Collective Robotics Search using Particle System

Ahmed Ibraheem Abdulkareem †

Control and Systems Engineering Department, University of Technology

E-mail: ahmedibraheem1978@gmail.com

Received: 2/06/2014

Accepted: 11 /09 /2014

Abstract – This work introduces the implementation of particle system to be simulated to work as a group of unmanned mobile robots (swarm robots). These robots are able to locate a specified target in the predefined environment with high efficiency when driven by an optimized Particle Swarm Optimization (PSO) algorithm. The application of the particle system to the mobile robots to search for a target in the environment is called Collective Robotics Search (CRS) problem. The main benefit of this application is to evolve better solutions than using single robot through the collective interaction of all robots between them to achieve the searching task successfully. Particle system has been chosen in this work to employ the mobile robots in the CRS problem due to its simplicity and easy to implement. To measure the performance of this simulation, a simple obstacle free environment will be used to implement behaviors of the group of mobile robots when those robots are used to search for a single target. The results of this work show that applying PSO to a CRS problem in off-line and on-line approaches are efficient in terms of minimum error and also minimum number of iterations during the evolutionary process.

Keywords- Swarm intelligence, collective search, robotics, particle swarm optimization, exploration and exploitation.

.

† *This paper has been presented in ECCCM-2 Conference* and accredited for publication according to IJCCCE rules.

1. Introduction

Biologists and computer scientists in the field of artificial life have studied various types of animal behavior in nature, such as flocks of birds crossing the sky, groups of ants performing collective transport, groups of ants forming a bridge, and schools of fish swimming, turning, and fleeing together. They have attempted to understand how such social animals interact, achieve goals and evolve [1]. Their studies have shown that social animal behavior has a decentralized control, a lack of synchronicity and simple identical members. The conclusion of their studies is that the term "swarm" not only describes a name but also distinguishes that type of group [2]. In 1989, Beni and Wang described the term "Swarm Intelligence" as a type of artificial intelligence based on the collective behavior of decentralized and self- organized systems. This term has become more and more widely used over the last twenty years [3].
Wide applications of swarm intelligence-based algorithms have been presented such as the Ant Colony Optimization algorithm (ACO) proposed by Dorigo in 1992 in his PhD thesis [4], and the Particle Swarm Optimization algorithm (PSO) proposed by Kennedy and Elberet in 1995 [5]. These algorithms have been used to solve problems in combinatorial and continuous optimization, telecommunication and robotics. The application of swarm principle to the robots is called "Swarm Robotics", which are consists of many individual robots that interact strongly with each individual robot to achieve some purposeful behavior and a collective goal [1]. While there is no formal definition for swarm robotics, there are some characteristics that should be presented in a system in order for it to be generally accepted as swarm robotics. These characteristics include decentralized control, large numbers of robot members, high reliability, robust and cheap, collective behavior and communication capabilities, able to perform difficult tasks such as exploring and locating a particular target that is difficult or impossible for a single robot to achieve [6].
Several research studies have been carried out in the past to show how the collective search behavior between swarm robotics could be achieved in the environment. Brooks [7] and Liu and Passino [1] have considered the importance of collective behavior in swarm robots for solving complex problems such as box-pushing [8], and collective construction [9] by using a group of simple, decentralized, self-organized robots that interacted together to achieve an effective collective search.
The aim of this work is to implement the PSO algorithm on real time small mobile robots when they have been used in target search applications such as Collective Robotic Search (CRS), to explore and locate a specified target in the physical environment that is difficult or impossible for a single robot to achieve due to their constraints. Moreover, the performance of CRS behaviors has been demonstrated using two approaches. The first approach is to simulate the PSO algorithm using particles as mobile robots in off-line mode to influence one another's search for the specified target problem. While the second approach is to apply the PSO algorithm in on-line mode using real robots to accomplish the same task, where the fitness value has been calculated for each robot based on the position of the target and the position of each robot in the environment. The positions of globally and personally best robots are selected according to the fitness value in each iterative during the evolutionary process.

Consequently, the robots new positions are updated through PSO algorithm to move those robots toward the specified target. The simulation results and the real time implementation show that the robots reached their target easily and effectively. The rest of the paper is organized as follows: section 2 describes particle swarm optimization for the collective robotics search. Section 3 and section 4 present the details of CRS simulation setup in off-line and in on-line approaches respectively. Finally, the conclusion and future plan is given in Section 5.

2. Particle Swarm Optimization for CRS

Particle swarm optimization (PSO) is a swarm of individuals called particles, inspired by the social behavior of groups of birds or fish [5]. Kennedy and Eberhart suggest that in a PSO system, all particles should be initialized randomly so that they cover the entire search space and each particle represents a candidate solution to the optimization problem. Through successive generations, each particle evolves, and in the end, the global optimum result is found. In addition, in the case of a standard PSO with global neighborhood topology, each particle has its own velocity and best position that particle has ever visited (i.e. its own experience) known as personal best or PBest position. The best position among the entire swarm (i.e. the experience of all particles) is known as global best or GBest position; in this case the resulting algorithm is referred to as GBest PSO. The performance of each particle has been calculated using a fitness function that varies and depends on the problem under consideration [10].

The standard form of the PSO consists of a swarm of N particles, the position of each particle i at iteration t is written as $\vec{x_i}(t) = (x_{i1}, x_{i2}, \dots, x_{iD})$, and the velocity of each particle is represented by $\vec{v_i}(t) = (v_{i1}, v_{i2}, \dots, v_{iD})$, where D is the dimension of the optimization problem. In the iteration, each particle adjusts its position towards its personal best position (PBest) and the global best position (GBest) according to the following equations [11]:

$$\vec{v}_{ij}(t+1) = w\vec{v}_{ij}(t) + c_1\vec{r}_1\left(\overrightarrow{PBest}_{ij}(t) - \vec{x}_{ij}(t)\right) + c_2\vec{r}_2(\overrightarrow{GBest}_{ij}(t) - \vec{x}_{ij}(t)) \quad (1)$$

$$\vec{x}_{ij}(t+1) = \vec{x}_{ij}(t) + \vec{v}_{ij}(t+1) \quad (2)$$

Where c_1 and c_2 are positive constant parameters called the cognitive and social factors respectively. $r_1$and r2 are two independently uniformly distributed random vectors with the range of [0, 1] used to maintain population diversity in PSO. Subscripts j indicates the problem dimension (j=1, 2, 3... D).

PBest andGBest are the personal best and the global best position respectively of the particle i in dimension j. *w* is the inertia weight which controls the impact of the previous velocity of a particle on its current one. CRS can be considered as an optimization problem and therefore, it is simply adapted to be demonstrated using PSO. In PSO target search, a number of particles (robots) are randomly initialized into a specified environment (search space) and then the robots are navigated with one position per iteration. The coordinates of the target is known and the robots use a Euclidean distance as a fitness function to calculate the fitness value of each robot relative to the target. Then, the robots navigate through the search space while updating their PBest positions and the entire swarm GBest position based on the fitness values. Moreover, when any robot finds an optimal solution to the target, other robots migrate towards it, according to exploiting and exporting the best region of the search space.

2.1. *PSO-CRS Algorithm*

The PSO-CRS algorithm can be summarized in the following steps:

Step 1: A swarm of robots is initialized in the search environment containing a target, with random positions and velocities.

Step 2: Calculate the fitness value – Euclidean distance from each robot to the target as shown below:

$$\text{fitness} = \sqrt{(T_x - P_x)^2 + (T_y - P_y)^2} \qquad (3)$$

Where T_x and T_y are the target coordinate and P_x and P_y are the current coordinates of the robots.

Step 3: For every iteration, compare each robot's fitness value with its previous best fitness (PBest) obtained. If the current value is better than (PBest), then set PBest equal to the current value and the PBest location equal to the current location in the search environment.

Step 4: Compare PBest of robots with each other and update the swarm global best location with the greatest fitness (GBest).

Step 5: Update the velocity and position of the robot according to equation (1) and equation (2) respectively.

Step 6: Repeat Step 2-Step 5 until all robots converge to the target.

3. CRS Simulation Setup (Off-line approach)

For the purpose of simulation setup in off-line approach, the search space is set to 20 by 20 units with the centre point being the coordinate (0, 0). The maximum velocity (Vmax) is limited to 0.5 units. The initial positions for the robots are randomly generated but limited to the boundaries of the search space. For the CRS search in this work, 6 robots are used to search for the target. Figure (1) shows the initializing positions of the 6 robots and the target has been described as a circle 5 by 5 in the search space.

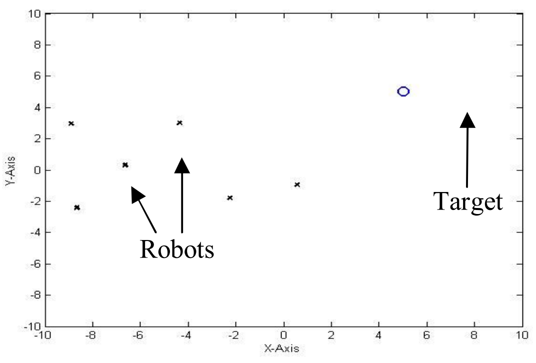


Figure 1. Robot's initializing and target locations

During the simulation, a population size of 6 is used in order to reduce the computation time significantly and also this number has been considered because only 6 mobile robots will be used in the on-line approach. This simulation has been done with a Dual Core 2.20GHz, 1GB memory, MATLAB R2009a working under Windows 7 operating system. Table 1 shows the PSO parameters used in the simulation of the CRS.

Table 1. Parameters of the PSO for CRS

Parameters	Value
N	6
c1, c2	1.49618
Vmax	Limited to 0.5
w	varying linearly from 1.2 to 0.2
Maximum iteration	100

Figure (2) and Figure (3) show the paths taken by the robots to converge at the target using standard PSO algorithm presented in section (2.1). Moreover, Table 2 and Table 3 describe the 6 robots final positions, minimum fitness value and the number of generations required to reach the target. It is clear from these figures and tables that the CRS are achieved successfully with minimum number of generations and fitness value.

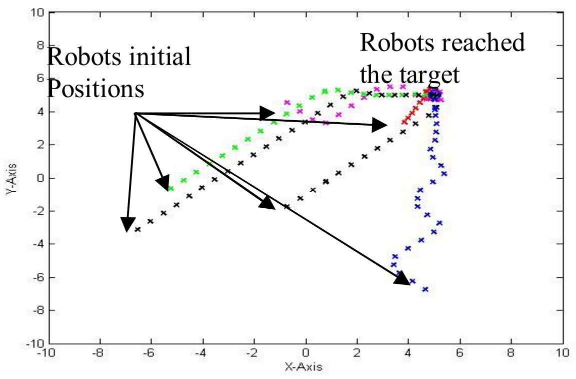


Figure 2. CRS search using PSO algorithm

Table 2. CRS statistical Results

Parameters	**Value**
Robot 1 (x, y)	(4.9379, 5.0033)
Robot 2(x, y)	(5.0036,5.0026)
Robot 3(x, y)	(5.0042 ,5.0031)
Robot 4(x, y)	(4.9988,5.0490)
Robot 5(x, y)	(4.9995 ,5.0000)
Robot 6(x, y)	(4.9975,5.0029)
Minimum Fitness	4.8506E-004
Generations	28

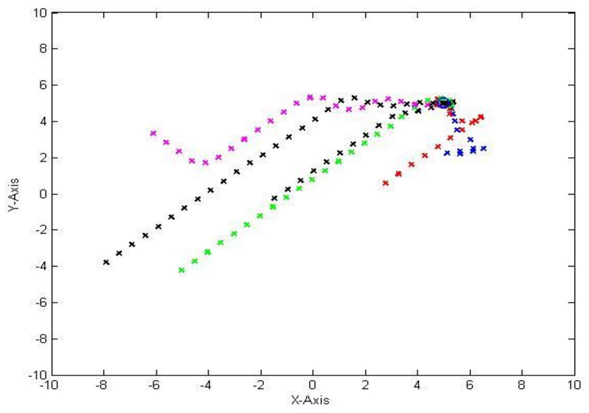


Figure 3. CRS search using PSO algorithm

Table 3. CRS statistical Results

Parameters	Value
Robot 1 (x, y)	(4.9979 ,5.0006)
Robot 2(x, y)	(4.9990,5.0002)
Robot 3(x, y)	(4.9990,4.9999)
Robot 4(x, y)	(4.9990,5.0000)
Robot 5(x, y)	(4.9990, 5.0001)
Robot 6(x, y)	(4.9990,4.9999)
Min. Fitness	9.7771E-004
Generations	38

4. Real Time CRS Implementation Setup (On-line approach)

In this section, a CRS has been implemented in on-line approach using PSO algorithm with small mobile robots when they have been used to explore and locate a specified target in the physical environment.

The requirements of the real time CRS implementation can be separated into two categories software and hardware requirements; the software includes: MATLAB, NXT BDK (Bluetooth Developer Kit) and SDK (Software Developer Kit) to run or control the Lego Mindstorms NXT robot from computer. While the hardware includes: Camera, NXT mobile robot, HDK (Hardware Developer Kit), Bluetooth adapter to communicate the NXT robot and the computer together, arena (209 × 153cm) – where the NXT robot has to be moved and from where the images have to be extracted.

Figure (4) shows the general CRS system setup, a camera has been mounted at 150cm height to cover the entire arena. This camera is connected to the main computer through a USB adapter for image acquisition purpose. While the computer controls all the NXT robots via a Bluetooth.

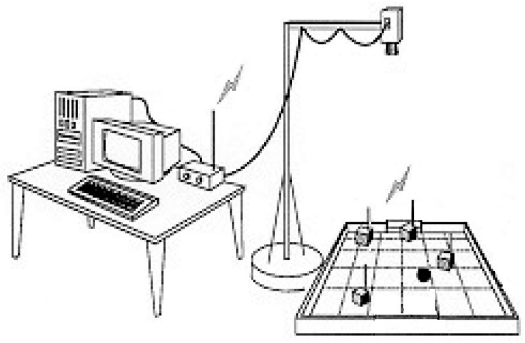


Figure 4. General CRS System Setup

4.1- The Robot Top Design

Six NXT mobile robots have been used in this implementation. Each NXT robot has been covered by a shape with black and white colors to make it different to another NXT and to be easily recognized by the image processing system without wheel and wire shadowing problem. Therefore, all robots would have the size with different number of dots. Figure (5) shows the robots top design. In this figure, each robot basically has two boxes one small and other big, which is required to find the direction of the robot, where a smaller box position has been considered as a robot position. Robots are divided to two groups: the first group represents (R1 to R3) in which there is no dot white color on the small box. While the second group with a white color in the small box represents (R4 to R6). The big box has been used to differentiate between robots within each group.

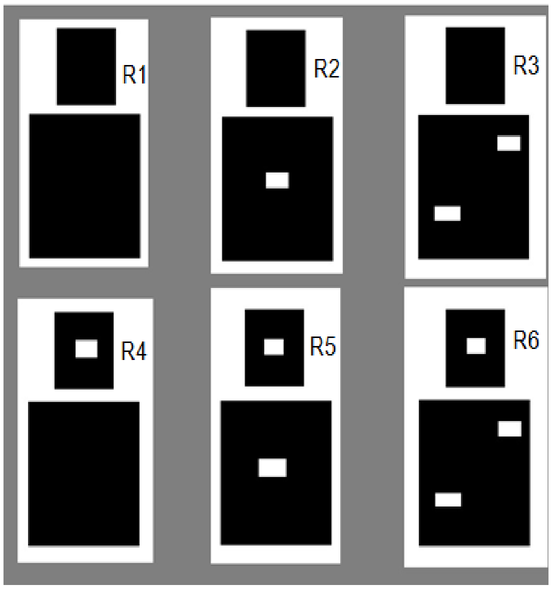


Figure 5. Robots top design

4.2- System Integration

Figure (6) summarizes the on-line CRS implementation using PSO algorithm.

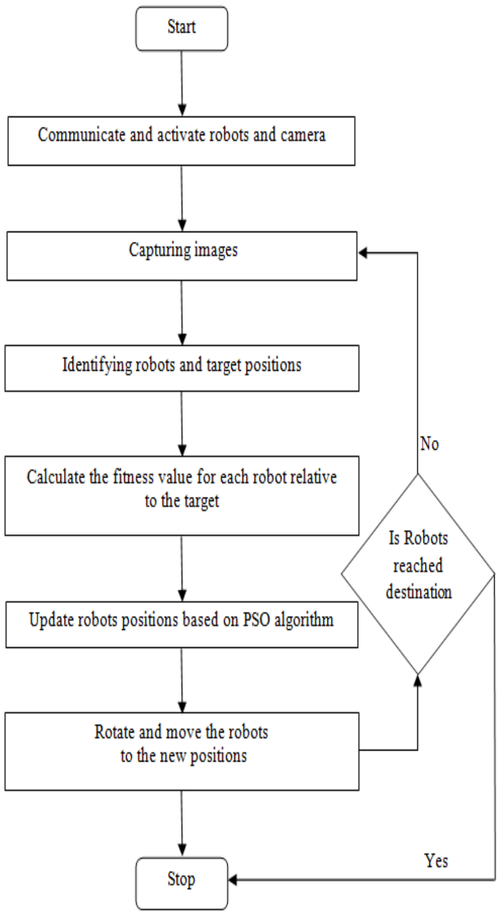

Start
Communicate and activate robots and camera
Capturing images
Identifying robots and target positions
Calculate the fitness value for each robot relative to the target
Update robots positions based on PSO algorithm
Rotate and move the robots to the new positions
Is Robots reached destination
No
Yes
Stop

Figure 6. Flowchart of on-line CRS Implementation using PSO algorithm

Figure (7) shows snapshot of the Graphical user Interface (GUI) using MATLAB.

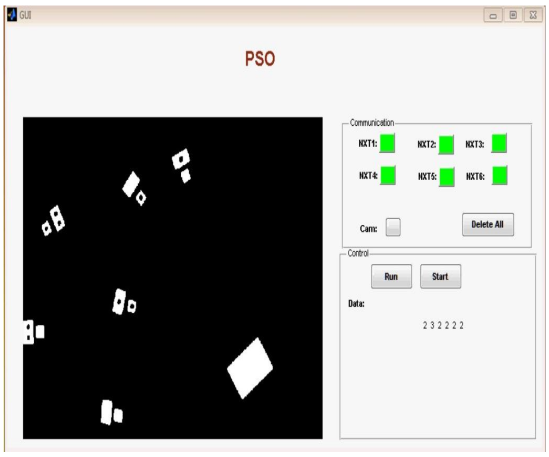


Figure 7. A snapshot from MATLAB Based GUI

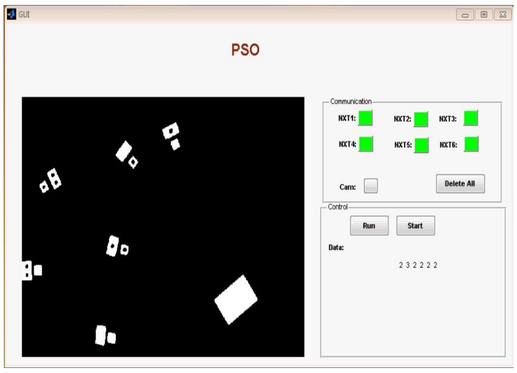
The new positions of the robots have been updated in the MATLAB system using runtime PSO algorithm. In which PSO will guide the robots through continuous interaction between them to reach to the target. Like in the off-line CRS approach, the Euclidean distance is used as a fitness function to control the robots. When the robots get closer to the target, the distance to the target location decreases and therefore, decreasing the fitness value.

The robot with least fitness value is considered as the GBest robot. Figures (8, 10) and Figures (9, 11) represent how the robots have managed to reach the target using GUI and real arena respectively. Figure (12) represents the typical evolving process of best fitness value for the on-line approach.

5- Conclusion and Future Plan

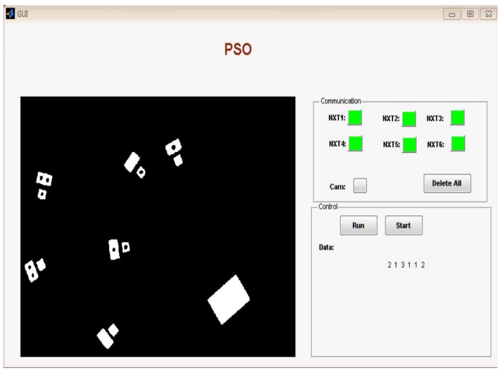
This work has presented a successful simulation and implementation of PSO-CRS algorithm to include not only off-line but also on-line approach to reach the target.

The results of this work show that applying PSO to CSR is achieved with a few numbers of iterations. Future plan will include testing the use of obstacle avoidance in the ground terrains and how the robots will manage to avoid them and reach the target.



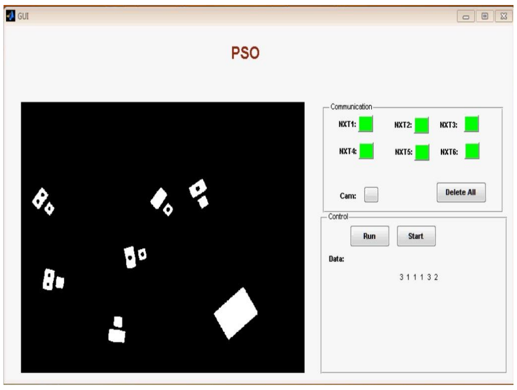
2 3 2 2 2 2

Step 1



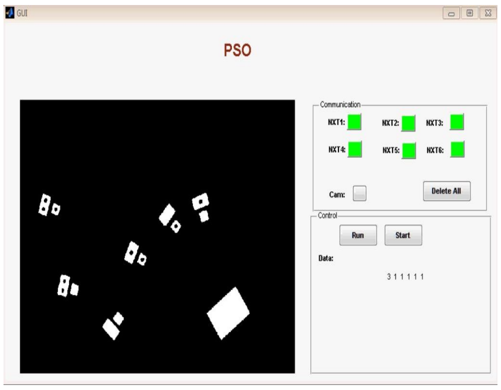
2 1 3 1 1 2

Step 2



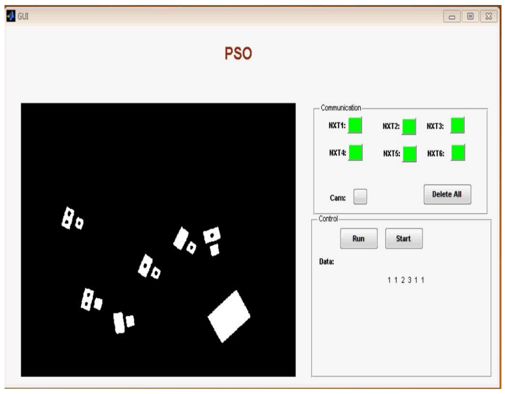
3 1 1 1 3 2

Step 3



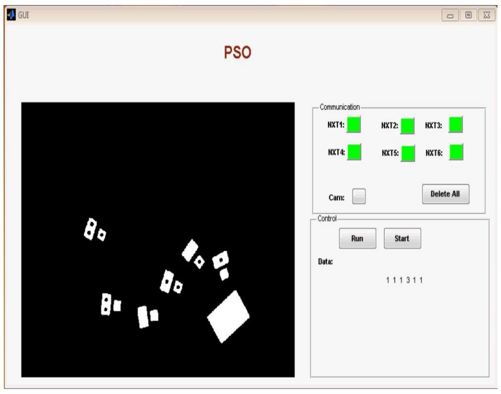
3 1 1 1 1 1

Step 4



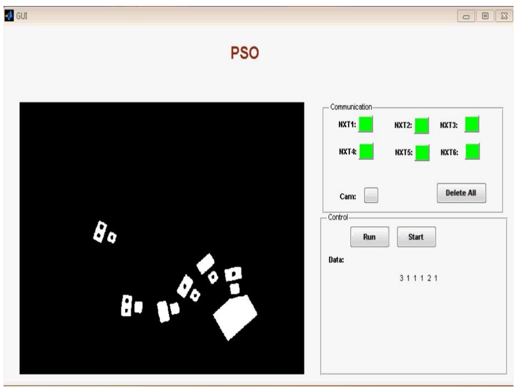
1 1 2 3 1 1

Step 5



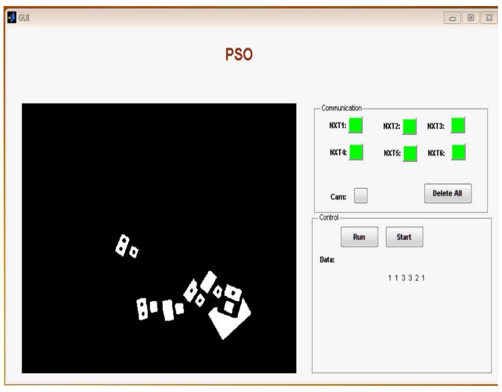
1 1 1 3 1 1

Step 6



3 1 1 1 2 1

Step 7



1 1 3 3 2 1

Step 8

Figure 8. CRS search using GUI

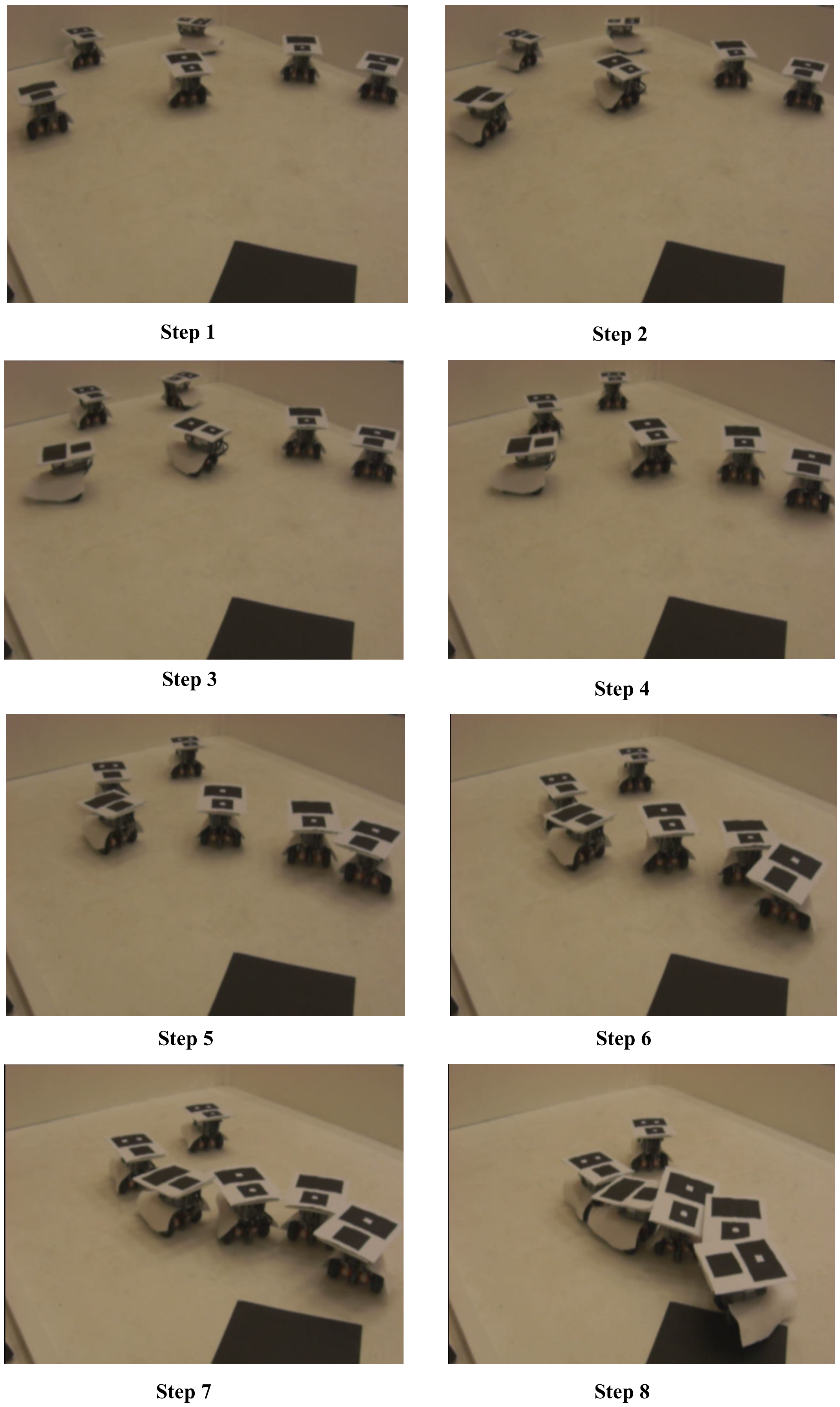
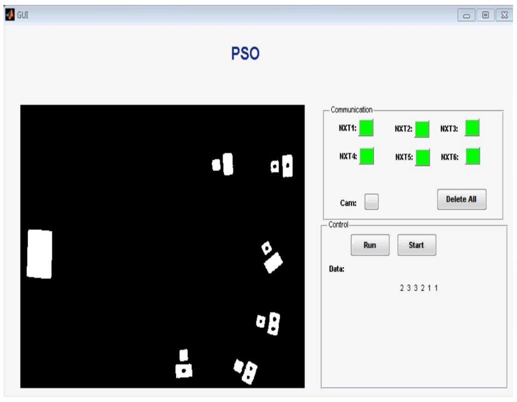
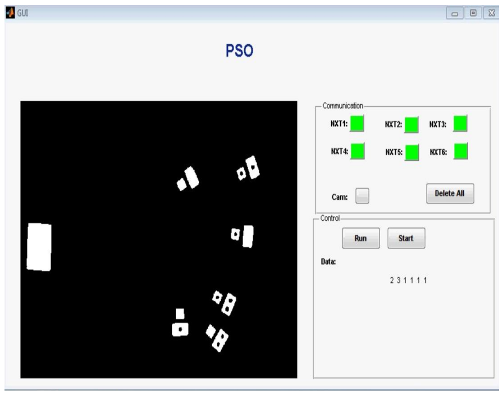


Figure 9. CRS search using real arena



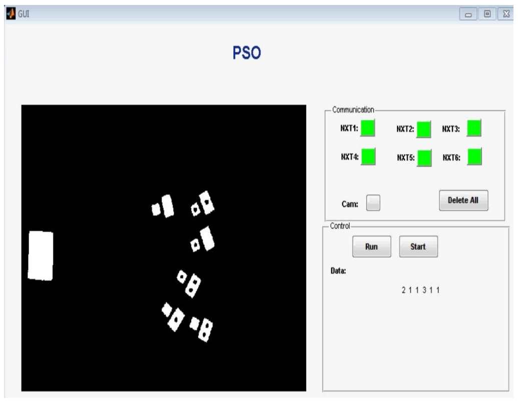
Data:
2 3 3 2 1 1

Step 1



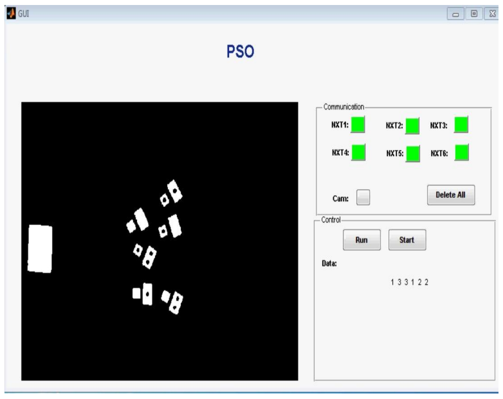
Data:
2 3 1 1 1 1

Step 2



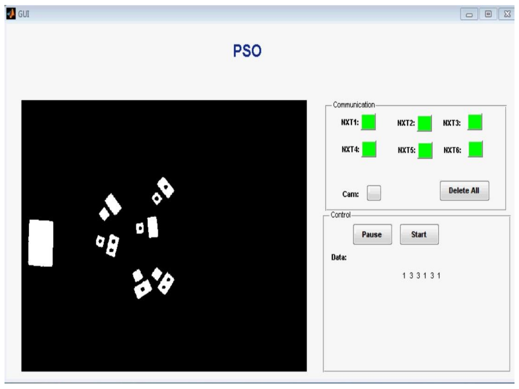
Data:
2 1 1 3 1 1

Step 3



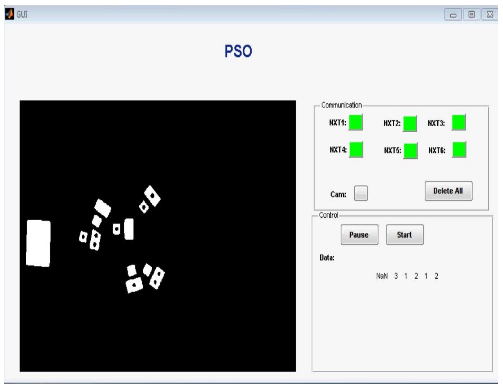
Data:
1 3 3 1 2 2

Step 4



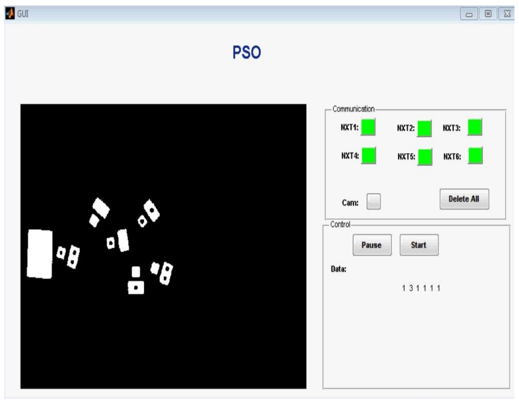
Data:
1 3 3 1 3 1

Step 5



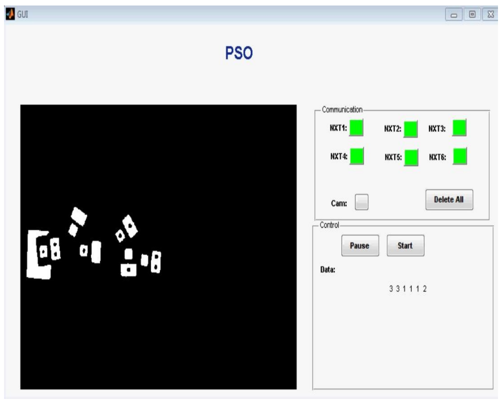
Data:
NaN 3 1 2 1 2

Step 6



Data:
1 3 1 1 1 1

Step 7



Data:
3 3 1 1 1 2

Step 8

Figure 10. CRS search using GUI

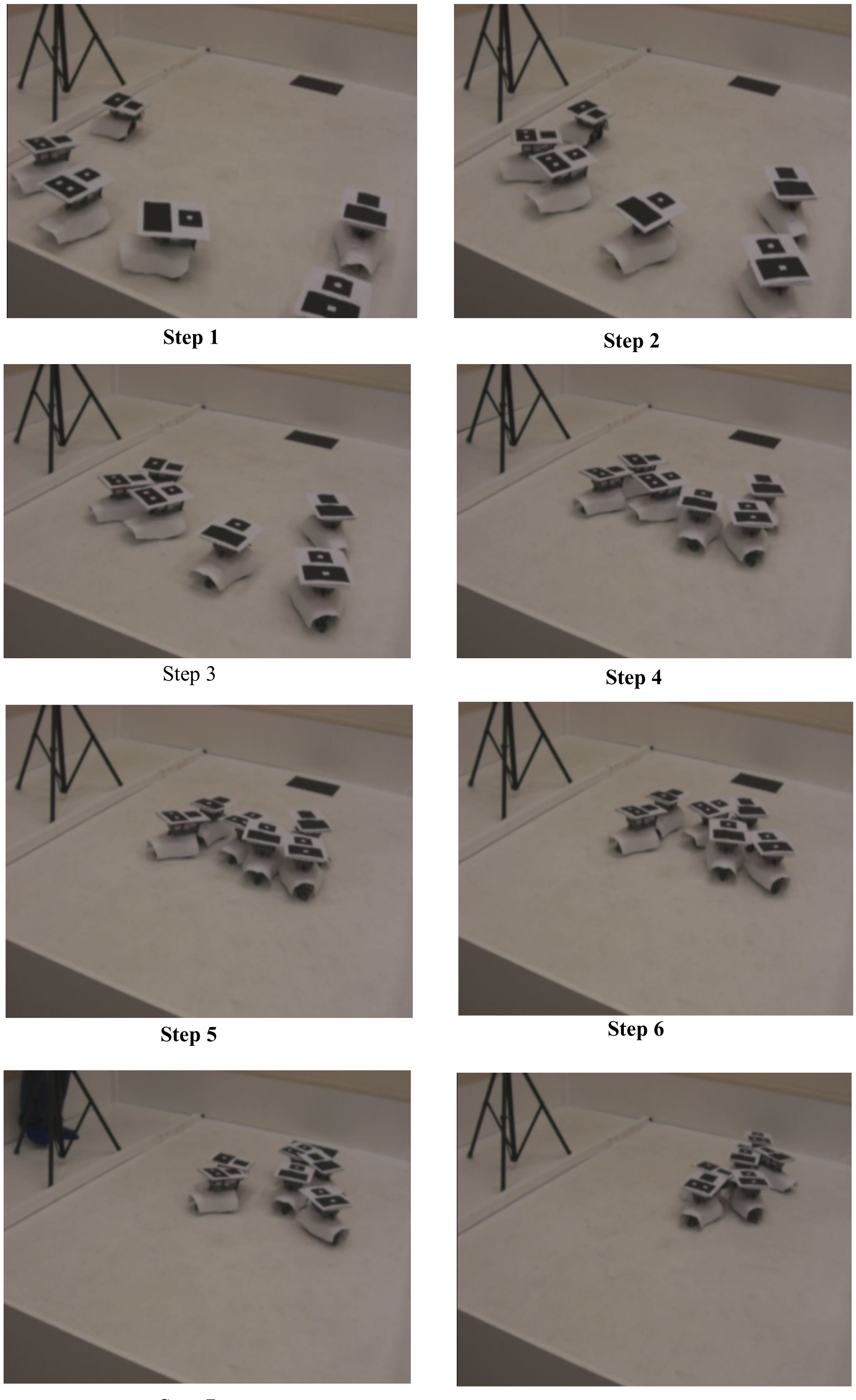


Figure 11. CRS search using real arena

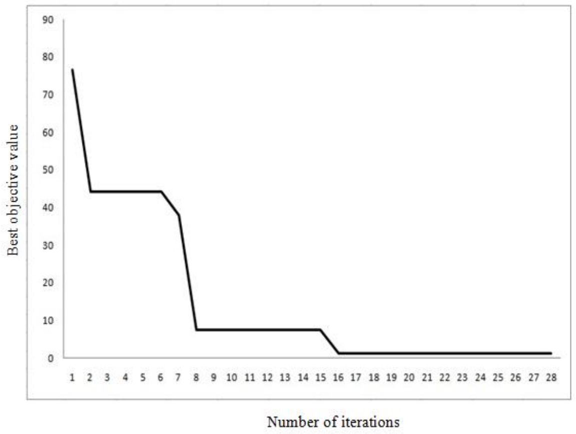


Figure 12. A typical evolving process of best objective value for PSO-CRS in on-line approach

References

[1] Liu Y. and Passino K., Swarm Intelligence: Literature Overview, Ohio State University, USA, 2000.

[2] Beni G., 'From Swarm Intelligence to Swarm Robotics [C]', in Şahin, E.andSpears, W. (Eds) Swarm Robotics: State-of-the-Art Survey. Lecture Notes in Computer Science 3342. Springer-Verlag, Germany,1-9, 2005.

[3] Beni G. and Wang J., 'Swarm Intelligence in Cellular Robotic Systems', Proceedings of the NATO Advanced Workshop on Robots and Biological Systems, Il Ciocco, Tuscany, Italy, 1989.

[4] Dorigo M., 'Optimization, Learning and Natural Algorithms', PhD Thesis, Politecnico di Milano, Italy, 1992.

[5] Kennedy J. and Eberhart R. C., 'Particle Swarm Optimization', in Proceedings of the IEEE International Conference on Neural Networks, Piscataway, NJ, USA, 1942-1948, 1995.

[6] Doctor, S. and Venayagamoorthy, G.'Unmanned Vehicle Navigation using Swarm Intelligence', in Proceedings of International Conference on Intelligent Sensing and Information Processing, Chennai, India, 249-253, 2004.

[7] Brooks, R. 'Intelligence without Reason', in Proceedings in the International Joint Conference on Artificial Intelligence, Sydney, Australia, 569-595, 1991.

[8] Mataric M. J. Nilsson, M.and Simsarian, K. T. 'Cooperative Multi-robot Box-Pushing', in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Pittsburgh, PA, 556-561, 1995.

[9] Stewart R. L. and Russell R. A., 'A Distributed Feedback Mechanism to Regulate Wall Construction by a Robotic Swarm', Adaptive Behavior 14(1):21-51, 2006.

[10] Omran, M. G. H. (2004) 'PSO Methods for Pattern Recognition and Image Processing', PhD Thesis, University Pretoria.

[11] Shi Y. and Eberhart R. C., 'A Modified Particle Swarm Optimizer', Proceedings of the IEEE International Conference on Evolutionary Computation, Piscataway, NJ, IEEE Press, 69-73, 1998.