Search and Rescue Operations Using Robotic Darwinian Particle Swarm Optimization

Arjun S Kumar, Gayathri Manikutty, Rao R Bhavani
AMMACHI Labs
Amrita School of Engineering, Amritapuri
Amrita Vishwa Vidyapeetham, Amrita University, India
arjunskumar22@gmail.com, gayathri.manikutty@ammachilabs.org, bhavani@amrita.edu

Micael S Couceiro
Ingeniarius, Lda.,
Institute of Systems and Robotics,
Coimbra, Portugal
micael@ingeniarius.pt

***Abstract*—Swarm robots have the potential to be utilized as a part of various applications due to the obvious advantages they offer, namely their resilience, their adaptability to different environments, and their reduced reliance on humans, particularly for hazardous or laborious tasks such as search and rescue operations (SaR). Several particle swarm optimization (PSO) algorithms have been proposed and developed for controlling a robot swarm to achieve the desired behavior in SaR operations. In this paper, we present an implementation of the previously proposed Robotic Darwinian Particle Swarm Optimization (RDPSO); an exploration algorithm which overcomes the limitation of convergence on multiple targets. A simulation of a SaR mission is presented using Robot Operating System (ROS) and Gazebo, with a visualization in Rviz using intensity maps that mimic real-world scenarios, such as victim identification by voice/sound intensity mapping, localization of fire source by temperature intensity mapping, and identification of sources of radioactive leaks by radiation intensity mapping and experiments were conducted using RPSO and RDPSO algorithms for two scenario missions, that is, scenario with two victims and another with four victims. Experimental result shows that RDPSO has better performance compared to RPSO for multiple target SaR operations.**

***Index Terms*—swarm robotics, ros, search and rescue operations.**

I. Introduction

Multiple robot systems are widely used for search and rescue operations (SaR),as well as coverage and exploration due to reduced cost of individual agents, the ability to decompose tasks to different agents within the system to cover wider spaces in a short time span and the inherent system robustness owing to the redundancies in the system. Typically, such systems are faster and more efficient than single robot systems, although the efficiency of such systems is highly dependent on the algorithms used.

In this paper, we implemented a ROS framework for SaR operations using autonomous ground swarm robots. We have simulated a real life search and rescue missions with multiple static targets and swarm robots. These robots do not have any prior knowledge of the location of victims or fire outbreaks in the environment and have only the knowledge of their initial positions, initial velocities and the environment to be explored. The universal grid map library is used to create an intensity map for the autonomous navigation of swarm robots. Two exploration algorithms were used to perform SaR operations and evaluated their performance in SaR operations with multiple targets.

Over the years, the number of solutions proposed in the literature encompassing an ever-increasing number of cooperating robots, such as swarm robots, has been rising dramatically. Swarm robotics is the study of how a large group of simple robots work together for executing complex tasks which are difficult for a single robot to execute. Their behavior is inspired by social biological beings that can execute a task that cannot be executed alone by an individual being such as an ant searching for food, bees making hives, etc. In such social groups, like swarms, there is no leader-follower system and all individuals work together to finish their task. It is a decentralized and distributed system where each individual shares its information with all other members in the swarm - for example, for executing a particular task, every individual needs to be able to share information with the swarm [1]. Each individual is aware of its immediate surroundings very well, such as sources of food, sources of threat etc. Nevertheless, they do not have any overall idea of the global scene.Fig 1 depicts some examples of swarm behaviours in natures.

In 1986, Craig Reynolds implemented the first simulation of bird flocking behaviour using an algorithm denoted as BOIDS which was based on three principles [2].

- Velocity matching: Keep on par with velocity of neighbors.
- Flock centering: Always steer to the centre of the group.
- Collision avoidance: Avoid colliding with the neighbors.

BOIDS used the above three rules to determine how they would move and a combination of these simple behaviours lead to the emergence of complex group behaviours. These three rules, referred to in the literature as cohesion, separation and alignment rules, are used even today for flocking simulations. A global version of Reynolds work led to the design of the original Particle Swarm Optimization (PSO) algorithm.

PSO is a stochastic optimization technique wherein each particle updates its velocity towards the best performing particle by comparing its fitness value associated to the entire swarm population globally [3] [4]. It has been effectively

978-1-5090-6367-3/17/$31.00 ©2017 IEEE

utilized as a part of numerous applications, such as robotics [5], stock market prediction, estimation of thyroid volume [6] and eye tracking system [7]. It works perfectly well with a single target operations, but when exposed to multi-target scenarios, it is unable to breakaway from sub-optimality.

Fig 1. Swarm Behaviours in nature.

The parameters to configure a PSO algorithm includes:

- The number of particles
- Dimension of particles' search space
- Range of particles' search space
- Inertial, cognitive and social coefficients
- The number of iterations
- The stopping condition

The particles or robots have to find an optimal solution by maintaining a communication with all other members in the swarm. Each robot or particle follows the best performing member in their group that is, the one that has obtained the maximum success. The above parameters highly influence how closely the solution meets the global optimum and the intelligence of the swarm to obtain global best. Many extended versions of the PSO have been suggested and developed in the past for controlling a robot swarm to overcome the problem of local solution and achieve the desired behavior in SaR operations. Among the other proposed algorithms, the Darwinian Particle Swarm Optimization (DPSO) [8] is the first extended version of PSO which shows the capability to get away from sub-optimal solutions by adopting a Darwinian approach [9].

Some of the other solutions include a Bacteria Foraging Algorithm (BFA) with PSO based approach for route planning and distribution of multiple robots in an environment with non-dynamic obstacles [10]. However, by using BFA the authors were only able to enhance the local search. They carried out the experiment with three robots but the mission did not have any sub optimal conditions. Also, in their work they did not consider collision avoidance or communication factor [11]. In [12], the authors implemented swarm aerial robotics using modified PSO under ROS framework. They have considered only single target scenarios in which the target object is either stationary or moving with respect to the quad-copters.

Recently, Couceiro et al [13] extended their work on optimizing the DPSO algorithm and came up with RDPSO (Robotic Darwinian PSO) for multi-robot applications, such as obstacle avoidance, dynamic path planning, etc. The authors work demonstrates the applicability of the RDPSO algorithm by dynamically splitting the robot population into smaller groups, thereby decreasing the communication complexity between the robots, while avoiding premature convergence. This algorithm has been proven to be scalable to a large population of robots. The swarm also does not possess any central agent to coordinate with other agents, which is another merit of the RDPSO algorithm.

II. SYSTEM DESCRIPTION

A. Robotic Particle Swarm Optimization (RPSO)

In RPSO a fitness function $f(x_n)$ is used iteratively to evaluate proposed solutions. Each particle remembers their local best solution obtained and makes this information available to its teammates. The algorithm tracks the global best solution obtained by considering all particles.

$$\begin{aligned} v_n[t+1] &= w_n * v_n[t] + c_1 * r_1 * (X_1[t] - x_n[t]) \\ &\quad + c_2 * r_2 * (X_2[t] - x_n[t]) \\ x_n[t+1] &= x_n[t] + v_n[t+1] \end{aligned} \tag{1}$$

Every particle or robot n navigates in a two dimensional search space according to a position (x_n) and velocity (v_n), which are highly dependent on the local (or cognitive) best success X_1 and global (or overall) best X_2 success.

w_n is the inertial parameter.

c_1 is the cognitive parameter.

c_2 is the social parameter.

r_1 and r_2 are the random weights ranges from (0,1).

The c_1, cognitive component is the personal best known success of each robot and c_2, social component is the best known success of the whole population or swarm.

In his work, Ventor [14] stated that a smaller value of c_1 and larger value of c_2 enhances the execution of the algorithm. In multiple target rescue missions, if the estimation of c_2 is such that it is much larger than c_1, then it will drag every robot in the swarm to a local solution from which they will not able to escape since they behave like "visually impaired" adherents. Conversely, if the value of c_1 is much larger than c_2, then it makes every robot get pulled into its very own best position which results in excessive wandering. Hence initial values for c_1 and c_2 must be very carefully chosen.

B. Robotic Darwinian PSO (RDPSO)

$$\begin{aligned} v_n[t+1] &= w_n * v_n[t] + c_1 * r_1 * (X_1[t] - x_n[t]) \\ &\quad + c_2 * r_2 * (X_2[t] - x_n[t]) \\ &\quad + c_3 * r_3 * (X_3[t] - x_n[t]) \\ &\quad + c_4 * r_4 * (X_4[t] - x_n[t]) \\ x_n[t+1] &= x_n[t] + v_n[t+1] \end{aligned} \tag{2}$$

Robotic Darwinian PSO (RDPSO), an extension of RPSO, takes into account obstacle avoidance factor and communication factor between the robots in the swarm. It dynamically enables the creation of groups within the swarm, or other swarms, using a reward and punishment mechanism. Hence, RDPSO permits a decrease in the amount of required information to be shared among robots [15].

In eq. 2, c_3 is the coefficient of the obstacle restraint component and c_4 is the coefficient of the communication component. X_3 denotes the best obstacle avoidance position vector. X_4 denotes the best communication position vector. In real life scenarios, we use parameters such as the intensity of voice, temperature, radioactivity etc in SaR to rescue the victims or humans [16]. In RDPSO, all the robots will initially explore (either solitarily or in small groups) in random directions until they receive a signal, say the sound of the human voice or intensity of fire outbreaks. Once any robot receives a known signal, it will compare the signal intensity with all of its partners to see which robot detected the maximum signal intensity. For instance, the robot that is closest to the victim will receive the maximum value of intensity. This robot is treated as the best performing robot of the swarm and will be rewarded by giving a partner robot. Other robots that are socially excluded wander in random directions until all the area has been covered. By this way, search and rescue operations are performed. All parameters of RPSO and RDPSO in 2D search space is illustrated in Fig 2.

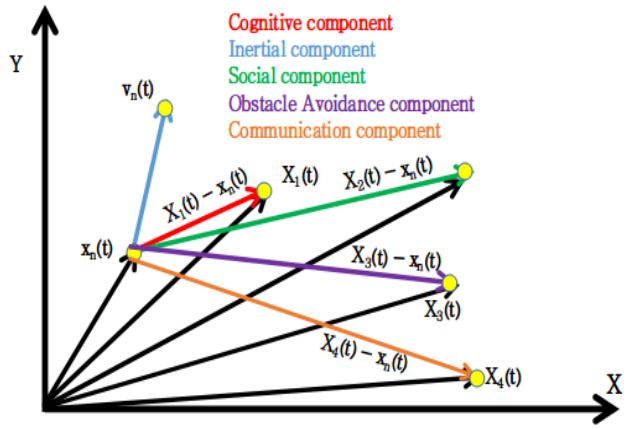

Fig 2. RPSO & RDPSO components in 2D space.

C. Gazebo and Rviz

Gazebo [17] is a well-known simulator in Robot Operating System (ROS) [18] which supports the open source evaluation of robotic approaches, thus allowing to simulate complex implementations like testing new algorithms, designing new robotic models, etc. It also provides tools to analyze the applicability of robotic algorithms. ROS visualization tool known as Rviz is a tool for visualizing sensory data, for example, camera data, data from distance measuring devices, GPS information, etc. Rviz also displays the state transitions occurring in an algorithm [19].

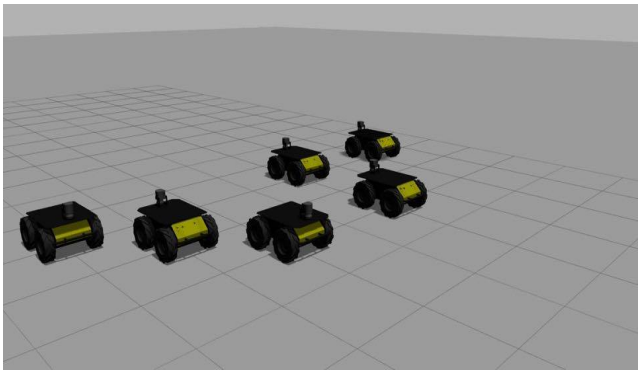

Fig. 3 Swarm of Husky Models in Gazebo.

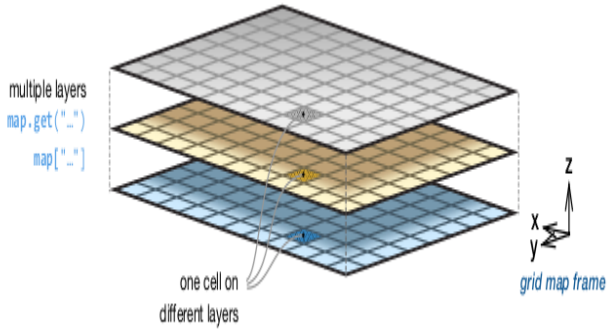

Fig. 4 Multi-layered Grid Map.

The grid map used for simulation purpose is a well known universal grid map, available in ROS library and developed by Robotics System Lab, ETH Zurich [20]. The grid map was designed in such a way that it is applicable for any robotics implementation especially for autonomous navigation. Each grid has a specific value associated with it which helps in implementing various navigation algorithms. The grid also provides multiple data layers signifying different parameters such as elevation, variance, color, occupancy etc. as shown in Fig 4.

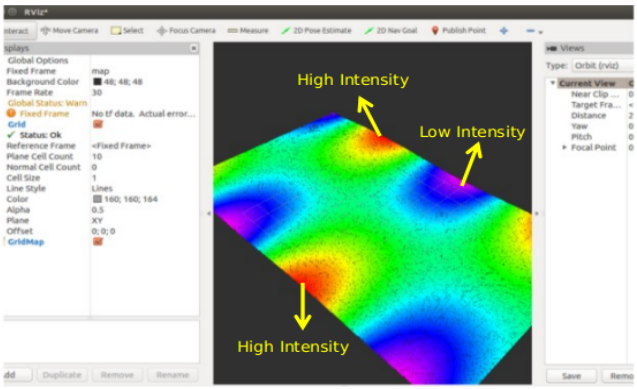

Fig. 5. Intensity Map in Rviz.

For this work, we have used a single layer with color intensity (for representing target) with a grid size of 40*40 m. It also follows the VIBGYOR color spectrum where the red color represents the maximum intensity and violet represents the least intensity. The color of each grid cell represents the

intensity of the signal at that point, which helps in tracking the progress of the algorithm in a visual manner. Fig 5. shows intensity map in Rviz where each grid represent an intensity value.

III. Experimental Results

The husky models, as shown in Fig.3, a popular unmanned ground vehicle, from Clearpath robotics were used for simulation in Gazebo using the two proposed algorithms; RPSO and RDPSO. A comparison was done between the two algorithms for a two victim rescue scenario and a four victim rescue scenario with 30 trials for each case. The victims were assumed to be static in all the trials. For the four victim rescue scenario, eight robots were used. For the two victim rescue scenario, four robots were used. Each phase of the simulation has been described in the subsequent sections and main differences between the two algorithms has been explained.

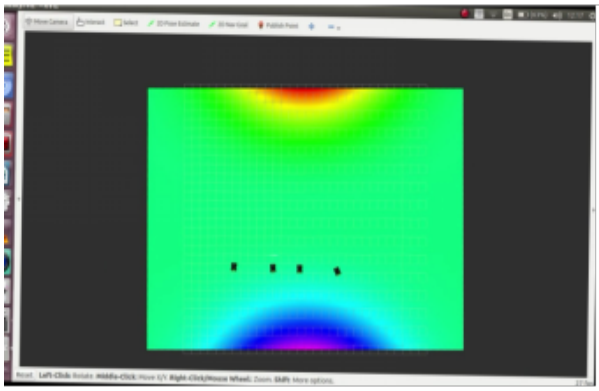

Fig. 6 Initial Stage of robots using RPSO in Rviz.

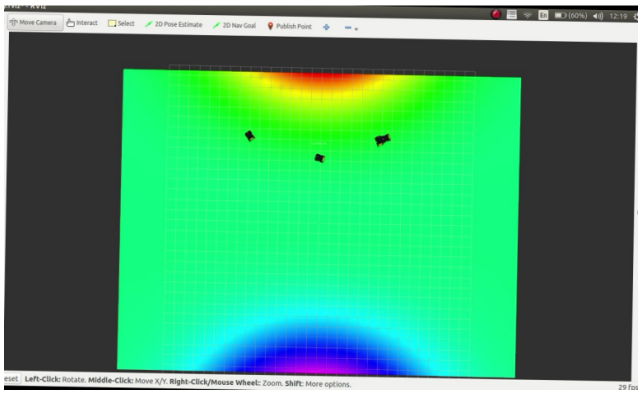

Fig. 7 Collision Stage of robots using RPSO in Rviz.

Fig 6 depicts the initial stage of the robots that are searching for the maximum intensity target locations. Fig 7 shows how the robots collided with each other before the final stage shown in Fig 8. This is because the RPSO algorithm does not take the communication parameter or obstacle avoidance parameter into consideration. In both the cases, that is, in both the four victim and the two victim rescue case, the RPSO algorithm got trapped in a local-minima. As explained earlier, this is one of the drawbacks of the RPSO that the simulation highlighted. All the robots were trying to concentrate at a single region from which they were unable to escape leading to a sub-optimal solution. In SaR operations, this algorithm cannot be used as the swarm will try to rescue victims only in a single region while overlooking other regions. In contrast, the RDPSO implementation converged to an optimal solution most of the time.

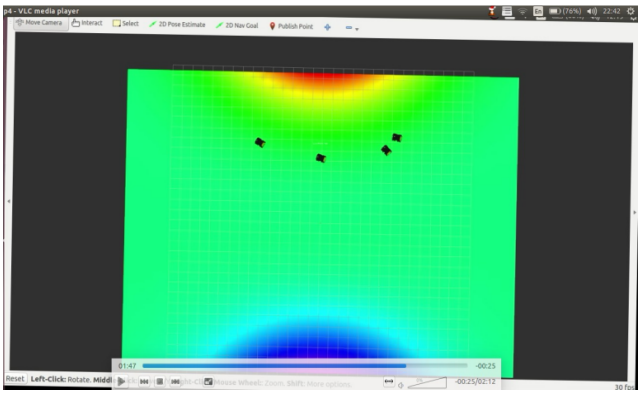

Fig. 8 Final stage of robots using RPSO in Rviz.

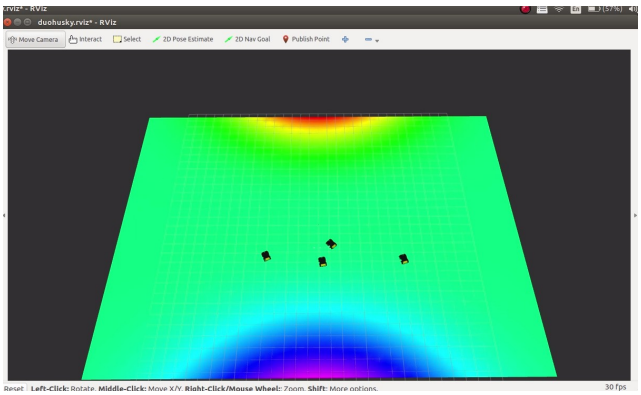

Fig. 9 Initial stage of robots using RDPSO in Rviz.

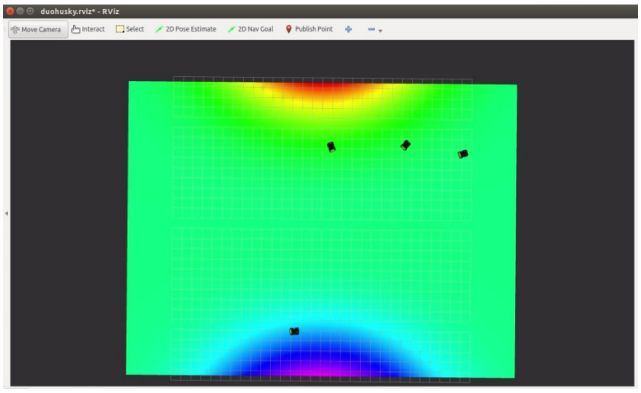

Fig. 10. Final stage of robots using RDPSO in Rviz.

Fig. 9 depicts the initial simulation phase and Fig. 10 depicts the finial or stopping stage of RDPSO algorithm, in which one robot escaped from the global optimum and moved to local optimum, thus ensuring every victim is rescued. The simulation results in Fig 11 shows that the RDPSO seeks out the optimal solution faster than RPSO. We have used ten minutes as the stopping condition of this system. The RPSO algorithm consumes a longer time to convergence for a single

target SaR operations while the RDPSO consumes nearly half of its time for the multiple SaR operations.

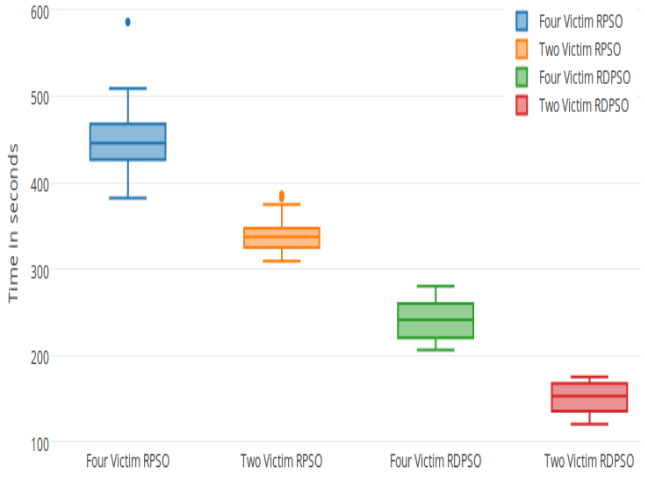

Fig. 11. Comparison of time taken by RPSO and RDPSO.

IV. Conclusion

To the best of the authors' knowledge, this is the first work to evaluate the RDPSO algorithm under the ROS framework under a realistic simulator, such as Gazebo. Swarms of robots pose numerous challenges for human operators, especially when distributed algorithms with complex dynamics are used. As such, enabling human operators to control robot swarms is still an open problem. Using ROS and Gazebo, we present the comparison of RPSO and RDPSO exploration algorithm with multiple robots swarms in ROS visualization (RViz) tool considering several real-world scenarios such as victim identification by voice, localizing on fire source by temperature sensing, identification of sources of radioactive leaks etc.

From the simulation results, one can clearly see that two swarms each having two robots evolved from a single initial swarm of four robots. This later split into two swarms of three and one robot with the help of RDPSO's rewards and punishment mechanism. We used four and eight husky robots for simulation of two real life scenario mission which can be further extended to any number of robots.

We made an assumption that we already have the map of the region to be explored but we do not have any prior knowledge of the victim location in the map. We used the universal grid map library to create the intensity map. The above results show that the RDPSO allows the robot to escape from sub-optimal solutions and overcome the problems in RPSO such as its inability to detect multiple target locations and collisions. Also, the performance of the RDPSO will not be affected by the distribution of the actual target locations since the algorithm makes no assumption about the targets.

V. Future Work

As a future work, we could move the work out of simulation into the real world with a Husky robot swarm. We can also perform comparison studies to see how RDPSO stacks up against other swarm robot algorithms in the real world. In this paper, we have assumed that the victim or gas leakage or fire outbreak is static. This could be extended to handle dynamic targets and an evaluation of the algorithms could be done for this condition as well.

VI. Acknowledgments

We wholeheartedly thank the researchers at AMMACHI Labs & Ingeniarius,Lda for their helpful feedback and important contributions throughout different phases of this work.

References

[1] Navarro, Inaki, and Fernando Matia, *An introduction to swarm robotics*, 3rd ed. ISRN Robotics, 2013.
[2] Reynolds, Craig W. "Flocks, herds and schools: A distributed behavioral model." ACM SIGGRAPH computer graphics 21.4 (1987): 25-34.
[3] Kennedy, James, and Russell Eberhart. "Particle swarm optimization." Neural Networks, 1995. Proceedings., IEEE International Conference on. Vol. 4. IEEE, 1995.
[4] Du, Ke-Lin, and M. N. S. Swamy. "Particle swarm optimization." Search and Optimization by Metaheuristics. Springer International Publishing, 2016. 153-173.
[5] Araujo, Andre, et al. "Integrating Arduino-based educational mobile robots in ROS." Autonomous Robot Systems (Robotica), 2013 13th International Conference on. IEEE, 2013.
[6] Geetha, K., and S. Santhosh Baboo. "Efficient thyroid disease classification using differential evolution with SVM." Journal Of Theoretical And Applied Information Technology 88.3 (2016): 410.
[7] Amudha, J., and K. R. Chandrika. "Suitability of Genetic Algorithm and Particle Swarm Optimization for Eye Tracking System." Advanced Computing (IACC), 2016 IEEE 6th International Conference on. IEEE, 2016.
[8] Couceiro, Micael, and Pedram Ghamisi. "Particle swarm optimization." Fractional Order Darwinian Particle Swarm Optimization. Springer International Publishing, 2016. 1-10.
[9] Couceiro, Micael S., Rui P. Rocha, and Nuno MF Ferreira. "A novel multi-robot exploration approach based on particle swarm optimization algorithms." Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on. IEEE, 2011.
[10] M. Lovbjerg, T. Krink, Extending particle swarms with self-organized criticality, in: Proc. IEEE Congr. Evol. Comput., vol. 2, 2002, pp. 1588-1593.
[11] Du, Ke-Lin, and M. N. S. Swamy. "Bacterial Foraging Algorithm." Search and Optimization by Metaheuristics. Springer International Publishing, 2016. 217-225.
[12] Ma'sum, M. Anwar, et al. "Autonomous quadcopter swarm robots for object localization and tracking." Micro-NanoMechatronics and Human Science (MHS), 2013 International Symposium on. IEEE, 2013.
[13] Couceiro, Micael S., et al. "A fuzzified systematic adjustment of the robotic Darwinian PSO." Robotics and Autonomous Systems 60.12 (2012): 1625-1639.
[14] G. Venter, Particle swarm optimization, in: Proceedings of 43rd AIAA/ASME/ASCE/AHS/ASC Structure, Structures Dynamics and Materials Conference, 2002, pp. 22-25.
[15] M.S. Couceiro, N.M.F. Ferreira, J.A.T. Machado, Fractional order Darwinian particle swarm optimization, in: 3th Symposium on Fractional Signals and Systems, FSS 2011, Coimbra, Portugal, 2011.
[16] Couceiro, M. S. (2015). An overview of swarm robotics for search and rescue applications. Handbook of Research on Design, Control, and Modeling of Swarm Robotics, 345.
[17] Koenig, Nathan, and Andrew Howard. "Design and use paradigms for gazebo, an open-source multi-robot simulator." Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on. Vol. 3. IEEE, 2004.
[18] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.
[19] Kam, Hyeong Ryeol, et al. "RViz: a toolkit for real domain data visualization." Telecommunication Systems 60.2 (2015): 337.
[20] Fankhauser, Peter, and Marco Hutter. "A universal grid map library: Implementation and use case for rough terrain navigation." Robot Operating System (ROS). Springer International Publishing, 2016. 99-120.