2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

# Guided Swarm Self-Clustering in Safe Area

Sweksha Jain[1], Leena Vachhani[1]

*Abstract*— **In high-stress situations, guiding a non-communicating, location-unaware swarm to the safe area is challenging, especially as they self-organize into clusters with nearby agents. This paper presents a novel decentralized non-interacting swarm algorithm to form *guided swarm clusters* autonomously in an area without nearby danger, termed as "safe area". The proposed strategy guides the swarm of robots towards safe area to form self-clusters in a bounded environment. It is achieved by fixing the locations of static obstacles which prevents cluster formation in the danger area. We evaluate the performance based on cluster formation in the safe area. The Monte-Carlo simulations are performed with varying percentage occupancy of obstacles in danger area, where we achieved percentage of convergence (PoC) in safe area over 85% (on an average). The proposed algorithm potentially find its application in a crowded environment, while evacuating to a safer area during emergency.**

**Keywords: Swarm Robotics, Collective Behavior, Communication-free Environment, Swarm Self-Clustering**

## I. Introduction

Swarm robotics involves the collaboration of multiple robots to accomplish tasks without centralized control. Inspired by the collective intelligence seen in various biological organisms, such as insects, birds, and fish, swarm robotics explores complex behaviors like aggregation, flocking, clustering, foraging, and pattern formation [1]. These robots typically have limited sensing, processing power, and communication abilities. Like their natural counterparts, swarm robots are designed to exhibit decentralized behavior, mirroring the autonomy seen in biological systems. The collective behaviors observed in swarm robotics are commonly classified into cue-based and self-organized behavior [2]. Cue-based behavior involves the swarm responding collectively to environmental cues like light or sound sources. On the other hand, self-organized behavior emerges from the interactions among the robots themselves, utilizing their limited capabilities to exhibit collective behavior without explicit external cues.

A key task in swarm robotics is swarm aggregation, where robots group together to form a single collective unit. Several methodologies and controllers have been proposed to exhibit this collective behavior. The evolving aggregation process of robots has been studied after taking inspiration from biological organisms [3], such as Beeclust algorithm [4], Ant-colony optimization [5] and Particle swarm optimization (PSO) [6]. In a similar context, swarm clustering is another collective behavior, where multiple groups of robots are formed in different spatial locations [7] using a self-organized behavior. This collective behavior has been classified into two categories: token clustering and robot clustering. In token clustering, the robots collect the objects/tokens of similar features [8], thereby forming clusters of similar objects. The robot clustering creates clusters of homogeneous robots with similar features [9]. Segregation in swarm robotics is defined as a collective behavior where robots group together based on shared characteristics like their size, shape etc. [10], [11]. A decentralized methodology was proposed for segregating heterogeneous robots by extending ORCA (Optimal Reciprocal Collision Avoidance) and integrating it with a PSO-inspired navigation strategy [12].

The swarm-guided navigation is emerging area of interest where agents of the swarm guide each other to a desired location [13]. The problem of safe navigation of a swarm of robots is inspired by the numerous works in the field of navigation during emergencies. The work in [14] involves the detection of safe areas and then determining the navigation strategy to reach these areas. Some solutions have been provided for emergency navigation involving drones searching and guiding the survivors out of danger zones [15]. Forming guided swarm clusters avoiding danger areas without predefined cluster location presents an intriguing problem. In this swarm clustering problem, the robots in the swarm are unaware of their position within the environment or position relative to the safe area. Moreover, robots do not communicate with each other. This paper aims to provide a novel solution by introducing a decentralized approach to form clusters of robot swarms, guided by obstacles' positioning in the environment. In the proposed approach, static obstacles serve as guides by their positioning, directing the swarm of robots to form clusters within *"safe area"* (to develop preliminary results, the safe area is devoid of obstacles).

The concept of guided swarm clusters is inspired by real-life scenarios where certain areas of a bounded environment are impacted by disasters like fires, creating *danger areas*. In response, emergent behavior is observed where humans navigate to a safer place. Humans in emergency situation deals with limited Field of View (FoV) without communicating due to crowded environment, guided only by the positions of the obstacles. The illustration in figure 1 gives visual representation of a similar behavior of humans in a bounded environment. The illustration is taken from a study [16] on crowd behavior under evacuation and shows that the obstacles' placement guides the human through free area to reach their desired "safe area" behind exit gates. The problem is more challenging to safely guide a swarm of robots while forming clusters, where the robots are unaware of the "safe area" and have a limited field of view as in the humans' case.

The key contributions of this work are:

[1]All authors are affiliated with the Centre for Systems and Control, Indian Institute Of Technology, Bombay sweksha.jain@iitb.ac.in

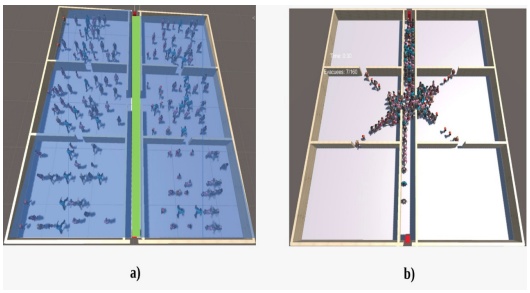


Fig. 1: a) The green region is free area and the blue region is obstacle area. b) Simulation of human evacuation in a crowded environment during emergency, where humans are guided by obstacle position to evacuate through free area to reach safe area (exit). Figures taken from [16].

- Developed a novel decentralized, swarm self-clustering algorithm for guiding the robots to a safe area by obstacles placement in danger area to prevent clustering. The self-clustering is achieved by independent decision-making by each agent using only range sensing without communication.
- A novel strategy to identify between robots and obstacles solely based on range sensing in situations appropriate for the proposed algorithm.
- Result analysis through Monte-Carlo simulations with varying levels of obstacle occupancy in danger area to evaluate convergence percentage of cluster formation within the safe area.

This paper is organized as follows: the problem on self-clustering to be guided to a safe area is presented in Section II along with problem formulation. The detailed methodology of the algorithm is presented in Section III. Section IV provides experimental results, and corresponding observations and analysis. This section also presents observable similarities with human behavior during evacuation. Section V concludes with the features of the algorithm and future work.

## II. Problem Formulation

Given the swarm of robots with sensing capabilities in the bounded environment, every robot is equipped with limited sensing FoV with angle $2\Delta$, and sensing range $R$. The limited FoV is considered to make the problem more realistic and the corresponding solution realizable in the real world. Each robot detects its neighbors within its sensor's FoV, illustrated in Figure 2, depicting the interactions of two robots: $i$, and $j$. An avoidance distance $D_a$ is the *minimum distance* between two robots to avoid inter-robot collisions. Figure 2 illustrates a scenario where the distance of robot $i$ concerning robot $j$ is less than the sensing range. Hence, robot $j$ gets detected by robot $i$ but the converse is not true. Further, the command inputs to the robot $i$ are linear velocity $v_i$ and angular velocity $\omega_i$. The goal distance $D_g$ is the *maximum distance* to the goal for a robot to stop.

The following conditions should be satisfied for the robot $j$ to get detected by the robot $i$:

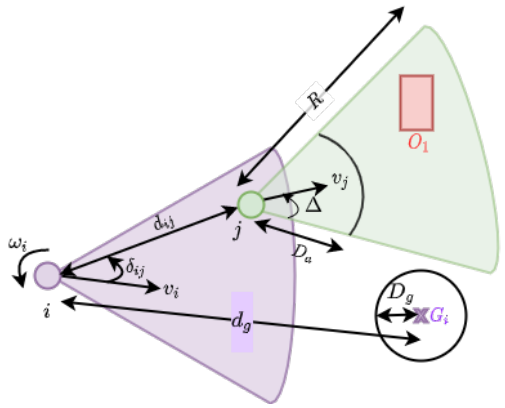


Fig. 2: Representation of Inter-robot detection, where $i, j$ represents the 2D robots with conic FoV with sensing range $R$, angle $2\Delta$, and a avoidance distance $D_a$. The distance and bearing of robot $i$ with respect to robot $j$ are $d_{ij}$ and $\delta_{ij}$, respectively. The distance of robot $i$ to its instantaneous goal position $G_i$ is $d_g$. $O_1$ represents the obstacle detected by robot $j$.

- Inter-robot distance $d_{ij}$ is less than the sensing range of the robot, i.e., $d_{ij} < R$, and;
- The bearing of the robot $i$ with respect to the robot $j$ is less than the half of FoV angle of the robot $i$ i.e. $\delta_{ij} < \Delta$.
- If robot $i$ is not occluded by any other robot $k$ or obstacle, given the above two conditions are satisfied.

The problem statement is formulated as follows:

***Problem 2.1:*** **Design a communication-free, decentralized swarm algorithm for a bounded environment with static obstacles. The swarm of robots must autonomously form clusters in the "safe area" by just utilizing range sensing thus using minimal information. This self-organized behavior should emerge without prior knowledge of the areas marked as safe or dangerous.**

## III. Methodology

In this work, we develop an algorithm to enable self-clustering ensuring that the clusters are formed in a safe area i.e., the region devoid of obstacles. The algorithm utilizes these obstacles' positioning to prevent cluster formation in the danger area. The obstacles signify danger, which should be avoided effectively and the area covered within the obstacles is termed "danger area". Similar to real-life cases where the safer place during evacuation is unknown to residents apriori, the robots are unaware of the safe area apriori. The proposed algorithm aims to prevent forming clusters near any obstacle or wall present effectively showcasing the self-organized behavior of a swarm of robots.

The proposed approach extends the direction of our previous work [17], where the main objective was to form self-organized swarm clusters without using any inter-robot communication by utilizing range sensing only. The methodology proposed in [17] ensures the swarm of robots organizes itself into swarm clusters autonomously. In this work, our

objective is to form swarm clusters in an environment consisting of obstacles, where the main challenge is to identify the object detected by range sensing only. The proposed algorithm requires each robot to perform the following tasks: (i) *Detection and Identification*, (ii) *Estimation*, (iii) *Goal Assignment*, and (iv) *Guided Navigation*, to form clusters in the safe area. Each task listed above is discussed in the next sections.

### A. Detection and Identification

After initialization, the robot instantly starts detecting its surroundings. The robot detects any object only when the object lies in the robot's FoV. After detecting any object, the identification of the object detected becomes necessary. To do this, we have developed a novel identification strategy for a robot.

*Proposed Identification Strategy:* For this strategy, we have assumed that the obstacles present in the environment are not of the same size as that of the swarm of robots. For a given swarm of homogeneous robots, we gather data from 2D LiDAR sensor for each object detected and then group the measurements as shown in Figure 3a). The difference between two consecutive measurements creates the groups; if the difference exceeds a predefined threshold, a new group is formed. The proposed strategy calculates the aspect ratio denoted by $a_r$, shown in Figure 3b), for each object detected. The indicated aspect ratio $a_r$ is calculated as the fraction of the width of the robot $c$ and the center range measurement $d$. Given values of $a, b,$ and $\theta$ from the LiDAR data, we calculate the length of c in triangle abc using the cosine rule and aspect ratio:

$$\begin{aligned} c &= \sqrt{a^2 + b^2 - 2ab\cos\theta} \\ a_r &= \frac{c}{d} \end{aligned} \tag{1}$$

We observed that due to the homogeneity of robots, the aspect ratio for all the robots is within bounds such that $a_r \in (a_{r_{min}}, a_{r_{max}})$. Hence, the robot or static obstacle is identified by calculating the aspect ratio. When $a_r \in (a_{r_{min}}, a_{r_{max}})$, the detected object is a robot, else a static obstacle. The calculation of the $a_r$ for the robot model considered is shown in the results section.

*Definition 1:* A neighbor set of robot $i$ is defined as:

$$N_i = \{j \mid d_{ij} < R; \delta_{ij} < \Delta; a_{r_{min}} < a_r < a_{r_{max}}\} \tag{2}$$

where $d_{ij}$ represents the inter-robot Euclidean distance and $a_r$ is the aspect ratio.

**Remark 1:** The robot identifies among other robots and obstacles by utilizing the homogeneity of robots in the swarm.

According to the proposed algorithm, the robot will form its neighbor set if and only if the detected object is identified as a robot. The neighbor will contain the estimated positions of the identified robot.

**Remark 2:** After identification, the robots form the neighbor set only with other identified robots, not obstacles.

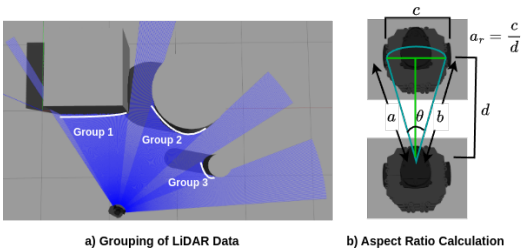


Fig. 3: Grouping of LiDAR data and Aspect Ratio calculation

### B. Estimation

The position of the identified robot $j$, denoted by $\{x_j, y_j\}$, is given in (3), with respect to the origin (local frame of robot $i$). Since obstacles can never be a neighbor to the robots, their positions will not be estimated.

$$\begin{aligned} x_j &= d_{ij}\cos(\delta_{ij}) \\ y_j &= d_{ij}\sin(\delta_{ij}) \end{aligned} \tag{3}$$

### C. Goal Assignment

After initialization, the robot is assigned a uniformly distributed random goal within the environment boundary if no object is detected or if the robot reaches the goal without encountering a neighbor. In both cases the neighbor set is empty. When a robot identifies another robot the goal assigned is the centroid of the positions of both the robots. For centroid calculation, the estimated position of the identified robot is used. On the other hand, if it identifies an obstacle/wall then the goal remains the same. The goal assignment for each robot is done in a decentralized manner where every robot makes independent decisions to reduce its distance to goal $d_g$ with its assigned goal. A goal is assigned to a robot based on certain conditions shown in (4). It is to be noted that $|N|$ is the cardinality of the neighbor set.

$$\textit{Goal Assignment} = \begin{cases} \text{Random}, & \text{if } |N| = \phi \\ \text{Centroid}, & \text{if } |N| \neq \phi \;\&\; d_g > D_g \\ \text{No change}, & \text{otherwise} \end{cases} \tag{4}$$

### D. Guided Navigation

The navigation of robots should effectively avoid collision with other robots and the obstacles present in the environment along with reaching their instantaneous goal. For this task, any controller to reach a goal with collision avoidance can be used. The controller used for implementation is taken from [18]. The navigation of a robot follows a simple strategy such that a collision avoidance controller is applied if another robot is identified within the avoidance distance $D_a$. Meanwhile, the *wall-following strategy* is utilized when an obstacle or wall gets identified within the avoidance distance $D_a$. The proposed algorithm capitalizes on the positioning of static obstacles such that the resulting swarm clusters are formed outside of the danger area. The proposed algorithm proceeds in such a way that the robot navigates to its goal

avoiding collisions, and will automatically get guided by the obstacles placement to form swarm clusters in the "safe area".

The proposed algorithm combines all the tasks described above to provide a guided swarm self-clustering strategy presented in Algorithm 1. The proposed algorithm is decentralized and it gets implemented to each robot independently. After initialization, robot $i$ is assigned a uniformly distributed

**Algorithm 1** Guided Swarm Self-Clustering Algorithm for robot $i$

**Input** $R,\ D_g,\ \Delta,\ D_a$
**Output** $v_i, \omega_i$

1: **while** True **do**
2: $x_{g_i} \leftarrow rand(Boundary)$
3: $y_{g_i} \leftarrow rand(Boundary)$
4: $\{v_i, \omega_i\}$ = Navigate to goal
5: $d_{g_i} = \sqrt{x_{g_i}^2 + y_{g_i}^2}$ ▷ distance to goal of robot $i$
6: Calculate $a_r$
7: **if** $a_r$ $exists$ **then** ▷ Object Detected
8: $N_i$ = Identify_Neighbors$(R, \Delta, a_r)$
9: **if** $a_{r_{min}} < a_r < a_{r_{max}}$ **then**
10: $|N|$ Updates
11: $\{x_{g_i}, y_{g_i}\} \leftarrow Centroid(\{x_i, y_i\}, \{x_j, y_j\})$
12: **else** ▷ Wall/Obstacle Detected
13: Wall Following strategy
14: **end if**
15: **if** $d_g \leq D_g$ **then** ▷ Goal is reached
16: **if** $|N_i| \geq 1$ **then** ▷ Neighbor present
17: $\{x_{g_i}, y_{g_i}\} = \{0, 0\}$
18: **else**
19: $x_{g_i} \leftarrow rand(Boundary)$
20: $y_{g_i} \leftarrow rand(Boundary)$
21: **end if**
22: **end if**
23: **end if**
24: **if** $\{x_{g_i}, y_{g_i}\} = \{0, 0\}$ **then**
25: $\{v_i, \omega_i\}$ = $\{0, 0\}$
26: **else**
27: $\{v_i, \omega_i\}$ = Navigation_Controller$(x_{g_i}, y_{g_i})$
28: **end if**
29: **end while**

random goal within the environment boundary (lines 2-3). The robot starts navigating towards its assigned goal till it detects an object (lines 4-6). The robot starts identifying the object using the proposed identification strategy (line 8). If robot $i$ identifies another robot $j$, the goal is updated using the positions of both the robot $i$ and robot $j$ by calculating the centroid (lines 9-11). If it identifies a wall/static obstacle, it follows the wall-following strategy (lines 12-13). The robot since its initialization constantly calculates its instantaneous distance to goal $d_g$. When this distance to goal becomes equal to the predefined goal distance $D_g$, the robot checks for the number of neighbors (lines 15-16). If the robot has atleast a neighbor then it stops forming a cluster with its neighbor (lines 16-17), else it again gets assigned a random goal. While navigating a navigation controller navigates the robot to its assigned goal (lines 24-28).

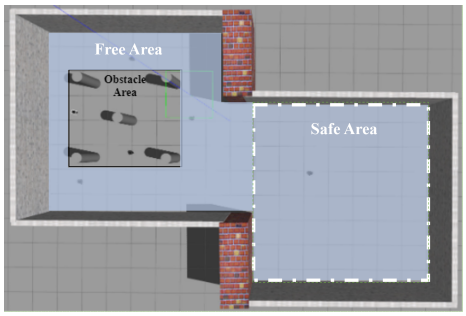


Fig. 4: Simulation setup highlighting the "free area" using a blue shaded region and the "safe area" with a white dashed rectangle.

## IV. Results

The simulations are obtained on $12^{th}$ Gen Intel® Core™ i7-12700F × 20 processor with the Robot Operating System (ROS Noetic version) and Gazebo on Ubuntu 20.04 LTS. The robot is Turtle-bot3, a differential drive robot equipped with a 2D LiDAR. The environment considered is bounded and the robots are unaware of their surroundings. One of the environment setups considered for the simulations is shown in Figure 4. The robots are supposed to form clusters in the area without any obstacle nearby (*expected region of convergence or safe area*). The placement of obstacles is not done by any defined strategy but to make sure the danger area is covered by the obstacles.

### A. Aspect Ratio for Robot Identification

We have considered a swarm of Turtle-bot3 robots, spawned in the Gazebo environment. The aspect ratio range will change for different robot models because the width of the robot will vary. To find out the range of this ratio for the Turtle-bot3, we considered conic FOV with fixed angle of 120° and sensing range varying from $R$ to $D_a$. We calculated the ratio by placing the robots at different positions and orientations within the FoV. It was observed that the aspect ratio follows a certain pattern and it lies between $0.03 < a_r < 0.19$. Hence, if the $a_r \in (0.03, 0.19)$ for any object detected, then it is a robot, else a static obstacle.

**Remark 3:** The proposed identification strategy may fail for the cases where a portion of the obstacle gets detected and becomes equal to the robot's width.

### B. Performance Metric and Safe Area Calculation

The metric used for performance evaluation and analysis of the proposed algorithm is the Percentage of Convergence (PoC). The PoC is defined as the percentage of instances in which the robots successfully converge within the expected region of convergence i.e., the safe area. To generate results and calculate PoC, we calculate the desired region of convergence *"safe area"*. For instance, a set of static obstacles in the environment $O_i$, is given by:

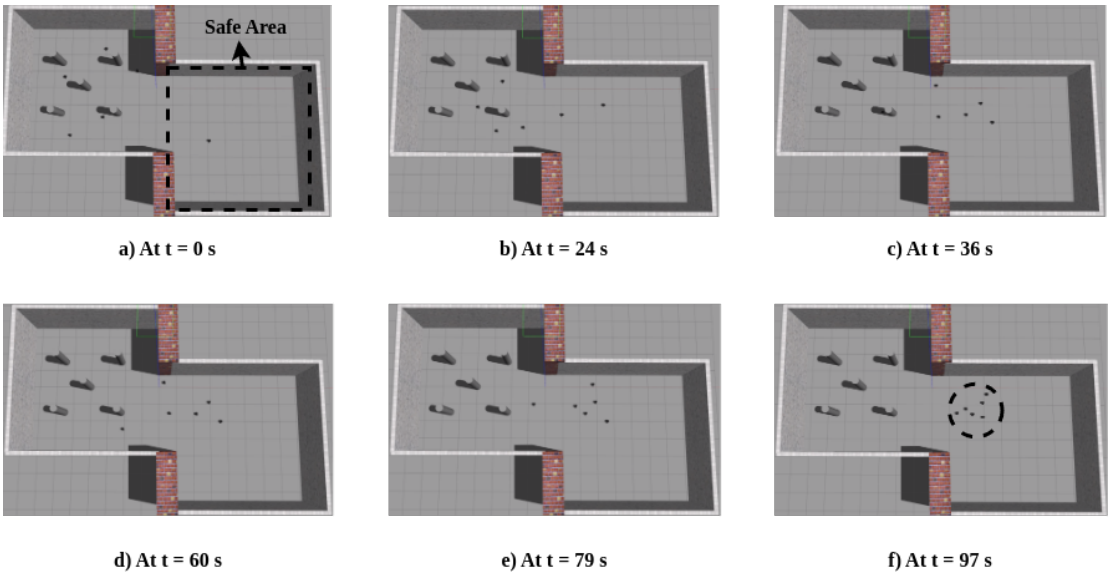


Fig. 5: Guided swarm clusters formation for $S = 6$, with $R = 3$ and $2\Delta = 120°$.

$$O_i = \{O_1, O_2, ....., O_n\} \tag{5}$$

Each obstacle is represented using the coordinates of the center of mass, such that, $O_1 = \{x_1, y_1\}$. The area covered with the obstacles is calculated using the positions of the static obstacles (can be imported from the Gazebo environment), which is equal to the total area excluding the area covered by obstacles. The pseudo-code to calculate the safe area is given by Algorithm 2.

**Algorithm 2** Algorithm For Safe area Calculation

**Input** *Bounded Environment*
**Output** *Safe Area*

1: **while** True **do**
2: $O_i = \{\{x_1, y_1\}, ...., \{x_i, y_i\}\}$
3: $Obstacle\ Space = polygon(O_i)$
4: $Obstacle\ Area = convex\ hull(Obstacle\ Space)$
5: $Free\ Area = Total\ Area \setminus Obstacle\ Area$
6: $Safe\ Area = Largest\ Rectangle\ Area(Free\ Area)$
7: **end while**

The safe area calculation described in Algorithm 2 steps, is as follows:

- Collect each obstacle coordinate.
- Each coordinate is considered as the vertices to create a polygon.
- The convex hull of the polygon gives the required area covered under the obstacles.
- The free area is the area excluding the obstacle area, and the area of the largest rectangle in the free area is "safe area".

### C. Simulation Results

The simulation results at different time instances are shown in Figure 5. The proposed methodology capitalizes on the static obstacle positions finally controlling the convergence area of the clusters formed. To validate, we performed Monte-Carlo simulations by varying the number and positions of the obstacles (consisting of 40 runs for each case), where the PoC calculation is enlisted in Table I. It is observed that the proposed method achieves a PoC of more than **85%**(average), by placing the obstacles in a way that completely covers the "danger area". It must be noted that all the simulations are done on a physics-based simulator which provides the results as good as the real world. However, by increasing the number of obstacles, the percentage of convergence decreases drastically but can be controlled with the constraint on the minimum cluster size, denoted by $M$. It can be observed that the swarm of robots then achieves a higher PoC than that without the constraint.

TABLE I: Percentage Of Convergence with the parameter values considered for implementation are $v_{max} = 0.22$ m/s, $D_g = 0.875$ m, $D_a = 0.775$ m, $R = 3$ m, and $2\Delta = 120°$.

| Swarm Size | M | No. Of Obstacles | PoC |
|---|---|---|---|
| 6 | 2 | 4 | 91.67 % |
| 6 | 2 | 5 | 88.75 % |
| 9 | 2 | 6 | 90 % |
| 9 | 2 | 8 | 64.38 % |
| 9 | 3 | 8 | 100 % |

The visualization of the final convergence of robots in one such experiment is shown in Figure 6, where the swarm size considered is $S = 6$, number of obstacles is 5, and $M = 2$. All the sensor and algorithm design parameters considered are the same as for experiments listed in Table I. The safe area calculation is done using Algorithm 2. The axes and the area for the plot are equivalent to the simulation scenario shown in Figure 4. The plot indicates the final positions of all the robots. It can be observed that all the robots have converged to the center-right of the environment which is

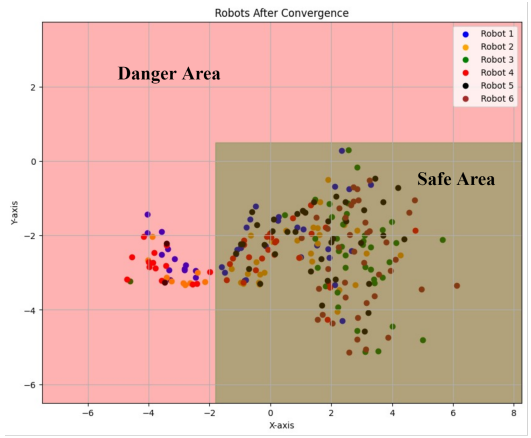


Fig. 6: Final positions of the robot after convergence, where the colored dots represent the final position of robots after convergence.

the desired region of convergence, the "safe area" for the majority of runs.

We have observed the behavior of the swarm of robots not just in terms of performance but also in terms of collective intelligence. It has been observed that in high-stress scenarios such as evacuation during a fire emergency, humans tend to achieve collective self-organized behavior even without communicating and limited field of view [16], [19]. In [19], the authors conducted a study to understand human behavior in stress scenarios. They concluded that when the crowd was split unevenly by the placement of the obstacles, participants tended to follow the majority of the crowd. This was likely because humans can perceive dynamic information more easily than static information, particularly when they are dealing with high uncertainty and their attention capacity is reduced. The proposed algorithm enables the swarm of robots to showcase similar behavior where robots always follow the crowd to form clusters in the "safe area", guided by the placement of obstacles.

## V. Conclusions

This paper presents a novel decentralized *guided safe swarm self-clustering algorithm* for forming clusters without inter-robot communication. The algorithm utilizes only sensory inputs to prevent the formation of swarm clusters in the *danger area*. The proposed algorithm is robust and scalable as showcased by showing the cluster formation by making independent decisions using sensory inputs alone and the higher convergence rate in the "safe area". More importantly, the prevention of cluster formation in the "danger area" is obtained by introducing simple obstacles. This work showcases that the proposed robots' behavior can be prototype to understand human behavior in emergency scenario. Hence it will find its applications in studying human behavior in emergencies for developing efficient evacuation or search and rescue strategies especially when the individual agents are intending to follow the crowd. Future work will include extending this work to calculate the minimum number of obstacles guaranteeing swarm cluster formation. Another promising direction for future work is to establish stronger baseline comparisons with human behavior.

## VI. Acknowledgements

We thank Jayesh Khalane for his contribution in developing robot identification strategy, during his e-Yantra internship at IIT Bombay.

## References

[1] Iñaki Navarro and Fernando Matía. An introduction to swarm robotics. *International Scholarly Research Notices*, 2013(1):608164, 2013.

[2] Levent Bayındır. A review of swarm robotics tasks. *Neurocomputing*, 172:292–321, 2016.

[3] Vito Trianni, Roderich Groß, Thomas H Labella, Erol Şahin, and Marco Dorigo. Evolving aggregation behaviors in a swarm of robots. In *Advances in Artificial Life: 7th European Conference, ECAL 2003, Dortmund, Germany, September 14-17, 2003. Proceedings 7*, pages 865–874. Springer, 2003.

[4] Michael Bodi, Ronald Thenius, Martina Szopek, Thomas Schmickl, and Karl Crailsheim. Interaction of robot swarms using the honeybee-inspired control algorithm beeclust. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):87–100, 2012.

[5] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.

[6] 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.

[7] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7:1–41, 2013.

[8] Sanza Kazadi, A Abdul-Khaliq, and Ron Goodman. On the convergence of puck clustering systems. *Robotics and Autonomous Systems*, 38(2):93–117, 2002.

[9] C Lee, M Kim, and Sanza Kazadi. Robot clustering. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1449–1454. IEEE, 2005.

[10] Roderich Groß, Stéphane Magnenat, and Francesco Mondada. Segregation in swarms of mobile robots based on the brazil nut effect. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4349–4356, 2009.

[11] Manish Kumar, Devendra P. Garg, and Vijay Kumar. Segregation of heterogeneous units in a swarm of robotic agents. *IEEE Transactions on Automatic Control*, 55(3):743–748, 2010.

[12] Fabrício R. Inácio, Douglas G. Macharet, and Luiz Chaimowicz. Pso-based strategy for the segregation of heterogeneous robotic swarms. *Journal of Computational Science*, 31:86–94, 2019.

[13] Aleksis Liekna, Janis Grundspenkis, et al. Towards practical application of swarm robotics: overview of swarm tasks. *Engineering for rural development*, 13:271–277, 2014.

[14] Najmeh Neysani Samany, Mahdi Sheybani, and Sisi Zlatanova. Detection of safe areas in flood as emergency evacuation stations using modified particle swarm optimization with local search. *Applied Soft Computing*, 111:107681, 2021.

[15] 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):1–18, 2018.

[16] Coşkun Şahin, Jon Rokne, and Reda Alhajj. Human behavior modeling for simulating evacuation of buildings during emergencies. *Physica A: Statistical Mechanics and its Applications*, 528:121432, 2019.

[17] Sweksha Jain, Rugved Katole, and Leena Vachhani. Swarm synergy: A silent way of forming community, 2023.

[18] Dhruv Shah and Leena Vachhani. Swarm aggregation without communication and global positioning. *IEEE Robotics and Automation Letters*, 4(2):886–893, 2019.

[19] Jing Lin, Runhe Zhu, Nan Li, and Burcin Becerik-Gerber. Do people follow the crowd in building emergency evacuation? a cross-cultural immersive virtual reality-based study. *Advanced Engineering Informatics*, 43:101040, 2020.