compile the application. Our experiment was conducted two phases, in the first phase we used Webots to implement and in the second phase we used real robots setup and deployed our implemented algorithms on them and evaluated the results.

Webots provides strong features to use various kinds of robots out of which we have used E-Puck robots fully equipped with 8 distance and range sensors acting as a mobile robot responsible for sensor deployment.

The Region of interest can be of any variable size, dimensions and geometry; however we have selected an area of 10 x 10 sq.mtr. The obstacle distribution has been randomized so that the implementation could be scalable and would perform as expected in any ROI with any combinations of obstacle distributions. The sensing range of each sensor can be any value, but for the sake of uniformity we have set to Sr, and the robot uses this as its hop length. Sr was adjustable in a range of 10-50 mm.

To ensure the correctness and reliability of the algorithms we have tested and validated the results in 50 different setups with varying ROI dimensions and varying obstacle number, shape and distributions. For the sake of documentation we have used five out of those 50 scenarios. Table 5.2. shows the various parameters used in the five of the scenarios.

Environment Params	Scenario – 1	Scenario -2	Scenario -3	Scenario -4	Scenario -5
Area (Sq.m.)	200 * 200	300 * 300	200 * 400	500 * 500	300 * 600
No. of Robots	1	1	2	2	4
Unit grid lengths	10	20	10	30	40
Robot's sensing range	Adjustable	Adjustable	Adjustable	Adjustable	Adjustable
Sensing range	10	10	10	20	30
Obstacle type	Symmetric	Symmetric	Asymmetric	Asymmetric	Symmetric
No. of Obstacles	3	3	5	5	3

Table 5.2. Evaluation parameters used for all the four versions of SCAN algorithms experiment.

The above parameters are just for the result comparison, as our algorithm design and implementation are done in a way that can be tested in dynamic environment with various obstacles and sensing ranges.

5.2.5. Implementation of Basic SCAN and Basic SCAN with F-Coverage:

In this section we provide the implementation details of both BASIC SCAN and BASIC SCAN with Focused Coverage. Figure 5.8 shows a sample Webots world with three random obstacles and an E-puck robot entering the environment from the START (0, 0) position. The figure represent the expect trajectory of the robot during the sensor deployment. The solid black arrow refers to the robots forward path, and the dashed red arrow refers to the backward path. Blue arrows represents the robots downwards movement. (View pre-running the simulation). The Green arrows represents the Focused coverage movement post completion of Basic SCAN. Figure 5.9 shows the actual implementation of the Basic SCAN algorithm. After completion of BASIC SCAN algorithm, the robot initiates the F-Coverage algorithm. To do so, all the nodes with blank neighbors send a beacon message to the location (0,0) indicating the presence of hole. The robot then identifies the node with minimum sensor ID and initiates the focused coverage from that coordinate. Figure 5.10 shows the result post running the F-Coverage algorithm.

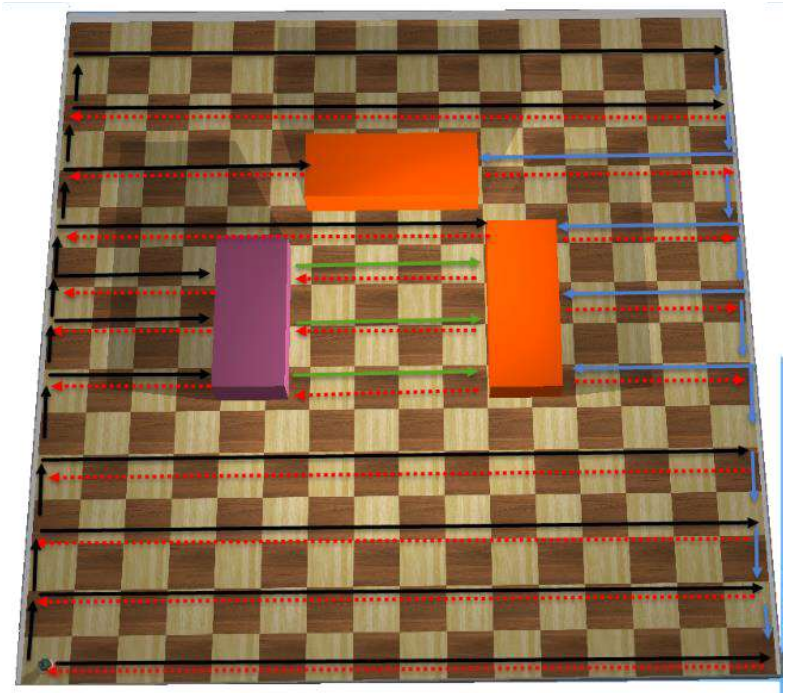

Figure 5.8. Basic SCAN deployment by a robot. The solid black arrow refers to the robots forward path, and the dashed red arrow refers to the backward path. Blue arrows represents the robots downwards movement. (View pre-running the simulation). The Green arrows represents the Focused coverage movement post completion of Basic SCAN.

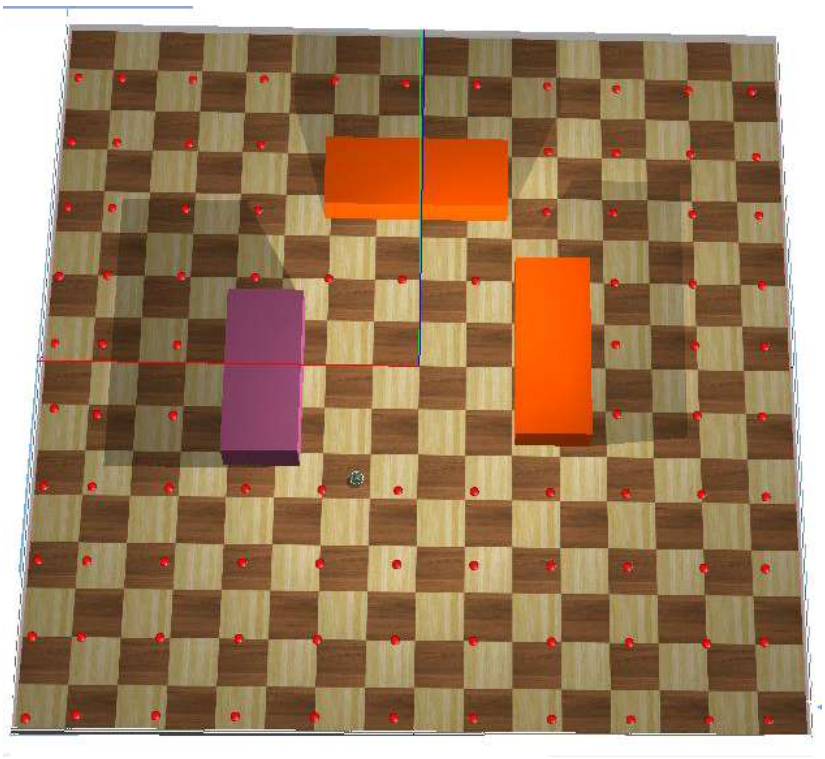

Figure 5.9. Actual view of sensor deployment using Basic SCAN.

Figure 5.9. is the result of running F-Coverage algorithm, it is clear from the figure that the central area which was previously uncovered is now deployed with sensors. Hence, we can conclude that by executing both BASIC SCAN & F-Coverage algorithm, we obtain optimal deployment results with lesser execution time and in any dynamically varying environment. However, the time required to complete the execution is relative higher, as the robot moves back and forth several times to complete the deployment task.

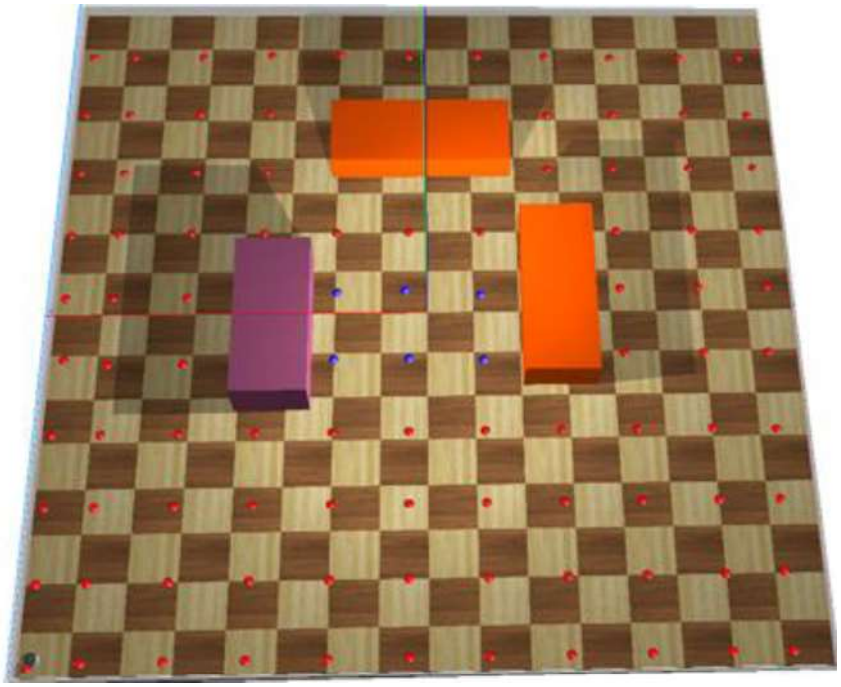

Figure 5. 10. SCAN-FC sensor deployment. The blue objects represent the deployed sensors using focused SCAN. Blue sensors are placed after the initial SCAN algorithm is completed. (Actual experimental view).

5.2.6. Opportunistic SCAN and Opportunistic SCAN with F-Coverage:

This section provides the implementation results of Opportunistic SCAN (OPP-SCAN) and Opportunistic SCAN with Focused Coverage (OPP-SCANFC). Figure 5.11 shows the theoretical view of the algorithm whereas Figure 5.12 shows the result post execution of the algorithm. Once the OPP-SCAN completes we can see that there are still some gaps in the ROI which would be covered by F-Coverage version of the algorithm. Figure 5.13 shows the result of OPP-SCAN with F-Coverage.

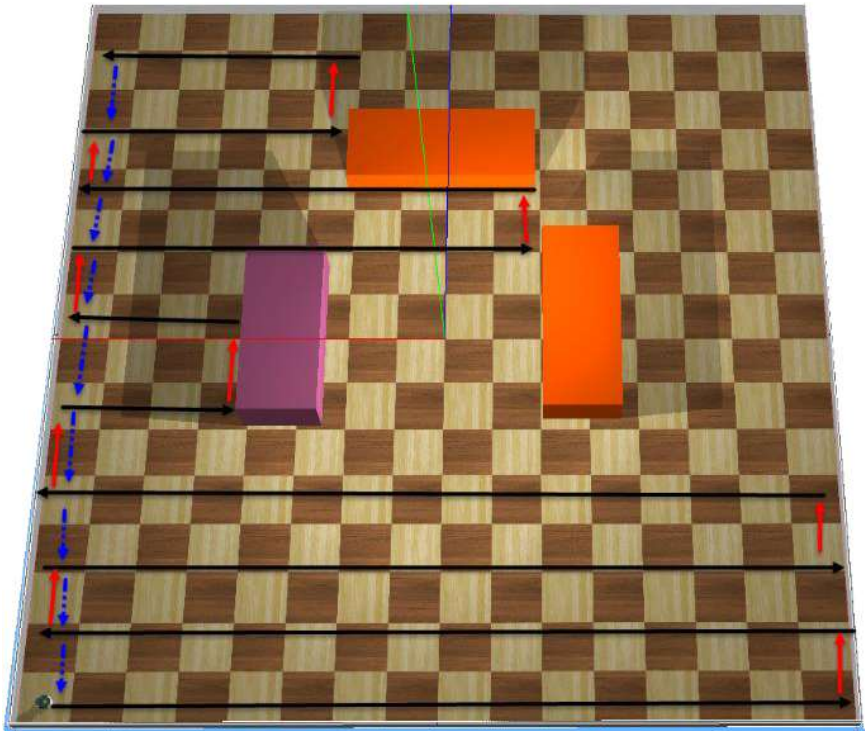

Figure 5.11. Theoretical view Opportunistic SCAN deployment by robot, shows less coverage compared to basic SCAN algorithm implementation for this particular obstacle distribution.

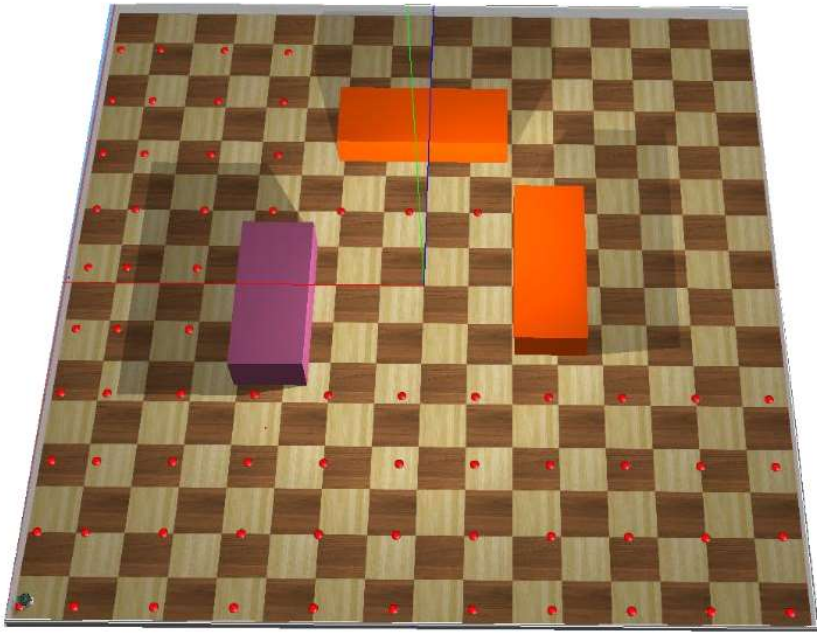

Figure 5.12. Theoretical view Opportunistic SCAN deployment after completion.

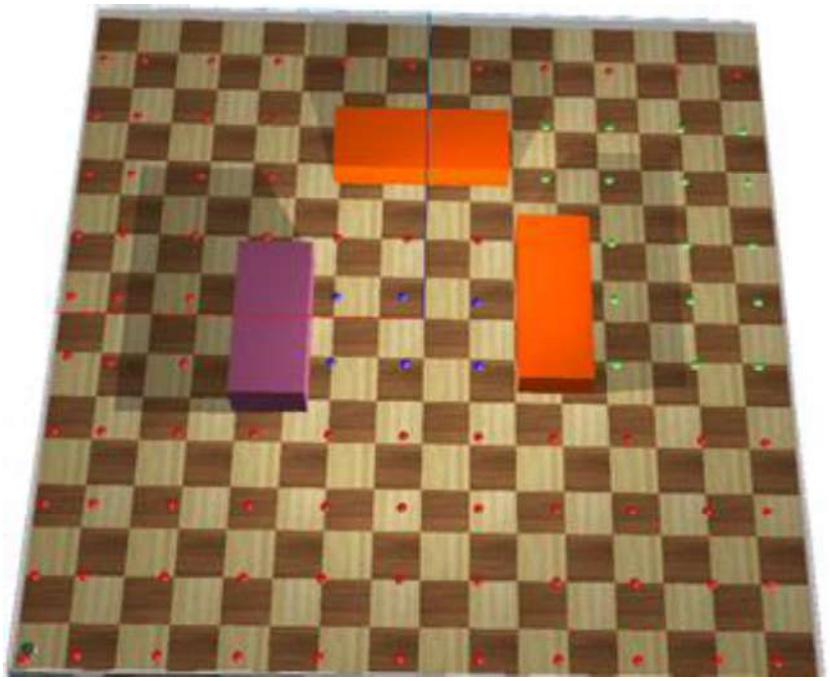

Figure 5.13.OP-SCAN-FC sensor deployment uses both opportunistic SCAN and Focused coverage to ensure 100% coverage. (Actual simulation view)

5.2.7. Implementation of Backtracking based sensor deployment (BTD):

This section provides a theoretical overview and simulation overview for back tracking based deployment (BTD). As per [52] each deployed sensor stores three pieces of information, sequence number, color, and back pointer. A sensor colors itself white if it is adjacent to an empty point and black otherwise. It updates its own color dynamically. This means that a white sensor may become black, as the robot continues to place sensors throughout the ROI. Figure 5.14 presents the original simulation output of BTD algorithm. The robot moves quite similarly as when using SCAN, except it changes the color of the placed sensors to black as soon as all the sensors has all the neighbors filled. Once the robot stuck at a particular location or reaches the maximum coordinates it finds the white color sensor node with minimum ID and continues it

movement from that location. This scheme of continuously finding the minimum white node and placing the sensor is repeated until all the nodes become black. There are various shortcomings of this algorithm, the most important of which is the robot's deadlock condition. In certain cases the robot loses its back pointer which results in the deadlock of the entire deployment scenario. Another shortcoming is that of very high deployment time and lower coverage. The comparison results of all algorithms are presented in the next chapter.

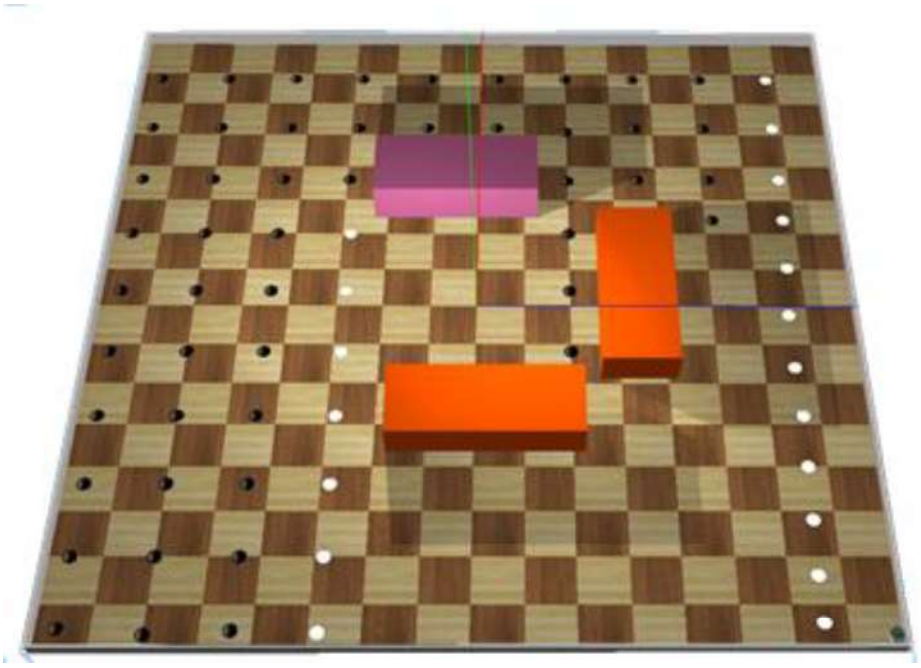

Figure 5.14. BTD sensor deployment, showing more empty spaces than all the above mentioned deployment due to the opportunistic behavior. When the robot reaches (200,200), it stops because it has already visited four corners and cannot find any empty nodes in its neighborhood.

CHAPTER 6

PERFORMANCE EVALUATION

This section provides the performance analysis of Basic SCAN, Basic SCAN - FC, Opp-SCAN Opp-SCAN-FC and Back Tracking deployment (BTD) algorithms against four metrics i.e. Coverage, Total distance travelled by robot, Message overhead and Deployment time. The presented performance evaluation are for both MATLAB and WEBOTS environment. In each section of the chapter, the evaluation is presented first for MATLAB followed by Webots. The chapter is divided into four sections, in the first section we present the various metrics of comparison and compare the performance of Basic SCAN against Basic SCAN with FC. In the second section we compare Opportunistic SCAN against Opportunistic SCAN with FC. In the third section we compare the performance of all Basic SCAN, Basic SCAN-FC, Opportunistic SCAN and Opportunistic SCAN-FC. Finally, in the last section we compare the performance of all the algorithms against the classic Back Tracking Deployment (BTD) algorithm.

6.1. Comparison Metrics:

- **Coverage:**

Coverage in wireless sensor networks is usually defined as a measure of how well and for how long the sensors are able to observe the physical space. [98] Coverage and connectivity are two of the most fundamental issues in WSNs, which have a great impact on the performance of WSNs. [99] In case of sensor deployment in Wireless Sensor and Robotics Networks (WSRN), coverage can be defined as the ratio of total number of deployed sensors to the total number of optimal sensors to get best results.

In simple equation, the coverage can be represented as

$$Coverage = \sum Deployed\ sensors \Big/ Total\ no\ of\ sensor\ to\ be\ deployed \quad \ldots\ldots (10)$$

$$C = \frac{n_d}{N - n_{ol}} \ldots\ldots (11)$$

$$n_{ol} = \frac{A_o}{S_r} \quad \ldots\ldots (12)$$

Where, C = Coverage, n_{ol} = number of sensors that could be placed if obstacles were not present. A_0 = total area covered by the obstacles, and nd = number of sensors actually placed, and N = total number of sensors those have to be placed if there were no obstacles.

- **Robot Distance Travelled:**

This is one of the most important criteria for any algorithm selection, as robot's energy consumption is heavily dependent on its distance travelled. An ideal deployment algorithm needs to cover maximum deployment area with minimum travelled distance. As mentioned earlier, the total distance travelled by the robot is denoted by the equation

$$\sum_{c=1}^{4} \sum_{1}^{n} D_n = n \times S_r, \quad \forall \in R \quad \ldots..(13)$$

- **Optimal distance for ideal deployment algorithm:**

The optimal distance for any deployment algorithm represents an approximate distance which a robot travels to obtain optimum coverage. In this section we would derive an equation for such an optimum distance value. This equation would act as baseline and measuring scale in providing an approximate best case distance.

Consider a region of interest of dimension "W x L" (W – Width & L – Length).

Let ***"O"*** represents the configurable deployment offset.

The factor $Width/_{Offset}$ => $W/_{O}$ **re**presents the total number of times a robot would horizontal traverse the RoI.

Thus, multiplying this value with length of RoI would provide the total distance travelled by robot to achieve the deployment.

$$\boldsymbol{Distance\ (D) = \frac{Width}{Offset} \times Length} \quad \text{..... (14)}$$

$$\textbf{i.e. } \boldsymbol{Distance\ (D) = \frac{W}{O} \times L} \quad \text{..... (15)}$$

Let us consider that there are some obstacles in the RoI with width W_o L_o.

Thus, this area covered by the obstacles is given by the equation.

$$\boldsymbol{Obstacle\ (D_o) = \frac{Width\ of\ Obstacle}{Offset} \times Length\ of\ Obstacle} \quad \text{..... (16)}$$

$$\textbf{i.e. } \boldsymbol{Distance\ (D) = \frac{W_o}{O} \times L_o} \text{ ..... (17)}$$

The above distance has to be subtracted from the original distance as this distance would not be covered by the robot.

$$\boldsymbol{Distance\ (D) = \left\{\frac{Width}{Offset} \times Length - \sum_{i=1}^{m} \frac{Width\ of\ Obstacle}{Offset} \times Length\ of\ Obstacle\right\}} \text{ ... (18)}$$

$$\textbf{i.e. } \boldsymbol{Distance\ (D) = approx\left\{\frac{W}{O} \times L - \sum_{i=1}^{m} \frac{W_o^i}{O} \times L_o^i + n \times o\right\}} \text{ ... (19)}$$

Where D -> Robot distance travelled.

W-> Width of RoI

L-> Length of RoI

O -> Robot offset.

m -> No. of obstacles.

W_o^i -> Width of i^{th} obstacle

L_o^i -> Length of i[th] obstacle

Example: Let us explain the above equation with a sample scenario. Consider a RoI of dimensions 100 meters × 100 meters and let offset be 10 meters with two obstacles of dimension 20 meters × 30 meters and 10 meters × 10 meters. Substituting the values in above equation.

$$Distance\ (D) = approx\left\{\frac{100}{10} \times 100 - \frac{20}{10} \times 30 - \frac{10}{10} \times 10 + 10 \times 10\right\}$$

$$Distance\ (D) = 1000 - 60 - 10 + 100$$

$$Distance\ (D) = 1030 \text{ meters.}$$

Thus, the optimum distance required for the robot to complete the deployment would be 930 meters.

- **Message Overhead:**

  Message overhead refers to total number of messages sent from any point of ROI to the START position (Command & Control center). In areas where communication back bone is not strong, the communication happens through multihop sensor to sensor forward. Hence, in such cases message overhead might congest the network. Therefore, ideally the lower the message overhead the better would be the algorithm performance.

- **Deployment / Execution time:**

  Execution time refers to the amount of time consumed by the algorithm to complete the deployment task. In our work, we have proved that OPP-SCAN algorithm is the most optimal algorithm and completes faster with more coverage and lesser message overhead and the most expensive is BTD algorithm.

6.2. Performance evaluation of Basic SCAN & Opp-SCAN algorithms:

This section provides the performance comparison of the Basic SCAN and Opp-SCAN algorithms in terms of coverage, robot distance travelled, message overhead and execution time.

6.2.1. Robot Distance Travelled (MATLAB & WEBOTS):

Figure 6.1 shows the comparison result of Basic SCAN & Opp-SCAN algorithms in terms of total distance travelled by the robot after the initial round. We compared the distance with various sensing ranges (S_r) of the sensor, which is the step size of the robot. Opportunistic SCAN performs better i.e. the total distance travelled by robot is always less for any sensing range (10-45 m).

As the sensing range increases the total travelled distance decreases for both algorithms, as expected, because the total number of points to be covered with sensors becomes less. As explained in the earlier chapters, the total distance travelled by robot in case of Opp-SCAN would be much lesser than the Basic SCAN algorithm as in Basic SCAN algorithm, the robot moves back and forth in the ROI whereas in Opp-SCAN algorithm the robot adds offset and moves to the next level instead of moving back and forth.

An important observation from the graph is that there is a sharp fluctuation when the sensing range is set to 40 i.e. both SCAN & Opp-SCAN algorithm takes higher robot distance at a sensing range of 40. The reason behind this fluctuation, was that the obstacle distribution in ROI didn't support the sensing value of 40. So it is very important to consider a suitable sensing range corresponding to the obstacle distribution. The adjustment of sensing range plays a critical role in obtaining optimal results.

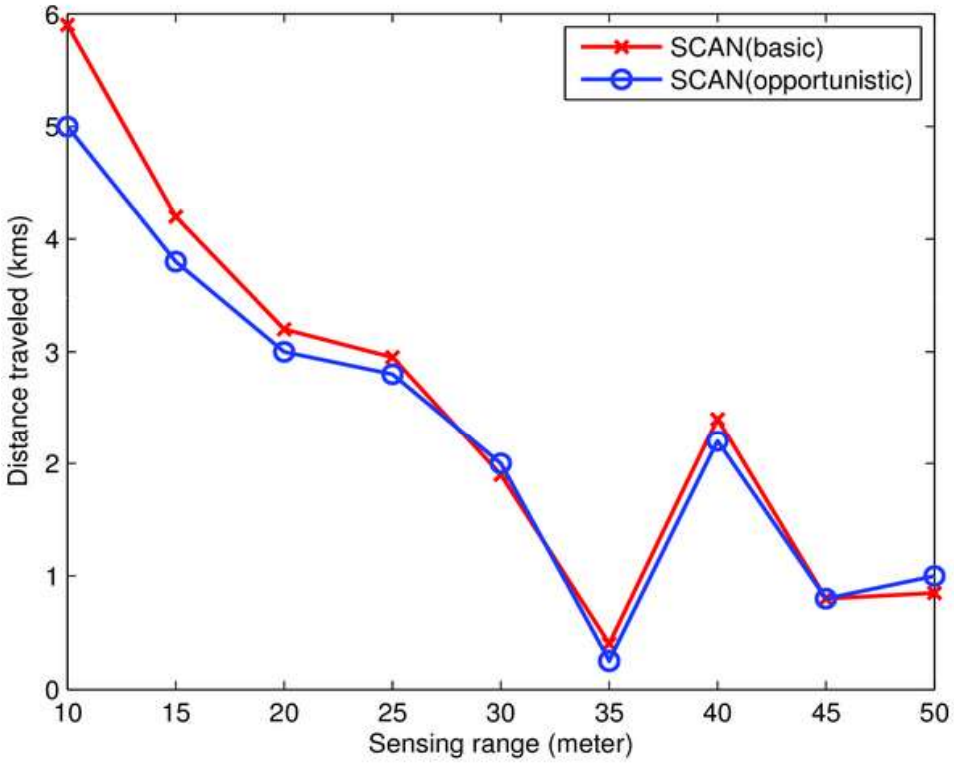

Figure 6.1. MATAB Results for Basic SCAN and Opportunistic SCAN algorithm performance in terms of distance travelled.

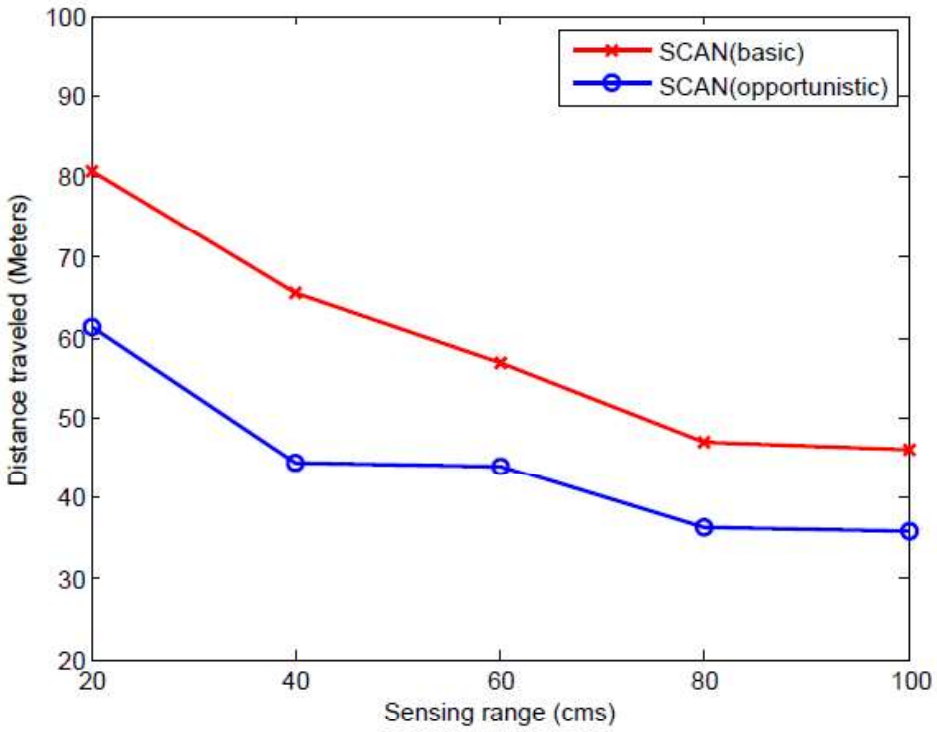

Figure 6.2. WEBOTS Results for Basic SCAN and Opportunistic SCAN algorithm performance in terms of distance travelled.

Hence, from the above graphs it is evident that for all the sensing ranges from 10 to 35, the basic scan always require higher robot distance to complete the deployment. Whereas the Opp-SCAN algorithm requires lesser robot distance. Thus, we can conclude that the Opp-SCAN algorithm performs better than Basic SCAN algorithm in terms of distance.

6.2.2. Coverage:

Figure 6.3 shows the sensing range versus coverage plot. The coverage for both SCAN (Basic) and SCAN (Opportunistic) performance converges when the sensing range becomes greater than certain value i.e. 50 cms in this case. So, there are trade-offs for using them for different requirements. If the desired result is to have the maximum coverage with short sensing range and shorter traveling distance, Opportunistic version works well.

It is worth noting that the coverage provided by Basic SCAN algorithm is better than Opp-SCAN algorithm at all sensing ranges, however as discussed earlier, there is a trade-offs between the coverage and total distance travelled.

Another important point to be noticed from the graph is a sharp drop in the coverage at a sensing range of 30 cms. This drop in coverage for both Basic & Opportunistic SCAN algorithms is due to certain obstacle distribution which doesn't favor the robot for sensing range of 30 cms.

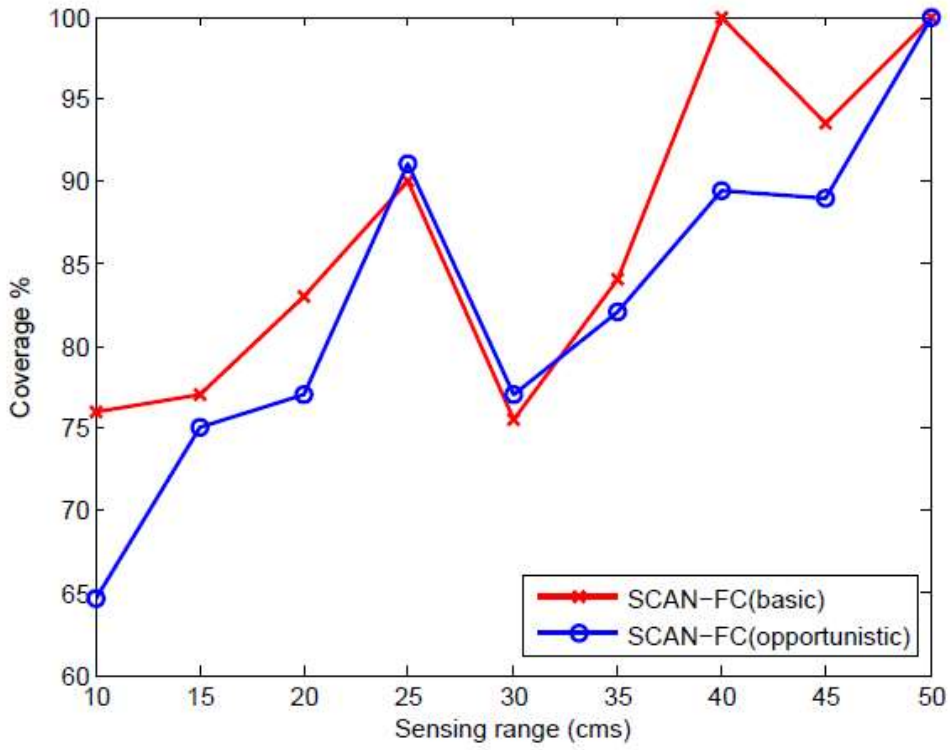

Figure 6.3. MATLAB Results for coverage percentage of BASIC SCAN and Opp-SCAN. Basic SCAN performs provides better performance in terms of coverage.

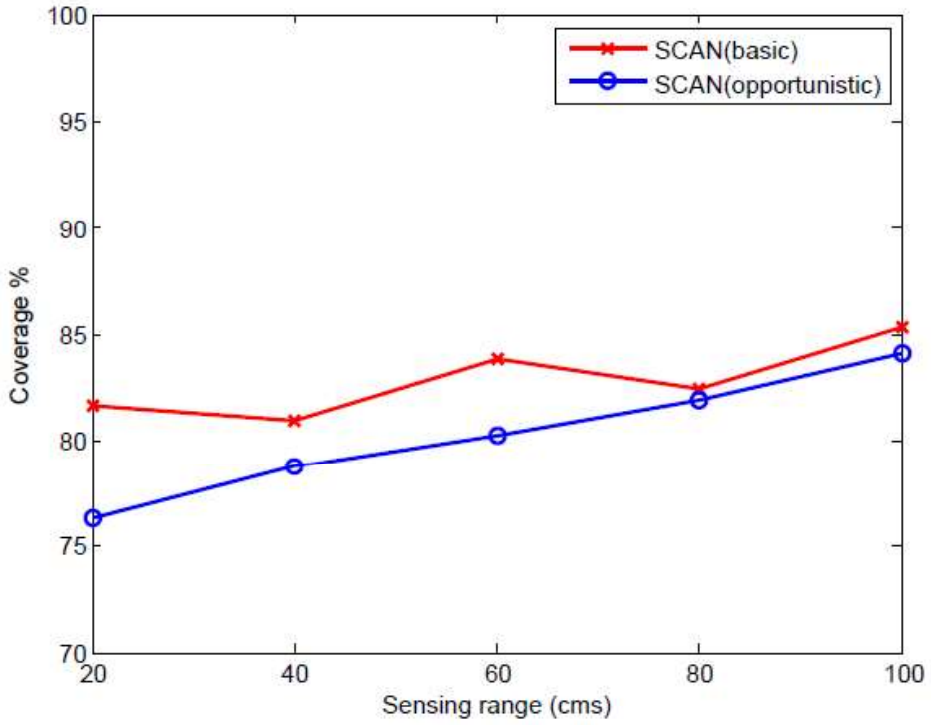

Figure 6.4. WEBOTS Results for coverage percentage of BASIC SCAN and Opp-SCAN. Basic SCAN performs provides better performance in terms of coverage.

6.2.3. Message Overhead:

The comparison result of Basic & Opp – SCAN algorithms in terms of message overhead is represented in Figure 6.5. We have calculated the overhead messages from the notification messages (either for hole or failure messages) sent to the START. The unit of measurement is the count of messages being generated from the sensors to the START point. This result demonstrates that opportunistic SCAN provides more overhead, the reason behind this behavior is that the Opportunistic SCAN algorithm provides lesser coverage, which implies there are more holes in network, which results in higher number of empty neighbors for sensor, which indeed results in higher messages flow in the network which ultimately increases the message overhead.

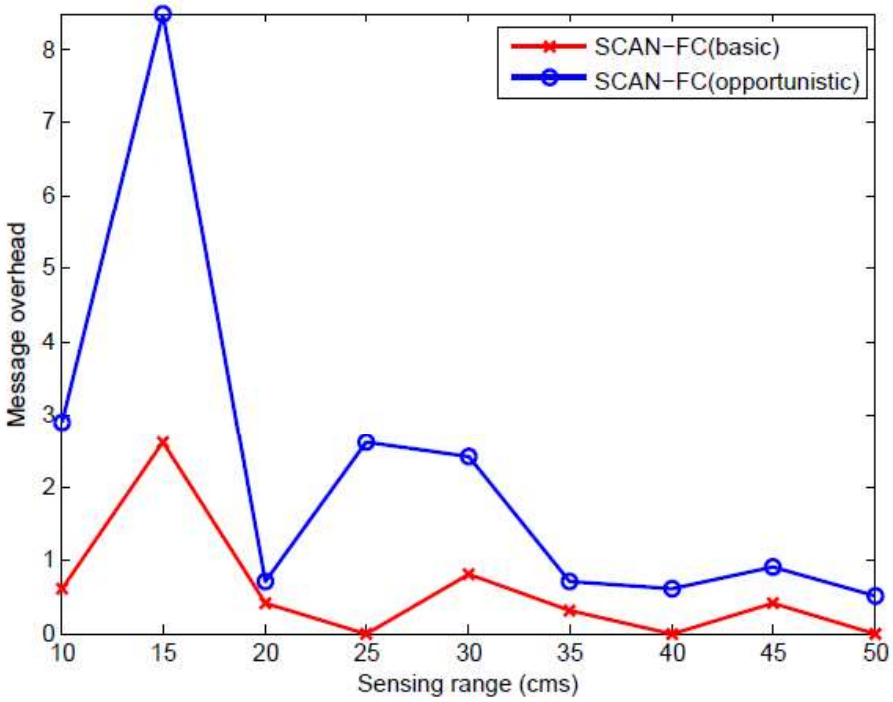

Figure 6.5. Message overhead by the variant of F-coverage algorithm execution on sensor nodes. This graph shows the additional message exchange to ensure extended deployment in the yet unexplored region.

For example, the previous situation arises for Opportunistic SCAN when the robot has already reached the four corners of the area and stays at some corner from which it can only go back to its START location. But, this will vary due to the different obstacle locations and their geometric properties. This abnormal behavior occurs due to the initial opportunistic search in the OPP-SCAN algorithm. OPP-SCAN enforces the robot to travel in the ROI in such a way that it can avoid following reverse path. This result in leaving more empty points compared to the basic SCAN algorithm. Thus, the overhead messages sent by the sensors becomes more than the basic SCAN algorithm which is very rare.

An important observation from the result is that, if the sensing range is set to 15 cms then there is an exponential increase in the messages being generated, this is due the fact that at this particular sensing range, the total number of sensors being deployed are equally distributed in upward and downwards deployment, which means, during the deployment, the robots deploy equal number of sensor while going upwards and coming downwards which implies that there is huge communication in both the directions.

6.2.4. Simulation Time:

Figure 6.6. shows the simulation time required for all four versions of our algorithm on MATLAB environment and Figure 6.7. shows the simulation time on WEBOTS environment. Basic SCAN spends most of the time traveling. Additionally, robot travels without fetching any placed sensors twice for almost 95% of its first tour. This is one of the strongest reasons for having less travel distance in Opportunistic SCAN. On the contrary, Basic SCAN passes by the already deployed sensors at least twice. It is worth noting that the deployment type drops sharply when the sensing range is set to 15 cms and keeps reducing and finally becomes stable at a sensing range of 50 cms.

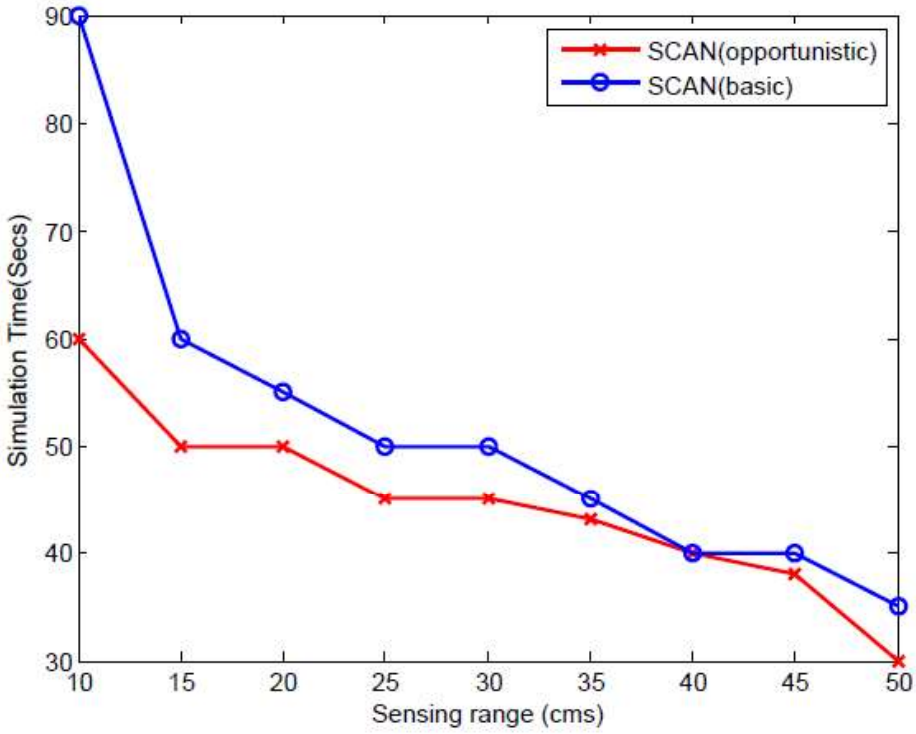

Figure 6.6. MATLAB results for Simulation time comparison for Basic SCAN & Opp-SCAN.

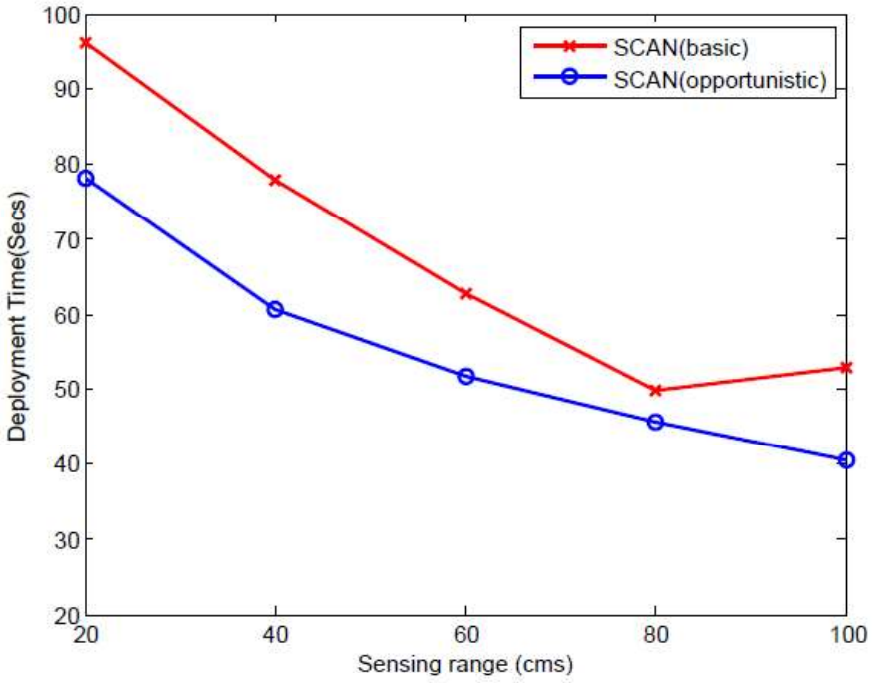

Figure 6.7. WEBOTS based results for Simulation time comparison for Basic SCAN & Opp-SCAN.

6.3. Performance evaluation of SCAN-FC, Opp-SCAN-FC & BTD algorithms MATLAB & WEBOTS:

In this section we present the comparison of focused coverage version of both SCAN and Opp-SCAN algorithm. The reason of having this detail as a separate section is to clearly compare the results of our algorithms and the BTD algorithm and also to explain the effect of applying the concept of F-Coverage algorithm on both the versions of SCAN algorithms.

6.3.1. Robot Distance Travelled (MATAB & WEBOTS):

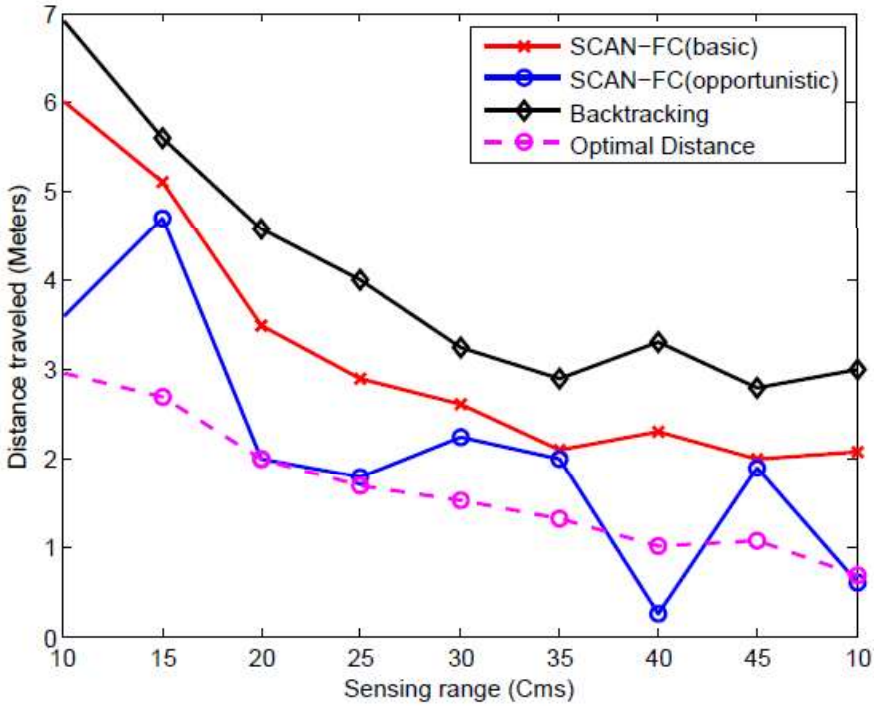

Figure 6.8. MATLAB Results of Basic SCAN and Opportunistic SCAN algorithm performance in terms of distance travelled.

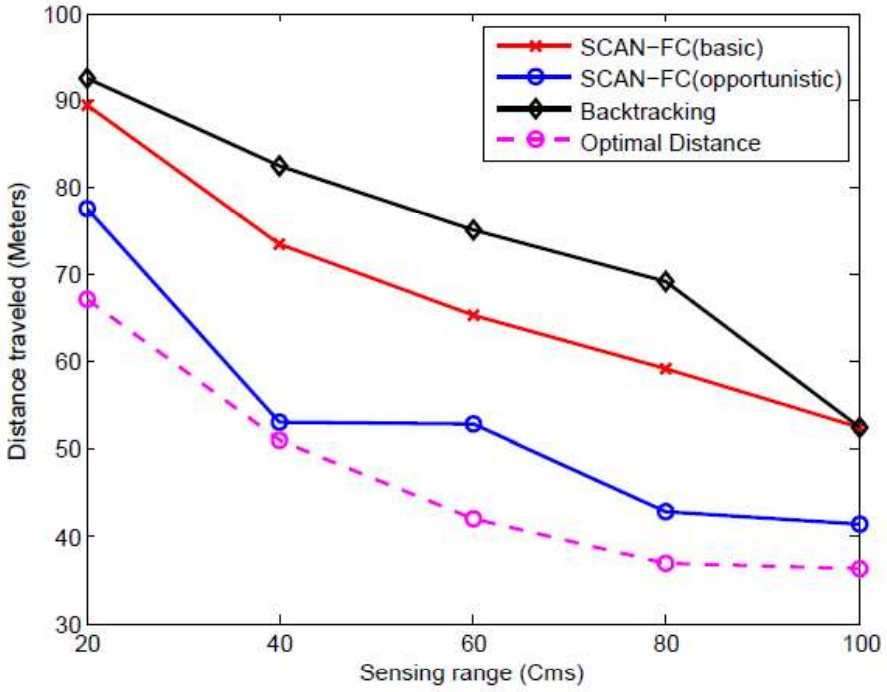

Figure 6.9. WEBOTS Results of Basic SCAN and Opportunistic SCAN algorithm performance in terms of distance travelled.

As show in the figure 6.8. as the sensing range increases the total distance required by the robot to complete the deployment reduces. It is also clear that opportunistic version of SCAN-FC outperforms BTD and SCAN-FC basic.

6.3.2. Coverage (MATLAB & WEBOTS):

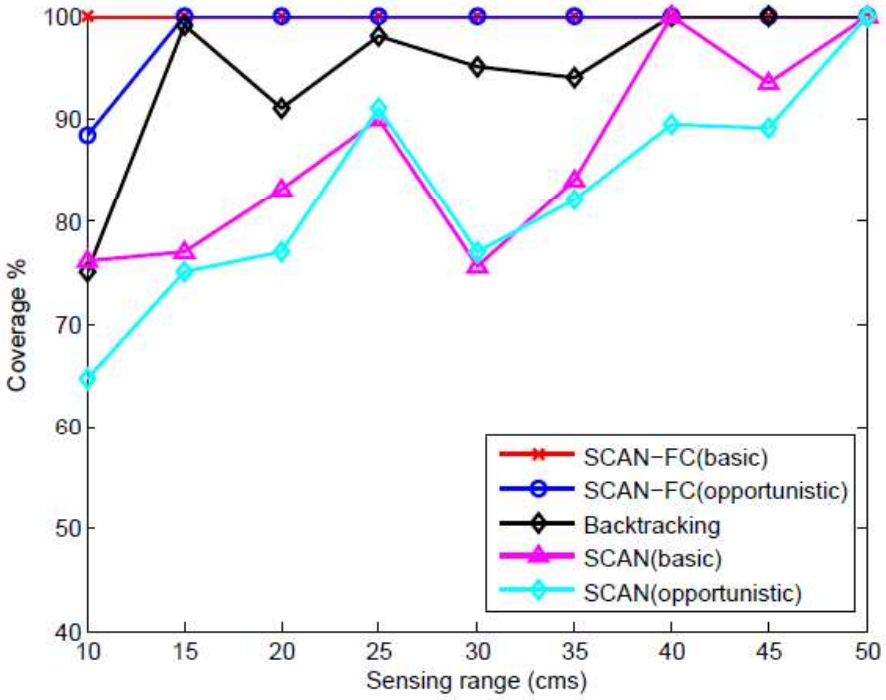

Figure 6.10. MATAB results of Coverage percentage using each of the four versions of SCAN. SCAN-FC for both the basic and opportunistic algorithms provides better performance in this particular scenario.

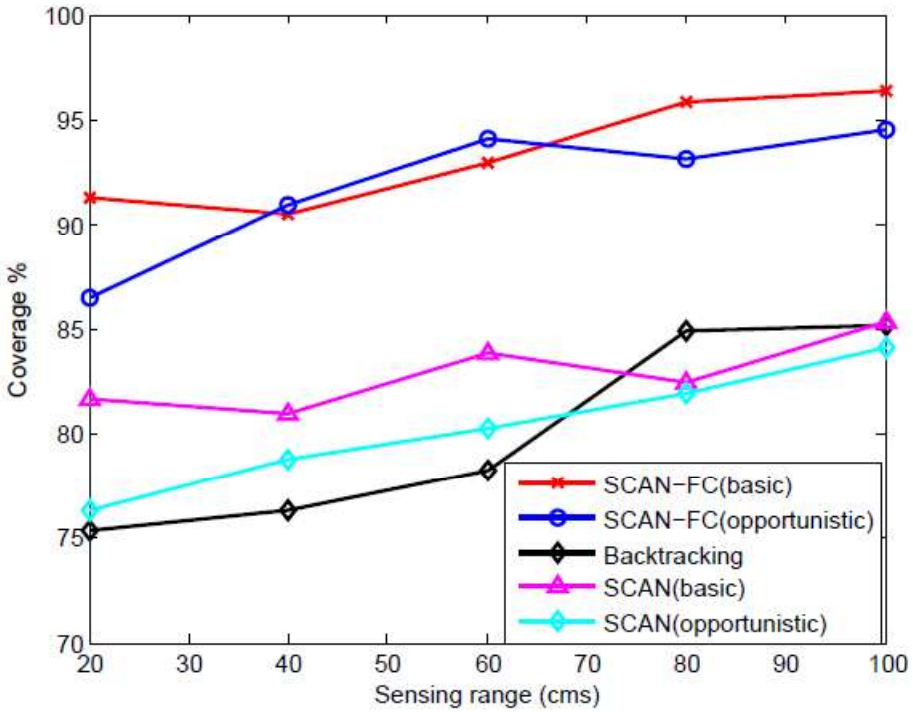

Figure 6.11. WEBOTS results of Coverage percentage using each of the four versions of SCAN. SCAN-FC for both the basic and opportunistic algorithms provides better performance in this particular scenario.

When it comes to coverage, the Basic SCAN-FC provides the maximum and guaranteed coverage. Whereas the Opportunistic SCAN provide least coverage. However, the coverage of all the algorithms converges at sensing range greater than 50 cms.

6.3.3. Message Overhead:

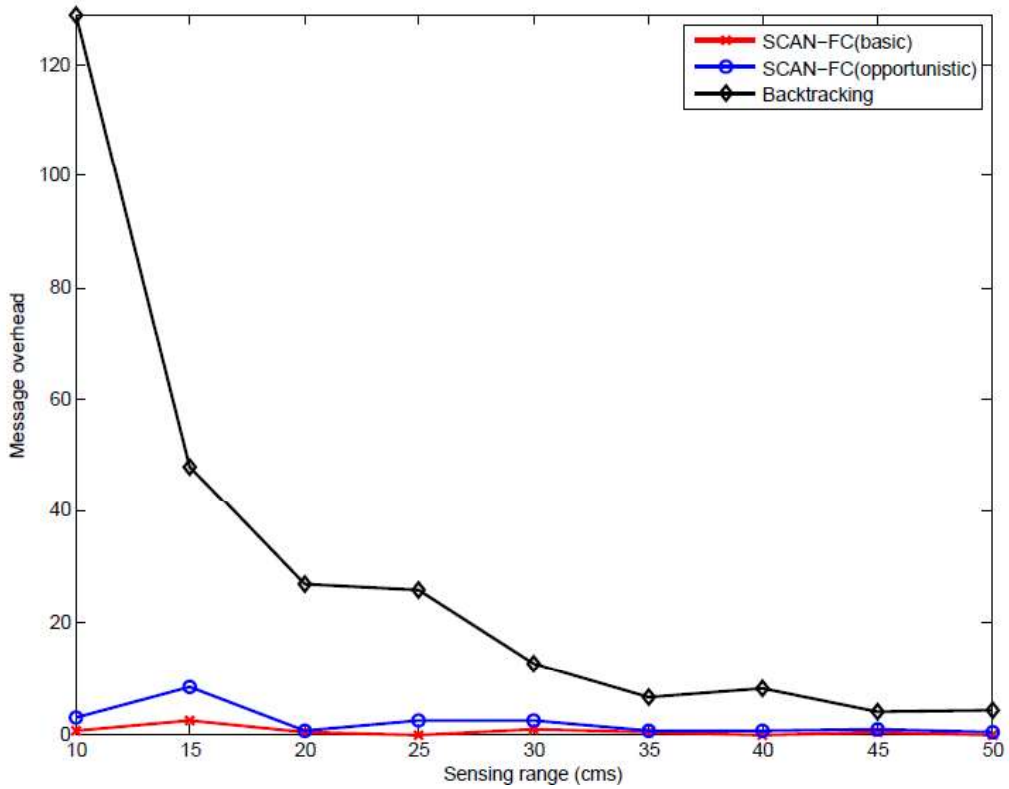

Figure 6.12. Message overhead by the variant of F-coverage algorithm execution on sensor nodes. This graph shows the additional message exchange to ensure extended deployment in the yet unexplored region.

It is clear from the figure 6.12. That the backtracking has maximum coordination and involves highest message overhead, this is expected behavior as the robot has to perform high level of communication at each step of deployment. Whereas, the SCAN-FC (basic) involves least message overhead.

6.3.4. Simulation Time:

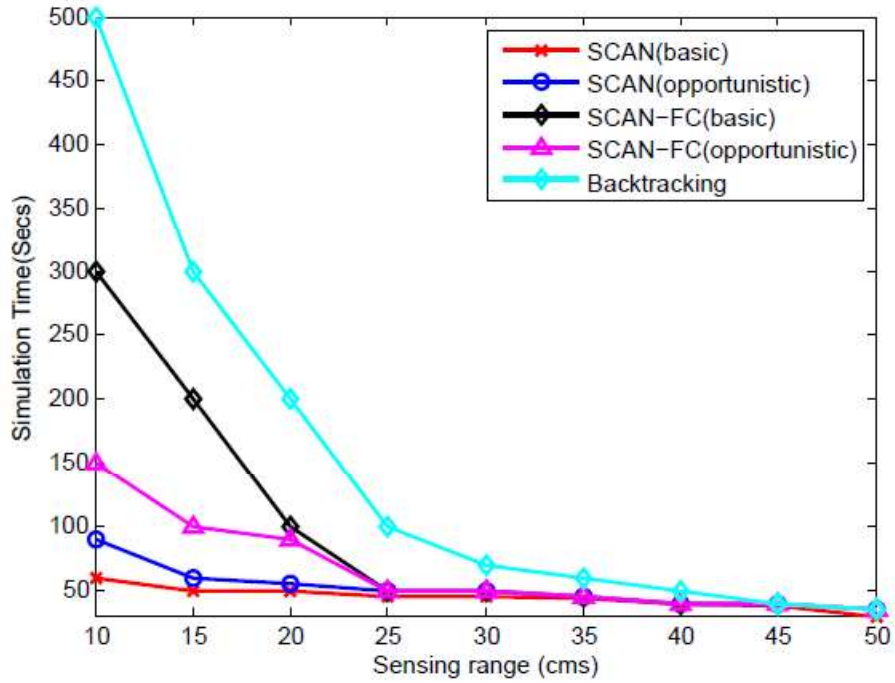

Figure 6.13. MATLAB based Simulation time comparison for Basic SCAN & Opp-SCAN.

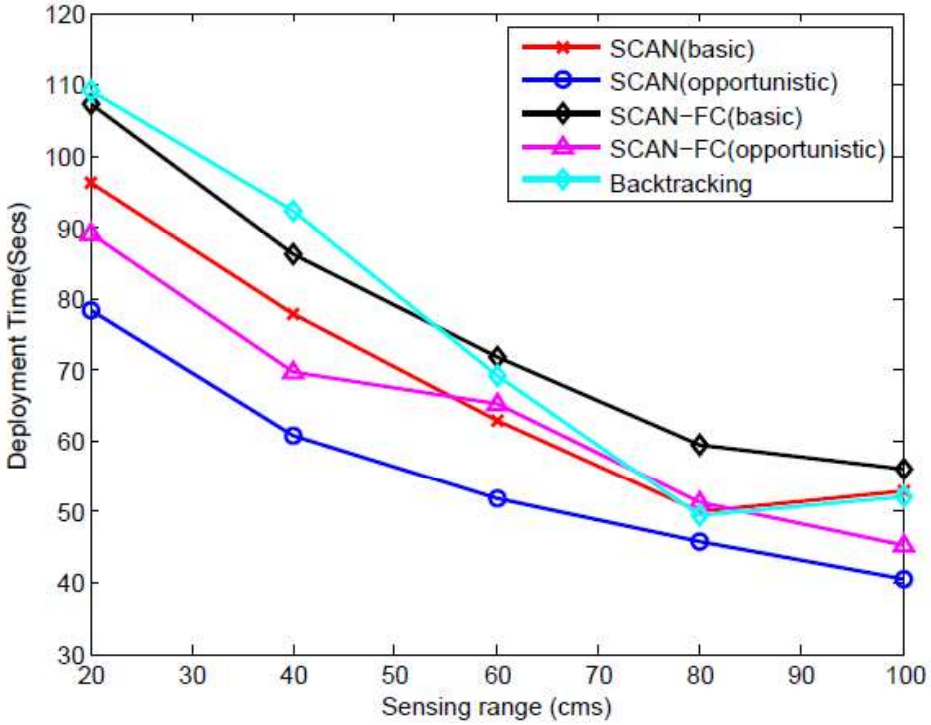

Figure 6.14. WEBOTS based Simulation time comparison for Basic SCAN & Opp-SCAN.

In terms of time of completion, as expected all versions of SCAN algorithms outperforms Backtracking algorithm and out of SCAN algorithms the best one is SCAN-Basic.

CHAPTER 7

MULTI ROBOT IMPLEMENTATION

In this chapter we present the design, implementation and comparison of multi robot implementation of our algorithms. We have extended our work by applying all the designed algorithms and concepts to multi robot scenarios.

7.1. Advantages of Multi-Robot Implementations:

There are various advantages of using multi robot application as single robot solutions are no longer considered optimal solution and may not provide efficient results to real time challenges like Search and Rescue operations, Mapping / Investigating an unknown and hazardous regions etc. In such mission critical applications it becomes important to deploy multi robot solutions. Moreover, there are numerous advantages of using this technique, some of the reason are highlighted below:

7.1.1. Inability of a single to perform tasks alone:

In most of the cases, a single robot lacks the capabilities and resources to complete the task.

For example, given our classic task of deploying sensors vast and unknown ROI is assigned to a single robot. The robot initiates the deployment task and may not complete the area as its energy might deplete in the middle of task completion. And in such cases the robot will fail in the middle of the application execution and the entire setup might fail badly.

In such scenarios, if a team of robots is deployed then if one of the robot fails, we can have a backup robot or a passive robot which can replace the failed robot.

7.1.2. Faster Execution of the task:

The total time taken to complete a task reduces drastically in case of multi robot scenario as a task can be sub-divided into sub-tasks and be allocated to individual robots, in which case the total time of execution is divided by the number of robots, which significantly reduces the time of completion.

Let the set of n robots, denoted I1, I2, I3……., In

Let set of m tasks, denoted by J1, J2, J3 …… Jm

Then the simple equation of task execution U is given by:

$$U = \frac{\sum_{i=0}^{n} I_i}{\sum_{j=0}^{m} J_j} \quad \ldots\ldots (19)$$

7.1.3. Easier Localization:

A proven fact about multi robot implementation is that the robots cooperative with each other and make other robots learn about the information one robot has. This cooperation and information sharing results in easier, reliable and much faster localization and mapping of the unknown region without a need of GPS.

As an example, consider an environment where a single robot is being assigned a task of region localization, in which case the robot heavily relies on its own capabilities and cannot get any sort of informative help from any other entity, which implies that, if this alone robot fails then the entire application fails.

7.1.4. Robustness and Reliability of Solution:

One of the most distinguishing factor of multi robot selection is the solution reliability and robustness to failure, which can be achieved by introducing redundancy, and avoiding single point of failure (only a single robot).

7.1.5. Wider Solution Scope:

It is very common fact that the types and varieties of solution that can be provided by multi-robot application is far more than that of single robot. This can be achieved by using the opportunistic behavior to dynamic conditions with creative and innovative means.

As an example, if one of the robot's distance sensor fails in which case it loses its capability to detect an obstacle, another robot with an operational camera aid the failed robot to carry out the task by avoiding the obstacle with the information received from the working robot. [100]

Similarly, another example could be of a robot which is stuck in a mud or a terrain then one or more robotic teammates can help needy robot.

7.1.6. Using Specialists robots rather than Generalists:

In case of multi robot applications, we can apply the concept of multiple specialized robots and assign specific tasks to specific robots which are good at executing those tasks, rather than requiring a single robot with generic capabilities to perform all tasks. As an example, we can have some robots with cameras providing visual information and some robots with physical strength, used to do rugged tasks and some other delicate robots with minimal energy to just forward the obtained information. Such a design might yield in an optimized working environment.

7.2. Challenges of Multi-Robot applications:

The single robot scenario is relatively very simple and doesn't involve major complexities, however when it comes to multi robot implementations, various complexities arise. Some of the key issues which need to be tackled in case of multi robot implementation are as follows:

7.2.1. Robot coordination:

One of the major challenges when it comes to multi robot implementation is that of robots coordination, especially in case of deployment algorithms. The first and foremost issue in multi robot systems is to solve the question of how to achieve cooperation in a team of robots.

Coordination among the robots is the basic requirement for the multi robot scenario to work. Failure to implement robot coordination might result in deadlock and ultimately result in complete application disaster. Hence, in case of multi robot implementation, the coordination among the robot must be given special design considerations.

7.2.2. Task allocation:

As a researcher when we design, implement and use multi robot coordination techniques a very important problem that arises is that of "Which robot should be assigned which task?" and "How to assign tasks to a robot" in areas where the communication is minimal. [101]In recent times, the researchers has come up with the technique of multi-robot task allocation (MRTA) and out forward a technique called swarm robotics. [102] In this work we have used the concept of MRTA with an enhanced approach of optimized MRTA allocation.

7.2.3. Resource division:

Resource consumption is an important aspect when it comes to WSRN as the resources are relatively scarce and should be consumed in most optimal way. Failure to handle this parameter might directly impact the success of the algorithm.

Researchers in recent times has merged the concept of Utility can be applied to multi robot implementation. Utility is unifying, if sometimes implicit, concept in economics [103], game theory [104], and operations research [105]. The basic idea behind Utility is that, each robot can

estimate the cost of executing a certain task which is sometimes called as fitness, valuation and cost.

Consider a Robot R and a task T, if R is capable of executing T, then we can derive a standard scale. Q_{RT} and C_{RT} as the quality and cost, respectively, expected to result from the execution of T by R.

$$U_{RT} = \begin{cases} Q_{RT} - C_{RT} & \text{If R is capable of executing T and } Q_{RT} > C_{RT} \text{ otherwise} \\ 0 & \end{cases} \quad \ldots (20)$$

7.2.4. Avoiding deadlock conditions:

In case of multi robot scenarios more than one tasks are executed in parallel, which requires a constraint to the dynamic resource assignment algorithm. This is required to avoid assignments which results into a conflicting situation of share resources and ultimate results in complete deadlock. [106]One of major shortcoming of Back Tracking Based deployment [52] is that the algorithm might end up in deadlock condition in case of multi robot scenarios.

Thus, deadlock prevention and avoidance techniques plays major role for the success of the algorithms. A comprehensive prove of deadlock avoidance has already been presented in Section 4.4, where we have presented four Lemmas and proved the point that the deadlock would never happen during the execution of algorithms.

7.3. ALGORITHM DESIGN FOR MULTI-ROBOT:

In this section we present the architecture, design and implementation of multi-robot based SCAN algorithms. We have extended our work of single robot implementation to multi-robot and designed two novel approaches for sensor deployment with a team of mobile robots. We propose two modes of multi robot sensor deployment termed as Divide & Conquer (D&C) mode

and Cooperative Deployment Mode (CDM). The D&C mode is much simpler with less complexity when compared to CDM which is much intelligent and require much more cooperation among the robots for carrying out the task.

This section consists of three parts, in the first part we present technical details of D&C mode followed by the implementation of D&C approach. In the next section we present CDM implementation and finally we present the comparison results of both the approaches.

7.3.1. Divide & Conquer (D&C) mode:

From the classical mathematical problem, the essence of Divide and Conquer technique involves three steps:

1. **Divide** a bigger problem into a set of several smaller problems (The logic behind the smaller problems would be similar to the original).
2. **Conquer** the individual sub-problems with an individualistic problem solving approach.
3. **Combine** the results obtained from the individual smaller problems to aggregate the results.

The above concept can be applied to sensor deployment problem with a team of robots. In the first phase the task is divided among the robots and each robot takes the responsibility of its own sub-area for deployment process. In the second phase (Conquer) the team of robots carry out the task of sensor deployment in its own area of interest. In this way, the SCAN, OPP-SCAN, F-SCAN and F-OPPSCAN algorithms can be executed by each robot to carry the sensor deployment process.

The algorithm executes with an average complexity of $O(n\,logn)$ and proves to be one of the fastest approaches to solve the problem of sensor deployment. The results of our experiment

proves that the D&C technique achieves faster and most optimal coverage results. In the next section of this chapter we present the implementation and performance of this algorithm.

Fig 7.1 shows an overview of main ROI sub divided into multiple smaller sub-ROI and each robot running the basic SCAN algorithm. Consider a scenario that a team of four robots R1, R2, R3 and R4 has been assigned a task of deploying sensors in an unknown area A. The robots initially coordinates with Command & Control center (C & C) and obtains the start location of its own sub-area, this implies that the C & C center divides the area into smaller sub-areas and assign each sub area to one of the robots. In the current scenario, let robot R1 is assigned Areas A1, R2 is assigned A3, R3 with A3 and R4 with A4.

Each robot takes the starting position of its own area and mutually coordinate with other robots and executes any one among the two Basic-SCAN or OPP-SCAN algorithms and deploy the sensors. The algorithm completes when all the four robots get back to the starting point of their deployment, after which they can optionally execute F-Coverage version of algorithm to ensure better network coverage.

This mode of operation involves less coordination among the robots and the inter robot communication is minimal. Thus, reducing the complexity of the task which ultimately results in the better performance of the algorithm.

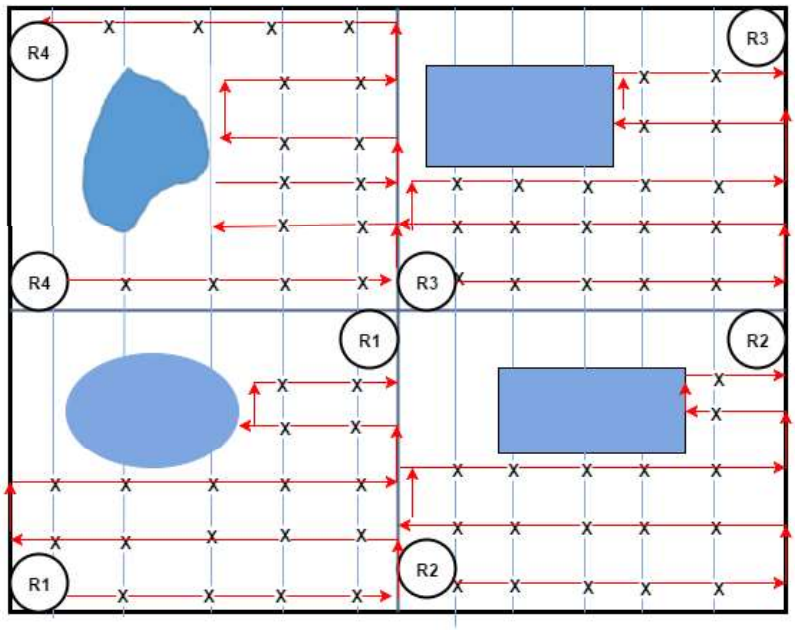

Figure 7.1. Multi robot Divide and Conquer mode of sensor deployment where the entire ROI is divided into smaller sub-ROI and each robot is responsible of its corresponding region.

7.3.1.1. Implementation of Multi-Robot D&C mode:

In this section we present the practical implementation of Divide & Conquer technique of multi robot sensor deployment. Figure 7.2 shows a sample Webots world in which the main area is divided into two smaller sub-areas and two E-puck Robots R1 and R2 are assigned to each sub-area. Figure 7.1 shows the theoretical view of the path the robots R1 & R2 are expected to take and Figure 7.3 shows the actual result after running the experiment.

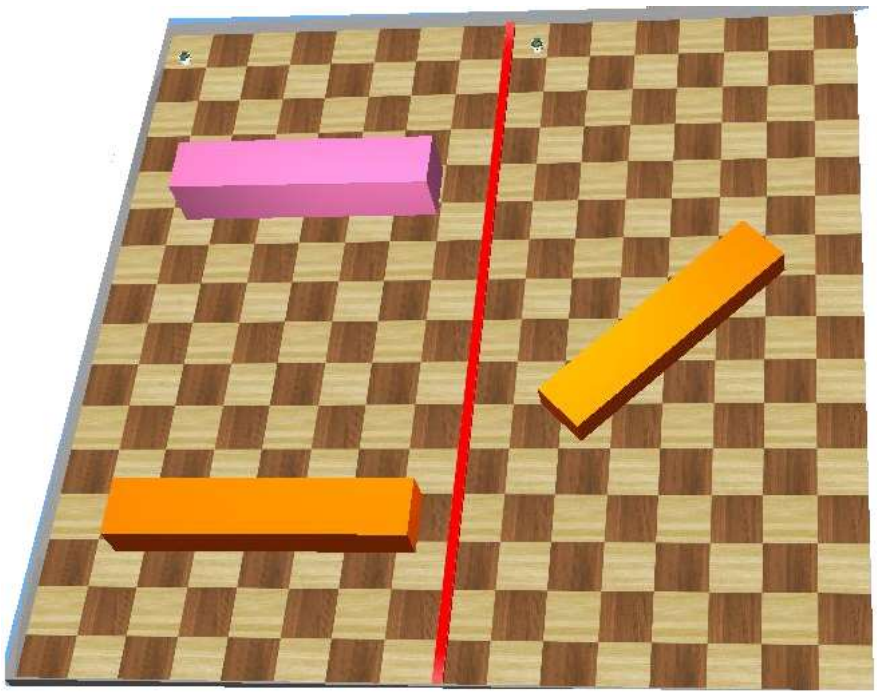

Figure 7.2. A sample Webots ROI showing the D & C mode of operation, the actual area is divided into two parts and each robot is responsible of sensor deployment in its own area of interest.

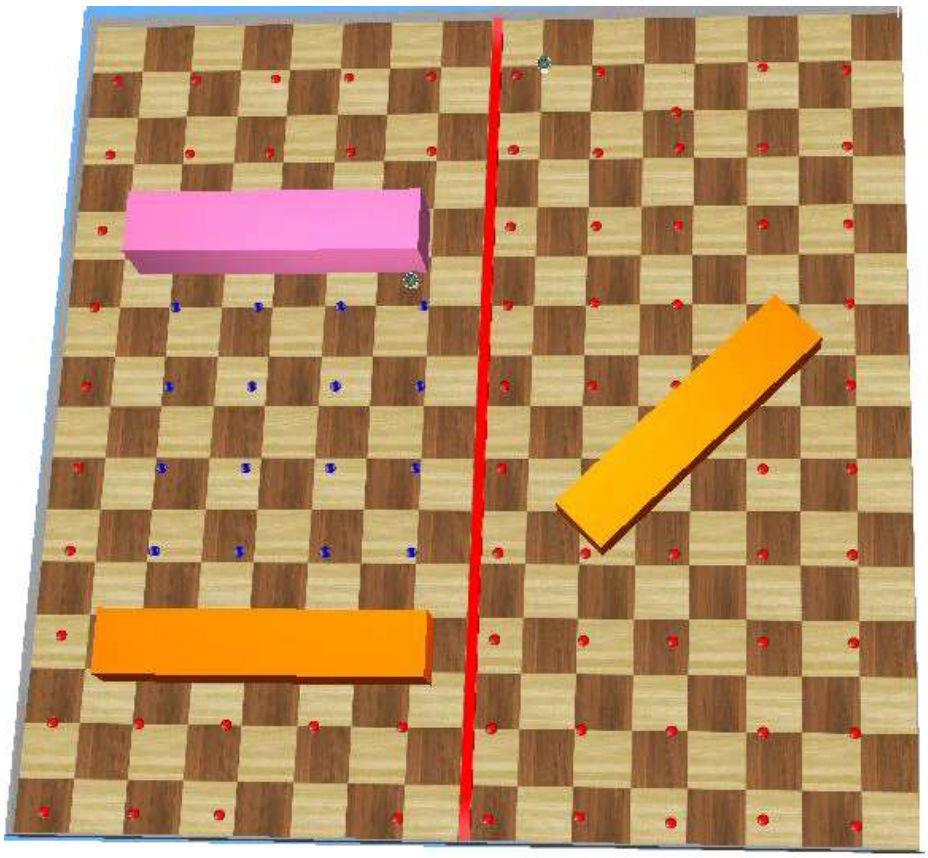

Figure 7.3. Actual view of experimental results after running the D&C algorithm with multiple robots.

7.3.2. Cooperative Deployment Mode (CDM):

CDM is another multi robot sensor deployment in this mode of operation, a single ROI would be covered by multiple robots with as stronger dependency on the robots coordination. Fig 7.4 shows a sample scenario where 2 robots carry out the deployment process in a cooperative manner. Thus all the four SCAN versions can be applied in this mode of operation as well.

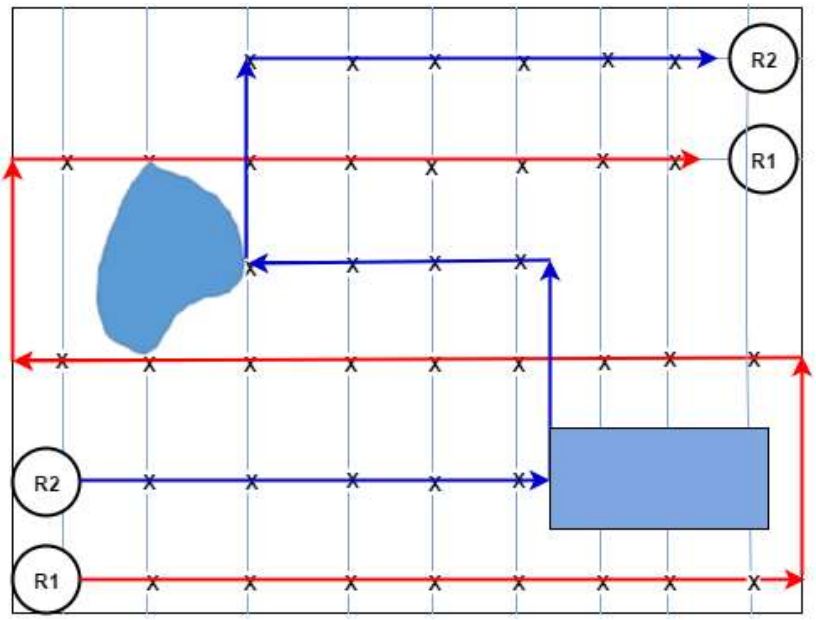

Figure 7.4. Multi robot cooperative deployment mode where robots cooperatively deploy the sensors in an incremental fashion.

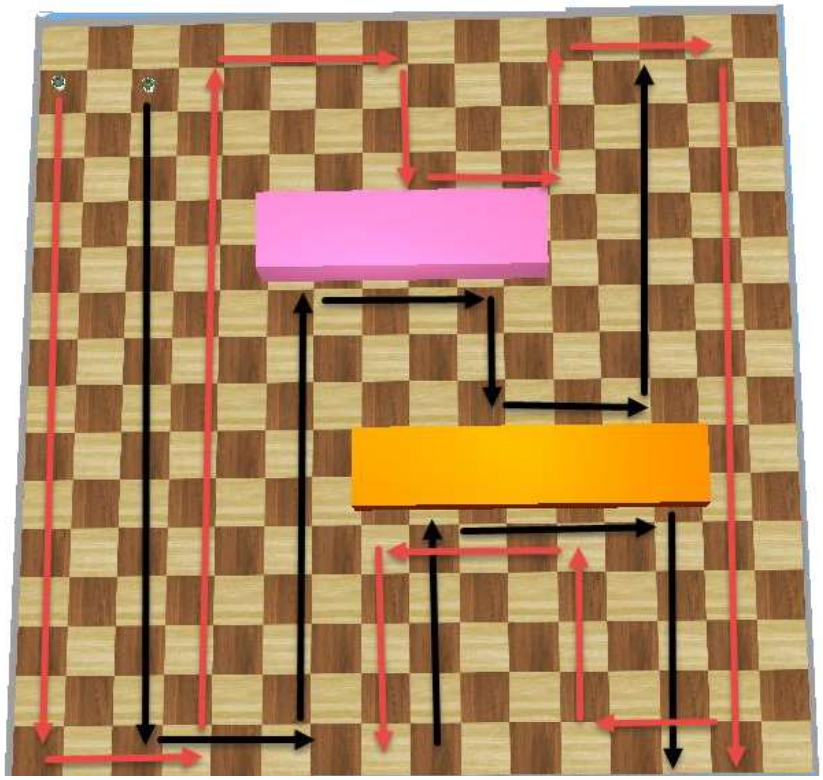

Figure 7.5. A sample Webots ROI showing the expected trajectory of the robots in CDM mode of operation, both the robots starts from start position and (start position + offset) and deploy the sensor in cooperative mode.

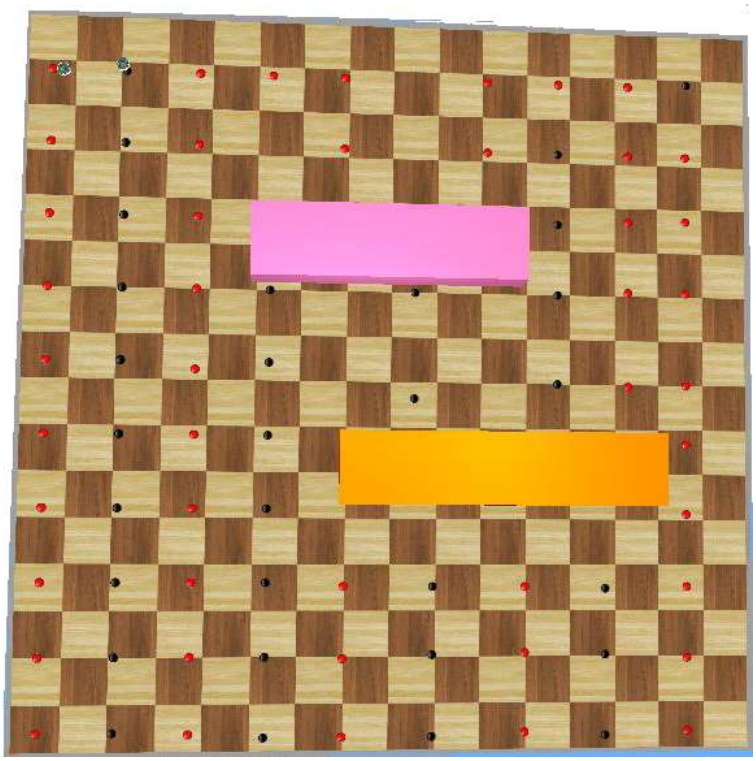

Figure 7.6. Actual view of experimental results after running the CDM algorithm with multiple robots.

7.4.Performance evaluation of D & C and CDM algorithms

7.4.1. Robot Distance Travelled:

Figure 7.7 shows the comparison result of Divide & Conquer and Cooperative Deployment Mode algorithms in terms of total distance travelled by the robot after the initial round. We compared the distance with various sensing ranges (S_r) of the sensor, which is the step size of the robot. CDM performs better i.e. the total distance travelled by robot is always less for any sensing range (10-45 m).

As the sensing range increases the total travelled distance decreases for both algorithms, as expected, because the total number of points to be covered with sensors becomes less. In case of D&C mode the distance gradually decreases with increase in sensing range without any deviation, this is expected as the ROI is divided among the robots and the results would be average of the distance travelled by each robot. But in the case of CDM, we observe a sharp deviation in robot distance travelled at sensing range of 45. This behavior is due to the fact that, at that particular sensing range the coverage obtained is relatively less, which implies that the robot has deployed lesser sensors. From this we can conclude that the robot distance travelled would be less. Moreover, the obstacle alignment in ROI might not support the sensing range of 45. So it is very important to consider a suitable sensing range corresponding to the obstacle distribution. The adjustment of sensing range plays a critical role in obtaining optimal results.

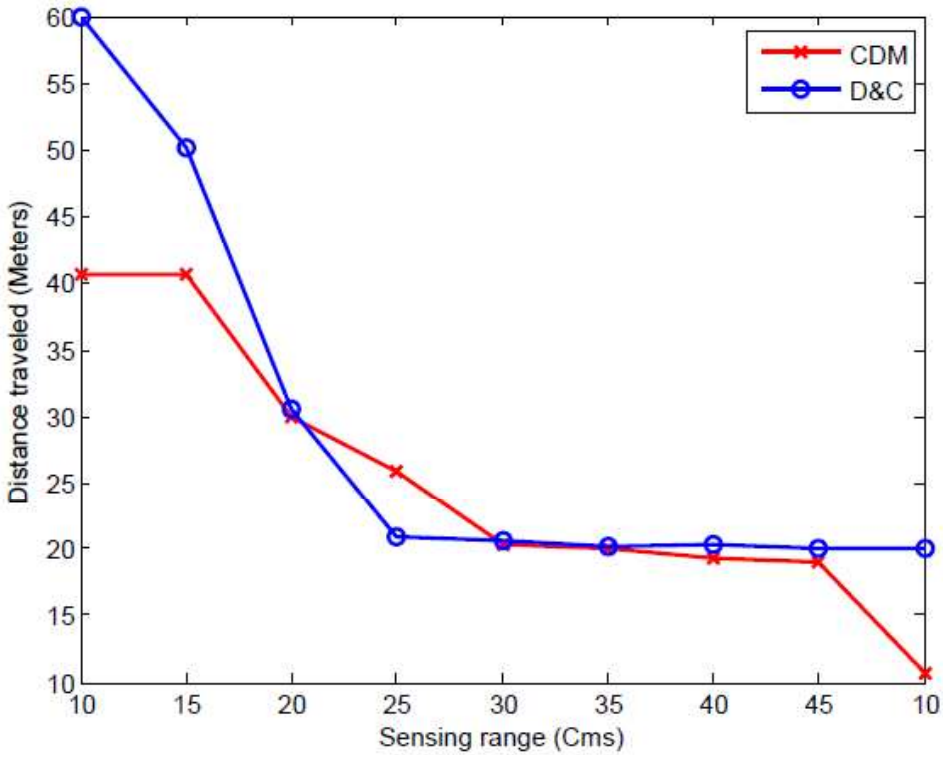

Figure 7.7. Comparison of D&C and CDM algorithms performance in terms of distance travelled.

7.4.2. Coverage:

Figure 7.8 shows the sensing range versus coverage plot. The CDM outperform D&C in terms of coverage and deploy higher number of sensors at all sensing ranges. However, the both D&C and CDM coverage performance converges when the sensing range becomes greater than certain value i.e. 50 cms and both provide 100% coverage. So, there are trade-offs for using them for different requirements. If the desired result is to have the maximum coverage with short sensing range without considering robot distance (as the distance in CDM is much higher) then CDM works well. On the other hand, if QoS configurations are such that the robot distance travelled is to be minimized and sensing range can be higher in such cases D&C is the solution.

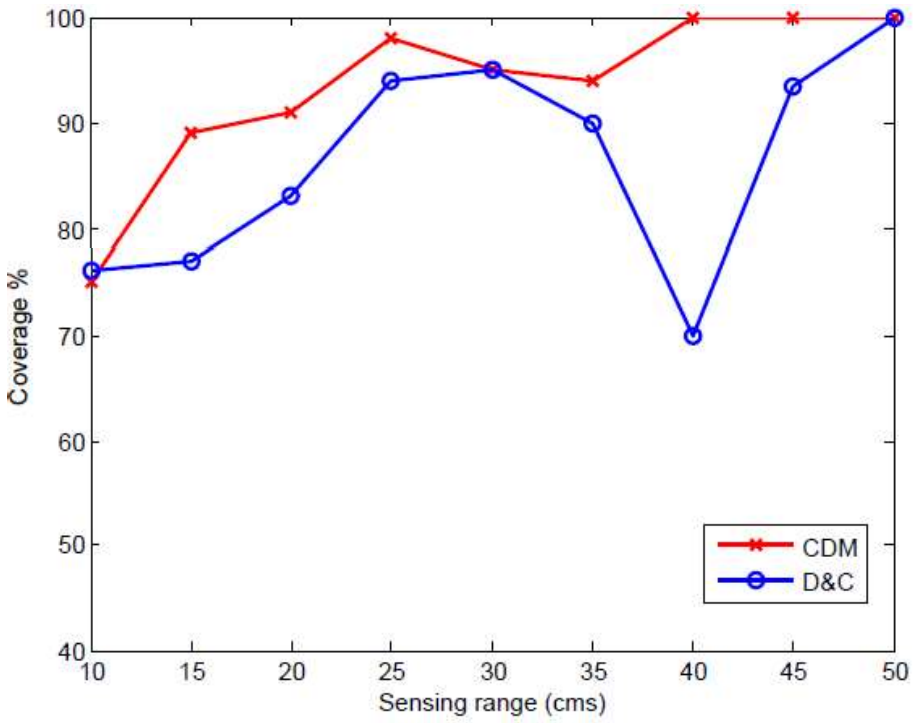

Figure 7.8. Coverage percentage using each of the four versions of SCAN. SCAN-FC for both the basic and opportunistic algorithms provides better performance in this particular scenario.

7.4.3. Message Overhead:

As explained in earlier chapter, the design and architecture of CDM was such that there would be high level of cooperation among robots, which inherently requires lot of inter robot communication and between the robots and command and control center. In case of D&C mode, each robot works independently of others and doesn't involve in any communication with other robots. As a result of such behavior, the message overhead is restricted only to individual robot and command & control center, which ultimately results is a drastic decrease in message overhead. The above stated discussion is clearly supported by graph in Figure 7.9, which shows that CDM requires higher message overhead when compared to D&C.

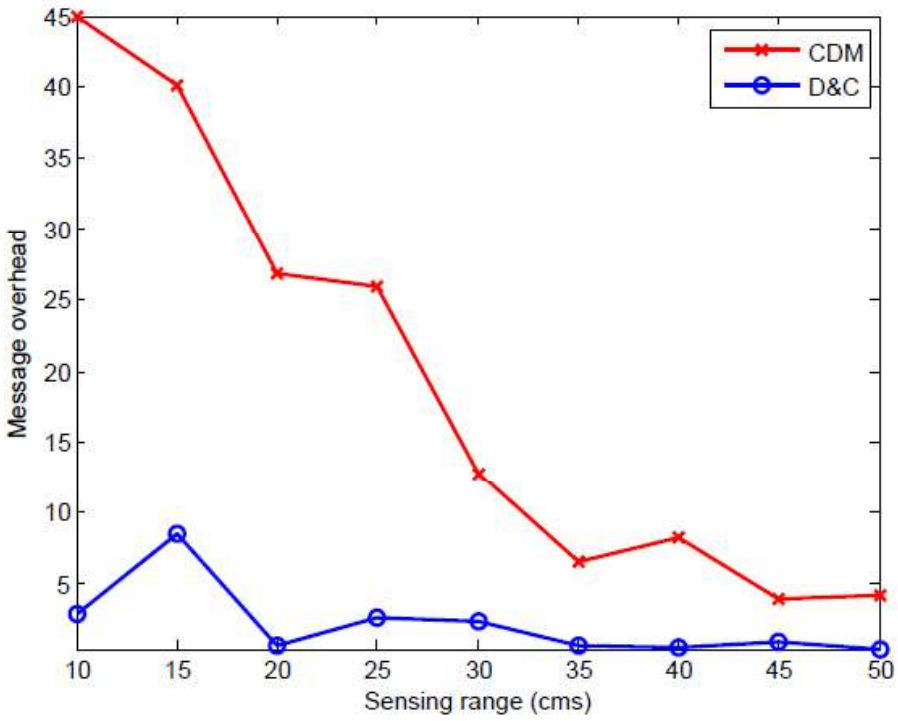

Figure 7.9. Message overhead by the CDM and D&C algorithms. This graph shows the message exchange to ensure deployment in the ROI.

7.4.4. Deployment Time:

Figure 7.10 shows the time required to complete the deployment process for both CDM and D&C modes of operation. It is very clear from the figure that the CDM requires higher time to complete the deployment when compared to D&C mode of operation. This is expected behavior, as in CDM robots cooperative covers the ROI and involves complex operation to be executed that too in collaboration with each other, whereas in D&C the task is divided among the robots which implies that the deployment time is also divided among the robots which results in reduced time but at the cost of lower coverage.

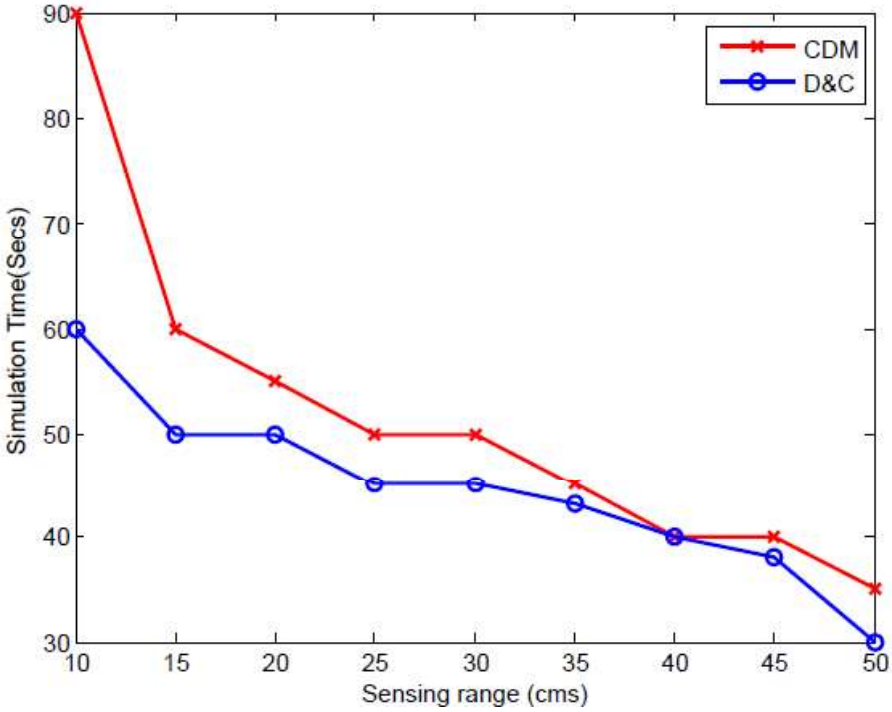

Figure 7.10. Simulation time comparison for CDM and D&C algorithms.

CHAPTER 8

CONCLUSIONS & FUTURE WORK

Application of Wireless Sensor Networks (WSN) on ROBOTICS is a topic of great interest and application of which is increasing rapidly for the last few decades. There are numerous mission critical applications currently deployed on WSRN, some examples of which are military surveillance, event monitoring & detection, Oil and gas exploration, Health monitoring, toxic gas emission detection etc.

One of the challenging requirements of WSRN is to deploy a maximum coverage sensor networks using mobile robots. Most of the existing solutions for sensor deployment using robots are purely based on simulations & emulations and lacks the real characteristics of Wireless Sensor Robot Network (WSRN) like heterogeneity of environment, dynamic obstacle distribution, non-availability of GPS, robot's obstacle & sensor detection capabilities etc.

In this thesis, we presented solutions to major issues related to deployment algorithms and propose four novel algorithms, two of them for single robot applications and two for multi-robot scenarios. The major contribution of this work was to propose novel approaches of sensor deployment especially in the areas where there is no GPS and where human reachability is not possible. Also, the major distinguishing aspect of our work from previous works is that it is backed by solid mathematical background and real robot implementation of all the algorithms.

In the first part of the thesis, SCAN based algorithms were presented which guarantees maximum coverage with minimum robot distance in minimum execution time. We also

performed an exhaustive experiments and results proves that SCAN based algorithms outperform all the existing deployment algorithms. However, not every algorithms are fully equipped with the all-in-one solution. Thus, one of the major shortcoming of SCAN algorithm was the fact that it works perfectly only in single robot applications. We solved this issue by extending our initial work which was the second part of the thesis, in which, we present other two novel multi-robot deployment algorithms namely Cooperative Deployment Mode (CDM) and Divide & Conquer Deployment mode (D&C). With this we were able to provide end-to-end solution for sensor deployment approaches in both single robot as well as multi-robot scenarios.

We have evaluated the performance of our proposed model by considering key factors like coverage, robot distance, completion time and message overhead and prove that SCAN based deployment outperforms the back tracking deployment in both single and multi-robot scenarios. In terms of coverage, BASIC SCAN algorithm performs better than all other algorithms, but at the cost of higher total distance travelled. When it comes to distance and message overhead, Opportunistic SCAN algorithm performs better than all others. In order to ensure the consistency of the proposed model, we have rigorously tested the algorithms in numerous scenarios with dynamically changing obstacle distributions with an optimum confidence interval.

To ensure the correctness and reliability of our algorithms, we present various real application scenarios and case studies considering all real world challenges. Therefore, the novelty of this thesis can be extended further to be applied across the real industry domains.

Finally, although our proposed algorithms offer solutions to wide range of sensor deployment problems, we envision several improvements which can enhance the application scope. Below is a list of improvements which can be considered as future work:

- **Large Scale Deployment:** We have applied our algorithms in various real scenarios and rigorously tested them, however in order to be applied in real industry domain, they have to be tested in real test bed scenarios with large scale deployment.
- **Considering Highly Complex ROI's:** The proposed deployment algorithms need to be enhanced and tested considering various complex scenarios like environment with water, terrains etc.
- **Post deployment Activities:** Another important aspect which was considered out of scope of this work was network maintenance and restoration phase. As mentioned earlier, a critical factor in WSN is continues connectivity, thus we need to ensure that connectivity is ensured all the times. Thus, we see a major enhancement in current work to be extended to maintenance phase.

REFERENCES

[1] V. Kae, Y. Fukushima and H. Harai, Design and implementation of dynamic mobile sensor network plat.

[2] M. Dunbabin, P. Corke, I. Vasilescu and D. Rus, Data muling over underwater wireless sensor networks using an autonomous underwater vehicle," in Robotics and Automation, ICRA 2006. Proceedings 2006 IEEE International Conference on. IEEE, 2006, pp. 2091–2098. Form," Communications Magazine, IEEE, vol. 53, no. 3, pp. 48-57, 2015..

[3] A. Iqbal and M. Murshed, A hybrid wireless sensor network framework for range-free event localization," Ad Hoc Networks, 2014..

[4] J. Zhu, C.-H. Lung and V. Srivastava, A hybrid clustering technique using quantitative and qualitative data for wireless sensor networks,, Ad Hoc Networks, vol. 25, pp. 38–53, 2015..

[5] W. Hu, C. T. Chou, S. Jha and N. Bulusu, Deploying long-lived and cost-effective hybrid sensor networks,, Ad Hoc Networks, vol. 4, no. 6, pp. 749–767, 2006..

[6] X. Li, G. Fletcher, A. Nayak and I. Stojmenovic, Randomized carrierbased sensor relocation in wireless sensor and robot networks, Ad hoc networks, vol. 11, no. 7, pp. 1951–1962, 2013..

[7] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk and a. J. Anderson, Wireless sensor networks for habitat monitoring, in Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications. ACM, 2002, pp. 88{97..

[8] A. Pantelopoulos and N. G. Bourbakis, A survey on wearable sensor-based systems for health monitoring and prognosis,, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 40, no. 1, pp. 1{12, 2010..

[9] W. Tsujita, H. Ishida and a. T. Moriizumi, Dynamic gas sensor network for air pollution monitoring and its auto-calibration, in Sensors, 2004. Proceedings of IEEE. IEEE, 2004, pp. 56{59..

[10] O'ROURKE. and S. B. J. CHRIS, Mobile ad hoc networking revamps, Available: http://www.xes-inc.com/assets/files_indexed/193327_AdHocNetworks-COTS-Nov-

2011.pdf.

[11] T. Melodia, D. Pompili and a. I. F. Akyldiz, Handling mobility in wireless sensor and actor networks, IEEE Transactions on Mobile Computing 9:160–173, 2010..

[12] Y. Zeng, C. J. Sreenan and G. Zheng., A real-time architecture for automated wireless sensor and actuator networks, In Proceedings of the 2009 Fifth International Conference on Wireless and Mobile Communications, ICWMC '09, Washington, DC. IEEE Computer Society, 2009, pp. 1–6..

[13] Margara, G. Cugola and Alessandro, Slim: Service location and invocation middleware for mobile wireless sensor and actuator networks., International Journal of Systems and Service-Oriented Engineering 1(3):60–74, 2010..

[14] F. Xia, X. Kong and Z. Xu, Cyber-physical control over wireless sensor and actuator networks with packet loss., Computing Research Repository, 2010, pp. 85–102.

[15] U. Baroudi, M. Azharuddin and M. Haque, "SCAN: A Sensor Deployment Algorithm in Hazardous Regions via Mobile Actuators," *Sensors Journal, IEEE*, vol. PP, no. 99, p. 1, 2016.

[16] M. C. Akewar and N. V. Thakur, A Study of Wireless Mobile Sensor Network Deployment, IRACST – International Journal of Computer Networks and Wireless Communications (IJCNWC), ISSN: 2250-3501 Vol.2, No4, August 2012.

[17] J. Chen, EntongShen and YouxianSun, The deployment algorithm in Wireless Sensor Network: A Survey, Information Technology Journal 2009..

[18] L. F. M. Vieira, M. A. M. Vieira, L. Ruiz, A. A. Loureiro, D. Silva and Ant.onioOt´avioFernandes, Efficient Incremental Sensor Network Deployment Algorithm, 2002..

[19] J. Lee, A. Dharne and S. Jayasuria, Potential Field Based Hierarchical Structure for Mobile Sensor Network Deployment, Proceedings of the 2007 American Control Conference, July 2007.

[20] G. Tan, S. A. Jarvis and A.-M. Kermarrec, Connectivity-Guaranteed and Obstacle-Adaptive Deployment Schemes for Mobile Sensor Networks, IEEE transactions on mobile computing, vol. 8, no. 6, June 2009..

[21] J. Li, B. Zhang and L. Cui, "A Restricted Delaunay Tringulation Graph Based Algorithm for Self-Deployment in Mobile Sensor Networks, Journal of Computational Information

Systems 2010..

[22] J. Wu and S. Yang, SMART: A Scan-Based Movement-Assisted Sensor Deployment Method in Wireless Sensor Networks, IEEE Transaction on parallel and distributed systems, VOL. 18, No. 8, August 2007.

[23] W. G, C. G and P. TL, Movement-assisted sensor deployment., IEEE Trans Mob Comput 6:640–652.

[24] F. Aurenhammer, Voronoi Diagrams—A Survey of a Fundamental Geometric Data Structure, ACM Computing Surveys 23, pp. 345-405,1991..

[25] S. Shakkottai, R. Srikant and N. Shroff, Unreliable Sensor Grids:Coverage, Connectivity and Diameter., In IEEE Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM 2003) (New York, NY, USA, Mar. 2003), vol. 2,ACM,pp. 1073–1083..

[26] W. Li, Y. I. Kamil and A. Manikas, Wireless Array Based Sensor Relocation in Mobile Sensor Networks, ACM 2009..

[27] N. A. B. A. Aziz, A. W. Mohemmed and M. Y. Alias, A Wireless Sensor Network Coverage Optimization Algorithm Based on Particle Swarm Optimization and Voronoi Diagram, Proceedings of the 2009 IEEE International Conference on Networking, Sensing and Control, Okayama, Japan, March 26-29, 2009..

[28] C.-Y. Chang, Y.-C. Chen and H.-R. Chang, Obstacle-resistant deployment algorithms for wireless sensor networks, Vehicular Technology,IEEE Transactions on, vol. 58, no. 6, pp. 2925–2941, 2009.

[29] C.-H. Wu, K.-C. Lee and Y.-C. Chung, A Delaunay Triangulation Based Method for Wireless Sensor Network Deployment.

[30] G. M, G. M, C. CF and L. E, "A distributed sensor relocation scheme for environmental control.," in *The ACM/IEEE Proc. of MASS*, 2007.

[31] C. M, Y. Y and W. J, "Non-uniform sensor deployment in mobile wireless sensor networks," in *Proc. Of WoWMoM, pp 1–8.*

[32] W. Li and C. G. Cassandras, A minimum-power wireless sensor network self-deployment scheme, in Proc. Annu. IEEE WCNC, New Orleans, LA, Mar. 2005, vol. 3, pp. 1897–1902..

[33] H. N and V. P, Energy-efficient deployment of intelligent mobile sensor networks., IEEE Trans Syst Man Cybern 35:78–92..

[34] A. Howard, M. J. Mataric and G. S. Sukhatme, An incremental self-deployment algorithm for mobile sensor networks,", Autonomous Robots,vol. 13, no. 2, pp. 113–126, 2002..

[35] S. Toumpis and L. Tassiula, Packetostatics: Deployment of massively dense sensor networks as an electrostatics problem, in Proc.IEEE INFOCOM, Miami, FL, Mar. 2005..

[36] M. K, Z. Y and TrappeW, Managing the mobility of a mobile sensor network using network dynamics., IEEE Trans Parallel DistribSyst 19(1):106–120.

[37] B. Liu, P. Brass and O. Dousse, Mobility Improves Coverage of Sensor Networks, Proc. ACM MobiHoc, 2005..

[38] G. Wang, G. Cao, T. L. Porta and W. Zhang, Sensor relocation in mobile sensor networks., In Proceedings of the 24th Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM),Vol. 4, Miami, FL, 2005, pp. 2302–2312..

[39] C.-H. Liu and Kuo-FengSsu, A Moving Algorithm for Non-Uniform Deployment in Mobile Sensor Networks, The International conference on Mobile Technology, Application and System 2008,(Mobile Conference) 10-12 September, 2008, ACM..

[40] T. Ogawa, T. Shinjo, S. kitajima and T. Hara, Node control methods to reduce power consumption using push-based broadcast for mobile sensor networks, journal of mobile multimedia, vol. 6, no.2 (2010) 114-127..

[41] JiLuo, DanWang and Q. Zhang, Poster Abstract: Double Mobility:Coverage of the Sea Surface with Mobile Sensor Networks, Mobile Computing and Communications Review, Volume 13, Number 1..

[42] L. Hu and D. Evans, Localization for Mobile Sensor Networks, MobiCom'04, Sept. 26.– Oct. 1, 2004, Philadelphia,Pennsylvania, USA.2004 ACM 1-58113-868-7/04/0009..

[43] V. Dyo and C. Mascolo, A Node Discovery Service for Partially Mobile Sensor Networks, ACM 978-1-59593-929-6/07/11..

[44] C. W. Yu, C.-H. Wang, L. C. Hsu and K. J. Cheng, Coverage Algorithms in GPS-less Wireless Mobile Sensor Networks, IEEE 2006.

[45] M. Kalantary and M. R. Meybodi, Mobile Sensor Network Deployment Using Cellular Learning Automata Approach, International Congress on Ultra Modern

Telecommunications and Control Systems and Workshops (ICUMT).

[46] FaridBenbadis, K. Obraczka, J. Cort´es and AlexandreBrandwajn, Exploring landmark placement strategies for selflocalization in wireless sensor networks, The 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'07).

[47] A. Sekhar, B. S. Manoj and C. S. R. Murthy, Dynamic coverage maintenance algorithms for sensor networks with limited mobility, in Proc. 3rdIEEE Int. Conf. Pervasive Comput. Commun. (PerCom), Kauai Island, HI, Mar. 2005, pp. 51–60..

[48] P. S and S. GS, Constrained coverage for mobile sensor networks., In: Proc. of IEEE ICRA.

[49] X. Li, H. Frey, N. Santoro and I. Stojmenovic., Localized self-deployment of mobile sensors for optimal focused-coverage formation., Technical Report TR-2007-13, SITE, University of Ottawa, December 2007..

[50] M. R. Pac, A. M. Erkmen and IsmetErkmen, Scalable Self-Deployment of Mobile Sensor Networks:A Fluid Dynamics Approach, Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 9 - 15, 2006, Beijing, China.

[51] L. Xu, N. Amiya and S. Ivan, Sensor placement in sensor and actuator networks. Algorithms and Protocols for Scalable Coordination and Data Communication, Eds.Wiley VCH, Hoboken, NJ, 2010, Chapter 10..

[52] G. Fletcher, X. Li, A. Nayak and I. Stojmenovic, Back-tracking based sensor deployment by a robot team, in Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on. IEEE, 2010, pp. 1–9.

[53] P. Yuanteng and M. MattW, STARS: Static Relays for Multi-Robot Real-time search and monitoring, In Proceedings of the 7th IEEE Int'l Conference on Distributed Computing in Sensor Systems, DCOSS, Barcelona, Spain, 2011, pp. 1–8..

[54] F. Rafael, XuLi and A. Nayak, Carrier-based focused coverage formation in wireless sensor and robot networks., IEEE Transactions on Automatic Control 56(10):2406–2417, 2011..

[55] X. Li, N. Mitton, I. Ryl and D. Simplot, Localized sensor self-deployment with coverage guarantee in complex environment., In Proceedings of the 8th International Conference on AD-HOC Networks & Wireless (AdHoc-Now) LNCS 5793, Murcia, Spain 2009, pp. 138–

151..

[56] P. Bose, P. Morin, I. Stojmenovic and J. Urrutia, Routing with guaranteed delivery in ad hoc wireless networks., Wireless Networks 7(6):609–616, 2001..

[57] A. Howard, A. Howard and G. S. Sukhatme, Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection, The International Journal of Robotics Research, May 2006; vol. 25, 5-6: pp. 431-447.

[58] H. DURRANT-WHYTE and T. BAILEY, Simultaneous localization, IEEE Robotics and Automation Magazine, pp. 1070–.

[59] M. Csorba, Simultaneous Localisation and Map Building, PhD Thesis.

[60] M. Csorba and H. Durrant-Whyt, A new approach to simultaneou localisation and map building., In Proceedings of SPIE Aerosense, Orlando, 1996.

[61] J. Leonard and H. Feder, A computational efficient method for large-scale concurrent mapping and localisation, In J. Hollebach and D. Koditscheck, editors, Robotics Research, The Ninth International Symposium (ISRR'99), pages 169–176. Springer Verlag, 2000.

[62] J. Castellanos, J. Tard´os and G. Schmidt, Building a global map of the environment of a mobile robot:The importance of correlations., In IEEE International Conference on Robotics and Automation, pages 1053–1059, 1997..

[63] S. Williams, P. Newman, G.Dissanayake and H. D. Whyte, Autonomous underwater simultaneous localisation and map building., In Proc. IEEE International Conference on Robotics and Automation (ICRA), pages 1793–1798, San Francisco,USA, April 2000..

[64] K. Chong and L. Kleeman, Feature-based mapping in real,large scale environments using an ultrasonic array, International Journal Robotics Research, 18(1):3–19, 1999..

[65] M. Deans and M. Hebert., Experimental comparison of techniques for localization and mapping using a bearing-only sensor, In Int. Symp. Experimental Robotics, 2000..

[66] H. Durrant-Whyte, Uncertain geometry in robotics., IEEE Trans.Robotics and Automation, 4(1):23 31, 1988..

[67] R. Smith and P. Cheesman., On the representation of spatial uncertainty., Int. J. Robotics Research, 5(4):56–68, 1987..

[68] J. Guivant and E. Nebot, Optimization of the simultaneous localization and map-building

algorithm for real-time implementation., IEEE Trans. Robotics and Automation, 17(3):242–257,2001..

[69] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, FastSLAM:A factored solution to the simultaneous localization and mapping problem., In AAAI National Conference on Artificial Intelligence, pages 593–598, 2002..

[70] J. Castellanos, J. Martnez, J. Neira and J. Tard´os, Experiments in multisensor mobile robot localization and map building., In Third IFAC Symposium on Intelligent Autonomous Vehicles, pages 173–178, 1998..

[71] A. Davison, Y. Cid and N. Kita, Real-time 3D SLAM with wide-angle vision., In IFAC/EURON Symposium on Intelligent Autonomous Vehicles, 2004.

[72] M. Bosse, P. Newman, J. Leonard and S. Teller, Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework., Int. J. Robotics Research,23(12):1113–1140, 2004.

[73] J. Folkesson and H. I. Christensen, Graphical SLAM - a selfcorrecting map., In Proc. IEEE Int. Conf. Robotics and Automation, pages 791–798, 2004.

[74] J. Kim and S. Sukkarieh, Airborne simultaneous localisation and map building., In Proc. IEEE Int. Conf. Robotics and Automation,pages 406–411, 2003..

[75] P. Newman and J. Leonard., Pure range-only subsea SLAM., In Proc. IEEE Int. Conf. Robotics and Automation, 2003.

[76] R. Eustice, H. Singh, J. Leonard, M. Walter and R. Ballard, Visually navigating the RMS Titanic with SLAM information filters., In Robotics: Science and Systems, 2005.

[77] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli and G. Sukhatme, Autonomous deployment and repair of a sensor network using an unmanned, in Robotics and Automation, 2004. Proceedings.ICRA'04. 2004 IEEE International Conference on, vol. 4. IEEE, 2004,pp. 3602–3608.

[78] G. Buskey, J. Roberts, P. Corke, P. Ridley and G. Wyeth, Sensing and control for a small-size helicopter, in Experimental Robotics, B. Siciliano and P. Dario, Eds. Springer-Verlag, 2003, vol. VIII, pp476 487.

[79] S. Saripalli, J. M. Roberts, P. I. Corke, G. Buskey and G. S. Sukhatme, A tale of two helicopters, in IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, USA, Oct 2003.

[80] S. Saripalli, J. F. Montgomery and G. S. Sukhatme, Visually-guided landing of an unmanned aerial vehicle, IEEE Transactions on Robotics and Automation, vol. 19, no. 3, pp. 371–381, June 2003..

[81] O. Shakernia, T. J. K. Y. Ma and S. S. Sastry, Landing an unmanned air vehicle:vision based motion estimation and non-linear control, in Asian Journal of Control, vol. 1, September 1999, pp. 128–145..

[82] T. Suzuki, R. Sugizaki, K. Kawabata, Y. Hada and Y. Tobe, Autonomous Deployment and Restoration of Sensor Network using Mobile Robots, International Journal of Advanced Robotic Systems, Vol. 7, No. 2 (2010).

[83] Ollero, M. L. C. Anibal, M. Bernard and L. v. Hoesel, Aware:Platform for autonomous self-deploying and operation of wireless sensor-actuator networks cooperating with unmanned aerial vehicles, in Proceedings of the 2007 IEEE,International Workshop on Safety,Security and Rescue Robotics. IEEE, 2007..

[84] E. Gat, Integrating Planning and Reacting in a heterogeneous asynchronous architecture for controlling Real-World mobile robots, AAAI-92,Proceedings, 1992.

[85] R. Reid, A. Cann, C. Meiklejohn, L. Poli, A. Boeing and T. Braunl, Cooperative multi-robot navigation, exploration, mapping and object detection with ROS, Intelligent Vehicles Symposium (IV), 2013 IEEE,IEEE.

[86] M. A. Batalin and G. S. Sukhatme, The Analysis of an Efficient Algorithm for Robot Coverage and Exploration based on Sensor Network Deployment, Proceedings of the 2005 IEEE,International Conference on Robotics and Automation, 2005.

[87] C.-Y. Chang, J.-P. Sheu, Y.-C. Chen and S.-W. Chang, An obstacle-free and power-efficient deployment algorithm for wireless sensor networks, Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, vol. 39, no. 4, pp. 795–806, 2009..

[88] S. S. Dhillon and K. Chakrabarty, Sensor placement for effective coverage and surveillance in distributed sensor networks., IEEE, 2003,vol. 3..

[89] X. Li and N. Santoro, ZONER: A ZONE-based sensor relocation protocol for mobile sensor networks., In Proceedings of the 31st IEEE Conference on Local Computer Networks, Tampa, FL, November 2006, pp. 923 930..

[90] X. Li, N. Santoro and I. Stojmenovic, Mesh-based sensor relocation for coverage maintenance in mobile sensor networks., In Ubiquitous Intelligence and Computing Vol.

4611 of Lecture Notes in Computer Science, Jadwiga Indulska, Jianhua Ma,Laurence Yang, Theo Ungerer, and Jiannong Cao, Eds. Springer, Berlin, Heidelberg, 2007,pp. 696–708..

[91] J. Wu and Z. Jiang, A hierarchical structure based coverage repair in wireless sensor networks., In Proceedings of the 19th Int'l Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Cannes, France, 2008, pp. 1–6..

[92] G. Fletcher, X. Li, A. Nayak and I. Stojmenovic, Randomized robot-assisted relocation of sensors for coverage repair in wireless sensor networks., In Proceedings of the 72nd IEEE Vehicular Technology Conference (VTC-Fall), Ottawa, Canada, October 2010, pp. 1–5..

[93] R. Falcon, X. Li, A. Nayak and I. Stojmenovic, The one-commodity traveling salesman problem with selective pickup and delivery: An ant colony approach., In IEEE Congress on Evolutionary Computation (CEC), Barcelona, Spain, 2010, pp. 4326–4333..

[94] R. Falcon, B. i. Depaire, A. Caris, X. Li, K. Vanhoof, A. Nayak and I. Stojmenovic, A Hybrid Metaheuristic Approach to Sensor Relocation with a Single Robot., Manuscript submitted..

[95] X.-S. Yang, Nature-Inspired Metaheuristic Algorithms, 2nd ed. Luniver Press, UK, 2010.

[96] M. Sharifi, S. Sedighian and M. Kamali, Recharging sensor nodes using implicit actor coordination in wireless sensor actor networks., Wireless Sensor Network 2:123–128, 2010..

[97] https://www.cyberbotics.com/guide/introduction-to-webots.php.

[98] R. Mulligan, Coverage in Wireless Sensor Networks: A Survey,, Network Protocols and Algorithms ISSN 1943-3581 2010, Vol. 2, No. 2.

[99] C. Zhu, C. Zheng, L. Shu and G. Han, A survey on coverage and connectivity issues in wireless sensor networks, Journal of Network and Computer Applications, Volume 35, Issue 2, March 2012, Pages 619–632.

[100] S. Simmons, R. Hershberger and T. Smith, First Results in the Coordination of Heterogeneous Robots for Large-Scale Assembly, Proceedings of the International Symposium on Experimental Robotics (1SER), 2000..

[101] B. P. Gerkey and M. J. Mataric, Multi-Robot Task Allocation: Analyzing the Complexity and Optimality of Key Architectures,, IEEE International Conference on Robotics and Automation (ICRA 2003) September, 2003..

[102] Deneubourg, J. Louis, G. Theraulaz and R. Beckers, Swarm-made architectures, In Proc. of the European. Conf. on Arti£cial Life (ECAL), pages 123–133, Paris, France, 1991..

[103] F. Y. Edgeworth, Mathematical Psychics: An Essay on the Application of Mathematics to the Moral Sciences, 1881, . Translated and reprinted, Augustus M. Kelley, New York, 1967.

[104] Morgenstern, J. v. Neumann and Oskar, Theory of Games and Economic Behavior. J. Wiley, New York, 3 rd edition, 1964.

[105] D. P. Bertsekas, The Auction Algorithm for Assignment and Other Network Flow Problems: A Tutorial, Interfaces, 20(4):133–149, July 1990.

[106] P. Ballal, A. Trivedi and F. Lewis, International Journal of Advanced Robotic Systems, Vol. 5, No. 3 (2008).

[107] [Online]. Available: https://www.cyberbotics.com/guide/introduction-to-webots.php.

[108] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, Wireless sensor networks: a survey,, Computer networks38 (4) (2002) 393-422..

[109] H. Durrant-Whyte, D. Rye and E. Nebot, Localisation of automatic guided vehicles, In G. Giralt and G. Hirzinger, editors, Robotics Research: The 7th International Symposium (ISRR'95) pages 613–625. Springer Verlag, 1996..

[110] X. Li, H. Frey, N. Santoro and I. Stojmenovic, Strictly Localized Sensor Self Deployment for Optimal Focused Coverage., IEEE transactions on mobile computing, vol. 10, no. 11, November 2011.

[111] B. Stefano, C. Marco, G. Silvia and S. Ivan, Mobile Ad Hoc Networking: Cutting Edge Directions, JohnWiley & Sons, Inc., 2013.

VITAE

- Name: Mohammed Azharuddin
- Nationality: Indian
- Date of Birth: 12 – January – 1988
- Email: azhar46@gmail.com, azhar.md@saudirailways.org
- Permanent Address: Hyderabad, India.
- Bachelor of Engineering (B.E.) in Computer Science, Osmania University, Hyderabad, India.
- Master of Sciences (M.Sc) in Computer Networks, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.
- Work Experience:
  - **IT Consultant** at Saudi Railways Organization (2011-2016)
  - **Sr. Software Engineer**, Infosys Technologies Limited, Hyderabad, India. (2009-2011)
  - **Software Engineer Intern**, Electronics Corporation India Limited (ECIL), India. (2009)
  - **Software Engineer Intern**, Google Technologies, India (2008 2009)
- Publications:

- Uthman Baroudi, Md. Enamul Haque and Mohammed Azharuddin; SCAN: A Sensor Deployment Algorithm in Hazardous Regions via Mobile Actuators.
- Uthman Baroudi and Mohammed Azharuddin; Robot assisted real world implementation Of Sensor deployment algorithms in Hazardous Regions, (Submitted).

- Patents:
  - Uthman Baroudi, Md. Enamul Haque and Mohammed Azharuddin; APPARATUS AND METHOD OF SENSOR DEPLOYMENT.