Development of Architectural Framework for Formation Control of Swarm robots

Thesis submitted by

Dibyendu Roy

Thesis submitted for the

Doctor of Philosophy (Engineering)

Electrical Engineering Department
Faculty Council of Engineering & Technology
Jadavpur University
Kolkata, India

2022

JADAVPUR UNIVERSITY
KOLKATA – 700032, INDIA

INDEX NO. **189/17/E**

1. Title of the thesis :

Development of Architectural Framework for Formation Control of Swarm robots

2. Name, Designation & Institution of the Supervisors :

Dr. Madhubanti Maitra
Professor,
Electrical Engineering Department
Jadavpur University
Kolkata, 700032, India.

Dr. Samar Bhattacharya
Former Professor,
Electrical Engineering Department
Jadavpur University
Kolkata, 700032, India.

3. List of Journal Publications:

   (a) **Dibyendu Roy**, Arijit Chowdhury, Madhubanti Maitra and Samar Bhattacharya, "Geometric Region-Based Swarm Robotics Path Planning in an Unknown Occluded Environment", in *IEEE Transactions on Industrial Electronics*, vol. 68, no. 7, pp. 6053-6063, July 2021.

   (b) **Dibyendu Roy**, Madhubanti Maitra and Samar Bhattacharya, "Exploration of Multiple Unknown Areas by Swarm of Robots Utilizing Virtual-Region-Based Splitting and Merging Technique," in *IEEE Transactions on Automation Science and Engineering*, 2021.

   (c) **Dibyendu Roy**, Madhubanti Maitra and Samar Bhattacharya, "Adaptive formation-switching of a multi-robot system in an unknown occluded environment using BAT algorithm," in *International Journal of Intelligent Robotics and Applications (IJIRA)* 4, 465–489 (2020). https://doi.org/10.1007/s41315-020-00150-3.

4. List of Patents: **NIL**

5. List of Presentations in National/International Conferences and Workshops:

   (a) **Dibyendu Roy**, Arijit Chowdhury, Madhubanti Maitra and Samar Bhattacharya, "Virtual Region based Multi-robot Path Planning in an Unknown Occluded Environment," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 588-595, doi: 10.1109/IROS40897.2019.8968177.

   (b) **Dibyendu Roy**, Arijit Chowdhury, Madhubanti Maitra and Samar Bhattacharya, "Multi-Robot Virtual Structure Switching and Formation Changing Strategy in an Unknown Occluded Environment," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4854-4861.

   (c) **Dibyendu Roy**, Arijit Chowdhury, Madhubanti Maitra and Samar Bhattacharya, "Robust Path Planning of Swarm Robots using PSO assisted Bacterial Foraging," *In The Second Workshop "Evaluating General-Purpose AI"(EGPAI2017) in conjunction with IJCAI*, 2017.

   (d) **Dibyendu Roy**, Madhubanti Maitra and Samar Bhattacharya, "Study of formation control and obstacle avoidance of swarm robots using evolutionary algorithms," *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2016, pp. 003154-003159, doi: 10.1109/SMC.2016.7844719.

"Statement of Originality"

I, **Dibyendu Roy**, registered on 07.08.2017 do hereby declare that this thesis entitled ***"Development of Architectural Framework for Formation Control of Swarm robots"***, contains literature survey and original research work done by the undersigned candidate as part of Doctoral studies.

All information in this thesis have been obtained and presented in accordance with the existing academic rules and ethical conduct. I declare that, as required by these rules and conduct, I have fully cited and referred all materials and results that are not original to this work.

I also declare that I have checked this thesis as per the "Policy on Anti Plagiarism, Jadavpur University, 2019", and the level of similarity as checked by iThenticate software is 7%.

Signature of Candidate: Dibyendu Roy

Date: 20/06/2022.

Certified by Supervisors:
(Signature with date, and seal)

1. Madhubanti Maitra 20.06.2022

2. Debalina Roy 20220620

Professor
Electrical Engineering Department

Former Professor
Electrical Engineering Department
JADAVPUR UNIVERSITY
Kolkata - 700 032

CERTIFICATE FROM THE SUPERVISORS

This is to certify that the thesis entitled "***Development of Architectural Framework for Formation Control of Swarm robots***", submitted by **Sri Dibyendu Roy**, who got his name registered on 07/08/2017, for the award of **Ph.D. (Engg.)** degree of Jadavpur University, is absolutely based upon his own work under the joint supervision of **Prof. Madhubanti Maitra** and **Prof. Samar Bhattacharya** and that neither his thesis nor any part of the thesis has been submitted for any degree/diploma or any other academic award anywhere before.

Madhubanti Maitra 20.06.2022

(Prof. Madhubanti Maitra)

Professor
Electrical Engineering Department
JADAVPUR UNIVERSITY
Kolkata - 700 032

20220620

(Prof. Samar Bhattacharya)

Former *Professor*
Electrical Engineering Department
JADAVPUR UNIVERSITY
Kolkata - 700 032

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisors, Prof. Madhubanti Maitra, and Prof. Samar Bhattacharya, for their patient efforts in deciding upon the problem that eventually led to this thesis. Without their iterative feedback and suggestions, it would not have been possible to come up with this direction of research. I am grateful to them for their painstaking review of my work and meaningful suggestions, which ultimately has given shape to this thesis. I could not have imagined having a better advisor and mentor for my PhD study.

I am thankful to the members of the PhD Research Committee for their review and valuable inputs on this research work. I am also thankful to the Head of the Department of Electrical Engineering, Jadavpur University, for allowing me to pursue the research work. I would also like to express my heartily thank Mr. Sreejeet Maity and Mr. Arijit Halder, students in the Department of Electrical Engineering, Jadavpur University, for their invaluable assistance in advancing this study.

I would like to thank Tata Consultancy Services (TCS) and Dr. Arpan Pal, Chief Scientist and Research Area Head, Embedded Systems and Robotics, TCS Research, for their kind permission to carry out my entire research work and successfully bring it to execution, while myself being an employee of TCS. I would also like to thank all of my colleagues of TCS Research, who have directly or indirectly provided me the motivation and mental support for this research work, with a special mention to Dr. Aniruddha Sihna, Principal Scientist, Dr. Tapas Chakravarthy, Principal Scientist, and Mr. Avik Ghose, Principal Scientist. I would like to acknowledge the support of Mr. Arijit Chowdhury, Scientist in TCS Research, for his kind assistance in developing the mathematical background of this research work.

Last but not the least, I deeply thank my parents, Mr. Sankar Roy and Mrs. Runu Roy for their love and inspiration throughout this work. Finally, I thank my wife, Pratyasha, for her unwavering support and encouragement during the writing of this thesis and my life in general.

Dibyendu Roy . 20/06/2022

Dibyendu Roy

“This work is dedicated to my beloved parents, who have spent their whole life, in providing the best of everything required for me, to be where I am today and continue in future ...”

Abstract

Autonomous robots have played a significant role in civic applications in recent years. Automated search and rescue mission is one such use case. In this application, a robotic swarm has the potential to significantly improve the efficiency of the system with faster search and rescue of victims, early evaluation and mapping of the environment, real-time monitoring, and surveillance operations. On the other hand, the biological creatures often work together in groups in order to search for food, such as swarming bees, flocking birds, schooling fishes, etc.

Motivated by the swarming characteristics of the creatures in order to surveillance an unknown environment efficiently, a swarm of robots could be employed. However, the fundamental objectives to model such a system are to preserve coherent behaviors and respond effectively to the environmental stimuli during navigation across the environment. In this dissertation, our major aim is to establish a fault-tolerant cohesive swarm searching for a target or multiple targets while navigating through an unknown occluded environment. The proposed approach utilizes a novel trajectory planning scheme for a robotic swarm employing a region-based shaped control technique to ensure that the system maintains inert-agent cohesion throughout the mission. The control rules are modified during the exploration phases to avoid challenging barriers based on the sensory information of all the agents in the swarm. As a result, the adaptive feature of the control approach has been established to steer the system towards the target. Subsequently, we witness adaptive inter-agent formation switching among the swarm during navigation. Furthermore, in multi-path scenarios, the system can split into sub-swarms and rejoins (if required).

Hence, the proposed approach can effectively be used in various environmental conditions, including single or multiple targets, single or multiple pathways, and so on, such as a typical rescue operation in which several victims may be found in different locations.

To validate the efficacies of the proposed control strategy as well as to examine the scalability in terms of the number of agents in the swarm, extensive simulation studies along with real-time experiments have been performed in different environmental conditions, as presented in this thesis.

List of Acronyms

aABC – Arrhenius Artificial Bee Colony
ABC – Artificial Bee Colony
ACO –Ant Colony Optimization
APF – Artificial Potential Field
BAT – BAT Algorithm
BG – Bacterial Foraging Optimization Algorithm
BGPSO – Bacterial Foraging with Particle Swarm Optimization
CC – Convergence Controller
CG – Circle-based gathering
CI – Probability of cohesiveness
COV – Coefficient of variation
DE – Differential Evaluation
EC – Environmental Complexity
EFA – Elliptical Fourier analysis
FA – Firefly Algorithm
FE – Formation error
FOV – Field of View
GA – Genetic Algorithm
GPS - Global Positioning System
GNSS – Global Navigational Satellite System
H/W – Hardware
IQR - Inter-quartile Range
L-F – Leader-Follower
MD – Minkowski difference
MRTA – Multi Robot Task Allocation
NPSO – Negative Particle Swarm Optimization
NSGA-II – Non-dominated sorting genetic algorithm
OAC – Obstacle Avoidance Controller
PRM - Probabilistic road-map
PSO – Particle Swarm Optimization

PSOABG – Particle Swarm Optimization assisted Bacterial Foraging
RFID - Radio Frequency Identification
sd – Standard deviation
SI – Swarm intelligence
SONAR – Sound Navigational and Ranging
SPACO – Shortest Path Ant Colony Optimization
S/W – Software
UAV – Unmanned Aerial Vehicle
UGV – Unmanned Ground Vehicle
TC – Time Complexity
TVI - Travel variance index
VS – Virtual Structure

List of Symbols

a – Major axis of Ellipse
A_c – Area of the circular region
AD – Average Euclidean distance during dynamic obstacle avoidance
A_e - Area of Ellipse
A_o - Loudness
A_Φ - Area of the polygonal contour Φ
A_{min} - Minimum loudness
$\bar{A}^t$ - Average loudness
α_e - Orientation of Ellipse
b – Minor axis of Ellipse
$B_{r_c}(O_c)$ - Circular Region
$B_{r_e}(O_e)$ - Elliptical Region
β_i - Orientation of the i^{th} agent from world coordinate frame
b_{fs} - Free space
$\alpha_n, \beta_n, \gamma_n, \delta_n$ - EFA coefficients
α_0, β_0 - Horizontal and vertical offsets
$\circ$ - Hadamard Product
$c_s = \{c_s^1, c_s^2, ..., c_s^N\}$ - Coordinates of packed circles
$\ominus$ - MD operator
D - Structural Density
δ - Repulsion control action triggering parameter
Δ - Adjacency matrix
D - Degree matrix
d_i – Sensed distance of the obstacle by the i^{th} agent
e^i - Error margin
$e_{pi} = x_i - \bar{x}$ - Position error of the i^{th} agent
$e_{vi} = v_i - \bar{v}$ - Velocity error of the i^{th} agent
$\emptyset$ - Null set
F_i - Distance matrix of the i^{th} agent from its neighbors
f_i - Frequency

f_{max} - Maximum Frequency
G – Graph
k - Number of points in Φ
k_i^a - Positional Attraction gain of the i^{th} agent
k_i^{OA} – Obstacle avoidance gain of the i^{th} agent
k_c^i - Convergence gain of the i^{th} agent
k_f^i – Formation gain of the i^{th} agent
k_i^r - Repulsion gain of the i^{th} agent
k_i^v - Velocity Attraction gain of the i^{th} agent
$k_{v_{cir}}$ - Translational velocity controller's gain
L – Graph Laplacian
L^j - Leader agent of the j^{th} sub-swarm
L_V - Travel variance index
L_U – Lattice structure
$l^i \in [l_1^i, l_2^i]$ - local minimum points around i^{th} peak
λ - Eigenvalues
λ_{min} - Minimum eigenvalue
m – Number of distance sensors
n - Fourier Harmonics
N - Number of agents
N' - Number of failure agents
N_i - Neighboring of agents
$O = \begin{bmatrix} O_x & O_y \end{bmatrix}$ - Location of the obstacle
$O_c = \begin{bmatrix} x_c & y_c \end{bmatrix}$ - Coordinate of Circle's center
$O_e = \begin{bmatrix} x_e & y_e \end{bmatrix}$ - Coordinate of Ellipse's center
Ω - 2D Topological space
$\Omega^i(t+\tau_1)$ - i^{th} sub-swarm
$\Phi = [\Phi_x \quad \Phi_y]$ - Virtual Boundary
r - Pulse emission rate
r_c - Radius of circle
ρ - Desired inter-agent distance matrix
r_i^s - Repulsion region of the i^{th} agent
$\tilde{R}_i = \begin{bmatrix} \tilde{R}_i^x & \tilde{R}_i^y \end{bmatrix}^T$ - Predicted positions of obstacles
R_s - Minimum distance requirement for cohesive swarm
S – Spanning Tree
$S = (S_1, S_2, ..., S_R)$ - Number of identical circles
SG^i - Number of agents in the i^{th} sub-swarm
SO - Sensing an obstacles by an agents

SOI - Obstacle sensing index
T - Location of target
Θ_i - Optimization function
θ_i - Orientation of sensor of the i^{th} agent from world coordinate frame
U - Total number of obstacles
u_a – Control input for major axis of the ellipse
u_{α_e} – Orientation control input
u_b – Control input for minor axis of the ellipse
UA_Φ - Unused area within Φ
$u_i = \begin{bmatrix} u_i^x & u_i^y \end{bmatrix}$ - Control input of the i^{th} agent
u_i^a - Attraction control input of the i^{th} agent
u_{cc}^i - Convergence control of the i^{th} agent
u_f^i - Formation control of the i^{th} agent
u_i^{OA} – Obstacle avoidance control input of the i^{th} agent
u_i^r - Repulsion control input of the i^{th} agent
u_θ - Angular control input
u_v - Velocity control input
v – Number of vertices
$\bar{v}$ - Average velocity of the swarm
$v_i = \begin{bmatrix} v_i^x & v_i^y \end{bmatrix}$ - Velocity of the i^{th} agent
v_e - Velocity vector of Ellipse
$\bar{x}$ - Average position of the swarm
ξ - Wavelength
$x_i = \begin{bmatrix} p_i^x & p_i^y \end{bmatrix}$ - Position of the i^{th} agent
$x_{ij} = ||x_i - x_j||$ - Euclidean distance between the i^{th} and j^{th} agents
$x_j = \begin{bmatrix} p_j^x & p_j^y \end{bmatrix}$ - Position of the j^{th} agent
x_* - Global best solution
z – Sensor's spacing
ζ - Direction and intensity of the random walk

Table of contents

List of figures

List of tables

Chapter 1

Introduction

A robot is a device that is used to replicate human activities. It can perform a series of tasks autonomously without any human intervention (if it is properly programmed). The field in which such frameworks are studied and programmed, is typically called robotics [4]. It connects several interdisciplinary engineering domains such as mechanical, electrical, computer science, electronics, etc. for the productive operation of robotic hardware. Robots can be employed in several circumstances where human efforts cannot be utilized because of environmental criticality and uncertainty. Nowadays, robots are extensively being used in super-fine surgical procedures, rescue and search operations, nuclear industry, high-temperature manufacturing industry, radioactive process industry so and so forth [5], [6], [7]. Thus, robots can replace human interventions in many hazardous conditions as well as in situations where human intrusions are not at all feasible.

The notion behind inventing robots to operate autonomously emerges in the 20^{th} century. The first physical robot has been constructed in 1961 for uplifting and piling the hot metal parts from a die casting machine [8]. From the development perspective of these robots, it was realized that one day the robots will mimic human behavior and execute tasks in a human-like fashion and will be part and parcel of modern-day society. Considering this scenario, the researchers and scientists started to explore various application areas where a unitary robot or several robots can be used to perform tasks thereby either supplementing or replacing the human endeavor. Consequently, robotic science and technologies started taking shapes engulfing the areas like modeling, analysis, and synthesis of controlled dynamics of a solitary robot or a group of robots to cater to the need of a specific and specialized job. With the evolution of robotics or to be more precise, mechatronics as a discipline, presently, robots are being employed commercially to perform distinct tasks more accurately and reliably than human beings in an inexpensive way. The realms which have embraced robotic mediations in serious practice are the medical fraternity, manufacturing and assembly industries, mining, transportation trades, disaster management, and military applications, just to name a few [9], [10], [11].

However, from its nascent state till date the robotic science is more concerned about investigating the efficacy of a single robotic framework like a single robot manipulator [12] or gripper [13] and eventually, they are like to perform a series of sequential tasks.

Fig. 1.1 The Rescue robot, called RoboCue [1].

If we take an example of the rescue robot, namely RoboCue (as shown in Fig. 1.1) [1], which had been developed to assist the operation team in search and rescue [14] of humans in a disaster management situations like earthquake or Tsunami, we would observe that to execute the mission successfully, the robot needs to perform a serial of sequential activities such as searching and locating casualties, mapping of the environment, removing rubbles, etc. It also contains a manipulator and gripper section (attached to the head) to provide medical assistance and rescue the victims. For detecting trapped humans and gently loading the injured person onto the cart, it uses several sensors like ultrasonic sensors, expensive infrared cameras, high-resolution vision sensors, etc. So, the hardware architecture of such systems becomes complex [15]. Furthermore, to prioritize the task to be performed, it has to decide (on its own) depending on the sensory data. Depending on the priority of the task, the actuation will take place which is also reliant on the sensory information. Hence, to achieve each task efficiently, the robot has to perform several activities, and consequently, the system consumes a huge execution time [16] to complete the mission.

In light of the above example, we can list the following generic challenges that may arise for any kind of robotic system [4].

1. **Robustness:** The robustness of a robotic framework defines the satisfactory performance of the system in the presence of inaccurate sensing inputs or the failure of certain modules of the system. As the mission objectives are entirely dependent on the performance of the robot, so, even in the presence of erroneous sensing data or the failure of certain modules, the system should behave appropriately.

2. **Execution Time:** The time of execution to finish the mission is another important feature that requires to be addressed when working in robotics. Considering the same example as described earlier, if the robot consumes a large amount of time to complete the mission, then, the number of casualties might increase. So, the time of execution should be as low as possible.

3. **Handling Environmental Uncertainties:** While working in a disaster-like environment, a robot may face several environmental uncertainties. Thus, it should be programmed in such a way that it has the capability to overcome such uncertainties. Otherwise, the performance of the system may degrade, and it would create a severe impact on the mission accomplishment.

4. **Autonomous Navigation:** Finally, to navigate across the unknown environment, the robot needs to plan its trajectory autonomously (i.e., without the help of any human operator) and accordingly map the environment. The environmental mapping could assist the rescue team to find the victim's location.

To resolve the above challenges, the common idea is to deploy a number of robotic agents instead of a single unit. The key idea behind this fact is that by employing such a framework, the time of execution to complete the mission will significantly be reduced as the environment will now be explored by several robots. Moreover, the robustness of the system is guaranteed as if a robotic agent fails during the operation, its tasks could be performed by another agent (i.e., in a similar manner as a human being does). To formulate such a system, it is to be presumed that each robot can accomplish several activities with the aid of other robots in the team. So, the entire task will eventually be distributed among them. The advantages of such a system are listed below:

1. *Time efficient solution:* As multiple robots are working together, so the effective execution time to complete the mission will be reduced than the unitary one.

2. *Enhancing robustness:* During the mission, if any of the agents fails, its responsibilities could be performed by another agent in the group, thus the robustness of the system increases more than the solitary framework.

3. *Handling environmental uncertainty:* To prioritize the list of tasks, the system could employ the sensing data of each robot of the team. Depending on this information, the system could detect the upcoming uncertainties of the environment and act on them accordingly.

4. *Flexibility of the system:* As multiple robots are working together as a team, hence, robots could dynamically allocate themselves to various tasks to match the necessities of the specific environments as well as the operating conditions.

5. *Task distribution:* In a multi-robotic system, a complex task could be partitioned into several sub-tasks so that each sub-task is allotted to different robots so that all of the sub-tasks could be performed in a parallel fashion.

The domain of robotics where the multi-robot systems have been studied is generally called *Swarm Robotics* [17], [18], [19].

1.1 Swarm Robotics

Swarm robotics [18] is the subject that describes the coordination among a large number of simple robots to accomplish a desired collective behavior. Such kinds of collective responses emerge from the interactions among the robots with their surroundings. This sort of collective social behavior is commonly observed in zoological species for example, in swarming of bees [20], in flocking of birds [21], or in schooling of fish [22] as represented in Fig. 1.2. If we analyze the behavioral patterns of these creatures, we find that ants do form a line in quest of food; same do the birds for searching for a goal, be it food or the nest at the end of the day. The school of fishes and herd of cattle always swim or move either for food or for evading and eluding the predators. From the previous discussion, it is clear that the animal kingdom does employ its intelligence to survive against all kinds of odds. Hence, the principles that are developed for a swarm of robots are often called *swarm intelligence* [23], [24].

(a) Swarming of Bees [20].

(b) Flocking of Birds [21].

(c) Schooling of Fish [22].

Fig. 1.2 Swarm aggregation principle inspired from the social creatures.

The word *swarm* was first introduced in [25]. In the paper, robots were characterized as cellular systems that could work in an n-dimensional space without the involvement of any centralized unit. They could collaborate among themselves to perform common goals. This idea was then extended in [26] where the group of robots could act like the cells of a human body to accomplish a complex task.

In 1993, the first multi-robot system was developed inspired by the collective behavior of a natural swarm [27]. After that, the research focus on swarm robotics was steered towards understanding and thereafter simulating the cooperative behaviors exhibited by different creatures like birds, fish, and others. Several studies were reported in later stages [28], [29], where the key issue was implanting

the intelligent behaviors of the natural beings artificially in a multi-robotic system. Thus, natural swarming principles had always been the main motivation behind the genesis of the science of swarm robotics.

In this process of evolution of a structured robotic crew, gradually, some important notions were further envisaged [18], [19], like the simplicity of swarm architecture, scalability of the system, the self-organizing criterion of individual member and robustness of the entire group. The concepts were devised to ensure the success of the planned mission. Even in the present day scenario, we find that these properties are still very relevant and influence, at the same time, the basic building blocks of any swarm robotic framework. Moreover, those characteristics are considered to be the benchmarks for evaluating and distinguishing the efficacy of a swarm robotic system over the others employed for the same assignment.

The next sub-section elaborates on the necessary properties of a swarm network which have been derived from nature itself. The basic premise of our work is the stable formation of swam. Therefore, in the following section, we would like to elucidate the terms robustness, scalability, flexibility, and time-efficiency associated with the congregation of robots [19], [23], [24].

1.1.1 Robustness

In swarm robotics, one major challenge is that the system needs be fault-tolerant [30], [31], [32], i.e., failure of an agent or agents will not create any hindrance towards the fulfillment of the mission. Generally, failures of agents mainly occur due to failure of a sensor or actuator [31], [33]. To ensure this robustness study [24] of the swarm system requires serious consideration. The term *robustness* is defined as the degree to which a system can continue its functionalities in the face of partial failure of its agents or when subjected to some other abnormal conditions. Therefore, in swarm robotics, robustness refers to the capability of the system to work even if some of the individuals fail or there are unpredictable situations in the environment. The social insects are extremely robust in nature as they can finish their task successfully even after losing a few of their neighbors [34], [35].

In this work, we have sincerely addressed this issue and have devised a robust swarm network.

1.1.2 Scalability

Scalability [36], in general, of a network, is the ability to perform satisfactorily with widely varying dimensions. The addition or elimination of individual elements or agents does not result in a drastic change in the performance of the system. In the parlance of the present work, to warrant the scalability aspect of a swarm of robots, it is necessitated that the associated control laws are executed perfectly with any number of robots in the team. Thus, the performance of the system should be consistent enough with a varied number of agents in the team. One important advantage of a scalable swarm is that if an agent fails to perform its responsibilities during the mission, its pending tasks could be performed by another agent in the group.

Scalability has been thoroughly studied and ensured as far as our swarm robotic frameworks, designed in this work, are concerned.

1.1.3 Flexibility

Flexibility [37] is the capability of the swarm system to adapt to changing environments. A swarm robotic system should be developed in such a manner that it will be competent enough to deal with a wide spectrum of environments and operating conditions. An environment may not be static always; it might have several dynamic conditions that may obstruct the motion of the swarm towards the destination. In those situations, flexibility, as well as adaptability, enables the swarm to self-organize itself to fit into the occluded surroundings. In other words, the flexibility and adaptability of a system allow the robots to dynamically position themselves to persist within a changing environment. In this work, we have extended the notion of flexibility to include the fact that the robots do pursue their tasks depending only on the philosophy of connecting and communicating. Here, connect implies connecting to the sensors, and communicate means sharing information gathered via a connection. Connect and communicate mode of operations do not require any infrastructure-based communication network.

Accordingly, the goal-specific swarm navigation schemes proposed in this work always assure the flexibility and adaptability of those despite circumstantial changes.

1.1.4 Time efficiency

Finally, the task completion time [38] is considered one of the major performance criteria of a swarm robotic system. For a given task, the system needs to be intelligent so that it will consume the minimum possible time to complete the task. Therefore, while designing such systems, it is recommended to devise the overall control algorithms in such a way that a specific task or a series of sequential tasks are completed in a time-efficient manner. To address this issue, in this work, we have assumed a scenario in which a swarm of robots can approach a predefined goal in an unknown occluded environment in minimum time. Moreover, we have endeavored to consider this aspect and consequently have proposed a split-joint mechanism which we shall discuss later on in detail.

To conclude this sub-section, we can state that inspired by the biological systems, the researchers and scientists endeavor to develop robotic systems that manifest swarm intelligence features similar to the characteristics portrayed by the social animals. Hence, it is expected that a swarm robotic system should be robust, scalable, flexible, and time-efficient in dynamic circumstances, and this work has been carried out to present some new directions toward swarm navigation appropriately emphasizing these measures.

1.2 Challenges in Swarm Robotics

Several potential challenges are required to be solved while working in the field of swarm robotics [19], [39]. Those are listed below.

1.2.1 Centralized vs. Decentralized Motion Planning

The first and foremost challenge that needs to be addressed while developing a swarm robotic system is the motion planning of all the agents in the team [19], [7]. The term motion planning in robotics defines a computational problem for attaining the continuity of movements of a robotic agent from the source to the destination in dynamic circumstances such that the effective time consumption and path length to reach the goal are less [40]. For a multi-robot system, the sequence of movements of each robot should be computed in such a manner that there would not be any collision with the nearby agents as well as impending obstacles. Moreover, as several robots are required to be controlled, thus, the planning should be performed in a distributed fashion i.e., without any involvement of centralized control architecture [41]. However, while planning the movement of the swarm in an obstructed environment, researchers are often divided into groups based on their preferences for centralized control or decentralized control [42]. Though each control proposition either cited in the literature or being practiced [42] has its own merits and demerits, we here report two classical cases where the conflict between the control schemes can be understood simply intuitively.

(a) Motion planning in solitary path condition. (b) Motion planning in multiple path condition.

Fig. 1.3 Challenges in motion planning of a swarm robotic framework.

Let us assume two scenarios as shown in Figs. 1.3a, and 1.3b, where a swarm of robots is required to pass through the obstacles. In the first case, there exists a single path that directly connects the initial location of the swarm and the ultimate goal. However, in contrast, we see that in the second case, two different paths are available to the swarm for reaching the target. It is needless to say that in both cases, the swarm should be able to identify and avoid the obstacles faced along. The obstacle avoidance is facilitated with the help of sensing information gathered by the agents. In subsequent chapters, we shall discuss how sensing information helps in avoiding the obstacles at length. For the time being, let us consider the situations portrayed above. The solitary path scenario poses typical challenges associated with the motion planning of a group of robots. However, the

multi-path condition opens up two distinct possibilities based on which the agents can pass through the obstacles. Those possibilities are

1. Robots can approach the target along any single path as dictated by a centralized control algorithm [43].

2. They can split into sub-swarms and can explore both paths simultaneously. Hence, as investigated further, a decentralized control scheme would be necessary [44].

In this context, we would like to reiterate our basic premise that all the agents in a robust formation should reach the goal to execute the tasks assigned to them. In continuation to the above consideration, we can observe that out of these two possibilities, the second approach is temporally more optimal than the previous one as it provides a concurrent exploration of the entire environment.

Nonetheless, it has been observed and established in our work that a central control scheme fails to accomplish such kind of optimum motion planning scheme. Therefore, in the swarm robotic motion planning system, the main objective is that each agent in the swarm can autonomously plan its path utilizing its sensing ability. Hence, during navigational steps, any decision has to be taken by the robot itself so that the entire system can move in a decentralized manner. This consequence creates a major challenge to develop a swarm robotic system.

1.2.2 Formation Control

The second important aspect which arises in this context is the formation control among the swarm robotic system that describes how a swarm of robots will maintain a desired spatial pattern or shape while navigating through an unknown environment [45]. Maintaining formation during the navigation steps has several advantages over moving individually [46], [47]. One such benefit is to expose only some of the agents to the proximity of enemies or to improve the group's abilities by allowing individuals to limit the perceptual focus to one small part of the environment. Hence, the basic advantage of maintaining formation is to improve the robustness and efficiency of the system.

However, to achieve and maintain a spatial pattern in a swarm robotic system, different control topologies are required to develop, and depending on the environmental condition, any of them can be selected adaptively. To rationalize this statement, the following example is manifested.

Now, let us consider a scenario shown in Fig. 1.4, it is observed that the agents are initially placed randomly. Afterward, while passing through the first set of obstacles, the system creates a hexagonal formation in order to avoid the obstacles. However, as we can see, the next set of obstacles makes the situation critical as the passage in between is constricted, and hence the swarm as a whole with its earlier hexagonal shape will not be able to move through maintaining a bare minimum clearance from the impediments. The bottom line is the formation of the robots, along a complete path with varied kinds of obstacles, should be able to change in accordance with the varying nature of the obstructed surroundings. Hence, formation control calls for an adaptive control mechanism that will serve the above-mentioned purpose.

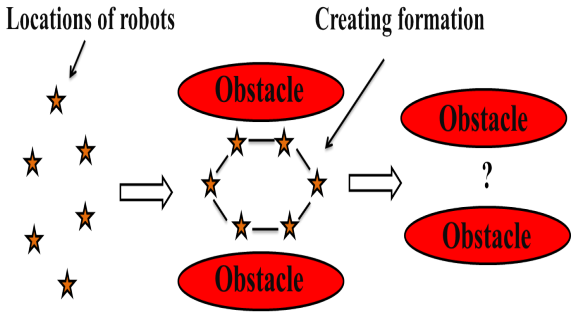


Fig. 1.4 Challenges in formation control of a swarm robotic system.

From the above discussion, it is apparent that a dichotomy is associated with formation control. We express the dichotomy in the form of a question that is

Whether a swam should maintain a fixed formation throughout its journey or it should be allowed to reshape itself, say from a circle to an ellipse, depending on the dynamic nature of the surroundings?

The answer to this question will actually dictate the control law. And we find that there is no contradiction between the situations mentioned in the question posed. Actually in real-life situations, some of which may demand fixed formation, whereas, some other may call for formation switching or change.

For example, maintaining a specific formation is sometimes utterly beneficial to accomplish some archetypal tasks like object carrying, security patrolling, search and rescue mission, etc. [5], [48]. The reason behind this fact is to limit the sensory abilities by allowing individuals to focus only on a given area depending on their position in the formation. This criterion may also increase the ability of the team to navigate and reduce the decision-making time, for example, a flight pattern of a group of drones allows one member to predict the likely positions and movements of others, decreasing the number of factors that must be considered when making a change in movement. For object transportation tasks, strict formation control is always needed as the object being carried can't change its shape. Hence, for this specific application, a strong inter-agent formation amongst the agents of the swarm is essential. However, for the other applications, formation control along with formation switching would be beneficial.

However, in an irregularly blocked environment, maintaining strict formation among a group of robots is not often possible [45]. Maintaining fixed formation even becomes an unacceptable proposition where the blockages change with time or the moving obstacles lead to a completely dynamic milieu. In this kind of scenario, the group should be able to change their contour adaptively to cope with the changing environmental conditions [49]. Therefore, adaptive control laws would be required in such situations for the respective group [50]. The scenario where sustaining a fixed formation, even in a static terrain has been depicted in Fig. 1.3a for clear understanding.

However, formation control needs more deliberation as it encompasses some other challenging concerns down the line as well. The intricate concerns further are the stability of the formation,

controllability of the formation control, robustness issues like safety and uncertainties associated with the formation, etc. [51], [52]. It is obvious that these criteria pose vital challenges, particularly for the swarm robotics navigational problem.

1.2.3 Task Assignment

The main purpose of employing a swarm of robots is to execute a specific task or a sequence of tasks [53]. In this aspect, the most challenging problem for a swarm of robots is how to optimally allocate a set of tasks to a group of robots such that the given assignments are accomplished optimally [54], [55], [56]. This specific problem in robotics is known as *Multi-robot Task Allocation* (MRTA) problem [53], [38], [57]. It is a complex problem especially when it comes to heterogeneous unreliable robots [53] equipped with different skills and capacities which are required to perform various tasks collectively with several requirements and constraints. In a lucid form, the above problem can be expressed as: in an unknown environment, when a specific task is identified, the system should decide which agent (or agents) is (or are) capable of performing this task efficiently? Based on this knowledge, the task is required to be assigned to a robot or a sub-group of robots to perform it successfully.

Now, let us focus on a scenario as shown in Fig. 1.5, where a group of seven agents (as marked by brown stars) are required to perform seven individual tasks (denoted by circles).

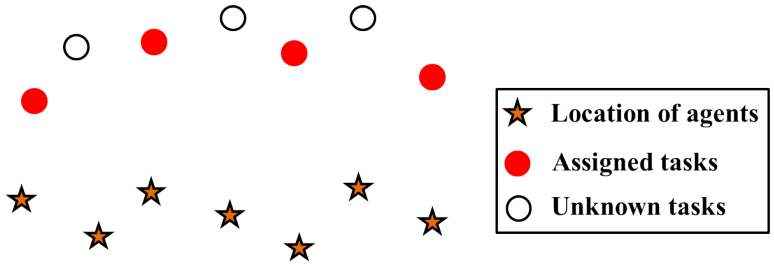


Fig. 1.5 Challenges in task assignment of swarm robotic system.

Some of the tasks are already identified by the system (red circles), and the rests are still unknown to the system (white circles). These tasks can be completed by the swarm either concurrently or sequentially as discussed below.

1. To perform each identified task concurrently, the swarm system will assign an agent or agents. Therefore, to finish four identified tasks (as shown in Fig. 1.5), at least four agents will be allotted. And the remaining agents of the swarm will explore the environment for searching the unidentified tasks. Once an unknown task is recognized, it will be solved immediately by any of the remaining agents as assigned by the swarm system.

2. To complete the identified tasks sequentially, the robotic swarm will approach each task. After finishing this task successfully, the system will progress to the next task, and the process continues until all of the identified tasks have been completed. Once this job is accomplished,

the system would explore the environment for the unidentified tasks and perform them in a similar fashion.

Out of these two approaches, it is obvious that the sequential method consumes a large execution time to finish all the tasks compared to the concurrent-based approach. However, if we carefully look into the above scenarios, we would observe that the sequential scheme is more robust than the concurrent-based approach because it can handle the agent-failure situation, for example, while performing a task, if an agent fails, the failure agent can easily be replaced by one standby agent of the swarm.

Therefore, working with the MRTA problem, we need to focus on reducing the complexity of the task by distributing among several agents [58] or increasing the performance of the system by enhancing the robustness [57].

1.2.4 Inter-agent Communication

It is apparent that a swarm of robots can operate cooperatively and collaboratively only if there exists a reliable communication system (amongst them) that helps in exchanging sensory data, state information, information related to the executable and executed tasks, etc. [59], [60], [61]. Actually, it is quite justified to claim that swarm robotics rest on two major pillars; one is control and the other one is communication [62]. Presently, the field of communication or to be more precise the field of wireless communication is undergoing paradigm shifts every alternate day. Thus, like control design, devising a robust communication strategy with well-defined protocols is the call of the time and requires extensive research in this area [62].

In robotics, generally, three types of communication arrangements are mainly used [6], i) environmental interaction, ii) sensorial interaction and iii) communicational interaction.

For the environmental interaction process [61], an operator or a centralized controller is needed to control the overall system. The function of such a control scheme is to plan the trajectory of an agent. As a swarm robotic system contains more than one agent in the group, so, this protocol is not used in swarm robotics.

The sensorial interaction [63] or the implicit communication system allows the inter-agent interaction amongst the robots. An agent can sense its neighbors by using various sensors attached to it. In this mode of communication, the robot itself performs various navigational functions such as obstacle avoidance, leader following, collision avoidance, etc. by employing its sensory data [62]. To communicate with the neighboring agents via the sensorial interaction topology, the radio or infrared communication system [64] would be required.

The third technique i.e. communicational interaction or explicit communication introduces the immediate transfer of information between two agents as a deliberate form of communication [65]. In this mode of interaction, the sender agent has the goal of sharing specific information like speech or gesture recognition [225] with the collocutor agent. So, it is a direct and deliberate form of

communication where, there is a clear associated intent for the transmitted information to be received by another agent over a communication channel. Thus, to utilize such a communication protocol, an agent must have a high efficient onboard communication component in the form of hardware.

In most of the swarm robotic applications, the sensorial interaction or the implicit communication system has been employed as this protocol allows the robot to share the sensory information with its neighboring agents. In this work, we also utilize a similar scheme for achieving inter-agent communication amongst the swarm system.

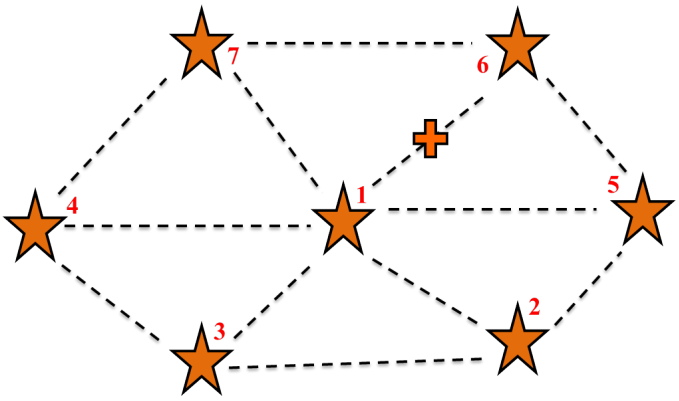


Fig. 1.6 Challenges in inter-agent communication of swarm robotic framework.

1.2.5 Inter-agent Communication with link failure

To describe the hurdles faced in inter-agent communication of a swarm robotic system, as an example, here we present a case in Fig. 1.6 with seven robotic agents. From the diagram, we note that all the agents are initially connected via communication links to their immediate neighbors. Thus, each agent in the team is permitted to interact with its neighboring agents only and as per this logic, we find that agent 6 in the figure is connected to its neighboring agents 1, 5, and 7. Now, after a while during navigation, let us assume that the communication links of agent 6 with its neighboring agents 1, 5, and 6 have been broken down due to some reasons or others.

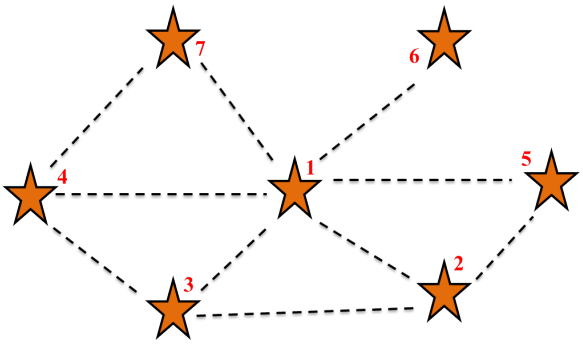


Fig. 1.7 Restructured system with single link establishment.

Hence, in this case, to resume the navigation, the swarm would need to decide upon one of the following possibilities.

1. All of the broken communication links are required to be restored with immediate effect (if possible).

2. If the previous plan fails, then, the system would need to re-establish at least one communication link with the agent 6. Let us assume that the link between agents 1 and 6 is re-initiated. Now, the system is required to decide whether it can progress with the existing setup i.e., with a single communication link between the agent 1 and 6 (as shown in Fig. 1.7.

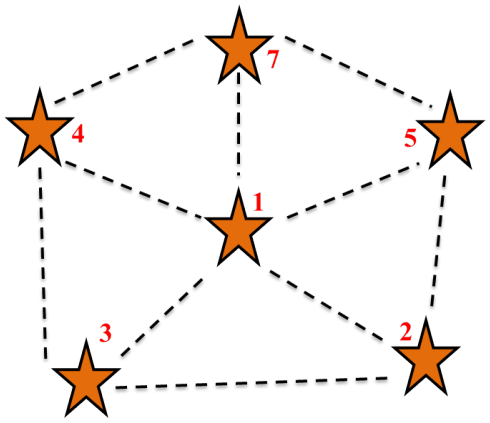


Fig. 1.8 Restructured system by discarding the agent 6.

3. If all of the previous methods (i.e., cases 1 and 2) fail, then the only possibility is that the system would discard agent 6 and continue the navigation with six agents. In this case, a new communication architecture (as shown in Fig. 1.8) is expected to establish among the agents in the swarm; so that, based on this framework, the system can approach further.

1.2.6 Other notable challenges

In swarm robotics, as large numbers of robots are operating, hence, miniaturization and cost-reduction (to design those robots) are the key aspects [39], [66]. Those features are the main constraints in creating a large group of robots; therefore the simplicity of an individual team member should be emphasized. The integrity in the hardware framework often motivates the robotics researchers to implement swarm-intelligent approaches for achieving meaningful behavior at the swarm level instead of the individual level. Thus, the evolution of simple robots for swarm intelligence research is a very important aspect of the field. The goals include keeping the cost of individual robots low to allow scalability, making each member of the swarm less demanding of resources and more power or energy efficient.

1.3 Application areas of Swarm Robotics

The swarm robotics architecture has several benefits over traditional solitary robotic architecture [67] in terms of quick and efficient accomplishment of the mission objectives even in complex circumstances. Moreover, while navigating through a hazardous environment, the disappointment of one robot should not prompt the catastrophe of the entire system [68]. Thus, a swarm robotic system should always be robust to partial agent failure and should be flexible enough to the demands of various missions [30]. In a multi-robot task allocation (MRTA) problem, when a complex task is assigned to a group of robots [53], [57], it is needless to mention that the overall task is partitioned into several subtasks and distributed among the various members of the team. So, the complexity of the task and associated time criticality are reduced significantly in this process. Depending on the above considerations, it is essential to mention that there are several application areas [69], [70] where a swarm of robots can be deployed to perform several activities successfully. Some of them are discussed below.

One of the most promising applications of swarm robotics is in disaster rescue missions [69], [71], [72]. Customarily catastrophic management of a hazardous situation like earthquakes, tsunamis, or even break out of a fire in a high-rise is fought against using mostly human power, where paramilitary forces, commandos, special task forces, or even army peoples are deployed to salvage the situation. Indeed this has been a long-drawn exercise but under these situations, along with the lives of the mass to be rescued, the lives of the rescuers also face life threats. Hence, it has been felt that human-led rescue or even surveillance operations call for automation to a certain degree [73]. Nowadays, quadruped, or hexapod robots, drones, humanoids, or even simple robotic arms or manipulators, which are duly equipped with necessary sensors and actuators, can perform tasks like human beings [74]. Hence, in disaster rescue missions where the rescue workers cannot reach safely to detect the presence of life, it is always beneficial to deploy swarms of heterogeneous robots [72]. Such kinds of systems can also be useful in mining [75] or agricultural area surveillance [76].

Underwater area exploration [72], [75], [77] is a subject of extensive research in the recent decades. The primary focus of this investigation is mostly on military applications such as mine detection [75]. However, recent aerial disasters [224], where the airplane had disappeared in the ocean, indicate the utility of underwater exploration. In such scenarios, a swarm of robots can be employed for search and rescue under seawater [72], intelligent surveillance and reconnaissance [78], multi-vehicle mission [79], and so on.

Presently, underwater areas are mostly inspected by the expensive Sound Navigation and Ranging (SONAR)-equipped surface vessel [14] to navigate the whole surface area using a lawnmower trajectory control approach. This procedure consumes a large investigation time. So, to reduce the execution time of this application area, an autonomous swarm robotic system would be very beneficial for exploring the entire zone. Controlling a swarm of robots in the underwater environment poses several challenges, such as underwater communication, positioning estimation, unpredictable disturbances, etc. [78]. The environmental disturbances due to climate changes, waves, wind, and

ocean currents have a significant impact on robotic movement and stability. Thus, these kinds of disturbances play a vital role in defining the degree of autonomy and mission planning.

Further, in military applications, swarms of robots can form an autonomous army. Recently, the U.S. Naval forces have tested a swarm of autonomous boats that can steer and take aggressive actions by themselves [78]. The boats are unmanned and can be fitted with any kind of small-scale arsenals to prevent and destroy enemy vessels. As a case in point, a report by the Russian forces claimed that during the Syrian Civil War, in 2018 [79], the main air-force base of Russia's vast Hmeimim air base in northwestern Latakia province was attacked by swarms of fixed-wing drones loaded with explosives. Recently, the researchers of Harvard University [80] have devised a shape-modulated cohesive swarm consisting of 1024 robots, and they have also demanded that the swarm is the largest swarm existing in practice till date.

Other mention-worthy application areas where swarms of robots can play key roles are like tracing of chemical plumes [81], exploration of unknown areas [82], and monitoring of environments [83]. Several issues related to these fields may be solved using swarms of micro-aerial vehicles which are also being thoroughly explored nowadays [84], [77], [85]. The pioneering study of invasive maneuvering of aerial swarms has been simulated in a laboratory environment using precise motion capture systems and has been reported too in [86], [87]. Nonetheless, some recent works with Shooting Star drones [46] have demonstrated that aerial vehicles can also be controlled in an outdoor environment. Furthermore, Faigl et al [88] have shown that a team of micro aerial vehicles can fly in a synchronous motion using Global Navigation Satellite System (GNSS) systems (such as Global Positioning System (GPS)) and can even stabilize themselves autonomously using onboard localization systems where GPS is unavailable. In addition to that, there are several application areas where the concept of cooperative swarms in the form of unmanned ground and aerial vehicles has been utilized and some of the areas are cooperative environment monitoring [83], simultaneous localization and mapping [89], caging [90], moving target tracking [91], etc.

A huge progression has already been achieved as far as the real-life application of autonomous swarms is concerned. The noteworthy advancement could be found in the field of manufacturing, known as swarm 3D printing [70]. It is a digital manufacturing platform that utilizes a swarm of robots having different functionalities to operate together for printing and assembling products based on a digital design. The operating principles of this system are:

1. Separating the actual design into smaller components based on the geometry and compatibility of the actual product
2. Allotting each component to a group of specialized robots for printing
3. Finally, assembling those printed materials to develop the actual product as defined by the digital design.

This technology is particularly useful for the production of large structures and components, where traditional 3D printing cannot be utilized due to the size of the manufactured product [92].

While still in its nascent stage of development, swarm 3D printing is currently being commercialized by several startup companies. Rosotics [93] is the first company to demonstrate a swarm 3D printing system using a metallic payload to achieve metallic 3D printing from an airborne platform [94].

So far, we have attempted to present the inception of the concept of swarm robotics, its evolution toward an overwhelmingly matured domain of interest, the challenges and issues associated with the field along with probable application areas. A brief literature survey has helped us to establish the present-day relevance of swarm robotics and the possible scope of carrying out meaningful research endeavors in this area. Accordingly, now we take the privilege to outline our research pursuits that have been reported in this dissertation.

1.4 Contributions of the present work

In this thesis, the motion planning aspect of a swarm robotics framework has been studied to achieve a cohesive swarm that can adaptively change its inter-agent formation to avoid obstacles while approaching the target in dynamic environments. The three main characteristics (such as robustness, flexibility, and scalability) [18] that are generally witnessed in biological creatures are also considered in this work. The robustness and scalability perspectives are managed while creating a cohesive swarm, and the flexibility feature is controlled by consolidating the adaptive formation control scheme. There are four key contributions of the present work that are enumerated below.

1. Our initial work (as explained in Chapter 3) is to achieve complete autonomy of a group of robots (swarms) towards achieving a fixed mission. In this regard, we have proposed evolutionary algorithm-based hierarchical control strategies for navigating the swarm through the landscape to a predefined target position without crashing into static and dynamic obstacles. The entire control proposition rests on three dedicated controllers: attraction controller (Eq. 3.3), repulsion controller (Eq. 3.4), and obstacle-avoidance controller (Eq. 3.21). To avoid challenging barriers, the system can switch its formation while maintaining inter-agent cohesiveness during the mission. Moreover, the stability of the proposed controllers has also been studied in the sense of Lyapunov.

2. As the evolutionary procedures typically consume a large amount of time to compute, hence, with a scalable swarm, it is very challenging to complete the mission in a stipulated time. So, in the next work (as detailed in Chapter 4), to overcome this limitation, an elliptical virtual-region-based shape control scheme (Eqs. 4.31-4.35) has been introduced for path planning of a swarm of robots. To alter the inter-agent formation among the agents in the group (based on the upcoming environmental conditions), a spanning-tree assisted shape matching algorithm has been proposed. To measure the robustness of the system, we have also revealed that the control scheme can handle the situation well in the face of agent failure by making the system fault-tolerant. Depending on the consequences, we can claim that the proposed control scheme confirms the robustness, scalability, as well as flexibility of the swarm arrangement.

3. Despite the optimal characteristics of the previous approach, the inter-agent cohesiveness deteriorates significantly in constricted environments or narrow pathways where the shape of the virtual ellipse is further expanded along the major axis while the minor axis is shrunken and may culminate into a line-formation in an extreme scenario. To overcome such limitations, in our next work (as discussed in Chapter 5), the elliptical regions have been replaced by arbitrarily shaped polygonal virtual boundaries (Eqs. 5.5-5.8). A novel circle packing strategy has been proposed for accommodating the agents inside the virtual boundary. As a consequence, more agents could be placed inside the newly fitted virtual region, thus enhancing the inter-agent cohesion in narrow corridors. The proposed framework is scalable in terms of an increasing number of robotic agents. This approach produces robustness, scalability, flexibility as well as strong inter-agent cohesiveness during the navigation.

4. Finally, we have addressed the navigational issue of a swarm of robots in multi-path/multi-target scenarios (as illustrated in Chapter 6). The elliptical region-based shape control technique and the traditional leader-follower scheme have been fused to split or merge the swarm. During navigation, if the leader robot detects multiple pathways to proceed further, the parent swarm will be broken down into several sub-swarms to perform the splitting phase. Nonetheless, it might so happen that after a while, two or more paths would merge and create a single common passage to progress further. In such a situation, if the obstacles do not create any hindrance, the corresponding sub-swarms (operating in those passages) will be routed through those converging paths in order to merge or rejoin. This approach also ensures robustness, scalability, flexibility, and strong inter-agent cohesiveness during the mission.

5. All of the above-mentioned approaches are simulated in software settings with scalable agents in the team. Additionally, real-time experiments (as presented in Chapter 7) have been carried out to validate the efficacies of the proposed schemes. In this regard, raspberry-pi-controlled two-wheeled mobile robots (locally fabricated), namely 'SonPI' having Wi-Fi (802.11) communication protocol and n array of five-sonar sensors, have been employed. The hardware experimental outcomes illustrate the viability of the proposed control actions even in a dynamic circumstance.

1.5 Organization of the Thesis

The objective of the present work is to create an adaptive cohesive swarm for an unknown occluded environment that can autonomously change its inter-agent formation in response to the dynamic environmental condition while approaching the target. Moreover, in the case of a multi-path or multi-target scenario, the swarm can split into sub-swarms and then rejoin if needed. The remaining portain of the thesis is organized as follows.

- In Chapter 2, we have presented the literature surveys to solve the path planning and formation control problems of a swarm robotic system. The major approaches which are generally used to solve the formation control problems of a swarm robotic system, such as the Behavioral-based approach, Leader-follower strategy, Swarm intelligence scheme, Artificial potential field approach, and Virtual structural approach are surveyed thoroughly by mentioning the pros and cons of each approach.

- In Chapter 3, we have proposed evolutionary algorithm-based hierarchical control strategies for navigating a swarm to a predefined target location while avoiding static and dynamic obstacles. Three dedicated controllers (attraction controller, repulsion controller, and obstacle-avoidance controller) have been conceived. The stability of the controllers has also been investigated in the sense of Lyapunov. The bio-inspired bat algorithm has been employed for switching the inter-agent formation during the navigational steps to avoid obstacles. Then, several well-known evolutionary schemes have been utilized to examine the performance of the proposed approach. Finally, the simulation results and comparison analysis have been provided to describe the feasibilities of the proposed approach.

- In Chapter 4, we have introduced a generalized adaptive control scheme that exploits an elliptical region-based virtual shape control technique for avoiding obstacles in an unknown occluded environment. A spanning tree-assisted shape matching algorithm has been proposed to switch the inter-agent formation of a swarm robotic system even in a dynamic situation. The fault tolerance capability, inter-agent cohesiveness, and scalability issue of the proposed scheme have also been studied in this chapter. The simulation results and comparison analysis describe the efficacies of the proposed approach.

- In Chapter 5, we have extended the elliptical region-based scheme for an arbitrarily shaped polygonal region. The Elliptical Fourier Analysis (EFA) has been employed to etch the arbitrarily shaped region. To accommodate all the agents inside the virtual polygonal boundary, the circle packing strategy has been utilized. This proposed approach improves the inter-agent cohesiveness of the system. The simulation results and comparison analysis describe the viabilities of the proposed scheme.

- In Chapter 6, we have introduced the split-join aspect of the swarm robotic system under the assumption that the unknown environment contains several pathways through which the group should travel during the navigational process to approach multiple targets. So, for successful and complete exploration of the environment, the patent-swarm of robots requires to be split into several sub-swarms. Moreover, when two sub-swarms detect an identical pathway to approach further, in this case, the sub-swarms will rejoin to form a super-swarm, and then together as a group will progress. The traditional leader-follower approach has been integrated with the region-based shape control scheme for achieving the above-mentioned objectives. The simulation results and comparison analysis describe the effectiveness of the proposed strategy.

- In Chapter 7, we have presented the hardware implementations of all the proposed techniques in a detailed manner. Initially, the architectural framework of a single robotic framework called 'SonPI' has been described. After that, the outcomes of the experiments of each of the proposed approaches, for example, evolutionary-based, virtual elliptical-region based, arbitrarily shaped polygonal-region based, and finally, the split-rejoin schemes are thoroughly presented. The detailed results of the hardware experiments and the comparison analyses describe the significance of the proposed strategies.
- Chapter 8 summarizes the work of this thesis. It also points out the limitations of the present work that could be addressed in the future.

1.6 Summary

In this chapter, we have summarized the evolution of swarm robotics in light of the unitary robotic framework. Several characteristics of a swarm robotic system have been thoroughly addressed, including scalability, robustness, flexibility, time efficiency, and so on. Furthermore, the issues in multi-robot systems, such as centralized vs. decentralized schemes, formation control, task assignment, inter-agent communication, etc. have been discussed. The potential application areas, such as disaster management, underwater area exploration, unknown area surveillance, and so on where a swarm of robots could be efficaciously used, have been meticulously discussed. Finally, the contributions and organization of this thesis are presented.

Chapter 2

Literature Survey

2.1 Introduction

The coordination strategy among a swarm of robots plays a crucial role in order to plan the trajectories of all the agents. So, the multi-objective cooperative control framework [95], [96] has been the subject of extensive research in recent decades. To conceptualize the movement of a robotic group, scientists from various fields are cooperating to formulate necessary arrangements such that the swarm must be competent enough to adapt the best way of motion pattern, especially, when troublesome impediments including dynamic obstacles show up in the environment [97], [98]. Therefore, it is necessary to establish a control methodology among the individual members in the group to illustrate how a team of homogeneous robots go through a territory without crashing into hindrances or other robots along their paths while maintaining desired inter-agent formation as well.

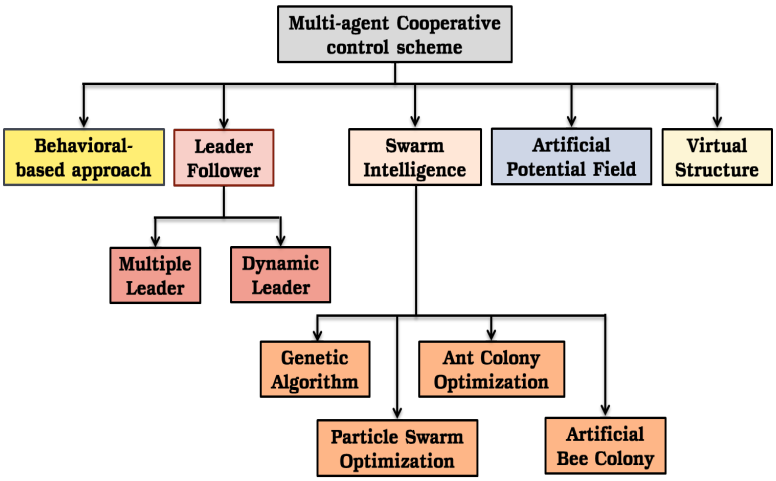


Fig. 2.1 State of the art techniques of cooperative control framework.

There exist some multi-robot cooperative control strategies in the literature as shown in Fig. 2.1. The behavioral-based approach, leader-follower scheme, swarm intelligence architecture, artificial

potential field, and virtual structural-based method are now being utilized to solve the multi-robot motion planning problem. We mainly discuss the state of the art of those control approaches (as shown in Fig. 2.1) in the remainder of this chapter.

2.2 Behavioral-based approach

The first and foremost technique that has been introduced to control a multi-robot system is the behavioral-based robotics [99], [100]. In this approach, [99], a set of predefined behavioral modules such as keeping a specific formation, moving to a target, avoiding obstacles, etc. are implemented on each robot, and the global behavior of the entire system is reliant on the combined behavior of each agent in the group. So, a robotic group can perform a specific set of operations at a given instant of time. To make the process more intelligent, in [101], [47], several sub-behaviors are implemented on an agent, which is triggered sequentially based on the information as detected from the surrounding environment. Moreover, the intensity of a behavior can be altered by adjusting some parameters during the navigation [100].

This method is specifically suitable when a particular task is expected to perform by a multi-robot system. The process is neither scalable in terms of the number of agents in the team nor time saving to accomplish a specific task [53], [38]. Moreover, failure of an agent can create a significant effect on the overall performance of the system; thus, the behavioral-based technique does not have any fault tolerance capability [30], and in complex circumstances, it is difficult to guarantee the stability of the overall system. Because of these reasons, this modality is generally not used in swarm robotics navigational problems.

2.3 Leader-follower strategy

The second most relevant technology that has been employed to control a robotic swarm is the leader-follower strategy [102], [103], [95]. In this approach [102], a particular robot has been selected as the leader agent that will move along a predefined trajectory inside the environment, while the other agents, called followers, will accompany the lead agent by keeping a certain distance among the other agents in the group. Therefore, by commanding the leader robot only, the entire system can eventually be managed, so this methodology is very simple to design and scalable in terms of the number of agents in the group [95]. This approach is specifically reliant on the leader robot for accomplishing an objective, so, the failure of that particular robot causes the entire mission to fail. Hence, such a technique is fault-intolerant [30]. Furthermore, while navigating across the environment, the feedback from the followers is not utilized by the lead robot hence the inter-agent cohesiveness becomes disjoint, and any follower robot can be lost inside the environment if it is not able to track the motion of the leader precisely [104].

To overcome the above-mentioned challenges, it is necessary to build an updated control rule for making the system fault-tolerant. In this aspect, the following methodologies have already been proposed over the last decade. Those techniques which are the extended version of the traditional leader-follower mechanism [104], [105], are described below.

2.3.1 Multiple Leaders approach

In this approach, multiple leaders are considered over the unitary leader framework [104], [106], [107]. Hence, the failure of any leader robot would not create any impact on the performance of the entire system [107]. In this scenario, the inter-agent cohesiveness of the entire group is solely dependent on the number of leaders based on the control command provided to the leaders [108]. The distribution of few leaders limits the shape of the group, whereas the control complexity will be increased significantly with too many leaders. Moreover, the distribution of tasks among the leaders also creates a major issue [108], [109] as the dynamics of the system become complex with more leader agents.

2.3.2 Dynamic Leader approach

To get rid of the above-mentioned situation and make the system robust as well as simple, the concept of a dynamic leader has been introduced in [110], [111], [105]. In this approach [110], instead of multiple leaders, it is presumed that any of the followers can be appointed as the lead agent during the navigational procedure. Therefore, during the exploration steps, if the currently acted leader robot is about to fail, then it can select any arbitrary agent (among the followers) as the lead and hand over the navigational plan with it [105]. From the next iteration onwards, the navigation would be performed by the newly elected leader robot to escort the swarm towards the target.

2.4 Swarm Intelligence

Next down the line, to build up a coordination strategy amongst a swarm of robots, the swarm intelligence (SI) technique has been presented [112], [113]. The mentioned approach is completely decentralized over all the agents in the framework [112], [2]. Inspired by the group foraging features of the social insects, the SI scheme has evolved. The main hypothesis that has been utilized here is called the *self-organization* principle [112], [114]. A group of self-organizing robots looks like swarms as they travel across the environment, where every robot updates its position such that it could maintain a protected separation from its neighboring agents [115]. The swarms likewise change their formation to suit the need for variations in the operating environment [116]. This criterion would allow the swarm to fit into narrow passages and to go around upsetting obstacles. To create the self-organization among a robotic swarm, the bio-inspired strategies [114], [117], [118] are usually employed. These methodologies imitate fruitful groupings of creatures in nature, for example, flocking

of birds [21], foraging of bugs [119], schooling of fish [22], and swarming of honey bees [20]. In this aspect, inspired by the flocking behavior of birds, the pioneering work has been proposed in [120]. This work has been emerged based on three basic principles such as *Separation*: to avoid local crowd, *Alignment*: to move towards the average heading of the neighboring mates, and *Cohesion*: to steer towards the average position of the local flock-mates.

Inspired from the above literature, several algorithms to create a coordination scheme among a swarm of robots on the ground of evolutionary techniques have been proposed, which are listed below.

2.4.1 Genetic Algorithm

The Genetic Algorithm (GA), introduced in [121], [122], is a metaheuristic optimization algorithm based on the mechanics of the natural adoption process. The basic principle behind this scheme is to mimic the notion of the survival of the fittest [123] to simulate the rules as witnessed in a natural system where the strong leads to adapt and survive while the weak tends to perish.

In the multi-robot path planning problem, GA is mainly applied to shorten the path length of each agent in the group [124]. In [125], the probabilistic road-map scheme has been fused with GA in order to study the efficiency of attaining a viable route. The non-dominated sorting genetic algorithm (NSGA-II) has been used in [126] for optimizing the path length of a multi-robot system. The above work has been extended in [127] to improve the path length, safety, and smoothness. Moreover, the adaptive schemes are modeled in [128], [129] to bypass the local trap, early convergence, and optimum path length for enhancing the Pareto-optimal solutions. In [130], GA has been applied to the directed acyclic graph for improving the computational efficiency hence, the mentioned process would generate multiple feasible minimum paths. An improved crossover mutation operator has been modeled in [131] for static environmental conditions. In [132], the potential field method is integrated with NSGA-II to enhance the smoothness of the path for a multi-robot system. Furthermore, a virtual structural approach has been introduced for an agent to approach the target safely while avoiding obstacles. Depending on the above state of arts, we can mention that for swarm robotic motion planning architecture, GA can optimize the path length; however, in presence of obstacles, using the traditional operators such as crossover, mutation, etc. the feasibility of the solution degrades.

2.4.2 Ant Colony Optimization

Ant Colony Optimization (ACO) is one of the well-known evolutionary techniques motivated by the foraging behaviors of ants [133], [134], [135]. There are four major components (such as ant, pheromone, daemon action, and decentralized control) that are employed to create the ACO algorithm for mimicking the exploration and exploitation of the search space. The pheromone is a chemical substance spread by ants over the route they travel, and its intensity changes over time due to evaporation [136]. To design the ACO algorithm, it is assumed that each of the ants drops pheromones while moving across the search space, and the amounts of pheromones indicate the

intensity of the trail that assists to determine the direction of the path of the succeeding ants. So, this activity is considered the global information of the system. To gather all of the global information for deciding the convergence criterion, the Daemon action is performed. Finally, the decentralized action is implemented to make the algorithm robust and flexible within a dynamic environment so that it offers optimum performance during the failure of ants. Thus, these components would result in a cooperative interaction that leads to the emergence of shortest paths [133], [134].

In [137], ACO with backtracking characteristics has been adopted for a robotic system to obtain the shortest path for approaching the goal with collision avoidance. For a dynamic scenario, the ACO algorithm has also been applied in [133]. In [138], [139], [140] updated rules with penalty functions are added in ACO in order to get the shortest path for a multi-robot framework. These specific schemes have been named the Shortest Path Ant Colony Optimization (SPACO). In [141], [142], [143] ACO algorithm has been applied for a multi-robot system that is operating in an unknown environment with static obstacles in different orientations. In those works, the environments are considered as a grid map with an identical number of rows and columns. Based on the sensory information of every robot, the map has been updated in each time step.

2.4.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an efficient optimization technique introduced in [144]. It adopts a simple mechanism to imitate the swarming behavior of birds [21] and fish [22] by managing the particles to search for the global optimal solutions [144], [145]. This specific evolutionary algorithm is very efficient to solve multi-dimensional problems, thus, it has immense applications in the path planning of a swarm of robots. Moreover, the PSO algorithm does not use the gradient of the problem being optimized, so it does not require the optimization problem to be differential [146]. Therefore, stochastic problems can also be solved efficiently using PSO [146], [147].

In swarm robotics motion planning problems, the PSO algorithm has been applied very frequently than the other well-known meta-heuristic techniques. In [148], inspired by the flocking behavior of birds, the fuzzy logic has been integrated with PSO for navigating a swarm of robots in a static environmental condition. To obtain the shortest path of a robot in dynamic circumstances, PSO has been employed in [97] including the adaptive inertia weight rule. Moreover, in [149], a probabilistic roadmap (PRM) has been fused with PSO to get a time-efficient solution in terms of path length. In this work, PSO does global planning, whereas, local planning is performed by PRM. The efficiency of this technique has also been compared with the negative PSO (NPSO) and PRM. In [150], the artificial potential field (APF) scheme has been merged with PSO. In this approach, the primary drawback occurs due to the lack of heuristic information; the proposed algorithm easily falls into the local extremal. To circumvent this issue, the GA has been fused with PSO in [151] such that the crossover and mutation mechanisms of GA maintain the diversity of the particles in the iterative process, and hence, the particles will converge to the global optimum point. In [152], an improved linear PSO algorithm with adaptive acceleration factors has been introduced to improve the convergence timing

of the algorithm. The above work has been extended further for hybrid PSO with GA in [153]. To enhance the accuracy of path planning of a multi-robot architecture, a multi-objective optimization strategy has been employed in [154], where, to modify the search functionality, standard PSO is applied, then, the chicken swarm algorithm is introduced to eliminate the stagnant particles. Although these improvements enhance the planning performance relative to the standard PSO to a certain extent, however, this scheme does not solve the problems of path planning. Moreover, the algorithmic complexity and the local convergence problem caused by the lack of heuristic information remain. To solve this specific problem of the PSO algorithm, owing to the lack of heuristic information, the authors propose a heuristic elastic PSO path planning algorithm in [155], [98].

2.4.4 Artificial Bee Colony

Artificial Bee Colony (ABC) is one of the most recent swarm intelligence algorithms. It is introduced in [156], and while analyzing the performance of ABC [157], it is witnessed that it performs better than several other meta-heuristics approaches. This algorithm is proposed based on the intelligent behavior of the honey bees in searching for food sources, known as nectar, and the sharing of information about that food source among other bees in the nest. This algorithm is claimed to be as simple and easy to implement as PSO [156], [158].

In [159], the authors offer a distributed control rule for a multi-robot system based on ABC while expressing it as an optimization problem. The proposed solution has been applied to solve transportation problems for a swarm of robots (30 robots). Moreover, a comparative analysis has been performed to show the effectiveness of the proposed scheme. To solve the navigational problem of swarm robots in a multi-target unknown environment, a robust procedure based on the ABC strategy has been presented in [160]. The mentioned approach has two steps. First, depending on the sensory information, the intermediate path connects the starting point and current point of the mobile robot via several common points. The second step is the optimization step in which the ABC-based evolutionary algorithm has been used to smoothen the intermediate path such that the resultant path is collision-free and shorter. Numerous benchmark maps (having different scenarios) are employed to verify the viability of the proposed approach. Finally, in [161] a unique feature of the ABC algorithm, namely the Arrhenius ABC algorithm (aABC) has been introduced for solving robot path planning problems. This idea originates from the exclusive concept given by Arrhenius. Based on the outcomes, it is observed that the exploration capability of the ABC algorithm is improved significantly, thus, it can be concluded that aABC is a better choice than ABC for solving complex optimization problems like robotics path planning problems. Furthermore, the proposed method can also be applied to different categories of problems.

2.4.5 Other notable techniques

There are so many other evolutionary algorithms available that can also be used or have already been used for solving the multi-robot motion planning problems such as Bacterial Foraging Optimization Algorithm (BG) [162], Differential Evaluation (DE) [163], Firefly Algorithm (FA) [118], Bat Algorithm (BAT) [113], etc. However, those methods are not discussed thoroughly in this thesis because the purpose of this section is to introduce and discuss the well-known and commonly used SI-based approaches that are mainly employed to solve the swarm robotic motion planning problem till date.

2.5 Artificial Potential Field

In multi-robot path planning problems, the method that has been commonly used to attain a collision-free navigational path towards the goal in an unknown occluded environment is called the Artificial Potential Field approach (APF) [51], [52], [164]. In this scheme, the environment has been treated as the electrostatic potential field with the robot configured as a point. The process combines an attraction force towards the target and a repulsive force from the obstacles and the neighboring agents. So under the joint influence of those control forces, a collision-free path is produced towards the target in a time-efficient manner; thus, in several well-cited works, this methodology is normally utilized [164], [96], [165].

In [166], [167], the notion of APF has been fused with the consensus protocol to address the navigational problem for a multi-robot system under a variety of assumptions on the network topology with directed or undirected information flow. The proposed approach produces robust output in the face of an obstructed environment with static obstacles. In [51], [52], the most powerful studies based on the principle of APF have been introduced. In those works, the agent-to-agent interactions are assumed to be the 'attraction-repulsion' type; so, each agent seeks to be in a position that is comfortable relative to its neighboring agents. Attraction indicates the grouping and cohesiveness criteria for a robotic swarm. On the other hand, to avoid static obstacles and inter-agent collisions, the repulsion mechanism has been formulated. Moreover, the stability of the proposed control approach has also been analyzed. In [164], the authors have integrated the leader-follower scheme with APF to avoid the collision of multiple unmanned aerial vehicles (UAVs) in 3-D space. One agent has been selected as the virtual leader that will fly through the desired trajectory using APF, whereas, the followers would track the leader for maintaining the formation flight. For avoiding inter-agent collision among the UAVs, a repulsion action has been proposed. Finally, the simulation results have been provided to analyze the efficacies of the proposed scheme. Recently, one work has been published in [96], where the authors have discussed a novel decentralized controller inspired by APF for aggregation of a robotic swarm in the absence of communications. Employing the control strategy, they have proved that the stable aggregation among a robotic swarm is possible to achieve using only a local sensing scheme with no inter-agent communication. Furthermore, the APF methodology has also been employed to achieve the swarm split control of a multi-agent system [165]. The simplistic

attraction-repulsion controller schemes have been proposed so that based on the user input, a patent swarm can split into several clustered sub-swarms along the direction perpendicular to the common heading direction of agents.

Based on the above discussions, the main advantages of the APF approach are listed below:

1. Employing attractive and repulsive control actions, a swarm of robots can be controlled in a decentralized manner for the static environmental condition.
2. This specific method ensures scalability in terms of the number of agents in the swarm.
3. Formulating a suitable attraction-repulsion control action, it is possible to achieve a cohesive swarm operating across an unknown environment.

2.6 Virtual Structural approach

Finally, the virtual structural (VS) approach has been discussed for driving a swarm of robots in an occluded environment while maintaining a specific inter-agent formation. In this approach [168], [169], the desired pattern that needs to be followed by the system is considered as a single entity, and the desired motion is assigned to the respective structure. Hence, all the agents in the team will maintain a rigid formation as a geometric relationship exists between the locations of the agents with respect to the virtual structure [170]. Therefore, in the face of challenging obstacles, it is impossible to alter the current formation, and this specific approach is not even scalable for the number of agents in the team. In this aspect, another method has been proposed to control a robotic swarm for building a formation by applying the constrained optimization [48]. This approach has also encountered similar problems as that of the virtual structural technique as the complexity of the constrained relationships increases with an increasing number of agents; thus, this method is not suitable for controlling a large number of robots.

To get rid of those situations, recently, a region-based shape control scheme has been proposed in [171], [172]. In this approach, each robot in the group stays within a predefined region while maintaining a certain distance from each other. The defined region can be specified as various shapes; hence, depending on the desired shape, various kinds of inter-agent formations could be formed. The robots in the group are only required to communicate with their neighbors and not the entire population. The system is scalable so that any robot can attach or leave the formation without affecting the other robots. In [173], the authors extend the above idea to a group of homogeneous mobile robots moving into the desired region. A multilevel structural approach has been employed to present different shapes of swarm robots in the desired region. The APF technique has been combined to control the robots in forming the desired formation shape. Finally, simulations and experiments are performed to demonstrate the effectiveness of the proposed approach. In [174], the flocking behavior is combined with the shape control scheme to achieve the regional shape formation of a multi-agent system. It has been shown that all the agents in the team can acquire the same speed while moving

into the desired region. The stability of the proposed control action has also been studied. In [175], the multi-objective region reaching the control scheme for a multi-agent system has been presented. Two distributed control algorithms are formulated for the cases of static and moving target regions. The proposed control scheme is scalable, and the convergence criteria are proved to be stable in the sense of Lyapunov.

To compare the performances of all the above-mentioned literature holistically, we list our observations below.

1. In behavioral-based approaches [99]-[47], it is difficult to analyze the entire system analytically to realize insights into the control problems. Therefore, the convergence criterion cannot be measured here; thus, maintaining formation among the agents in the group is not possible. So, this approach is usually not used in multi-robot path planning problems.

2. In the Leader-Follower schemes [102], [105], the function of the followers is to track and maintain the desired distance to their respective leaders; so, the inter-agent formation becomes rigid. Creating rigid formation in a dynamic environment is often undesirable to fulfill the mission objectives. Moreover, the stated approach is fault-intolerance [101], i.e., the failure of leaders will lead to the failure of the entire mission.

3. In swarm intelligence procedures [112], [113], all the agents in the swarm will self-organize themselves based on the upcoming environmental situation. The performance of this process is observed to be very robust for a dynamic environment. However, to achieve self-organization among the robots generally, evolutionary methods are employed. To execute such algorithms in a real-robotic platform, it is difficult to achieve the desired performance, as such techniques consume a large amount of time before concluding. So, this methodology is only limited to software simulation.

4. The artificial potential approaches [51], [167] are produced robust output in the swarm robotics path planning problems. However, the major problem is that a robot can easily be trapped at the local minimum point before reaching the target. The stated approach is solely dependent on the formulation of the attraction and repulsion control actions; hence, creating the desired formation and altering it based on the upcoming environmental situation; are difficult to achieve.

5. As mentioned previously, the virtual-structural approaches [168], [170] are not scalable in terms of the number of agents in the team. Furthermore, with enhancing agents in the swarm, the complexity of the problem will increase significantly sometimes; it would be challenging to converge to a specific solution.

6. The region-based shape control scheme [171], [172] is found to be robust to analyze the performance in all of the above respects. However, till today, it has not been applied for a dynamic situation, where, the estimated area should be adaptive enough so that all the agents could change their position within the stipulated region for avoiding obstacles. This specific

criterion is very essential for the fruitful operation of a swarm robotic framework in static and dynamics environmental conditions.

2.7 Summary

In this chapter, we have summarized the literature surveys for solving the multi-robot navigational problem. Initially, the behavioral-based approach has been articulated, along with its merits and demerits. Then, the traditional leader-follower scheme has been discussed. To make a leader-follower system to be fault-tolerant, the multi-leader approach and the dynamic leader strategy have also been discussed. After that, the swarm intelligence technique, as well as, the important evolutionary methodologies to solve the multi-robot navigational problem have been addressed. Then, we have discussed the literature survey of the artificial potential field-based scheme. Finally, the virtual structural method, the most recent technique to solve the navigational problem, has been mentioned.

Chapter 3

Evolutionary-algorithm based motion planning of a swarm of robots

3.1 Introduction

The motion planning [163] of a robot is a computational problem. It is necessary to determine the sequence of movements of the robot from the source location to the destination such that there would not be any collision with the nearby barriers. For a unitary robotic platform, such kind of planning can be accomplished by a dedicated human operator [43]. However, in the context of a swarm robotic path planning problem, employing a human operator is not feasible as a human operator cannot plan the trajectories of each robotic agent in the swarm at a single point of time [48]. In the case of a swarm, if the number of agents is pretty less, a non-automated solution could be thought of. But gripping manual control over a perfect swarm with a sufficiently large number of agents is simply impossible. Moreover, manual control has its limitations, like quite a localized range of operation. In safety-critical situations, the role of a human operator gets severely delimited, and hence the implementation of manual control is notionally rejected. In spite of all these reasoning, the bottom line of robotic research is to create an automated system with either a single robot or a swarm of them and to minimize human intervention. So, to design a swarm robotic framework, in particular, an automated control solution (for each agent) would be required to lead the entire system from the source to the destination.

In this chapter, we have mainly addressed the issues of automated path planning and obstacle avoidance for a swarm of robots working in an occluded 2D environment. The main objective of the work presented in this chapter is to ensure that all the robotic units would be able to approach a predefined target autonomously based on the sensing information of each agent. Further, it has been conceived that the agents would be significantly cohesive with each other to form a well-defined swarm. During the navigation, how the behavior of the swarm has to be altered to avoid challenging dynamic barriers is primarily taken into account in this study. First and foremost, to plan such a path

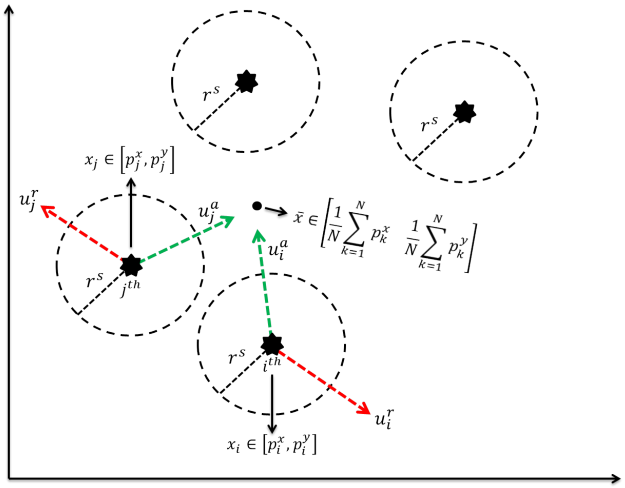


Fig. 3.1 The attraction and repulsion control actions acting on i^{th} and j^{th} agents.

for the swarm from the start position to the endpoint, suitable controllers need to be devised. Second, to achieve such an efficacious path planning mechanism, in a constrained situation, it is found that tuning certain gains of the controllers required, in this situation, is utterly necessitated during the navigational process.

With this introduction, in this study, we attempt to address the above-mentioned issues and present the controllers that are synthesized based on the sensory information gathered from the nearby obstacles. Evolutionary techniques have been employed here to tune those controllers' gains.

3.2 The Backdrop

Consider a group of N identical robotic agents is operating in $2D$-plane (as shown in Fig. 3.1). The motion dynamics of the i^{th} agent can be expressed as [51], [52]

$$\begin{aligned} \dot{x_i} &= v_i \\ \dot{v_i} &= u_i \end{aligned} \tag{3.1}$$

where $x_i = \begin{bmatrix} p_i^x & p_i^y \end{bmatrix} \in \Re^2$ is the position, $v_i = \begin{bmatrix} v_i^x & v_i^y \end{bmatrix} \in \Re^2$ is the velocity and $u_i = \begin{bmatrix} u_i^x & u_i^y \end{bmatrix} \in \Re^2$ is the control input of the i^{th} agent. It is assumed that an agent can sense the position and velocity of its neighboring agents (N_i). Under this specific assumption, the agent-to-agent interactions are chosen to be *attraction-repulsion* type [51], [164] that refers to the fact that each agent seeks a comfortable position in the swarm relative to its neighbors [142]. Thus, the control input (u_i) for the i^{th} agent would be defined as

$$u_i = -u_i^a + u_i^r \tag{3.2}$$

where u_i^a and u_i^r represent the attraction and repulsion forces acting on the i^{th} agent that would be generated from the dedicated attraction and repulsion control actions. Fig. 3.1 represents one such instance with i^{th} and j^{th} agents (positioned at $x_i = (p_i^x, p_i^y)$, and $x_j = (p_j^x, p_j^y)$ respectively) of the swarm. The directions of the attraction and repulsion control forces are also presented in the respective figure.

Now, the question we pose that how the agents will create a cohesive swarm and approach the target while preserving the strong inter-agent bonding in a cluttered environment (full of static and moving obstacles) employing the attraction-repulsion control actions (vide Eq. (3.2))? In this study, strong cohesion [52] refers to the closeness amongst the group members of the swarm.

So, in order to ensure a cohesive swarm, the agents are required to create an agent-interconnected formation. However, in the presence of obstacles (assuming the distance between two obstacles does not allow the swarm to maintain the formation); the system has to alter its swarm shape to pass through those obstacles. All these concerns could be mitigated by designing attraction and repulsion controllers. In the following subsection, therefore, we discuss the detailed control mechanism that would be required for the group of agents to evolve as a perfect swarm.

3.3 Control mechanism for the swarm robotic system

The proposed control framework contains three hierarchical control schemes, such as attraction control, repulsion control, and obstacle avoidance control. The function of the attraction controller is to create a cohesive swarm, whereas, the repulsion control will actuate to avoid the inter-agent collision. And the obstacle avoidance controller will ensure to bypass the nearby obstacles.

3.3.1 Attraction controller

As the name suggests, the task of the attraction controller is to generate necessary force amongst the members of the group (force acting between a member and its neighboring member) so that all the members of the swarm may remain enclosed in a planar region, thus creating a well-shaped swarm. In other words, the main purpose of this controller is to achieve grouping of all the agents and to guarantee cohesion [166], [52] among all the agents in the team.

The main features of the attraction controller [51], [52] are to 1) steer an agent towards the average heading, and 2) move towards the average position of the swarm. So, under the actuation of this controller, an agent would approach the average position of the swarm while maintaining an identical heading with respect to its neighboring agents. Therefore, the generalized form of the force produced by the attraction controller (u_i^a) for the i^{th} agent to the average position ($\bar{x}$) of the swarm and the average heading ($\bar{v}$) of the neighboring agent can be represented as [167]

$$u_i^a = k_i^a(x_i - \bar{x}) + k_i^v(v_i - \bar{v}) \tag{3.3}$$

where, $k_i^a > 0$ and $k_i^v > 0$ are the strength of attraction, $\bar{x} = \frac{1}{N}\sum_{k=1}^{N} x_k$ and $\bar{v} = \frac{1}{N_i}\sum_{k=1}^{N_i} v_k$ are the average position and the average heading of the group respectively. Hence, depending on the control action (u_i^a), the force of the i^{th} agent will be directed towards $(x_i - \bar{x})$ while maintaining similar orientation towards the average heading ($\bar{v}$) of its neighboring agents N_i. Thus, for the N number of agents, an evolving behavior of the group would be created where all the agents attempt to dynamically attain aggregate position and velocity and finally would manage to get close to each other and eventually move in the same direction with nearly the same velocity.

However, the sole activation of this controller may result in an inter-agent collision. Hence, to alleviate this problem, a repulsion controller is to be devised. It is needless to say that the repulsion controller should act in tandem with the attraction controller.

3.3.2 Repulsion controller

Theoretically, the role of the attraction controller is to force the agents to be as close as possible so as to create and form a swarm. However, it might so happen that due to the attraction force the distance between two agents at any instant would become zero. So the immediate effect would be a collision between the agents and in the worst case that might lead to agent breakdown thereby compromising the strength of the swarm and intuitively, we can claim that the strength is empirically proportional to the number of active agents at a point of time. Hence a repulsion controller is conceived.

A repulsion controller produces a repulsive force between two agents so that they can repel each other (despite the presence of the force of attraction working on the agents concerned). So the forces generated by the two controllers would act in the opposite directions and our aim is to strike a balance so that the agents would remain close to each other and at the same time they would be allowed to maintain a predefined minimum distance between them. Thus a repulsion controller would not disturb a swarm formation. Instead, the controller would judiciously increase the area of the swarm and negate the possibility of agent collision. Because of the repulsion controller, each member of the group will enjoy a region around them where they can execute some unconstrained motion with two degrees of freedom.

We reiterate that the function of the repulsion controller is to avoid the inter-agent collision and maintain the safest distance from the neighboring agents [51], [52]. So, the force exerted by the repulsion control action is to increase the distance between two agents such that the inter-agent collision could be avoided (Fig. 3.1). Based on this principle, the repulsion control (u_i^r) acting on the i^{th} agent with respect to the j^{th} neighbor can be represented as

$$u_i^r = k_i^r \times \exp\left(-\frac{x_{ij}^2}{\delta \times (r_i^s)^2}\right) \times (x_i - x_j) \tag{3.4}$$

where $k_i^r > 0$ is the magnitude of the repulsion force, r_i^s is the region of repulsion around the i^{th} agent, and $x_{ij} = ||x_i - x_j||$ is the is the Euclidean distance between the i^{th} and j^{th} agents respectively as

shown in Fig. 3.1. Thus, under the activation of the repulsion controller, the i^{th} agent will be directed along $(x_i - x_j)$. Here, the parameter δ defines a unique distance to trigger the repulsion action as shown in Fig. 3.2. When, $(\delta \times (r_i^s)^2)$ is less than x_{ij}^2 (as shown in Fig. 3.2(a)), the repulsion regions

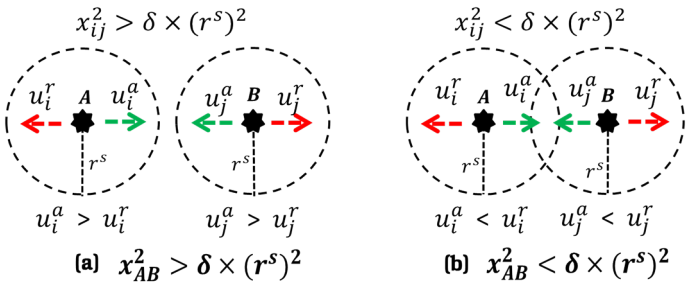


Fig. 3.2 Force distribution of the attraction-repulsion controllers based on the term $(\delta \times (r_i^s)^2)$.

of the i^{th} and j^{th} agents will not overlap, hence, the attraction force will actuate to form a cohesive swarm. So, here, the magnitude of the repulsion forces of both the agent will be less as compared to the attraction forces. Thus, $u_i^a > u_i^r$, and $u_j^a > u_j^r$. However, in the contrary condition, when $(\delta \times (r_i^s)^2)$ is greater than x_{ij}^2 (Fig. 3.2(b)), the repulsion regions of the i^{th} and j^{th} agents will overlap. As a consequence, there is a high possibility of collision between the agents. So, in this kind of situation, the magnitudes of the repulsion forces are required to be greater than the attraction forces so that the distance between those agents could be increased. Hence, $u_i^a < u_i^r$, and $u_j^a < u_j^r$. And, the respective forces are required to be activated in the opposite directions of both the agents in order to maintain a safe distance.

Therefore, depending on the joint activation of the attraction-repulsion control actions, the i^{th} and j^{th} agents of the swarm will approach the average position with identical heading while maintaining a safe distance between each other. The following theorem establishes the stability of the controllers.

Theorem 1: In light of the parameter δ, there exists a force balancing condition, where, $u_i^a = u_i^r$. Employing this principle, we need to find the limiting value δ, so that the above-mentioned control laws become stable. The corresponding analytical proof has been given below.

proof: Combining Eqs. 3.3 and 3.4, we get

$$u_i = -k_i^a e_{pi} - k_i^v e_{vi} + k_i^r \sum_{j \in N_i} \exp\left(-\frac{x_{ij}^2}{\delta \times (r_i^s)^2}\right) \times (x_i - x_j) \tag{3.5}$$

where, $e_{pi}(= x_i - \bar{x})$ and $e_{vi}(= v_i - \bar{v})$ are the attraction error of the i^{th} agent from the average position and heading of the group respectively.

Now, let us assume that the error dynamics [52] of the swarm system is defined as $E = [E_1^T \quad E_2^T \quad \ldots \quad E_N^T]^T$; where $E_i = \begin{bmatrix} e_{pi} & e_{vi} \end{bmatrix}$. Therefore, from Eq. 3.5, the error dynamics becomes

$$\begin{aligned} \dot{e}_{pi} &= e_{vi} \\ \dot{e}_{vi} &= \dot{v}_i - \dot{\bar{v}} \end{aligned} \tag{3.6}$$

where $\dot{\bar{v}} = \frac{1}{N}\sum_{l=1}^{N} u_l$. As all the agents are identical in nature, $k_i^a = k^a$, $k_i^v = k^v$, $k_i^r = k^r$ and $r_i^s = r^s$ for all $i \in N$. Therefore, $\dot{e}_{vi}$ becomes

$$\begin{aligned}\dot{e}_{vi} = u_i - \frac{1}{N}\sum_{l=1}^{N} u_l = -k^a e_{pi} - k^v e_{vi} + k^r \sum_{j\in N_i} \exp\left(-\frac{x_{ij}^2}{\delta \times r^{s2}}\right) \times (x_i - x_j) \\ + \frac{1}{N}\left[k^a \sum_{l=1}^{N} e_{pl} + k^v \sum_{l=1}^{N} e_{vl} - k^r \sum_{l=1}^{N}\left[\sum_{j\in N_i} \exp\left(-\frac{x_{lj}^2}{\delta \times r^{s2}}\right) \times (x_l - x_j)\right]\right]\end{aligned} \tag{3.7}$$

From Eq. 3.7, it is observed that $\sum_{l=1}^{N} e_{pl} = \sum_{l=1}^{N}(x_l - \bar{x}) = 0$, and $\sum_{l=1}^{N} e_{vl} = \sum_{l=1}^{N}(v_l - \bar{v}) = 0$. Now, let us assume that the last term of Eq. 3.7 is equal to G, so,

$$G = -\frac{k^r}{N}\sum_{l=1}^{N}\left[\sum_{j\in N_i} \exp\left(-\frac{x_{lj}^2}{\delta \times r^{s^2}}\right) \times (x_l - x_j)\right] \tag{3.8}$$

The generalized form of Eq. 3.8 can be written as (assuming all agents are neighbors, i.e. $N_i = \{1,2,...,N\}$),

$$G = -\frac{k^r}{N}\sum_{l=1}^{N}\left[\sum_{j=1}^{N} \exp\left(-\frac{||x_l - x_j||^2}{\delta \times r^{s^2}}\right) \times (x_l - x_j)\right] \tag{3.9}$$

Now, if we exand Eq. 3.9, we would get

$$\begin{aligned}G = -\frac{k^r}{N}[[&\exp(-\frac{||x_1 - x_2||^2}{\delta \times r^{s^2}})(x_1 - x_2) + \exp(-\frac{||x_1 - x_3||^2}{\delta \times r^{s^2}})(x_1 - x_3) \\ &+ ... + \exp(-\frac{||x_1 - x_N||^2}{\delta \times r^{s^2}})(x_1 - x_N)] \\ &+ [\exp(-\frac{||x_2 - x_1||^2}{\delta \times r^{s^2}})(x_2 - x_1) + \exp(-\frac{||x_2 - x_3||^2}{\delta \times r^{s^2}})(x_2 - x_3) \\ &+ ... + \exp(-\frac{||x_2 - x_N||^2}{\delta \times r^{s^2}})(x_2 - x_N)] \\ &+ ...+ \\ &+ [\exp(-\frac{||x_N - x_1||^2}{\delta \times r^{s^2}})(x_N - x_1) + \exp(-\frac{||x_N - x_2||^2}{\delta \times r^{s^2}})(x_N - x_2) \\ &+ ... + \exp(-\frac{||x_N - x_{N-1}||^2}{\delta \times r^{s^2}})(x_N - x_{N-1})]]\end{aligned} \tag{3.10}$$

From Eq. 3.10, it is seen that for each agent there will be an equal and opposite force exerted on it by its neighbors for stabilizing the entire system. Therefore, the final result of G should be equal to zero.

Hence, based on the above outcome, Eq. 3.7 could be modified as

$$\dot{e}_{vi} = -k^a e_{pi} - k^v e_{vi} + k^r \sum_{j \in N_i} \exp\left(-\frac{x_{ij}^2}{\delta \times r^{s2}}\right) \times (x_i - x_j) \tag{3.11}$$

Now, expanding the exponential term (while neglecting the higher order terms) of Eq. 3.11, we get

$$\dot{e}_{vi} = -k^a e_{pi} - k^v e_{vi} + k^r \sum_{j \in N_i} \left[1 - \left(\frac{x_{ij}^2}{\delta \times r^{s2}}\right)\right] \times (x_i - x_j) \tag{3.12}$$

Hence, the error dynamic equation can be evaluated as (from Eqs. 3.6, and 3.12)

$$\dot{E}_i^T = A_i \times E_i^T + B_i \left[\phi_i^r - \phi_i^\delta\right] \tag{3.13}$$

where $A_i = \begin{bmatrix} 0 & 1 \\ -k^a & -k^v \end{bmatrix}$, $B_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $\phi_i^r = k^r \sum_{j \in N_i}(x_i - x_j)$, and $\phi_i^\delta = k^r \sum_{j \in N_i} \left(\frac{x_{ij}^2}{\delta \times r^{s2}}\right)$ $(x_i - x_j)$.

To prove the stability of Eq. 3.13, we have considered the following (generalized) Lyapunov candidate [175] function such that the corresponding function is positive definite. Now, for the system to be stable (as per the Lyapunov stability theorem), the time-derivative of the candidate function is required to be negative definite.

$$V_i(E_i) = E_i^T P_i E_i \tag{3.14}$$

The time-derivative of Eq. 3.14 becomes

$$\dot{V}_i(E_i) = \dot{E}_i^T P_i E_i + E_i^T P_i \dot{E}_i \tag{3.15}$$

Inserting the values of $\dot{E}_i^T$, and $\dot{E}_i$ from Eq. 3.13 into Eq. 3.15, we get

$$\dot{V}_i(E_i) = -E_i^T Q_i E_i + 2E_i^T P_i B_i (\phi_i^r - \phi_i^\delta) \tag{3.16}$$

where, $Q_i = -(P_i A_i + A_i^T P_i)$. Therefore, considering the entire system, the Lyapunov candidate becomes

$$\dot{V}(E) = \sum_{i=1}^{N} \dot{V}_i(E_i) = \sum_{i=1}^{N} \left[-E_i^T Q_i E_i + 2E_i^T P_i B_i (\phi_i^r - \phi_i^\delta)\right] \tag{3.17}$$

In order to prove the stability in the sense of Lyapunov, $\dot{V}(E)$ should be negative semi-definite; hence the rightmost term of Eq. 3.17 needs to be less than or equal to zero which corresponds to $\phi_i^r \leq \phi_i^\delta$. Thus, from Eq. 3.13, we get

$$k^r \sum_{j \in N_i} (x_i - x_j) \leq k^r \sum_{j \in N_i} \left(\frac{x_{ij}^2}{\delta \times r^{s2}}\right) (x_i - x_j) \tag{3.18}$$

Solving Eq. 3.18, we get

$$x_{ij} \geq \sqrt{\delta} \times r^s \tag{3.19}$$

In a simplistic scenario (considering Fig. 3.2(a)), we can conclude that to balance the attraction and repulsion forces between the i^{th} and j^{th} agents, the minimum possible value of x_{ij} is required to be at least $2r^s$. Hence, to satisfy Eq. 3.19, the limiting value of $\sqrt{\delta}$ should be at least equal to 2 for achieving the force balance condition between the i^{th} and j^{th} agents. So, in this case, $x_{ij} \geq 2 \times r^s$. Under this specific consideration, the Lyapunov stability holds true and theorem 1 has been proved successfully.

In an obstacle-free environment, the swarm system will reach the destination under the influence of the attraction and repulsion control actions as described in the earlier section. However, we can intuitively place our arguments that in an obstructed environment, the path planning of the swarm would be heavily dependent on the presence of static or moving impediments. The swarm needs to detect those obstacles well in advance and the free passage as well through which it would be able to pass. Obstacle detection and path planning are dependent on many factors, like sensory information. Hence, it is judicious to devise a dedicated controller, namely an Obstacle Avoidance Controller which will act in an integrated manner with the attraction-repulsion controllers. In the following sub-section, we discuss the obstacle avoidance controller in detail.

3.3.3 Obstacle avoidance controller

For avoiding obstacles by an agent, we have devised an obstacle-avoidance control law such that the overall system becomes capable enough to bypass upcoming hindrances while maintaining inter-agent cohesiveness with its adjacent agents. The following assumption has been taken to accomplish this objective.

Assumption: Each agent of the team is equipped with an array of distance sensors (such as Sonar [125] or Lidar [176]). Hence, in the presence of obstructions, the relative distance of the nearby obstacles can be sensed (vide Fig. 3.3) if the obstacle is within the field-of-view (FOV) of any one of the sensors.

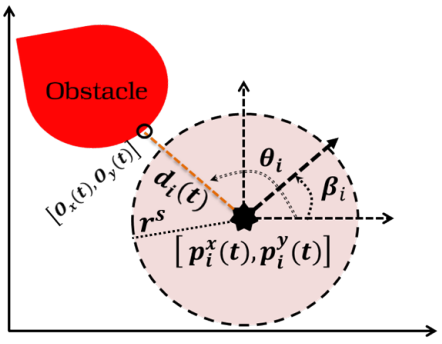


Fig. 3.3 Sensing and estimating the nearby obstacles $[O_x(t), O_y(t)]$.

Now, during the navigation of the i^{th} agent (located at $x_i(t) = [p_i^x(t), p_i^y(t)]$) at the t^{th} time, we assume that an obstacle is within the sensing FOV of that particular agent as shown in Fig. 3.3. Moreover, the sensed distance of the obstacle from the k^{th} sensor is $d_i(t)$. Hence, based on the distance information obtained from the k^{th} sensor, the approximate position of the obstacle ($O(t) = [O_x(t), O_y(t)]$) can be calculated as (Fig. 3.3)

$$\begin{bmatrix} O_x(t) \\ O_y(t) \end{bmatrix} = \begin{bmatrix} p_i^x(t) \\ p_i^y(t) \end{bmatrix} + d_i(t) \times \begin{bmatrix} \cos[\theta_i(t) + \beta_i(t)] \\ \sin[\theta_i(t) + \beta_i(t)] \end{bmatrix} \tag{3.20}$$

where, $\theta_i(t)$ and $\beta_i(t)$ are the orientations of the sensor and the i^{th} agent with respect to the world-coordinate-frame-of-reference.

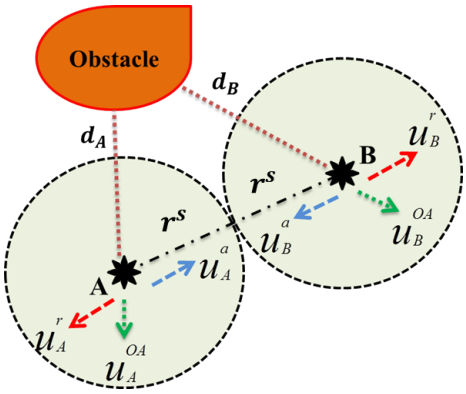


Fig. 3.4 Triggering of the obstacle-avoidance controller based on the sensed distance $d_i(t)$.

Now, if the sensed distance ($d_i(t)$) is greater than the repulsion range (r^s) for the i^{th} agent, the possibilities of collision of that agent with the corresponding obstacle would be less (Figs. 3.3, and 3.4). And hence, the associated control law employing the attraction-repulsion action (vide Eq. 3.5) can actuate the agent towards the target.

Now, if $d_i(t) \leq r^s$, then, the risk of collision with the obstacle is high, so, in order to avoid such situation, an obstacle avoidance control law (u_i^{OA}) is required to be triggered for the i^{th} agent. And, based on this control action, a generalized force along the direction of $(x_i(t) - O(t))$ will be produced as shown in Fig. 3.4. Thus, analytically, this specific control action can be defined as [51]

$$u_i^{OA} = k_i^{OA} \exp\left(-\frac{d_i(t)^2}{(r_i^s)^2}\right) \times (x_i(t) - O(t)) \tag{3.21}$$

Where, $k_i^{OA} > 0$ is the obstacle avoidance gain. Therefore, from the Eq. 3.21, we can conclude that when, $d_i(t) >> r^s$; i.e., the sensed distance of the obstacle is far as compared to the repulsion range of the agent, $u_i^{OA} \to 0$. Hence, this specific control action will only be activated, when, $d_i(t) \leq r^s$.

Now, to describe the activation of this specific control action, we need to look into Fig. 3.4. In this figure, we have shown two agents, A, and B navigating across the environment such that

they are maintaining the force balance condition (as defined in Theorem 1). Hence, the Euclidean distance between them should be equal to $2 \times r^s$. The sensed distances of the nearby obstacles by those two agents are d_A, and d_B respectively. The blue, red, and green arrows are representing the directions of the attraction, repulsion, and obstacle-avoidance control forces respectively. Now, during the navigation, if the sensed distance of the obstacle is less than r^s for any agent, then the obstacle-avoidance control law will be triggered for that specific agent. Otherwise, employing the attraction-repulsion control rules, the environment will be explored by those agents.

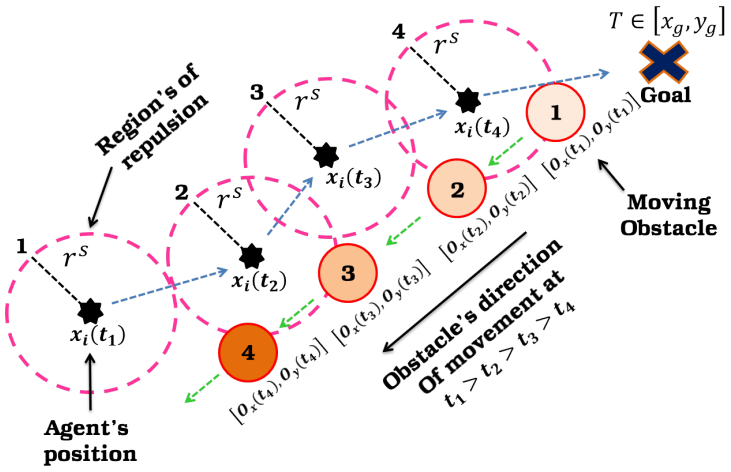


Fig. 3.5 Avoiding moving obstacle by an agent and its corresponding trajectories of movement in various time-steps.

Using the proposed obstacle-avoidance control action, the swarm would readily circumvent static impediments while approaching the destination. However, in presence of dynamic obstacles, how this controller would behave, is mainly described below (referred to Fig. 3.5). In this aspect, we have assumed that the velocity of the obstacle is less than or equal to the relative velocity of the swarm.

While designing the control law for the dynamic scenario of the i^{th} agent, we are specifically assuming that the obstacle is approaching towards the agent. At the t_1^{th} time, the position of the agent and the obstacle are $x_i(t_1) \in [p_i^x(t_1), p_i^y(t_1)]$, and $O(t) \in [O_x(t_1), O_y(t_1)]$ respectively, such that the sensed distance of the obstacle from the agent is $d_i(t_1) > r^s$. Hence, u_i^{OA} (vide Eq. 3.21) would not be triggered. At the t_2^{th} time, both the obstacle and the agent are approaching towards each other, subsequently, the sensed distance ($d_i(t_2)$) of the obstacle from the agent is $d_i(t_1) > d_i(t_2) > r^s$ and u_i^{OA} is in the idle mode. After a certain time, a situation may occur, where the obstacle is within the repulsion region of that agent as shown in Fig. 3.5. In this case, the sensed distance of the obstacle from the agent is less than or equal to the repulsion region r^s. Thus, u_i^{OA} will be activated to generate a repulsive force in the reverse direction until the distance of the obstacle from the agent will be greater than r^s. After avoiding the obstacle, the agent will again approach the target (T) under the actuation of the attraction and repulsion controllers (vide Eq. 3.21). The complete control mechanism representing the overall system is further explained in the following section.

3.4 Block diagram representing the complete control mechanism

As per the control laws, we can infer that the proposed mechanism contains three hierarchical control strategies, namely attraction, repulsion, and obstacle-avoidance controllers which will execute in a parallel fashion inside an agent. Fig. 3.6 represents the proposed architectural overview for the agent in the swarm. Depending on the control inputs from the attraction, repulsion, and obstacle-avoidance controllers, the agent block generates its position information (including the location coordinate of its neighboring agents) and the obstacles' sensing information. Based on those outputs, the controller blocks will be activated and generate necessary control inputs for the agent block.

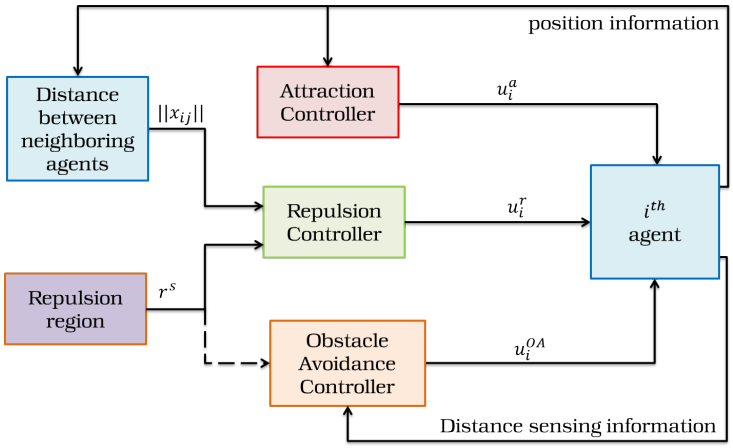


Fig. 3.6 Block diagram of the proposed control architecture with hierarchical scheme of the i^{th} agent.

Now, combining all the agents in the swarm, the block diagram of the entire system is shown in Fig. 3.7. From the figure, it is observed that each agent will be actuated based on the control inputs generated from the controller blocks until the entire swarm is reached the location of the target.

The main objective of the proposed approach is to place an agent inside the environment such that there would not be any repulsive force from its neighboring agents or obstacles, hence, allowing it to approach the target safely while forming a cohesive swarm with the other agents of the team. Subsequently, when all of the agents are stabilized with a collision-free distance from their neighbors, a unique formation will emerge amongst them. To achieve this objective, we need to formulate an optimization function for tuning the gains of the attraction controller ($k^a = \begin{bmatrix} k_1^a & k_2^a & \dots & k_N^a \end{bmatrix}, k^v = \begin{bmatrix} k_1^v & k_2^v & \dots & k_N^v \end{bmatrix}$ of Eq. 3.3), repulsion controller ($k^r = \begin{bmatrix} k_1^r & k_2^r & \dots & k_N^r \end{bmatrix}$ of Eq. 3.4), and obstacle avoidance controller ($k^{OA} = \begin{bmatrix} k_1^{OA} & k_2^{OA} & \dots & k_N^{OA} \end{bmatrix}$ of Eq. 3.21) for each agent during the navigation towards the target. In the following section, we have mainly discussed about this formulation procedure.

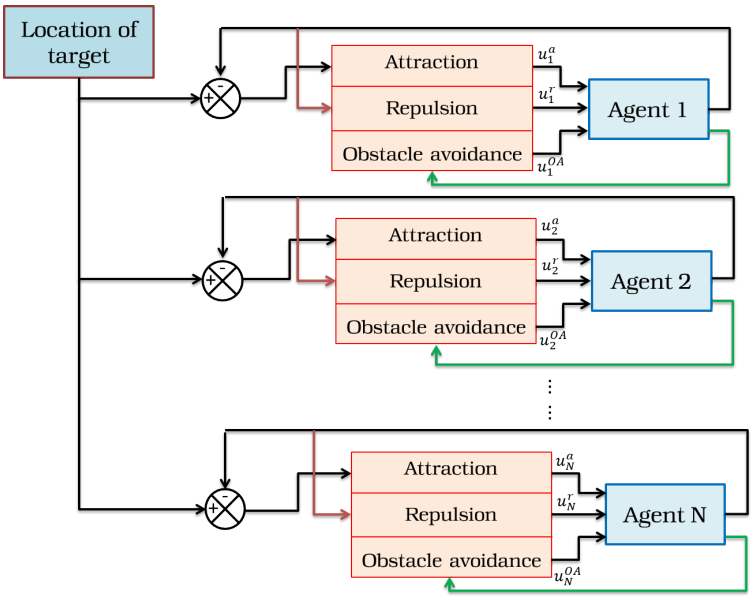


Fig. 3.7 Block diagram of the proposed control scheme for the entire system.

3.5 Formulation of the Optimization function

To formulate the optimization function, we first need to address the following question: in a cluttered environment (full of static and moving obstacles), how will the arbitrarily positioned agents form a cohesive group, create formations (absence of any obstacles), and avoid obstructions (if required) by changing the present inter-agent formation in a decentralized manner?

To solve this challenge, it is necessary to intelligently actuate the controllers (vide in Eqs 3.3, 3.4, and 3.21) so that in an obstacle-free situation; only the attraction-repulsion controller is activated, whereas in an occlusion scenario, the obstacle-avoidance controller is activated along with the attraction-repulsion controller. Therefore, it is important to adaptively regulate the controller gains (k^a, k^v, k^r, and k^{OA}) based on the current environmental condition. If the gains remain constant throughout the navigation, three possible scenarios may have occurred as enumerated below.

1. If the attraction and repulsion gains are too high compared to the obstacle-avoidance gain, the system may fail to avoid obstacles. The reason behind this fact is that the attraction and repulsion functions are only utilized for iterative path-planning (not for avoiding obstacles). So, if these gains are larger than the obstacle-avoidance gain, the system would navigate across the environment without avoiding any obstacles.

2. If the obstacle-avoidance gain is larger than the attraction-repulsion gains, then the system would prioritize avoiding obstacles above path planning. As a consequence, the system would lose inter-agent cohesiveness.

3. If all of the gains are identical, then, in the narrow pathways, the magnitude and direction of the control forces become equal and opposite. As a result, the system may be unable to proceed

further (stuck on this specific point of the path), and hence, a deadlock condition could have arisen which prohibits the system to accomplish the objectives.

So, the fundamental goal of developing the optimization function is to let an agent adapt the associated gains in each time step based on the current environmental condition. Analytically, this function (Θ_i) for the i^{th} agent at the t^{th} instance can be defined as

$$\begin{aligned} \min < \Theta_i(t) >= k_i^a \times ||\bar{x}(t) - T|| + k_i^v \times (v_i(t) - \bar{v}(t)) + k_i^r \sum_{j=1}^{N_i} \left(\frac{||x_i(t) - x_j(t)||}{2r^s} - 1 \right) \\ + \frac{1}{k_i^{OA} \times d_i^2(t)} \quad i \in [1,2,...,N] \end{aligned} \tag{3.22}$$

Where T is the location of the target and N_i defines the set of neighboring agents around the i^{th} agent. The optimization function $\Theta_i(t)$ needs be minimized during the t^{th} navigational step based on the following aspects:

1) The Euclidean distance between the average position of the swarm ($\bar{x}(t)$) with respect to the target ($T \in \Re^2$) should be minimum (the first term of Eq. 3.22).

2) The heading of the i^{th} agent should be similar than the average heading of the swarm, hence, the second term ($v_i(t) - \bar{v}(t)$) of Eq. 3.22 should be minimum.

3) For avoiding the inter-agent collision as well as producing a swarm like structure, the cohesive term (the third term of Eq. 3.22) is required to be minimized.

4) Finally, to bypass the nearest obstacles, the distance ($d_i(t)$) between an agent and the obstacle is expected to be maximized (the fourth term of Eq. 3.22).

So, the minimization of the function $\Theta_i(t)$ of the i^{th} agent (at the t^{th} time) is dependent on the current environmental condition as sensed by that agent, the positions of its adjacent agents, and the location of the goal by tweaking the controllers' gains (k^a, k^v, k^r, and k^{OA}).

So, from Eq. 3.22, we have seen that the proposed optimization problem is a 4-dimensional problem, hence, employing any classical optimization techniques [177], [178] such as dynamic programming, this problem cannot be solved as it is NP-hard [178]. So, we have used evolutionary techniques for solving the optimization function and obtaining the optimal values of the controllers' gains as described in Eq. 3.22. Initially, we have employed the traditional Particle Swarm Optimization (PSO) algorithm [98], [140]. To evaluate the performance of the PSO scheme, we have explored another optimization technique, called the Bacterial foraging optimization (BG) algorithm [162], [163] to minimize the function $\Theta_i(t)$.

3.6 Solution methodologies

The main reason for choosing the evolutionary algorithms is that these techniques employ a heuristic-based approach for solving problems that cannot be easily solved in polynomial time, such as classically NP-Hard problems, and anything else that would take too long to exhaustively process [177], [178].

3.6.1 Tuning gains using PSO scheme

The PSO algorithm [140], [149] is an evolutionary method that iteratively optimizes a problem by improving a candidate solution with respect to a given measure of quality. It solves a problem by generating a population of candidate solutions, called particles. And those particles are allowed to move around in the search space using a simple mathematical formula based on the particle's position and velocity. Finally, from the positions of all of the particles, the global best solution has been acquired. The reasons for choosing the PSO algorithm are as follows:

1. PSO is mainly used to obtain the global optimum point of various nonlinear functions.
2. Inspired by the social behavior of birds and fish, this method is mainly useful for problem-solving in groups.
3. The multidimensional problem can also be solved using the PSO procedure.

The detailed PSO algorithm can be found in Appendix 1.

From the simulation output (as shown in Fig. 3.11), it is observed that using PSO, although the swarm can approach the target successfully, however, the proposed approach does not maintain any inter-agent cohesiveness during the navigational steps. So, forming a cohesive swarm is not possible using the PSO-based approach. Hence, we have extended this work with another well-known evolutionary approach, namely Bacterial Foraging Optimization (BG) scheme [162].

3.6.2 Tuning gains using BG scheme

The BG algorithm is one of the nature-inspired optimization algorithms based on the foraging characteristics of the *E.coli* bacteria [179], [180]. The self-adaptability of an individual in bacterial foraging activities has concerned a great deal of interest today. BG has already been employed in several engineering problems such as RFID network planning [181], complex network [182], transmission loss reduction and machine learning [183] etc. The main feature of this scheme is to eliminate bacterium with poor foraging strategies and favors the propagation of genes of those bacteria that have successful foraging strategies [179]. The corresponding algorithm has four sub-phases;

1. *Chemotaxis:* It is the mechanism by which a bacterium moves by taking small steps while hunting for nutrients [176], and the main notion of BG is to replicate the chemotactic movement of virtual bacteria in the problem search space.

2. *Swarming:* In this phase of operation, when a single bacterium reaches the goal position, it delivers an attraction signal to other agents so that they can swarm together [179].

3. *Reproduction:* In this process, the group of bacterium splits into two groups based on their cost function value: the highest value means the least healthy bacteria whereas the lowest cost function value indicates the healthiest bacteria [179].

4. *Elimination & Dispersal:* In this final step, the group of bacteria which has the lowest cost function value is eliminated. Whereas, the other group exists in its current location [180].

The detailed BG algorithm can be found in Appendix 2.

From the simulation result (as shown in Fig. 3.12), it is observed that the swarm employing the BG-based scheme always maintains inter-agent cohesiveness during the navigation towards the goal. However, in a constricted passage, because of the strict cohesiveness characteristics, often this method fails to plan the next path for the robotic swarm, thus, the system approaches the deadlock condition. Whereas, in PSO, non-such situations have been observed.

So, in order to reap the benefits of both strategies, we have employed some hybrid procedures as explained below. Here, we expect that the fusion of the BG and PSO approaches would provide sufficient cohesiveness even in the narrow pathways.

3.6.3 Tuning gains using hybrid technique

The hybrid procedures have been developed using the concepts of BG and PSO principles.

BGPSO approach

The BGPSO approach [184] combines both the BG and PSO schemes. The main objective of this approach is to exchange social information using PSO and find a new solution by the elimination-dispersal phase of BG. The detailed flowchart of the BGPSO scheme is shown in Fig. 3.8.

From the above flowchart, it is seen that the PSO algorithm starts after the elimination-dispersal loop of the BG scheme in order to get the global optimal point. In this approach, we have not received any mention-worthy improvement; the simulation result follows a similar trend as that of PSO. However, the main reason for reporting this algorithm is to gain insight into the fusion techniques so that the merits of the PSO and BG schemes could be further exploited.

PSO assisted BG approach

In this approach, the BG and PSO have been fused in a sequential fashion [185]. As we previously mentioned that the BG scheme often approaches the deadlock condition in a narrow passage. So, to overcome this, PSO initiates assisting the system to avoid the deadlock situation. So, in this method, BG executes the majority of times. PSO only executes when the system fails to move further employing the BG scheme. The flowchart of the proposed algorithm is shown in Fig. 3.9.

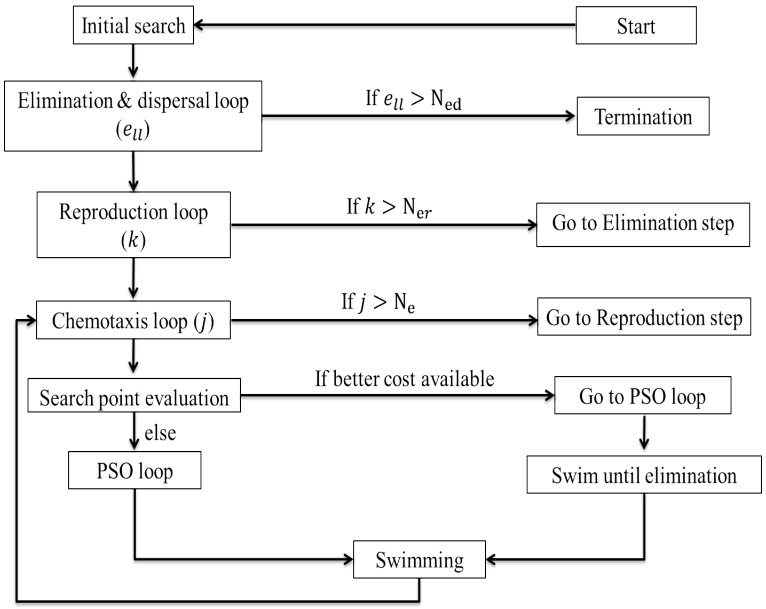


Fig. 3.8 Flowchart of the BGPSO fusion technique.

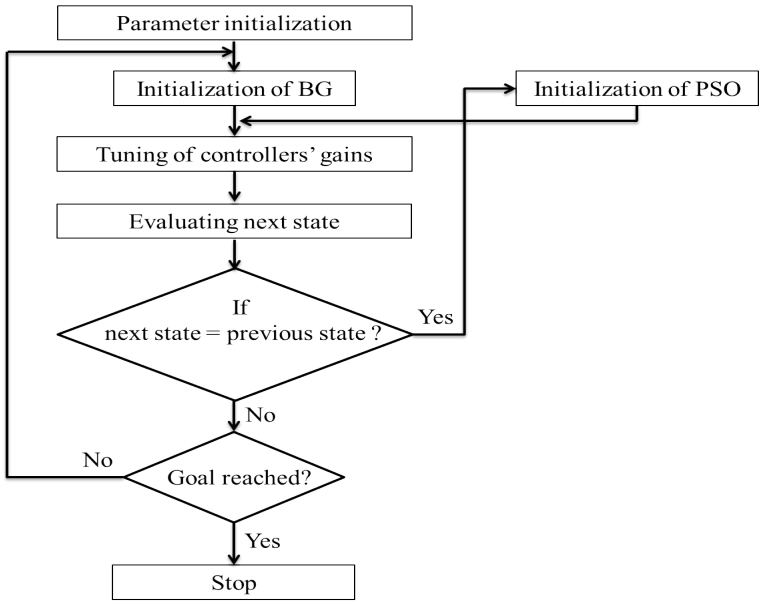


Fig. 3.9 Flowchart of the PSO assisted BG fusion technique.

From the simulated output (as shown in Fig. 3.13), we can infer that the inter-agent cohesiveness reduces specifically when PSO starts. Moreover, the performance of the proposed algorithm has been degraded with scalable agents in the swarm.

To make an adjustment between the two critical conditions (i.e. PSO and BG approaches), we have proposed a hierarchical navigation scheme inspired by the echolocation principle of the BAT algorithm [186], [187] to ensure flexible connectivity during the movement towards the target in dynamic scenarios. The work endeavors to generate a self-organizing formation control strategy for a multi-robot system and adaptive switching of the group formation based on the environmental condition to ascertain that the individual robots could avoid a collision from the nearby obstacles and neighboring robots while approaching the desired location. The corresponding approach has been described in the next section. The reasons behind the choosing BAT algorithm are:

1. It utilizes a frequency tuning phenomenon to expand the diversity of the solution in the population.

2. Moreover, it applies the automatic zooming strategy to adjust the exploration and exploitation during the search process by changing the pulse-emission rates and loudness when searching for prey.

3. Finally, among all the other swarm intelligence schemes, the bat algorithm can be implemented in real scenarios as it consumes less execution time as compared to the other techniques.

3.6.4 Tuning gains using BAT scheme

All the bats apply echolocation to detect distance and estimate the surroundings in some supernatural way [188]. They fly arbitrarily with a speed of v_i having a fixed frequency f_i with fluctuating wavelength ξ, and loudness A_0 to chase for pray. Bats can instinctively modulate the wavelength (or frequency) of their emitted pulses and modify the pulse emission rate $r \in [0,1]$, and loudness $A_0 \in [0,1]$, contingent upon the closeness of the object. In this paper, it is assumed that the loudness can shift from a substantial (positive) $A_0 = 1$ to a minimum constant $A_{min} = 0$, and frequency f_i fires on a bounded interval of $[0, f_{max}]$. In light of these approximations, the basic steps of the bat strategy are depicted as follows.

Every bat is characterized by its position $x_i(t)$, speed $v_i(t)$, frequency f_i, loudness $A_i(t)$, and the pulse emission rate $r_i(t)$ in a d-dimensional search space. The updated position $(x_i(t+1))$ and speed $(v_i(t+1))$ at the $(t+1)^{th}$ time step are given by [189]

$$\begin{aligned} f_i &= f_{min} + \beta \times (f_{max} - f_{min}) \\ v_i(t+1) &= v_i(t) + f_i \times (x_i(t) - x_*) \\ x_i(t+1) &= x_i(t) + v_i(t+1) \end{aligned} \tag{3.23}$$

where $\beta \in [0,1]$ is the randomly generated number, x_* is the current global best solution.

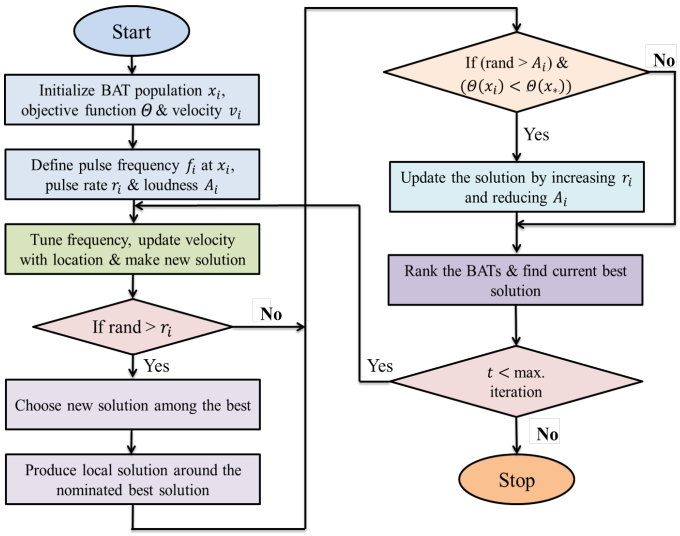


Fig. 3.10 Flowchart of the proposed Bat Algorithm.

One optimal solution is selected among the best solutions for the local search method and then the random walk is applied that can be expressed as [189]

$$x_{new} = x_{old} + \zeta \times \bar{A}^t \tag{3.24}$$

where $\bar{A}^t$ represents the average loudness of all the bats together, $\zeta \in [-1,1]$ is the random number for representing the direction and intensity of the random walk.

The Loudness and the Pulse emission rate must be reassigned as iteration continues. When a bat gets nearer to its prey, the loudness A typically increases and the pulse emission rate r decreases which are represented by the following equations [188]

$$\begin{aligned} A_i(t+1) &= \alpha A_i(t) \\ r_i(t+1) &= r_i(0)[1-e^{-\gamma t}] \end{aligned} \tag{3.25}$$

where α, γ are constant, $r_i(0)$ and A_i are the random values and $A_i(0)$ and $r_i(0)$ are in the range of $[0-1]$ and $[0-1]$ respectively. The algorithmic flowchart is represented in Fig. 3.10.

3.7 Simulation Results

To measure the efficacies of the above-mentioned approaches, we have performed a sequence of contextual simulations utilizing MATLAB 2015a software environment with the Intel-core i7 processing unit. The simulation results and discussions are explained in the upcoming sections.

Table 3.1 Parameters used in PSO

Parameters	Value
Number of particles	50
Number of iteration	50
Local weight (c_1)	1.5
Local weight (c_2)	1.5
Inertia weight (w)	0.5
Dimension of search space	4

3.7.1 Performance analysis of PSO scheme

To study the performance of the PSO scheme, the values of the various parameters are shown in Table 3.1.

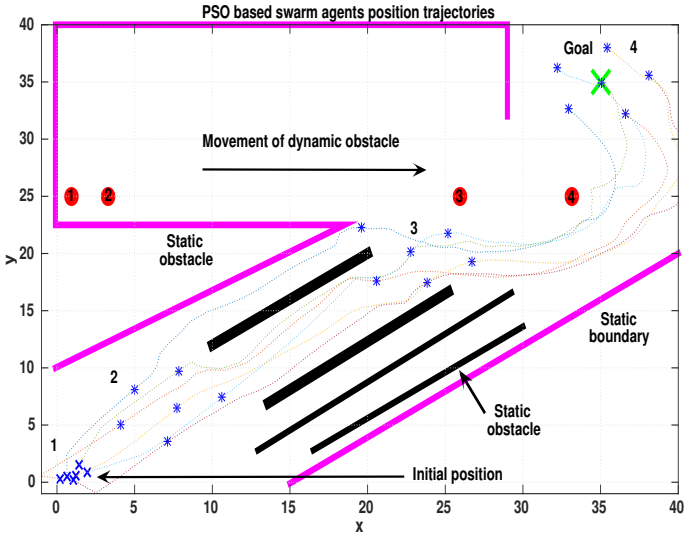


Fig. 3.11 Swarm agent position trajectories using PSO.

One of the simulation outcomes is presented in Fig. 3.11. The main objective of the optimization technique is to maximize the probability of cohesion while approaching the target by minimizing the chance of collision with obstacles and nearby agents.

In this particular study, the environment is designed with static obstacles (*black*=-*lines*), static boundaries (*magenta*-lines) and 1 dynamic obstacle (*red*-circle). The initial positions of all the agents are shown in *blue*-cross. The locations of the swarm (*blue*-star) and the positions of the dynamic obstacle at different instances of time are presented by numeric numbers. The location of target is shown in *green*-cross, and the area of the workspace is chosen to be $[40 \times 40]$ $unit^2$.

Starting from the randomized positions (as shown in 1 of Fig. 3.11), all the agents are approaching the target under the influence of the controllers as described in Eqs. 3.3, 3.4, and 3.21. At $t = 15$ sec., the agents are forming a swarm-like pattern (as shown in 2 of Fig. 3.11). In the next course of movement, the obstacles come within the field-of-view (FOV) of the swarm. In order to avoid the

Table 3.2 Parameters used in BG

Parameters	Value
Number of bacteria	100
Number of chemotaxis step	5
Number of swim	4
Number of reproduction step	4
Elimination-dispersal	2
Probability	0.25
Dimension of search space (w)	4

same, the inter-agent formation has been sacrificed (3 of Fig. 3.11). Finally, all the agents has been reached to the target location at $t = 192$ sec (4 of Fig. 3.11).

From the above outcome, it is needless to mention that using PSO, although the swarm can approach the target successfully, however, there does not exist any inter-agent cohesiveness while avoiding obstacles. So, forming a cohesive swarm is not possible using the PSO-based approach. Hence, we have extended this work with another well-known evolutionary approach, namely Bacterial Foraging Optimization (BG) scheme [162].

3.7.2 Performance analysis of BG scheme

The parameters that have been chosen for this study are presented in Table 3.2.

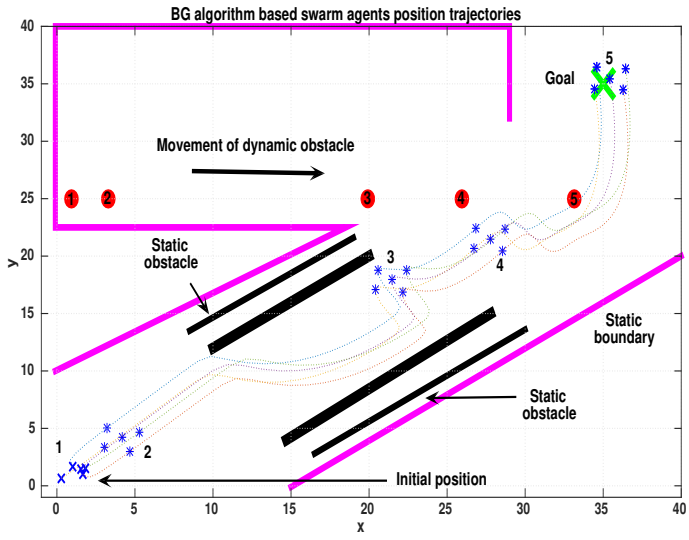


Fig. 3.12 Swarm agent position trajectories using BG.

In this simulation (as presented in Fig. 3.12), it is seen that starting from the initial position (marked by 1), all the agents get converged into a specific formation (marked by 2 of Fig. 3.6). In presence of the obstacles (both the static and moving obstacles), the inter-agent formations have not been altered (shown in 3, and 4 of Fig. 3.6). Finally, the entire system has reached the target at t = 259 sec.

From the above result, we can infer that the inter-agent cohesiveness is strictly preserved in the BG-based approach. However, here, the main drawback is that in a constricted region, often the BG-based scheme fails to plan the next path of the swarm. The reason behind this fact is that maintaining tight cohesiveness is the key feature of the BG approach, so, keeping a specific inter-agent formation does not allow the swarm system to pass through that path. As a consequence, the system approaches the deadlock situation. Hence, to get the benefits from both the strategies (PSO and BG); we have fused them to form a hybrid approach.

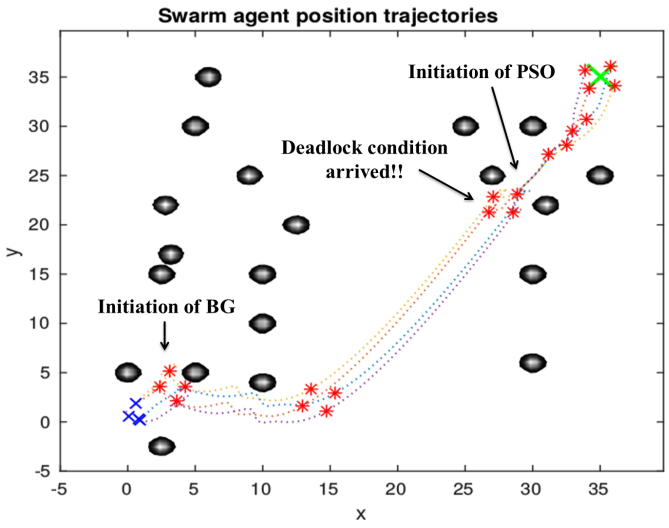


Fig. 3.13 Swarm agent position trajectories using PSOABG.

3.7.3 Performance analysis of hybrid scheme

From the simulation result of the PSO-assisted-BG approach (as shown in Fig. 3.13), it is observed that four agents (as marked by the red star) create a quadrilateral formation in order to form strict inter-agent cohesiveness. With this specific formation, the system is able to avoid obstacles. After that, a typical situation arrives, where the strict formation does not allow them to pass through narrow obstructions. In such kinds of scenarios, PSO starts, and as a consequence, the strict cohesiveness breaks so that all the agents in the swarm can easily avoid those obstacles. After avoiding it, BG starts again, and the whole system is converted into the previous formation and reached the desired location successfully.

From the above outcome, we can witness that the inter-agent cohesiveness significantly reduces when PSO starts to operate. So, this hybrid procedure does not fulfill our expectations.

3.7.4 Performance analysis of BAT scheme

To check the performance of the BAT algorithm, we have performed a few simulations (in the same software environment as stated above) with varying environmental conditions. For this study, the values of the various parameters are shown in Table 3.3.

Table 3.3 Parametric values chosen for the BAT algorithm

Parameter	Value	Parameter	Value
N	6	$r_i^s \quad i \in [1,2,...,N]$	1
δ	4	T	[35,35] unit
A_{min}	0	A_0	1
r	[0 to 1]	f_i	[0 to 1]
β	[0 to 1]	ζ	[−1 to 1]
α	0.05	γ	0.95

One of the simulation outcomes is shown in Fig. 3.14. Starting from the randomized positions, all the agents are fused into an arbitrary pattern to form a swarm-like architecture. At t = 15.5 s, a pentagonal shape has been created by the team. After sensing the static hindrances, distinctive formations (i.e. *quadrilateral*, and *elliptical*) are produced among the agents at $t = 37.4$ sec and $t = 57.33$ sec respectively. Suddenly, the path has been obstructed by a dynamic obstacle. To keep away from the same, the current formation has been reshaped into a *rectangle* at $t = 75.51$ sec. Finally, when all obstacles are avoided, initial *pentagonal*-formation has been recovered at $t = 149.21$ sec, and the entire system has reached the target zone with the specific formation at $t = 200$ sec.

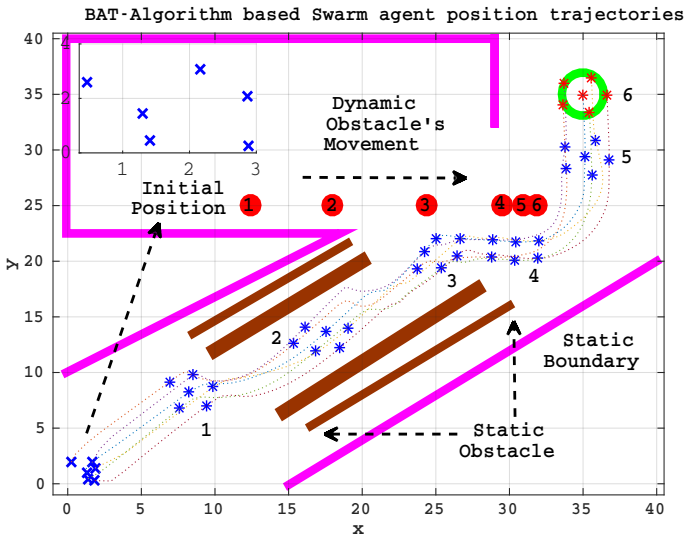


Fig. 3.14 Swarm agent position trajectories using BAT algorithm.

From the outcomes of the simulation, we have observed that for preserving grouping and cohesion as well as circumventing obstacles, several new formations have been accomplished within the swarm during the navigation towards the goal. Moreover, the initial and final formations are identical which represents the nonexistence of the obstacle-avoidance controller during the obstruction-less

environment. Similarly, without the influence of any formation controller, the swarm can restructure itself; hence, this control scheme is called *self-organizing* formation control.

3.8 Performance Comparisons

To measure the performance of all the above-mentioned techniques, we have chosen the following performance indices; such as inter-agent cohesiveness, formation adaptability with environmental constrain, path length, and algorithm convergence time. The detailed comparative analysis is given below.

3.8.1 Cohesiveness measure with different gain parameters

As mentioned previously, in every course of the movement, all of the algorithms evaluate the controllers' gains (k^a, k^v, k^r, and k^{OA}) with respect to the surrounding environment while minimizing the function Θ_i for the i^{th} agent. A comparison study has been conducted to evaluate the mean and the standard deviation of the coefficients for all the agents during the navigation as presented in Table 3.4.

Table 3.4 Comparison with other well-known swarm intelligence techniques

Algorithm	k^a		k^r		k^v		k^{OA}	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
BG	8.8	0.22	8.97	0.48	0.36	1.03	0.19	0.43
PSO	3.03	0.5	4.95	2.26	1.05	1.20	4.6	7.09
BAT	4.4	0.09	6.64	0.42	0.31	0.52	2.5	0.87
BGPSO	3.8	0.5	2.75	2.71	2.41	2.63	6.37	9.91
PSOABG	8.02	1.42	8.20	1.83	0.99	2.15	1.68	4.52

While comparing the strict formation (BG) and no formation (PSO) cases, it is observed that for strict formation, k^a (mean=8.8, standard deviation (std)=0.22) and k^r (mean=8.97, std=0.48) are relatively higher than k^{OA} (mean=0.19, std=0.43), whereas for no formation scenario, reverse condition is observed ((mean(k^a)=3.03, std(k^a)=0.5), (mean(k^r)=4.95, std(k^r)=2.26) and (mean(k^{OA})=4.6, std(k^{OA})=7.09)). Moreover, the variations of evaluated parameters for no formation are much higher than strict formation which is probably the main reason for non-arrangement situation. In BAT algorithm approach, less variations in the evaluated parameters have been seen ((mean(k^a)=4.4, std(k^a)=0.09), (mean(k^r)=6.64, std(k^r)=0.42) and (mean(k^{OA})=2.5, std(k^{OA})=0.87)) and the parameters are within the definite limits of the above two cases which assesses flexible cohesiveness and smarter path planning of the swarm.

These results are also compared with the trivial BGPSO along with the PSOABG scheme. Representative plots of the evaluated coefficients for the first agent in the team are shown in Figs. 3.15 – 3.18. For BGPSO, k^{OA} provides additional enforcement for avoiding undesirable hindrances, hence cohesiveness fails to achieve like PSO, whereas in PSOABG, though the inter-agent cohesion is mostly achieved (as BG runs a majority of times) hence, mean(k^a) is alike as BG with slightly greater

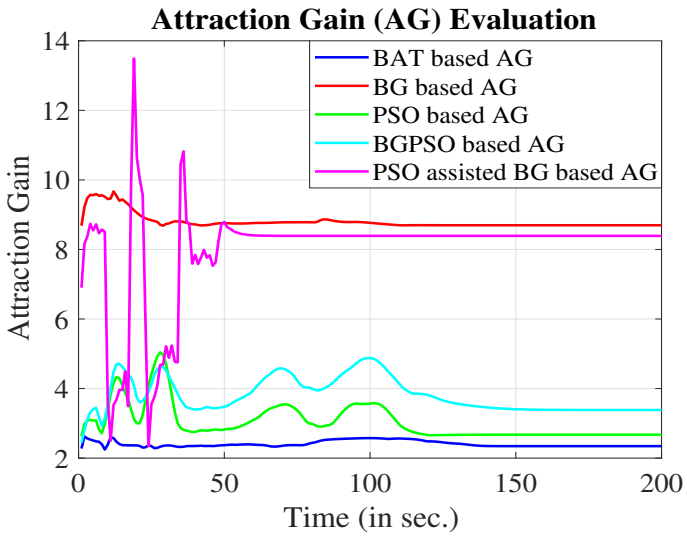


Fig. 3.15 Attraction gain.

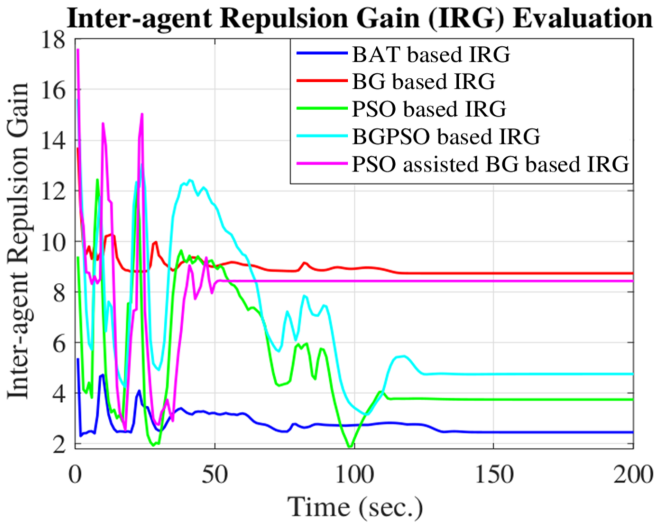


Fig. 3.16 Repulsion gain.

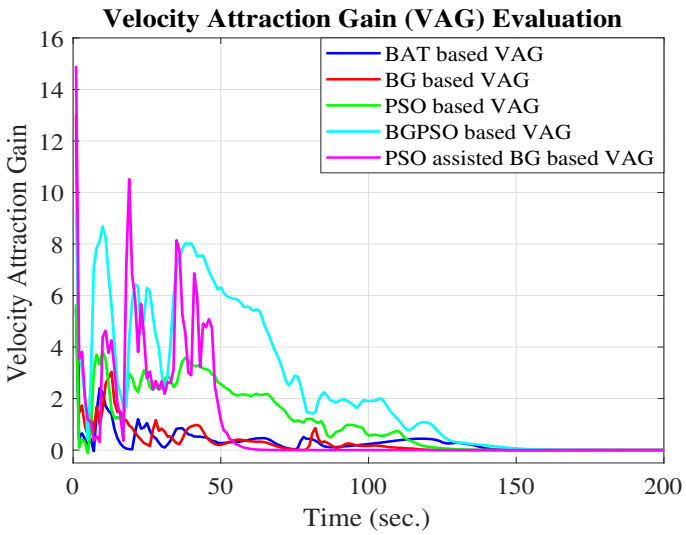


Fig. 3.17 Velocity attraction gain.

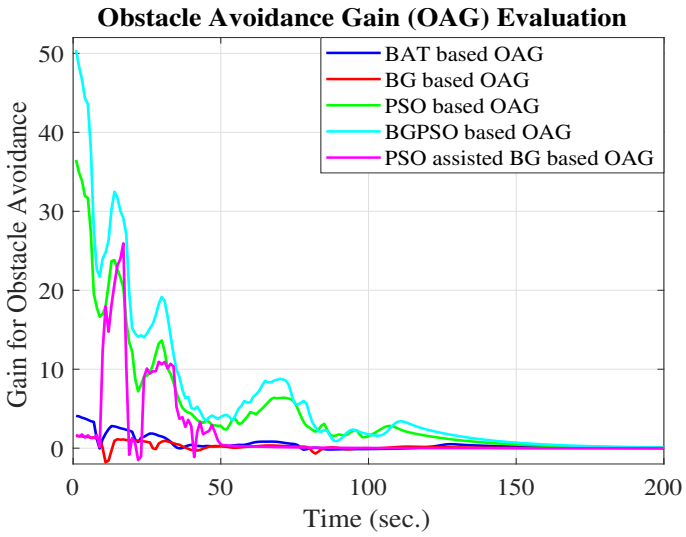


Fig. 3.18 Obstacle avoidance gain.

variations, but cohesiveness breaks when troublesome snags are in the field-of-view (FOV) of any agent and then PSO starts. This inference confirms the effectiveness of the BAT algorithm in the multi-robot path-planning problem.

3.8.2 Formation switching vs. obstacle sensed

In order to study the flexibility of the swarm system, the mean distance of all agents with respect to the group centroid is calculated. From this evaluated value, the probability of cohesiveness (CI) is calculated. Table 3.5 represents a detailed analysis. For strict formation (BG), this value is nearly constant; hence the probability of achieving formation is high with less variation (having mean and coefficient-of-variation (COV) of 0.97 and 0.01 respectively). While in the case of no-formation (PSO), as there is no grouping among agents, the probability of CI is lower with some variation (mean=0.57 and COV=0.21). Here, each agent is attempting to evade obstruction individually without any collaboration from its neighbors. However, for the BAT-based flexible formation, the controllers are always endeavoring to maintain cohesiveness among agents by slightly deforming the current pattern (if possible). Hence, the evaluated value is marginally lesser than strict formation whereas larger than no-formation with a small variation (mean=0.95 and COV=0.02) as formation needs to be sacrificed often for avoiding slow-moving hindrance. In comparison with the fusion techniques such as BGPSO (mean=0.53 and COV=0.21) and PSOABG (mean=0.82 and COV=0.08), it is seen that the results are not comparable with the BAT. The plot is shown in Fig. 3.19.

Table 3.5 Cohesive Index (CI), Success Rate and Path Length Comparison

Algorithm	CI		Success Rate of Reaching Target	Mean Path Length
	Mean	Coefficient of Variation		
BG	0.97	0.01	52%	67.94
PSO	0.57	0.21	100%	53.97
BAT	0.95	0.02	99.3%	55.82
BGPSO	0.53	0.21	100%	54.73
PSOABG	0.82	0.08	98.5%	63.75

3.8.3 Comparison of success rate and path length

In any kind of robotic path-planning framework, the preliminary goal is to increase the success rate and reduce the path length. In this paper, similar assessments have been presented for the said parameters (the entire outcomes can be found in Table 3.5) with the various techniques. It is observed that maximum-success-rate and minimum-path-length are accomplished by the no-formation (PSO) scenario, whereas lowest-success-rate and highest-path-length are achieved by the strict-formation (BG) instance. However, for formation-switching (BAT) circumstances, the success rate is much higher than BG and the path length is slightly higher than PSO. Henceforth, it can be claimed that our

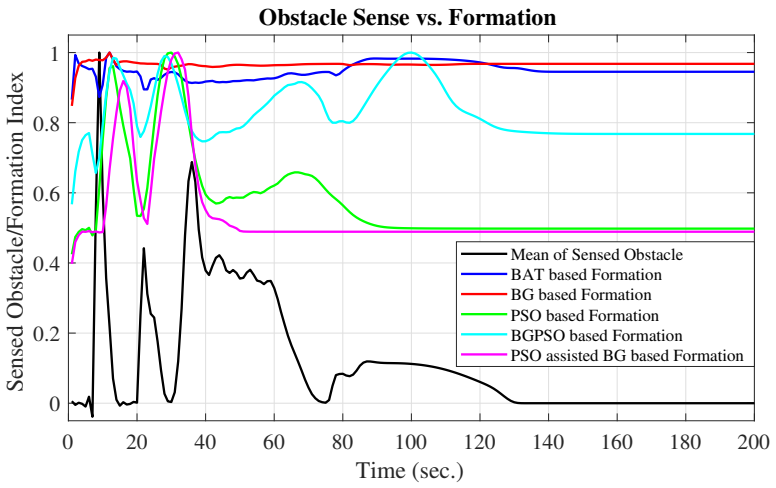


Fig. 3.19 Sensed Obstacle vs. Formation Change.

proposed technique can be an appropriate alternative to others for ensuring solidity in terms of path length, success rate, and inter-agent coordination.

3.8.4 Analyzing the capability of formation switching of each scheme

To characterize a particular formation (as shown in Fig. 3.20), the Euclidean distance of the nearest neighbors is taken into consideration. Hence, the distance matrix ($F_i \in \Re^2$) of the i^{th} agent with respect to its neighboring agents N_i can be written as

$$F_i = \sum_{j \in N_i} x_{ij}; \quad x_{ij} \leq 2r^s \tag{3.26}$$

Thus, the entire distance matrix ($F_{N \times N}$) becomes

$$F = \begin{bmatrix} F_1^T & F_2^T & \dots & F_N^T \end{bmatrix}^T \tag{3.27}$$

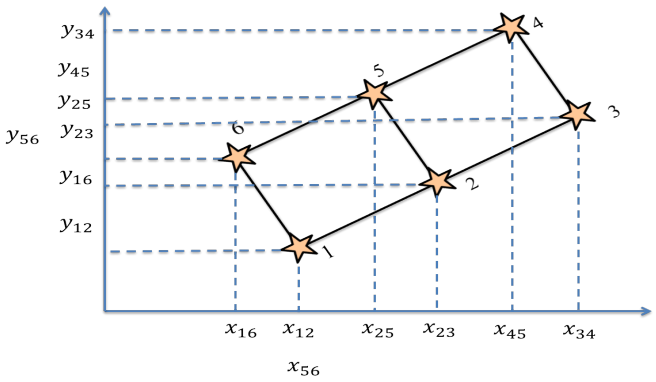


Fig. 3.20 An example of "strict" rectangular formation of 6 agents.

For a specific formation, λ_{min} (which is the minimum eigenvalue of the distance matrix F in the Eq. 3.27) will be strictly bounded over the entire region [190]. The various algorithmic outcomes in terms of λ_{min} are shown in Fig. 3.21. From the figure, it is seen that for the *BG*-inspired technique the λ_{min}-boundary region is the smallest (region *E*) which is occurred due to the strict cohesiveness among the agents. For the *PSO*-inspired method, the boundary region (region *A*) spreads over the entire region due to the lack of inter-agent cohesiveness. This inference can easily be witnessed from the figure. For the other techniques, such as *BGPSO* (region *B*) and *PSO-assisted-BG* (region *C*) the entire boundary is partially intersected by the regions *A* and *E* because of the feeble coordination among the agents. For the *bat*-inspired method (region *D*), there is a small intersection with the region *A* and a large overlapping with the region *E* that ensures the formation switching hypothesis. Moreover, in the obstacle-less environment, region *D* will be overlapped with region *E*. Hence, from this observation, we can conclude that the bat-inspired scheme is functioning like the BG-inspired technique in obstacle-less environments which is already claimed in [187].

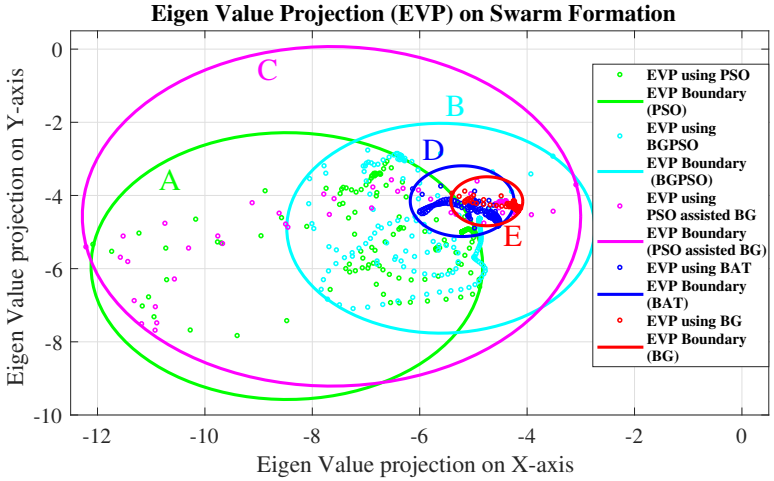


Fig. 3.21 Eigen value projection based on the inter-agent cohesiveness.

To analyze the scalability in terms of the number of agents in the team, we only consider the BAT-based approach because of the following reasons.

1. Considering the PSO-based approach, it is seen that the system does not follow any inter-agent cohesiveness. So, with increasing agents in the swarm, the cohesiveness further degrades. Hence, the PSO-based approach has not been considered for the scalability analysis.

2. For the BG-based approach, the system maintains strong cohesiveness throughout the exploration step. However, with increasing agents in the team, the complexity of the inter-agent formation would be significantly increased, hence, the success rate reduces. So, for the scalability analysis, we would not consider the BG-based approach.

Table 3.6 Scalability analysis with various number of agents

No. of Agents	Algorithm convergence time (in sec.)	
	Circle based robot gathering [2]	Proposed strategy in software simulation
6	357	200
12	698	386
30	1225	812

3.8.5 Scalability analysis

The scalability has been analyzed with various numbers of agents in the team against the performance of the BAT-based control structure so far as the convergence time is concerned. The results are represented in Table 3.6. In order to assess the performance of the proposed scheme, the results are compared with the circle-based gathering technique as presented in [2]. From Table 3.6, it is observed that with an increasing number of agents, our BAT-based method outperforms the Circle-based Robot Gathering procedure [2] in terms of the algorithm convergence timing.

From all of the above analysis, we can argue that in the multi-robotics smarter path-planning situation, the bat-inspired technique would be a robust alternative in terms of obstacle-avoidance, shortest-path-planning, and approaching target while maintaining strong inter-agent cohesiveness.

3.9 Summary

In this chapter, to achieve complete autonomy of a group of robots (swarms) towards achieving a fixed mission, we have proposed an evolutionary algorithm-based hierarchical control strategy for navigating the swarm through the landscape to a predefined target position without crashing into static/dynamic obstacles. The entire control proposition rests on three dedicated controllers: attraction controller, repulsion controller, and obstacle-avoidance controller. The stability of the proposed controllers has also been studied in the sense of Lyapunov. After that three traditional evolutionary algorithms, namely Particle Swarm Optimization (PSO), Bacterial Foraging (BG), and BAT algorithms have been employed in order to measure the performance of the system. From the outcomes, we have observed that the PSO-based system ensures less cohesiveness with the lowest time of arrival to the goal, the BG-based system confirms the highest cohesiveness with the highest time of arrival, and the BAT-based technique follows adaptive cohesiveness with the lowest time of arrival. The simulation results describe the efficacious the proposed approach.

Chapter 4

Geometric region based motion planning of swarm robots

4.1 Introduction

The motion planning of a group of robots is a computational problem in which the collective behavior of the swarm emerges through the interactions between individual robots with their neighboring agents, as well as the surrounding environment [63], [65]. In chapter 3, we have employed the evolutionary-based approach [162], [98], [188] in order to design the motion plan of a multi-robot system. However, the major drawback of the scheme is the computational time [189], [186] i.e., the process consumes huge execution time to yield a solution. Thus, this solution may not be implementable in the real environment.

In this chapter, to alleviate the above-mentioned problem, we have introduced an adaptive control scheme that exploits a geometrical region-based virtual shape control technique for avoiding obstacles in an unknown occluded environment of a swarm of robots working in a 2D environment. The proposed control action is a two-level hierarchical control scheme [191] where a local controller running inside an agent is responsible for actuating the agent to its target location while maintaining a specific distance from its neighboring agents. So, the local controller, as it appears to be, is not a single controller. Instead, two controllers, namely, the Convergence Controller and the Formation Controller are working in tandem inside this control architecture. In this context, we would like to mention that as per the local control logic, the swarm is required to converge inside the virtual geometrical region under the influence of the Convergence controller. Furthermore, it must maintain a precise formation to avoid inter-agent collision (realized by the Formation controller). As the attraction-repulsion control rule described in chapter 3 may not be sufficient to meet the above requirements, the Convergence-Formation control pair has been idealized to do so.

On the other hand, the global controller, acting as a supervisory control mode, is accountable for detecting the nearby obstacles and updating the parameters of the virtual geometrical region per

iteration by employing the sensory information from all the agents in the swarm. The concept of a virtual geometric region will be duly explained in this chapter. The block diagram of the proposed two-level hierarchical control scheme has been displayed in Fig. 4.1. The green and red rectangular

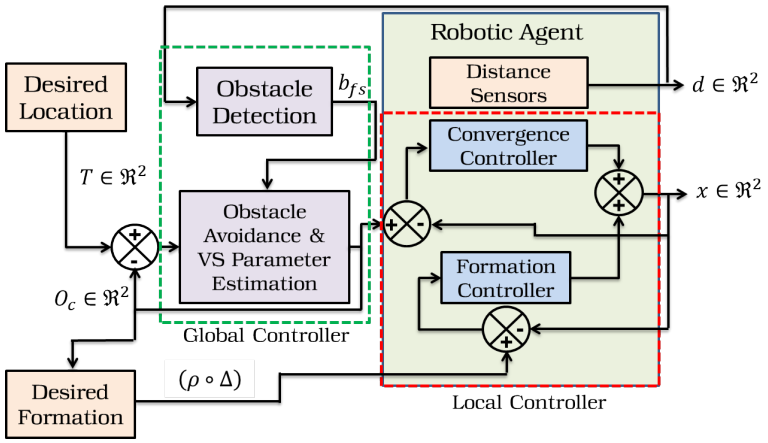


Fig. 4.1 Block diagram of the proposed control scheme.

boxes in the diagram indicate the global and local control actions respectively. Each agent in the swarm which is equipped with distance sensors to sense the distance of the nearby obstacles ($d \in \Re^2$), triggers the local controller inside it. The position of an agent is denoted by $x \in \Re^2$. Depending on the distance of the nearby obstacles, the obstacle-detection controller (of the global control action) evaluates available free space b_{fs} ahead. Employing this variable and the location of the desired target T, the obstacle avoidance controller then computes the coordinate ($O_c \in \Re^2$) of the center of a virtual geometrical region. Once the virtual region's information has been exchanged with the agents, the local control action of each agent activates to calculate the error between its current position and the center-location coordinate of the geometrical contour. Based on this error variable, the convergence controller actuates for an agent to accumulate it inside the virtual region. When all the agents approach the virtual zone, they produce a specific formation (specified as $(\rho \circ \Delta)$) under the influence of the formation controller.

The Convergence and Formation controllers are part of the local controller that executes inside a robotic agent. The distance sensors are responsible to sense the nearby environment. The global controller employs an obstacle detection rule for locating open space inside the environment and an obstacle avoidance law for updating the structural parameters of the virtual region. The desired location specifies the coordinates of the target, whereas the desired formation defines the inter-agent pattern that the swarm must maintain throughout the navigation.

With this introduction, now, we would explain the detailed proposition and evaluation of the control scheme along with the dynamics of the robotic swarm in the subsequent sections.

4.2 Dynamics of a swarm robotic system

Consider a group of N robotic agents in a 2D space are approaching towards a predefined target $T \in \Re^2$ simultaneously. The motion dynamics of the i^{th} agents, positioned at $x_i \in \Re^2$, can be written as [51]

$$\dot{x}_i = u_i; \quad \forall i = \{1,2,...,N\} \tag{4.1}$$

Where, $u_i \in \Re^2$ represents the control input acting on the i^{th} agent for approaching the target while maintaining a specific distance from its neighboring agents. The reason behind maintaining a specific distance from the nearest agents is to create a formation in the swarm.

Hence, the control input (u_i) is required to be devised in such a manner that while approaching T, all the agents will avoid the obstacles by forming a *swarm*-like framework in order to maintain the desired pattern.

To solve this hypothesis, in this work, we have borrowed the idea of a region-based shape control scheme [171] where a virtual circular region will be initially chosen as the standard region. All the agents are required to be accumulated inside it. During the course of movement, the parameters of the circles will be updated based on the agents' sensory information. And the process will continue until the swarm reaches the target. The overall idea is presented in Fig. 4.2 where the green-dashed lines represent the region of sensing to find out the open space for the swarm to pass on through the environment.

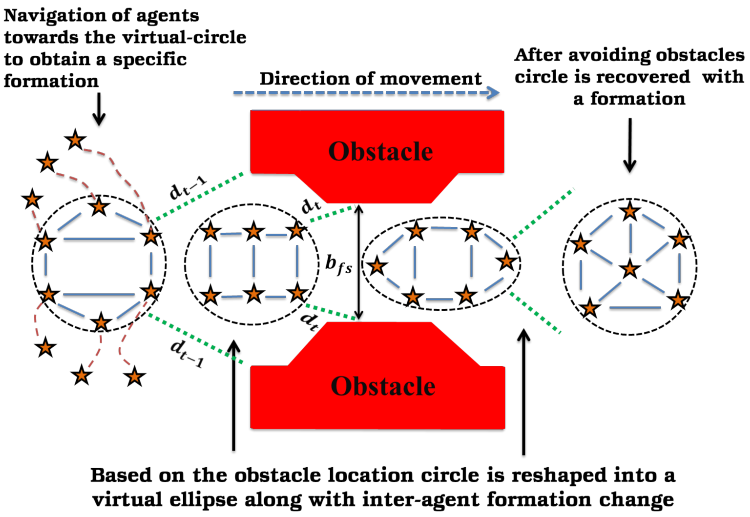


Fig. 4.2 Geometric region based swarm robotic path planning scheme.

To achieve the above-mentioned objectives, our proposed two-level hierarchical control scheme is already displayed in Fig. 4.1. In the upcoming sections, we would explain the detailed modeling of the local and the global control actions.

4.2.1 Modeling of Local Controller

The function of the local control action is not only to actuate an agent towards the target location but also to maintain a safest distance from the nearby agents in the swarm. So, this specific controller is executed inside each agent of the group. As stated earlier, the local controller can be perceived as a two-level controller comprising the Convergence Controller followed by the Formation Controller as shown in Fig. 4.1. The detailed functionality of each controller is explained below. However, before discussing the controllers let us introduce the concept and need for a virtual geometric structure.

Virtual Geometric Structure

A virtual structure is a small geometrical region inside the environment that is devoid of any obstacles [171]. To etch it inside an environment based on the sensing information, either a human operator [72] or a super-efficient robot [115] could be utilized. In this chapter, we use this technique to navigate a swarm of robots in an unknown area for avoiding obstacles. Depending on the sensory information of all the agents in the swarm, the virtual geometrical region will be formed in each time step. Subsequently, each robot is required to converge inside that region. Thus, during the exploration, several virtual regions will be created for guiding the system to approach the target while avoiding nearby impediments. To prevent the inter-agent collision and accumulate an agent inside the virtual contour, the Convergence-Formation control pair has been formulated in this work as discussed below.

Before moving on to the next section, we would like to state that we have taken the following assumption.

Assumption 1: Assuming a circular virtual structure, having a center at $O_c = [x_c, y_c]$, and radius r_c (which should be reliant on the number of agents in the swarm) is created inside the unknown environment (as shown in Fig. 4.3).

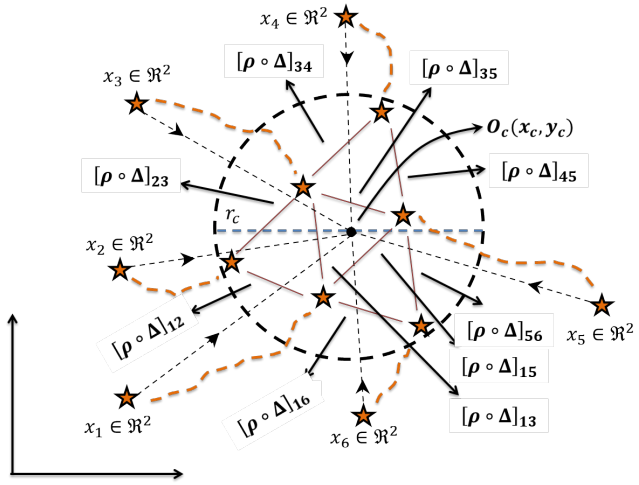


Fig. 4.3 Converging agents with desired formation inside the virtual geometrical region.

In Fig. 4.3, we have shown six agents to be moved inside a virtual circular region having center at $O_c = [x_c, y_c]$, and radius r_c. In addition, the agents are required to maintain a specific formation inside the virtual region as presented in Fig. 4.4. The shape of the formation is defined as $(\rho \circ \Delta)$

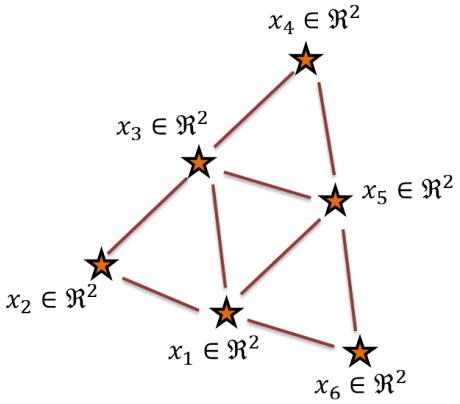


Fig. 4.4 Desired Formation pattern inside the virtual region as mentioned in Fig. 4.3.

where $\rho \in \Re^{N \times N}$ is the desired inter-agent distance matrix, $\Delta \in \Re^{N \times N}$ is the associated adjacency matrix [192], and the operator $\circ$ is the matrix element-wise multiplication or the Hadamard product [193]. From Fig. 4.4, it is seen that in order to maintain the specific formation, agent 1 is required to be connected to agents 2, 3, 5, and 6. So, these agents are considered as the neighbors of agent 1. Similarly, for agent 2, the neighbors are 1, and 3; for agent 3, the neighbors are 1, 2, 4, and 5; so on and so forth. Hence, depending on the desired formation pattern (as shown in Fig 4.4), the parameters ρ and Δ could be defined as

$$\rho = \begin{bmatrix} \rho_{11} & \rho_{12} & \rho_{13} & \rho_{14} & \rho_{15} & \rho_{16} \\ \rho_{21} & \rho_{22} & \rho_{23} & \rho_{24} & \rho_{25} & \rho_{26} \\ \rho_{31} & \rho_{32} & \rho_{33} & \rho_{34} & \rho_{35} & \rho_{36} \\ \rho_{41} & \rho_{42} & \rho_{43} & \rho_{44} & \rho_{45} & \rho_{46} \\ \rho_{51} & \rho_{52} & \rho_{53} & \rho_{54} & \rho_{55} & \rho_{56} \\ \rho_{61} & \rho_{62} & \rho_{63} & \rho_{64} & \rho_{65} & \rho_{66} \end{bmatrix}; \Delta = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.2}$$

where, $\rho_{ij} = \rho_{ji}$.

So, the desired formation $(\rho \circ \Delta)$ would be

$$\rho = \begin{bmatrix} 0 & \rho_{12} & \rho_{13} & 0 & \rho_{15} & \rho_{16} \\ \rho_{21} & 0 & \rho_{23} & 0 & 0 & 0 \\ \rho_{31} & \rho_{32} & 0 & \rho_{34} & \rho_{35} & 0 \\ 0 & 0 & \rho_{43} & 0 & \rho_{45} & 0 \\ \rho_{51} & 0 & \rho_{53} & \rho_{54} & 0 & \rho_{56} \\ \rho_{61} & 0 & 0 & 0 & \rho_{65} & 0 \end{bmatrix}; \tag{4.3}$$

Therefore, the desired distance between the i^{th} agent with respect to the j^{th} neighbor is represented by the $[\rho \circ \Delta]_{ij}$.

To converge inside the virtual region along with the desired formation, the Convergence-Formation control pair is needed to be executed in tandem as explained below.

Convergence Controller

The Convergence Controller is mainly responsible for actuating an agent towards the center of the virtual region. So, this specific action is considered as the first level controller of the local control action (Fig. 4.1). As per the control schema, when a virtual region having center at $O_c(t)$, is formed (assumption 1), the i^{th} agent positioned at $x_i(t)$ will trigger the Convergence Controller for approaching towards $O_c(t)$ (Fig. 4.2). Analytically the Convergence Control function (u^i_{cc}) for the i^{th} agent is defined as [52]

$$u^i_{cc} = -k^i_c(t) \times (x_i(t) - O_c(t)) \tag{4.4}$$

where $k^i_c(t)$ is the associated convergence gain. So, under the influence of u^i_{cc}, the i^{th} agent will approach towards $O_c(t)$ at the t^{th} time.

The sole activation of this control action in all the agents in the swarm may result in an inter-agent collision. Thus, the Formation Control action is required to be activated in tandem.

Formation Controller

The Formation control has been utilized in this work not only to create the desired formation but also to avoid the inter-agent collision. So, this specific action is considered as the second level controller of the local control action (Fig. 4.1). Mathematically, the function of this controller (u^i_f) for the i^{th} agent is defined as

$$u^i_f = k^i_f(t) \sum_{j=1, j\neq i}^{N} \left[\exp\left(-\frac{||x_{ij}||}{[\rho \circ \Delta]_{ij}} \right) \times (x_i(t) - x_j(t)) \right] \tag{4.5}$$

Where $k^i_f(t)$ is the generalized formation gain, $||x_{ij}|| = ||x_i(t) - x_j(t)||$ is the Euclidean distance between at the t^{th} time, $[\rho \circ \Delta]_{ij}$ is the desired distance that needs be maintained by the agent i with respect to the j^{th} neighbor. So, when $||x_{ij}|| < [\rho \circ \Delta]_{ij}$, the repulsion force u^i_f will be large to generate a force in the reverse direction of $(x_i(t) - x_j(t))$ until $||x_{ij}|| = [\rho \circ \Delta]_{ij}$ and vice versa.

Combining Eq. 4.4, and Eq. 4.5, the dynamics of the local controller of the i^{th} agent at the t^{th} time can be expressed as (from Eq. 4.1)

$$\dot{x}_i = -k^i_c(t) \times (x_i(t) - O_c(t)) + k^i_f(t) \sum_{j=1, j\neq i}^{N} \left[\exp\left(-\frac{||x_{ij}||}{[\rho \circ \Delta]_{ij}} \right) \times (x_i(t) - x_j(t)) \right] \tag{4.6}$$

Therefore, if we analyze the function of the local controller, we would observe that under the actuation of this particular controller, all the agents in the swarm will approach towards the virtual

circular region (centered at $O_c(t)$) while creating a specific inter-agent formation as defined by $(\rho \circ \Delta)$ (Fig. 4.3). Now, it is to be noted that the above-mentioned actions should be performed within a finite time τ, otherwise, the time to complete the whole mission will be large. And it will significantly affect the performance of the system. The following theorem can be used to determine the value of τ.

Theorem 1: Consider a multi-robot system with N number of agents as described by Eq. 4.6, the i^{th} agent will converge inside the virtual circle, defined by $B_{r_c}(O_c)$ to form a swarm, where the radius of the circle (r_c) will be

$$r_c = \frac{Nk_f}{k_c^i} \times [\rho \circ \Delta] \times \exp(-1)$$

and the convergence will happen within a finite time

$$\tau = \max_{i \in N} \{-\frac{1}{2k_c^i} \ln(\frac{r_c^2}{2V_i(0)})\}$$

.

Proof: Let $V^i = \frac{1}{2}(e^i)^2$ is the Lyapunov function [166] of the i^{th} agent, such that $e^i = (x_i - O_c)$ is the associated error between the current location of the agent with respect to the coordinate of the circle's center (as shown in Fig. 4.3), so, $\dot{e}^i = (\dot{x}_i - O_c)$ as O_c is constant. Taking derivative of the Lyapunov function, we get

$$\dot{V}^i = e^i \dot{e}^i = e^i \times (\dot{x}_i - O_c) \tag{4.7}$$

Inserting the values of $\dot{x}_i$ from Eq. 4.6, we obtain

$$\dot{V}^i = -k_c^i (e^i)^2 + k_f^i(t) \sum_{j=1, j \neq i}^{N} \left[\exp\left(-\frac{||x_{ij}||}{[\rho \circ \Delta]_{ij}} \right) \times (x_i - x_j) e^i \right] \tag{4.8}$$

To analyze the effect of the formation, we consider that all the agents are interconnected and equidistant from the i^{th} agent, hence, we can assume that

$$[\rho \circ \Delta]_{ij} = [\rho \circ \Delta]; \quad \forall j \in \{1, 2, ..., N\}$$

Now, as per the Lyapunov exponential stability theorem [166], the time-derivative of the candidate function $(\dot{V}^i)$ is required to be negative-definite, i.e. $\dot{V}^i < 0$. Hence, the right-most term of Eq. 4.8 should be less than zero, which corresponds to

$$e^i > \frac{k_f^i}{k_c^i} \sum_{j=1, j \neq i}^{N} \left[\exp\left(-\frac{||x_{ij}||}{[\rho \circ \Delta]} \right) \times (x_i - x_j) \right] \tag{4.9}$$

From Eq. 4.9, it is observed that the error inequality is dependent on the positional information of the i^{th} agent. So, if we maximize the function, we would obtain the positional-independent error function.

Hence, the time-derivative of the above function with respect to $(x_i - x_j)$, we obtain

$$\frac{\delta}{\delta(x_i - x_j)} \left[\frac{k_f^i}{k_c^i} \sum_{j=1, j\neq i}^{N} \left[\exp\left(-\frac{||x_{ij}||}{[\rho \circ \Delta]} \right) \times (x_i - x_j) \right] \right] = 0 \tag{4.10}$$

This corresponds to

$$\sum_{j=1, j\neq i}^{N} \left[\exp\left(-\frac{||x_{ij}||}{[\rho \circ \Delta]} \right) - \exp\left(-\frac{||x_{ij}||}{[\rho \circ \Delta]} \right) \times \frac{x_i - x_j}{\rho \circ \Delta} \right] = 0 \tag{4.11}$$

Solving the above equation, we get

$$\sum_{j=1, j\neq i}^{N} \left[\exp\left(-\frac{||x_{ij}||}{[\rho \circ \Delta]} \right) \left[1 - \frac{x_i - x_j}{\rho \circ \Delta} \right] \right] = 0 \tag{4.12}$$

So,

$$(x_i - x_j) = [\rho \circ \Delta] \tag{4.13}$$

Inserting the result of Eq. 4.13 into Eq. 4.9, we obtain

$$e^i > \frac{k_f^i}{k_c^i} \sum_{j=1, j\neq i}^{N} [\exp(-1) \times [\rho \circ \Delta]] \tag{4.14}$$

Now, as we consider that all the agents are equidistant from the i^{th} agent, then, Eq. 4.14 could be written as

$$e^i > \frac{(N-1)k_f^i}{k_c^i} \times [\exp(-1) \times [\rho \circ \Delta]] \tag{4.15}$$

From the above equation, we can infer that as time $t \to \infty$, the i^{th} agent converges inside the virtual circular region $(B_{r_c}(O_c))$ having radius of $r_c' = \frac{(N-1)k_f^i}{k_c^i} \times [\exp(-1) \times [\rho \circ \Delta]]$.

However, in our case, the i^{th} agent is required to converge inside the virtual circular region $(B_{r_c}(O_c))$ within a finite time τ. So, for the exponential stability of the system, we further consider that

$$r_c = \frac{Nk_f^i}{k_c^i} \times [\exp(-1) \times [\rho \circ \Delta]] \tag{4.16}$$

As, $r_c > r_c'$; therefore it is ensured that the i^{th} agent will converge inside the virtual circle $(B_{r_c}(O_c))$ within a finite time τ. And furthermore, as the agent i is an arbitrary agent of the swarm, the above result hold true $\forall j \in [1, 2, .., N]$.

Now, to obtain the upper-bound of τ, we need to perform the following calculation. As we already obtain that $\dot{V}^i$ is negative-definite, so for $e^i \geq r_c$; from Eq. 4.8, we can get

$$\dot{V}^i \leq -k_c^i \times (e^i)^2 \tag{4.17}$$

Inserting $(e^i)^2 = 2V^i$ (Lyapunov candidate function) on the above equation, we get

$$\dot{V}^i \leq -2k_c^i V^i \tag{4.18}$$

The analytical solution (of V^i) of the above equation should satisfy the following criterion.

$$V^i \leq V^i(0)e^{-2k_c^i \tau} \tag{4.19}$$

Thus, for $e^i = r_c$; we have $V^i = \frac{1}{2}r_c^2$. So, from the derivation, we can infer that the i^{th} agent will enter within the vicinity of virtual circular region ($B_{r_c}(O_c)$) at time τ, when the RHS of Eq. 4.19 satisfies

$$V^i(0)e^{-2k_c^i \tau} = \frac{1}{2}r_c^2 \tag{4.20}$$

Solving Eq. 4.20, we could get the upper-bound of τ as

$$\tau \leq -\frac{1}{2k_c^i}\ln(\frac{r_c^2}{2V_i(0)}) \tag{4.21}$$

Since $i \in \{1,2,...,N\}$ is a finite set, hence $\max(\tau)$ should exist which proves the Theorem 1.

In addition to that, inside the virtual circular region $B_{r_c}(O_c)$, all the agents should create and maintain a specific formation. The following theorem is presented to confirm this hypothesis.

Theorem 2: Consider the multi-robot system with N number of agents as described by Eq 4.6, inside the circular-region $B_{r_c}(O_c)$, all the agents of the swarm will maintain a specific formation defined by $[\rho \circ \Delta]$.

Proof: To prove this theorem, first of all, we need to formulate the formation error of the system. In a multi-robot system, the term formation error refers to the difference in distance between the desired and actual formations. So, in our case, the formation error of the system could be defined as

$$e_f = \sum_{i=1}^{N}\sum_{j\in N_i}\left((\rho \circ \Delta)^2 - ||x_{ij}||^2\right)^2 \tag{4.22}$$

From Eq. 4.22, we can infer that when the desired formation as defined by $(\rho \circ \Delta)$ is perfectly matched with respect to the current locations of all the agents in the swarm, then, $||x_{ij}|| = (\rho \circ \Delta)$, which subsequently results in $e_f = 0$ (Fig. 4.3). Now, for a desired formation $(\rho \circ \Delta)$ as mentioned in Eq. 4.5, individual robot's contribution should be in such a manner that e_f is reduced to zero or can attain a minimum value.

Now, we consider that $V_f = \frac{1}{2}(e_f)^2$ is the generalized Lyapunov candidate function [166]. Taking derivative of the Lyapunov function, we get

$$\dot{V}_f = e_f \dot{e}_f \tag{4.23}$$

Computing $\dot{e}_f$ from Eq. 4.22, we obtain

$$\dot{e}_f = -4 \times \sum_{j \in N_i} \left((\rho \circ \Delta)^2 - ||x_{ij}||^2 \right) (||x_{ij}||) \tag{4.24}$$

Inserting this result in Eq. 4.23, we get

$$\dot{V}_f = -4 \times (e_f)^2 \times \sum_{j \in N_i} (||x_{ij}||) \tag{4.25}$$

From the above equation, it is observed that Eq. 4.25 is always negative definite for any values of x_{ij}, which stabilizes the control law of the formation error. Thus, under the influence of the local controller, now, it is proven that all the agents will converge within the virtual region ($B_{r_c}(O_c)$) with the desired pattern as defined by $(\rho \circ \Delta)$.

However, during the navigational process, the sensors attached within an agent should be executing in parallel with the local controller that would assist in estimating the next location of the virtual region. This functionality is solely governed by a dedicated global controller, as shown in Fig. 4.1. Before, moving into the detailed procedures, following assumptions are made.

Assumption 2: Each agent in the team is equipped with m number of distance sensors (such as Lidar [194], and Sonar [195]) spaced at z radian apart such that an overall 2π radian is covered by an agent.

Assumption 3: Inside the virtual region, the i^{th} robot knows or senses the relative positions its neighboring agents (N_i).

4.2.2 Modeling of Global Controller

The global controller, acting in a supervisory control mode, is accountable for detecting the nearby obstacles and updating the parameters of the virtual region in each and every iteration by using the sensory information gathered by all the agents. Hence, in this work, the role of the global controller (which can also be viewed as a *centralized* controller) is to detect the free space ahead (which is realized by the first level of the global controller, i.e., the *obstacle-detection* controller) and to fit a virtual circle inside it for the safest navigation of the swarm through the environment (performed by the second level of the global controller, i.e., the *obstacle-avoidance* controller). The topological model of the corresponding scheme is presented in Fig. 4.1.

Obstacle Detection Controller

The first level of the global controller, namely obstacle detection controller is utilized to sense the nearby obstacles of the swarm. As per the assumption 2, each agent in the swarm will sense the distance of the neighboring obstacles by employing all of the sensors attached to it. Let us assume

that $\mathbf{d^i(t)} = \begin{bmatrix} d_1^i & d_2^i & ... & d_m^i \end{bmatrix}$ is the distance matrix of the m sensors for the i^{th} agent such that

$$\mathbf{d^i(t)} = \{\mathbf{d^i(t)} \mid \mathbf{d^i(t)} \geq (r_c - ||x_i(t) - O_c(t)||)\} \tag{4.26}$$

However, it is to be noted that the possibilities of presence of any obstacles outside the virtual region ($B_{r_c}(O_c)$) is much higher than inside. Hence, it is beneficial for an agent to sense an obstacle outside the virtual boundary. To achieve this objective, the sensor specific threshold limit has been chosen to be $(r_c - ||x_i(t) - O_c(t)||)$. Hence, an agent will only sense an obstacle if it is outside the virtual boundary. Depending on this information, the predicted positions of the obstacles ($\tilde{\mathbf{R}}_i(t) = \begin{bmatrix} \tilde{\mathbf{R}}_i^x(t) & \tilde{\mathbf{R}}_i^y(t) \end{bmatrix}^T$) with respect to the global coordinate frame-of-reference will be

$$\begin{bmatrix} \tilde{\mathbf{R}}_i^x(t) \\ \tilde{\mathbf{R}}_i^y(t) \end{bmatrix} = \begin{bmatrix} p_i^x(t) \\ p_i^y(t) \end{bmatrix} + \sum_{j=1}^{m} \left[\mathbf{d}_j^i(t) + \begin{bmatrix} \cos[(j-1)z + \beta^i(t)] \\ \sin[(j-1)z + \beta^i(t)] \end{bmatrix} \right] \tag{4.27}$$

Where $x_i(t) = \begin{bmatrix} p_i^x(t) & p_i^y(t) \end{bmatrix}^T$ is the current position of the i^{th} agent and $\beta^i(t)$ is the orientation of that agent with respect to the global coordinate frame of reference at the t^{th} time.

In a special situation, it may happen that the i^{th} agent is eclipsed by the j^{th} neighbors such that $||x_i - x_j|| > (r_c - ||x_i(t) - O_c(t)||)$ and that may cause a sensing error. In such scenarios, the location of the predicted obstacles' will be updated as

$$\tilde{\mathbf{R}}_i^{(x,y)}(t) = \{\tilde{\mathbf{R}}_i^{(x,y)}(t) \mid \tilde{\mathbf{R}}_i^{(x,y)}(t) \neq x_j(t)\} \quad \forall j \in N_i \tag{4.28}$$

Now, $\forall i \in [1, 2, ..., N]$, the position of the sensed obstacles can be computed as (shown in Fig. 4.5)

$$\tilde{\mathbf{R}}^{(x,y)}(t) = \begin{bmatrix} \tilde{\mathbf{R}}_1^{(x,y)}(t) & \tilde{\mathbf{R}}_2^{(x,y)}(t) & ... & \tilde{\mathbf{R}}_N^{(x,y)}(t) \end{bmatrix} \tag{4.29}$$

In Fig. 4.5, we have shown one such instance where the current positions of the agents are marked in blue star, and the location coordinates of the sensed obstacles are marked in red cross.

After sensing, now, we need to determine the closest distance among the obstacles to fit in a virtual region (either a circle or an ellipse). In this aspect, specifically, we assume that there always exists a single path through which the swarm can progress toward the goal. This specific functionality is performed by the second level of the global controller, namely the obstacle avoidance controller. The detailed design is presented below.

Obstacle Avoidance Controller

The function of this controller is not only to determine the closest distance among the obstacles but also to decide and evaluate the structural parameters of the virtual boundary.

To compute the minimum distance among the locations of the sensed obstacles, we need to first segment the sensed points into two clusters. These clusters would provide the associated free space

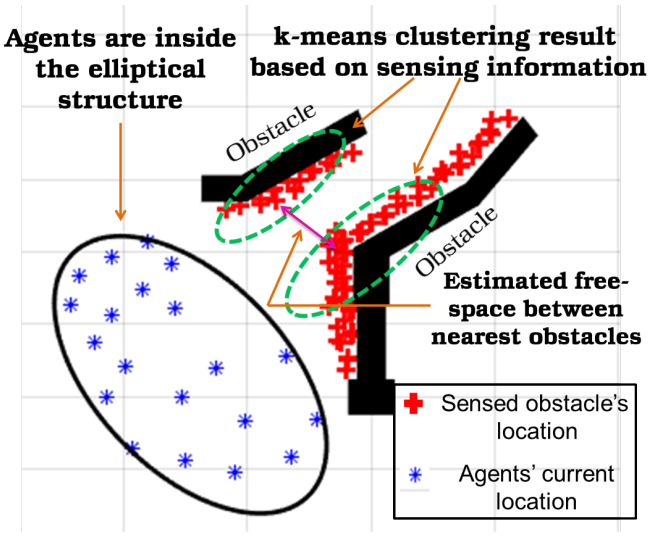


Fig. 4.5 Identification of the distance between the two nearest sensed obstacles using the k-means clustering algorithm..

available between the two nearest obstacles. The clustering has been performed using the traditional k-means clustering algorithm [196] (with $k = 2$). The outcome of the k-means algorithm is presented in the green-dashed-contour of Fig. 4.5.

After the clustering, we will measure the center coordinates of the clustered sections to determine the free space available for traversing the swarm. Hence, the Euclidean distance between the center coordinates of the clustered sections will provide the minimum distance among the obstacles. In Fig. 4.5, the magenta arrow depicts the free space between two obstacles through which the swarm is required to be passed on.

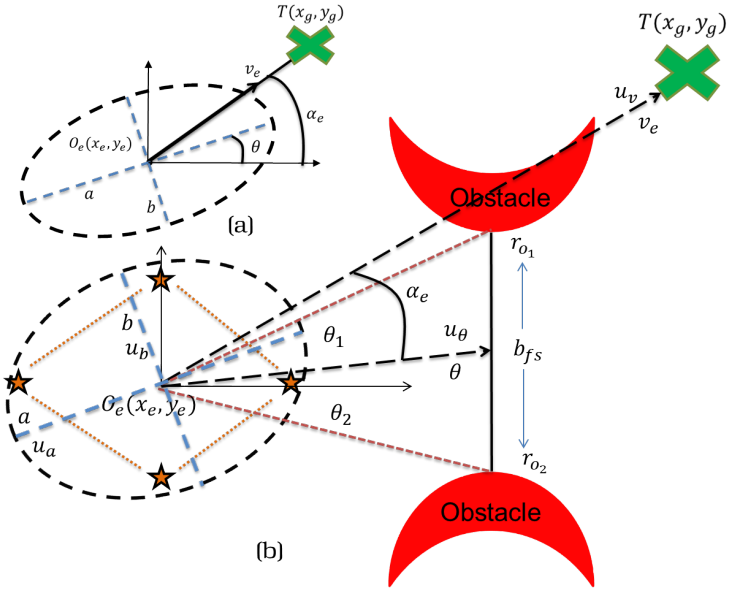


Fig. 4.6 (a) Representative elliptical structure. (b) Basic concept behind the parameter estimation of the virtual-ellipse.

Let us assume a scenario as shown in Fig. 4.6 that the evaluated center-coordinates of the clustered sections are $r_{o1}(t) \in \Re^2$ and $r_{o2}(t) \in \Re^2$ respectively. The associated orientations of these two points with respect to the center of the virtual structure $O_c(t)$ are θ_1 and θ_2 respectively at the t^{th} time. So, the available free-space (b_{fs}) between these two points is

$$b_{fs} = ||r_{o1}(t) - r_{o2}(t)|| \tag{4.30}$$

Now, we need to insert a virtual region (either a circle or an ellipse) inside the available free space b_{fs} for navigating the swarm. To fit in a virtual region inside the sensed free space, the controller has to first decide whether a circle will be fitted or an ellipse will replace it. And accordingly, the structural parameters of the region will be updated.

To fit an ellipse, the controller requires renewing the structural parameters based on the available free space (b_{fs}), which is mainly portrayed here. A representative virtual ellipse is shown in Fig. 4.6(a), where $O_e(t) = [x_e(t), y_e(t)]$ is the location of the elliptical center, $v_e(t)$ is the velocity vector toward the target $T(x_g, y_g)$, $\theta(t)$ is the orientation of the elliptical major axis ($a(t)$) with respect to the horizontal axis, $\alpha_e(t) = tan^{-1}(T - O_e(t))$ is the orientation of the ellipse with respect to the target, and $b(t)$ is the length of the minor axis at the t^{th} time. So, the dynamic equation of an elliptical structure can be represented as (Fig. 4.6(a)).

$$\begin{bmatrix} x_e(t+1) \\ y_e(t+1) \\ v_e(t+1) \\ \theta(t+1) \\ \alpha_e(t+1) \\ a(t+1) \\ b(t+1) \end{bmatrix} = \begin{bmatrix} x_e(t) \\ y_e(t) \\ v_e(t) \\ \theta(t) \\ \alpha_e(t) \\ a(t) \\ b(t) \end{bmatrix} + \begin{bmatrix} v_e(t) * cos(\theta(t)) * dt \\ v_e(t) * sin(\theta(t)) * dt \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + dt * \begin{bmatrix} 0 \\ 0 \\ u_v \\ u_\theta \\ u_{\alpha_e} \\ u_a \\ u_b \end{bmatrix} \tag{4.31}$$

Where, the control inputs u_v, u_θ, u_{α_e}, u_a, and u_b will estimate the next footprint of the virtual ellipse, as represented in Fig. 4.6. The velocity (u_v) and angular control inputs (u_θ) could be evaluated as

$$u_v = k_{v_{cir}} ||O_e(t) - T||; \quad u_\theta = \alpha_e(t) - \frac{\theta_1 + \theta_2}{2} \tag{4.32}$$

where, $k_{v_{cir}}$ is the gain of the translation-velocity controller for approaching towards T, θ_1 and θ_2 are the orientations of the two nearest obstacles, having locations at $r_{o_1}(t)$ and $r_{o_2}(t)$ from the elliptical-center $O_e(t)$ respectively (shown in Fig. 4.6(b)). On the other hand, the orientation control input (u_{α_e}) could be calculated as

$$u_{\alpha_e} = tan^{-1}(T - O_e(t)) \tag{4.33}$$

Therefore, the control inputs u_θ and u_{α_e} will try to align the major axis of the virtual ellipse toward the target, and maintain a perfect orientation in order to pass through the sensed obstacles.

Now, to determine the length of the major and minor axis of the elliptical structure, the control inputs u_a and u_b are required to be computed. The corresponding control inputs could be calculated by the following equations.

$$u_b = \begin{cases} r_c & \text{if} \quad b_{fs} \geq r_c \\ b_{fs} & \text{else} \end{cases} \tag{4.34}$$

Where, r_c is the existing circular radius. From the above equation, we can infer that if the sensed distance (b_{fs}) is greater than the circular radius r_c; then, the virtual circular region could be fitted inside the free space. However, the main problem occurs when b_{fs} is less than r_c. In such kinds of situations, a virtual ellipse would be fitted in place of the circle. And, the length of the minor axis of the ellipse will be b_{fs}. Now, we assume that the area of the circular region is $A_c = \pi r_c^2$. So, to place an ellipse in place of the circle, the prior information of the circular area has been utilized to determine the control input u_a. Thus, the control input u_a is evaluated as

$$u_a = \frac{r_c^2}{u_b} \tag{4.35}$$

Once, the virtual ellipse has been adjusted in the available free space, all the agents are expected to converge inside the newly created region with the formation as defined by $(\rho \circ \Delta)$. The above-mentioned control methodology would work well in an environment having static obstacles only.

However, in dynamic obstacle's scenario, this hypothesis may not be optimum, so, we need to formulate a generalized way of finding b_{fs} in such kind of circumstances. Let us assume that an obstacle is moving with a constant velocity and it is inside the sensing range of the swarm. Hence, at the t_1^{th} time, the predicted locations of the sensed obstacles' are $\tilde{\mathbf{R}}^{(x,y)}(t_1)$, such that $\tilde{\mathbf{R}}^{(x,y)}(t_1) = \{r_1, r_2, ..., r_n\} \in \Re^{2\times n}$, and $[r_{o_1}(t_1), r_{o_2}(t_1)]$ are the nearest obstacles' locations. To fit in a virtual circle/ellipse inside the sensed free-space (b_{fs}), the average Euclidean distance ($AD(t_1)$) from the midpoint of the points $[r_{o_1}(t_1), r_{o_2}(t_1)]$ with respect to each element of $\tilde{R}^{(x,y)}(t_1)$ can be calculated as

$$AD(t_1) = \frac{1}{n}\sum_{i=1}^{n}\left[\{||r_i - R|| \mid r_i \in \tilde{R}^{(x,y)}(t_1)\}\right] \tag{4.36}$$

Where, $R = \frac{r_{o_1}(t_1)+r_{o_2}(t_1)}{2}$ is the midpoint of the points $[r_{o_1}(t_1), r_{o_2}(t_1)]$ and n is the number of sensed obstacles. The similar analysis is performed at the t_2^{th} time ($t_2 > t_1$) considering the same midpoint (R) as previous. Let us assume that the corresponding value is $AD(t_2)$. Now, to estimate the free-space ($b_{fs}(t_2)$) inside the environment at the t_2^{th} time, the following operation is carried out

$$b_{fs}(t_2) = ||r_{o_1}(t_1) - r_{o_2}(t_1)|| \quad if \quad AD(t_2) \geq AD(t_1) \tag{4.37}$$

Now, if $AD(t_2) < AD(t_1)$, i.e. the moving obstacle blocks the estimated free-space, then the same procedure (as in Eqs. 4.36 and 4.37) will be reiterated for t_3^{th} time. Once, the free-space is correctly

estimated for the dynamic obstacles, the same procedures as mentioned in Eqs. 4.31-4.35 are followed by the global controller to fit in an ellipse or circle inside the sensed free space.

Now, one major thing to be noted is that while pushing the agents inside the newly created region, the inter-agent formation is required to be updated as well. Otherwise, some of the agents might be outside the stipulated region in order to maintain the previous formation. So, to update the formation inside the virtual region, the second level of the local control law, i.e., the formation control has to be modified. The upcoming section will address this specific issue.

4.3 Changing formation inside the virtual region: updated Formation Controller

To change the inter-agent formation inside the newly created virtual region, the adjacency matrix (Δ) and the desired distance matrix (ρ) among the agents in the swarm are required to be updated. As a consequence, the parameter ($\rho \circ \Delta$) as in Eq. 4.5 would be modified. However, to automate the scheme during the navigational process, we have proposed an algorithm, namely *spanning-tree-assisted-shape-matching* algorithm that would adjust the parameter ($\rho \circ \Delta$). After the adjustment, the formation control law of the local controller would be updated.

4.3.1 Spanning-tree expansion of a graph

A spanning tree (S) [197] of an undirected graph (G) is defined as a subgraph which is a tree that contains all of the vertices of G with a minimum possible number of edges. Tree of graph [192] can be defined as the connected unidirectional graph with no cycles. Hence, the spanning tree of a graph is a maximal set of edges of the graph that includes no cycle, or the minimal set of edges that connect all the vertices of the graph. If we consider a unidirectional graph with six numbers of edges (Fig. 4.7), two possible spanning trees with their corresponding adjacency matrices are provided as well.

Now, we are assuming a graph G with v number of vertices, having non-zero eigenvalues of $[\rho_1, \rho_2, ..., \rho_{v-1}]$ of the graph Laplacian L. Then, the possible number of spanning trees of G [192] would be

$$S(G) = \frac{1}{v} \times (\rho_1 \times \rho_2 \times ... \times \rho_{v-1}) \tag{4.38}$$

In our problem, L will be directly evaluated from the adjacency matrix Δ, and the degree matrix D. Hence, a number of spanning-tree matrices S will be generated by the well-known *cut-formation* technique [198].

After spanning-tree expansion, there will be numerous numbers of possible trees that can replace the existing formation in order to be inside the virtual elliptical region. However, to select the best one out of the possible alternatives, we should have an algorithm for generating the optimum one. The shape matching scheme can be employed in this regard as explained below.

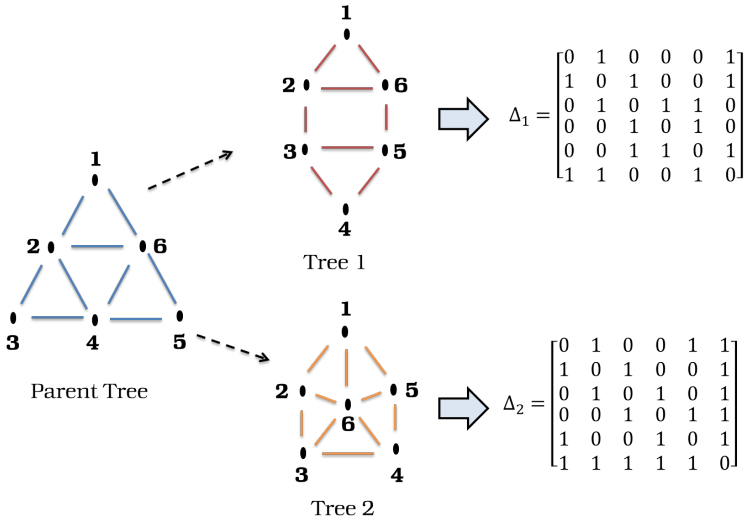


Fig. 4.7 Spanning trees of a graph having six numbers of edges.

4.3.2 Shape matching algorithm

The shape-matching algorithm has been utilized to obtain the best-spanning tree that can be employed as the current formation structure of the swarm. One important aspect to be noted is that maintaining the specific formation as obtained from the spanning tree expansion; the swarm should be within the virtual structure. To achieve this objective, we have chosen the virtual elliptical region as the template shape, then, sequentially evaluate the maximum correlation [199] to determine the mismatch factor of all the shapes as obtained from the spanning tree expansion.

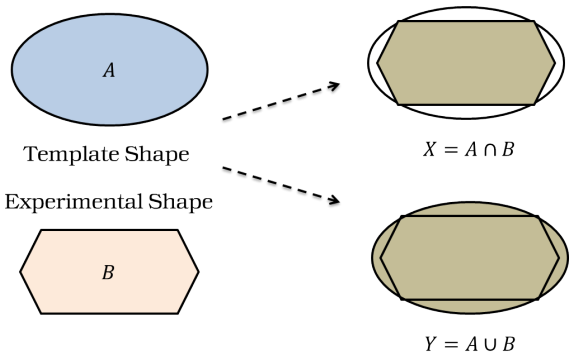


Fig. 4.8 Mismatch factor calculation of a template shape vs. an experimental shape.

4.3.3 Mismatch factor calculation

In this specific computation, one shape is considered as the template shape (the virtual elliptical region in our case), and the other one is considered as the experimental shape (the graph as obtained from the spanning expansion) as shown in Fig. 4.8. Then, the maximum correlation method has been utilized

by superimposing the experimental shape on the template shape. Depending on the superimposing result, the mismatch factor (ζ) is evaluated as defined by the following equation [199]

$$\zeta = \frac{Z}{Z+X} \tag{4.39}$$

Where, X is the area of intersection of the experimental shape and the template shape. If we assume the template shape as A, and experimental shape as B, then, X would be $A \cap B$. And, the value of Z is evaluated by the following equation

$$Z = \left(\frac{A \cup B}{A \cap B}\right) \tag{4.40}$$

After the computation, the graph (out of the spanning trees) which has a minimum mismatch factor (ζ), is considered as the updated structure of the formation for the swarm. From this specific graph, next the adjacency matrix (Δ) and the desired inter-agent distance (ρ) have been evaluated. Based on this outcome, the formation control law of the local controller will be updated. The entire *spanning-tree-assisted-shape-matching* algorithm is presented in algorithm 1. Once, the local control law has

Algorithm 1 Spanning-tree-assisted-Shape-matching algorithm

```
1:  procedure UPDATEDADJACENCYMATRIX(Δ)
2:      if b ≤ 2r then
3:          A_e = formVitualCircle(b, r);
4:          if conHull(ρ ∘ Δ) ≥ A_e then
5:              for i = 1 : S do
6:                  Δ(i) = cutFormationSpanningTree();
7:                  Δ_d(i) = ρ ∘ Δ(i);
8:                  ζ(i) = shapeMatching(conHull(Δ_d(i)), A_e);
9:              end for
10:             index = min(ζ);
11:             Δ = Δ(index);
12:         end if
13:     end if
14: end procedure
```

been updated for all the agents in the swarm, the system will actuate towards the newly created virtual elliptical region with the updated formation. The simulation result of this proposed approach can be found in Sec. 4.6.1.

One important issue of any kind of swarm robotic navigational problem is the scalability which defines the efficiency of the algorithm to work with the scalable agents in the team. In the following section, we elaborate on the scalability aspect of our proposed algorithm.

4.4 Scalability analysis with updated Formation Control Law

To analyze the scalability of the proposed approach in terms of the number of agents in the team, it is required to mention that with a large number of robotic agents, maintaining strong inter-agent cohesiveness is more beneficial than realizing a specific formation [52] which would improve the robustness of the system [200]. Therefore, if we can attain a certain level of cohesiveness among the agents [20], [21], [22], then any kind of desired pattern could be possible to achieve by updating the formation-term of the Eq. 4.8 which is already presented in the previous section. So, in this section, to analyze the scalability of the system, we have preferred to study the cohesiveness over the formation.

For achieving reliable and efficient navigation of a large group of robots in an unknown occluded environment, one possible approach is to keep close to the teammates such that the risk of possible interference among the distinct groups is significantly reduced [51]. This type of unique feature is usually observed in the biological systems [20], [21], [22]. In the swarm robotics framework, this postulate is known as *inter-agent cohesiveness*. In this work, to attain a similar objective we improvise the formation controller such that with the increasing number of agents, the group will maintain a certain level of cohesiveness. Therefore, Eq. 4.8 has been modified as

$$\dot{x}_i = -k_c^i(t) \times (x_i(t) - O_e(t)) + k_f \sum_{j=1, j\neq i}^{N} \left[\exp\left(-\frac{||x_{ij}||}{[R_s \circ \Delta]_{ij}} \right) \times (x_i(t) - x_j(t)) \right] \tag{4.41}$$

Where, R_s is assumed to be the minimum distance that each robot in the team is required to maintain from its neighboring agents to circumvent inter-agent collision inside the virtual region as well as to maintain a strong inter-agent cohesiveness. Theorem 3 has been provided to justify the above assumption.

Theorem 3: Consider a robotic system with N number of agents as shown in Eq 4.41, inside the elliptical-virtual-region $B_{r_e}(O_e)$, having a major and minor axis of $a(t)$ and $b(t)$ respectively, the i^{th} agent should maintain a certain distance of R_s from its neighboring agents in order to attain stability such that

$$||x_{ij}|| \geq [R_s \circ \Delta]_{ij}$$

and all the members of the swarm would achieve a maximum cohesiveness inside $B_{r_e}(O_e)$, if

$$R_s = \sqrt{\frac{a(t) \times b(t)}{N}}$$

Proof: Let $G_i = \frac{1}{2} e_i e_i^T$ is the generalized Lyapunov function [167] for the i^{th} agent, where $e_i = x_i(t) - O_e$. Then taking the derivative along the trajectory of the i^{th} member, we obtain the

following equation.

$$\dot{G}_i = e_i \left[-k_c^i e_i + k_f^i \sum_{j=1, j\neq i}^{N} \left[\exp\left(-\frac{||x_{ij}||}{[R_s \circ \Delta]_{ij}} \right) (x_i - x_j) \right] \right] \tag{4.42}$$

Expending the exponential term of Eq. 4.42, while neglecting the second and higher order terms, we obtain

$$\dot{G}_i = -k_c^i e_i^2 + k_f^i e_i \sum_{j=1, j\neq i}^{N} \left[\left(1 - \frac{||x_{ij}||}{[R_s \circ \Delta]_{ij}} \right) (x_i - x_j) \right] \tag{4.43}$$

In order to maintain the stability of the system, $\dot{G}_i$ of Eq. 4.43 is required to be negative semi-definite, i.e. $\dot{G}_i \leq 0$, which will only satisfy, if, $\left(1 - \frac{||x_{ij}||}{[R_s \circ \Delta]_{ij}}\right) \leq 0$, hence,

$$||x_{ij}|| \geq [R_s \circ \Delta]_{ij} \tag{4.44}$$

Therefore, the i^{th}-agent is required to maintain a minimum distance of R_s from its j^{th} neighbor inside $B_{r_e}(O_e)$ that has a major and minor axis of $a(t)$ and $b(t)$ respectively.

Assuming N agents are converged inside $B_{r_e}(O_e)$ (having an area of $\pi \times a(t) \times b(t)$), therefore, to accommodate all the agents separated by a distance of R_s from neighbors at the t^{th} time, i.e. to achieve a maximum inter-agent cohesiveness inside the virtual region, the following condition is required to be satisfied

$$N \times \pi R_s^2 = \pi a(t) b(t)$$

that generates

$$R_s = \sqrt{\frac{a(t) \times b(t)}{N}} \tag{4.45}$$

The simulation result with the scalability aspect is shown in Sec. 4.6.2.

In any kind of multi-robot path-planning problem, the major hurdle is to make the system fault-tolerant [31], [32] i.e. failure of an agent or agents may not hamper the mission accomplishment. In robotics, the agent failure is mainly occurred due to the sensing failure or the actuation failure [31], [33]. In our scheme, the sensing failure can be easily handled by the supervisory controller, because during navigation, each robot is required to transmit the sensing information to the controller. However, during the actuation failure [201], the robot may not progress further but may transmit the erroneous information to the supervisory controller which might create a major problem for assessing the next footprint of the virtual structure.

4.5 Handling agent-failure condition

To handle this, we implement an additional decision block inside the global controller so that the agent's failure can be detected earliest. This specific block evaluates the error margin ($e(t)$) of all the agents with respect to the center of the virtual structure ($O_e(t)$) in each iteration. Hence, for the i^{th}

agent, this error-margin ($e^i(t)$) at the t^{th} time will be

$$e^i(t) = x_i(t) - O_e(t) \quad \forall i \in N \tag{4.46}$$

When an actuation fault occurs inside an agent, the time derivative of the error margin is solely governed by the relative velocity of the virtual region's center. From this information, the faulty agent/s can be easily identified. After the detection, the structural information of the virtual region is required to be modified for reducing the redundant area requirement. Hence, the control input u_a will be modified as (from Eq. 4.35)

$$u_a = \frac{r_c^2}{u_b} \times \frac{N - N'}{N} \tag{4.47}$$

Where N' indicates the number of failure agents. The simulation results of the above-mentioned approaches are given below to measure the efficacy of our proposed scheme.

4.6 Simulation results and Discussions

This section presents exhaustive simulation results to demonstrate the performance of the proposed global and local controllers. All simulations are performed in the same software-hardware environment as mentioned in Sec. 3.7. Several maze-like obstructed environments have been defined, where a team of scalable agents, initially placed in arbitrary locations, are required to assemble inside a virtual region and approach the target location T while circumventing the barriers. In this simulation study, the following parameter values have been chosen, given in Table 4.1.

Table 4.1 Parametric value chosen in the simulation

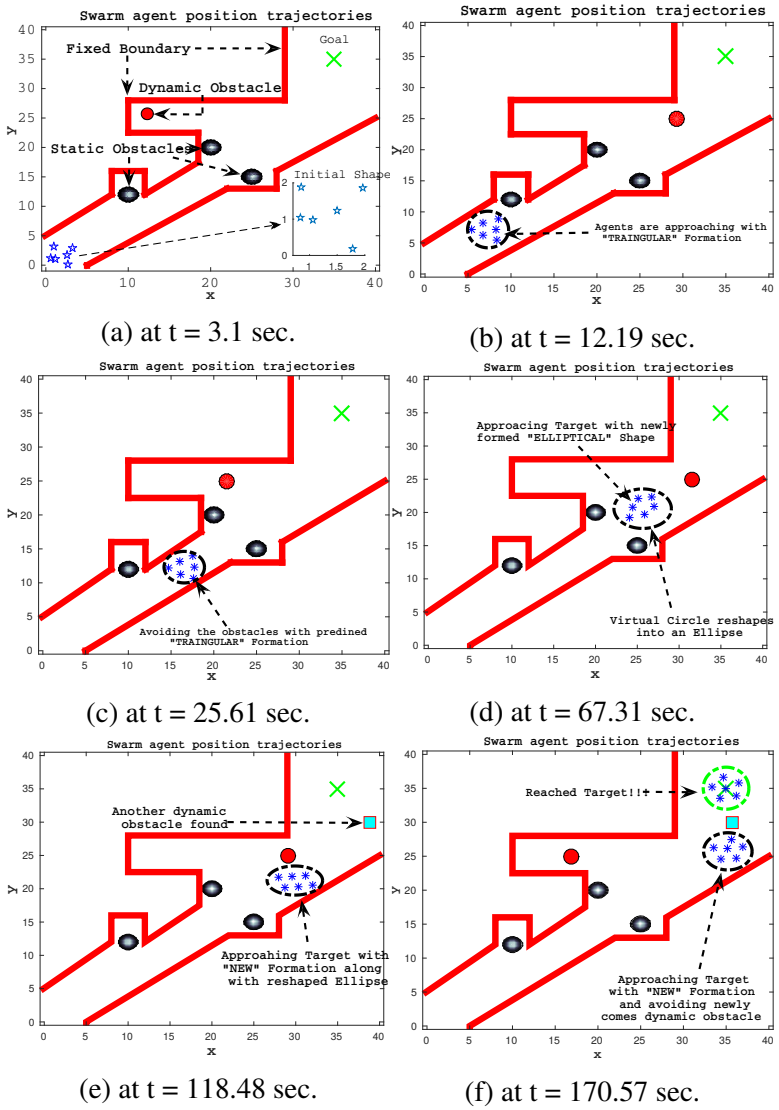
Symbol	Description	value
N	Number of agents	6
$T(x_g, y_g)$	Location of the Target	[35 unit, 35 unit]
$O_c(x_c, y_c)$	Initial location of the Virtual Circle	[5 unit, 5 unit]
r_c	Initial radius	2.5 unit
m	Number of distance sensors	8
z	Sensor's spacing	45^0
k_f	Formation gain	5
$k_{v_{cir}}$	Translation velocity gain	10

4.6.1 Simulation study with small number of agents with spanning-tree-assisted-shape-matching scheme

The complete simulation results are shown in Fig. 4.9. Primarily, a virtual circle is created at O_c with a radius of $r(initial) = r_c$ (referred to Table 4.1). All agents are forced to be inside the virtual region under the impact of the convergence controller. After converging inside the stipulated region, the

formation controller is triggered for a desired inter-agent formation. Initially, a triangular formation has been chosen (vide Fig. 4.4). The corresponding adjacency matrix ($\Delta_{triangular}$) will be

$$\Delta_{triangular} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$



(a) at t = 3.1 sec. (b) at t = 12.19 sec.

(c) at t = 25.61 sec. (d) at t = 67.31 sec.

(e) at t = 118.48 sec. (f) at t = 170.57 sec.

Fig. 4.9 Swarm-agent position trajectories in different instances in an occluded environment with the proposed technique.

After developing the desired triangular formation inside the virtual region (Fig. 4.9b), the group approaches the target T. For dodging the obstacles, the circle is marginally deformed into an ellipse.

However, the previous triangular formation is still maintained (Fig. 4.9c). At $t = 67.31$ sec. (Fig. 4.9d), the shape of the ellipse is significantly changed because of the challenging obstacles. To accommodate all the agents inside it, a new formation has been generated by the swarm. A similar phenomenon has also been observed at $t = 118.48$ sec.(Fig. 4.9e), where the agents are creating a quadrilateral formation inside the virtual region. Finally, when all obstacles are avoided, a pentagonal formation has been developed, and the entire system reaches the target zone at $t = 171$ sec (Fig. 4.9f).

4.6.2 Simulation study with scalable agents in the swarm

To analyze the performance of the scalability aspect, we have performed a simulation with 50 numbers of agents in the team. In this simulation study, we have designed a maze-like obstructed environment of 80×80 unit^2 of an area where the chain of obstacles leads to a narrow corridor for the swarm to pass as shown in the Fig. 4.10.

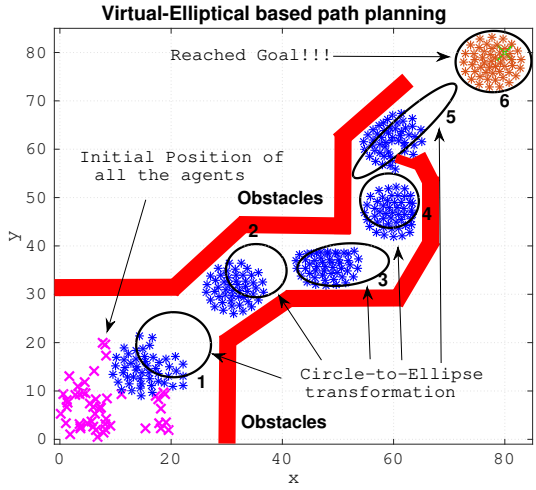
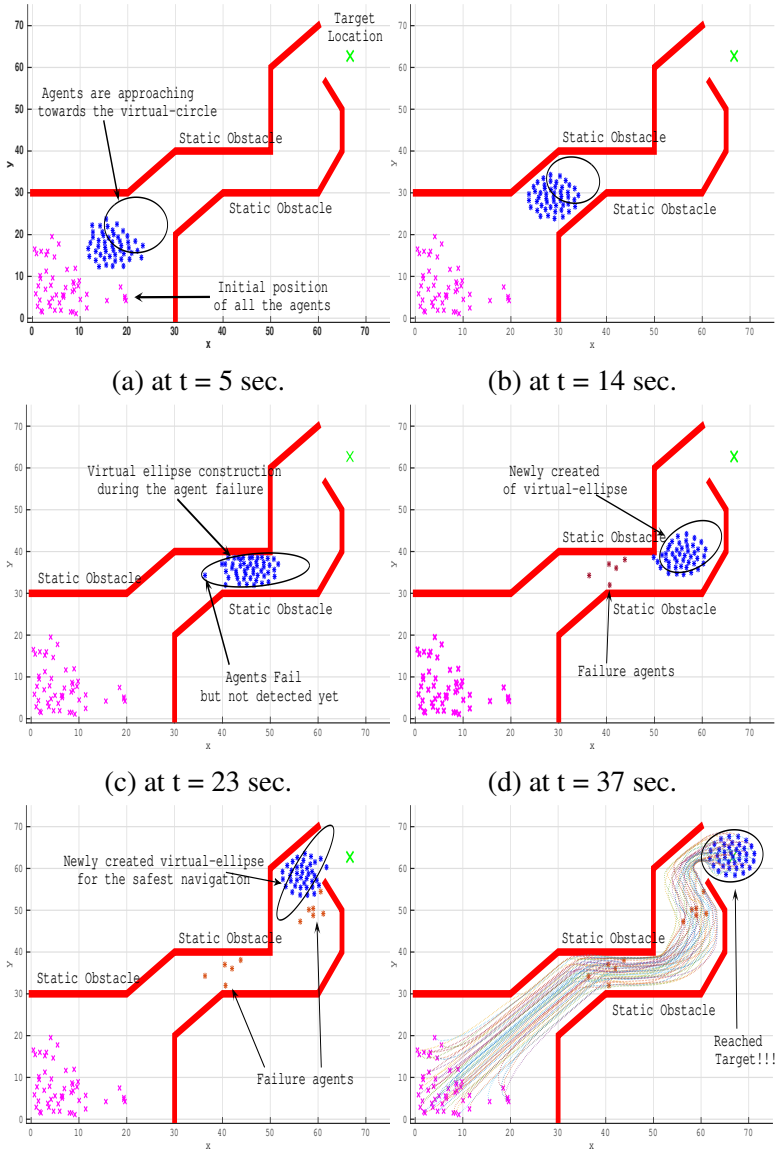


Fig. 4.10 Simulation result of the proposed scheme with 50 agents: *magenta*-cross represents the initial positions of the agents, *blue*-stars and *black*-contours are the locations of the agents and the virtual-regions at various instances respectively, the obstacles are marked by *red* and the final position of all the agents are marked by *brown*.

In this simulation study, a team of 50 robots is expected to be inside a predefined virtual-circular region as marked '1' in Fig. 4.10. In the next step, the contour has been constricted for accommodating all the agents inside it in order to obtain a strong inter-agent cohesion ('2' in Fig. 4.10) based on the agents' sensory information. To avoid narrow passages, the circle has been reshaped into an ellipse under the actuation of the global controller ('3, 4, 5' in Fig. 4.10). Finally, all the agents have reached the target location approximately at $t = 150$ sec.

4.6.3 Simulation study to handle the agent-failure condition

To study the performance of the proposed control action during the agent failure, a simulation has been done with 50 agents in a similar environment as stated in the earlier section. The entire result is



(a) at t = 5 sec. (b) at t = 14 sec.

(c) at t = 23 sec. (d) at t = 37 sec.

(e) at t = 160 sec. (f) at t = 250 sec.

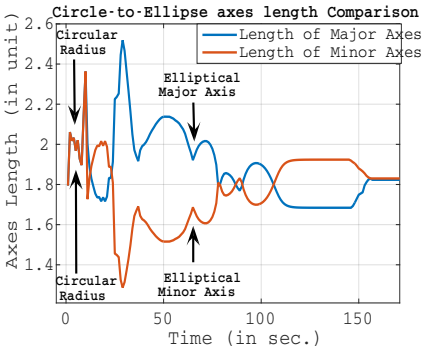
Fig. 4.11 Performance of the proposed control scheme during agent failures: *magenta*-cross represents the initial positions of the agents, *blue*-stars and *black*-contours are the locations of the agents and the virtual-contours at various time instances respectively, the obstacles are marked by *red*, the *green*-cross defines the goal-location and the *brown*-stats represent the failure agents.

represented in Fig. 4.11. At $t = 5$ sec., all the agents are progressing towards the predefined virtual region under the actuation of the local controller for obtaining a strong inter-agent cohesiveness as shown in Fig. 4.11a. To avoid obstacles, the initial structure has been reshaped into an ellipse at $t = 14$ sec (Fig. 4.11b). Suddenly, the actuation failure occurs in a few agents in the team. However, it is not yet identified by the global controller. Consequently, the global controller does not treat the situation as an exception and in a default mode creates a constricted elliptical structure as shown in Fig. 4.11c. Once the failed agents are recognized, a new elliptical contour is generated by the global controller having an area less than the previous one (Fig. 4.11d). At $t = 160$ sec., another instance is shown where a few agents fail to move further and it is properly detected by the respective controller (Fig. 4.11e). Finally, rest of the agents arrive at the target location at $t = 250$ sec as shown in Fig. 4.11f. From the Fig. 4.11, although it appears that the robots are traveling an equal distance during the discrete time periods, however, the covered distance is not identical.

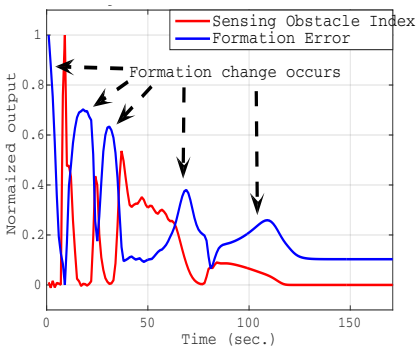
4.7 Performance analysis

In this section, the performances of all the proposed control actions are analyzed. Initially, the formation control aspect of the *spanning-tree assisted shape matching* scheme is studied, then, the proposed scalability approach along with the agent-failure handling strategies are analyzed. Finally, to address the efficacies of the proposed approach, the inter-agent cohesiveness, and the travel variance index [202], [203] have been compared with some recent well-cited literature.

4.7.1 Spanning-tree assisted shape matching scheme



(a) Axes length comparison.



(b) Sensed obstacle index vs. Formation error.

Fig. 4.12 Performance analysis of the shape matching scheme.

Fig. 4.12a represents the evaluated values of the radius/axes that govern the circle-to-ellipse conversion over time. Initially, the circular radius (r_c) is increased to some extent because of the erroneous sensing, as the agents are impending towards the virtual boundary. When reached and settled with the desired formation, tangible sensing takes place. Depending on this information, the

circle is redesigned into an ellipse, having major (a) and minor (b) axes. The entire information is depicted in Fig. 4.12a. The obstacle's sensing information adjusts the length of the two axes simultaneously and almost symmetrically to cope with the area requirement.

In Sec 4.2, we have discussed the rule behind the alteration of the shape of the virtual structure based on the obstacles' sensing information. In general, the change in formation is dependent on the relative position of the obstacles. To illustrate this phenomena, a vector ($SO(t)_{1\times U}$) has been computed for every t^{th} instances, such that

$$SO(t) = \begin{bmatrix} n_1 & n_2 & ... & n_U \end{bmatrix}_t \tag{4.48}$$

where n_i denotes the number of robots that have sensed the $i^{th}; \forall i \in \{1,2,...,U\}$ obstacle at the t^{th} time, and U defines the total number of obstacles. Hence, the obstacle-sensing index ($SOI(t)$) at the t^{th} interval is defined as

$$SOI(t) = \frac{max\,(SO(t))}{\sum [SO(t)]} \tag{4.49}$$

To explain the consequence of formation change depending on the sensing of obstacles, the standardized formation error (FE) is plotted along with SOI against time which is portrayed in Fig. 4.12b. The equation of formation error (FE) could be found in Eq. 4.22. Initially, the agents are configured into the desired formation (i.e., *triangular*); so, FE gradually reduces. With the rising SOI, several new formations are produced (by *spanning-tree assisted shape-matching* algorithm) that results in FE increasing to some extent (during evaluation) and then reducing to the smallest value (when the formation is realized). Finally, when all the obstacles are avoided, the SOI becomes low. In this scenario, a fixed and rigid formation is created among the swarm; subsequently, FE gets stabilized as seen in the figure.

4.7.2 Performance analysis of the proposed scheme with scalable agents involving agent-failure

In this section, we have analyzed the performance of the controllers with scalable agents. The study has been extended to handle the exception condition that leads to agent failure (Fig. 4.13). Moreover, to analyze the efficacies of the proposed scheme, the inter-agent cohesiveness and the travel variance index [202], [203] have been compared with some recent literature. The results are presented in Figs. 4.14, and 4.15 respectively.

Before comparing the evaluated area and axis length of the virtual regions, we would like to mention that the result as shown in Fig. 4.13 is simulated with 50 agents, operating in an unknown environment as described in Figs. 4.10, and 4.11 respectively.

During the navigational process (as presented in Fig. 4.10), the mean evaluated area of the virtual region is computed as 57.65 $unit^2$ having a standard deviation of 1.09 $unit^2$. Hence, it can be claimed that the computed area of the virtual regions during the exploration is almost identical.

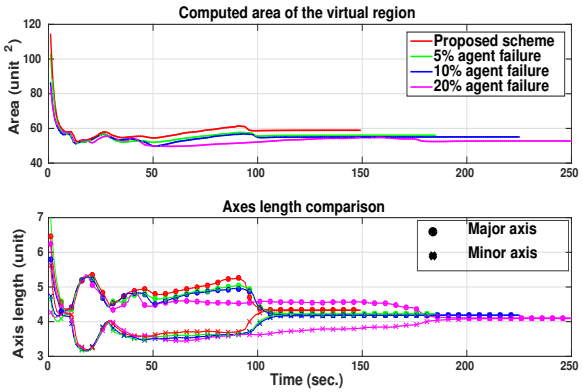


Fig. 4.13 Parametric comparison of the virtual regions with 50 agents in normal (Fig. 4.10) and agent failure conditions (Fig. 4.11).

To compare the evaluated areas of the virtual-region during the agents' failure scenario (Fig. 4.11), it is observed that for 5%, 10% and 20% agent-failure, the final computed areas of the virtual structure are 55.19 $unit^2$, 52.56 $unit^2$ and 47.80 $unit^2$ respectively which is nearly matched with our theoretical outcomes (Eq. 4.35). From this result, we can claim that the proposed controllers are performing as per our expectations during the agent-failure scenario. While comparing the axes length of the virtual regions for all the above-stated circumstances, it is observed that when all the obstacles are circumvented, the virtual structure has been reshaped into a circle i.e. the elliptical major and minor axis are converged to an identical value as shown in the Fig. 4.13.

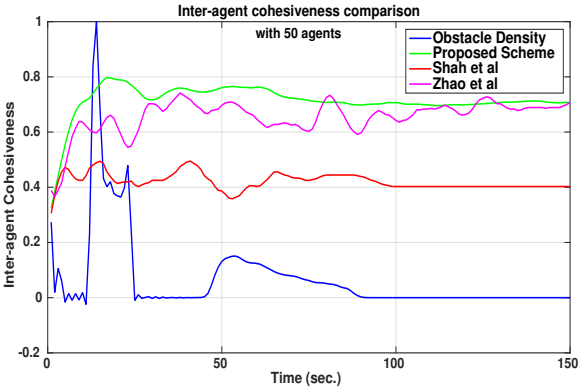


Fig. 4.14 Cohesiveness comparison for 50 agents.

To analyze the inter-agent cohesiveness, we compute the *structural density* [204] of the swarm for all iterations. Two agents in the group are considered to be cohesive if the Euclidean distance between them is greater than or equal to R_s and less than $2R_s$ (Theorem 3). Hence, the structural density ($D(t)$) can be defined as the ratio of the number of cohesive links to the maximum number of links (after discarding the self-loops), as shown in Eq. 4.50

$$D(t) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} g_j^i(t)}{N(N-1)}; g_j^i = \begin{cases} 1 & \text{if } 2R_s > ||x_{ij}|| \geq R_s \\ 0 & \text{otherwise} \end{cases} \tag{4.50}$$

The result is shown in Fig. 4.14. From the plot, it is observed that when the obstacle density is high, i.e. the robotic group is fully obstructed and the structural density attains a maximum value in our proposed approach. This ensures a strong inter-agent connection that maximizes inter-agent cohesiveness. Comparing with the other techniques, the lowest inter-agent cohesion is witnessed in [96] when there is no internal communication among the agents during the movement. In another work [202], the cohesiveness is found to be reduced because of the self-organization principle among the agents during the advancement towards the target.

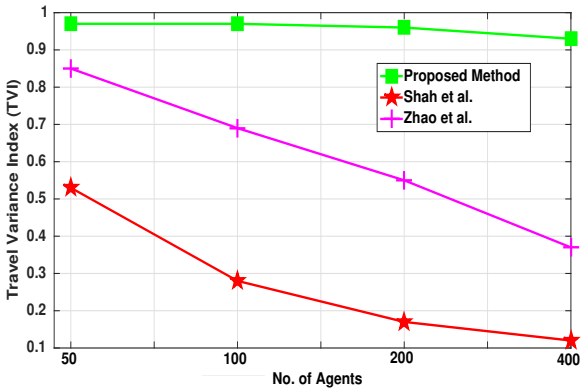


Fig. 4.15 TVI comparison with the scalability of agents.

To analyze the travel time, we have exploited the concept of *travel-variance-index* (TVI) [202] with the various number of agents in the swarm. It defines the variation of the travel distances of all the agents (i.e. N) during the navigational process. So, a TVI value of 1 indicates that all of the agents in the swarm move almost the same distance during collective motion towards the target. As a consequence, each robot consumes the same amount of energy during the navigation, extending the lifetime of the system. So, for the N number of agents, it can fluctuate from $1/N$ (the worst case) to 1 (the best case). TVI (L_V) can analytically be defined as [204]

$$L_V = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N \times \sum_{i=1}^{N} (x_i)^2} \tag{4.51}$$

In our analysis (as shown in Fig. 4.15), we have observed that with an increasing number of agents the value of the TVI is close to 1 in our proposed approach. On the other hand, for the no inter-agent communication scenario [96] L_V is deteriorating as the travel distances of all of the agents fluctuate due to lack of communication. Similarly, for the self-adaptive organization method [202], the TVI is degrading with the increasing number of agents in the team.

4.8 Summary

In this chapter, we have proposed a novel virtual structure switching technique for a robotic swarm to avoid obstacles in an unknown challenging environment. Initially, a circular region is consid-

ered for accommodating all agents with a specific formation inside the shape. In order to dodge obstructions, the virtual region is modified to an ellipse of equivalent area, additionally; the existing inter-agent formation is updated based on the newly arrived virtual-shape information. Furthermore, we establish the scalability aspect of our proposed technique to prove that within the virtual region each agent maintains strong inter-agent cohesiveness. Moreover, to handle the agent failures during the navigational process, our control strategy can identify the faulty robots and thereby update the virtual region's information adaptively. To illustrate the performance of the proposed control actions, extensive simulation studies have been carried out for various scenarios. The comparison results show the considerable potency of our scheme.

Chapter 5

Polygonal virtual-region based motion planning of a swarm of robots

5.1 Introduction

Swarm behavior is defined as the collective motion of a large number of self-propelled entities working together to achieve a common goal [27], [28], [165]. For a swarm of robots, the objective is to work collaboratively for achieving desired goals in order to successfully complete a mission [148], [160], [202].

In the previous chapter (Chapter 4), we have proposed an adaptive control scheme that utilizes a geometric region-based virtual structural technique for avoiding obstacles in an unknown environment. A virtual-circular region initially created, is considered for accommodating all the agents, with a specific formation, inside the structure. The circular structure is then modified to an ellipse of an identical area to that of the circle so as to dodge obstructions and thereby, topologically creating a narrow path to pass through for the swarm. To circumvent the challenge, the virtual circular shape has been forcefully (with the help of suitable controllers) modified to an ellipse having a larger major axis in either direction depending on the nature of the constriction. It is needless to say that the formation switching eventually results in an alteration of the current inter-agent formation. Though this technique can provide an adaptive solution in swarm robotics path planning problems, however in a narrow pathway (through which the group cannot pass while maintaining the current formation), the shape of the virtual ellipse is further expanded along the major axis while the minor axis is shrunken (to achieve the equal-area criterion) and that, in an extreme scenario, may culminate into a line-formation. Consequently, the inter-agent cohesiveness is reduced significantly [205]. Further, a marginal sensing error of an agent can easily detach it from the group which would significantly reduce the robustness of the framework [205].

To alleviate this problem, in this work, we set our objective to maximize the robustness of the swarm framework by enhancing the inter-agent cohesiveness such that failure of agent/agents due to

sensing error or battery run out or for any other reason like collapse due to hard collision, whatsoever, the remaining robotic swarm could maintain an arbitrarily shaped formation and can continue to pursue the goal. Hence, the notion of an arbitrarily shaped virtual boundary or randomly chosen boundary (in place of the virtual ellipse) has been conceived. The rationale behind this assumption is to carve out an effective surface area that is certainly greater than the area of consideration for an ellipse or circle. Hence, more number of agents can be accommodated inside the newly fitted virtual region which automatically increases the inter-agent cohesion in a narrow corridor. Moreover, the framework is scalable in terms of an increasing number of robotic agents.

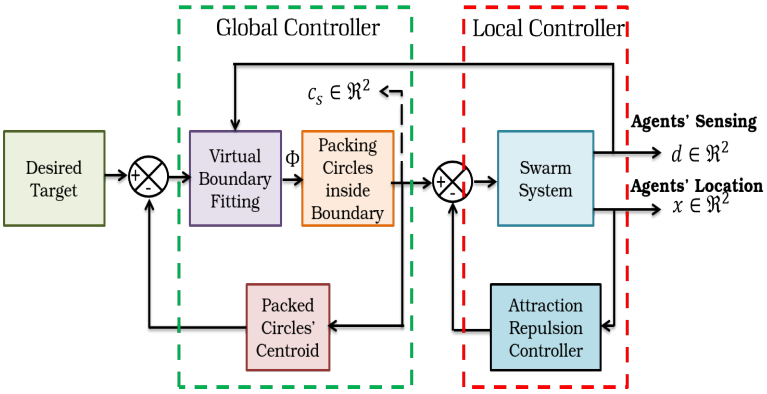


Fig. 5.1 Block diagram representation of the proposed system; Φ represents the virtual-boundary, c_s represents the set of coordinates of the circles' center fitted inside Φ.

Here also (like the control scheme proposed in Chapter 4), we have employed a two-level hierarchical control scheme [191] where a local controller running inside an agent is only responsible for actuating the agent to its target location while avoiding inter-agent collision with its nearby agents. On the other hand, the global controller, acting in a supervisory control mode, is responsible for detecting the nearby obstacles, etching arbitrary shaped virtual boundary regions, and packing circles inside the virtual contour. The block diagram of the overall control scheme is shown in Fig. 5.1. The green and red rectangular boxes in the diagram indicate the global and local control actions respectively. Each agent in the swarm which is equipped with distance sensors to sense the distance of the nearby obstacles ($d \in \Re^2$), can trigger the local controller inside it. The position of an agent is denoted by $x \in \Re^2$. Based on the detection of the nearby obstacles, the global controller computes an arbitrarily shaped virtual region (Φ) in each iteration, followed by packing circles (inside Φ). The set of coordinates of the circles' center is denoted as $c_s \in \Re^2$. Following that, the global controller will execute a circle-assignment strategy to allocate circles to each agent in the swarm. Assuming that the coordinate of the assigned circle for the i^{th} agent is $c_s^i \in c_s$. The global controller shares this particular information with each agent in the swarm. After receiving this message, the local controller of each agent attempts to push the agent into the designated circle. The next iteration begins when all of the agents have arrived at their targeted circular locations. Finally, when, the arbitrarily shaped virtual region reaches the desired target location, the process ends.

With this introduction, now, we would explain the problem statement and evaluation of the control scheme along with the dynamics of the robotic swarm in the subsequent sections.

5.2 Problem description

At the very onset, let us assume that a team of N robotic agents in a 2-dimensional topological space $\Omega \in \Re^2$ are approaching towards a predefined target location $T \in \Re^2$. The motion dynamics of the i^{th} agent, positioned at $x_i \in \Re^2$, can mathematically be characterized as [51]

$$\frac{dx_i}{dt} = f_i(x_i) \quad \forall i = \{1,2,...,N\} \tag{5.1}$$

where the function $f_i(x_i)$ defines the local control input of the i^{th} agent for approaching T while evading obstacles and maintaining solid cohesiveness among the other members of the team. Now the question is how to formulate $f_i(x_i)$ such that all agents will form a swarm-like architecture for dodging the challenging hindrances while approaching T in an unknown occluded environment?

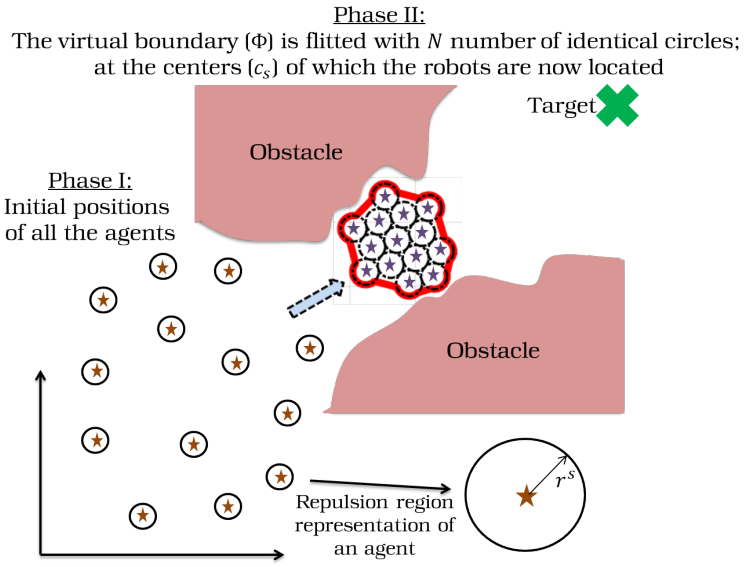


Fig. 5.2 Swarm-robotics path planning based on an arbitrarily shaped boundary fitting strategy. The virtual-boundary (Φ) is represented by the *red*-contour. Each agent (marked by *brown*-star) is associated with a repulsion region of r^s. The dotted circular regions within Φ represent the packed circles.

One thing to keep in mind is that the arbitrarily shaped polygonal region (Φ), so evolved, is extremely difficult to define analytically. Thus, locating the boundary of Φ, in general, the perimeter of the region poses a serious challenge. Hence, to tackle this particular issue, in this work, we employ the circle packing strategy [206], [207]. Already we have discussed that an agent can easily identify a specific circular region as it is the simple planar geometrical shape that can be described by even simpler mathematics. Hence if we can conceive the irregularly shaped virtual polygon to be composed

of a number of non-overlapping circles, then the respective controllers can easily push an agent to the center of such a circle within the virtual region. In the circle packing strategy, instead of decomposing the virtual region into circles, the controllers start inserting pre-defined circular regions within the virtual region. In other words, the controllers do pack circles within the aforementioned region and subsequently assign a circle for each agent and then actuate the agent towards the circle. In this way, the agents are finally forced to move into the virtual region Φ (Fig. 5.2). Thus, as per the circle packing technique, at least N number of circles (where N defines the number of agents in the swarm) are required to be packed within the Φ. The controllers are designed as follows. For each agent, a specific circle will be allotted utilizing a circle assignment strategy by the global controller. After assigning the circular regions for every agent, the local controller will be actuated so that an agent could approach its allotted circular region while avoiding the inter-agent collision. As a consequence, a strong inter-agent cohesiveness will be built up amongst the agents in the swarm (as presented in Fig. 5.2).

To handle the above-mentioned challenge, we use the two-level hierarchical control scheme (as defined in Chapter 4). The control strategy has already been displayed in Fig. 5.1. In the upcoming sections, we would explain the detailed modeling of the local and the global control actions.

5.3 Modeling of Local controller

The purpose of the local control action is not only to actuate an agent towards the target zone but also to keep it at a safe distance from the other agents of the swarm. So, this specific control action is executed inside each agent of the group. As per the objectives of the controller, it is perceived that the local control action is nothing but an attraction-repulsion controller [51], [52]. So, for the i^{th} agent, this specific control scheme could be modeled as the summation of the attraction and repulsion controllers [51], [52]. Thus, it can be expressed as (from Eq. 5.1)

$$\begin{aligned} f_i(x_i) &= -f_i^a(x_i) + f_i^r(x_i) \\ &= -k_a^i \times (x_i - c_s) + k_r^i \times \sum_{j=1, j\neq i}^{N} \left[\exp\left(-\frac{x_{ij}}{2 \times r_i^s} \right) \times (x_i - x_j) \right] \end{aligned} \tag{5.2}$$

where $f_i^a(x_i)$ and $f_i^r(x_i)$ are the attraction and repulsion functions with the associated gains k_a^i and k_r^i of the i^{th} agent respectively. $c_s \in \Re^2$ is defined as an arbitrary point inside the virtual boundary Φ. The information of c_s will be provided by the global controller (after executing the circle assignment strategy) in each step of movement. x_{ij} is the Euclidean distance [170] between the agent i and j such that $x_{ij} = ||x_i - x_j||$ and r_i^s is the region-of-repulsion of the i^{th} agent [52].

To formulate the problem more precisely, the following assumptions are made:

Assumption 1: Considering all the agents are identical in nature, presume that the associated attraction-gain, repulsion-gain and the region-of-repulsion are similar $\forall i \in \{1, 2, ..., N\}$, which can be defined as: $k_a^i = k_a$, $k_r^i = k_r$ and $r_i^s = r^s$.

Assumption 2: Each agent in the group is equipped with m number of distance sensors (such as Lidar [194] and Sonar [195]) spaced at a certain (z) radian apart.

For a successful path-planning, initially, each agent needs to sense the nearby obstacles. As all the agents are furnished with the distance sensors, at the t^{th} time-instance, the sensed obstacle's distance $\mathbf{d}^i(t)$ around the i^{th} agent will be

$$\mathbf{d}^i(t) = \{\mathbf{d}^i(t) \mid d_{thre_+} \geq \mathbf{d}^i(t) \geq d_{thre_-}\} \tag{5.3}$$

where $\mathbf{d}^i(t) = \begin{bmatrix} d_1^i & d_2^i & \dots & d_m^i \end{bmatrix}_t$ is the distance-matrix of the m sensors, d_{thre_+} and d_{thre_-} are the sensor specific threshold limits such that $d_{thre_-} > 2r^s$.

Once, $\mathbf{d}^i(t)$ has been collected by the i^{th} agent, it will share this particular information with the global controller. Depending on the sensory information of all the agents in the swarm, the global control will evaluate the coordinates of the sensed obstacles in order to etch the arbitrarily shaped polygonal virtual region as explained in the upcoming section.

5.4 Modeling of Global controller

The global controller, operating in supervisory control mode, is responsible for recognizing surrounding impediments and updating the virtual polygonal region in each iteration using the sensory data acquired by all the agents. Hence, in this work, the role of the global control action can be viewed as a *centralized* controller (Fig. 5.1).

5.4.1 Recognizing nearby obstacles

To identify the nearby obstacles, the global controller utilizes the sensed obstacles' distance ($\mathbf{d}^i(t); \forall i \in [1,2,...,N]$) information of all the agents in the swarm. Based on this, it will evaluate the estimated obstacles' position $\tilde{\mathbf{R}}_i(t) = [\tilde{\mathbf{R}}_i^x(t) \quad \tilde{\mathbf{R}}_i^y(t)]^T$ with respect to the global coordinate frame of reference, can be written as

$$\begin{aligned} \tilde{\mathbf{R}}_i^x(t) &= p_i^x(t) + \sum_{j=1}^{m} \left[d_j^i(t) \times \cos[(j-1)z + \theta^i(t)]\right] \\ \tilde{\mathbf{R}}_i^y(t) &= p_i^y(t) + \sum_{j=1}^{m} \left[d_j^i(t) \times \sin[(j-1)z + \theta^i(t)]\right] \end{aligned} \tag{5.4}$$

where $x_i(t) = [p_i^x(t) \quad p_i^y(t)]^T$ is the present location of the i^{th} agent and $\theta^i(t)$ is the orientation of the agent with respect to the global coordinate frame of reference at the t^{th} instance.

Therefore $\forall i \in [1,2,...,N]$, the predicted obstacles' locations $\tilde{\mathbf{R}}(t) = [\tilde{\mathbf{R}}^x(t) \quad \tilde{\mathbf{R}}^y(t)]^T$ will be

$$\begin{aligned} \tilde{\mathbf{R}}^x(t) &= \begin{bmatrix} \tilde{\mathbf{R}}^x_1(t) & \tilde{\mathbf{R}}^x_2(t) & ... & \tilde{\mathbf{R}}^x_N(t) \end{bmatrix} \\ \tilde{\mathbf{R}}^y(t) &= \begin{bmatrix} \tilde{\mathbf{R}}^y_1(t) & \tilde{\mathbf{R}}^y_2(t) & ... & \tilde{\mathbf{R}}^y_N(t) \end{bmatrix} \end{aligned} \tag{5.5}$$

Eq. (5.5) will provide the coordinate information of the nearby sensed obstacles (within the range of $[d_{thre_-}, d_{thre_+}]$ from the present location of the agents) with respect to the global coordinate frame of reference. Once, the coordinates of the nearby obstacles have been determined, the next aim of the global controller is to fit in a virtual arbitrarily shaped polygonal boundary region inside the sensed space as explained below.

5.4.2 Arbitrarily shaped polygonal boundary fitting

A polygon is a two-dimensional geometrical figure with a finite number of sides. Any polygon consists of connected short straight line segments. Triangles, rectangles, and hexagons are some examples of polygons. However, an arbitrarily shaped polygon, which we call our virtual region (etched by the global controller), cannot be defined with the help of a known geometrical shape and it may contain an infinite number of small line segments. Thus, in this chapter, we refer to the fitted virtual regions as the *arbitrarily shaped polygonal boundary* regions.

For obtaining an arbitrary outline from the detected obstacles' coordinates ($\tilde{\mathbf{R}}(t)$), there are several methods available in the literature for identifying a boundary. To name a few we can mention propositions like polygonal curve fitting [208], the principle-components based morphological analysis [209], the Eigenshape analysis [210], the Elliptical Fourier Analysis [211], [212] (EFA) and so on and so forth. Out of these methods, EFA is a robust way of drawing the shape of a 2-D region by taking the Fourier Transform [211] on the evaluated coordinates of the sensed obstacles' data.

Background of EFA

EFA is first introduced in 1982 by Kuhl and Giardina [211] to quantitatively define a closed contour in a morphological form. It has two sets of equations containing sine and cosine terms [212], one is for x-coordinates, and the other is for y-coordinates. Combining these two equations for different harmonics, a series of overlapping ellipses could be obtained which would approximate the boundary of the arbitrarily shaped virtual region.

We know that a periodic non-sinusoidal waveform can be expressed as a summation or superposition of a number of sinusoidal or cosinusoidal waveforms having a fundamental frequency component and some higher harmonic components (at times, a dc signal can also be present) with the help of Fourier Analysis by determining the Fourier Coefficients [211]. This idea has been extended for EFA. With the help of this technique, an arbitrarily shaped region can be decomposed into a series of

intersecting ellipses, the superposition of which would result in a non-geometrically defined region (vide Fig. 5.3).

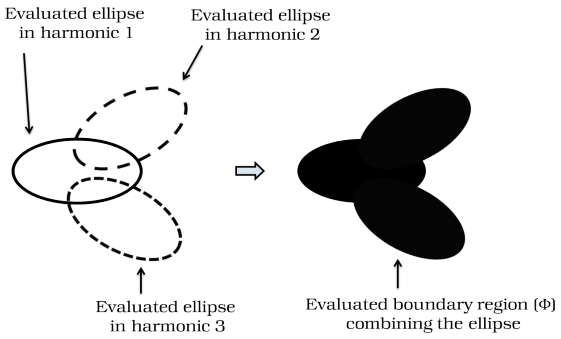


Fig. 5.3 Generating the boundary of the virtual region by superposing the elliptical components.

The major difference between traditional Fourier series [211] and EFA is that the latter utilizes two sets of equations for each harmonic as an ellipse is planer topology. Thus, for each harmonic, four coefficients (α_n, β_n, γ_n, and δ_n) are obtained for defining the boundary region of an elliptical contour. On the other hand, the Fourier series [211] employs a single periodic non-sinusoidal function, with each harmonic consisting of only two coefficients (a_n, b_n).

In this problem, the EFA technique has been applied to the evaluated coordinates of the sensed obstacles (of each of the agents) to etch the arbitrarily-shaped virtual region by the global controller. From these location coordinates, by EFA, the harmonic coefficients are evaluated and the ellipses are constructed.

Now we try to justify the fact why the EFA technique has been used in this work. It is needless to say that if we join the sensed obstacles' location coordinates (or in general the sensed data) in a proper sequence, then also, we could obtain the boundary of the arbitrarily shaped virtual region. In our situation, however, the agents individually and simultaneously share their sensed data at every time instant with the global controller. But the global controller does not have any mechanism to map the sensed positions topologically on and around the obstacles. So the global controller cannot decide whether the position sensed by agent 1 is to be connected to the point sensed by agent 2 or it should be connected to the position sensed by some other agent, say agent 7. As a result, the global controller fails to generate the virtual region by successively joining the points. Therefore, we need to use the EFA technique to approximate the boundary of the virtual region by summing a series of overlapping ellipses.

The detailed methodology of the EFA technique is explained below.

Methodology of EFA technique

As mentioned earlier, in the EFA technique, there are two sets of equations including sine and cosine terms [211] to reconstruct the virtual region $\Phi(t)$ at the t^{th} time. Mathematically, the EFA equations are defined as [212]

$$\begin{aligned}\Phi_x(t) &= \alpha_0 + \sum_{n=1}^{\infty}\left(\alpha_n\cos(\frac{2n\pi t}{T_f}) + \beta_n\sin(\frac{2n\pi t}{T_f})\right)\\ \Phi_y(t) &= \gamma_0 + \sum_{n=1}^{\infty}\left(\gamma_n\cos(\frac{2n\pi t}{T_f}) + \delta_n\sin(\frac{2n\pi t}{T_f})\right)\end{aligned} \tag{5.6}$$

where $(\Phi_x(t), \Phi_y(t))$ are the evaluated points on the perimeter of the virtual boundary at the t^{th} time, α_n, β_n, γ_n, and δ_n are the EFA coefficients, α_0, and γ_0 are the horizontal and vertical offsets (which are equivalent to the DC components of simple Fourier analysis), n defines the number of harmonic element, and T is the time period of the fundamental component.

While evaluating the contour ($\Phi(t)$), Δt_i is the time required for drawing the line-segment of the contour that links $\tilde{\mathbf{R}}^{i-1}(t) \rightarrow \tilde{\mathbf{R}}^{i}(t)$ and V is the total time taken to draw the entire contour having k number of coordinates, so $V = \sum_{i=1}^{k}\Delta t_i$. Therefore, the EFA coefficients are described as [212]

$$\begin{aligned}\alpha_n &= \frac{V}{2n^2\pi^2}\sum_{i=1}^{k}\frac{\Delta\mathbf{R}_x^i}{\Delta t_i}\left[\cos\frac{2n\pi t_i}{V} - \cos\frac{2n\pi t_{i-1}}{V}\right]\\ \beta_n &= \frac{V}{2n^2\pi^2}\sum_{i=1}^{k}\frac{\Delta\mathbf{R}_x^i}{\Delta t_i}\left[\sin\frac{2n\pi t_i}{V} - \sin\frac{2n\pi t_{i-1}}{V}\right]\\ \gamma_n &= \frac{V}{2n^2\pi^2}\sum_{i=1}^{k}\frac{\Delta\mathbf{R}_y^i}{\Delta t_i}\left[\cos\frac{2n\pi t_i}{V} - \cos\frac{2n\pi t_{i-1}}{V}\right]\\ \delta_n &= \frac{V}{2n^2\pi^2}\sum_{i=1}^{k}\frac{\Delta\mathbf{R}_y^i}{\Delta t_i}\left[\sin\frac{2n\pi t_i}{V} - \sin\frac{2n\pi t_{i-1}}{V}\right]\end{aligned} \tag{5.7}$$

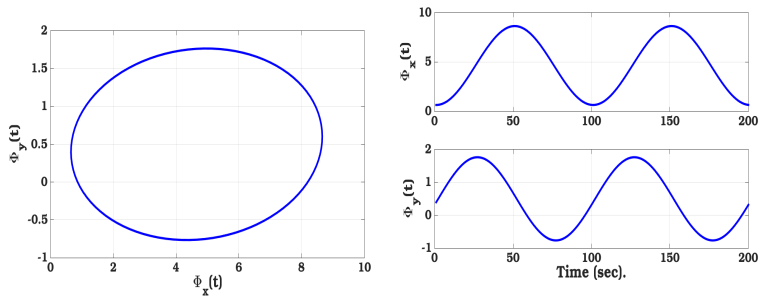
In most of the cases, the constriction is not symmetrical about (X,Y)-axis; hence, the horizontal and the vertical offsets of the contour can be computed as [212]

$$\begin{aligned}\alpha_0 &= \frac{1}{V}\sum_{i=1}^{k}\left[\frac{\Delta\mathbf{R}_x^i}{2\Delta t_i}(t_i^2 - t_{i-1}^2) + \zeta_i(t_i - t_{i-1})\right] + \mathbf{R}_x^0\\ \gamma_0 &= \frac{1}{V}\sum_{i=1}^{k}\left[\frac{\Delta\mathbf{R}_y^i}{2\Delta t_i}(t_i^2 - t_{i-1}^2) + \eta_i(t_i - t_{i-1})\right] + \mathbf{R}_y^0\end{aligned} \tag{5.8}$$

where $\zeta_i = \sum_{j=1}^{i-1}\Delta\mathbf{R}_x^j - \frac{\Delta\mathbf{R}_x^i}{\Delta t_i}\sum_{j=1}^{i-1}\Delta t_j$, $\eta_i = \sum_{j=1}^{i-1}\Delta\mathbf{R}_y^j - \frac{\Delta\mathbf{R}_y^j}{\Delta t_i}\sum_{j=1}^{i-1}\Delta t_j$, and $\zeta_1 = \eta_1 = 0$.

As per the outcomes of the above-mentioned process (Eqs. 5.6)-5.8), each set of four coefficients (α_n, β_n, γ_n, and δ_n) produce an ellipse with a certain orientation and starting point as defined by α_0, and γ_0, having an identical major and minor axis. The resultant contour ($\Phi(t)$) will be obtained by adding all of the computed coefficients (Eq. 5.6).

We reiterate that the basic notion behind the EFA procedure is that once the location coordinates of the sensed obstacles ($\tilde{R}(t)$) are identified, then the coefficients of EFA are evaluated (vide Eq. 5.6 for $n = 1$, i.e., the first harmonic sinusoids $\Phi_x(t)$ and $\Phi_y(t)$ based on the evaluated coefficients and

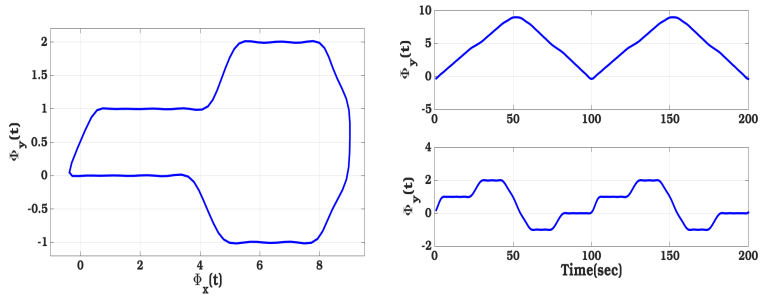


(a) Elliptical contour for $n = 1$. (b) Outcome of EFA scheme for $n = 1$.

Fig. 5.4 Outcomes of the EFA technique for the first harmonic, i.e. for $n = 1$.

are shown in Fig. 5.4b. Combining $\Phi_x(t)$ and $\Phi_y(t)$, a perfect elliptical contour will be produced (Fig. 5.4a).

Now, increasing the value of n, i.e., for higher harmonics $(n > 1)$, again a set of coefficients values could be obtained. Adding those coefficients as per Eq. 5.6 and combining $\Phi_x(t)$ and $\Phi_y(t)$, a gradually distorted virtual contour could be obtained. One such example has been shown in Fig. 5.5 for $n = 14$.



(a) Elliptical contour for $n = 14$. (b) Outcome of EFA scheme for $n = 14$.

Fig. 5.5 Outcomes of the EFA technique for the 14^{th} harmonics, i.e. for $n = 14$.

In this way, if we go on increasing the value of n, the EFA algorithm results in a virtual contour that matches the shape of the constriction as provided by the obstacles.

One such example can be seen in Fig. 5.6 where $n = 14$ and $k = 100$. 14 agents (indicated by blue circles) in the swarm detect the coordinates of obstacles (magenta dots) during a navigational step. If we apply the EFA procedure to these evaluated coordinate data, we can get an elliptical contour (red ellipse of Fig. 5.6) in the first harmonic. If we compute the 14^{th} harmonic of EFA coefficients because there are 14 agents in the swarm, we would obtain a green-marked arbitrary-shaped region, which perfectly identifies the free space across the environment [213].

Once the boundary has been created, all agents are required to approach the newly created virtual boundary region. This would lead to approximate path planning thereby forming a complete swarm-

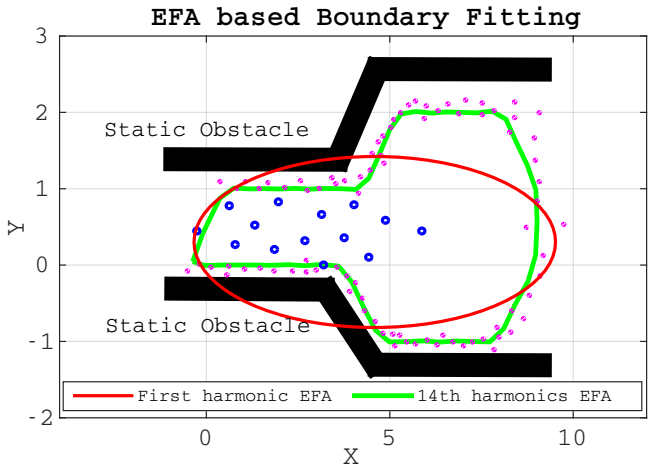


Fig. 5.6 EFA based virtual-boundary fitting: *blue*-circles represent the agents' present location, *magenta*-dots represent the detected obstacles' coordinates

like architecture. This proposition would also maximize inter-agent cohesiveness. To converge within the boundary ($\Phi(t)$), each agent will have to minimize its current position with respect to a point nearest to it and located on the boundary of $\Phi(t)$ continuously (vide the attraction term of Eq. 5.2).

Now, as per our proposed strategy, $\Phi(t)$ so evolved, is an arbitrarily shaped polygonal region. Hence, it is very difficult to express $\Phi(t)$ analytically. Thus, locating a point on the perimeter of $\Phi(t)$ for an agent poses a series problem.

Therefore, to tackle this challenge, we need to apply intelligent techniques such that all agents will move towards some specific points inside the contour for accomplishing a strong inter-agent cohesion. Consequently, we utilize the traditional circle-packing concept [206], [207] such that the virtual-region is fitted with N (i.e. the number of agents in the team) number of identical circles, each having a radius of r^s (i.e. repulsion region of each agent).

5.4.3 Packing circles inside the virtual region

While packing circles inside the virtual contour, it should be noted that each circle should be inside the virtual region such that no two circles would intersect with each other and the maximum area should be covered by the fitted circles. Moreover, out of many well-known circles packing methods [206], [207], the lattice-packing [214] technique is one of the robust techniques which allows a dense packing of circles with less execution time. Hence in this work, we have employed this technique.

Preliminaries of Lattice packing scheme

Let us assume that $\vec{U} = \{\vec{u}_1, \vec{u}_2, ..., \vec{u}_r\}$ is a set of r independent vectors in an r-dimensional space E^r and $\mathbf{M}$, the generator matrix, is an $r \times r$ matrix formed by the r vectors. Then for any integer vector $\vec{x} = (x_1, x_2, ..., x_r)^T$, $\mathbf{M}\vec{x}$ is called the lattice point in E^r [207]. Therefore, the set of integer vector $\vec{x}$ will form a lattice structure L_U.

For each lattice L_U, there will be a polyhedron B in E^r which is formed by the set of vertices, $[v_1, v_2, ..., v_r]$, such that $v_i = \mathbf{M}\vec{e}_i$ for $1 \leq i \leq r$. So, B is called the basis block of L_U which would form a tile in E^r. In 2D space, the basic block is a parallelogram which is also called as the basis parallelogram [207]. The necessary condition behind fitting identical circles is that each lattice L_U permits packing of circles having radius r^s if the Euclidean distance between any pair of lattice point is greater than or equal to $2r^s$.

Once the set of lattices have been formed, our next target is to fit circles inside the outline. To do so, the *translational* based *lattice packing* technique [206] has been utilized as discussed below.

Two dimensional Lattice packing with Translation

For packing circles having radius r^s inside the arbitrary boundary Φ, we assume that the entire domain Φ is treated as a single cell [214].

Let us assume that Φ is an n-vertices polygon, and $U = \{\vec{u}_1, \vec{u}_2\}$ is the basis of the lattice L_U on the plane P such that L_U allows packing of a set of S number of identical circles, such that $S = [(S_1, S_2, ..., S_R) \mid R \geq N]$ having radius r^s. Next, we need to identify a set of N number of circles s, such that $s \subset S$ and $n(s) = N \leq R$ (where $n(s)$ defines the number of elements in s), which are completely lain within the boundary region Φ. In order to achieve this, we compute the Minkowski difference (MD) [207] which can analytically be defined as

$$\Phi_M^q = (\Phi \ominus S_q) \quad \forall q \in [1, 2, ..., R] \tag{5.9}$$

where $\ominus$ is the MD operator [207], Φ_M^q is the region which decides whether a circle S_q is within the boundary region Φ. S_q is accommodated inside Φ, i.e. $S_q \subset s$, if and only if $\Phi_M^q = \emptyset$.

While experimenting with the translational lattice packing technique, it is often observed that when the boundary region is narrow with-respect-to the N number of agents in the team, this method may not generate N number of identical circles inside Φ.

If we consider Fig. 5.7(a), we would observe that inside the arbitrarily-shaped boundary region (Φ) employing the two-dimensional lattice packing technique, six identical circles could be fitted. However, if the number of agents (N) is more than six (assuming N=8), then, the lattice packing technique is unable to pack N numbers of circular regions. In such kinds of scenarios, an updated method is required to be used such that at least N numbers of circles would be fitted within Φ. To achieve this objective, in this chapter, we use a new scheme, called the trimming and packing technique. This specific technique is only executed inside the global controller during the navigational steps if the lattice packing strategy fails to generate the N number of circles. The detailed methodology of the proposed approach is discussed below.

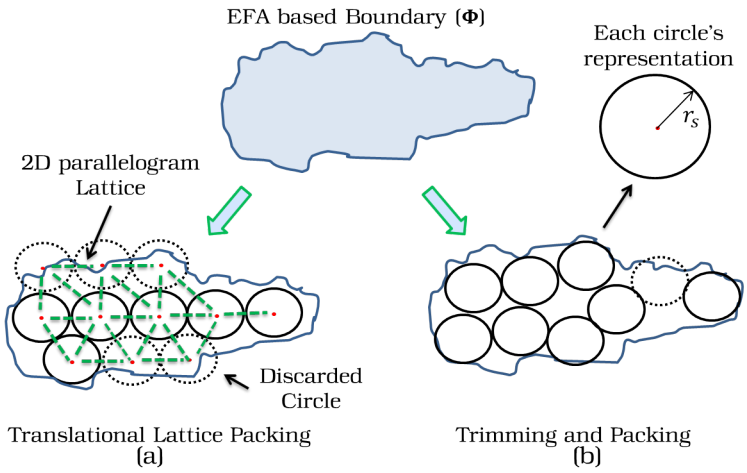


Fig. 5.7 Packing circles inside the virtual boundary (Φ) with different techniques: (a) Translation Lattice packing can fit 6 circles (b) Trimming and Packing can fit 8 circles.

Trimming and Packing technique

Considering the outcome of the lattice packing technique in the preceding section (Fig. 5.7(a)), we have seen that the circles are mostly packed in the middle of the boundary region Φ. Hence, there exist some circles which cannot be wholly fitted within the boundary of Φ (as presented in Fig. 5.7(a)). Therefore, there exists some *unpackable* free spaces scattered through the boundary of Φ.

To utilize those small areas in an intelligent way, the trimming technique has been used which tries to push the circles from the middle of Φ towards the boundary and thus, attempts to create some free spaces in the central areas for packing more circles. In the next step, the virtual triangles and trapeziums are fitted in those free spaces in order to constitute a lattice pattern [215].

Therefore, as per the trimming procedure, initially, the arbitrary-shaped region Φ is partitioned into a set of triangles or trapezoids. In Fig. 5.7(b), we can see that a good number of such triangles and/or trapezoids are engraved within the region. After that, to obtain at least N number of sub-regions, it may be necessary to merge some of the neighboring triangles or trapezoids (if the numbers of such virtual sub-regions created so far are greater than the number of agents N).

Before trimming, already we have seen that a good number of circles are initially fitted (Fig. 5.7(a)), and then pushed towards the boundary region. Hence, if we can further fit in at least N more number of circles, then we will be in a safe position to claim that N plus number of circles have been fitted in the entire virtual region Φ. Subsequently, in those N number of sub-regions, N number of circles would be packed and agents will then be pushed to the centers of those N number of circles having a radius of r^s.

Hence, the algorithmic steps of the trimming technique could be defined as:

1. The *unpackable* free spaces of Φ is initially partitioned into N number of cells (by forming virtual triangles and trapeziums). The set of cells defines a dual-graph G_N such that each

vertex v of G_N is for a cell and an edge connects two vertices if the corresponding cells share a common overlapping area.

2. The process repeats until $G_N = \emptyset$.

Once, the above-mentioned trimming algorithm creates N number of cells within the free spaces of Φ, the packing technique (as mentioned in the earlier section) will fit circles in each of the cells.

The main advantage of the trimming and packing procedure is that as the packable cells are always connected, the unused areas are less as compared to the two-dimensional lattice packing strategy.

One such outcome of the trimming and packing technique has been shown in Fig. 5.7(b). From the figure, we have seen that the two-dimensional lattice packing solution could manage to pack six circles within Φ, whereas, the trimming and packing scheme can pack eight circles.

After the execution of this algorithm, the virtual region (Φ) is packed with N number of identical circles (each with a radius of r^s). Let us assume that the set of circular-centers are $c_s = \{c_s^1, c_s^2, ..., c_s^N\}$. After evaluating the centers of the circular regions, each agent needs a circle to be allocated such that all the agents can efficiently approach their allotted circles without blocking the path of another agent. To handle this specific condition, we have implemented a circle assignment strategy for the i^{th} agent, which is described as

$$c_s^i = \{c \mid \underset{c \in c_s}{\operatorname{argmax}} ||x_i(t) - c\} \quad \forall i \in \{1, 2, ..., N\} \tag{5.10}$$

Once the circles' are allotted, our next target is to actuate the N number of agents towards the N number of circles' by firing the local controller of each agent in the swarm.

5.5 Convergence of agents inside the circular region

The local controller will be activated to enforce each and every agent to be inside Φ. Subsequently, each agent will approach towards the assigned circle's center. Therefore, the *attraction*-term of the local control action of the i^{th} agent can be modified as (from Eq. (5.2))

$$f_i(x_i) = -k_a \times (x_i - c_s^i) + k_r \times \sum_{j=1, j \neq i}^{N} \left[\exp\left(-\frac{x_{ij}}{2 \times r^s} \right) \times (x_i - x_j) \right] \tag{5.11}$$

To summarize the proposed two-level control action in a swarm robotics path planning problem, we can state that in an occluded environment, each agent can successfully approach the target while maintaining strict cohesiveness with the other agents in the group. The next section represents the effectiveness of the complete control strategy. In this aspect, we would like to mention that all of the simulations are performed in the same software-hardware environment as mentioned in Sec. 3.7.

5.6 Simulation results and discussions

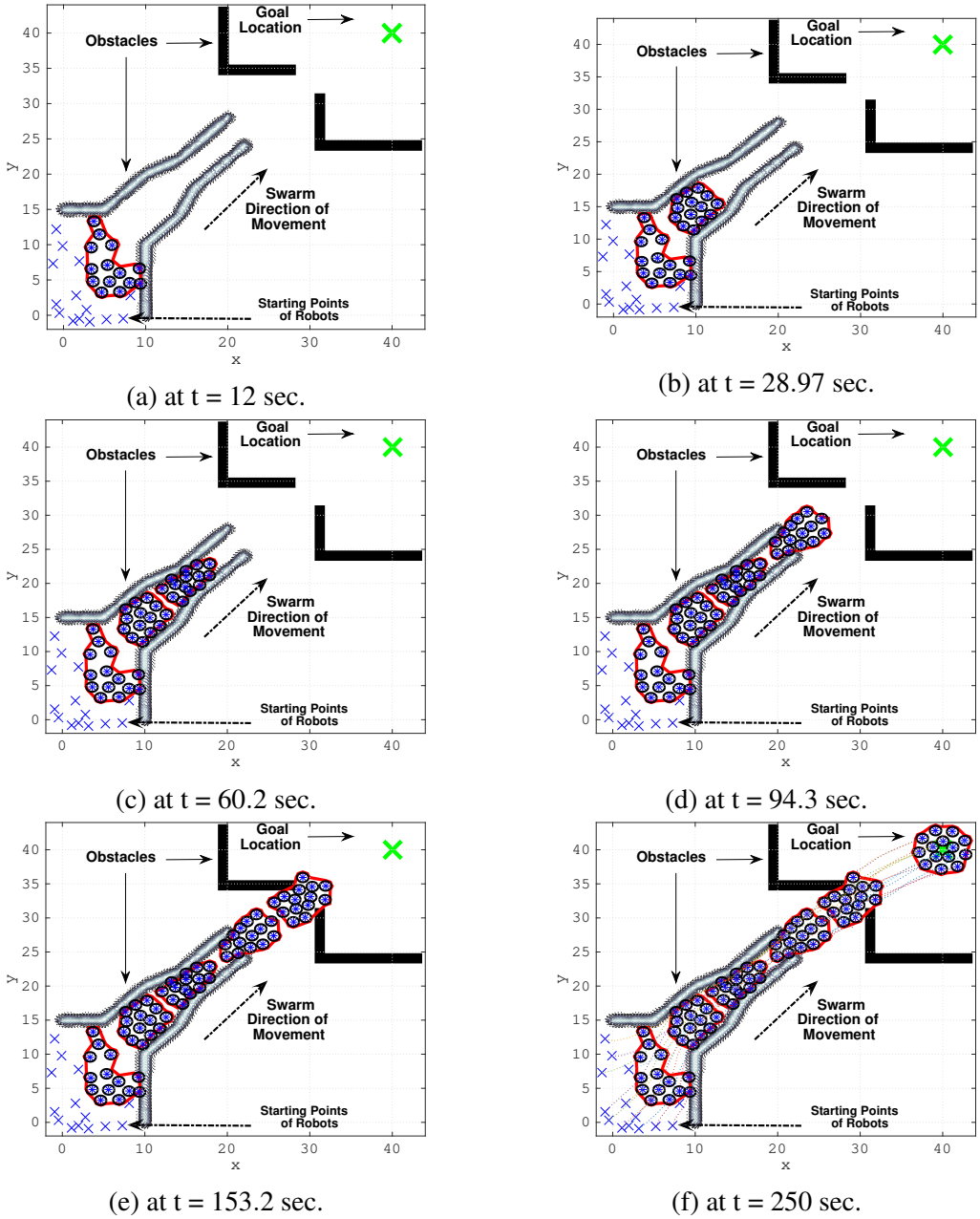
This segment presents the simulation result to demonstrate the performance of the proposed controllers in an obstructed environment which consists of several regular and irregular-shaped obstacles having narrow corridors as shown in Fig. 5.8. A team of 14 robots initially placed in arbitrary scattered positions (marked by *blue*-cross) are required to form a swarm-like framework and approach towards the target T while circumventing barriers. In this simulation study, the following parametric values have been chosen (Table 5.1).

Table 5.1 Values of the Parameters

Parameter	Description	Value
N	No. of agents	14
Ω	Topological Space	[50 unit, 50 unit]
T	Target Location	[35 unit, 35 unit]
k_a	Attraction Gain	22.4
k_r	Repulsion Gain	190.45
r^s	Repulsion Region	1 unit
m	No. of Distance Sensors	8
z	Sensor's Spacing	45^0
k	No. of Points	100
n	Fourier Harmonics	14
d_{thre_+}	Upper Threshold Limit	5 unit
d_{thre_-}	Lower Threshold Limit	0.5 unit

Primarily, based on the agents' sensing information, an arbitrarily shaped virtual boundary (marked as *red*-contour) has been created under the activation of the EFA-guided virtual boundary fitting controller (i.e. the global controller), followed by N number of packed-circles (marked by *black*-circles) inside the virtual region as shown in Fig. 5.8a. Then, the circles are allotted for each agent by the global controller employing the circle assignment scheme. After assigning the circles for all of the agents, the local controller will be activated in each agent for actuating the agent towards its assigned circular region. At approximately $t = 12$ sec (Fig. 5.8a), all of the 14 agents have successfully arrived to their assigned circles (marked by *blue*-star). Then, each agent starts sensing the nearby obstacles and shares the information with the global controller. A compact contour is created (as the free space ahead is narrower) followed by 14 assigned circles so that all agents can form a swarm (presented in Fig. 5.8b). Once they arrive at the assigned circular zone at $t = 29$ sec, the free space ahead gets more slender. Therefore, in order to preserve the swarm, an adaptive ellipsoidal-based contour is created (shown in Fig. 5.8c). In this situation, since the translational lattice packing technique fails to pack 14 circles inside the stipulated region, hence, the trimming and packing method has been initiated to fit 14 circles with optimal circumferences inside the virtual region, followed by the circle-assignment strategy.

At $t = 60.2$ sec, all the agents have successfully arrived inside the allocated circles (Fig. 5.8c). After that, again a newly shaped arbitrary region is formed ahead with 14 packed circles (represented in Fig. 5.8d). After arriving at the stipulated circular regions, the swarm has successfully avoided



(a) at t = 12 sec.

(b) at t = 28.97 sec.

(c) at t = 60.2 sec.

(d) at t = 94.3 sec.

(e) at t = 153.2 sec.

(f) at t = 250 sec.

Fig. 5.8 Virtual Region based swarm-agent position trajectories in different instances in an occluded environment: blue-crosses represent agents' initial position, blue-star represents agents' current location, and green-cross represents the target location.

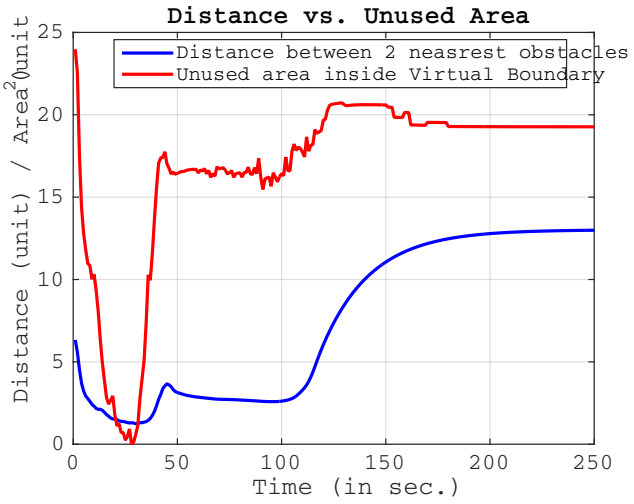


Fig. 5.9 Time vs. Distance between two-nearest sensed obstacles vs. the unused area inside the virtual boundary.

the irregularly-shaped obstructions at $t = 94$ sec. Now, the path of the swarm has been blocked by a pair of regular-shaped snags (*black*-box as shown in Fig. 5.8e). In order to evade the same, a virtual contour along with the fitted circles has been generated and all the agents reach inside the region at $t = 153.2$ sec. Finally, when all the obstacles are successfully avoided, the virtual region again reshapes into a circular contour with the packed circles. All the agents effectively reach to the target location T at $t = 250$ sec (referred to the Fig. 5.8f).

To study the performance of the proposed strategy as compared to the other schemes, the analyses and comparisons are provided in the upcoming sections.

5.6.1 Time vs. distance and Unused area

To measure the optimized circles' fitting inside $\Phi(t)$, we have calculated the unused area (denoted by $UA_\Phi(t)$) inside the boundary ($\Phi(t)$) after packing the circles at the t^{th} time which can analytically be evaluated as

$$UA_\Phi(t) = A_\Phi(t) - N \times \pi r_s^2 \tag{5.12}$$

where $A_\Phi(t)$ is the present area of the evaluated contour $\Phi(t)$. At the same time, the distance between two nearest sensed obstacles are measured (from Eq. 5.5). Let us assume that $b_{fs}(t)$ is the Euclidean distance between two nearest obstacles $r_1(t)$ and $r_2(t)$ at the t^{th} time. The corresponding nearest obstacles distance ($b_{fs}(t)$) vs. the unused area ($UA_\Phi(t)$) plot is represented in Fig. 5.9.

From the Fig. 5.9, it can be argued that when the distance between two-nearest obstacles is less i.e. the free space ahead is very narrow, the virtual boundary is fitted in such a manner that the maximum N number of circles can be packed inside it which will subsequently reduce the unused area. In our simulation result, this argument can be clearly observed i.e. when the distance between two-nearest

obstacles is minimum (the minimum point of the *blue*-curve), the effective unused area ($UA_{\Phi}(t)$) inside $\Phi(t)$ attains a minimum value which would significantly increase the inter-agent cohesiveness.

5.6.2 Comparative analysis

In this section, we compare our proposed approach in terms of the agents' cohesiveness and execution time during the movement towards the goal against the two other well-known techniques: [200] relies on the global pose information of every agent, and [96] uses a decentralized controller that does not require the global positioning information of an agent.

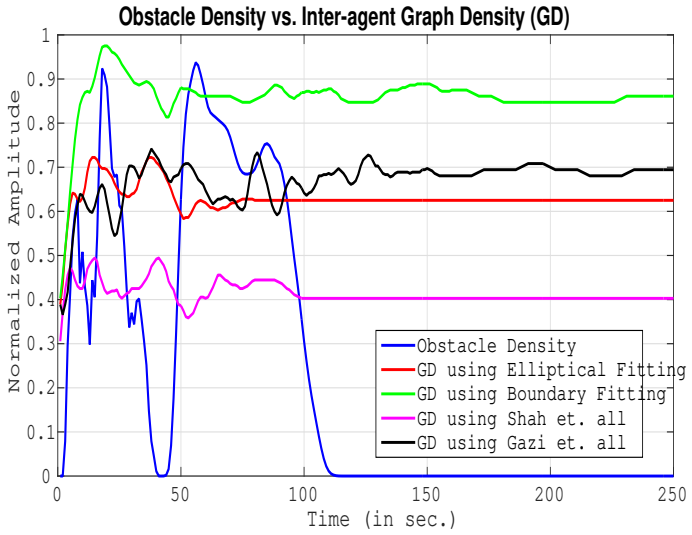


Fig. 5.10 Comparison of Inter-agent graph density vs. normalized obstacle density

To evaluate the cohesion inside a virtual boundary, we measure the *structural density* [216] of the swarm inside the contour. We assume that N agents are forming a graph (G_{N_s}) where the vertex set is equal to the number of agents in the team. Moreover, two agents are assumed to be interconnected if the distance between them is greater than or equal to $2r^s$ and less than or equal to $4r^s$ i.e.

$$e_{ij} = \begin{cases} 1, & \text{if } 4r^s \geq x_{ij} > 2r^s \\ 0, & \text{otherwise} \end{cases} \tag{5.13}$$

where r^s is the region of repulsion of each agent. Then the structural density ($D(t)$) of the graph G_{N_s} at the t^{th} instance can be defined as the ratio of the number of present links to the number of possible links after discarding the self-loops [182], i.e.

$$D(t) = \frac{\sum_{i=1}^{N} \left[\sum_{j=1}^{N} e_{ij}(t) \right]}{N(N-1)} \tag{5.14}$$

At the same time, the normalized obstacles' density is calculated based on the agents' sensing information. The entire plot is represented in Fig. 5.10.

From the Figure, it is observed that when the obstacle density is higher i.e. all the agents' are fully surrounded by the barriers, the graph density or the structural density attains a maximum value in the proposed strategy, which signifies strong inter-agent connection and thus maximizes the agents' cohesiveness. In this specific situation, all agents are strictly connected with each other for maximizing inter-agent cohesion. Comparing with the other techniques, the lowest inter-agent cohesion is observed when there is no internal communication among the agents [96]. For the elliptical-based fitting technique (as mentioned in Chapter 4), the cohesiveness is reduced because of the structural flexibility of the elliptical structure.

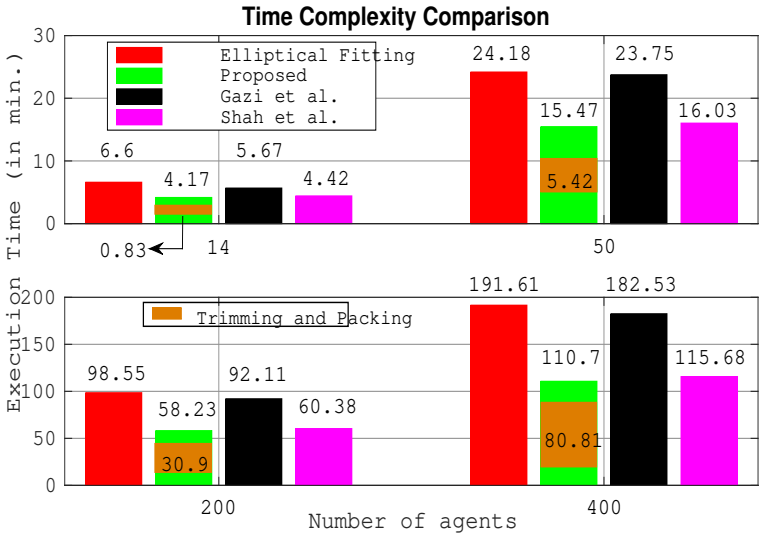


Fig. 5.11 Time Complexity Comparison

While comparing the execution time (as depicted in the Fig. 5.11) with the varying number of agents in the team, it is observed that our proposed method takes the least time to actuate a team of 14 robots towards the target in spite of planning several activities during the journey. The scalability analysis is further carried out considering 50, 200, and 400 agents in the team. As expected, it is seen that with the increasing number of agents in the team the execution time for the trimming is increased significantly. However, in most cases, the trimming strategy provides better packing density but suffers from a large execution time. In contrast, the translational-based technique cannot produce such a quality trimming and as a result, the algorithm runs faster. Hence, we have fused these two methods to reap the benefits of both.

5.7 Summary

In this chapter, we have proposed an arbitrary-shape control technique for avoiding obstacles in an unknown challenging environment. In order to evade the obstacles, more specifically from the

narrow pathways and corners, arbitrarily shaped virtual-region followed by a number of packed circles are produced which result in a strong inter-agent cohesiveness among the agents for achieving a swarm-like architecture. In these regards, we have proposed a two-level hierarchical control scheme. To illustrate the performance of the proposed controllers, simulation results have been presented. The comparison studies verify the effectiveness of the proposed approach.

Chapter 6

Exploration of Multiple Unknown Areas using Splitting and Merging Technique

6.1 Introduction

In all our previous works (as described in Chapters 3, 4, and 5), we have specifically assumed that there exists a single path through which the swarm can progress towards the target. However, a real environment may not be that simple; it may contain several pathways through which the group may travel during the navigational process. Hence, for successful and complete exploration of the environment, the parent-swarm of robots requires to be split into several sub-swarms [217]. Moreover, during navigation, if there is only a single path available ahead through which a few sub-groups can travel further at the same time, under such situation, it would not be wise enough to segregate the agents in the groups but to combine them so that they can form a stronger and energy efficient super-group. Hence, in this circumstance, we need to conglomerate the agents belonging to the sub-groups, previously traversing different paths into a coherent swarm that would walk through the single pathway, thus approachable ahead of the joint where the multiple tracks met. We believe that this perception represents the complicated labyrinth-like plan [226] of most of the cities.

To address the multi-path navigation issue, the present chapter adopts the elliptical region-based shape control strategy (vide Chapter 4) where each agent in the swarm is required to converge inside an elliptical region without having any collision. Additionally, the process utilizes the traditional leader-follower scheme [111], [218], where the function of the leader robot is to fit in virtual regions during the navigational steps using the sensing information from the follower robots. However, when multiple pathways are detected to proceed further for justifying a purpose (like multiple targets) during navigation, then a decision has to be taken by the leader to explore all the paths by splitting the parent swarm. To accomplish this decision-making task, the leader robot shares the sensing information with a dedicated centralized controller. In this work, the function of the central controller is only to create

sub-regions. The trajectory planning is solely performed by each of the agents in the swarm based on the created sub-regions. Thus, the main purposes of adopting the centralized controller are

1. Creating sub-swarms by fitting in multiple elliptical virtual regions along each path.
2. Selecting agents for each of the sub-groups.
3. Assigning new leader robots (for each of the sub-swarms).

Once the leaders are specified, the remaining process would be carried out by them along each path like a single-path solution. Additionally, it might so happen that after a while, two or more paths merge, forming a single passage. In that situation, the proposed controller will guide the sub-swarms routed through those converging paths to merge or rejoin. This specific approach shall conserve all the swarms (full of their energies) without letting them get lost and subsequently remain unused. One such scenario is shown in Fig. 6.1.

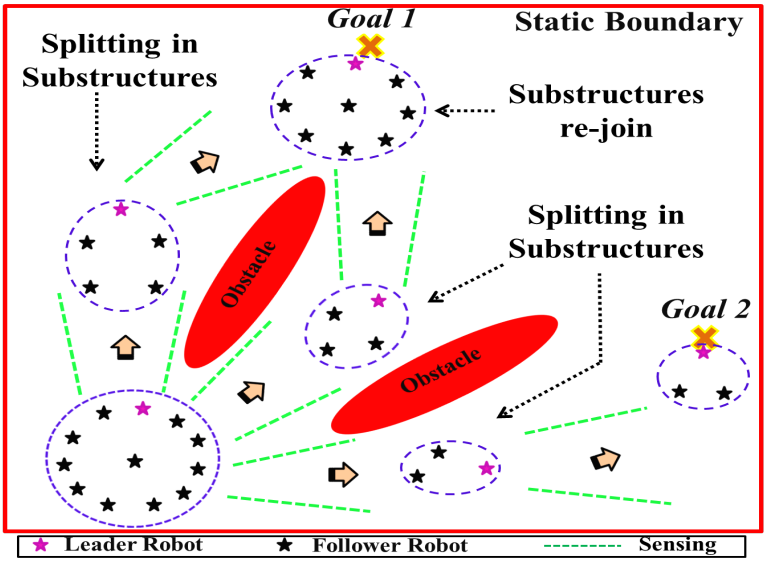


Fig. 6.1 Splitting and merging of a swarm during navigation.

From Fig. 6.1, it is seen that a team of 12 agents is exploring an unknown environment while preserving a specific formation within a virtual region. The magenta and black stars represent the leader and follower agents respectively, and the virtual regions are highlighted in violet contours. Suddenly, during the exploration phase, the leader agent detects three possible pathways to navigate further. In such kind of scenario, the parent swarm will be split into three sub-swarms (each with a leader agent), as illustrated in the figure, and the process continues. Then, after a while, two paths converge to form a single corridor through which the sub-swarms (operating in these paths) are required to be headed. As a consequence, the sub-swarms navigating through those paths are merged to create a super-swarm that effectively approaches the target while the third sub-swarm alone approaches the second target.

For achieving the above-mentioned objectives i.e. splitting and merging of a swarm of robots during the navigational phase, we employ the region-based shape control architecture (referred to Chapter 4). In the upcoming section, we recapitulate the previous control and sensing mechanisms briefly.

6.2 Background

The region-based shape control scheme (Chapter 4) consists of two dedicated controllers, namely global and local controllers. To produce a virtual elliptical contour, each agent in the swarm is required to sense the surrounding environment first. Based on this sensing information, the global controller adjusts the parameters of the elliptical shape. Depending on the geometry information of the elliptical zone, agents are actuating their local controllers to accommodate themselves inside it. The detailed sensing-actuation framework is discussed below.

6.2.1 Sensing

We assume that N numbers of robotic agents are operating in an unknown environment. Each agent in the team was equipped with m number of distance sensors [195], [194], with a spacing of $2\pi/m$ radian apart from each other. At the t^{th} time, the distance of the sensed obstacles $\mathbf{d^i(t)}$ around the i^{th} agent would be

$$\mathbf{d^i(t)} = \{\mathbf{d^i(t)}\} \quad \text{scuh that} \quad d_{th_+} \geq \mathbf{d^i(t)} \geq d_{th_-} \tag{6.1}$$

where $\mathbf{d^i(t)} = \begin{bmatrix} d_1^i & d_2^i & \dots & d_m^i \end{bmatrix}_t$ is the distance-matrix of the m sensors, d_{th_+} and d_{th_-} are the sensor specific threshold limits. Based on those information, the location of the sensed obstacles $\tilde{\mathbf{R}}^i(t)$ with-respect-to the world coordinate frame of reference could be predicted as (for the i^{th} agent at the t^{th} time)

$$\begin{bmatrix} \tilde{R}_i^x(t) \\ \tilde{R}_i^y(t) \end{bmatrix} = \begin{bmatrix} p_i^x(t) \\ p_i^y(t) \end{bmatrix} + \sum_{j=1}^{m} \left[d_j^i(t) \times \begin{bmatrix} \cos[(j-1)z + \theta^i(t)] \\ \sin[(j-1)z + \theta^i(t)] \end{bmatrix} \right] \tag{6.2}$$

where, $x_i(t) = [p_i^x(t) \quad p_i^y(t)]^T$ is the present location of the i^{th} agent, $\theta^i(t)$ is the orientation of that agent with respect to the global coordinate frame of reference at the t^{th} time, and $z = 2\pi/m$.

The basic philosophy behind the sensing scheme is the fact that all the agents would individually estimate the locations of nearby obstacles and would pass on the data to the global controller. Upon receiving this information from all the agents, the global controller would estimate the free space ahead and fit in a virtual elliptical region to accommodate all the agents inside it.

6.2.2 Modeling of Global Controller

For defining the global controller, again we assumed that after consolidating the predicted locations of the nearby obstacles from all the agents in the team, the controller would create a matrix $\tilde{\mathbf{R}}(t) =$

$[\tilde{R}^x(t) \quad \tilde{R}^y(t)]^T$ in the t^{th} time as

$$\tilde{\mathbf{R}}^x(t) = \begin{bmatrix} \tilde{\mathbf{R}}^x_1(t) & \tilde{\mathbf{R}}^x_2(t) & \ldots & \tilde{\mathbf{R}}^x_N(t) \end{bmatrix} \tag{6.3}$$

$$\tilde{\mathbf{R}}^y(t) = \begin{bmatrix} \tilde{\mathbf{R}}^y_1(t) & \tilde{\mathbf{R}}^y_2(t) & \ldots & \tilde{\mathbf{R}}^y_N(t) \end{bmatrix} \tag{6.4}$$

Depending on $\tilde{\mathbf{R}}(t)$, the global controller estimates the free-space ($b_{fs}(t)$) ahead. The detailed procedure for determining the free-space estimation can be found in Chapter 4. After evaluating the free-space ($b_{fs}(t)$), a virtual-elliptical contour having a center at $O_e(t)$ is duly fitted inside that region, and this information is shared with each agent in the swarm for activating the local controller.

6.2.3 Modeling of Local controller

Upon receiving the information regarding the center of the virtual region ($O_e(t)$), the local controller of each agent would be triggered. Specific control action of the i^{th} agent at the t^{th} time is derived as

$$\dot{x}_i = -k_c \times (x_i(t) - O_e(t)) + k_f \sum_{j=1, j\neq i}^{N} \left[\exp\left(-\frac{||x_i(t) - r(t)||}{r^s} \right) \times (x_i(t) - r(t)) \right] \tag{6.5}$$

where, k_c and k_f are the associated attraction and repulsion coefficients, $r(t)$ is the location of the closest agent or obstacle (which one is nearest), and r^s is the minimum distance (i.e. repulsion region) that are required to be maintained by the i^{th} agent from $r(t)$.

Thus, under the actuation of the local controller, the i^{th} agent would approach towards $O_e(t)$ while avoiding collision from the neighboring agents and obstacles as defined by the term $r(t)$.

This control schema, as reported by us, worked well when the environment contains a solitary path, leading towards the target (as discussed in Chapter 4). However, in reality, there might have several pathways and/or multiple targets. To solve these challenges, the swarm needs to be broken down into separate subgroups. In other words, while exploring the unknown environment, if the swarm detects more than one path to travel further, then it would split into sub-swarms adaptively. The present work is, actually, an extension of our previous work (as described in Chapter 4). The present considerations alleviate the real-time restriction put on the notion of a single path scenario that may not be truly realistic.

In the proposed work, we generalize and endeavor to conceive a more convincing picture, in the context of tasks to be performed during surveillance and rescue operations, which often call for aiming at multiple targets/victims, following different tracks. Further, in this work we have kept the provision for the merger of sub-swarms, in case, two or more paths lead to the same target. Thus, they can eventually merge to form a larger set of sub-swarms to ensure the life and conservation of energy of each agent as well as the collective swarm. We reiterate that in this proposition we attempt to make our control strategy to be scalable, to be able to make or break the swarms/sub-swarms, and not to lose any agent under any circumstances. Our approach is to have all the N numbers of agents fully

active, either as a member of a sub-team or as a member of the original team. To substantiate this fact, we reiterate that unless some hardware failure deactivates an agent, we would always try to maintain the number of active agents participating in the exploration process as declared during the initiation of the mission.

Before moving on to the detailed procedures, in this work, we have explicitly considered the following assumptions.

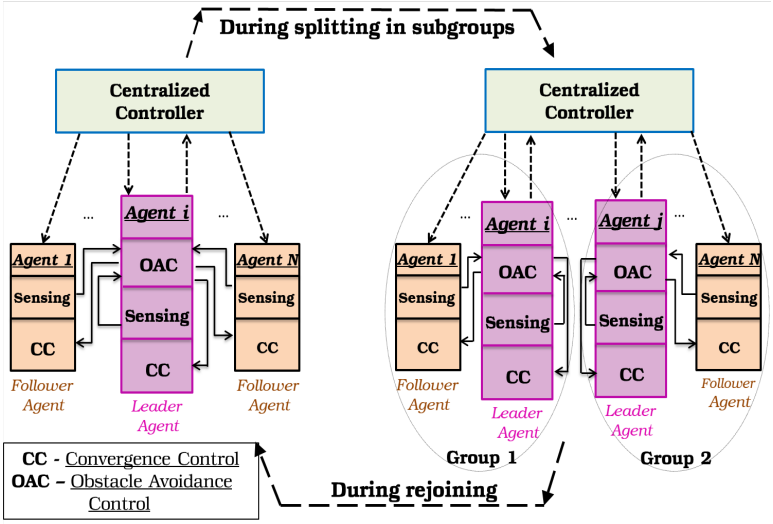


Fig. 6.2 Block diagram representation of the proposed split-merge system.

Assumption 1: In this chapter, we have relabeled the global and local control actions (as stated previously) as the *Obstacle Avoidance* Controller (OAC) and the *Convergence* Controller (CC) respectively. All the agents in the swarm are capable of executing both the control actions (if needed) inside their processing unit.

Assumption 2: At the beginning of the navigational procedure, the user would select an arbitrary agent as the *leader* robot, in which both the Obstacle Avoidance and Convergence Control actions would be executed. The remaining agents are considered as the *follower* robots, where the Convergence Control action is only operating and the OAC is in the idle mode (as shown in Fig. 6.2).

The block diagram of the proposed control action is presented in Fig. 6.2. The OAC and CC are both running inside a leader agent, whereas CC is only executing inside a follower agent, as shown in the diagram. During the navigational phase, OAC is in charge of adjusting the parameters of the virtual elliptical region, while CC is responsible for actuating an agent inside the virtual contour.

During splitting, a number of sub-swarms will be produced from the parent swarm under the influence of a centralized controller. Subsequently, a leading agent for each of the sub-swarm will be assigned for assisting the swarm in path planning.

During the rejoining phase, two or more sub-swarms will be merged to create a super-swarm. In this case, a leader agent will be allotted for the super-swarm for guiding the swarm to navigate.

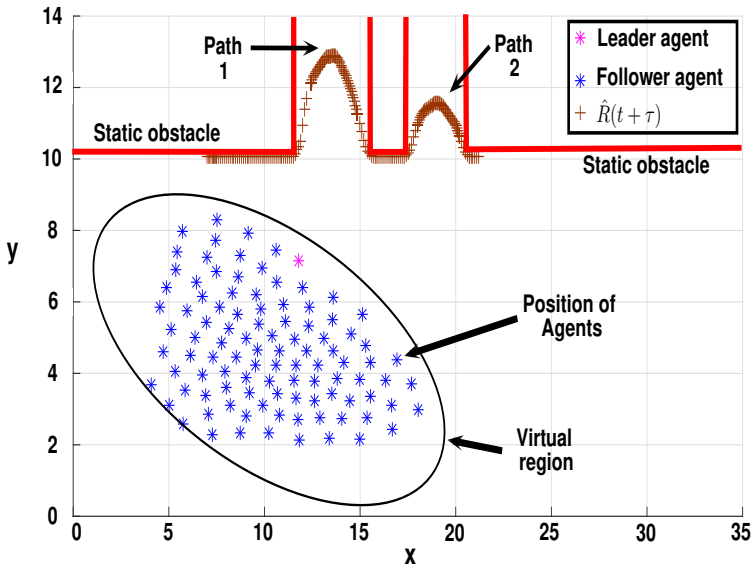


Fig. 6.3 A small section of the environment having multiple pathways.

Therefore, as per the navigational plan, initially, the agents, which are acting as followers, sense and evaluate the nearby obstacles and share this information with the leader of the team. Upon receiving this from all the agents, the leader agent actuates the OAC to estimate the next coordinates of the virtual structure (as discussed in Chapter 4) and broadcasts this message to the followers for actuating the CC. In the presence of multiple paths, this methodology may not work; hence, the control laws are required to be updated such that sub-swarms can be formed from the parent swarm (i.e. for splitting to occur). To create sub-swarms, we need to consider the following aspects: 1) detecting the tracks, 2) forming sub-regions for each of the tracks by fitting in a virtual region in each of the tracks, 3) identifying the agents for each subgroup, 4) promoting leader for each sub-swarm (vide Figs. 6.1 and 6.2 respectively).

These aspects are addressed in the following section.

6.3 Creation of Sub-swarm

To create sub-swarms, first of all, the leader should identify (from $\tilde{\mathbf{R}}(t)$ (Eqs. 6.3, 6.4)) whether there are multiple paths ahead or not. Sub-swarms will only be created if multiple pathways are detected, otherwise, the process remains the same as stated in Chapter 4.

Let us assume a scenario that at the $(t+\tau)^{th}$ time, there exist multiple paths ahead and it is within the sensing range (d_{th_+}) of most of the agents (as shown in Fig. 6.3). Incorporating the sensing information from all the followers, the leader computes the location coordinates of the nearby obstacles ($\tilde{\mathbf{R}}(t+\tau) \in \Re^2$) (refer to Eqs. 6.2, 6.3, and 6.4). The significance of $\tilde{\mathbf{R}}(t+\tau)$ can be explained in a lucid form with the help of a simple analogy exhibited by the human brain. If an

unobstructed path that could be taken by a person is to be decided by him or her, the eyes of that person sense the void area lying ahead along the track, thus, acting as sensors, and to be precise the clearance of the passage is also roughly estimated by the eyes. Thus, in this work, the location coordinates of the nearby obstacles ($\tilde{\mathbf{R}}(t+\tau)$) actually indicate the void space ahead.

Our eyes (positioned somewhat distantly from the open space) with the help of our brain do so by actually calculating and joining the $x-y$ coordinates of the points lying on the gateway of the track. Thus we can conceive that a contour is actually carved which represents the shape of the doorway to the free space. The depth of field (in the case of a multi-path scenario) designates the overall clearance of the path. In Fig. 6.3, we have simulated this scenario with the help of distance sensors mounted on the agents only. In this context, we specifically mention that for identifying the presence of multiple paths, we have not used any cameras.

In Fig. 6.3, we have portrayed the fact that each agent in the swarm senses the presence of a void space ahead ($\tilde{\mathbf{R}}^i(t+\tau)); \forall i \in [1,2,...,N]$. The x and y coordinates (i.e., $\left[\tilde{\mathbf{R}}_i^x(t+\tau)), \tilde{\mathbf{R}}_i^y(t+\tau))\right]$ of Eqs. 6.3 and 6.4) designate the location of clear space. The entire x vs. y values of $\tilde{\mathbf{R}}(t+\tau)$ along the void space have been plotted as a brown curve in Fig. 6.3 having a pronounced peak, which determines the maximum average distance of the clearance as estimated by all the agents. Likewise, we find that the environment does have a second path. The same procedure has been carried out to identify it to be yet another path. The value of the peak, in this case, is less. The rationale behind this fact is that the swarm is oriented (clear from the diagram) in such a manner that the average number of agents is comparatively far away from that passageway. And hence the average maximum in y-values is less than the previous one. It is needless to mention that, in general, the brown contours will assume shapes and peaks depending on the orientation of the virtual region (encircling the agents) with respect to the orientations of the paths. Moreover, it is apparent from the above discussion that whenever the leader detects multiple peaks (as depicted in Fig. 6.4) during the navigation, it will come to a conclusion that multiple lanes are available (the number being equal to the number of peaks detected) to continue further. Once, the multi-paths have been detected, some necessary actions are expected to be performed by the swarm system sequentially and those steps can heuristically be enumerated as

1. Depending on the number of tracks, the free spaces along the paths are required to be estimated first from $\tilde{\mathbf{R}}(t+\tau)$.

2. In each estimated free space, a virtual elliptical contour is required to be fit in.

3. Based on the effective areas of the virtual regions, the agents are required to be clustered proportionally into several sub-groups, thus constituting the sub-swarms.

4. For each of the sub-swarm, a corresponding leader agent needs to be promoted for advancing further.

To process the above-mentioned operations, a high computing processor would be required that may not available in any of the robotic units. So, when the leader detects multiple paths ahead, the

corresponding sensing information $\tilde{\mathbf{R}}(t+\tau)$ is passed on to a specialized high computing centralized controller from the leader agent for splitting the swarm.

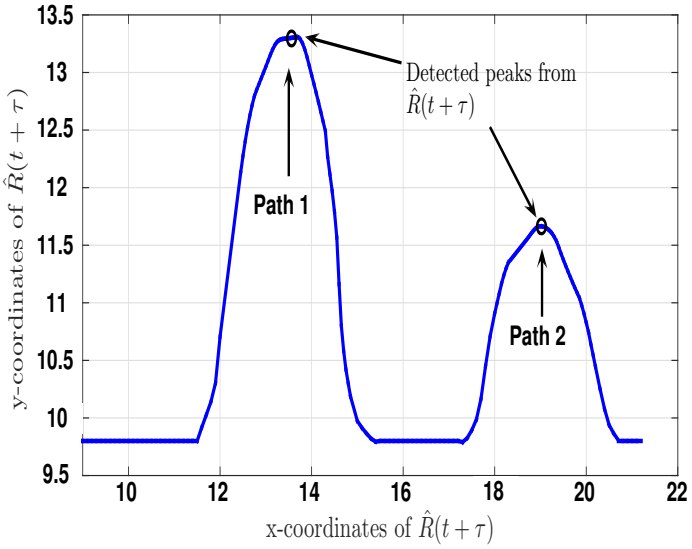


Fig. 6.4 The sensing coordinates $\tilde{R}(t+\tau)$ plot with detected peaks.

6.3.1 Estimation of free-spaces

To detect the number of tracks from the coordinates of sensed obstacles $\tilde{\mathbf{R}}(t+\tau) = [\tilde{\mathbf{R}}^x(t+\tau), \tilde{\mathbf{R}}^y(t+\tau)]$, the centralized controller employs the similar approach as described in the earlier section. So, the free spaces are required to be estimated next through which the swarm may navigate further (shown in Fig. 6.5).

Let us assume that $\mathbf{s} = \{s^1, s^2, ..., s^q\}; \forall s^i \in \Re^2$ are the set of the coordinates of the peaks as obtained from $\tilde{\mathbf{R}}(t+\tau)$. After identifying these, the controller searches for two local minimum points around each peak (i.e., the leftmost and rightmost points lying along the opening of that path correspond to a peak as shown in Fig. 6.5). Therefore, the local minimum points ($l^i \in [l^i_1, l^i_2]$) around the i^{th} peak (s^i) is

$$l^i = l^i \quad \text{such that} \quad \left[\frac{d\tilde{\mathbf{R}}(t+\tau)}{dt}\Big|_{l^i} \approx 0 \quad \& \quad \frac{d^2\tilde{\mathbf{R}}(t+\tau)}{dt^2}\Big|_{l^i} > 0 \right] \tag{6.6}$$

Hence, the free space (b^i_{fs}) between the minimum points (l^i_1, l^i_2) around the i^{th} peak (s^i) will be

$$b^i_{fs} = ||l^i_1 - l^i_2|| \tag{6.7}$$

Thus, for q number of peaks, a set of free spaces $\mathbf{b}_{fs} = \{b^1_{fs}, b^2_{fs}, ..., b^q_{fs}\}$ has been obtained and for each free-space (b^i_{fs}), a virtual elliptical contour is expected to be fitted in for the successful navigation

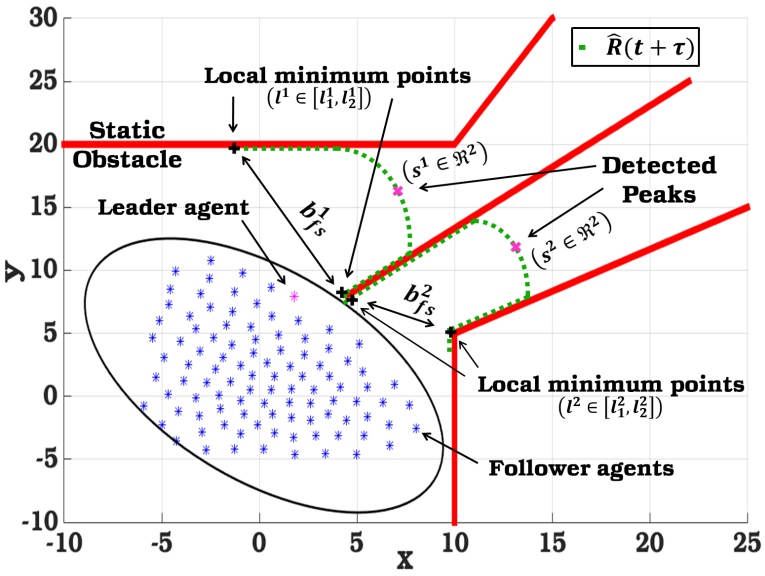


Fig. 6.5 Estimation of free spaces to fit in virtual regions.

of the swarm. In this aspect, it is demanded that the area of each contour should be optimally evaluated such that agents are proportionately distributed in each of those regions.

6.3.2 Fitting in virtual-elliptical sub-regions

The main purpose of fitting in multiple virtual regions (in each of the sensed free spaces) is to create sub-swarms such that the parent swarm would be divided into sub-groups to explore the entire environment. Let us assume that A_e is the minimum area that would be required to accommodate N number of agents, thus, $A_e = N \times \pi(r^s)^2$, where r^s is the identical region-of-repulsion of each agent in the team. Therefore, the area of the i^{th} virtual region is defined as

$$A_e^i = \frac{b_{fs}^i}{\sum_{i=1}^{q} b_{fs}^i} \times A_e; \quad \forall i \in [1,2,...,q] \tag{6.8}$$

Based on the above outcome, the length of the major-axis (a^i) of the i^{th} elliptical region ($O_e^i(t+\tau)$) is evaluated as

$$a^i = \frac{A_e^i}{\pi \times b_{fs}^i} \tag{6.9}$$

And, the minor axis length (b^i) of the i^{th} elliptical region ($O_e^i(t+\tau)$) will be equal to b_{fs}^i.

Hence, the i^{th} virtual structure has an area of A_e^i, major and minor axis lengths of a^i and b^i respectively. Subsequently, the coordinate of the elliptical center can be defined as

$$O_e^i(t+\tau) = \left(\frac{l_{1,x}^i + l_{2,x}^i}{2}, \frac{l_{1,y}^i + l_{2,y}^i}{2} \right) \tag{6.10}$$

where, $l_1^i = (l_{1,x}^i, l_{1,y}^i)$ and $l_2^i = (l_{2,x}^i, l_{2,y}^i)$ are the coordinates of the local minimum points around the peak s^i (Fig. 6.5). The same process is iterated to estimate q number of virtual regions. Once the virtual regions are created, our next aim is to split the parent swarm into sub-swarms based on the areas of the sub-regions, such that all the agents are well-distributed in each of the sub-swarms.

6.3.3 Identifying agents for each sub-group

As per our formulations (Eqs. 6.8 and 6.9), we can emphasize the fact that the shapes and areas of the virtual contours are not identical. Thus, based on the above fact, the number of agents in each of the sub-swarm would be proportionately identified. Therefore, the number of agents (SG^i) that can be accommodated inside the i^{th} virtual region ($O_e^i(t+\tau)$) (having an area of A_e^i), can be evaluated as

$$SG^i = \frac{b_{fs}^i}{\sum_{i=1}^{q} b_{fs}^i} \times N; \quad \forall i \in [1,2,...,q] \tag{6.11}$$

where b_{fs}^i is the free-space around the i^{th} peak (s^i) and N is the total agents in the swarm. The same process iterates for the q number of sub-regions. So, from Eq. 6.11, the number of agents that can be accommodated inside the i^{th} sub-swarm will be SG^i. Thus, combining the q number of sub-regions, the total number of agents should be equal to N, hence,

$$\sum_{i=1}^{q} SG^q = N \tag{6.12}$$

Selection of agents in each sub-group

After deciding the number of agents in each sub-group, the next goal of the central controller is to select agents for each of the sub-groups. In this aspect, we have assumed that the controller knows the location of each agent in the swarm. Moreover, from Eq. 6.11, we know that for the j^{th} sub-swarm, SG^j number of agents has to be assigned. So, to allocate SG^j number of agents to the j^{th} sub-swarm $\forall j \in [1,2,...,q]$, the following condition has been checked (for all the available agents)

$$SG^j = \{i\} \quad \text{such that} \left\{ \underset{i \in N}{\operatorname{argmin}} ||x_{ij}(t+\tau)|| \right\} \tag{6.13}$$

where, $x_{ij}(t+\tau) = ||x_i(t+\tau) - O_e^j(t+\tau)||$ is the Euclidean distance between the current location of the i^{th} agent ($x_i(t+\tau)$) with-respect-to the center-coordinate of the j^{th} virtual region ($O_e^j(t+\tau)$) at the $(t+\tau)^{th}$ time. Hence, as per the outcome of Eq. 6.13, those agents are selected for the j^{th} sub-group, which are comparatively nearer (with respect to the rest of the agents) to the center of the j^{th} virtual region. And, the numbers of agents that would be selected for the j^{th} sub-group, should be always less than or at best equal to (SG^j).

After the selection of agents, the central controller is now in charge of assigning leader robots for each of the sub-regions. In this aspect, we assume that all of the agents in the swarm are identical in nature.

6.3.4 Promoting a leader agent for each sub-swarm

The main requirement of assigning a lead robot (for each of the sub-groups) is that it will escort its team to progress further while initiating the OAC (Fig. 6.1). Thus, it will solely estimate the virtual region for its team, employing the sensory information of its followers, so that the corresponding sub-group can autonomously approach the target. To assign a leader agent (L^j) for the j^{th} sub-swarm, the central controller searches for an agent (amongst the agents) of that specific sub-group which is nearest to the center of the j^{th} virtual-region $O_e^j(t+\tau)$. Hence, L^j would be defined as

$$L^j = k \quad \text{such that} \quad \left[\underset{k \in SG^j}{\operatorname{argmin}} ||x_k(t+\tau) - O_e^j(t+\tau)||\right] \tag{6.14}$$

To summarize the responsibilities of the central controller holistically, we reiterate that during the exploration of an unknown environment by a robotic swarm, if the leader senses multiple paths ahead, it will pass on this information along with the sensing data ($\tilde{\mathbf{R}}$) to a dedicated centralized controller. The role of the central controller is to identify the free spaces (b_{fs}), fit in virtual-elliptical regions in each of the free spaces, cluster agents to create sub-swarms, and finally select the leader agent for each sub-group. Once the lead agents are endorsed, the centralized controller broadcasts the leader agents' information to all the follower agents and the sub-swarms' information to all the lead agents. After that, the leader of each sub-group will take care of the navigational process of its sub-swarm.

Next, we perceive a scenario, where we consider that there is only a single path available ahead through which a few sub-groups can travel further at the same time [219]. This implies that after a while, some of the separate tracks detected earlier, merge into a single path. In such kind of situation, the agents in those groups would combine in order to form a stronger and energy-efficient super-group. Consequently, in the next section, we have addressed the issue of rejoining some of the sub-swarms to form a super-swarm.

6.4 Rejoining of sub-swarm

To illustrate the rejoining or merging phenomenon, we presume a scenario where two sub-groups, namely $\Omega^i(t+\tau)$ and $\Omega^j(t+\tau)$ respectively approach towards a single target along two different paths at $(t+\tau)^{th}$ time. However, after a while, there exists a single lane ahead, through which they have to travel at the same time (as shown in Fig. 6.6).

In such a kind of scenario, two possibilities emerge realistically; either the sub-groups will approach the target sequentially along the path ahead, or they would merge to form a unified group and then proceed towards the target. Creating a group (the constraint is the free space available

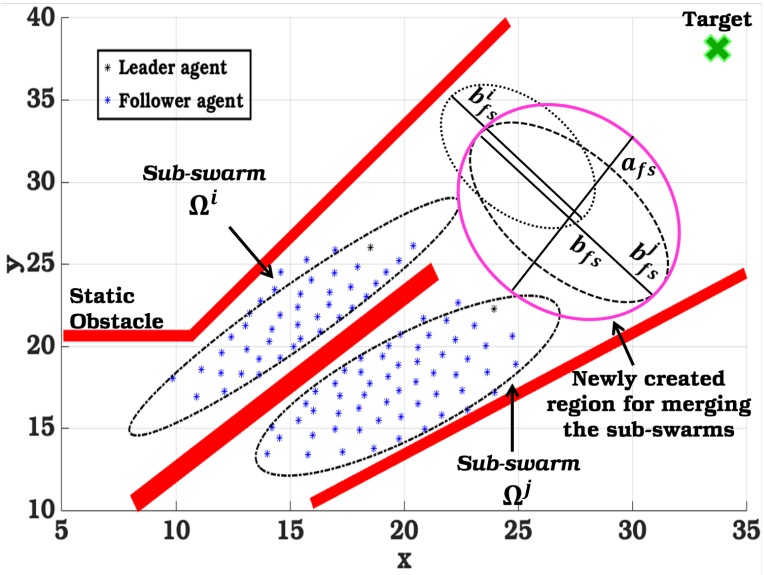


Fig. 6.6 Rejoining of two sub-swarms.

onwards) is always beneficial in terms of maintaining cohesiveness, ensuring fault-tolerance, and less communicational and computational overhead along with promising an energy-efficient scalable swarm system. Thus, in presence of a solitary track and/or unique target, the sub-groups would rejoin to create a super-group. It is needless to say that the process of rejoining two or more sub-swarms will be completely controlled by the central controller. For rejoining to happen, the respective leaders of the sub-groups communicate with the central controller. In this aspect, the function of the central controller is to decide upon the rejoining phase while assessing the area of intersection between the next virtual structures as computed by the leaders individually before merging. If there is no area of intersection, the process continues as it is, otherwise, the merging would happen between the sub-groups while fitting in an elliptical region (consolidating the virtual structural information of those sub-groups) by the centralized controller.

We assume that at the time instant $(t+\tau_1)$ (where, $\tau_1 > \tau$), two virtual sub-swarms, defined as $\Omega^i(t+\tau_1)$ and $\Omega^j(t+\tau_1)$ have been etched by the leaders of the respective sub-groups, having SG^i and SG^j number of agents respectively (vide Fig. 6.6). The regions are having the centers at $O^i_e(t+\tau_1)$ and $O^j_e(t+\tau_1)$, and minor-axis lengths of b^i and b^j respectively. The virtual regions have been drawn with black dotted lines in Fig. 6.6. We find that there is an area of intersection between those virtual regions. Upon receiving this information from the leaders, the function of the central controller is to create an elliptical region such that the agents in those sub-groups could accommodate themselves into the newly developed virtual region (hence, merging or rejoining happens). To create the virtual region, the central controller requires estimating the length of the minor and major axis of the elliptical structure that can be evaluated as

$$b(t+\tau_1) = \max(b^i(t+\tau_1), b^j(t+\tau_1)) \tag{6.15}$$

$$a(t+\tau_1) = \frac{A_e^i(t+\tau_1) + A_e^j(t+\tau_1)}{\pi \times b(t+\tau_1)} \tag{6.16}$$

where $b(t+\tau_1)$, $a(t+\tau_1)$ are the length of the minor and major axis of the newly formed elliptical region, $A_e^i(t+\tau_1)$ and $A_e^j(t+\tau_1)$ are the areas of the virtual structures as estimated by the leaders of the i^{th} and j^{th} subgroups respectively.

Therefore, for n numbers of subgroups to be merged at the t^{th} time, the length of the minor and major axis of the resulting elliptical region would be

$$b(t) = \max\{b^1(t), b^2(t), ..., b^n(t)\} \tag{6.17}$$

$$a(t) = \frac{\sum_{i=1}^{n} A_e^i(t)}{\pi \times b(t)} \tag{6.18}$$

where $\mathbf{A_e}(t) = \begin{bmatrix} A_e^1(t) & A_e^2(t) & ... & A_e^n(t) \end{bmatrix}$ are the set of area of the n number of subgroups at the t^{th} time. Therefore, the resultant area of the final structure can be determined by the following theorem.

Theorem 1: If n numbers of virtual regions having different structural areas (namely, A_e^1, A_e^2, ..., A_e^n) with varying number of agents in each substructure (assuming Q^1, Q^2, ..., Q^n) are required to merge, then, the minimum area (A) of the final structure to accommodate all the agents should satisfy

$$A \geq \pi (r^s)^2 \times \sum_{i=1}^{n} \left(Q^i\right)$$

where r^s defines the region-of-repulsion of each agent.

Proof: To accommodate Q^i number of agents having a repulsion region of r^s (each agent), the minimum required area (A_e^i) of a virtual structure will be

$$A_e^i \geq \pi (r^s)^2 \times Q^i; \quad \forall i \in \{1, 2, ..., n\} \tag{6.19}$$

Now, to merge n numbers of virtual regions, we consider that the lengths of the minor and major axis of the final contour are b and a respectively; such that

$$b = \max\{b^1, \quad b^2, \quad ..., \quad b^n\} \tag{6.20}$$

$$a = \frac{\sum_{i=1}^{n} A_e^i}{\pi \times b} \tag{6.21}$$

Moreover, the resultant area (A) of the final region should be

$$A = \pi \times b \times a \tag{6.22}$$

Equating Eqs. 6.21 and 6.22, we obtain

$$A = \sum_{i=1}^{n} A_e^i \tag{6.23}$$

Now, comparing Eqs. 6.19 and 6.23, we get

$$A \geq \pi(r^s)^2 \times \sum_{i=1}^{n}(Q^i) \tag{6.24}$$

Thus, the theorem 1 has been proved.

So, from the above results, we can infer that to place all the agents inside an amalgamated virtual region, the minimum area of that region would be dependent on the number of agents to be merged and their associated repulsion regions. After forming the region, the central controller assigns a lead agent (which is nearest to the center of the newly formed contour (Eq. 6.14) for that specific super-swarm so that it can guide its team to progress further.

To demonstrate the performance of the proposed strategy, we have performed several software simulations and hardware experiments that are provided in the subsequent section. All simulations are performed in the same software-hardware environment as mentioned in Sec. 3.7.

Table 6.1 Parameters chosen for the simulation study

Symbol	Description	Value
d_{thre_+}	Max. threshold limit	5 unit
d_{thre_-}	Min. threshold limit	2 unit
m	Number of distance sensors	8
k_c	Attraction coefficient	12
k_f	Repulsion coefficient	2.5

6.5 Results and Discussions

In this simulation study, we have specifically designed some environments that consist of obstacles and single or multiple targets having narrow corridors as shown in Figs. 6.7, 6.9, and 6.11 respectively. The parametric values as displayed in Table 6.1 have been chosen for the simulations.

Case 1: Sub-swarm creation in a multi-target environment

In this study, a team of 50 robots is expected to reach the predefined target locations (as represented by the black cross) while avoiding static obstacles (marked in the red line) in a topological environment having an area of 100×100 unit^2 as shown in Fig. 6.7. In order to achieve a coherent swarm, the region-of-repulsion (r^s) of each agent has been chosen to be 3.2 units. At the beginning of the simulation, an arbitrary agent is selected as the lead agent (black-star), and the rest are acting as followers (blue-star). Based on the sensory information of every agent, the leader creates a virtual region as shown in 'station 1' of Fig. 6.7. To accumulate inside that region, the convergence controller of all the agents is fired after that. The initial contour has been constricted further to accommodate all agents inside it for achieving a strong inter-agent cohesiveness among them ('station 2' of Fig. 6.7). At t = 16 sec, the lead robot identifies multiple paths ahead (in this simulation three different

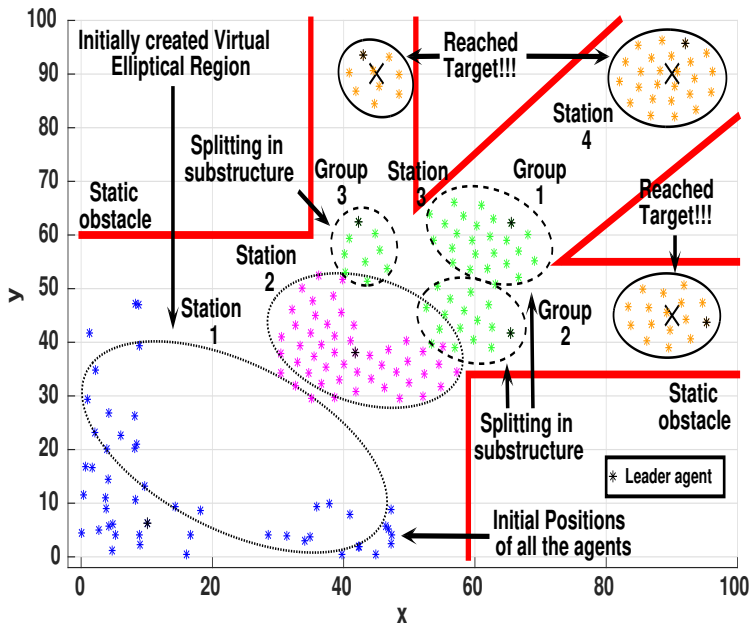


Fig. 6.7 Creation of sub-swarms in a multi-target environment with 50 agents in the group.

paths ahead) based on the sensory information of all the agents in the team. Hence, it shares this specific data with the centralized controller. As per the outcomes of this controller, three different virtual regions (thus, three sub-swarms) are created to be fitted in the three different passage ways. In addition, one lead agent has been elected for each sub-group and the authority of path-planning has been transferred to the leaders. Subsequently, all the sub-swarms are approaching their target locations ('station 3' of Fig. 6.7) at t=20 sec. Finally, all the sub-groups have successfully reached their destinations approximately at t = 82 sec ('station 4' of Fig. 6.7).

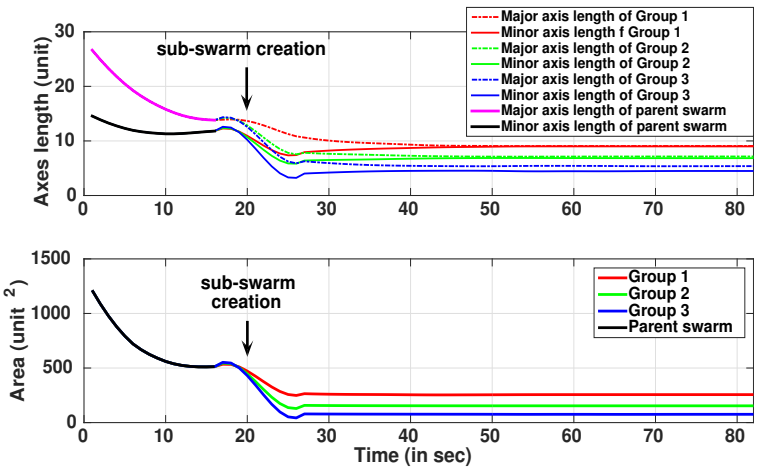


Fig. 6.8 Axis length and area computation of the subgroups in a multi-target environment (case 1) with 50 agents in the swarm.

The length of the major, and minor axis and the areas of the virtual subregions of the above experiment are presented in Fig. 6.8. From the figure, it is observed that at $t = 15$ sec, the effective area of the virtual region is around 500 unit^2, which is almost equal to the minimum required area for the virtual region to accommodate 50 agents (having r^s of 3.2 units) as displayed in 'station 2' of Fig. 6.7. At $t \approx 16$ sec, the process of sub-group creation has been started inside the central controller. At $t \approx 20$ sec, three sub-swarms are configured, and they are approaching to their respective targets ('station 3' of Fig. 6.7).

As the sensed free-space of subgroup 1 is wider than others, the number of agents in this subgroup is larger which effectively enhances the area of that specific sub-swarm (Fig. 6.8). Moreover, it is perceived that the summation of the areas of all the subgroups is nearly equal to the area of the virtual region which is created just before the formation of the sub-groups ('station 2' of Fig. 6.7).

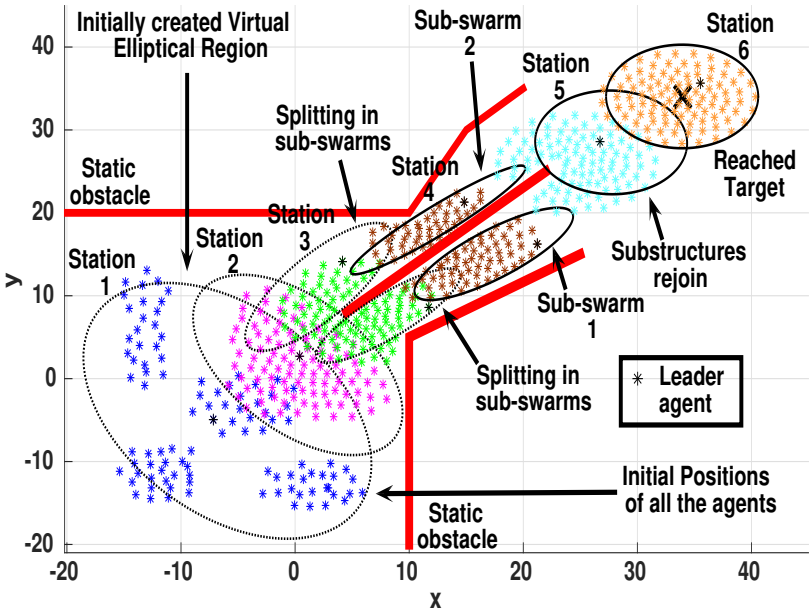


Fig. 6.9 Splitting and rejoining of sub-swarms in a single target environment with 100 agents in the team.

Case 2: Split and rejoin in an obstructed environment

To observe the split-join aspect, a simulation has been performed with 100 agents in an environment having a dimension of 40×40 unit^2 with a single target (as shown in Fig. 6.9). In this study, the region-of-repulsion (r^s) of each agent has been specified to be 0.5 unit. During the navigational steps, the lead agent creates several virtual-regions based on the sensory information of all the agents in the team (station 1 and station 2 of Fig. 6.9). At $t = 10$ sec, the parent swarm has been split into two sub-groups (the environment demands so) under the influence of the centralized controller (station 3 of Fig. 6.9) and the process continues until $t = 45$ sec (station 4 of Fig. 6.9). After that, the two

lanes merge into a single track, and it is leading toward the target. To handle this situation, both the sub-swarms rejoin to form a cohesive super-swarm (station 5 of Fig. 6.9). Finally, the entire group arrives at the destination at $t = 60$ sec (station 6 of Fig. 6.9).

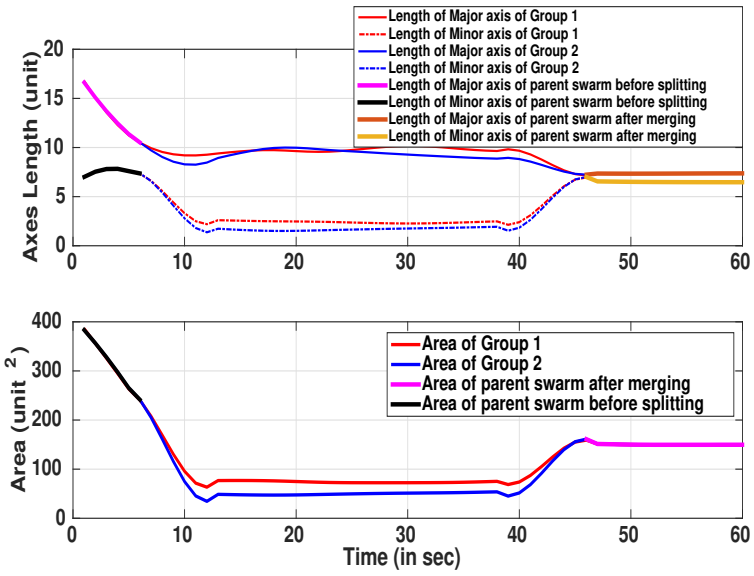
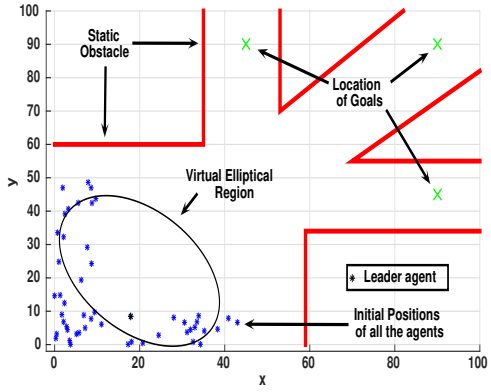


Fig. 6.10 Axis length and area computation of the subgroups during the split-join scenario (case 2) with 100 agents in the swarm.

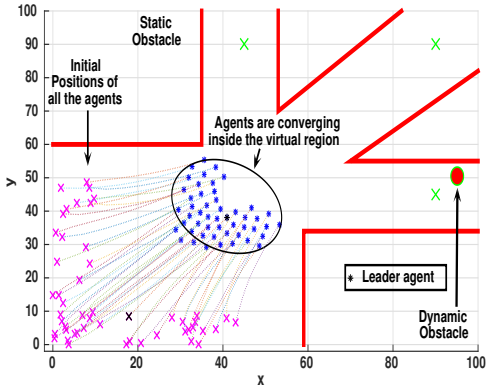
To analyze the split-join behavior of the proposed scheme, axis lengths, and areas of the virtual regions during the navigational steps are computed as shown in Fig. 6.10. From the figure, it is seen that at $t = 8$ sec, the effective area of the virtual region is around 150 unit^2 which is nearly equal to the minimum required area to accommodate 100 agents (having r^s of 0.5 unit). From this time onwards, the parent swarm is divided into two subgroups. At $t = 45$ sec, the merging phase has been started. At the end of this stage, the effective area of the super-warm is observed to be 150 unit^2. Therefore, from the above results, we can claim that using the virtual-region-based shape control approach, it is possible to create cohesive swarms/sub-swarms in multi-target/multi-path scenarios.

Case 3: Sub-swarm creation in a multi-target environment occluded with both static and dynamic obstacles

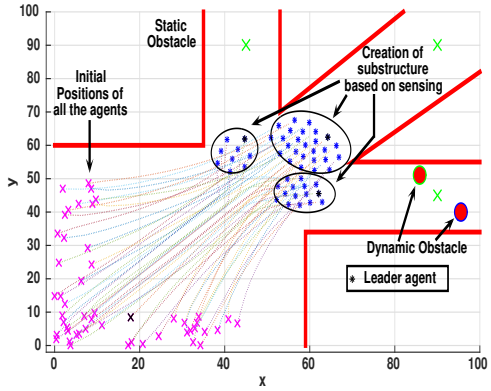
Finally, to measure the efficacy of the proposed scheme in a similar scenario (as depicted in Fig. 6.7) with dynamic obstacles, simulation has been performed with 50 agents as shown in Fig. 6.11. Here also we assume r^s to be 3.2 units. Several virtual-elliptical regions are created during the course of the movement (shown in Figs. 6.11a, and 6.11b). Suddenly, because of multiple paths, the swarm is split into three subgroups (Fig. 6.11c) at $t = 35$ sec. At $t = 50$ sec, two obstacles are approaching towards the subgroup 2 (Fig. 6.11d). To avoid those obstacles, a constricted elliptical region has been developed by the lead agent at $t = 62$ sec (Fig. 6.11e). Finally, all the sub-swarms have successfully reached to there respective target locations at $t = 85$ sec (Fig. 6.11f).



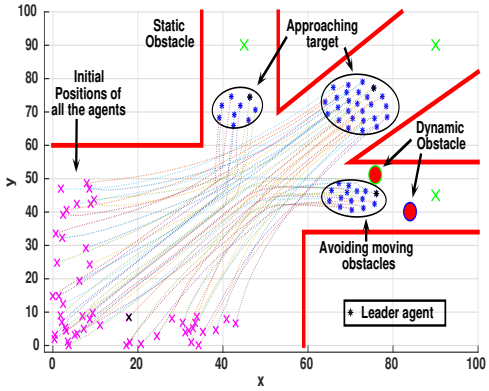
(a) at t = 0 sec.



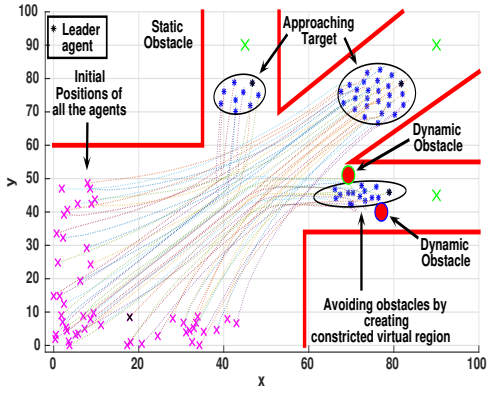
(b) at t = 22 sec.



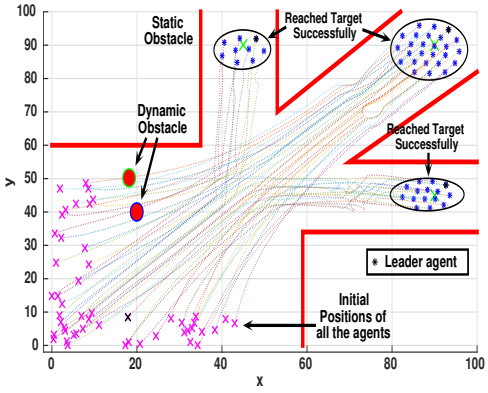
(c) at t = 35 sec.



(d) at t = 50 sec.



(e) at t = 62 sec.



(f) at t = 85 sec.

Fig. 6.11 Sub-swarm creation of 50 agents in an dynamic environment with multiple targets.

To assess the performance of the proposed scheme in a dynamic scenario, the axis length and areas of the virtual regions are computed during the navigational process as shown in Fig. 6.12. From the figure, it is noticed that the presence of dynamic obstacles does not change the effective areas of the sub-swarms. However, the axis lengths are significantly altered for subgroup 2 during the period of $t = 55$ sec to $t = 70$ sec where the moving obstacles are blocking the path of that respective subgroup. The above feature defines the usability of the proposed strategy in the swarm robotics path planning problem.

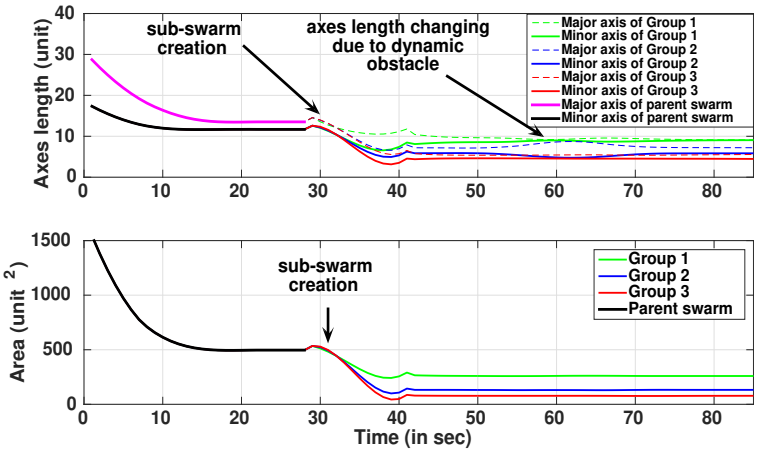


Fig. 6.12 Axis length and area computation of 50 agents in dynamic obstacles' scenario (case 3) in multi-target environment.

6.5.1 Performance analysis

To analyze the efficacies of the proposed scheme, we have executed each simulation multiple times. Based on those outcomes, we have analyzed the performance in terms of the inter-agent cohesiveness, execution time, and scalability of agents.

Inter-agent Cohesiveness

Two agents i and j are considered to be cohesive if they maintain at least $(2 \times r^s)$ distance between themselves, where r^s is the region-of-repulsion of each agent. So, to analyze the inter-agent cohesiveness, we compute the structural density [216] of the swarm in each iteration. Mathematically, the structural density $(D(t))$ can be defined as the ratio of the number of cohesive links to the maximum number of links, thus, the expression of $D(t)$ at the t^{th} time for N numbers of agents would be

$$D(t) = \frac{\sum_{i=1}^{N} \left[\sum_{j=1}^{N} g_j^i(t) \right]}{N(N-1)} \tag{6.25}$$

where, $g_j^i(t) = \begin{cases} 1, & \text{if } ||x_{ij}|| \leq 2r^s \\ 0, & \text{otherwise} \end{cases}$, $||x_{ij}||$ is the Euclidean distance between agents i and j. So, for a highly cohesive swarm where all of the agents are maintaining at least $(2 \times r^s)$ distance from each other, the value of $D(t)$ would be close to one, whereas, for a loosely coupled swarm, its value will be near to zero.

Here again, we take the liberty to depict the different cases that we have simulated earlier for analyzing the performance of the proposed control scheme for investigating $D(t)$.

Scenario 1: Splitting and rejoining of the swarm in an obstructed environment with only static obstacles (Fig. 6.9).

Scenario 2: Splitting of the swarm in an obstructed, multi-target environment with static and dynamic obstacles (Fig. 6.11).

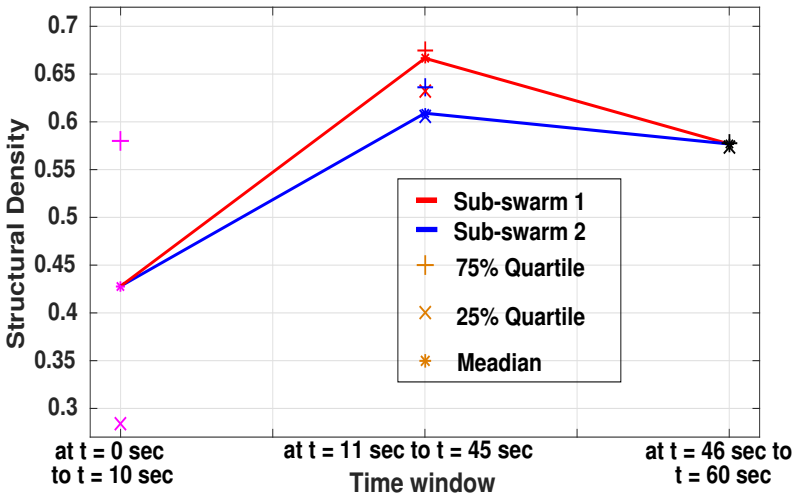


Fig. 6.13 Inter-agent cohesiveness analysis in scenario 1 for 100 agents in the swarm.

Analyzing the average structural density in scenario 1 (Fig. 6.9) in different time windows (Fig. 6.13), we have observed that the inter-agent cohesions, in the beginning, are less as all the agents are attempting to accommodate inside the virtual region ('station 1' of Fig. 6.9). In this specific situation, the area of the virtual region is large, thus, the agents are loosely coupled inside it. As a result, the structural density is reduced. After that, based on the environmental condition two sub-swarms are formed ('station 3' of Fig. 6.9). Here, the areas of the virtual regions of both the sub-swarms are smaller than the parent swarm. As a consequence, all of the agents within the virtual territories would maintain strong cohesiveness, resulting in an increasing structural density score (Fig. 6.13). Finally, during rejoining phase ('station 5' and 'station 6' of Fig. 6.9), again the area of the super-swarm becomes larger than the areas of the sub-swarms, thus, the cohesiveness reduces, subsequently, the structural density decreases (Fig. 6.13).

In Scenario 2 (Fig. 6.11), initially, the structural density (as shown in Fig. 6.14) is less because of the converging characteristics of the swarm inside a huge virtual region (Figs. 6.11a, and 6.11b).

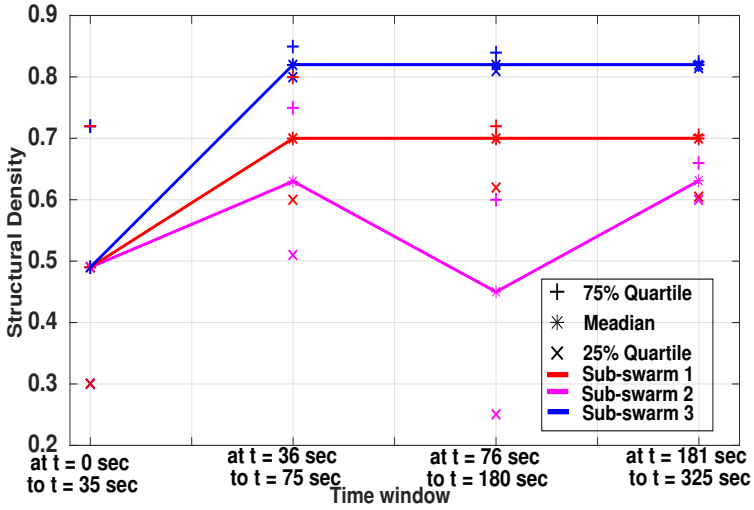


Fig. 6.14 Inter-agent cohesiveness analysis in scenario 2 for 50 agents in the swarm.

After that, three sub-swarms are produced (Fig. 6.11c) from the parent swarm. As the areas of all the sub-swarms are less than the parent swarm, the inter-agent cohesiveness, as well as structural densities, get increased during this time frame (Fig. 6.14). To bypass incoming moving obstacles (Figs. 6.11d, 6.11e), the area of the sub-swarm 2 is further constricted, however in this particular state, the length of the major axis of the virtual elliptical region is comparatively higher than the minor axis. As a result, the agents are spread inside the region to avoid the dynamic obstacles. So, the structural density reduces during this time window (Fig. 6.14). Finally, when all the obstacles are avoided, the inter-agent cohesiveness, as well as structural density, attains a steady-state value.

In this specific study, it should be noted that the average inter-agent cohesiveness in scenarios 1 and 2 are around 0.55 and 0.6 respectively (Fig. 6.13, and 6.14). In light of these outcomes, we can conclude that an agent is strongly virtually connected or maintaining at least $(2r^s)$ distance to more than 50% of total agents during the navigational process.

Scalability of agents

In this chapter, the scalability of agents has been studied with respect to the inter-agent cohesiveness (by computing the structural density) and mission completion time (or execution time).

Evaluating structural densities for scenario 1 and scenario 2 as shown in Figs. 6.15 and 6.16 respectively, it is observed that with increasing agents (in the swarm), the average cohesions are exhibiting a decreasing trend. This phenomenon is quite obvious because, with increasing agents in the swarm, the areas of the virtual regions will be enhanced. As a result, for a large swarm, an agent cannot maintain $(2r^s)$ distance with its neighboring agents in the swarm.

From the outcomes, we should note one important feature of our proposed approach is that with 500 and 1000 agents (in a team), the average structural densities are observed to be around 0.43, 0.4 (for scenario 1), and 0.325, 0.31 (for scenario 2) respectively. In light of these outcomes, we might

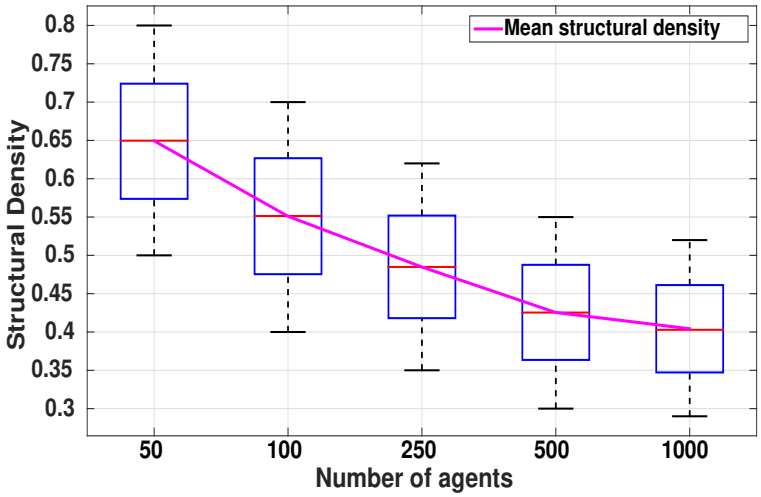


Fig. 6.15 Comparison of Structural density with scalability of agents in scenario 1.

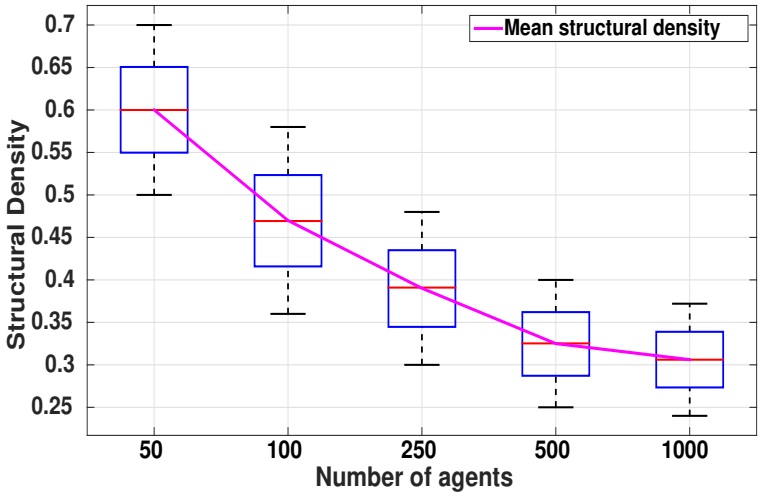


Fig. 6.16 Comparison of Structural density with scalability of agents in scenario 2.

conclude that even with 1000 agents in a swarm, an agent maintains strong cohesiveness with at least 30% of the total agents by employing our proposed technique.

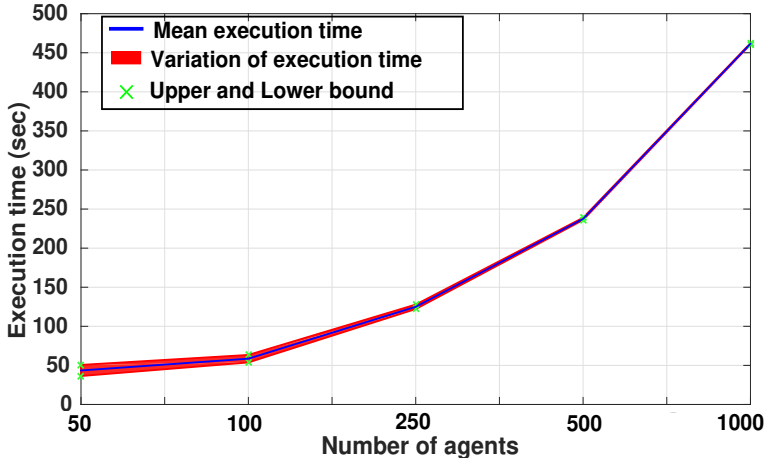


Fig. 6.17 Comparison of Execution time with scalability of agents in scenario 1.

Comparing the executing time with scalable agents (Figs. 6.17, and 6.18), it is seen that the mean execution time is following an increasing pattern. Moreover, while simulating with 1000 agents for scenarios 1 and 2, the proposed scheme consumes around 425 sec and 5900 sec respectively to complete the mission objectives. Now, if we look into the deviation plots (of Figs. 6.17, and 6.18), we would observe that the variations of execution times are reduced with scalable agents in the swarm. The reason behind this fact is that as per the navigational rule, each agent shares the location coordinates of the sensed nearby obstacles with the leader agent. Based on that information, the leader agent evaluates the virtual regions in each step of movement. Thus, for a large swarm, the computed regions would be optimal as the numbers of sensing coordinates are considerable. As a result, the time variations for large swarm will be less.

Now, statistically, we have examined the deviation plots of Figs. 6.17, and 6.18 to assess the repeatability and variability of the proposed scheme. The detailed analysis can be found in Figs. 6.17 and 6.18, as well as Tables 6.2, and 6.3 respectively.

Table 6.2 Statistical analysis on the execution time with scalable agents for scenario 1

No. of Agents	**Mean time (in sec)**	**std (in sec)**	**COV**	**1^{st} quartile (Q_1) (in sec)**	**3^{rd} quartile (Q_3) (in sec)**	**IQR ($Q_3 - Q_1$) (in sec)**	**Median (Q_2) (in sec)**	**Max. time (in sec)**	**Min time (in sec)**
50	43.18	7.06	0.164	37.26	49.19	11.9	42.94	55.9	31.03
100	58.69	4.92	0.085	54.55	63.03	8.48	58.87	67.28	50.02
250	124.95	2.9	0.023	122.47	127.4	4.93	124.88	129.99	120.00
500	237.41	1.47	0.006	236.09	238.7	2.6	237.45	239.97	235.00
1000	461.52	0.87	0.002	460.73	462.26	1.53	461.53	462.99	460.00

In Scenario 1, we have only considered the split-merge aspect of a robotic swarm in a static circumstance. The simulation result for 100 agents in the swarm is shown in Fig. 6.9. The results of the statistical analysis are presented in Fig. 6.19, and Table 6.2. We can make the following observations based on the results.

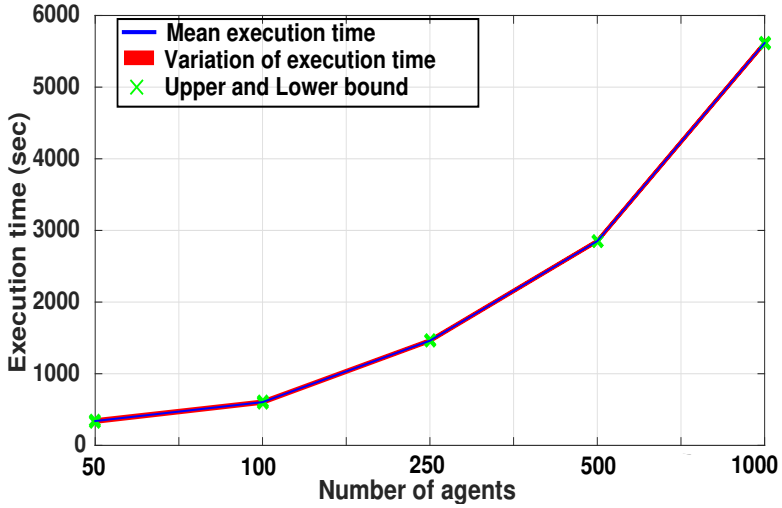


Fig. 6.18 Comparison of Execution time with scalability of agents in scenario 2.

1. The mean and median of the variations in execution time grow as the number of agents in the swarm increases.
2. The standard deviation, coefficient of variation, and inter-quartile range decrease as the number of agents in the team rise.

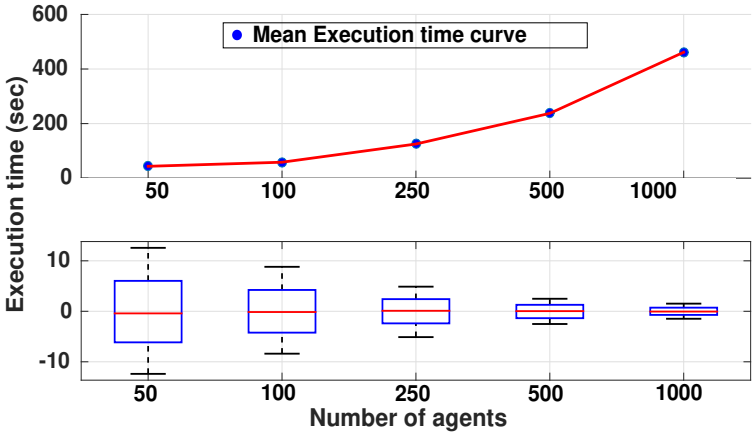


Fig. 6.19 Analysis on scalability of agents with execution time of scenario 1.

In Scenario 2, we have examined the splitting feature of the robotic swarm in a dynamic environment. The results of simulations for 50 agents in the swarm are depicted in Fig. 6.11. From the statistical analysis, as shown in Fig. 6.20, and Table 6.3 with scalable agents, it is observed that

1. The mean and median execution times increase as the number of agents enhances, whereas the standard deviation, coefficient of variation, and inter-quartile range are following a decreasing trend with the scalable agents.

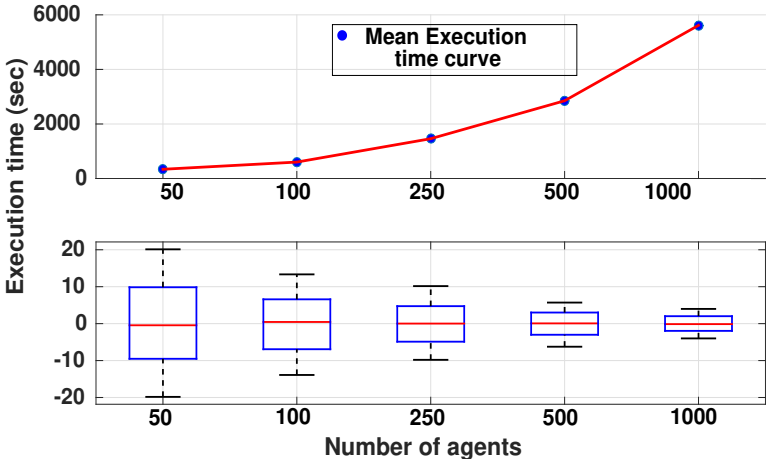


Fig. 6.20 Analysis on scalability of agents with execution time of scenario 2.

Table 6.3 Statistical analysis on the execution time with scalable agents for scenario 2

No. of Agents	Mean time (in sec)	std (in sec)	COV	1^{st} quartile (Q_1) (in sec)	3^{rd} quartile (Q_3) (in sec)	IQR ($Q_3 - Q_1$) (in sec)	Median (Q_2) (in sec)	Max. time (in sec)	Min time (in sec)
50	340.54	11.6	0.034	330.6	350.8	20.13	340.6	360	320
100	603.27	7.75	0.013	597.01	609.7	12.68	602.9	617.3	590
250	1460.17	5.77	0.004	1455.9	1466.03	10.05	1461.3	1471	1451
500	2850.92	3.5	0.0012	2848.1	2854.1	5.98	2851.2	2857	2845
1000	5615.91	2.32	0.0004	5613.9	5617.9	4.06	5615.8	5620	5612

From the outcomes of the analysis, we have witnessed similar results as obtained from the simulation outputs, i.e., the mean and median execution times increase, and the standard deviation, coefficient of variation, and inter-quartile range decrease with the scalable agents of the swarm. The fundamental reason behind this fact is that as the number of sensing points increases with scalable agents, the leader agent evaluates optimal virtual regions in each step of movement. Thus, the same procedure would not be repeated in the next iteration. Subsequently, the standard deviation, coefficient of variation, and inter-quartile range reduce.

6.5.2 Performance Comparison

To determine the efficacies of the proposed approach, we have performed comparative analyses with the other techniques cited in the literature and the metrics are inter-agent cohesiveness and the convergence times of the well-cited algorithms vis à vis our control proposition while ensuring the scalability of the agents in the swarm.

For scenario 1 (i.e., static obstacle avoidance case as shown in Fig. 6.9), we have investigated the performance of our proposed solution and compared the same with the performance of some other existing solutions [165], [217] and [219] where the split-merge aspect is considered for a robotic swarm operating in an environment having only static obstacles. The results are presented in Figs. 6.21 and 6.22.

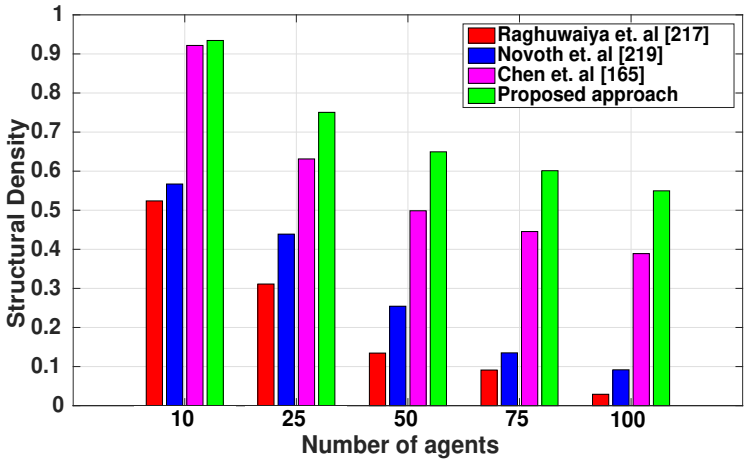


Fig. 6.21 Structural density comparison with other techniques in scenario 1.

The variation of the cohesiveness index with the number of agents in the swarm for scenario 1 has been plotted in Fig. 6.21. From the plot, it is evident that all the techniques (including the proposed one) exhibit a decreasing pattern with an increasing number of agents in the team. So, we can infer that when there are fewer agents in the team, inter-agent cohesion increases, and vice versa. Also, we observe that our proposed approach outperforms the previous strategies [165], [217] and [219] in the presence of static barriers and narrow pathways. Moreover, in [165], the authors have explicitly imposed tight inter-agent cohesiveness in their control design. Even then when we compare their techniques with ours, we find that our strategy improves the inter-agent cohesion in general. The rationale behind this fact is that as per our control proposition, all the agents are required to converge inside the virtual region etched by the leader agent.

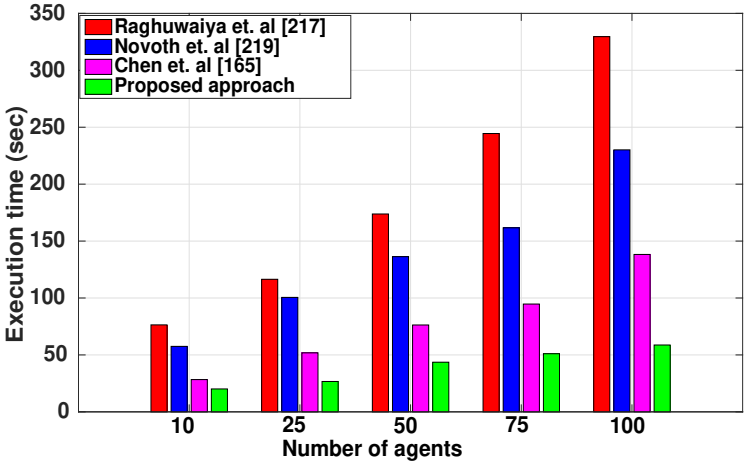


Fig. 6.22 Execution time comparison with other techniques in scenario 1.

Finally, comparing the execution times (Fig. 6.22), it is perceived that even with 100 agents in the swarm, our proposed solution consumes less time than the other traditional techniques because when

an agent approaches a specific virtual region, it does not need to communicate with its neighboring agents, hence, our proposed scheme allows all the agents to navigate independently.

Since the above-mentioned works have not considered an environment for dynamic obstacles; hence, for scenario 2, we compare our results with [44] and the performance comparisons have been shown in Figs. 6.23 and 6.24 respectively.

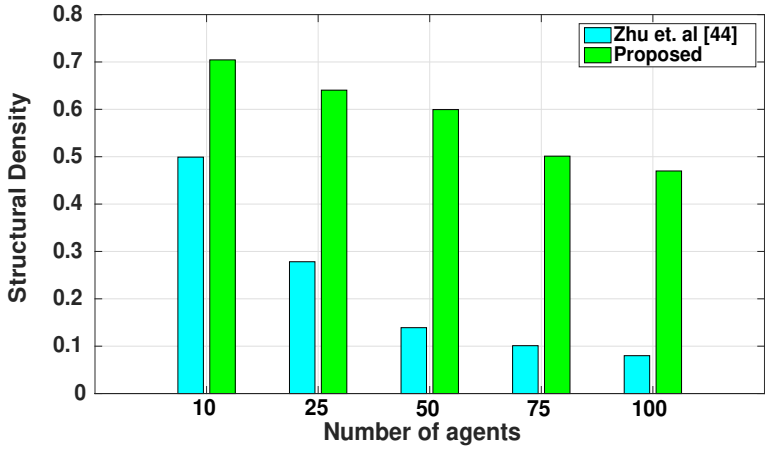


Fig. 6.23 Structural density comparison with other techniques in scenario 2.

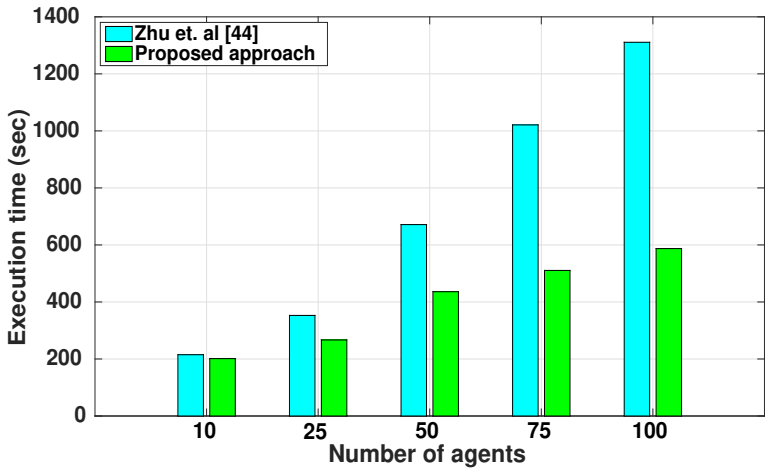


Fig. 6.24 Execution time comparison with other techniques in scenario 2.

While we compare the performances of the proposed technique with that of [44] in the dynamic environment (scenario 2), it is observed that the inter-agent cohesion reduces and the execution time increases with the number of agents in the swarm for both of the approaches. The performance comparisons are presented in Figs. 6.23 and 6.24. However, the cohesiveness of our method is substantially higher than [44], and the execution time is less. The key reason behind this fact is that in [44], the convex optimization strategy was applied to the image data as captured by each agent in the

group. Depending on this output, the position of that agent had been calculated irrespective of the location of its neighboring agents. In dynamic conditions, this particular mechanism assisted each agent of the group in deciding whether to split or merge. In our case, we achieve similar consequences by employing the data from the distance sensors only. So, our technique is much more efficacious than the previous schemes in both static and dynamic circumstances.

6.6 Summary

In this chapter, we have conceptualized a novel approach to perform the split-join aspect of a robotic swarm. The region-based shape control scheme has been linked with the leader-follower methodology to achieve the splitting and merging of a swarm of robots. The proposed control method can effectively be applied in multi-target as well as multi-path scenarios. Moreover, in the face of the incoming dynamic obstacles, the swarm can break into sub-swarms autonomously. Two sub-swarms moving along the same path can rejoin to form a super-swarm (if required) utilizing the offered procedure. The entire control proposition is dependent on the creation of the virtual elliptical regions during the navigation steps. Additionally, the leader-follower architecture has been adopted to optimize the overall performance of the system. To synthesize the control scheme, it is verified that the proposition would be sufficiently scalable in terms of the number of agents in the swarm, sub-swarm, and super-swarm. Furthermore, within a team, all the agents will maintain cohesiveness among themselves during the navigational steps. To illustrate the performance of the proposed control action, several simulation results have been performed for various scenarios. The performance analysis shows the efficacies of the proposed scheme.

Chapter 7

Hardware Implementation

7.1 Introduction

In this chapter, we provide the results of the hardware implementation of the proposed control methods as described in Chapter 3 to Chapter 6. To assess the performance of the various control actions, we conduct several experiments in real environments with diverse environmental scenarios containing static and dynamic obstacles. Furthermore, to analyze the scalability of the proposed control systems in terms of the number of agents in the swarm, various experiments have been performed with scalable two-wheeled mobile robots.

The architectural overview of the robots has been described in the upcoming section. The motion planning of a multi-robot system using the bat algorithm-based evolutionary approach is then tested in different environmental conditions. After that, we examine the geometrical-region-based shape control scheme with scalable agents in the team, followed by the polygonal-region-based approach. Finally, we provide the hardware implementation result of the split-join scheme. The detailed results and analysis of all the experimental outcomes are given below.

7.2 Architectural framework of the robotic platform and environments

The architectural framework of the robotic platform has been described in this section. In this regard, we would like to point out that we do not employ any off-the-shelf hardware for the experiments; we design and fabricate the platforms ourselves. The detailed architectural overview (as shown in Fig. 7.1) is explained next.

The designed two-wheeled Unmanned-Ground-Vehicle (UGV) models (namely, *SonPI*) as shown in Fig. 7.1a is controlled by a Raspberry pi based 1.4 GHz, ARM quad-core processor. The hardware architecture is outlined in Fig. 7.1b. The WiFi floater (IEEE 802.11) provides a communication

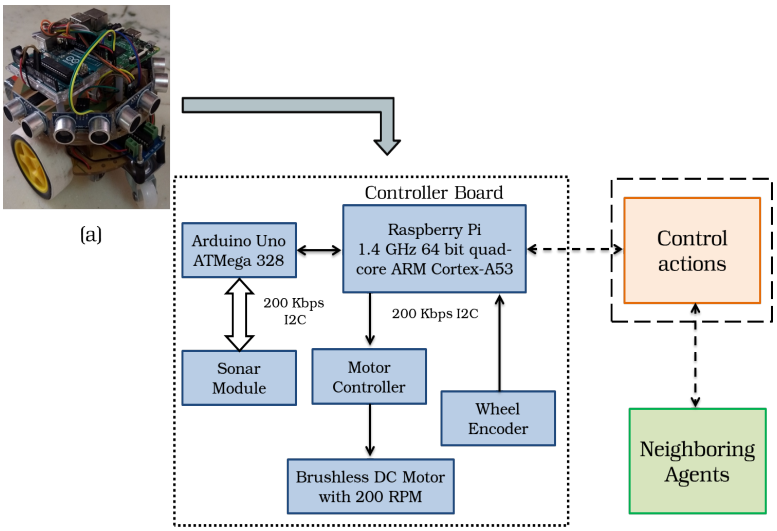


Fig. 7.1 Architectural overview of the test platform.

channel among the UGVs and the centralized server. To sense the current position, the high-precision wheel-encoders [220] are attached to the wheels. These will also help to estimate the relative positions of the neighboring agents. To detect the nearest obstacles and to apply the proposed algorithms, each robot in the team is equipped with 5 Sonar modules (HC-SR04 [186]) spacing at 45^0 apart (so that it can sense up to 180^0 around itself). The collected sonar data from Arduino ATMega328 are passed on to the pi-board via I2C communication protocol [221] for sharing with the controller block. Based on this information, the necessary actuation is passed on to the wheel-actuators of all the agents either distributively or via the central controller.

The bat-algorithm-based approach (as outlined in Chapter 3) has initially been implemented on the proposed system. Then, the geometric-region-based approach (as described in Chapter 4) has been performed. Next, down the line, the polygonal-region-based scheme (as explained in Chapter 5) has been studied. Finally, the split-join strategy (as proposed in Chapter 6) has been examined on the real robotic platform. The algorithmic implementations and test results are described in the upcoming sections.

7.3 Implementation of bat algorithm-based approach

To implement the bat algorithm (discussed in Chapter 3) in robotic setups, we have employed the affordable sonar sensors (HC-SR04 [186]) that are working in 5V DC power supply to estimate the distance of the nearest barriers for mimicking the echolocation principle of bats. All of the sensors are operating at a constant frequency (f_{max}) of 40 kHz while generating 8 cycle-burst of ultrasonic pulses (r) that are very similar to the typical values of the real bats [187]. Depending on the sensing data as collected by a robot, further processing has been performed on the Raspberry pi unit. During the

navigational process, sometimes an agent/agents may get stuck at some local minimum points. In this kind of scenario, the loudness (A) and pulse emission rate (r) of the respective sensors are modulated in software to obtain the global best solution.

The entire hardware testing has been divided into four parts: initially, an investigation has been executed in an obstacle-less environment. After that, it has been extended to cluttered environments with static obstacles and low-complex dynamic obstacles. Finally, the previous experiment has been continued for several moving obstacles.

7.3.1 Design consideration

It is apparent that during the navigational process, the position of each robot is indeed mostly sensed by the wheel-encoders. However, the wheel-encoders may introduce errors in getting the position of each robot. To overcome this shortcoming, in addition to the wheel-encoders, we have employed an overhead camera too [222], under the assumption that the camera can cover the entire workspace and at the same time, can accurately evaluate the positions of all the agents in the team in each iteration by employing Lucas-Kanade optical flow technique [223]. Thus, the camera acts as a complementary sensor placed in yet another feedback loop (the vision loop) for the controlled system, the comparator of which computes an error function between the sensed positions of the camera and the encoder, for each agent in every time steps. This generates a corrective motion signal through which each robot updates its position and hits the desired location coordinates. This multi-loop control strategy, eventually, reduces the wheel encoders' error.

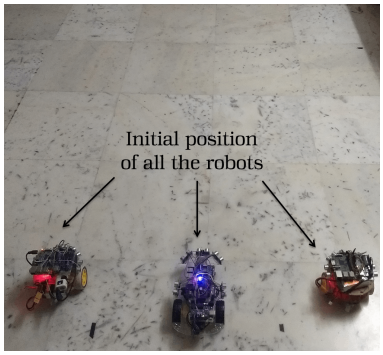
After each iteration/time step, once the positions of all the robots are duly sensed (sensed through the overhead camera), the relative positions of all the neighbors of a given agent can be easily determined. The information about the relative positions amongst the robots is broadcast over the entire swarm. Hence, the positional information of the complete swarm topology is available to each and every agent in the swarm, thus making an agent capable enough to identify its neighbors irrespective of the distance between the agent and one or some of its neighbors. In the next step, using this information, the attraction/repulsion control action is actuated to maintain the strong inter-agent cohesiveness as discussed in Chapter 3.

7.3.2 Results and discussions

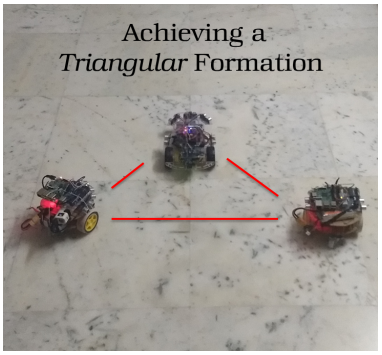
The bat algorithm-based motion planning approach has been tested in four different cases; the initial study contains the formation control of three robotic agents in an obstacle-less environment. The next investigation includes several scalable agents forming adaptive formations in static environments. In experiment 3, we have experimented with four robots in a low-complex environment. Finally, in experiment 4, a complex environment has been created with a dynamic occluded environment to measure the adaptability of formation with four agents in the swarm.

Case 1: Achieving formation in obstacle-less environment

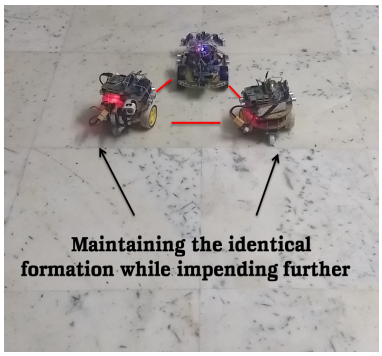
In this experiment, it is shown how quickly a formation can be accomplished among the robots in an obstacle less environment. *Three* robots, starting from some arbitrary locations (as shown in Fig. 7.2a), are progressing towards a target. At t=38 sec., a *triangle*-formation has been achieved as represented in Fig. 7.2b. In Fig. 7.2c, it is shown that while maintaining the requisite formation, all the agents are approaching further at t=71 sec.



(a) at t = 2 sec.



(b) at t = 38 sec.



(c) at t = 71 sec.

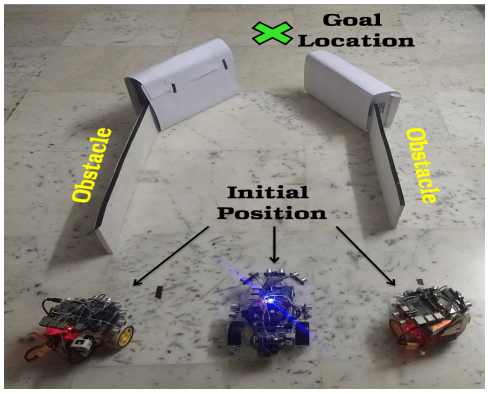
Fig. 7.2 Three robots maintain a specific formation while operating in an obstacle-less environment.

Case 2: Approaching target with adaptive formation in static environments

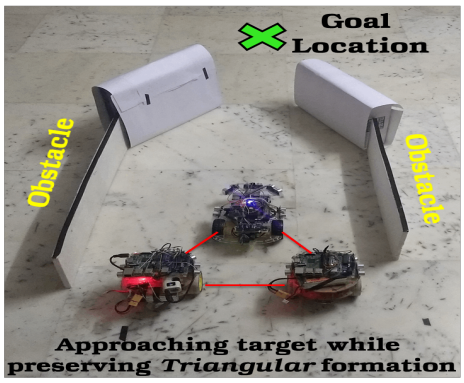
In this case study, two experiments have been performed with scalable agents in the team. The environments are designed with static obstacles/boundaries with narrow pathways to validate the viability of the proposed control action.

Three agents in the team

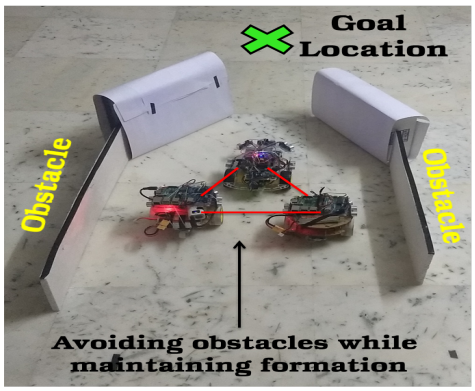
In this experiment, *three* robots are employed such that they are required to reach a predefined target location (marked in green-cross of Fig. 7.3). Starting from some arbitrary positions (shown in Fig. 7.3a); all the robots create a triangular-formation (Fig. 7.3b) at t=21 sec. While maintaining the



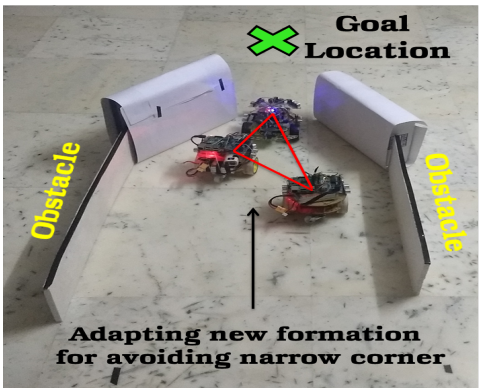
(a) at t = 1 sec.



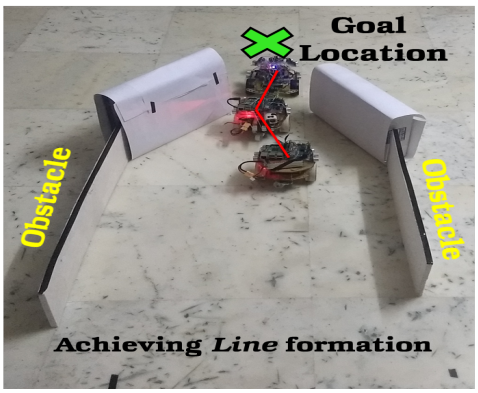
(b) at t = 21 sec.



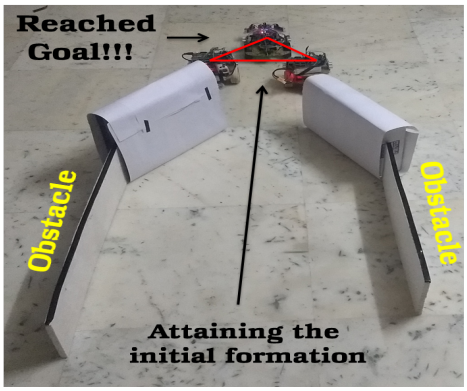
(c) at t = 27 sec.



(d) at t = 62 sec.



(e) at t = 183 sec.



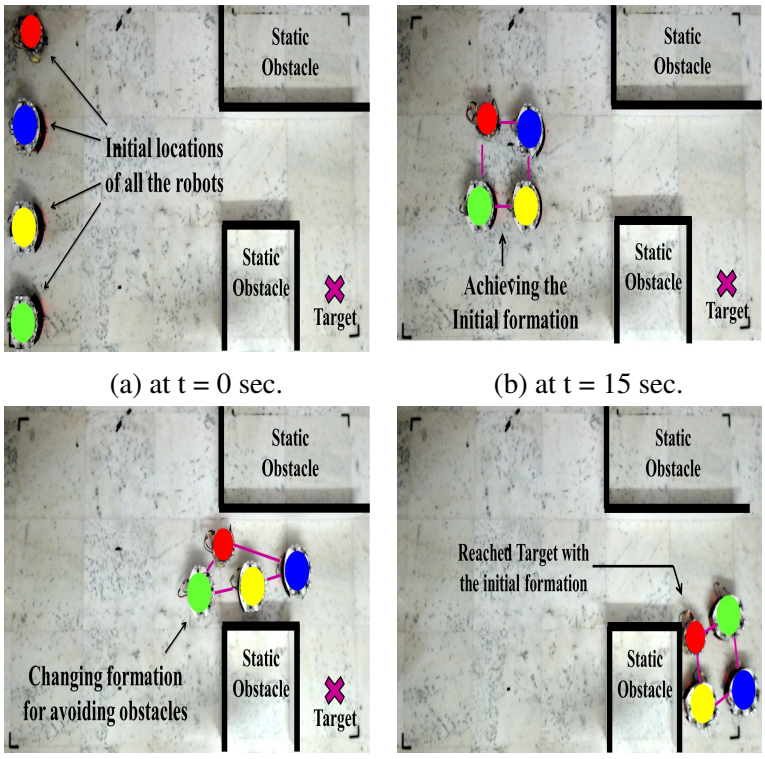
(f) at t = 249 sec.

Fig. 7.3 Three robots are approaching the target location in an environment having static obstacles.

same, they are approaching further at t=27 sec. (Fig. 7.3c). Suddenly, the path is being obstructed by a narrow passage (shown in Fig. 7.3d), and consequently, the agents are changing their current inter-agent pattern for avoiding the barrier. At t=183 sec., a line-formation has been accomplished by the agents to pass through the narrow zone (Fig. 7.3e). Finally, they have reached the target location at t=249 sec (Fig. 7.3f).

Four agents in the team

In this experiment, four robots have been used to reach a predefined target location (marked in *magenta* cross of Fig. 7.4). For recognizing the agents by the overhead camera, a specific marker has been attached to each robot. During the navigational process, several inter-agent formations (as shown in Fig. 7.4b, 7.4c) have been created by the multi-robot system for avoiding obstacles. Finally, all the agents have reached the goal location at $t = 245$ sec (Fig. 7.4d).



(a) at t = 0 sec. (b) at t = 15 sec.

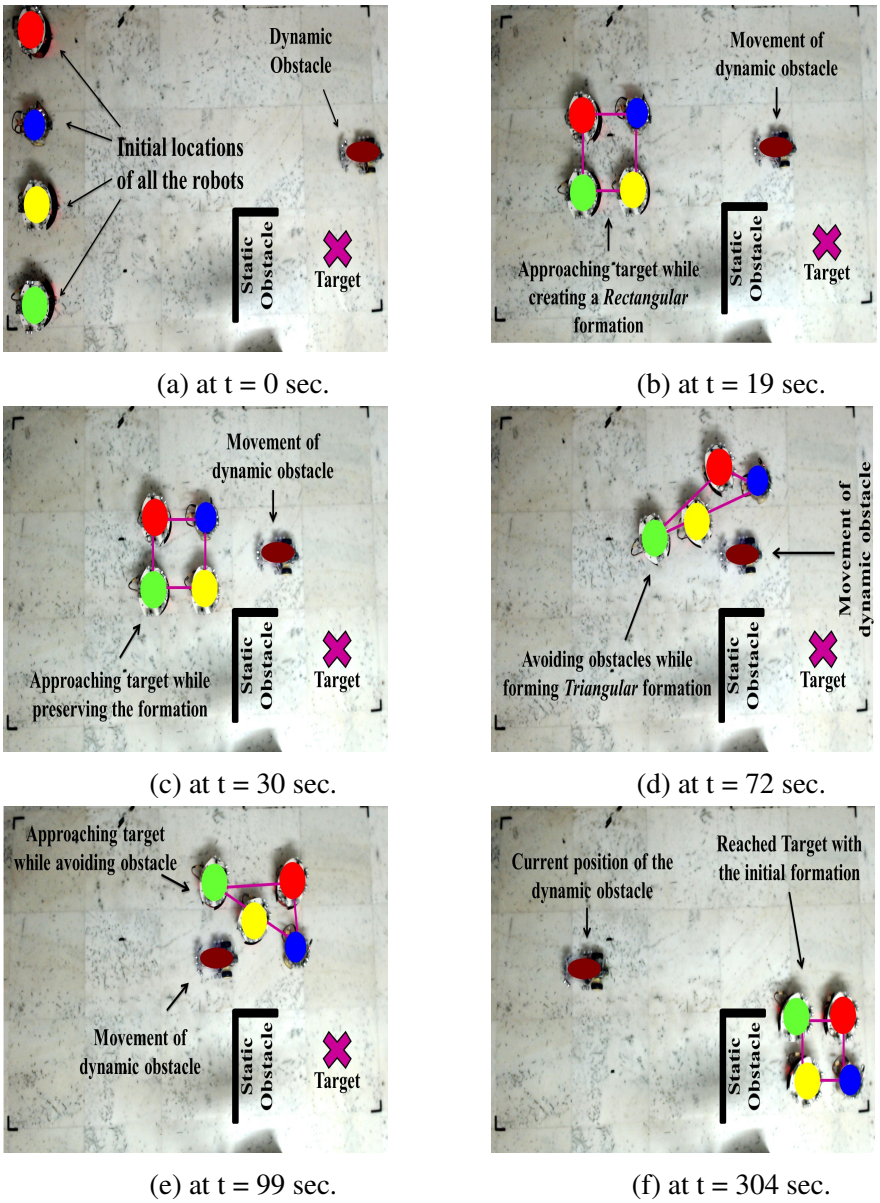
(c) at t = 32 sec. (d) at t = 245 sec.

Fig. 7.4 Four robots are approaching the target locations in an environment having static obstacles.

Case 3: Approaching target in a low-complex dynamic environment

In this case study (as shown in Fig. 7.5), the experimentation has been conducted with *four* robots, in a low-complex dynamic environment that contains static obstacles/boundaries (*black*-line) and a moving obstacle (*brown*-ellipse). Starting from some randomized positions (as displayed in Fig. 7.5a),

all the agents are progressing towards the goal (*magenta*-cross). At $t = 19$ sec (vide Fig. 7.5b), it is observed that a *rectangular*-formation is created among the agents which is sustained until $t = 30$ sec (Fig. 7.5c). After that, a moving obstacle which is advancing towards the swarm has been sensed. To avoid that *triangular*-formations have been developed among the agents as presented in Figs. 7.5d and 7.5e respectively. Finally, all the agents have reached the goal location successfully at $t = 304$ sec while forming the initial *rectangular* shape as depicted in Fig. 7.5f.



(a) at t = 0 sec. (b) at t = 19 sec.

(c) at t = 30 sec. (d) at t = 72 sec.

(e) at t = 99 sec. (f) at t = 304 sec.

Fig. 7.5 Four robots are approaching the goal locations in a low-complex environment having one moving obstacle.

Case 4: Approaching target in a complex dynamic environment

Our conclusive experiment has been performed with four robots in a complex environment having dynamic obstacles and static boundaries (vide Fig. 7.6). Starting from the arbitrary positions (Fig. 7.6a), all the robots are producing several new formations (as shown in Figs. 7.6b, 7.6c, 7.6d, and 7.6e) in order to avoid dynamic obstacles while navigating towards the goal location through the narrow pathways. After avoiding all the moving obstacles, a *rectangular*-like shape has been created by the multi-robot system (Fig. 7.6f). Finally, all the robots have reached the goal location at $t = 575$ sec (Fig. 7.6g).

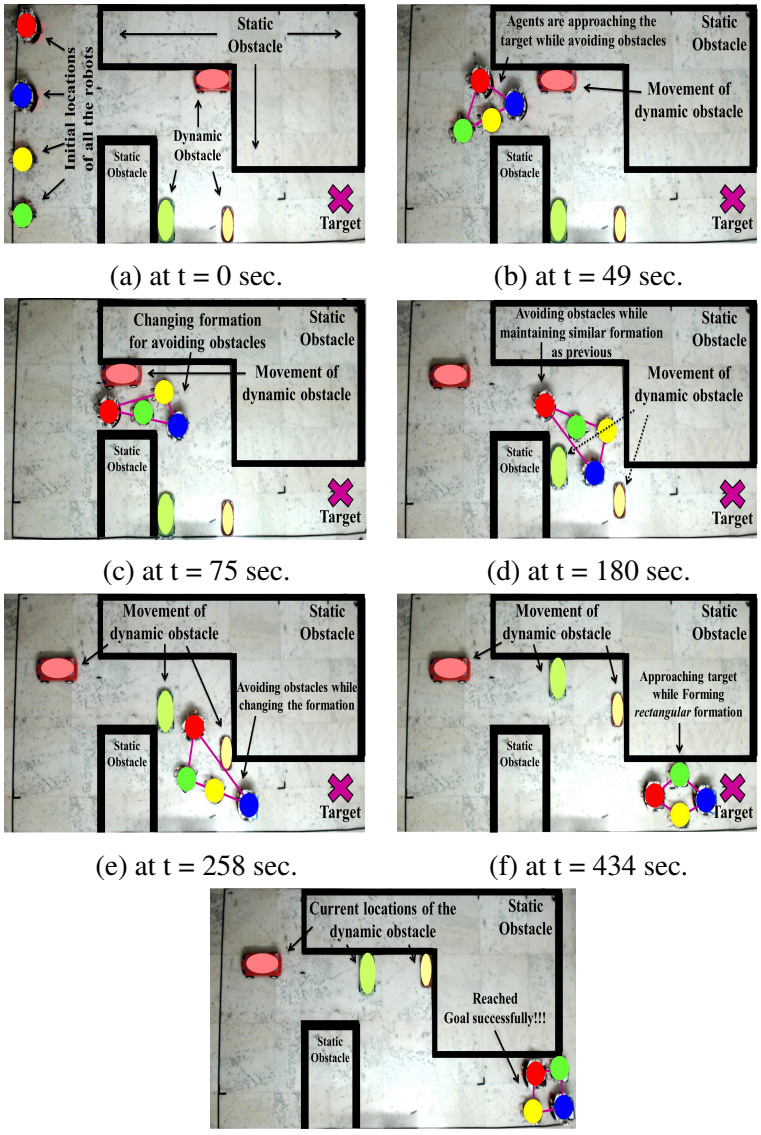


Fig. 7.6 Four robots are approaching the goal location in a complex dynamical environment having *three* moving obstacles and narrow pathways.

From all of the above experimental outcomes, we have discovered that by employing the bat algorithm-based approach, an agent with open space ahead always leads the swarm system during navigation. For example, in Fig. 7.6, it can be seen that the blue-marked agent has open space ahead 7.6a during the initialization of the experiment. Thus, in the entire navigation, this particular agent is leading the swarm toward the direction of the target.

7.3.3 Performance analysis

To analyze the performance of the proposed technique in the above-mentioned case studies, we have computed the time complexity, the inter-agent formation error, and the mean path length (of all the agents in the team) for all of the experiments. The analyzes are shown in Fig. 7.7, and Tables 7.1, 7.2, and 7.3 respectively. The following metric has been evaluated in each iteration to determine the formation error

$$FE = 1 - \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\Omega^{i}(t)}{N(N-1)}; \quad \Omega_{j}^{i} = \begin{cases} 1 & \text{if } 2r^{s} \geq ||x_{ij}|| \geq 3r^{s} \\ 0 & \text{otherwise} \end{cases} \tag{7.1}$$

where N defines the number of robots in the team, r^{s} is the associated repulsion region between the agent i and j. Two agents (assuming i and j) are expected to be in formation if the Euclidean distance ($||x_{ij}||$) between them is between $2r^{s}$ and $3r^{s}$. Therefore, from the outcome of Eq. 7.1, we can conclude that the formation error (FE) can fluctuate from 1 (worst case with no inter-agent formation) to 0 (best case with all the agents in formation). The time vs. formation error plot is presented in Fig. 7.7.

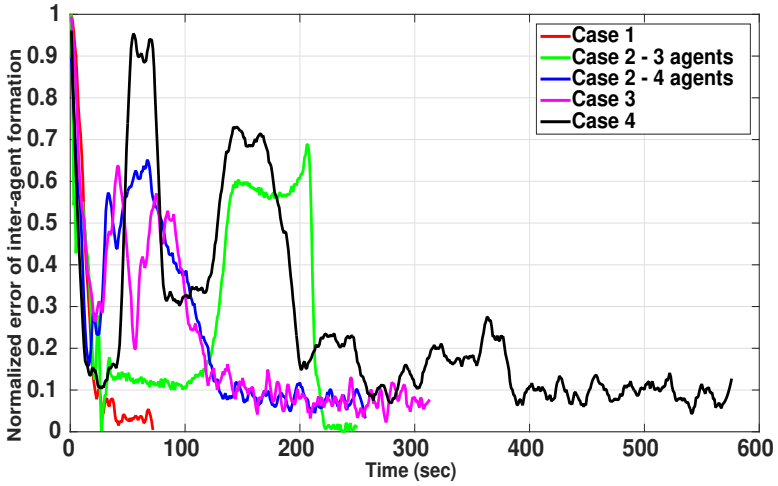


Fig. 7.7 Evaluated formation-errors and time complexities of the above-mentioned case studies.

In obstacle-less condition (case 1), the mean and standard deviation (sd) of the formation error is low for both the software simulation and hardware experimentation as in this scenario, all the agents

Table 7.1 Evaluated parameters in obstacle-less environment (case 1)

No. of agents	Time complexity (in sec.)		Formation error				Path length	
			S/W		H/W		S/W	H/W
	S/W	H/W	mean	sd	mean	sd	(in unit)	(in cm.)
3	35	71	0.09	0.01	0.21	0.05	50.5	51.2

Table 7.2 Evaluated parameters in static environments (case 2)

No. of agents	Time complexity (in sec.)		Formation error				Path length	
			S/W		H/W		S/W	H/W
	S/W	H/W	mean	sd	mean	sd	(in unit)	(in cm.)
3	98	249	0.13	0.2	0.25	0.19	51.82	53.07
4	100	245	0.26	0.2	0.32	0.21	54.41	57.93

are approaching the target while maintaining a specific inter-agent formation (Table 7.1 and Fig. 7.7). The mean path lengths are almost similar for both the software and hardware.

While comparing the performance with the scalable agents in the static environments (case 2), it is seen that adding one robot to the team does not create much impact on the overall time consumption of the system (Table 7.2) as the dimensions and the location of goals in both the environments are identical. While analyzing the formation error, it is perceived that the mean error of *4* robots is greater than the mean error of *3* robots. The primary reason behind this fact is that the triangular shape is much more structurally flexible than the quadrilateral shape in terms of sustaining cohesiveness. Finally, to compare the mean path length, we have witnessed that the path length of *4* agents is greater than the path length of *3* agents.

While examining the performance of the proposed scheme in dynamic environments (cases 3 and 4) (Table 7.3), it is seen that though the dimensions and location of targets of both the environments are identical, the multi-robot system in case 4 consumes a considerable amount of time as compared to the system in case 3 because of the varying complexity of the environments. Moreover, due to the adaptive formation features of the framework for avoiding obstacles during the navigation, the mean formation error and the path length are high for case 4 than in case 3. Hence, from all the above consequences, we can conclude that in any kind of environmental conditions with scalable agents in the team, a multi-robot system can eventually reach the goal location while impending difficult barriers with our proposed solution.

Table 7.3 Evaluated parameters in dynamic environments (case 3 and 4)

Environmental complexities	Time complexity in H/W (in sec.)	Formation error in H/W		Path length in H/W (in cm.)
		mean	sd	
Low	304	0.37	0.25	62.11
High	575	0.47	0.41	69.5

Table 7.4 Comparison analysis with circle-based gathering technique [2]

EC	No. of robots	TC (in sec.)		Formation error				Path length (in cm.)	
		CG [1]	*Prop.*	*CG [1]*		*Prop.*		*CG [1]*	*Prop.*
				mean	sd	mean	sd		
Low	3	451	260	0.1	0.05	0.27	0.22	60.03	59.27
	4	612	304	0.17	0.12	0.37	0.25	61.85	62.11
High	3	1169	466	0.22	0.04	0.37	0.32	68.34	67.03
	4	*Fail*	575	*Fail*	*Fail*	0.47	0.41	*Fail*	69.5

7.3.4 Comparative analysis

In this section, to evaluate the efficacies of the proposed solution, we have compared the performance of the system with the other well-cited literature in terms of the formation error, goal-reaching capability, execution time or time complexity (TC), and scalability. In this aspect, we would like to mention that the comparative analyses have been performed in similar environmental conditions as specified in cases 3 and 4 with scalable agents in the team. The corresponding results are shown in Tables 7.4 and 7.5.

Initially, we have examined the performance of the proposed system with the circle-based gathering scheme [2] (Table 7.4). Comparing the formation error in low-complex conditions with scalable agents, we observe that the circle-based gathering scheme [2] performs better than the proposed method in terms of maintaining a specific formation during the navigational steps. The main reason behind this fact is that during the exploration of the multi-robot system, the circular structure always sustains rigidity, so the formation error is low that subsequently improving the inter-agent coordination. However, in highly complex environments (including narrow passageways) with scalable agents, the circle-based system often fails to reach the goal location with the gathering principle. Moreover, if an environment allows them to pass through a narrow corridor (case 4 with 3 agents in the group), then we have witnessed that the execution time to arrive at the goal location is quite large as compared with our proposed technique. Hence, from the outcomes, we can confirm the superiority of the stated methodology in low as well as highly complex circumstances.

To check the superiority of the proposed bat algorithm, we have examined our system with another bat-inspired technique, namely, the binary cooperative bat algorithm [3]. The results are shown in Table 7.5. In the cooperative binary bat scheme, the velocity factor is mapped into a probability value between zero and one, and then based on that factor, the position of a bat will be updated to one or zero by comparing the mapped velocity and a random number. Experimenting with the framework (having 4-agents in the team) with the binary bat scheme in low (case 3) and high-complex (case 4) conditions, we observe that the formation errors and the consumed times remain nearly equivalent for both the schemes during the situations of the environment where the obstacle-density is low (i.e. before changing the initial formation and after achieving formation in the final step). However, in presence of dynamic obstacles and/or narrow pathways, the performance of the binary bat procedure (for both the formation error and consumed time) has significantly deteriorated because of the velocity

Table 7.5 Comparison analysis with Binary bat algorithm [3]

Environment	Phases	FE ($\mu \pm \sigma$)		Duration (in sec.)	
		Binary bat [3]	**Prop.**	**Binary bat [3]**	**Prop.**
Low complex (case 3 with 4 agents)	*Before changing initial formation*	0.36 ± 0.21	0.37 ± 0.2	$0 - 30$	$0 - 32$
	After changing initial formation	0.59 ± 0.3	0.52 ± 0.27	$31 - 185$	$33 - 123$
	Achieving final formation	0.25 ± 0.1	0.18 ± 0.05	$186 - 345$	$124 - 304$
High complex (case 4 with 4 agents)	*Before changing initial formation*	0.3 ± 0.27	0.28 ± 0.25	$0 - 44$	$0 - 50$
	Avoiding 1st moving obstacle	0.79 ± 0.2	0.73 ± 0.26	$45 - 124$	$51 - 88$
	Avoiding 2^{nd} and 3^{rd} moving obstacles	0.57 ± 0.31	0.45 ± 0.22	$125 - 459$	$89 - 290$
	Achieving final formation	0.24 ± 0.19	0.13 ± 0.049	$460 - 725$	$291 - 575$

saturation principle. Often, it is seen that the algorithm gets stuck on the local minimum points. For overcoming that situation, the algorithm requires to be iterated (one more time) with the updated sensing data and then searching for the global solution. This phenomenon would significantly enhance the time of operation of the binary bat scheme and, subsequently, the formation error increases. In our proposed procedure, we have handled such situations while modulating the pulse emission rate (r) and loudness (A) of the sensors in the software environment, so that they can be applied in real scenarios. Because of this reason, our solution consumes less execution time than the binary bat method. Therefore, interpreting all the comparisons, we can conclude that the mentioned scheme could be an alternative to the swarm robotics path planning problem.

7.3.5 Drawbacks

The main drawbacks of this approach are:

1. The time complexity is significantly increased with an increasing number of agents in the team.
2. During the experiments, if an agent gets stuck at the local minimum points, then, to overcome this issue, the loudness and the pulse emission rate are required to be modulated in software. This procedure consumes a large amount of time to execute.
3. Other than that, in this experiment, we have only considered the behavior of the bat-based scheme. However, we are not able to implement the BG and PSO approaches in the test platform because of the algorithmic constraints.

7.4 Implementation of virtual geometric-region approach

In these experiments, the local control actions are executed inside the pi board of each robot, and the global controller is running on a specialized unit having an Intel-core i7 processor with 16 GB of RAM. In this control approach, each UGV only needs to know the coordinate information of the virtual region's center for triggering the local controller. The attached sonar sensors are employed to sense the nearby environment for estimating the next footprint of the virtual structure (as discussed in Chapter 4).

7.4.1 Experimental procedure

In this experimental procedure, as per our design criteria, the sensing and actuation processes are running in parallel. The sampling rate of each sensor is 200 Hz. The sensed data acquisition time has been fixed to 250 msec. and after every 250 msec., the collected sonar data are sent to the global controller. Based on this collected sensing data from all the robots, the K-means clustering (where K=2) algorithm is performed. Typically, the algorithm consumes 50 msec. to compute. As a result of this algorithm, the nearest obstacle locations $[r_{0_1}, r_{0_2}]$, and the average Euclidean distance $AD(t)$ are evaluated. To ensure the correct estimation, the same procedure is followed for another 250 msec. Depending on this information, the sensed free-space b_{fs} has been estimated by the global controller. In this experiment, we are assuming that any of the moving obstacles would not hit the swarm within 1 sec.

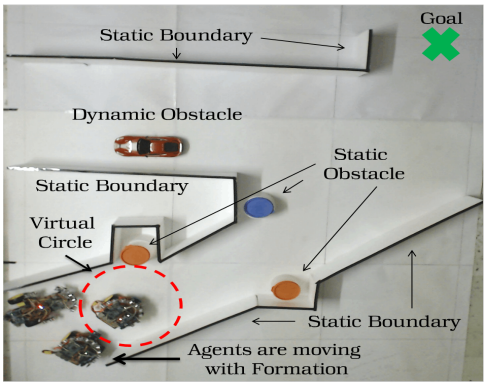
7.4.2 Results and discussions

To assess the performance of the proposed strategy in a complex environment, we have carried out several experiments with the varying number of robots in the team. The environment contains static obstacles, boundaries, and moving obstacles (e.g., programmable playing cars). The results are presented below.

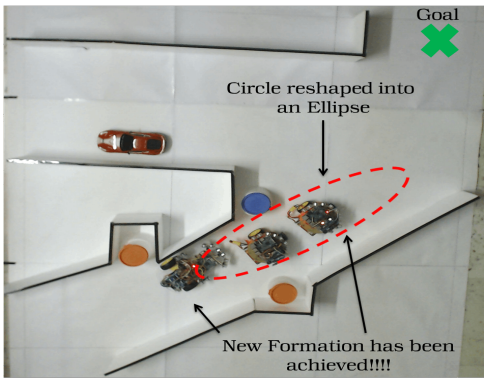
Three-robots operating in a cluttered environment

In this experiment, we have employed 3 robotic agents which are required to reach the desired goal in an obstructed environment as shown in Fig. 7.8. The environment, having a dimension of 350×175 cm^2, is a replication of the simulated environment (as depicted in Fig. 4.5 of Chapter 4) which contains 2 moving obstacles (programmable playing cars), 3 static obstacles, and static boundaries.

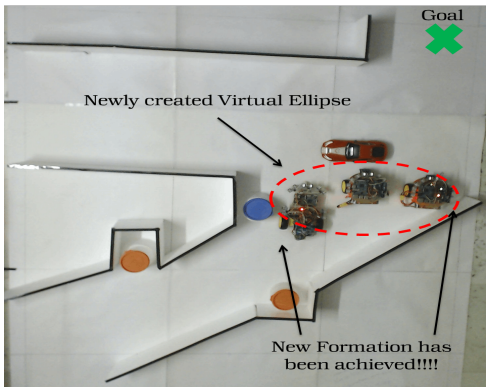
After the initial triggering of the local control action, all the robots are forced to converge inside the virtual circle while the robots do etch a *triangular* form (Fig. 7.8a). At around $t = 22.59$ sec., for fitting within the narrow corridor with a sharp corner, the virtual circle is reshaped into a virtual ellipse of an identical area (Fig. 7.8b) based on the real-time sensory information sent by all the robots. To accommodate all agents inside the virtual region, a *line*-formation (let us call it an inclined



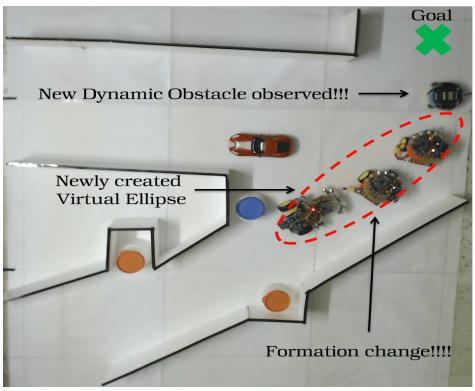
(a) at t = 1.1 sec.



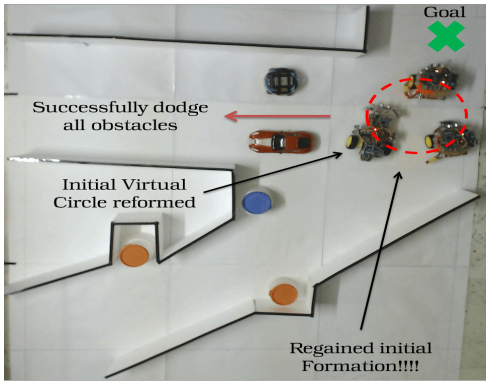
(b) at t = 22.6 sec.



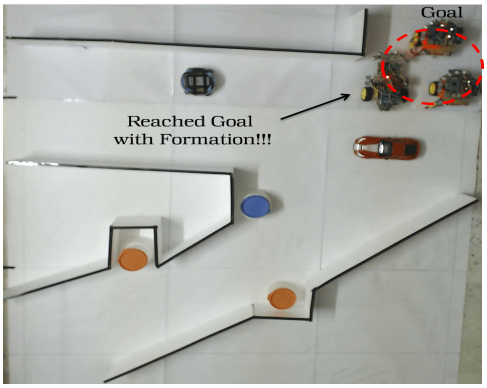
(c) at t = 35.48 sec.



(d) at t = 107.31 sec.



(e) at t = 178.5 sec.



(f) at t = 251 sec.

Fig. 7.8 Swarm agent position trajectories in a real environment with the proposed strategy: the *green*-cross represents the goal location and the *red*-contours represent the location of the virtual-structure at the various instances.

column-formation) has been formed. Next, we observed that the robots, at $t = 35.48$ sec, create a new *line*-formation, which compared to the previous one looks like a *row*-formation (Fig. 7.8c) for avoiding obstacles. After dodging all the snags, the entire group has reached the goal zone at $t = 251.08$ sec with the initial *triangular*-formation (Fig. 7.8f).

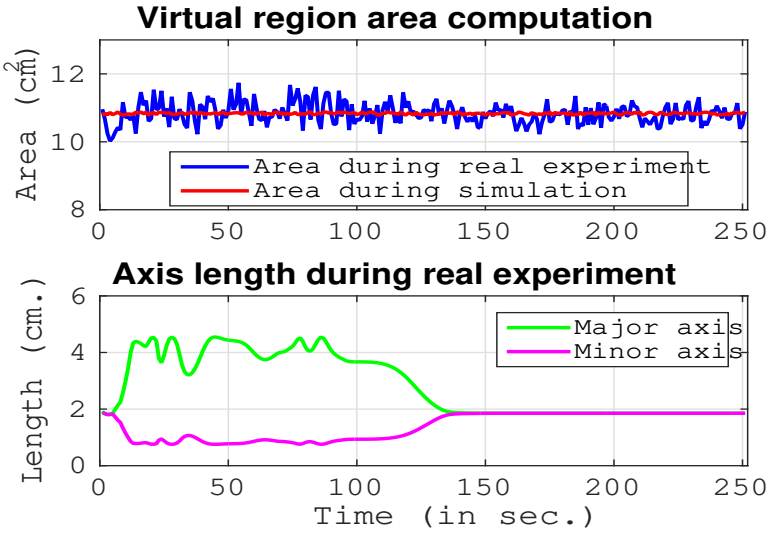


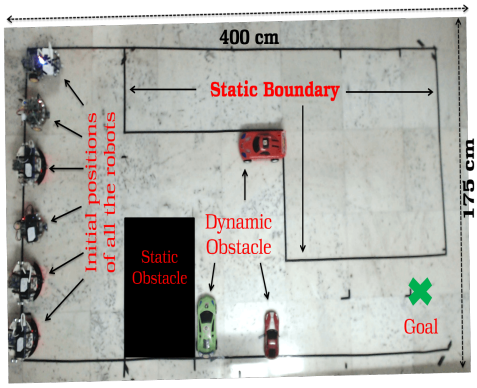
Fig. 7.9 Evaluated area and axis length of the virtual regions in the real experiment as shown in Fig. 7.8.

While comparing the experimental result with the simulated one as shown in Fig. 7.9, it is observed that the mean area of the virtual region is around 10.96 cm^2 having a standard deviation of 1.06 cm^2 which generates the coefficient-of-Variance (COV) of 0.097, on the other hand, for the simulation study, the value of COV is around 0.022 (having a mean area and standard deviation of 10.82 cm^2 and 0.24 cm^2 respectively). Moreover, the major and minor axis lengths during the experimentation are also provided in Fig. 7.9.

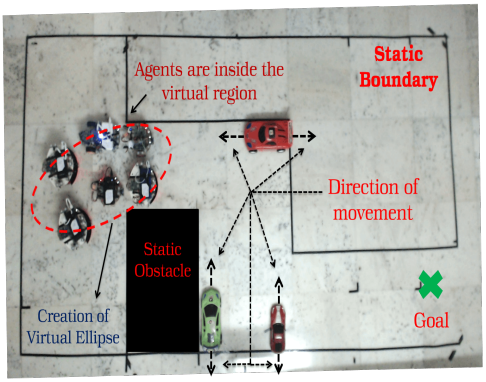
Six-robots operating in a cluttered environment

In this experiment, we have employed 6 two-wheeled mobile robots (as shown in Fig. 7.10). Starting from an initial position (as shown in Fig. 7.10a), all the agents are triggered by the actuation of the local controller. Consequently, all the robots try to assemble inside the virtual elliptical region (refer to Fig. 7.10b) while at the topmost level, the global or supervisory control unit generates a control law based on the sensory information from all the robots. To avoid obstacles (both static and dynamic), several virtual elliptical contours of identical areas are being generated as shown in the figures. Specifically, at $t = 219$ sec, a constricted elliptical region (where the length of the major axis is much greater than the minor axis) has been formed. In order to fit into the constricted corridor, the control law forces the robots to form a *line* (Fig. 7.10c). Finally, all the agents reach the goal location at $t = 615$ sec successfully.

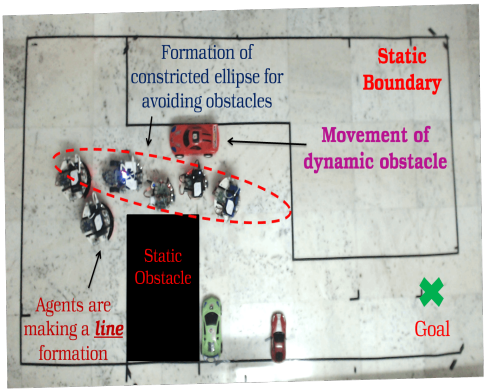
The basic premise of this work is the fact that, in all iterations, the area of the virtual regions will remain constant. However, it is apprehended that the sensing inaccuracies caused by the sensors may



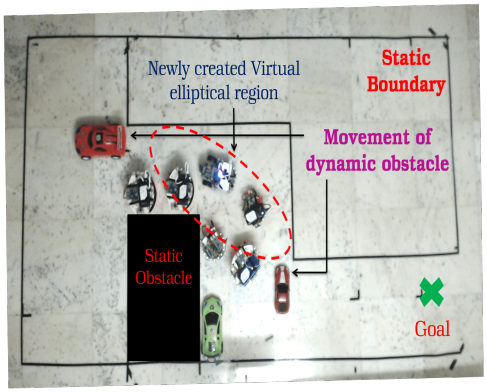
(a) at t = 0 sec.



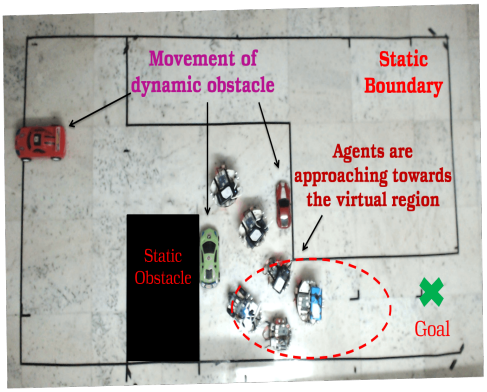
(b) at t = 172 sec.



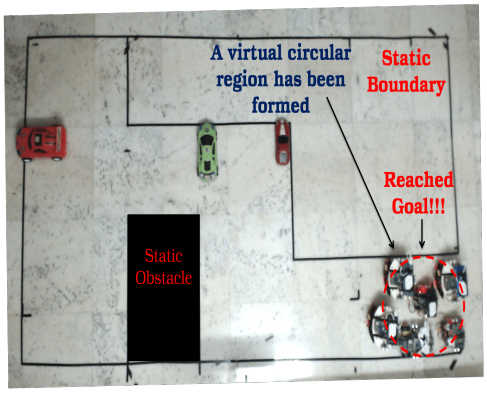
(c) at t = 219 sec.



(d) at t = 271 sec.

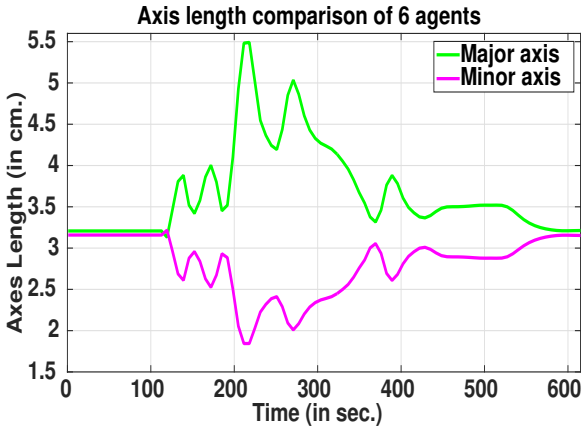


(e) at t = 391 sec.

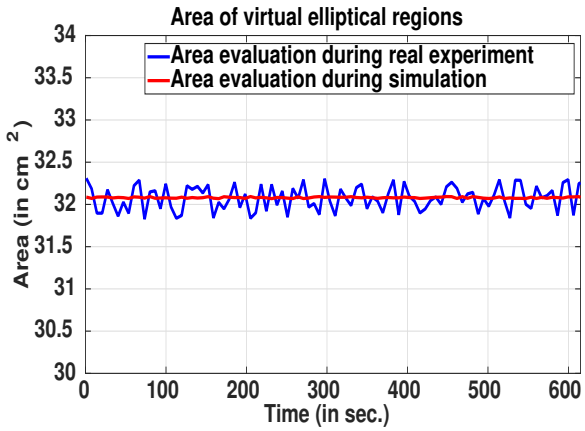


(f) at t = 615 sec.

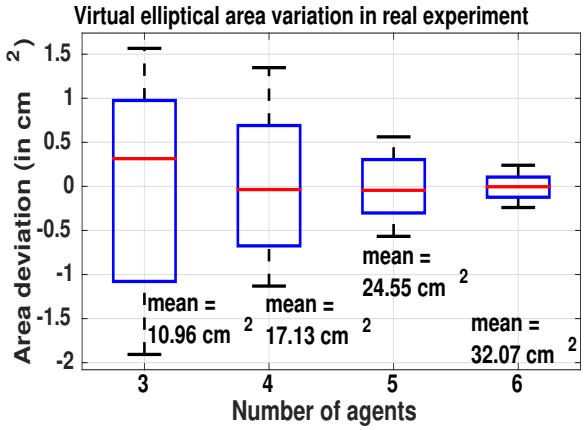
Fig. 7.10 Swarm agent position trajectories in a real complex environment with the proposed strategy: the *green*-cross represents the goal location and the *red*-contours represent the location of the virtual-structure at various instances.



(a) Evaluated Axis Length.



(b) Evaluated area.



(c) Variation of area.

Fig. 7.11 Area and axis length of the virtual regions in the real experiment (Fig. 7.10) and the area variation of the virtual region with scalable agents.

change the identical area requirement in real scenarios. To analyze this fact, we have evaluated the area of the virtual regions of the above-mentioned experiment (refer to Fig. 7.10) and have compared the same with the simulated one. We have observed that the mean evaluated area of the virtual region is around 32.07 cm^2 having a standard deviation of 0.14 cm^2 which generates the coefficient-of-variance (COV) of 0.02. On the other hand, for the simulation study, the value of COV is around 0.001 (having a mean area and standard deviation of 32.067 cm^2 and 0.0074 cm^2 respectively) as shown in Fig. 7.11b. The lengths of major and minor axes during the navigation process are also shown in Fig. 7.11a. From the result, it is observed that the virtual areas vary only slightly (and also within quite an acceptable limit) in the hardware experimentation. Moreover, to analyze the data (i.e. the evaluated areas of the virtual region) statistically (refer to Fig. 7.11c), it is seen that with an increasing number of agents in the team, the variations in areas of the virtual regions are being reduced. The pattern can be corroborated theoretically. The reason behind this is the fact that with the increasing number of agents, the number of sensing points in real-time increases. Consequently, the estimation error (positive/negative) associated with the sensing process of the robots gets almost nullified (ideal situation) and effectively decreases significantly (actual situation). So, we find that in our work we could establish a very strong statistical correlation between hardware experimentations and software simulations. Through these analyses, we can conclude that our proposed control strategy is satisfactorily immune to sensor noise. Or in other words, it seems that the control algorithm synthesized in this work is substantially robust so far as sensor noise is concerned.

7.4.3 Comparative analysis

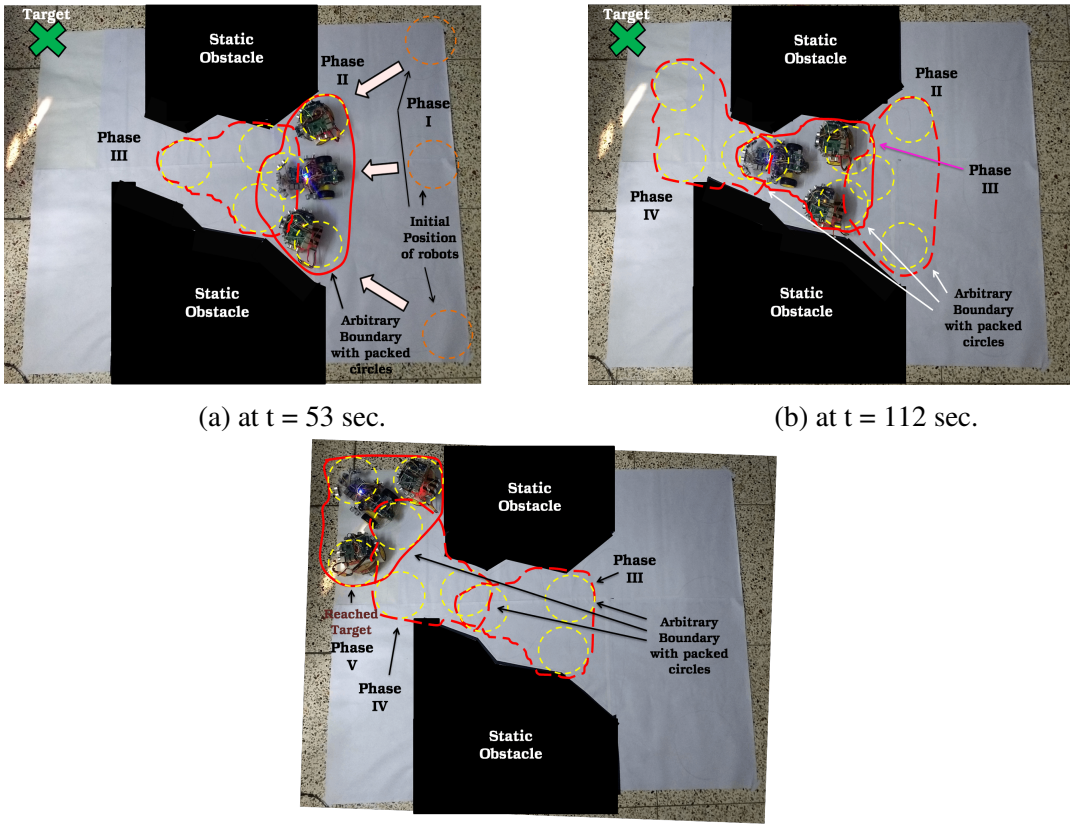
Table 7.6 Running time comparison with agent's scalability (**L-F = Leader-Follower**)

	Number of agents	**Vision based L-F [103] (in sec.)**	**Proposed scheme**	
			with L-F (in sec.)	**with Centralized controller (in sec.)**
Simulation	50	192	250	**150**
	100	396	503	**307**
	200	823	1022	**616**
	400	1722	2057	**1253**
Experiment	2	171	195	**143**
	3	291	342	**251**
	4	525	606	**433**
	5	732	858	**527**
	6	944	1029	**615**

To examine the effectiveness of the proposed strategy in both the simulations and hardware experiments, we assess the running time of the swarm for reaching the goal location with scalable agents in the team in an identical environment as depicted in Fig. 7.10. For the purpose of comparison, we borrow the concept of *Leader-Follower* framework [103] in our present work, where the adjustments of virtual regions are performed on a raspberry-pi (4GB of RAM) based leader robot. Hence, the

global controller resides inside that robot, so, no additional centralized controller is employed in this experiment. The results are shown in Table 7.6. From the outcomes, we can argue that though the leader-follower method can serve the purpose successfully, it consumes more time than the proposed centralized control approach.

Now, we are incorporating the hardware implementation result of the polygonal virtual-region approach in this next section.



(a) at t = 53 sec.

(b) at t = 112 sec.

(c) at t = 367 sec.

Fig. 7.12 Swarm-agent position trajectories in a real environment with the proposed control technique.

7.5 Hardware experimental results of polygonal virtual region approach

In this study, the same experimental steps as in the preceding section are used. However, as noted in Chapter 5, the main differentiating point is that the path of the robotic swarm is guided using arbitrarily shaped virtual polygonal regions with a circle packing scheme. In the previous case (Chapter 4), motion planning has been performed using geometric virtual regions (circle and ellipse).

The experimental implementation contains three robotic agents that are required to approach the target location. The detailed outcome is shown in Fig. 7.12.

Primarily (Phase I of Fig. 7.12a), starting from some randomized locations (*orange*-circles) all the robots form an arbitrary boundary (*red*-contour) with *three* packed-circles (*yellow*-disks) inside it in Phase II under the actuation of the global controller. At $t = 42$ sec, all the robots have arrived inside the allotted circles under the influence of the local controller to form a swarm-like architecture. Once reached inside the specific zones, a new arbitrary boundary is created based on the distance sensing information along with the packed circles (Phase III). At $t = 100$ sec (shown in Fig. 7.12b), all the robots have reached the assigned circular zone while preserving strong cohesiveness among them. After that a new boundary region is created along with *three* packed-circles at $t = 112$ sec (Phase IV). Finally, the entire group reaches the target zone successfully at $t = 367$ sec (Phase V) as represented in Fig. 7.12c.

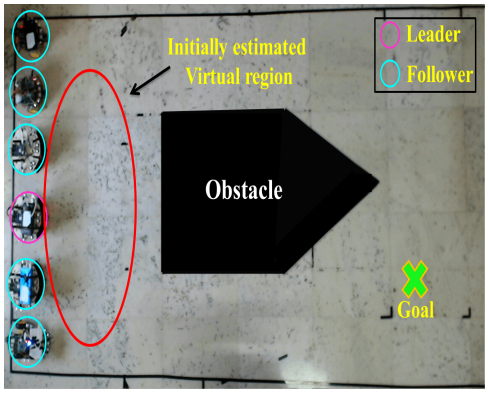
Finally, a similar hardware architecture has been utilized to study the efficacies of the split-join aspect in multi-robot navigational problems.

7.6 Hardware implementation results of splitting-merging approach

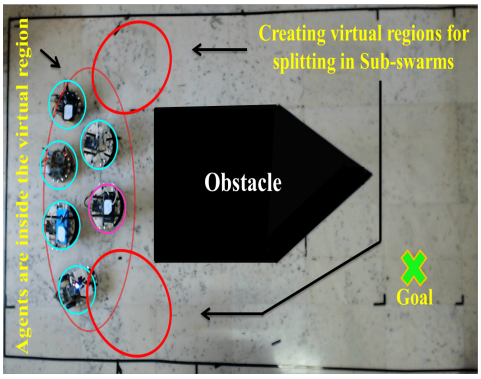
To assess the performance of the proposed strategy in a complex environment, we have carried out several experiments with the varying number of robots in the team. One such outcome is shown in Fig. 7.13. In this experiment, the environment has a dimension of 390×170 cm^2, containing static obstacles and boundaries with multiple tracks. Out of 6 UGVs, an arbitrary agent is selected as the lead agent (*magenta*-circle), and the rest are acting as followers (*cyan*-circle).

Starting from initial positions (as shown in Fig. 7.13a), all the agents are approaching (under the actuation of CC) a virtual region (*red*-ellipse) as etched by the lead agent. At $t = 36$ sec, all are assembled inside the virtual-region (refer Fig. 7.13b). After that, the lead robot detects two different lanes ((*red*-ellipses)) to move further. Thus, the control has been passed on to the centralized controller. As per the outcome of the central controller, two sub-swarms (each having a lead agent) have been formed. Now, the path-plan has been solely carried out by the respective leaders (Figs. 7.13c and 7.13d). At $t = 220$ sec, it is observed that the created virtual regions (*red*-dotted ellipse) are overlapped as both the tracks merge to a common passage. As a consequence, the two sub-swarms rejoin under the influence of the central controller (Fig. 7.13e). Finally, the entire system reaches the goal location at $t = 385$ sec successfully (Fig. 7.13f).

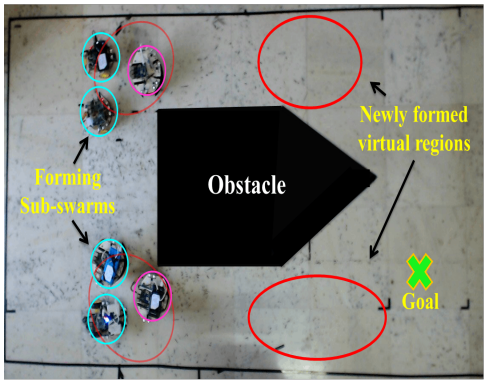
Plotting the lengths of the elliptical axes and areas of this experimental study (Fig. 7.14), it is observed that the splitting phase has been started around $t = 36$ sec. Before this time instance, the area of the virtual region was 3120 cm^2, the major and minor axis lengths were 48.75 cm and 20.4 cm respectively. As the repulsion range of each agent is around 11.3 cm, hence, we can infer that the agents do not produce a cohesive swarm as of now. After that, the sub-swarms are created with an



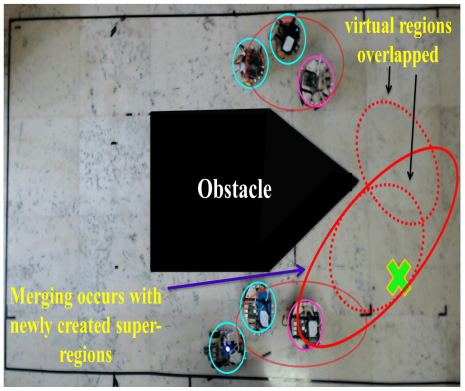
(a) at t = 0 sec.



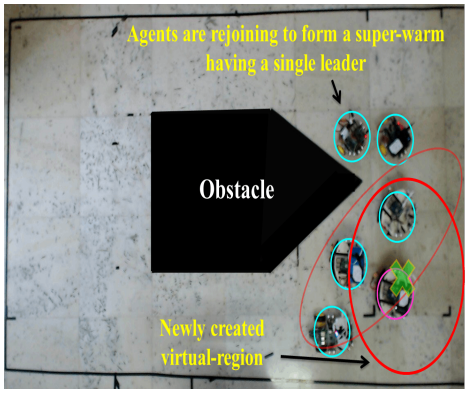
(b) at t = 36 sec.



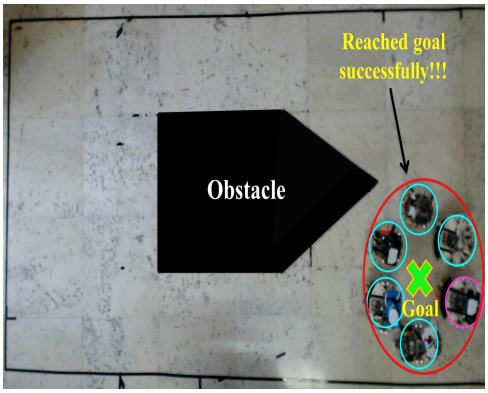
(c) at t = 55 sec.



(d) at t = 220 sec.



(e) at t = 262 sec.



(f) at t = 385 sec.

Fig. 7.13 Hardware implementation results of the proposed split-merge control approach with 6 agents in the team.

equal number of agents. So, the elliptical areas, as well as axis lengths of those sub-swarms, are found to be very close to each other. At $t = 260$ sec, the sub-swarms are initiating to rejoin for creating a super-swarm. Finally, when the super-swarm has reached the destination, the structural area of the virtual region is found to be around 2400 cm^2, thus producing a cohesive swarm.

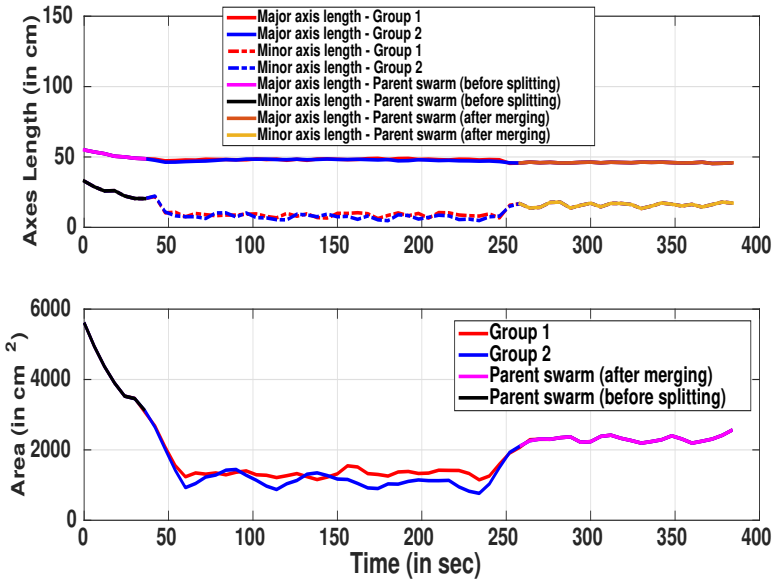


Fig. 7.14 Axis-lengths and the computed areas of the virtual-structures in the hardware experiment.

Table 7.7 Statistical analysis on the execution time with scalable agents for the hardware experiment

No. of Agents	Mean time (in sec)	std (in sec)	COV	1^{st} quartile (Q_1) (in sec)	3^{rd} quartile (Q_3) (in sec)	IQR ($Q_3 - Q_1$) (in sec)	Median (Q_2) (in sec)	Max. time (in sec)	Min time (in sec)
4	309.7	22.71	0.0733	289.84	328.86	39.02	309.53	350.53	270.71
5	349.56	16.11	0.0461	335.48	363.45	27.97	350.19	376.45	321.89
6	386.77	14.72	0.0381	373.74	399.19	25.45	386.58	411.96	362.02

Statistical analysis of hardware experiment

In the hardware experiment, we have created a sophisticated environment to validate our proposed methodology. The experimental studies have been carried out with four, five, and six robots in a similar environment as shown in Fig. 7.13. To verify the consequences as obtained from the statistical analysis of the simulation outcomes, we perform a similar study on the execution times with scalable agents in the swarm. The results are presented in Table 7.7, and Fig. 7.15.

From the outcomes of the analysis, we have witnessed that the mean and median execution times increase, and the standard deviation, coefficient of variation, and inter-quartile range decrease with the scalable agents of the swarm.

Therefore, from the outcomes, we can conclude that whether software simulations or real-world experiments, our analysis follows an identical pattern. The fundamental reason behind this fact is that

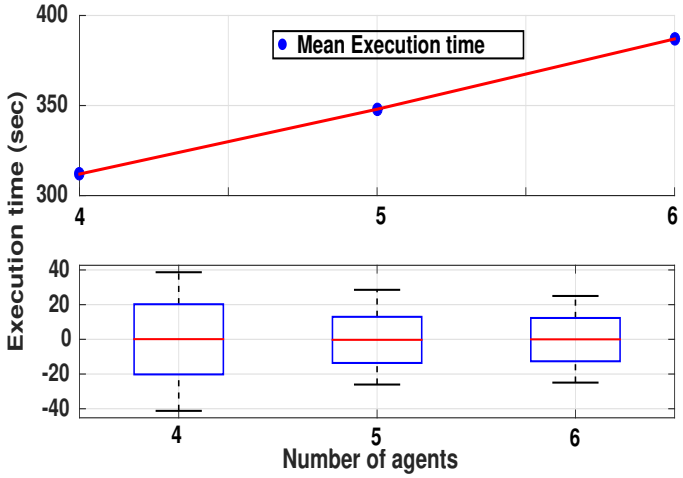


Fig. 7.15 Analysis on agents' scalability with execution time of hardware experiments.

as the number of sensing points increases with scalable agents, the leader agent evaluates optimal virtual regions in each step of movement. Thus, the same procedure would not be repeated in the next iteration. Subsequently, the standard deviation, coefficient of variation, and inter-quartile range reduce.

7.7 Limitation of the hardware experiments

The work could have been better presented if only we could arrange for a real swarm of robots, including at least 15 to 20 robots. The scalability of the proposed work could have been established in steps. Unfortunately, till now, we could fabricate only 6 robots (the time progression is one to three to four and then six) due to both temporal and cost constraints. The patterns (that we have observed in hardware simulation with these robots) have entailed us to strongly extrapolate the pattern of dynamics of a swarm of robots in real-time. This limitation in experimentation has tried to be traded off with the help of extensive simulation works and statistical analysis of the real-time behavior of the swarm. In this respect, we argue that from the experimental outcomes, it is apparent that our proposed solution would work in real-time too where the issue of scalability would be complied with satisfactorily.

7.8 Summary

In this chapter, we have implemented a hardware framework to test the proposed control strategies as discussed in Chapters 3, 4, 5, and 6. Several environments have been created with static and dynamic obstacles. Initially, we test the bat algorithm-based evolutionary approach. To measure the

adaptability of the proposed system, several complex dynamic environments have been created. The performance of the proposed system has been compared with the circle-based gathering approach and Binary bat scheme. After that, we have tested the virtual geometrical region approach with scalable agents in the swarm. The proposed scheme is compared with the traditional leader-follower approach. Then, we perform hardware experimentation of the polygonal region technique. Three-wheeled robots have been employed to test the viability of the proposed controller. Finally, the split-join aspect has been implemented with six mobile robots. Based on the outcomes and comparative studies, we can conclude that the proposed algorithms provide desired outcomes as per our expectations and outperform the existing techniques.

Chapter 8

Conclusion

In this dissertation, we have established a fault-tolerant cohesive swarm searching for a target or multiple targets while navigating through an unknown occluded environment. The proposed adaptive control laws can successfully drive the system towards the target point while avoiding obstacles during the exploration steps. The brief summaries of the thesis have been presented below.

8.1 Summary of the Dissertation

In Chapter 3, we have proposed a novel adaptive formation control strategy employing the bat algorithm-based evolutionary technique for a multi-robotic framework to handle the traditional obstacle avoidance problem. Three stabilized controllers have been formulated to trigger simultaneously on each agent in order to approach the target autonomously (i.e., without any human action). Moreover, to dodge the nearest (static/ dynamic) obstacles, decentralized self-organized flexible patterns are developed among all the agents in the group for accomplishing strong inter-agent cohesiveness. Our extensive simulation results and the hardware implementations demonstrate the viability of the proposed control scheme. From the discussions, we have identified several interesting behavioral patterns of the bat algorithm over the other well-known swarm intelligence techniques.

In Chapter 4, we have employed the concept of the geometrical region-based shape control technique of a swarm of robots to avoid obstacles in an unknown environment. To realize this objective, a virtual circular region is considered initially for accommodating all the agents having a strong inter-agent cohesion inside the contour. For dodging obstructions, the virtual region is further modified to an ellipse of an equivalent circular area. All of the agents are required to position themselves inside the created region (circle or ellipse) to attain strong cohesion. The proposed system can also handle the agent failure condition during the navigational process. To illustrate the performance of the proposed control actions, extensive simulation results are provided in various scenarios. The comparison results demonstrate the effectiveness of the proposed scheme. Furthermore, this technique is also examined in real environments with a team of two-wheeled mobile robots.

In Chapter 5, we have proposed an arbitrarily shaped polygonal region-based control technique for avoiding obstacles in an unknown challenging environment. To evade the obstacles, more specifically in the narrow pathways and corners, an arbitrarily shaped virtual region is initially created. To accommodate the agents inside the virtual boundary, several identical circular regions are packed employing the circle packing strategy. Finally, circles are allotted to each of the agents for forcing the robots toward the assigned circles. As per the outcome of the proposed approach, we observe strong inter-agent cohesiveness among the agents during the navigation. To illustrate the performance of the proposed controllers, simulation results along with hardware implementation have been presented.

In Chapter 6, we have conceptualized a novel approach to perform the split-join aspect of a robotic swarm. The region-based shape control scheme has been linked with the leader-follower methodology to achieve the splitting and merging of a swarm of robots. The proposed control method can effectively be applied in multi-target as well as multi-path scenarios. Moreover, in the face of the incoming dynamic obstacles, the swarm can break into sub-swarms autonomously. Two sub-swarms moving along the same path can rejoin to form a super-swarm (if required) utilizing the offered procedure. To synthesize the control scheme, it is verified that the proposition would be sufficiently scalable in terms of the number of agents in the swarm, sub-swarm, and super-swarm. Furthermore, within a team, all the agents will maintain cohesiveness among themselves during the navigational steps. To illustrate the performance of the proposed control action, simulation results along with hardware experimentation have been performed in various scenarios. The performance analysis shows the efficacies of the proposed scheme.

Finally, in Chapter 7, we have presented the hardware framework to test the proposed control strategies as discussed in Chapters 3, 4, 5, and 6. Several environments have been created with static and dynamic obstacles. Initially, we have tested the bat algorithm-based evolutionary approach. To measure the adaptability of the proposed system, several complex dynamic environments have been created. The performance of the proposed system has been compared with the circle-based gathering approach and Binary bat scheme. After that, we have tested the virtual geometrical region approach with scalable agents in the swarm. The proposed scheme is compared with the traditional leader-follower approach. Then, we perform hardware experimentation of the polygonal region technique. Three-wheeled robots have been employed to test the viability of the proposed controller. Finally, the split-join aspect has been implemented with six mobile robots. Based on the outcomes and comparative studies, we can conclude that the proposed algorithms provide desired outcomes as per our expectations and outperform the existing techniques.

In this context, the contributions of this thesis are reiterated as follows:

8.2 Contributions

1. In the initial work of this thesis, we propose a bat algorithm-based evolutionary technique for a multi-robot system for achieving a fixed mission. In this aspect, we have formulated

three dedicated controllers to perform the following operations during the navigation. 1). approaching the target, 2). avoiding static and dynamic obstacles, 3). maintaining inter-agent cohesiveness. The stability of the proposed controllers is studied with the help of a Lyapunov candidate function. The simulation results, comparative analyses, and hardware outcomes demonstrate the efficacy of the proposed scheme.

2. The evolutionary approaches generally consume a large amount of time to compute, hence with a scalable swarm, it is very challenging to complete the mission in a stipulated time. So, to overcome this limitation, in our next work, we propose a novel region-based shape control approach for the path planning of a swarm of robots. In this control scheme, the trajectories of the agents have been controlled by either a virtual circular region or a virtual elliptical region. To alter the inter-agent formation depending on the environmental condition, a spanning-tree assisted shape matching algorithm has been offered. The proposed framework can handle the situation well in the face of agent failure, thus making the system fault-tolerant. The simulation results, comparative analyses, and the hardware experimental outcomes confirm the robustness, scalability, as well as flexibility of the swarm arrangement.

3. Despite the optimal characteristics of the previous approach, the inter-agent cohesiveness deteriorates significantly in constricted environments or narrow pathways where the shape of the virtual ellipse is further expanded along the major axis while the minor axis is shrunken and, in an extreme scenario, may culminate into a line-formation. To overcome such limitations, in our next work, the elliptical region has been replaced by an arbitrarily shaped polygonal virtual region under the assumption that more agents can be accommodated inside the newly fitted virtual region. The proposed methodology enhances the inter-agent cohesion in narrow corridors, and the framework is scalable in terms of an increasing number of robotic agents. The hardware and software outcomes demonstrate the viability of the proposed approach in terms of scalability, flexibility as well as strong inter-agent cohesiveness.

4. Finally, we have addressed the navigational issue of a swarm of robots in multi-path/multi-target scenarios. Here, we adopt the elliptical region-based shape control technique and the traditional leader-follower scheme. During the navigation, if the leader robot detects multiple pathways to proceed further, the parent swarm will be broken down into several sub-swarms, thus, splitting will occur. Furthermore, it might so happen that after a while two or more paths would merge and create a single common passage. In that situation, the sub-swarms, functioning in those paths, will be routed through those converging paths to ultimately merge/rejoin. This approach also ensures robustness, scalability, flexibility, and strong inter-agent cohesiveness.

5. All of the above-mentioned approaches are simulated in software settings with scalable agents in the team. Moreover, to validate the efficacies of the proposed schemes, several real-time experimentations have been performed with raspberry-pi controlled two-wheeled mobile robots, namely 'SonPI' having Wi-Fi (802.11) communication protocol, equipped with an array of

five-sonar sensors. Our experimental outcomes provide the efficacies of each of the proposed approaches.

As per the outcomes of the software and hardware experiments, we can conclude that the proposed scheme could effectively be used in swarm robotics path planning problems.

8.3 Future Directions

Several problems remain open in the research of this dissertation which are listed below.

1. In this dissertation, we mostly focus on the adaptive behavior of a swarm of robots for avoiding obstacles inside an unknown occluded environment. However, in some application areas such as carrying an object from a source location to the destination employing a robotics swarm, strict formation control is utterly required. The region-based shaped control strategy could be used in order to achieve tight formation amongst the robots for transporting an object from one location to another.

2. We have not specified or tested the performance of the swarm in the presence of a noisy environment, such as the deep sea or deep space, in this research. In these types of situations, each agent in the swarm should have an additional control law to avoid the noise. This work could be expanded further in light of such circumstances.

3. Last but not the least, we have primarily used two-dimensional scenarios such as wheel robots to evaluate our proposed algorithms. These strategies, on the other hand, might be useful in three-dimensional circumstances, such as with drones or quadcopters.

References

[1] K Ota. Robocue, the tokyo fire department's rescue-bot. *Popular Science Magazine*, 3, 2011.

[2] Gokarna Sharma, Costas Busch, Supratik Mukhopadhyay, and Charles Malveaux. Tight analysis of a collisionless robot gathering algorithm. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 12(1):1–20, 2017.

[3] Haopeng Zhang. A binary cooperative bat algorithm based optimal topology design of leader–follower consensus. *ISA transactions*, 96:51–59, 2020.

[4] Elena Garcia, Maria Antonia Jimenez, Pablo Gonzalez De Santos, and Manuel Armada. The evolution of robotics research. *IEEE Robotics & Automation Magazine*, 14(1):90–103, 2007.

[5] Elio Tuci, Muhanad HM Alkilabi, and Otar Akanyeti. Cooperative object transport in multi-robot systems: A review of the state-of-the-art. *Frontiers in Robotics and AI*, 5:59, 2018.

[6] Zool Hilmi Ismail and Nohaidda Sariff. A survey and analysis of cooperative multi-agent robot systems: challenges and directions. In *Applications of Mobile Robots*. IntechOpen, 2018.

[7] Guang-Zhong Yang, Jim Bellingham, Pierre E Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, et al. The grand challenges of science robotics. *Science robotics*, 3(14):eaar7650, 2018.

[8] Kamal Asgar and Floyd A Peyton. Effect of casting conditions on some mechanical properties of cobalt-base alloys. *Journal of Dental Research*, 40(1):73–86, 1961.

[9] Robert U Ayres and Steven M Miller. Robotics: Applications and social implications. 1983.

[10] Lynne E Parker and John V Draper. Robotics applications in maintenance and repair. *Handbook of industrial robotics*, 2:1023–1036, 1998.

[11] Luigi Alfredo Grieco, Alessandro Rizzo, Simona Colucci, Sabrina Sicari, Giuseppe Piro, Donato Di Paola, and Gennaro Boggia. Iot-aided robotics applications: Technological implications, target domains and open issues. *Computer Communications*, 54:32–47, 2014.

[12] Vikas Panwar, Naveen Kumar, Nagarajan Sukavanam, and Jin-Hwan Borm. Adaptive neural controller for cooperative multiple robot manipulator system manipulating a single rigid object. *Applied Soft Computing*, 12(1):216–227, 2012.

[13] Suresh P Sethi, Jeffrey B Sidney, and Chelliah Sriskandarajah. Scheduling in dual gripper robotic cells for productivity gains. *IEEE Transactions on Robotics and Automation*, 17(3):324–341, 2001.

[14] MA CAMBRIDGE. Swarms of Underwater Drones Deliver the Goods in Naval Exercise. https://www.draper.com/news-releases/swarms-underwater-drones-deliver-goods-naval-exercise, 2018. [Online; accessed 13-September-2018].

[15] Jean-Christophe Baillie, Akim Demaille, Quentin Hocquet, Matthieu Nottale, and Samuel Tardieu. The urbi universal platform for robotics. In *First International Workshop on Standards and Common Platform for Robotics*, 2008.

[16] Alan K Mackworth. On seeing robots. In *Computer Vision: Systems, Theory and Applications*, pages 1–13. World Scientific, 1993.

[17] Heiko Hamann. *Swarm robotics: A formal approach*. Springer, 2018.

[18] Levent Bayindir and Erol Şahin. A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering & Computer Sciences*, 15(2):115–147, 2007.

[19] Eugene Kagan, Nir Shvalb, and Irad Ben-Gal. *Autonomous Mobile Robots and Multi-Robot Systems: Motion-Planning, Communication, and Swarming*. John Wiley & Sons, 2019.

[20] Anne Fünfhaus, Josefine Göbel, Julia Ebeling, Henriette Knispel, Eva Garcia-Gonzalez, and Elke Genersch. Swarming motility and biofilm formation of paenibacillus larvae, the etiological agent of american foulbrood of honey bees (apis mellifera). *Scientific reports*, 8(1):1–12, 2018.

[21] Fasheng Zou, Harrison Jones, Demeng Jiang, Tien-Ming Lee, Ari Martínez, Kathryn Sieving, Min Zhang, Qiang Zhang, Eben Goodale, et al. The conservation implications of mixed-species flocking in terrestrial birds, a globally-distributed species interaction network. *Biological conservation*, 224:267–276, 2018.

[22] Intesaaf Ashraf, Hanaé Bradshaw, Thanh-Tung Ha, José Halloy, Ramiro Godoy-Diana, and Benjamin Thiria. Simple phalanx pattern leads to energy saving in cohesive fish schooling. *Proceedings of the National Academy of Sciences*, 114(36):9599–9604, 2017.

[23] Adam Slowik and Halina Kwasnicka. Nature inspired methods and their industry applications—swarm intelligence algorithms. *IEEE Transactions on Industrial Informatics*, 14(3):1004–1015, 2017.

[24] Erol Şahin, Sertan Girgin, Levent Bayindir, and Ali Emre Turgut. Swarm robotics. In *Swarm intelligence*, pages 87–100. Springer, 2008.

[25] Gerardo Beni. The concept of cellular robotic system. In *Proceedings IEEE International Symposium on Intelligent Control 1988*, pages 57–62. IEEE, 1988.

[26] Toshio Fukuda and Seiya Nakagawa. Approach to the dynamically reconfigurable robotic system. *Journal of Intelligent and Robotic Systems*, 1(1):55–72, 1988.

[27] C Ronald Kube and Hong Zhang. Collective robotics: From social insects to robots. *Adaptive behavior*, 2(2):189–218, 1993.

[28] Geoffroy De Schutter, Guy Theraulaz, and Jean-Louis Deneubourg. Animal–robots collective intelligence. *Annals of Mathematics and Artificial Intelligence*, 31(1):223–238, 2001.

[29] Gerardo Beni. From swarm intelligence to swarm robotics. In *International Workshop on Swarm Robotics*, pages 1–9. Springer, 2004.

[30] Anders Lyhne Christensen, Rehan OGrady, and Marco Dorigo. From fireflies to fault-tolerant swarms of robots. *IEEE Transactions on Evolutionary Computation*, 13(4):754–766, 2009.

[31] Xinwu Liang, Hesheng Wang, Yun-Hui Liu, Weidong Chen, and Tao Liu. Formation control of nonholonomic mobile robots without position and velocity measurements. *IEEE Transactions on Robotics*, 34(2):434–446, 2017.

[32] Yujuan Wang, Yongduan Song, and Frank L Lewis. Robust adaptive fault-tolerant control of multiagent systems with uncertain nonidentical dynamics and undetectable actuation failures. *IEEE Transactions on Industrial Electronics*, 62(6):3978–3988, 2015.

[33] Yongzhao Hua, Xiwang Dong, Qingdong Li, and Zhang Ren. Distributed fault-tolerant time-varying formation control for high-order linear multi-agent systems with actuator failures. *ISA transactions*, 71:40–50, 2017.

[34] Horst F Wedde, Muddassar Farooq, and Yue Zhang. Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 83–94. Springer, 2004.

[35] Joshua P Hecker and Melanie E Moses. Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms. *Swarm Intelligence*, 9(1):43–70, 2015.

[36] Jan Dyre Bjerknes and Alan FT Winfield. On fault tolerance and scalability of swarm robotic systems. In *Distributed autonomous robotic systems*, pages 431–444. Springer, 2013.

[37] Marco Dorigo, Mauro Birattari, and Manuele Brambilla. Swarm robotics. *Scholarpedia*, 9(1):1463, 2014.

[38] Xinye Chen, Ping Zhang, Guanglong Du, and Fang Li. A distributed method for dynamic multi-robot task allocation problems with critical time constraints. *Robotics and Autonomous Systems*, 118:31–46, 2019.

[39] Evangelos Papadopoulos, Iosif Paraskevas, and Thaleia Flessa. Miniaturization and micro/nanotechnology in space robotics. In *Nanorobotics*, pages 69–92. Springer, 2013.

[40] Kikuo Fujimura. Path planning with multiple objectives. *IEEE Robotics & Automation Magazine*, 3(1):33–38, 1996.

[41] Yang Xue and Jian-Qiao Sun. Solving the path planning problem in mobile robotics with the multi-objective evolutionary algorithm. *Applied Sciences*, 8(9):1425, 2018.

[42] DD Šiljak. Decentralized control and computations: status and prospects. *Annual Reviews in Control*, 20:131–141, 1996.

[43] Dejan Milutinović and Pedro Lima. Modeling and optimal centralized control of a large-size robotic population. *IEEE Transactions on Robotics*, 22(6):1280–1285, 2006.

[44] Hai Zhu, Jelle Juhl, Laura Ferranti, and Javier Alonso-Mora. Distributed multi-robot formation splitting and merging in dynamic environments. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9080–9086. IEEE, 2019.

[45] Dongyu Li, Shuzhi Sam Ge, Wei He, Guangfu Ma, and Lihua Xie. Multilayer formation control of multi-agent systems. *Automatica*, 109:108558, 2019.

[46] Gábor Vásárhelyi, Cs Virágh, Gergo Somorjai, Norbert Tarcai, Tamás Szörényi, Tamás Nepusz, and Tamás Vicsek. Outdoor flocking and formation flight with autonomous aerial robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3866–3873. IEEE, 2014.

[47] Sergio Monteiro and Estela Bicho. A dynamical systems approach to behavior-based formation control. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 3, pages 2606–2611. IEEE, 2002.

[48] Javier Alonso-Mora, Stuart Baker, and Daniela Rus. Multi-robot formation control and object transport in dynamic environments via constrained optimization. *The International Journal of Robotics Research*, 36(9):1000–1021, 2017.

[49] Zhouhua Peng, Dan Wang, and Xiaojing Hu. Robust adaptive formation control of underactuated autonomous surface vehicles with uncertain dynamics. *IET control theory & applications*, 5(12):1378–1387, 2011.

[50] Bong Seok Park, Jin Bae Park, and Yoon Ho Choi. Adaptive formation control of electrically driven nonholonomic mobile robots with limited information. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(4):1061–1075, 2011.

[51] Veysel Gazi and Kevin M Passino. Stability analysis of swarms. *IEEE transactions on automatic control*, 48(4):692–697, 2003.

[52] Yanfei Liu. *Cohesive behaviors of cooperative multiagent systems with information flow constraints*. PhD thesis, The Ohio State University, 2004.

[53] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. Multi-robot task allocation: A review of the state-of-the-art. In *Cooperative Robots and Sensor Networks 2015*, pages 31–51. Springer, 2015.

[54] Kristina Lerman, Chris Jones, Aram Galstyan, and Maja J Matarić. Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research*, 25(3):225–241, 2006.

[55] Fang Tang and Lynne E Parker. A complete methodology for generating multi-robot task solutions using asymtre-d and market-based task allocation. In *Proceedings 2007 IEEE international conference on robotics and automation*, pages 3351–3358. IEEE, 2007.

[56] G Ayorkor Korsah, Anthony Stentz, and M Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[57] Michael Otte, Michael J Kuhlman, and Donald Sofge. Auctions for multi-robot task allocation in communication limited environments. *Autonomous Robots*, 44(3):547–584, 2020.

[58] Maryam Khani, Ali Ahmadi, and Hajar Hajary. Distributed task allocation in multi-agent environments using cellular learning automata. *Soft Computing*, 23(4):1199–1218, 2019.

[59] Michal Kudelski, Luca M Gambardella, and Gianni A Di Caro. Robonetsim: An integrated framework for multi-robot and network simulation. *Robotics and Autonomous Systems*, 61(5):483–496, 2013.

[60] Micael S Couceiro, Patricia A Vargas, and Rui P Rocha. Bridging the reality gap between the webots simulator and e-puck robots. *Robotics and Autonomous Systems*, 62(10):1549–1567, 2014.

[61] Y Uny Cao, Alex S Fukunaga, and Andrew Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous robots*, 4(1):7–27, 1997.

[62] Lorenzo Rosa, Marco Cognetti, Andrea Nicastro, Pol Alvarez, and Giuseppe Oriolo. Multi-task cooperative control in a heterogeneous ground-air robot team. *IFAC-PapersOnLine*, 48(5):53–58, 2015.

[63] Pablo Gil, Jorge Pomares, S vT Puente C Diaz, F Candelas, and F Torres. Flexible multi-sensorial system for automatic disassembly using cooperative robots. *International Journal of Computer Integrated Manufacturing*, 20(8):757–772, 2007.

[64] Yongtae Do and Jongman Kim. Infrared range sensor array for 3d sensing in robotic applications. *International Journal of Advanced Robotic Systems*, 10(4):193, 2013.

[65] Naomi Gildert, Alan G Millard, Andrew Pomfret, and Jon Timmis. The need for combining implicit and explicit communication in cooperative robotic systems. *Frontiers in Robotics and AI*, 5:65, 2018.

[66] Francesco Mondada, Edoardo Franzi, and Paolo Ienne. Mobile robot miniaturisation: A tool for investigation in control algorithms. In *Experimental robotics III*, pages 501–513. Springer, 1994.

[67] Koyippilly S Keerthi, Bandana Mahapatra, and Varun Girijan Menon. Into the world of underwater swarm robotics: Architecture, communication, applications and challenges. *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)*, 13(2):110–119, 2020.

[68] Ryuma Maeda, Takahiro Endo, and Fumitoshi Matsuno. Decentralized navigation for heterogeneous swarm robots with limited field of view. *IEEE Robotics and Automation Letters*, 2(2):904–911, 2017.

[69] Junyan Hu, Hanlin Niu, Joaquin Carrasco, Barry Lennox, and Farshad Arvin. Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. 2015.

[70] Yi Wei Daniel Tay, Biranchi Panda, Suvash Chandra Paul, Nisar Ahamed Noor Mohamed, Ming Jen Tan, and Kah Fai Leong. 3d printing trends in building and construction industry: a review. *Virtual and Physical Prototyping*, 12(3):261–276, 2017.

[71] Robin R Murphy, Satoshi Tadokoro, and Alexander Kleiner. Disaster robotics. In *Springer handbook of robotics*, pages 1577–1604. Springer, 2016.

[72] Ross D Arnold, Hiroyuki Yamaguchi, and Toshiyuki Tanaka. Search and rescue with autonomous flying robots through behavior-based cooperative intelligence. *Journal of International Humanitarian Action*, 3(1):18, 2018.

[73] Miguel Duarte, Vasco Costa, Jorge Gomes, Tiago Rodrigues, Fernando Silva, Sancho Moura Oliveira, and Anders Lyhne Christensen. Unleashing the potential of evolutionary swarm robotics in the real world. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pages 159–160, 2016.

[74] Weihua Sheng, Qingyan Yang, Jindong Tan, and Ning Xi. Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, 54(12):945–955, 2006.

[75] A Hemanth Reddy, Balla Kalyan, and Ch SN Murthy. Mine rescue robot system–a review. *Procedia Earth and Planetary Science*, 11:457–462, 2015.

[76] Dario Albani, Joris IJsselmuiden, Ramon Haken, and Vito Trianni. Monitoring and mapping with robot swarms for agricultural applications. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[77] Vijay Kumar and Nathan Michael. Opportunities and challenges with autonomous micro aerial vehicles. In *Robotics Research*, pages 41–58. Springer, 2017.

[78] Brad. Lendon. U.S. Navy could 'swarm' foes with robot boats. https://edition.cnn.com/2014/10/06/tech/innovation/navy-swarm-boats/index.html/, 2014. [Online; accessed 12-October-2014].

[79] Alexis C. Madrigal. Now the improvised explosive devices will find our warfighters. https://www.nextgov.com/emerging-tech/2018/03/drone-swarms-are-going-be-terrifying-and-hard-stop/146491/, 2018. [Online; accessed 08-March-2018].

[80] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[81] Guillaume Tochon, Jocelyn Chanussot, Mauro Dalla Mura, and Andrea L Bertozzi. Object tracking by hierarchical decomposition of hyperspectral video sequences: Application to chemical gas plume tracking. *IEEE Transactions on Geoscience and Remote Sensing*, 55(8):4567–4585, 2017.

[82] Matthew Budd, Bruno Lacerda, Paul Duckworth, Andrew West, Barry Lennox, and Nick Hawes. Markov decision processes with unknown state feature values for safe exploration using gaussian processes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[83] Daniel OB Jones, Andrew R Gates, Veerle AI Huvenne, Alexander B Phillips, and Brian J Bett. Autonomous marine environmental monitoring: Application in decommissioned oil fields. *Science of the total environment*, 668:835–853, 2019.

[84] Tomas Baca, Giuseppe Loianno, and Martin Saska. Embedded model predictive control of unmanned micro aerial vehicles. In *2016 21st international conference on methods and models in automation and robotics (MMAR)*, pages 992–997. IEEE, 2016.

[85] Shengkai Zhang, Wei Wang, and Tao Jiang. Wifi-inertial indoor pose estimation for micro aerial vehicles. *IEEE Transactions on Industrial Electronics*, 2020.

[86] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011.

[87] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, 35(4):287–300, 2013.

[88] Jan Faigl, Tomáš Krajník, Jan Chudoba, Libor Přeučil, and Martin Saska. Low-cost embedded system for relative localization in robotic swarms. In *2013 IEEE International Conference on Robotics and Automation*, pages 993–998. IEEE, 2013.

[89] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine*, 13(3):108–117, 2006.

[90] Guilherme AS Pereira, Mario FM Campos, and Vijay Kumar. Decentralized algorithms for multi-robot manipulation via caging. *The International Journal of Robotics Research*, 23(7-8):783–795, 2004.

[91] Zheng Pan, Shuai Liu, and Weina Fu. A review of visual moving target tracking. *Multimedia Tools and Applications*, 76(16):16989–17018, 2017.

[92] Graham Hunt, Faidon Mitzalis, Talib Alhinai, Paul A Hooper, and Mirko Kovac. 3d printing with flying robots. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 4493–4499. IEEE, 2014.

[93] Rosotics | The Future of Additive Manufacturing. https://www.rosotics.com/, 2019. [Online; accessed 01-January-2019].

[94] TECHNOLOGY. https://www.rosotics.com/technology, 2019. [Online; accessed 01-January-2019].

[95] Zhihua Qiao, Junzhe Zhang, Xiaoguang Qu, and Jiandong Xiong. Dynamic self-organizing leader-follower control in a swarm mobile robots system under limited communication. *IEEE Access*, 8:53850–53856, 2020.

[96] Dhruv Shah and Leena Vachhani. Swarm aggregation without communication and global positioning. *IEEE Robotics and Automation Letters*, 4(2):886–893, 2019.

[97] Maryam Yarmohamadi, H Haj Seyyed Javadi, and Hossein Erfani. Improvement of robot path planning using particle swarm optimization in dynamic environments with mobile obstacles and target. *Advanced Studies in Biology*, 3(1):43–53, 2011.

[98] PK Das and Prabir Kumar Jena. Multi-robot path planning using improved particle swarm optimization algorithm through novel evolutionary operators. *Applied Soft Computing*, page 106312, 2020.

[99] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *IEEE transactions on robotics and automation*, 14(6):926–939, 1998.

[100] Dongdong Xu, Xingnan Zhang, Zhangqing Zhu, Chunlin Chen, and Pei Yang. Behavior-based formation control of swarm robots. *mathematical Problems in Engineering*, 2014, 2014.

[101] Estela Bicho. *Dynamic Approach to Behavior-Based Robotics: design, specification, analysis, simulation and implementation*. Shaker Verlag, 2000.

[102] Luca Consolini, Fabio Morbidi, Domenico Prattichizzo, and Mario Tosques. Leader–follower formation control of nonholonomic mobile robots with input constraints. *Automatica*, 44(5):1343–1349, 2008.

[103] Hesheng Wang, Dejun Guo, Xinwu Liang, Weidong Chen, Guoqiang Hu, and Kam K Leang. Adaptive vision-based leader–follower formation control of mobile robots. *IEEE Transactions on Industrial Electronics*, 64(4):2893–2902, 2016.

[104] Nathan Sorensen and Wei Ren. A unified formation control scheme with a single or multiple leaders. In *2007 American Control Conference*, pages 5412–5418. IEEE, 2007.

[105] Yang Xu, Dongyu Li, Delin Luo, Yancheng You, and Haibin Duan. Two-layer distributed hybrid affine formation control of networked euler–lagrange systems. *Journal of the Franklin Institute*, 356(4):2172–2197, 2019.

[106] Wei Ren and Nathan Sorensen. Distributed coordination architecture for multi-robot formation control. *Robotics and Autonomous Systems*, 56(4):324–333, 2008.

[107] Jing Liu, Jian-an Fang, Zhen Li, and Guang He. Formation control with multiple leaders via event-triggering transmission strategy. *International Journal of Control, Automation and Systems*, 17(6):1494–1506, 2019.

[108] Junyan Hu, Parijat Bhowmick, and Alexander Lanzon. Distributed adaptive time-varying group formation tracking for multiagent systems with multiple leaders on directed graphs. *IEEE Transactions on Control of Network Systems*, 7(1):140–150, 2019.

[109] Jinxin Zhang and Housheng Su. Time-varying formation for linear multi-agent systems based on sampled data with multiple leaders. *Neurocomputing*, 339:59–65, 2019.

[110] Kun Cao, Xiuxian Li, and Lihua Xie. Preview-based discrete-time dynamic formation control over directed networks via matrix-valued laplacian. *IEEE Transactions on Cybernetics*, 50(3):1251–1263, 2019.

[111] Lvlong He, Peng Bai, Xiaolong Liang, Jiaqiang Zhang, and Weijia Wang. Feedback formation control of uav swarm with multiple implicit leaders. *Aerospace Science and Technology*, 72:327–334, 2018.

[112] Mohd Nadhir Ab Wahab, Samia Nefti-Meziani, and Adham Atyabi. A comprehensive review of swarm optimization algorithms. *PloS one*, 10(5):e0122827, 2015.

[113] Lijue Liu, Shuning Luo, Fan Guo, and Shiyang Tan. Multi-point shortest path planning based on an improved discrete bat algorithm. *Applied Soft Computing*, 95:106498, 2020.

[114] Hyondong Oh, Ataollah Ramezan Shirazi, Chaoli Sun, and Yaochu Jin. Bio-inspired self-organising multi-robot pattern formation: A review. *Robotics and Autonomous Systems*, 91:83–100, 2017.

[115] Qiang Lu, Qing-Long Han, Botao Zhang, Dongliang Liu, and Shirong Liu. Cooperative control of mobile sensor networks for environmental monitoring: An event-triggered finite-time control scheme. *IEEE transactions on cybernetics*, 47(12):4134–4147, 2016.

[116] Hanzhen Xiao, CLP Chen, and Dengxiu Yu. Two-level structure swarm formation system with self-organized topology network. *Neurocomputing*, 384:356–367, 2020.

[117] Xiangyu Wang, Shihua Li, Xinghuo Yu, and Jun Yang. Distributed active anti-disturbance consensus for leader-follower higher-order multi-agent systems with mismatched disturbances. *IEEE Transactions on Automatic Control*, 62(11):5795–5801, 2016.

[118] Sendren Sheng-Dong Xu, Hsu-Chih Huang, Yu-Chieh Kung, and Shao-Kang Lin. Collision-free fuzzy formation control of swarm robotic cyber-physical systems using a robust orthogonal firefly algorithm. *IEEE Access*, 7:9205–9214, 2019.

[119] Konstantin Böttinger. Hunting bugs with lévy flight foraging. In *2016 IEEE Security and Privacy Workshops (SPW)*, pages 111–117. IEEE, 2016.

[120] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, 1987.

[121] David E Goldberg and John Henry Holland. Genetic algorithms and machine learning. 1988.

[122] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[123] Sandeep Kumar, Sanjay Jain, and Harish Sharma. Genetic algorithms. *Advances in swarm intelligence for optimizing problems in computer science*, pages 27–52, 2018.

[124] Anand Nayyar, Dac-Nhuong Le, and Nhu Gia Nguyen. *Advances in swarm intelligence for optimizing problems in computer science*. CRC Press, 2018.

[125] Robert Martin C Santiago, Anton Louise De Ocampo, Aristotle T Ubando, Argel A Bandala, and Elmer P Dadios. Path planning for mobile robots using genetic algorithm and probabilistic roadmap. In *2017IEEE 9th international conference on humanoid, nanotechnology, information technology, communication and control, environment and management (HNICEM)*, pages 1–5. IEEE, 2017.

[126] Mansoor Davoodi, Fatemeh Panahi, Ali Mohades, and Seyed Naser Hashemi. Multi-objective path planning in discrete space. *Applied Soft Computing*, 13(1):709–720, 2013.

[127] Faez Ahmed and Kalyanmoy Deb. Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms. *Soft Computing*, 17(7):1283–1299, 2013.

[128] Amir Hossein Karami and Maryam Hasanzadeh. An adaptive genetic algorithm for robot motion planning in 2d complex environments. *Computers & Electrical Engineering*, 43:317–329, 2015.

[129] Shashi Mittal and Kalyanmoy Deb. Three-dimensional offline path planning for uavs using multiobjective evolutionary algorithms. In *2007 IEEE Congress on Evolutionary Computation*, pages 3195–3202. IEEE, 2007.

[130] Jaesung Lee and Dae-Won Kim. An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph. *Information Sciences*, 332:1–18, 2016.

[131] Chaymaa Lamini, Said Benhlima, and Ali Elbekri. Genetic algorithm based approach for autonomous mobile robot path planning. *Procedia Computer Science*, 127:180–189, 2018.

[132] Mohamed Elhoseny, Alaa Tharwat, and Aboul Ella Hassanien. Bezier curve based path planning in a dynamic field using modified genetic algorithm. *Journal of Computational Science*, 25:339–350, 2018.

[133] Michael Brand, Michael Masuda, Nicole Wehner, and Xiao-Hua Yu. Ant colony optimization algorithm for robot path planning. In *2010 international conference on computer design and applications*, volume 3, pages V3–436. IEEE, 2010.

[134] Fatin Hassan Ajeil, Ibraheem Kasim Ibraheem, Ahmad Taher Azar, and Amjad J Humaidi. Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments. *Sensors*, 20(7):1880, 2020.

[135] Dehui Zhang, Xiaoming You, Sheng Liu, and Han Pan. Dynamic multi-role adaptive collaborative ant colony optimization for robot path planning. *IEEE Access*, 8:129958–129974, 2020.

[136] Rafael S Parpinelli, Heitor S Lopes, and Alex Alves Freitas. Data mining with an ant colony optimization algorithm. *IEEE transactions on evolutionary computation*, 6(4):321–332, 2002.

[137] N Buniyamin, N Sariff, WAJ Wan Ngah, and Z Mohamad. Robot global path planning overview and a variation of ant colony system algorithm. *International journal of mathematics and computers in simulation*, 5(1):9–16, 2011.

[138] MA Porta Garcia, Oscar Montiel, Oscar Castillo, Roberto Sepúlveda, and Patricia Melin. Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. *Applied Soft Computing*, 9(3):1102–1110, 2009.

[139] A Mallikarjuna Rao, K Ramji, and BSK Sundara Siva Rao. Experimental investigation on navigation of mobile robot using ant colony optimization. In *Smart Computing and Informatics*, pages 123–132. Springer, 2018.

[140] T Herlambang, D Rahmalia, and T Yulianto. Particle swarm optimization (pso) and ant colony optimization (aco) for optimizing pid parameters on autonomous underwater vehicle (auv) control system. In *Journal of Physics: Conference Series*, volume 1211, page 012039. IOP Publishing, 2019.

[141] Siyuan Li, Cunbao Ma, Qi Li, Jun Zeng, and Lishan Wang. Application of improved ant colony algorithm in flight path planning. In *2020 IEEE International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA)*, volume 1, pages 763–771. IEEE, 2020.

[142] Guoliang Chen and Jie Liu. Mobile robot path planning using ant colony algorithm and improved potential field method. *Computational Intelligence and Neuroscience*, 2019, 2019.

[143] Xiaolin Dai, Shuai Long, Zhiwen Zhang, and Dawei Gong. Mobile robot path planning based on ant colony algorithm with a* heuristic method. *Frontiers in neurorobotics*, 13:15, 2019.

[144] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[145] Maurice Clerc. *Particle swarm optimization*, volume 93. John Wiley & Sons, 2010.

[146] Zhi Qi, Qian Shi, and Hui Zhang. Tuning of digital pid controllers using particle swarm optimization algorithm for a can-based dc motor subject to stochastic delays. *IEEE Transactions on Industrial Electronics*, 67(7):5637–5646, 2019.

[147] Emel Kızılkaya Aydoğan, Yılmaz Delice, Uğur Özcan, Cevriye Gencer, and Özkan Bali. Balancing stochastic u-lines using particle swarm optimization. *Journal of Intelligent Manufacturing*, 30(1):97–111, 2019.

[148] Ganesh K Venayagamoorthy, Lisa L Grant, and Sheetal Doctor. Collective robotic search using hybrid techniques: Fuzzy logic and swarm intelligence inspired by nature. *Engineering Applications of Artificial Intelligence*, 22(3):431–441, 2009.

[149] Ellips Masehian and Davoud Sedighizadeh. Multi-objective pso-and npso-based algorithms for robot path planning. *Advances in electrical and computer engineering*, 10(4):69–76, 2010.

[150] Firas A Raheem, Mustafa M Badr, et al. Development of modified path planning algorithm using artificial potential field (apf) based on pso for factors optimization. *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)*, 37(1):316–328, 2017.

[151] Baoye Song, Zidong Wang, and Lei Zou. On global smooth path planning for mobile robots using a novel multimodal delayed pso algorithm. *Cognitive Computation*, 9(1):5–17, 2017.

[152] Guido Ardizzon, Giovanna Cavazzini, and Giorgio Pavesi. Adaptive acceleration coefficients for a new search diversification strategy in particle swarm optimization algorithms. *Information Sciences*, 299:337–378, 2015.

[153] Zengwei Lyu, Zhenchun Wei, Jie Pan, Hua Chen, Chengkai Xia, Jianghong Han, and Lei Shi. Periodic charging planning for a mobile wce in wireless rechargeable sensor networks based on hybrid pso and ga algorithm. *Applied Soft Computing*, 75:388–403, 2019.

[154] Pradipta Kumar Das, Himansu Sekhar Behera, and Bijaya K Panigrahi. A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. *Swarm and Evolutionary Computation*, 28:14–28, 2016.

[155] Haiyan Wang and Zhiyu Zhou. A heuristic elastic particle swarm optimization algorithm for robot path planning. *Information*, 10(3):99, 2019.

[156] Dervis Karaboga and Bahriye Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied soft computing*, 8(1):687–697, 2008.

[157] Sandeep Kumar and Rajani Kumari. Artificial bee colony, firefly swarm optimization, and bat algorithms. *Advances in swarm intelligence for optimizing problems in computer science*, pages 145–182, 2018.

[158] Dervis Karaboga and Bahriye Akay. A comparative study of artificial bee colony algorithm. *Applied mathematics and computation*, 214(1):108–132, 2009.

[159] Marco A Contreras-Cruz, Jose J Lopez-Perez, and Victor Ayala-Ramirez. Distributed path planning for multi-robot teams based on artificial bee colony. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 541–548. IEEE, 2017.

[160] Abdul Qadir Faridi, Sanjeev Sharma, Anupam Shukla, Ritu Tiwari, and Joydip Dhar. Multi-robot multi-target dynamic path planning using artificial bee colony and evolutionary programming in unknown environment. *Intelligent Service Robotics*, 11(2):171–186, 2018.

[161] Anand Nayyar, Nhu Gia Nguyen, Rajani Kumari, and Sandeep Kumar. Robot path planning using modified artificial bee colony algorithm. In *Frontiers in Intelligent Computing: Theory and Applications*, pages 25–36. Springer, 2020.

[162] Arindam Jati, Garima Singh, Pratyusha Rakshit, Amit Konar, Eunjin Kim, and Atulya K Nagar. A hybridisation of improved harmony search and bacterial foraging for multi-robot motion planning. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.

[163] Marcello Cirillo, Tansel Uras, and Sven Koenig. A lattice-based approach to multi-robot motion planning for non-holonomic vehicles. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 232–239. IEEE, 2014.

[164] Jialong Zhang, Jianguo Yan, and Pu Zhang. Fixed-wing uav formation control design with collision avoidance based on an improved artificial potential field. *IEEE Access*, 6:78342–78351, 2018.

[165] Zhifu Chen, Tianguang Chu, and Jianlei Zhang. Swarm splitting and multiple targets seeking in multi-agent dynamic systems. In *49th IEEE Conference on Decision and Control (CDC)*, pages 4577–4582. IEEE, 2010.

[166] Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[167] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control*, 51(3):401–420, 2006.

[168] Wei Ren and Randal W Beard. Decentralized scheme for spacecraft formation flying via the virtual structure approach. *Journal of Guidance, Control, and Dynamics*, 27(1):73–82, 2004.

[169] Wei Ren and Randal W Beard. Formation feedback control for multiple spacecraft via virtual structures. *IEE Proceedings-Control Theory and Applications*, 151(3):357–368, 2004.

[170] Dingjiang Zhou, Zijian Wang, and Mac Schwager. Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures. *IEEE Transactions on Robotics*, 34(4):916–923, 2018.

[171] Chien Chern Cheah, Saing Paul Hou, and Jean Jacques E Slotine. Region-based shape control for a swarm of robots. *Automatica*, 45(10):2406–2411, 2009.

[172] Xuejing Lan, Zhenghao Wu, Wenbiao Xu, and Guiyun Liu. Adaptive-neural-network-based shape control for a swarm of robots. *Complexity*, 2018, 2018.

[173] Xiao Yan, Jian Chen, and Dong Sun. Multilevel-based topology design and shape control of robot swarms. *Automatica*, 48(12):3122–3127, 2012.

[174] Wenxin Fang, Jiabao Zhao, and Yuchen Pan. Combined flocking and region-based shape control for multi-agent systems. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 3557–3562. IEEE, 2019.

[175] Zhonghua Miao, Jinwei Yu, Jinchen Ji, and Jin Zhou. Multi-objective region reaching control for a swarm of robots. *Automatica*, 103:81–87, 2019.

[176] Anish Pandey and Dayal R Parhi. Optimum path planning of mobile robot in unknown static and dynamic environments using fuzzy-wind driven optimization algorithm. *Defence Technology*, 13(1):47–58, 2017.

[177] Bithin Datta, Dibakar Chakrabarty, and Anirban Dhar. Identification of unknown groundwater pollution sources using classical optimization with linked simulation. *Journal of Hydro-Environment Research*, 5(1):25–36, 2011.

[178] Michael Held and Richard M Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied mathematics*, 10(1):196–210, 1962.

[179] Swagatam Das, Arijit Biswas, Sambarta Dasgupta, and Ajith Abraham. Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications. In *Foundations of computational intelligence volume 3*, pages 23–55. Springer, 2009.

[180] Mohammed Isam Ismael Abdi, Muhammad Umer Khan, Ahmet Güneş, and Deepti Mishra. Escaping local minima in path planning using a robust bacterial foraging algorithm. *Applied Sciences*, 10(21):7905, 2020.

[181] Hanning Chen, Yunlong Zhu, and Kunyuan Hu. Multi-colony bacteria foraging optimization with cell-to-cell communication for rfid network planning. *Applied Soft Computing*, 10(2):539–547, 2010.

[182] Dibyendu Roy and Madhubanti Maitra. Studies on strong structural controllability and optimum link weight assignment of complex networks. In *2013 Annual IEEE India Conference (INDICON)*, pages 1–6. IEEE, 2013.

[183] S Mishra, G Dilip Reddy, P Eswar Rao, and K Santosh. Implementation of new evolutionary techniques for transmission loss reduction. In *2007 IEEE Congress on Evolutionary Computation*, pages 2331–2336. IEEE, 2007.

[184] K Vaisakh, P Praveena, S Rama Mohana Rao, and Kala Meah. Solving dynamic economic dispatch problem with security constraints using bacterial foraging pso-de algorithm. *International journal of electrical power & energy systems*, 39(1):56–67, 2012.

[185] Rahmat Allah Hooshmand and Shirin Soltani. Fuzzy optimal phase balancing of radial and meshed distribution networks using bf-pso algorithm. *IEEE Transactions on Power Systems*, 27(1):47–57, 2011.

[186] Yasufumi Yamada, Kentaro Ito, Takumi Tsuji, Kohei Otani, Ryo Kobayashi, Yoshiaki Watanabe, and Shizuko Hiryu. Ultrasound navigation based on minimally designed vehicle inspired by the bio-sonar strategy of bats. *Advanced Robotics*, 33(3-4):169–182, 2019.

[187] Patricia Suárez, Andrés Iglesias, and Akemi Gálvez. Make robots be bats: specializing robotic swarms to the bat algorithm. *Swarm and Evolutionary Computation*, 44:113–129, 2019.

[188] Mehran Rahmani, Ahmad Ghanbari, and Mir Mohammad Ettefagh. Robust adaptive control of a bio-inspired robot manipulator using bat algorithm. *Expert Systems with Applications*, 56:164–176, 2016.

[189] Chao Gan, Weihua Cao, Min Wu, and Xin Chen. A new bat algorithm based on iterative local search and stochastic inertia weight. *Expert Systems with Applications*, 104:202–212, 2018.

[190] Dimos V Dimarogonas and Karl H Johansson. Stability analysis for multi-agent systems using the incidence matrix: quantized communication and formation control. *Automatica*, 46(4):695–700, 2010.

[191] Jianfang Xiao, Peng Wang, and Leonardy Setyawan. Hierarchical control of hybrid energy storage system in dc microgrids. *IEEE Transactions on Industrial Electronics*, 62(8):4915–4924, 2015.

[192] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[193] Roger A Horn. The hadamard product. In *Proc. Symp. Appl. Math*, volume 40, pages 87–169, 1990.

[194] Daniel Göhring, Miao Wang, Michael Schnürmacher, and Tinosch Ganjineh. Radar/lidar sensor fusion for car-following on highways. In *The 5th International Conference on Automation, Robotics and Applications*, pages 407–412. IEEE, 2011.

[195] Jorg Muller, Axel Rottmann, Leonhard M Reindl, and Wolfram Burgard. A probabilistic sonar sensor model for robust localization of a small-size blimp in indoor environments using a particle filter. In *2009 IEEE International Conference on Robotics and Automation*, pages 3589–3594. IEEE, 2009.

[196] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[197] Juan Li, Xi-Yuan Ma, Chen-Ching Liu, and Kevin P Schneider. Distribution system restoration with microgrids using spanning tree search. *IEEE Transactions on Power Systems*, 29(6):3021–3029, 2014.

[198] Gabriel Taubin and Jarek Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics (TOG)*, 17(2):84–115, 1998.

[199] Andrea Torsello and Edwin R Hancock. A skeletal measure of 2d shape similarity. *Computer Vision and Image Understanding*, 95(1):1–29, 2004.

[200] Veysel Gazi. Swarm aggregations using artificial potentials and sliding-mode control. *IEEE Transactions on Robotics*, 21(6):1208–1214, 2005.

[201] Jun-Wei Zhu, Guang-Hong Yang, Wen-An Zhang, and Li Yu. Cooperative fault tolerant tracking control for multiagent systems: An intermediate estimator-based approach. *IEEE transactions on cybernetics*, 48(10):2972–2980, 2017.

[202] Haitao Zhao, Hai Liu, Yiu-Wing Leung, and Xiaowen Chu. Self-adaptive collective motion of swarm robots. *IEEE Transactions on Automation Science and Engineering*, 15(4):1533–1545, 2018.

[203] Haitao Zhao, Jibo Wei, Shengchun Huang, Li Zhou, and Qi Tang. Regular topology formation based on artificial forces for distributed mobile robotic networks. *IEEE Transactions on Mobile Computing*, 18(10):2415–2429, 2018.

[204] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews neuroscience*, 10(3):186–198, 2009.

[205] Nathan Michael and Vijay Kumar. Planning and control of ensembles of robots with non-holonomic constraints. *The International Journal of Robotics Research*, 28(8):962–975, 2009.

[206] Mihály Csaba Markót and Tibor Csendes. A reliable area reduction technique for solving circle packing problems. *Computing*, 77(2):147–162, 2006.

[207] Danny Z Chen, Xiaobo Hu, Yingping Huang, Yifan Li, and Jinhui Xu. Algorithms for congruent sphere packing and applications. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 212–221, 2001.

[208] James C Bezdek and Ian M Anderson. An application of the c-varieties clustering algorithms to polygonal curve fitting. *IEEE transactions on systems, man, and cybernetics*, (5):637–641, 1985.

[209] Jón Atli Benediktsson, Jón Aevar Palmason, and Johannes R Sveinsson. Classification of hyperspectral data from urban areas based on extended morphological profiles. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):480–491, 2005.

[210] Nicolae Duta, Anil K Jain, and M-P Dubuisson-Jolly. Learning 2d shape models. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 8–14. IEEE, 1999.

[211] Frank P Kuhl and Charles R Giardina. Elliptic fourier features of a closed contour. *Computer graphics and image processing*, 18(3):236–258, 1982.

[212] Yara E Sánchez-Corrales, Matthew Hartley, Jop Van Rooij, Athanasius FM Marée, and Verônica A Grieneisen. Morphometrics of complex cell shapes: lobe contribution elliptic fourier analysis (loco-efa). *Development*, 145(6), 2018.

[213] Dibyendu Roy, Arijit Chowdhury, Madhubanti Maitra, and Samar Bhattacharya. Virtual region based multi-robot path planning in an unknown occluded environment. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 588–595. IEEE, 2019.

[214] Andreea Panaitescu, Gregory M Grason, and Arshad Kudrolli. Persistence of perfect packing in twisted bundles of elastic filaments. *Physical review letters*, 120(24):248002, 2018.

[215] Ignacio Castillo, Frank J Kampas, and János D Pintér. Solving circle packing problems by global optimization: numerical results and industrial applications. *European Journal of Operational Research*, 191(3):786–802, 2008.

[216] Gustav Mårtensson, Joana B Pereira, Patrizia Mecocci, Bruno Vellas, Magda Tsolaki, Iwona Kłoszewska, Hilkka Soininen, Simon Lovestone, Andrew Simmons, Giovanni Volpe, et al.

Stability of graph theoretical measures in structural brain networks in alzheimer's disease. *Scientific reports*, 8(1):1–15, 2018.

[217] Krishna Raghuwaiya and Shonal Singh. Formation types of multiple steerable 1-trailer mobile robots via split/rejoin maneuvers. *New Zealand Journal of Mathematics*, 43:7–21, 2013.

[218] Shude He, Min Wang, Shi-Lu Dai, and Fei Luo. Leader–follower formation control of usvs with prescribed performance and collision avoidance. *IEEE Transactions on Industrial Informatics*, 15(1):572–581, 2018.

[219] Szilárd Novoth, Qian Zhang, Kang Ji, and Dingli Yu. Distributed formation control for multi-vehicle systems with splitting and merging capability. *IEEE Control Systems Letters*, 5(1):355–360, 2020.

[220] Takashi Emura and Lei Wang. A high-resolution interpolator for incremental encoders based on the quadrature pll method. *IEEE Transactions on industrial Electronics*, 47(1):84–90, 2000.

[221] Jayant Mankar, Chaitali Darode, Komal Trivedi, Madhura Kanoje, and Prachi Shahare. Review of i2c protocol. *International Journal of Research in Advent Technology*, 2(1), 2014.

[222] Zahra Ziaei, Reza Oftadeh, and Jouni Mattila. Global path planning with obstacle avoidance for omnidirectional mobile robot using overhead camera. In *2014 IEEE International Conference on Mechatronics and Automation*, pages 697–704. IEEE, 2014.

[223] WSP Fernando, Lanka Udawatta, and Pubudu Pathirana. Identification of moving obstacles with pyramidal lucas kanade optical flow and k means clustering. In *2007 Third International Conference on Information and Automation for Sustainability*, pages 111–117. IEEE, 2007.

[224] Chunbo Luo, Wang Miao, Hanif Ullah, Sally McClean, Gerard Parr, and Geyong Min. Unmanned aerial vehicles for disaster management. In *Geological Disaster Monitoring Based on Sensor Networks*, pages 83–107. Springer, 2019.

[225] Matthew Turk and Vassilis Athitsos. Gesture recognition. *Computer vision: A reference guide*, pages 1–6, 2020.

[226] Gilles Caprari, Kai Oliver Arras, and Roland Siegwart. Robot navigation in centimeter range labyrinths. In *Proceedings of The First International Conference on Autonomous Minirobots for Research and Edutainment (AMIRE)*. ETH Zürich, 2001.

Appendix 1

The traditional PSO [140] algorithm is presented below

1. The algorithm is initialized with a random particle population to search the optima by updating generations. Each particle has three main properties: the current position pos_i, velocity vel_i, and the personal best position in the search space y_i.
2. In each repetition, the position of the particles are updated using p_{best} (personal best), and g_{best} (global best) by minimizing the objective function as defined in Eq. 3.22.
3. After finding the best values, each particle will update using a modified velocity and position as shown below

$$vel_i^{k+1} = w \times vel_i^k + c_1 \times rand_1(p_{best,i}^k - pos_i^k) + c_2 \times rand_2(g_{best,i}^k - pos_i^k) \quad (1)$$

$$pos_i^{k+1} = pos_i^k + vel_i^{k+1} \quad (2)$$

   where, vel_i^{k+1}, vel_i^k are the velocities of the i^{th} particle at the $(k+1)^{th}$, and k^{th} iterations respectively, pos_i^{k+1}, pos_i^k are the positions of the i^{th} particle at the $(k+1)^{th}$, and k^{th} iterations respectively, c_1, c_2 are the positive constants, $rand_1$, and $rand_2$ are the random variables with uniform distribution between 0 and 1, and w is the inertial weight.
4. The process continues until all particles are exhausted and the final g_{best} is the optimal solution.

Appendix 2

The Bacterial Foraging Optimization algorithm (BG) contains four-sub phases [179] that are explained below.

Chemotaxis: This process simulates the movement of an *E. coli* cell through swimming and tumbling via flagella. An E.coli bacterium can move in two different ways. It can swim for a while in the same direction or it may tumble, and alternate between these two modes of operation for the entire lifetime. Suppose $\theta^i(j,k,l)$ represents i^{th} bacterium at j^{th} chemotactic, k^{th} reproductive and l^{th} elimination-dispersal step. $C(i)$ is the size of the step taken in the random direction specified by the tumble (the run length unit). Then in computational chemotaxis ($\theta^i(j+1,k,l)$), the movement of the bacterium may be represented by [176]

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + c(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{3}$$

where, Δ represents a vector in the random direction in $[-1,1]$.

Swarming: When a single bacterium reaches in the goal position, it delivers an attraction signal to other agent so that they can swarm together. The mathematical representation for swarming can be represented by

$$\begin{aligned} J_{cc}(\theta, p(j,k,l)) &= \sum_{i=1}^{s} j \sum_{i=1}^{s} j_{cc}^i(\theta, \theta^i(j,k,l)) \\ &= \sum_{i=1}^{s}\left[-d_{attractant} \times \left(-w_{attractant}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\right)\right] \\ &+ \sum_{i=1}^{s}\left[-h_{repellent} \times \left(-w_{repellent}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\right)\right] \end{aligned} \tag{4}$$

where, $J_{cc}(\theta, p(j,k,l))$ denotes the cost function that needs to be minimized iteratively while optimizing the parameters p present in each bacterium. The another parameters $d_{attractant}$, $w_{attractant}$, $h_{repellent}$, and $w_{repellent}$ are to be chosen adaptively.

Reproduction: To formulate the reproduction technique, after N_c chemotactic step size, the health of all bacteria is sorted in ascending order, based on their accumulated

cost function, can be articulated as

$$J_{health} = \sum_{j=1}^{N_c+1} J(i,j,k,l) \tag{5}$$

In this process, bacteria split into two groups based on their cost function value: the highest value means the least healthy bacteria whereas the lowest cost function value indicates the healthiest bacteria. Based on their health, the bacteria swarm is separated into two halves as

$$S_r = \frac{s}{2} \tag{6}$$

The least healthy bacteria die and the healthiest ones split into two parts at the same position such that the overall population remains fixed. After reproduction, bacteria continue the chemotaxis process until they achieve the maximum chemotactic number N_c, and then other reproduction events would be performed sequentially.

Elimination and Dispersal: The bacteria population in the nutrient media can be abridged or replaced directly because of exterior factors such as catastrophe that will make nutrient media poisoned instantly. In the modeling, after performing N_{re} reproduction steps, the elimination and dispersal events take place. Bacteria having a probability value (between 0 to 1) lesser than a certain threshold value (p_{ed}) are eliminated and detached to another location and bacteria which have a probability value higher than p_{ed} keep their existing location.

Alikyanda Roy