A Decentralized Mobile Computing Network for Multi-Robot Systems Operations

Jabez Leong Kit[1], David Mateo[2], Roland Bouffanais[3]
Engineering Product Development
Singapore University of Technology and Design
Singapore, Singapore
[1]jabez_leong@mymail.sutd.edu.sg,[2]david_mateo@sutd.edu.sg,[3]bouffanais@sutd.edu.sg

***Abstract*—Collective animal behaviors are paradigmatic examples of fully decentralized operations involving complex collective computations such as collective turns in flocks of birds or collective harvesting by ants. These systems offer a unique source of inspiration for the development of fault-tolerant and self-healing multi-robot systems capable of operating in dynamic environments. Specifically, swarm robotics emerged and is significantly growing on these premises. However, to date, most swarm robotics systems reported in the literature involve basic computational tasks—averages and other algebraic operations. In this paper, we introduce a novel Collective computing framework based on the swarming paradigm, which exhibits the key innate features of swarms: robustness, scalability and flexibility. Unlike Edge computing, the proposed Collective computing framework is truly decentralized and does not require user intervention or additional servers to sustain its operations. This Collective computing framework is applied to the complex task of collective mapping, in which multiple robots aim at cooperatively map a large area. Our results confirm the effectiveness of the cooperative strategy, its robustness to the loss of multiple units, as well as its scalability. Furthermore, the topology of the interconnecting network is found to greatly influence the performance of the collective action.**

***Index Terms*—collective computation, decentralized computing network, mobile computing network, swarm robotics**

I. Introduction

Many biological systems are known to be capable of performing highly complex computations in a fully decentralized fashion: e.g. the brain, some insect colonies, aggregating amoeboid cells, etc [3]. For instance, collective turns in flock of birds is one such kind of collective decision making achieved through a fully decentralized computational process. Specifically, each bird in a flock detects the direction of travel of some local neighboring conspecifics (according to a given interaction distance). This flow of behavioral information reaching each bird is then processed by the bird's central nervous system according to specific innate behavioral rules that involve some forms of computation of the available data. This complex process can be better understood when considering basic models of flocking based on self-propelled particles (SPPs). In Vicsek's model for instance, the direction of travel of all agents located within a given radius—a metric interaction distance—is simply averaged—the computational task—by each agent [18]. It is important to appreciate the fact that flocking birds (and SPPs) are essentially networked units, and that the topology of the underlying network of interaction plays a pivotal role in the effectiveness of the collective behavior [11].

Over the past five years, we have been witnessing dramatic advances in sensors, digital signal processing capabilities, low-cost single-board computers, storage devices, low-power communication devices. Simultaneously, the cost of hardware has been following an ever-decreasing trend. Combined with unabated developments in robotic software (the Robot Operating System, ROS, is celebrating its 10th anniversary this year), all these rapid technological advances are revolutionizing our ability to build massively distributed, rapidly deployable, self-calibrating multi-robot systems. This paves the way for a fundamental paradigm shift in robotics, in which large, costly, and task-specific robots are replaced by swarms of small, low-cost, and versatile units [17].

Swarm robotic systems allow unsophisticated, low-cost, modular robotic platforms to be dynamically reconfigured into a group capable of achieving a range of effective and responsive cooperative actions well beyond the capabilities of the individuals [4], [5]. This so-called "power of masses" requires the constituting units to sense and interact with the environment, while also sharing the sensed data with the rest of the swarm, thereby enabling a very effective form of collective computation. This paradigm of decentralized operations, inspired from natural swarms, offers the possibility of performing global collective computations under a wide range of group sizes (scalability), despite the possible sudden loss of multiple agents (robustness), and under unknown and dynamic circumstances (flexibility) [4]. Scalability, flexibility, and robustness are indubitably appealing features as they open the door to the possibility of operating over very large scales, in fully unstructured and dynamic environments, with progressive and graceful degradation of the systems operations in the presence of adverse conditions. However, it is critical to acknowledge the fact that the achievement of complex cooperative operations by swarm robotics systems hinges on our ability to tap into the effectiveness of their collective computation capabilities—well beyond the trivial averaging process performed during collective motion.

This work was supported by a MOE-Tier 1 grant #T1MOE17001. JLK is supported by a Presidential Graduate Fellowship from the Singapore MOE.

From a network perspective, collective computing is a paradigm sought after when the classical centralized computing paradigm (with a master node and slaves, i.e. the star network) of parallel computing breaks down [12]. That explains that the research community is actively exploring moving away from Cloud computing, and exploring Edge computing capabilities for certain applications that include mobile robotics. However, with dynamic networks of mobile sensors, robots or vehicles, Edge computing demands to leverage resources—computers, sensors, actuators—that may not always be available or connected to the network. Edge computing still requires a few servers throughout the network to serve the sensor clusters, thereby offering only a partial decentralization of the system. Therefore, Collective computing is one step ahead of the highly sought after Edge computing, bringing about complex computations all the way to the end nodes of the network. This significantly reduces communication resources required by Cloud computing to perform most complex computational tasks. More importantly, Collective computing yields an inherently robust, scalable and flexible computing paradigm since these critical features are natural by-products of the underpinning decentralized swarming framework.

In this paper, we introduce the concept of collective computing and present, in detail, one particular embodiment meant to characterize and illustrate the potential of this approach in terms of scalability and robustness. The considered application is collective mapping by a swarm of networked robots, with varying connectivity rules—i.e. varying the topology of the interconnecting network, and with imposed failure of up to 75% of the nodes of the system.

II. Collective Computing and Operation

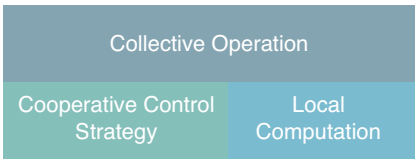

Fig. 1. Schematic of the Collective Operation and Computation framework. Although the cooperative control strategy and the local computation are shown here, for clarity, as two separate boxes, they happen to be deeply intertwined as a consequence of its distribution throughout the interconnecting network.

Swarming is known to be a powerful paradigm to achieve a scalable, robust, and flexible *collective operation* performed at spatial (and possibly temporal) scales much larger than the characteristic scales at which single agents operate. Both natural and artificial swarming systems typically operate by combining a decentralized *cooperative control strategy* with a set of *local computations* that each agent performs using only information from a certain local neighborhood.

For instance, flocks of birds are able to keep a consistent but flexible formation during long journeys without the need for a central controller. This behavior can be modeled by a collective operation with a cooperative control strategy based on global alignment of the birds' direction of motion. To obtain this global alignment, each agent only has to compute the average direction of motion for its local neighborhood, say its k closest neighbors, and the global, collective movement emerges. The concept of combining local computation with cooperative control strategy can be illustrated with an example of how a swarm determine its direction of motion based on *Vicsek's model* [18]. For instance, in [11], the local computation of (1) of the swarm shown computes the direction of motion of agent i at time $t + \Delta t$ with shared inputs from its neighbors (see Fig. 2, by finding the average of the difference of its neighbors and its own direction of motion of time t. The updated direction of motion will then be shared with its k nearest neighbors through a network shown in Fig. 2 for the on-going local computations. It is worth highlighting that the topology of the network is embedded in the definition of the local update rule (2) and is therefore an integral part of the cooperative control strategy. The properties of this network directly influence the local computational task involved but not its fundamental nature—e.g. for SPPs the computational task is an averaging but over more or less neighbors depending on the local density of agents within the radius of interaction. Moreover, The local computation outcome affects the dynamics of the node which affects the k nearest neighbor network and those of the swarming units connected to it through a complex propagation process. Both the dynamics of the node and the network defines the cooperative control strategy. It appears clearly that a swarming system operates with local computation and cooperative control strategy.

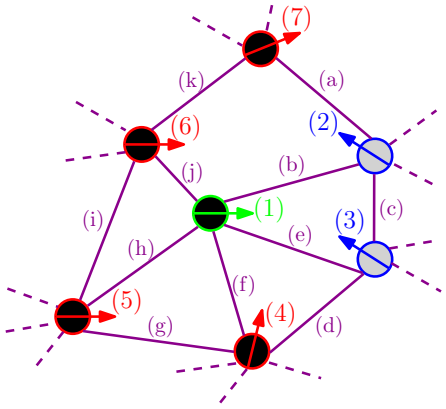

Fig. 2. Schematic of a subset of collectively moving agents within a swarm. Arrows show the direction of travel and straight lines represent the existence of an interaction link between any two agents. Agent (1) receives behavioral information from neighboring agents (2)–(7). If one assumes that agents (2) and (3) are picking up an external signal—e.g., oncoming obstacle—and are responding to it thereby triggering meaningful behavioral information in the form of changes to the agents heading represented by blue arrows. This meaningful information reaches agent (1) directly through network edges (b) and (e). This direct signal is reinforced by means of positive feedback loops such as (d)-(f), (a)-(k)-(j), (d)-(g)-(h) and many others. At the same time, agent (1) receives behavioral information from the bulk of the swarm represented by red-colored agents such as (4), (5), (6), and (7) for instance.

$$\theta_i(t+\Delta t) = \theta_i(t) + \frac{\Delta t}{k}[(\theta_j(t) - \theta_i(t)) + \cdots + (\theta_{j+k+1}(t) - \theta_i(t))] + \eta\xi_i(t) \quad (1)$$

where $\eta\xi_i(t)$ is a Gaussian white noise of magnitude η since $\xi_i(t) \in [-\pi, \pi]$.

This framework provides a useful guide to review existing robotic functions as collective operations and consider not only the local operations but how different models of cooperative control affect the efficiency of the collective operation. Figure 1 shows the modularity of the framework, yet hides the actual entanglement of both processes due to its distributed nature throughout the interconnecting network underpinning the process.

The effectiveness of the cooperative control strategy depends on (i) the dynamics of the agents, and (ii) the interaction between them. In network-theoretic terms, this means that the collective operation will be determined by both the dynamics *on* the network and the dynamics *of* the network itself.

In the next section, we present an application of the introduced Collective computing framework for applications of collective mapping by a swarm of networked mobile robots with particular attention paid to scalability and robustness.

A. Collective Mapping

Simultaneous Localisation and Mapping (SLAM) is a complex problem that has received sizable attention from the research community. This attention has produced some elaborate and efficient solutions for the problem in the case of single-robot SLAM. Recently, these results have been expanded to the multi-robot case in a framework that is referred to as multi-robot SLAM [16].

There are essentially two approaches in the multi-robot SLAM research. One is to propose new SLAM methods to handle multiple noisy sensors and improve the data association [8]—i.e., filtering [10], scan matching [6], and map-merging [2] techniques. The other is to focus on new multi-robot architectures with efficient data structures [1] and distributed computing robotic clusters [8], [12]. In order to apply these potential solutions to a wide range of environments, attention must be paid to assure that they guarantee not only the robustness but also the scalability of the approach.

Current multi-robot SLAM solutions require a sizable amount of sensor data processing from a central server, that sometimes is performed *a posteriori*, i.e. after the robots have completed their explorations. This requires reliable communication with a central controller, thereby severely limiting the scalability and applicability of the method. To overcome this serious limitation, one needs a decentralized approach.

In this section, we present the application of the Collective computing framework introduced in the previous section to the task of collective mapping, with simulations of swarming robots allowing us to analyze and assess its effectiveness. We intend to continue the testing of collective mapping with our swarm of ground vehicles [5], which has been designed with the ability impose any kind of network topology [14]. Therefore, the concepts presented here have been considered with real-life robotic implementations and applications in mind.

B. Simulation

We assume perfect sensory data and localization from the simulated robots. The local map of each individual robot is classically built with an occupancy grid approach [7] based on their simulated infra-red (IR) sensors. Each grid cell contains the probability of finding an obstacle. The collective map, defined as the union of all the robots' local maps, is assembled and built to study the efficiency of the approach. However, it is important to stress that it is not needed and that it plays no role in determining the robots' behavior. The simulated "sensor range" is a tile of 6×7 grid cells around it.

To perform the collective mapping operation, each robot computes its own local map based on the environmental data gathered from its own sensors and its neighbors' sensors, with the concept of "neighbor" understood in its network sense, i.e. the units directly connected to a robot through the underlying network of interaction (see Fig. 2). With this updated map, each robot decides independently its next target location by means of a frontier exploration algorithm [19]. While this process does not explicitly take into account the position of the neighboring robots, the information gathered from them does affect the map and thus the target location. Lastly, the robot runs an A* path planning algorithm [9] in order to reach the target location. The full process is summarized in Algorithm 1.

Algorithm 1 Collective Mapping algorithm

```
1: procedure CollectiveMapping(r, R, k, s, map)
2:     map ← UpdateOccupancyGrid(map, s)
3:     neighbors ← KNearestNeighbors(r, R, k)
4:     for n in neighbors do
5:         map ← UpdateOccupancyGrid(map, s_n)
6:     target ← FrontierExploration(map, r)
7:     path ← PathPlanning(r, target)
8:     return map, path
```

The interaction network that determines which robot transmits data to which, is defined using a k-nearest neighbor scheme, meaning that at any particular time-step a robot uses the sensing information from its closest k robots. This means that the network is directed (agent i may be using information from j without j using that of i), and dynamic (the network depends on the position of the robots, and is thus affected by their movement). Moreover, it is worth adding that with the particular choice of the k-nearest neighbors as interacting units, the network has a spatial embedding [3].

This scheme has been proven in [14] to be key in reproducing the collective behavior of natural swarming systems, and to provide surprisingly robust and responsive behaviors with a minimal amount of connections. For low values of k, the *instantaneous* networks generated by this scheme are

typically disconnected (see Fig. 6), but the dynamic stitching of the neighbors allows for the system to be connected over time as the system dynamically evolve over the large area to map.

The robots only share their current sensed data, meaning that when a robot connects to another it does not receive the history of measurements nor the current local map of the neighbor: this process is Markovian. Formally, the local map of robot i at time t is obtained by computing the posterior probability $p(m|S_t^i)$ for a collection of sensory data such that

$$S_t^i = \bigcup_{t'=1}^{t} \{s_j(t'); j | a_{ij}(t') = 1\} \tag{2}$$

with $a_{ii}(t) = 1 \, \forall t$.

As this work focuses on the swarming aspect of collective mapping, we have intentionally simplified the (local) mapping technique with a set of assumptions. Local map-merging can be considered when the robots do not know the relative positions of other robots. As the individual robot has limited processing power, we can distribute the heavy computational work, and thus propose a new strategy that requires less resources.

C. Results

In the various simulation runs, a swarm of robots is deployed within a Basilica, whose floor layout is shown in Fig. 3, and is tasked to map the interior of the building in a collective manner. As the robots explore the environment, each unit builds its own partial map, see Fig. 4. The termination criterion for the simulation is that the union of these maps covers a 100 % of the environment. Note that this termination criterion does require the full assembled map but in practice the swarm of robots shall continue mapping to detect possible dynamic changes in the layout. Hence, the termination criterion is only used here to assess the effectiveness of our proposed collective mapping approach.

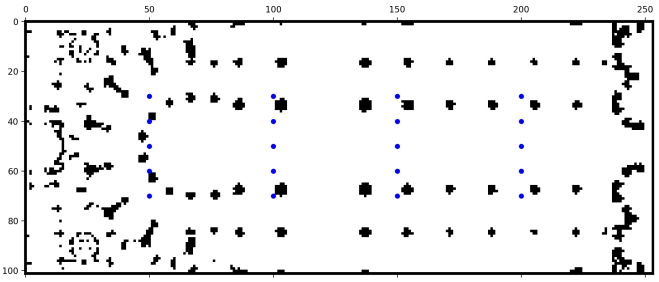

Fig. 3. An interior map of a Basilica used for the Collective Mapping operation. The blue dots represent the initial positions of the simulated robots.

1) Scalability: To test scalability, we run simulations with $k = 1$ nearest neighbor network topology and change the number of robots from 5, 10, 15, to 20. The termination criterion is that the collective map is 100% completed. The results are shown in Table I. As the number of robots increases, the collective mapping duration reduces. From 15 robots to 20

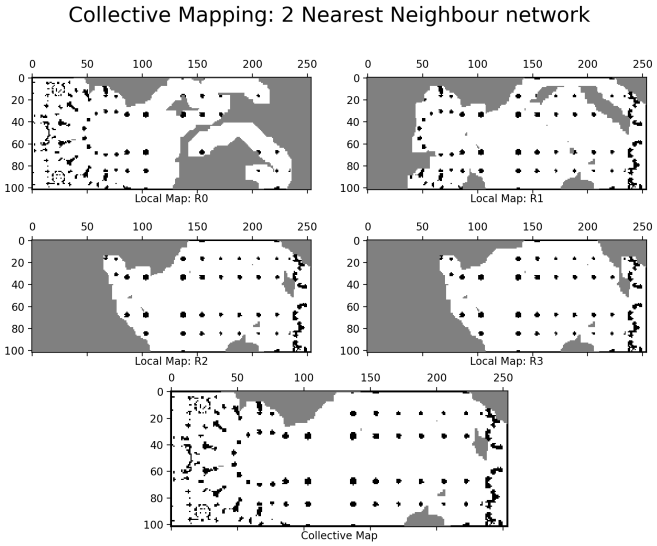

Fig. 4. Examples of the different local maps computed by each robot on a given run when interacting with their $k = 2$ closest neighbors.

robots, the decrease in number of iterations is not as significant due to the limiting factor of the environment space.

2) Robustness: To test robustness, we run simulations with $k = 2$ nearest neighbor network topology. Figure 5 shows the evolution of map coverage with time for three cases: a swarm of 20 units, a swarm of 5 units, and a swarm that starts with 20 and where 5 robots are regularly removed from it every 50 iterations. The 20-15-10-5 robots graph shows that our collective mapping approach is robust, even when robots are dropping out of the network during the operation, the system is able to complete 100% of the mapping. Interestingly, we can see that the 20 robots graph and 20-15-10-5 robots graph separated right after 5 robots are removed at iteration 50. The new 15 robots system started to slow down in its operation, and even more after subsequent removals. Initially, the 20-15-10-5 system operates well above the 5 robots system, after all the removals, both the system converges and completed the collective mapping 50 iterations apart.

3) Network Effect: In Table II one can see the average number of iterations taken to fulfill this criterion for different number of nearest neighbors. The simulations are based on 15 robots connected via the different network topology. Chain is a static network where each robot is connected to two other particular robots. 0NN means there is no interaction between robots at all with "NN" standing for "nearest neighbors". From 1NN to 6NN, these are the $k > 0$ nearest neighbor dynamic networks. The results in Table II shows the drastic improvement from a 0NN to $(k > 0)$NN, from a totally independent system to a swarming system. From the results, it is observed that the dynamic network are more efficient than a static network in performing collective mapping, and it becomes more efficient as k is increased. Based on this initial study, 5NN seems to be the optimal network for this operation. However, extensive study can be carried out to find the optimal

network as in [14].

No. of robots	5	10	15	20
No. of Iterations	1273	982	583	562

TABLE I
SCALABILITY TEST OF COLLECTIVE MAPPING: NUMBER OF ITERATIONS NEEDED FOR THE COLLECTIVE TO MAP 100% OF THE FLOOR.

Network	Chain	0NN	1NN	2NN	3NN	4NN	5NN	6NN
No. of Iterations	585	1092	583	469	344	322	303	310

TABLE II
EFFECT OF NETWORK ON COLLECTIVE MAPPING

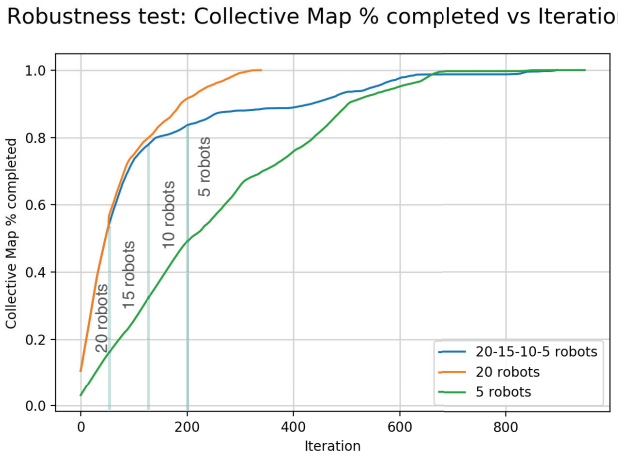

Fig. 5. Robustness test result showing the completion of Collective Mapping despite a disrupted network(20-15-10-5 robots graph)—i.e. removing 5 agents every 50 iterations.

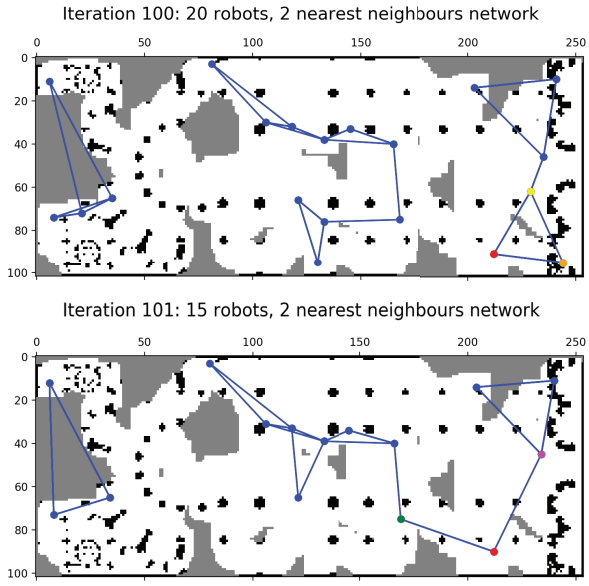

Fig. 6. Two snapshots of the dynamic network obtained during the simulations. While at any given iterations the system is likely split in disconnected clusters, this instantaneous clusters do communicate with each other thanks to the switching network. In this example, Robot 14 (red) changes its neighborhood from Robot 15 (yellow) and Robot 19 (orange) to Robot 11 (magenta) and Robot 13 (green).

III. DISCUSSION

Applying swarming concepts to robotic mapping operations opens interesting possibilities for the collective mapping of very large areas with a large number of robotic units. The intrinsic properties of swarming systems suits the intended principles of multi-robot SLAM. This motivated us to adapt the robotic mapping from the swarm robotics perspective. We applied a collective operation and computation framework to the mapping operation to define the collective mapping based on a decentralized process requiring only local computations combined with a cooperative control strategy, also defined locally for each unit. This collective mapping concept is thoroughly assessed and tested with simulations. The results show that this approach provides a decentralized, robust, and scalable mapping framework, beyond what is currently possible with Edge computing. As a next step, we will consider investigating the flexibility of the swarming system during this collective mapping process, and study its capacity to map changing environments.

In designing multi-robot systems, a great deal of attention is paid to the dynamics of the agents with relatively less attention focused on the effects of dynamic networks. In most cases, the infrastructure is so as to guarantee a static network that provides a constant and reliable information flow to a central station. However, it is known that different kinds of connectivities and degrees distributions can affect drastically the performance of the system. In particular, it has been shown that limiting the number of interacting agents increases the

system's responsiveness [13], [14].

Distributed heavy computations on swarming system is of particular interest in the area of environmental sensing. To this aim, we are currently developing a fault-tolerant robust collective computing framework suited for multi-robot systems and with fully decentralized operations in dynamic environments. As the system is meant to operate in the physical world, and is equipped with advanced sensors for environmental data collection, it can perform reconstruction or prediction of a quantity of interest depending on the collective task at hand. Such high-level applications tend to be complex, and thus require significant computational resources, which can be a deterring factor for many existing multi-robot system. Yet, this novel capability would have significant implications for the collective operation and system adaptability to changing circumstances and dynamics environments. It is worth adding that the proposed robust collective computing framework under development, given its distributed nature, should preserve robustness and flexibility. Scalability will be dependent on the effectiveness of the distributed network of communication. We reckon that this is the next advancement for multi-agent systems, also paving the way for AI-based operations, e.g. using collective reinforcement learning [15].

From the hardware perspective, all computations are performed by single-board computers placed in each agent, and in the absence of any central computer or any other supporting infrastructure. The system is expected to be capable of continuing and completing its robust collective computation with the removal or addition of any number of nodes. As a proof of concept, we successfully implemented and tested this robust collective computing framework on a swarm of buoys, developed by our group [20], connected by means of a low-bandwidth dynamic mesh network. The presented work on Collective computing is therefore the cornerstone to this robust collective computing framework. Both are designed to be platform- and environment-agnostic, i.e. they are completely independent of (i) the robotic platform's hardware, (ii) the environmental equation that the system aims to solve, and (iii) the numerical method considered for the solution of the partial differential equations governing the environmental model.

IV. Conclusion

We have presented a collective computing framework applied to the complex task of collective, distributed mapping. In contrast with current approaches, this framework allows for a completely decentralized and distributed mapping, thus affording scalability to autonomous systems operating in large environments. Simulations of this framework show the effectiveness, scalability, and robustness of the cooperative strategy where agents interact through a dynamic, switching network topology with fixed degree.

References

[1] Pratik Agarwal, Gian Diego Tipaldi, Luciano Spinello, Cyrill Stachniss, and Wolfram Burgard. Robust map optimization using dynamic covariance scaling. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 62–69. Ieee, 2013.

[2] Andreas Birk and Stefano Carpin. Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE*, 94(7):1384–1397, 2006.

[3] R. Bouffanais. *Design and Control of Swarm Dynamics*. Springer, Heidelberg, 2016.

[4] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[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. *Front. Robot. AI*, 4:Art. 12, 2017.

[6] Albert Diosi and Lindsay Kleeman. Laser scan matching in polar coordinates with application to slam. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3317–3322. IEEE, 2005.

[7] Alberto Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In *Proceedings of the Sixth Conference on Uncertainty in AI*, volume 2929, page 6, 1990.

[8] Bruno Duarte Gouveia, David Portugal, Daniel C Silva, and Lino Marques. Computation sharing in distributed robotic systems: A case study on slam. *IEEE Transactions on Automation Science and Engineering*, 12(2):410–422, 2015.

[9] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[10] Andrew Howard. Multi-robot simultaneous localization and mapping using particle filters. *The International Journal of Robotics Research*, 25(12):1243–1256, 2006.

[11] M. Komareji and R. Bouffanais. Resilience and controllability of dynamic collective behaviors. *PLoS one*, 8:e82578, 2013.

[12] Ali Marjovi, Sarvenaz Choobdar, and Lino Marques. Robotic clusters: Multi-robot systems as computer clusters: A topological map merging demonstration. *Robotics and Autonomous Systems*, 60(9):1191–1204, 2012.

[13] D. Mateo, Y. K. Kuan, and R. Bouffanais. Effect of correlations in swarms on collective response. *Sci. Rep.*, 7:10388, 2017.

[14] David Mateo, Nikolaj Horsevad, Vahid Hassani, Mohammadreza Chamanbaz, and Roland Bouffanais. Optimal network topology for effective collective response. *arXiv preprint arXiv:1807.04631*, 2018.

[15] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

[16] Sajad Saeedi, Liam Paull, Michael Trentini, and Howard Li. Multiple robot simultaneous localization and mapping. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 853–858. IEEE, 2011.

[17] Alan C Schultz and Lynne E Parker. *Multi-robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 NRL Workshop on Multi-robot Systems*. Springer Science & Business Media, 2013.

[18] Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel type of phase transition in a system of self-driven particles. *Phys. Rev. Lett.*, 75(6):1226, 1995.

[19] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151. IEEE, 1997.

[20] B. M. Zoss, D. Mateo, Y. K. Kuan, G. Tokić, M. Chamanbaz, L. Goh, F. Vallegra, R. Bouffanais, and D. K. P. Yue. Distributed system of autonomous buoys for scalable deployment and monitoring of large waterbodies. *Auton. Robot.*, 2018. In Press.