Robotic Swarm With Independent Selection Of Master-Slave Configurations

Dhruva Shaw [dhruvashaw@ieee.org]

***Abstract*—Traditional master-slave control limits robotic swarms in dynamic environments. This paper proposes a decentralized approach where individual robots independently select master-slave configurations based on local information. A bio-inspired algorithm using principles like quorum sensing enables self-organization and efficient resource allocation. This research promotes swarm performance, robustness, and paves the way for adaptable robotic systems for complex tasks.**

***Index Terms*—optimization, swarm, swarm robotics, master slave, configuration, Consensus control, Self-assembly, Task allocation, Fault tolerance, Scalability, Emergent behavior,Self-organization, Swarm intelligence, Adaptive control, Decentralized control, Cooperative control, Robotic swarm, Master-slave, configuration, Independent selection, Multi-robot systems, Distributed control**

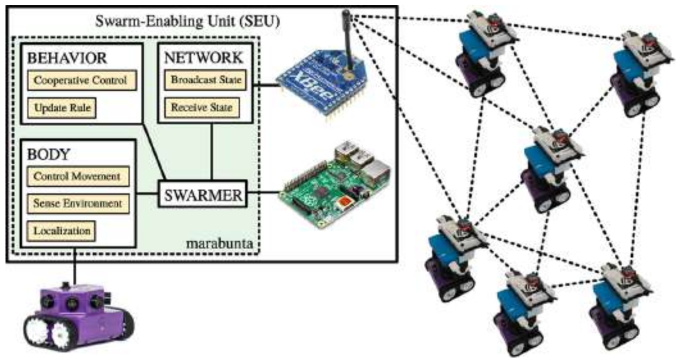


Fig. 1. Traditional swarm robot control strategies [5]

I. Introduction

ROBOTIC swarms hold immense potential for revolutionizing tasks like search and rescue over a diverse geographical terrain which is nearly inaccessible by humans. Also preferred for environmental monitoring in in accessible high altitude terrains and risk assessment due to their inherent adaptability and robustness. However, traditional control architectures with a single master robot struggles in such diversified dynamic environments. Centralized control becomes inefficient due to loss of contact and dependency on the single master transmitting frequency due to the rapid change in environment, hindering the swarm's ability to respond effectively[1]. This research addresses this limitation by proposing a novel, decentralized control architecture for robotic swarms. I introduce the concept of independent master-slave configuration selection, where individual robots within the swarm dynamically choose their roles based on local information and task demands. This eliminates the vulnerability of a single point of failure and fosters emergent coordination within the swarm. The core innovation lies in a bio-inspired optimization algorithm that empowers robots to make independent decisions about their roles. Drawing inspiration from social insect behavior, this algorithm facilitates self-organization and efficient task allocation within the swarm, like swarm of bees.

II. Current Limitations in the Swarm Control

While traditional master-slave control architectures have demonstrably enabled swarm robotics applications, several key limitations hinder their effectiveness in dynamic environments. Here, I discuss the most prominent challenges:

- **Limited Adaptability**: Current approaches often struggle to adapt to unforeseen changes in the environment. Centralized control structures require pre-programmed responses or rely on a central server to re-plan in real-time, which can be computationally expensive and slow down the swarm's reaction.[2] [3]
- **Task Allocation Bottleneck**: Efficiently allocating tasks amongst individual robots in a centralized system can be a complex challenge. Factors like robot capabilities, real-time sensor data, and dynamic task requirements require constant evaluation by the central controller, creating a potential bottleneck. [2]
- **Single Point of Failure**: A critical vulnerability of centralized control architectures is the presence of a single point of failure. If the central master robot or server malfunctions, the entire swarm can become incapacitated. [4]

Addressing the Centralization Issue

These limitations highlight the need for a more decentralized and adaptive control architecture, bringing in executional and operational speed, reliability while eliminating the failure factor. In my proposed approach of independent master-slave configuration selection, I have attempted to directly addresses these challenges. By empowering individual robots to make informed decisions based on local information and task requirements, the swarm can adapt more readily to dynamic environments. The distributed nature of the control eliminates the single point of failure present in centralized systems, fostering robustness and resilience.

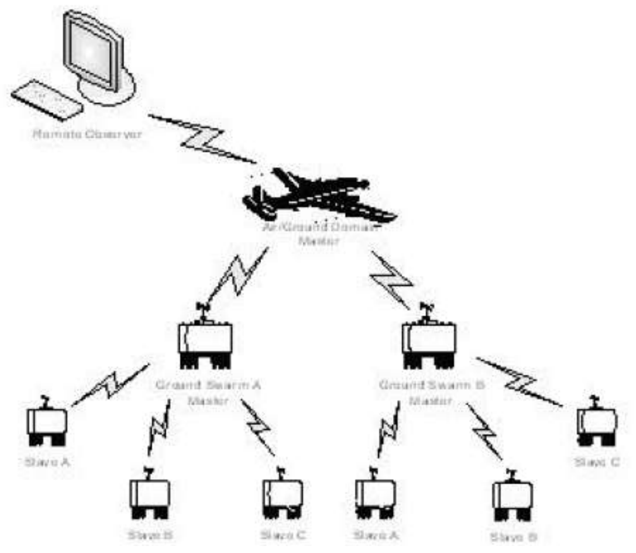


Fig. 2. Our proposed approach model of decentralization and role division in robotic swarms (implemented in UAV's)[2]

III. Motivation

The development of effective robotic swarms presents exciting possibilities for various applications. This research is motivated by the limitations encountered during the development of a robot swarm for the IIT Delhi Tryst's Robosoccer competition, 2024. However, achieving robust and adaptable behaviour remains a challenge.

A. The Challenges

The Robosoccer competition demanded a well-coordinated defense strategy. While a single robot can be controlled, a truly effective defense requires at least two: a defender responding to human commands and an autonomous goalkeeper. This scenario warranted architecture the need for an integrated, self-reliant & adaptive swarm that can adapt its leadership structure based on the situation.

B. Inspiration from nature and society

Human societies and military structures demonstrate the power of dynamic leadership selection. Leaders emerge based on real-time needs and individual strengths. This research aims to replicate this concept in robotic swarms.

In nature the swarm of bees show the adaptive nature with de-centralized command & control for an efficient win-win situation in any given circumstances. This similar behaviour can also be observed in siberian migratory bird.

IV. The Research Focus

This research explores the concept of "***independent selection of master-slave configuration***" within a robotic swarm. This approach seeks to develop swarms where robots can intelligently choose leadership roles (master/slave) based on real-time requirements, adapting to the dynamic change.

TABLE I
Lookup Components Priority Table

Component Priority Number	Components Name
1	Inertial Measurement Unit
2	Ultrasonic Distance Sensor
3	Infrared Sensor
4	GPS (Global Positioning System)
5	Wheel Encoder

V. Potential Benefits

- **Increased Adaptability**: Swarms can dynamically adjust their leadership structure to address changing situations.
- **Enhanced Collaboration**: Robots can specialize based on their capabilities, leading to more efficient teamwork.
- **Improved Fault Tolerance**: The loss of a leader wouldn't cripple the swarm as another robot could take over.

Overall, this research seeks to unlock a new level of adaptability and collaboration in robotic swarms, paving the way for more versatile and robust real-world applications.

VI. Proposed Approach

To implement our approach, we'll need to develop a lookup priority table to define the order of operations and a decision algorithm to guide our actions. We'll explore these in more detail in the following sections.

To validate our approach, I implemented the entire system in MATLAB Simulink. I utilized three distinct robot configurations, each equipped with a unique combination of the five different components: IMU, ultrasonic sensor, GPS, wheel encoders, and infrared sensor. By testing these varied configurations, I aimed to evaluate the effectiveness of our component-based master-slave selection under different conditions.

A. Swarm: Initial Connectivity

During initial swarm establishment, the server initiates communication by broadcasting a request to either the closest bot or a random bot within range. This request includes essential details enumerated in the component priority lookup table. Simultaneously, bots establish connections with each other, forming a circular network topology. See figure (2) .
Upon receiving the server's data, the designated bot relays it to all other swarm members. As per the design configuration, Individual bots then perform a matching process, verifying and correcting their locally stored data against the server's information. The data consistency is achieved by performing permutations and combinations at all the bots. The swarm transitions to the decision-making phase.

B. Priority Lookup Table

The priority lookup table is a critical element in determining the master-slave configuration within the robot swarm. This table dictates robot decisions based on the components each robot possesses. The system assumes a predefined task requiring specific components, such as sensors with varying data importance. The robot with the most necessary components for self-sustenance and data processing becomes the master.

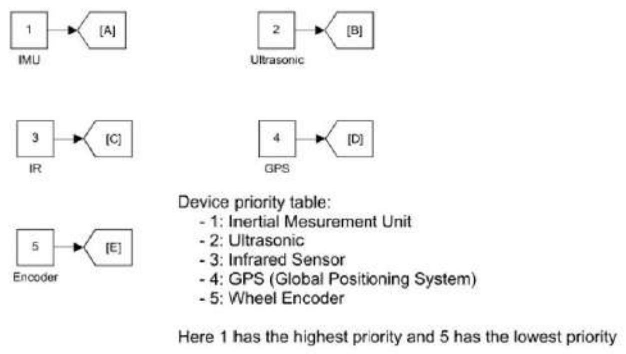

1
[A]
IMU
2
[B]
Ultrasonic
3
[C]
IR
4
[D]
GPS
5
[E]
Encoder
Device priority table:
- 1: Inertial Mesurement Unit
- 2: Ultrasonic
- 3: Infrared Sensor
- 4: GPS (Global Positioning System)
- 5: Wheel Encoder
Here 1 has the highest priority and 5 has the lowest priority

Fig. 3. Components Priority Lookup Table of the MATLAB simulation

In our MATLAB simulation, I utilize five components: IMU (Inertial Measurement Unit), Ultrasonic sensor, GPS, wheel encoders, and infrared sensor. Here, a lower number signifies higher priority. Consequently, the IMU has the highest priority due to its essential role. Initially, each robot stores a default priority table based on its individual connections. However, upon receiving the component priority lookup table from the server, this table takes precedence for the task duration. *Please refer table* (I) *and figure* (3)

C. Working of Decision Making Block

The decision-making block plays a central role in selecting robots for leadership and communication within the swarm. It leverages information gathered through established communication channels (described in "Swarm: Initial Connectivity" subsection) and pre-defined priority values assigned to sensor configurations (explained in "Priority Lookup Table" subsection)
[Note: Here the decision making block is shown centrally in simulation but in practical implementation this decision making would be there locally same in all the bots associated with the swarm].

1) Selection Process: This process involves several steps, potentially using various functions and blocks:

- **Data Acquisition**:

  1) Each robot analyzes its own sensor configuration.
  2) Robots calculate their corresponding priority values ("S1", "S2", "S3") based on the lookup table and their sensor configurations (adding all the .
  3) The **max block** output a constant value ("maxVal") represents the highest possible priority value (indicating a robot with all/max no of components connected to that individual robot).
  4) They then exchange information through communication channels to obtain priority values (including "maxVal" from others) of fellow robots in the swarm.
  5) Additionally, robots measure their distances to a designated master/central server (potentially using a specific protocol). This distance information of individual robots is represented by the "D1", "D2", "D3" tags of the "system1", "system2" and "system3" respectively in the simulation.

- **Acting Transmitter Selection (Independent Process)**:

  1) This independent selection identifies the robot closest to the server, utilizing the "distVal" measurements.
  [Note: "distVal" is the output of the "min" decision block, it compares the distance of bots from the main central server and outputs the the value which has the "least" distance of all.]
  2) A **min block** (potentially) could be used in some implementations to find the minimum distance value ('distVal') and identify this closest robot.
  3) The "system2" becomes the **acting transmitter** of the swarm, crucial for efficient communication between the swarm and the server.
  4) The selection of the acting transmitter offers flexibility. Developers can choose to calculate the minimum distance to the server for optimal communication or designate the acting master itself as the transmitter, simplifying the algorithm. This choice depends on prioritizing efficient communication or a streamlined selection process.

- **Current Acting Master Selection**:

  1) A separate mechanism, possibly involving the "maxVal" tag and the max block, identifies the current acting master.
  2) The "max block" might output the "maxVal" representing the highest priority (all components).
  3) The system selects the robot with a priority value **equal to "maxVal"** (maximum priority value from all the robots in the swarms) as the current acting master. This prioritizes a robot with the most complete sensor configuration for leadership tasks.

- **Next Potential Master (NPM) Selection**:

  1) The "fcn56" function analyzes various factors to proactively identify a suitable robot as the NPM in case the current master fails.
  2) System which has the second highest added priority value gets selected as the Next Potential Master (NPM).

VII. Results

Figure 7 illustrates the roles assigned to the systems within the swarm simulation.

- **Master:** System 1 is identified as the master based on having the **highest number of connected components**. This suggests it possesses the most suitable sensor configuration for overall swarm coordination.
- **Acting Transmitter:** While System 2 is not the master, it acts as the transmitter for communication with the server. This selection is likely due to its **shortest distance to the server** compared to other systems in the swarm.

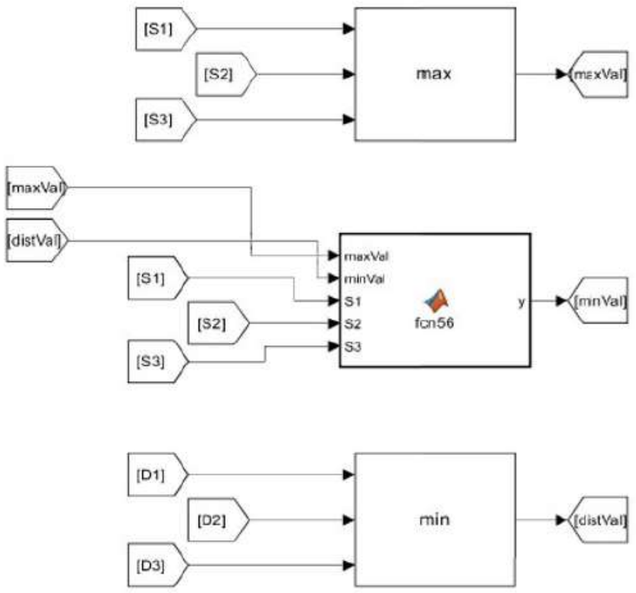
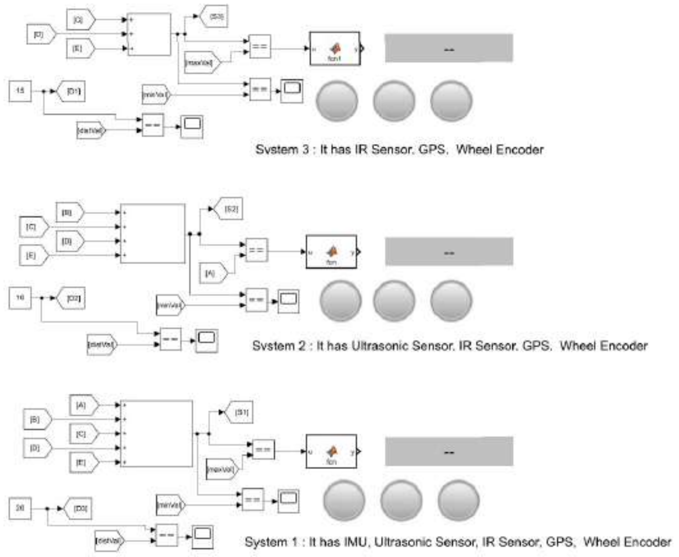


Fig. 4. Decision Making Blocks



System 3 : It has IR Sensor, GPS, Wheel Encoder
System 2 : It has Ultrasonic Sensor, IR Sensor, GPS, Wheel Encoder
System 1 : It has IMU, Ultrasonic Sensor, IR Sensor, GPS, Wheel Encoder

Fig. 5. Different Systems Configuration In Simulation

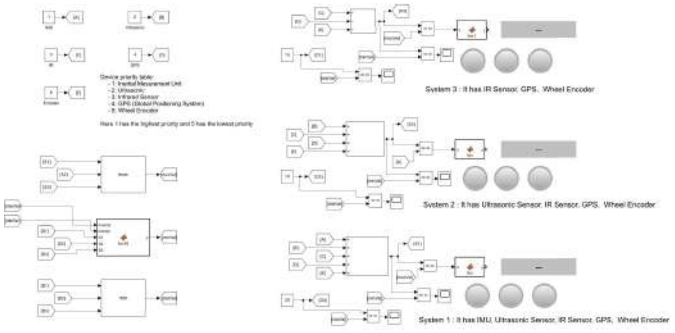


Fig. 6. Whole simulation before the running

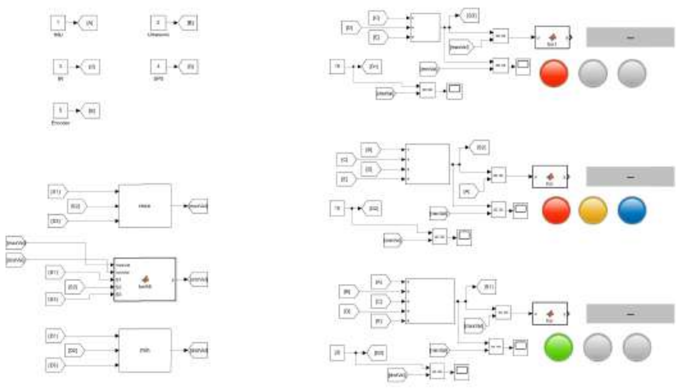


Fig. 7. Simulation Results of the whole configuration/algorithm (MATLAB/Simulink)

- **Next Potential Master:** In case of a master failure, the system has designated System 2 as the Next Potential Master (NPM). This selection might be based on System 2 having the **second highest number of connected components**, indicating a strong configuration for taking over the master role.

Color Legend for LED Indicators:

- **Red:** Slave (S),
- **Green:** Master,
- **Yellow:** Acting Transmitter,
- **Blue:** Next Potential Master

VIII. Mathematical Equations

$$Set(P) = \bigcup_{i=1}^{n} F(A_i) \ , \ A_i \in \{x_1, x_2, ..., x_n\}$$

$$\text{where, } F(A_i) = \sum_{j=1}^{\omega} a_j, a_j \in (A_i)$$

x_n, represents the configuration of the nth system/robot
n, total no of systems/robot connected to the swarm.
ω, total no of components connected to that particular system.
a_j, priority no of that component from the priority table I

$Set(D) = \{d_1, d_2, ..., d_n\}$
where, d_n is the calculated distance of n_{th} bot from the server.

A. *Current Acting Master Selection*

$$\alpha_{\text{current master}} = max(Set(P)) \tag{1}$$

B. *Current Acting Transmitter Selection*

$$\beta_{\text{acting transmitter}} = min(Set(D)) \tag{2}$$

C. *Next Potential Master Selection*

$$\gamma_{\text{npm}} = max(Set(P) - \alpha_{\text{current master}}) \tag{3}$$

IX. Matlab Function(s) Code

Here's the implementation of "fcn56", the function that determines the Next Potential Master (NPM) for the swarm.

```
function y = fcn56(maxVal, minVal, S1, S2, S3)
if S1 ~= maxVal && S1 ~= minVal
    y = S1;
elseif S2 ~= maxVal && S2 ~= minVal
    y = S2;
else
    y = S3;
end
```

Functions "fcn1" and "fcn" define the display strings based on the system's role (Master/Slave).

```
function y = fcn(u)
if u
    y = "Master";
else
    y = "Slave";
end
```

X. Advantages

A. *Distributed Decision-Making:*

- **Scalability:** This approach scales effectively to larger swarms. Each robot can determine its role independently, eliminating the need for a central coordinator that might become a bottleneck in large-scale deployments.
- **Fault Tolerance**: If a robot fails, the remaining robots can still independently determine new roles, ensuring the swarm's continued operation. This redundancy improves the overall robustness of the system.
- **Adaptability:** Individual robots can adjust their roles based on real-time changes in their own capabilities or the swarm's environment. This dynamic adaptation allows the swarm to respond effectively to unforeseen situations.

B. *Simplified Communication:*

- **Reduced Communication Overhead:** By eliminating the need for continuous communication to maintain a central master, this approach reduces the communication burden on the swarm, potentially improving overall efficiency.

C. *Additional Potential Advantages:*

- **Improved Task Execution**: Depending on your specific implementation, independent role selection based on sensor configurations could lead to a better distribution of tasks within the swarm, potentially optimizing overall performance.

Overall, the independent selection of master-slave configurations in a swarm presents a promising approach for developing adaptable and robust robotic swarms. This approach offers advantages in scalability, fault tolerance, and simplified communication, making it well-suited for various swarm robotics applications.

XI. Comparative Study: Independent Master-Slave Selection vs. Existing Approaches

Robotic swarm systems rely on efficient role assignment to individual robots for coordinated task execution. While traditional methods often employ centralized control or static role allocation, this research proposes a novel approach for **independent master-slave selection in a swarm. This section compares our approach with existing methods, highlighting its advantages and potential trade-offs**.

A. *Traditional Techniques:*

- **Centralized Control:** A central controller dictates roles (master, slave) to individual robots. This approach offers simplicity in system setup but can become a bottleneck and single point of failure in large-scale deployments [6] [2]. Additionally, centralized control limits adaptability, as the entire system hinges on the functionality of the central controller.
- **Static Role Assignment:** In some cases, roles might be pre-programmed or assigned based on robot IDs [7]. This approach simplifies initial deployment but lacks adaptability, as it doesn't account for dynamic changes within the swarm or the environment. Pre-assigned roles might not leverage the optimal capabilities of each robot for a specific task.

B. *Independent Selection:*

- **Distributed Decision-Making:** Our research introduces a distributed approach where each robot determines its role (master, slave, Next Potential Master (NPM)) considering its own properties (sensor configuration, etc.). This eliminates the need for a central controller, promoting scalability and fault tolerance. If a robot fails, the remaining robots can independently adjust roles, ensuring the swarm's continued operation.
- **Dynamic Role Selection:** The method allows for dynamic role selection based on real-time properties of each robot. This enables the swarm to adapt to changing conditions within the environment, potentially leading to more efficient task execution.

C. Comparative Advantages:

- **Scalability and Fault Tolerance:** Our distributed approach offers better scalability and fault tolerance compared to centralized control. The system can effectively handle larger swarms as there's no central bottleneck, and individual robot failures can be accommodated through independent role adjustments.
- **Adaptability:** The dynamic role selection based on real-time properties allows for better adaptation to environmental changes compared to static role assignment. This can lead to more efficient task execution as robots with the most suitable capabilities are assigned appropriate roles.

Overall, this research presents a promising alternative to traditional approaches by enabling distributed, adaptive, and scalable role selection in robotic swarms.

XII. Visualizing Dynamic Leader Selection in Robotic Swarms And Future Plans

This research is further supported by a video demonstration available at https://youtu.be/3P2LqaATrzw, The video provides a visual representation of the dynamic leader selection process within the robotic swarm simulation.
Additionally, the complete source code for the simulation, along with the implementation details and functionalities, can be found on the GitHub repository at https://github.com/Creatrix-Net/robotic-swarm-dynamic-leader-selection, The entire concept & approach for the decentralized robotic swarms aims for constructive participation by the people/engineers & thereby, creating an open-source platform for meaningful development of this project.

This repository also includes a roadmap for future enhancements and planned developments related to the research.

XIII. Conclusion

This research investigated a novel approach for independent master-slave selection in a swarm. The implemented system leverages a distributed decision-making process, where each robot determines its role (master, slave, Next Potential Master) considering its sensor configuration and other relevant properties. This approach offers several advantages, including:

- **Scalability and Fault Tolerance:** The distributed nature of role selection allows the swarm to function effectively even in large deployments and can adapt to robot failures.
- **Adaptability:** Dynamic role selection based on real-time properties enables the swarm to adjust to changing environmental conditions, potentially optimizing task execution.

The research findings demonstrate the feasibility and potential benefits of this approach for dynamic leader selection in robotic swarms.

References

[1] M. S. Kim, S. H. Kim, and S. J. Kang, "Middleware design for Swarm-Driving robots accompanying humans," *Sensors (Basel)*, vol. 17, no. 2, Feb. 2017.

[2] S. Misra, P. K. Deb, and K. Saini, "Dynamic leader selection in a master-slave architecture-based micro uav swarm," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[3] R. Lundh, L. Karlsson, and A. Saffiotti, in *Automatic Configuration of Multi-Robot Systems: Planning for Multiple Steps*, 01 2008, pp. 616–620.

[4] T. Suraj Duncan, T. R. Jayanthi Kumari, R. John, and B. P. Anirudha Prabhu, "Master–slave robots using swarm intelligence," in *International Conference on Artificial Intelligence and Sustainable Engineering*, G. Sanyal, C. M. Travieso-González, S. Awasthi, C. M. Pinto, and B. R. Purushothama, Eds. Singapore: Springer Singapore, 2022, pp. 41–48.

[5] M. Chamanbaz, D. Mateo, B. M. Zoss, G. Tokić, E. Wilhelm, R. Bouffanais, and D. K. P. Yue, "Swarm-enabling technology for multi-robot systems," *Frontiers in Robotics and AI*, vol. 4, 2017. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2017.00012

[6] P. Benavidez, K. Nagothu, A. Kumar Ray, T. Shaneyfelt, S. Kota, L. Behera, and M. Jamshidi, "Multi-domain robotic swarm communication system," in *2008 IEEE International Conference on System of Systems Engineering*, 2008, pp. 1–6.

[7] J. Deda and T. Mirosław, "Remotely controlled robot swarms: A structural analysis and model for structural optimization," *Applied Sciences*, vol. 11, no. 18, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/18/8539

Dhruva Shaw is a Robotics Engineering undergrad student from Lovely Professional University, with passion in Electronics Automation and software projects related to the automation while using inbuilt AI. Writeups and researches related to the projects are open-sourced and available for all towards a positive contribution. https://dhruvashaw.in