# Study of Cooperative Control System for Multiple Mobile Robots Using Particle Swarm Optimization

September 2013

Dwi Arman Prasetya

# ◇ ◇ ◇ ◇ CONTENTS ◇ ◇ ◇ ◇

# CHAPTER 1

# Introduction

The topic of coordinated control of multiple autonomous vehicles has been explored in considerable detail in the last several years. This is due to the reason that coordinating vehicles are expected to perform tasks more effectively than a single one, especially for complicated tasks. Systems of multiple vehicles can accomplish tasks that no single vehicle can accomplish since a single one irrespective of its capabilities, is spatially limited. Multiple vehicles in coordination are spatially distributed in two-dimensional or three-dimensional spaces, and they work together following some commands given by a supervisor in a centralized control architecture, or following some rules designed in advance in a distributed manner. Some examples include automated highway systems, multiple mobile robots, unmanned aerial vehicles for surveillance, terrain mapping, space missions, formation flight and fire detection.

## 1.1 Motivation

Cooperative behavior in large groups of individuals can be found abundantly in nature. Well-known examples are schools of fish, flocks of birds, collective food-gathering in ant colonies, as well as synchronization of flashing fireflies and pacemaker cells, see Strogatz (2003)[1] for a nice introduction with many examples. A fundamental property of this cooperation is that the group behavior is not dictated by one of the individuals. On the contrary, this behavior results implicitly from the local interaction between the individuals and their neighbours. For instance, every fish in a school knows where the other fish in its neighbourhood are heading, but it does not know the average heading of all fish in the school. Nonetheless, the fish in the school stay together and move as a group in a certain direction [2, 3].

Many engineering systems also consist of large groups of cooperating dynamic systems. They are called multi-agent systems (MAS) in the literature, see Olfati-Saber et al. (2007)[4]; Ren and Beard (2008)[5] for recent overviews. Various applications are provided in Murray (2002)[6]; we mention here only three examples: Communication networks like the Internet are composed of many routers with the aim to transmit information from millions of sources to equally many users respecting the capacity of each link of the network [7]. Transport systems consist of many trains, cars, or airplanes with the common aim to bring people and goods from one point to another, see Helbing (2001) for an overview on car-following. In power networks, the power generators have to cooperate in order to provide a constant voltage and frequency irrespective of how many consumers are connected to the network [9]. These applications show two main characteristics of MAS:

1. The group consists of a large number of subsystems and their number may even be time-varying as new agents join or leave the group.

2. the interconnection topology between the agents is usually unknown and changing over time.

These properties often render a centralized control of the MAS very difficult. Therefore, engineers seek to learn from nature how to implement a decentralized cooperative control strategy that achieves global goals based on local couplings [10].

In recent years, the multiple mobile robot system have been successfully used in many fields due to their abilities to perform difficult tasks in hazardous environments, such as robot rescuing, space exploring, and so on [11, 12, 13]. Therefore, the researchers have paid more attention to many cases of searching for one or more targets in an unknown and possibly dangerous (for humans) environment is a task that can be performed by deploying multiple mobile robots[14, 15].

Many potential applications exist for the deployment of small mobile robots. Teams of small robots may potentially provide solutions to surveillance, monitoring, search, and rescue operations[16]. However, small robots have limited mobility range, reduced sensing and computation capability due to their size and power constraints[17, 18].

In the behavioral control approach, primitives behaviors are defined for the mobile robots. Drive commands are generated by aggregating a collection of weighted primitives[19]. Many algorithms in multiple robot systems such as artificial potential field [20], genetic algorithm [21], neural network system [22, 23] and flocking algorithm [24]have been proposed for solving control method for multiple robots. However those algorithms work well for the robots traversing a known environment.

The idea of using multiple mobile robots for tracking targets in unknown environment can be realized with Particle Swarm Optimization proposed by Kennedy and Eberhart in 1995 [25]. The actual implementation of an efficient algorithm like Particle Swarm Optimization (PSO) is required when robots need to avoid the randomly placed obstacles in unknown environment and reach the target point [26]. However, ordinary methods of obstacle avoidance have not proven good results on route planning. PSO is a self-adaptive population based method in which behavior of the swarm is iteratively generated from the combination of social and cognitive behaviors and is an effective technique for collective robotic search problem[27]. When PSO is used for exploration, this algorithm enables robots to travel on trajectories that lead to total swarm convergence on some target.

The PSO algorithm is used for robots to find targets at unknown environment in an area of interest. But if the environment system become complex, the searching time required will be even longer. In order to improved the original PSO algorithm based for the search performance of the multiple robot system, Lu and Han [28] proposed a probability PSO with information-sharing mechanism for cooperative control system. Due to introducing the ideas of distribution estimation algorithm and niche, each robot can be provided an opportunity to choose an appropriate position in the search space such that the search performance of the robot group can be improved.

This research treats of the cooperative control of multiple mobile robots for tracking target. The control system should have an effective motion to reach one or more different position of target [29]. Here we only have basic information about the environment like the position of each mobile robot and relative distance between mobile robots and target. Hence real time planning using coordination among the mobile robots about their surrounding environment information is necessary.

We developed a cooperative control system with PSO and obstacle avoidance algorithm in each mobile robot. The problem deals with a number of mobile robots deployed in an unknown environment reaching and tracking their target by avoiding obstacles encountered on their way. we deploy a set of mobile robots at a corner of the space from where they start moving towards the target with random position. In this process, they broadcast the information from the sensor condition about their surroundings continuously to a host PC. A circular drift function is used here to effectively avoid collisions of robots with the obstacles.

In order to confirm the validity of the proposed control system, the mobile robots are produced to implement a cooperative control system for tracking targets in unknown environments using PSO and obstacle avoidance method of the size of the group.

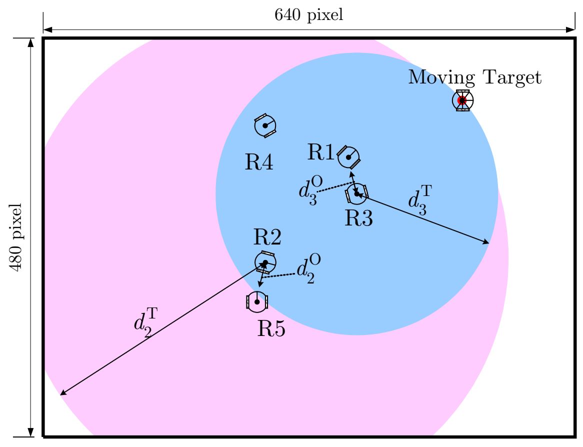
## 1.2 Problem Statement

A common trait among groups that exhibit collective motion is the capability to act without centralized control. By centralized control, we mean the actions of a single, omnipotent entity that organizes the behavior of the group. For example, the coordinated action of the tentacles of a octopus is likely generated by a central nervous system. Likewise, the scripted maneuvers of battalions within a military regimen may be orchestrated by a military commander. In both examples, the individual units tentacles and battalions, respectively certainly have some capacity for autonomous movement and interaction with one another and their environment, but without centralized control, organized behavior of multiple units may not reliably occur.

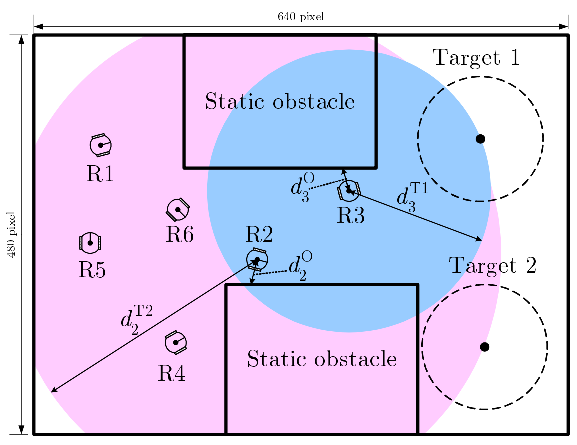
Coordination under decentralized control, which emerges so effortlessly in biological collectives, represents a major challenge for groups of autonomous vehicles. Decentralized control (also called distributed control) is a process by which each agent in a group executes a simple algorithm such that all of the agents converge to a common activity. Knowledge of a desired group activity may be available to a few agents or to none at all. Convergence to a common activity occurs literally or figuratively though the process of consensus.

The focus of this study is creating control algorithms for multiple mobile robot system. In the previous research [22], we had used neural network to organize the multiple mobile robot. However if the target position is unknown, setting the weight of the neural network will become complex. Therefore in this research we purpose a new algorithm to search and track the target by using PSO in unknown environment with obstacle.

A simple illustration for solving the problem of search and track target in unknown environment has been presented in Fig. 1.1. In initial condition, the mobile robots at



(a) Moving Target



(b) Static Target

Figure 1.1: The conception of purpose research.

a random position and random direction on the unknown environment. The mobile robots have mission to search and track the position of targets and avoid collisions of robots with the obstacles.

The environment and obstacle (i.e. Walls or other robots) position is unknown for each mobile robot. We use video cameras over the environment to get coordinate position of each mobile robot and target area. And the robots only have the information about the relative distance to the targets area and position of each mobile robot. The searching radius of each mobile robot is nearer distance from targets. For example, $d_3^{T1}$ is the nearer distance of R3 so the blue space is its searching area and $d_3^O$ is distance between R3 and static obstacle.

CHAPTER 2

# Particle Swarm Optimization

Particle swarm optimization is a stochastic population based optimization approach, first published by Kennedy and Eberhart in 1995 [25, 30]. Since its first publication, a large body of research has been done to study the performance of PSO, and to improve its performance. From these studies, much effort has been invested to obtain a better understanding of the convergence properties of PSO. These studies concentrated mostly on a better understanding of the basic PSO control parameters, namely the acceleration coefficients, inertia weight, velocity clamping, and swarm size [31, 32, 33, 34, 35, 36, 37]. From these empirical studies it can be concluded that the PSO is sensitive to control parameter choices, specifically the inertia weight, acceleration coefficients and velocity clamping. Wrong initialization of these parameters may lead to divergent or cyclic behaviour.

The empirical PSO studies do, however, provide some insight into the behaviour of particle swarms optimization (PSO), providing guidelines for parameter initialization. For example, Eberhart and Shi found empirically that an inertia weight of 0.7298 and acceleration coefficients of 1.49618 are good parameter choices, leading to convergent trajectories [38]. While such empirically obtained values do work well (in general), they should be considered with care, since the corresponding empirical studies are based on only a limited sample of problems. It should also be noted that PSO control parameters are usually problem dependent.

To gain a better, general understanding of the behaviour of particle swarms, indepth theoretical analyses of particle trajectories are necessary. A few theoretical studies of particle trajectories can be found, which concentrate on simplified PSO systems [39, 40, 41, 42, 43, 44, 45]. These studies facilitated the derivation of heuristics to select parameter values for guaranteed convergence to a stable point. This paper overviews these theoretical studies, and generalizes to more general PSO systems which includes the inertia component. The paper also provides a formal proof that particles converge to a stable point. This point is formally defined.

This chapter discusses a conceptual overview of the PSO algorithm and its parameters selection strategies, geometrical illustration and neighborhood topology, advantages and disadvantages of PSO, and mathematical explanation.

## 2.1 Basic of Particle Swarm Optimization

In the particle swarm optimization (PSO) algorithm, particles communicate with each other while learning their own experience, and gradually fly into better regions of the problem space. PSO is a stochastic global optimization method which is based on simulation of social behavior. As in GA and ES, PSO exploits a population of potential solutions to probe the search space. In contrast to the aforementioned methods in PSO no operators inspired by natural evolution are applied to extract a new generation of candidate solutions. Instead of mutation PSO relies on the exchange of information between individuals, called particles, of the population, called swarm. In effect, each particle adjusts its trajectory towards its own previous best position, and towards the best previous position attained by any member of its neighborhood. In the global variant of PSO, the whole swarm is considered as the neighborhood. Thus, global sharing of information takes place and particles profit from the discoveries and previous experience of all other companions during the search for promising regions of the landscape. To visualize the operation of the method consider the case of the singleobjective minimization case; promising regions in this case possess lower function values compared to others, visited previously.

The problem space is initialized with random solutions in which the particles search for the optimum. Each particle random searches in the problem space by updating itself with the best solution it ever found and the social information gathered from other particles. Within the defined problem space, the system has a population of particles. Each particle is randomized with a velocity and flies in the search space. The velocities and positions of the particles are constantly updated until they have all reached the target. The PSO particles are referred to as robots and the local version of the PSO algorithm is considered in the context of this application. Each mobile robot as particles communicate with each other while learning their own experience in the population begins with a randomized position $\vec{P}_i(t)$ and randomized velocity $\vec{V}_i(t)$ in the real environment search space. PSO has been used by researchers all over the world from various fields of research for different types of optimization.

Initially, let us define the notation adopted in this paper. The problem in unknown environment is initialized with random solutions in which the robots search for the optimum. Each robot random searches in the problem space by updating itself with the best solution it ever found and the social information gathered from other robots. As in general PSO, the mobile robots navigate through the environment with dynamic velocity while storing their personal previous best position ($pBest$) and the best position of the entire swarm relative to the target, know as the global best position ($gBest$). As one mobile robot finds an optimal solution, other robots migrate towards it, in effect exploiting and exploring the best sections of the search space. The velocities and positions of the robots are constantly updated until all robots reached the target position.

Velocity update equations based on the PSO are given by :

$$\begin{aligned} \vec{V}_i(t+H) &= \vec{V}_i(t) + c_1 rand(*)(pBest_i - \vec{P}_i(t)) \\ &+ c_2 rand(*)(gBest - \vec{P}_i(t)) \end{aligned} \tag{2.1}$$

where,

$H$: sampling time in the simulation with 40 ms.

$c_1$ is the cognitive acceleration coefficient so named for its term's use of the personal best, which can be thought of as a cognitive process whereby a particle remembers the best location it has encountered and tends to return to that state.

$c_2$ is the social acceleration coefficient so named for its term's use of the global best which attracts all particles simulating social communication. $c_1$ and $c_2$ are two positive constant and the balance factors between the effect of self-knowledge and social knowledge in moving the particle towards the target.

$rand(*)$: a random number between 0 and 1, and different in each iteration.

$pBest_i$: the personal best position of a particle $i$ (mobile robot).

$gBest_i$ : the best position within the swarm.

$\vec{V}_i(t+H)$: the velocity of mobile robots.

The farther a particle is from its personal best, the larger $(pBest_i - \vec{P}_i(t))$ is and the stronger the acceleration toward that point is expected to be. Notice that if a particular dimension of the current position is greater than the same dimension of the personal best, the acceleration on that dimension is negative, which means that the particle is pulled back toward that location on that dimension. Of course, this implies that when the personal best lies ahead of the current position, the particle will accelerate in the positive direction toward the personal best so that each particle is always pulled toward its personal best on each dimension. Similarly, the farther a particle is from its global best, the larger $(gBest_i - \vec{P}_i(t))$ is and the stronger the acceleration is toward that point.

The social and cognitive acceleration coefficients, $c_1$ and $c_2$ , determine the respective strengths of those pulls and relative importances of each best.

When each dimension of the social and cognitive terms is multiplied by a random number, the acceleration is not necessarily directed straight toward the best. Were the same random number used on all dimension, each pull will be straight toward its best.

Either way, particles are accelerated in two different directions at once so that they do not actually accelerate straight toward either best.

At each iteration, the previous velocity is reduced by the inertia weight and altered by both accelerations in order to produce the velocity of the next iteration. Treating each iteration as a unit time step, a position update equation can be stated as :

$$\vec{P}_i(t+H) = \vec{P}_i(t) + \vec{V}_i(t+H) \tag{2.2}$$

where,

$H$: sampling time in the simulation with 40 ms.

$\vec{P}_i(t+H)$: the new position of mobile robots $i$ for the next iteration.

$\vec{P}_i(t)$ :the current position of a mobile robot $i$.

$\vec{V}_i(t+H)$: the velocity of mobile robots.

Eq. (2.2) provides the new position of each particle, adding its new velocity, to its current position. The performance of each particle is measured according to a fitness function, which is problemdependent. In optimization problems, the fitness function is usually identical with the objective function under consideration.

## 2.2 The pseudo code of PSO

The pseudo code of the procedure PSO algorithm can be written as follows:

**Algorithm 2.2.1:** PSO($pBest_i, gBest$)

```
for each Particle
  do Initialize particle
repeat
  for each Particle
    do Calculate particle
  if The fitness value is better than its personal best
    then Set current value as the new $pBest_i$
  the particle with the best fitness value of all as $gBest$
  for each Particle
    do Calculate particle velocity according equation 2.1
       Update particle position according equation 2.2
until maximum iterations or minimum error criteria is not attained
```

In PSO, there are two types of information available for the particles so that they can make the best decision regarding where to move next.

## 2.3 PSO with inertia weight

The role of the inertia weight ($\omega$) is considered important for the PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current velocity.

### 2.3.1 Static inertia weight

There was a weakness inherent in velocity update eq. (2.1) that was fixed by the introduction of an inertia weight. For the following derivation, let $t = 0$ be the iteration at which particles have their positions and, optionally, their velocities randomly initialized. Then for any particle $i$ , the velocity at iteration 1 is :

$$\begin{aligned}\vec{V}_i(1) &= \vec{V}_i(0) + c_1 rand(*)(pBest_i(0) - \vec{P}_i(0)) \\ &+ c_2 rand(*)(gBest(0) - \vec{P}_i(0)) \qquad (2.3)\end{aligned}$$

Since a particle has only one position, $\vec{x}_i(0)$, from which to choose in order to determine its personal best, $\vec{P}_i(0)$, of necessity $\vec{P}_i(0) = \vec{x}_i(0)$ and the middle term of eq. (2.1) is zero, so the particle's velocity at iteration 1 can more succinctly be expressed as :

$$\begin{aligned}\vec{V}_i(1) &= \vec{V}_i(0) + c_1 rand(*)(0) \\ &+ c_2 rand(*)(gBest(0) - \vec{x}_i(0)) \qquad (2.4)\end{aligned}$$

Using eq.(2.1) again, the velocity of particle$i$ at iteration 2 is

$$\begin{aligned}\vec{V}_i(2) &= \vec{V}_i(1) + c_1 rand(*)(pBest_i(1) - \vec{P}_i(1)) \\ &+ c_2 rand(*)(gBest - \vec{P}_i(1)) \qquad (2.5)\end{aligned}$$

Substituting the value found in eq.(2.4) for $\vec{V}_i(1)$ , the velocity at the second iteration following initialization becomes

$$\begin{aligned}\vec{V}_i(2) &= \vec{V}_i(0) + c_1 rand(*)(pBest_i(1) - \vec{P}_i(1)) \\ &+ c_2 rand(*)(gBest - \vec{P}_i(1)) \qquad (2.6)\end{aligned}$$

Because the personal bests and global best can only improve over time, $\vec{V}_i(t + H)$ should rely more heavily upon recent bests than upon early values. The parameter $\omega$ regulates the trade off between the global (wide ranging) and the local (nearby) exploration abilities of the swarm. A large inertia weight facilitates exploration (searching new areas), while a small one tends to facilitate exploitation, i.e. fine tuning the current search area. A proper value for the inertia weight $\omega$ provides balance between the global and local exploration ability of the swarm, and, thus results in better solutions. Experimental results imply that it is preferable to initially set the inertia to a large value, to promote global exploration of the search space, and gradually decrease it to obtain refined solutions. The initial population can be generated either randomly.

Velocity update equations based on the PSO with inertia weight are given by :

$$\begin{aligned}\vec{V}_i(t+H) &= \omega\vec{V}_i(t) + c_1 rand(*)(pBest_i - \vec{P}_i(t)) \\ &+ c_2 rand(*)(gBest - \vec{P}_i(t))\end{aligned} \tag{2.7}$$

Additionally, a particle's initial velocity, which is not derived from any information, but randomly initialized to lie between the upper and lower velocity clamping values, becomes of less effect over time.

### 2.3.2 Time-Varying inertia weight

Decreasing the inertia weight over time would still allow the swarm to gradually forget early information of relatively low quality, as in the static case, due to the iterative multiplication of all past information by a fraction of one as in equation (2.7). For the decreasing weight, however, information is forgotten more quickly than were the initial value held constant. This time-decreasing weighting of information may provide more balance between early and recent information since early information is forgotten at a slower rate than later information due to the use of relatively large weights early in the simulation. In other words, all memory is adversely affected, but short-term memory is mostly affected. This potentially more balanced weighting of early information with late information might help the standard algorithm postpone premature convergence to candidate solutions of later iterations when appropriate initial and final values are used.

The decreasing inertia weight also allows early weights to be larger than were a static weight used throughout the search. This corresponds to larger velocities early in the search than would otherwise be seen, which may help postpone premature convergence by facilitating exploration early in the search. The rate of decrease from initial weight to final weight depends on the expected length of the simulation since the step size is a fraction of the total number of iterations expected; hence, the amount of time spent in the relatively explorative phase, as determined by the amount of time for which the decreasing weight is larger than the value that would have been used for a static weight, also depends on the expected length of the simulation.

Increasing the inertia weight, on the other hand, would cause past information to be forgotten more rapidly than recent information due to the weighting distribution, thus tremendously increasing the importance of the higher quality information of later iterations. For the right range of values, this could conceptually lead to quicker initial convergence due to less diversity being maintained; however, this could adversely affect solution quality on difficult functions by upsetting the balance between exploration and exploitation.

Quick convergence is desirable when successfully converging to a global minimizer, but it is undesirable when the search is so hasty as to converge prematurely to a local minimizer. There is a delicate balance to achieve in order to search efficiently yet thoroughly. The time-varying weight attempts to improve that balance as inferred from equation (2.7), which shows that at any iteration a particle's velocity vector is the result of weighted attractions toward past information, which frames a time-varying inertia weight as affecting the balance between the rates of short-term and long-term forgetfulness.

The first study to vary the inertia weight decreased it with the idea that this would help particles converge upon and refine a solution by reducing velocities over time. This appeared to work better over the thirty trials conducted [33]; but with only one benchmark tested, it is conceivable that this might have been a characteristic of that particular benchmark, which would be consistent with the findings of Meissner et al [46], who used particle swarm to optimize its own parameters with very different parameters being proposed per benchmark including an increasing inertia weight on some benchmarks and a decreasing weight on others. Since that experiment used Gbest PSO, which tends to stagnate before reaching a global minimizer, the parameter combinations recommended are likely not ideal, though they may be approximations of quality local minimizers.

Whereas [46] found an increasing weight to outperform on some benchmarks, [47] suggested that an increasing inertia weight outperformed on all benchmarks tested; however, a different formulation of PSO was used so that the quicker convergence claimed could not be attributed to the increasing weight alone. In an attempt to reproduce the results of [47] using standard Gbest PSO, increasing the inertia weight from 0.4 to 0.9 with the same swarm size of 40 particles, acceleration constants 1.49618, and 1,000 iterations as used in the paper resulted in worse performance on all nine benchmarks relative to decreasing the weight from 0.9 to 0.4. Therefore, decreasing the weight appears better than increasing it, at least for the range between 0.9 and 0.4. When the static weight was compared to decreasing, however, only the Ackley and Rastrigin benchmarks saw much improvement from decreasing the weight; and performance on Rosenbrock suffered from the decrease, so that decreasing the inertia weight is not always best.

It is noteworthy that Naka and Fukuyama showed a decrease from 0.9 to 0.4 to considerably outperform decreases from 2.0 to 0.9 and from 2.0 to 0.4 on their particular state estimation problem [48], but they did not generate any comparison data using the static inertia weight.

## 2.4 Velocity Clamping

Eberhart and Kennedy introduced velocity clamping, which helps particles take reasonably sized steps in order to comb through the search space rather than bouncing about excessively [25]. Clerc had hoped to alleviate the need for velocity clamping with his constriction models [41]. Eberhart, however, showed clamping to improve performance even when parameters are selected according to a simplified constriction model (2.7) [38].

Clerc then compared equation (2.13) with velocity clamping to his other constriction models without velocity clamping and concurred that velocity clamping does offer considerable improvement even when parameters are selected according to (2.7), so that the constriction models have not eliminated the benefit of velocity clamping [41]. Consequently, velocity clamping has become a standard feature of PSO. Velocity clamping is done by first calculating the range of the search space on each dimension, which is done by subtracting the lower bound from the upper bound.

As noted by Engelbrecht [49], clamping a particle's velocity changes not only the step size, but usually also the particle's direction since changing any component of a vector changes that vector's direction unless each component should happen to be reduced by the same percentage. This should not be thought of as a problem, however, since each dimension is to be optimized independently, and the particle still moves toward the global best on each dimension, though at a less intense speed.

Since the maximum iterative movement toward global best on any dimension is clamped, particles may be thought of as combing the search space a bit more thoroughly than were their velocities unclamped. Though the same velocity clamping percentage of fifty percent is used in most papers for sake of comparison, the value does not appear to have been optimized yet. Liu et al suggested a value of fifteen percent [50].

Clamping velocities to fifteen percent provided noticeably better performance in median and mean values on multi-modal functions of high dimensions, where cautious step sizes in light of new information proved most beneficial; Griewangk was the exception, since one poorly performing trial significantly affected the mean function value. Smaller step sizes seem to have helped avoid premature convergence to sub-optimal, local minimizers. It appears that the standard velocity clamping value of fifty percent widely used in the literature can be improved upon, and fifteen percent seems to work well in agreement with Liu's observation based on primarily different benchmarks of low dimensions [50].

## 2.5 Flowchart of PSO

The velocity value is calculated according to how far an individual's data is from the target. The further it is, the larger the velocity value. In the birds example, the individuals furthest from the food would make an effort to keep up with the others by flying faster toward the gBest bird.

If the data is a pattern or sequence, the velocity would describe how different the pattern is from the target, and thus, how much it needs to be changed to match the target. The basic flowchart of PSO algorithm is shown in Fig. 2.1.

This too makes sense because its main benefit is in early iterations where it provides momentum by which to propel the best particle, but after some time it effectively becomes noise diluting actual information. It can be concluded that PSO, as algorithm with low complexity and high feasibility, can finish the task of searching for the optimum solution without too much parameter adjustment.

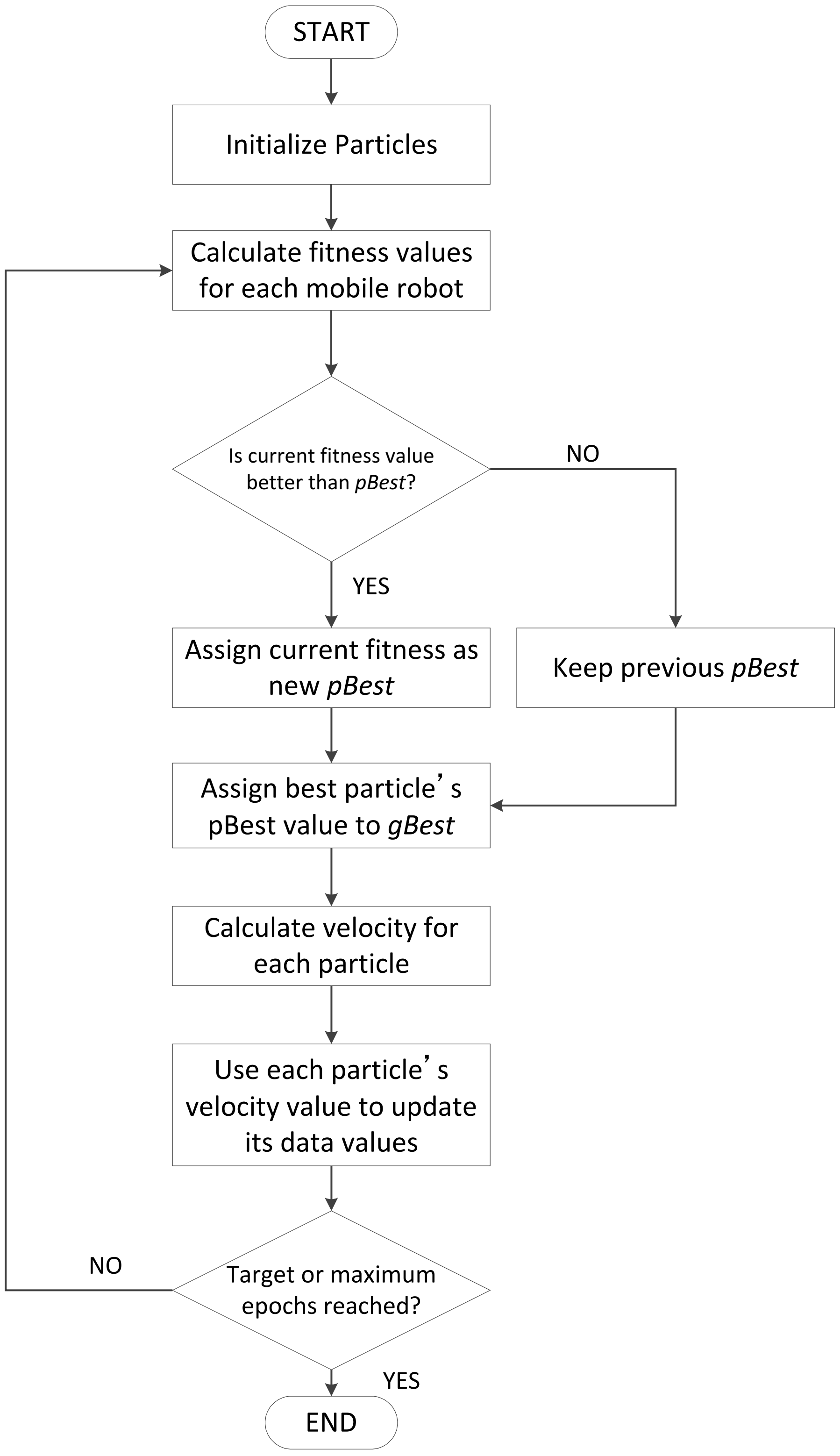

START
Initialize Particles
Calculate fitness values for each mobile robot
Is current fitness value better than *pBest*?
NO
YES
Assign current fitness as new *pBest*
Keep previous *pBest*
Assign best particle's pBest value to *gBest*
Calculate velocity for each particle
Use each particle's velocity value to update its data values
Target or maximum epochs reached?
NO
YES
END

Figure 2.1: Flowchart of the particle swarm optimization algorithm.

# CHAPTER 3

# Cooperative Control System

One of the current challenges in the development of robot control systems is making them capable of intelligent and suitable responses to changing environments. Learning and adaptation methods, as well as decision-making techniques, help to achieve these objectives. Nevertheless, it is technologically difficult and potentially dangerous to build complex systems that are only centrally controlled, since the system will fail if the control system does not work. With decentralized control, it is possible for the system to continue working even when a part of it fails. Although centralized control allows multiple goals and constraints to be coordinated in a coherent manner, decentralization provides flexibility and robustness.

This section presents the cooperative control method, based on particle swarm optimization, used to combine multiple controllers in the behavior of the mobile robots. Instead of developing only one very elaborate controller, we have designed several simple controllers aimed at treating different aspects of the control separately and unifying their actions to obtain a final complex behavior.

## 3.1 Previous work

To solve the navigation problem for the robot, researchers have proposed various methods. In conventional navigation methods such as cell decomposition[51] and road map[52], due to the high volume of calculations, we are not able to solve problems in complex environments.

Artificial potential field method [53], because of simplification frequently is used for local navigation. But due to stop at a local minima, This method will fail. In recent years a series of intelligent ideas, such as genetic algorithms and particle swarm optimization because of the robust and ability to the Simultaneous calculations to solve the navigation problems are used. Ghorbani and colleagues[54], use of the genetic algorithm for solving the problem of mobile robot navigation. Sugiwara and colleagues [55], used ants colony algorithm to solve the problem of navigation in a dynamic virtual environment. Qu and colleagues [56] used neural networks for navigation and obstacles avoid in dynamic environments. PSO, by Kennedy in 1995, based on observation of the collective behaviour of certain species of animals such as birds and fish have been proposed[25]. Due to simplicity, this method is used in robot navigation. Doctor and colleagues[18], using the PSO method for navigation an unmanned vehicle that can converge well. Chen and colleagues[57], suggests a soft and efficient navigation method for mobile robot using the Stochastic PSO. Qin and colleagues[38] used the Chaotic PSO with Mutation operator for navigation and moving the robot meets the immediate needs.

Hao and colleagues, [58] proposed a method of obstacles avoiding using the PSO and polar coordinate system in a dynamic environment. Wang and colleagues [60] used a PSO for navigation of soccer robot and Karimi[61] , using dynamic hybrid PSO algorithm to solve motion planning problem.

## 3.2 Cooperation and Consensus

As mentioned above, consensus and cooperation in networked multi-agent systems has recently attracted much attention in the research community. For a great introduction into the field and examples of its many, diverse applications see for instance the surveys[62], Olfati-Saber et al. (2007)[4] and Murray (2007)[63], as well as the collection of references at Reynolds (2001)[64].

### 3.2.1 History

Consensus and agreement problems were studied systematically as early as the 1960 in the context of management science and statistics [65, 66, 67, 68]. Later, those ideas were picked up in different contexts,such as fusion of sensor data [69, 70, 71, 72] or see the proceedings of the IEEE conferences on Multisensor Fusion and Integration for Intelligent Systems), medicine [73], decentralised estimation [74, 75, 76, 4], clock synchronisation or simulation of flocking behaviour [77, 78] just to name a few.

### 3.2.2 Networked dynamic systems

Particularly in the last decade the general problem of consensus finding in networked dynamic systems has been focused on intensely. It typically comes in many flavours depending on the application. These variations include whether the topology of the graph representing the inter-agent communications remains fixed or changes over time; it is undirected or directed; the agents can manipulate the state on which to reach consensus instantly or only with certain dynamics; if each node's state is scalar or multidimensional; whether there are delays in the information exchange; or if all nodes update their states in a synchronous fashion or on their own pace. While the initial work by [77, 78, 79] on consensus and coordination was based on bi-directional information exchange between neighbouring nodes (leading to undirected communication graphs) with rigorous convergence proofs given in[75], this has been extended to include directed communication graphs for instance in [4, 62]. Another generalisation allowed asynchronous consensus protocols so that not all nodes had to perform state updates at the same instant,[4]. Closely related was the work that also considered changing graph topologies. Further generalisations of the problem allowed the inclusion of agent dynamics (typically linear, second order systems) in the consensus problem[4], which play an important role in networks of mobile agents that move with finite dynamics. In some situations the consensus variable may not be directly altered by the nodes, but only implicitly.

However, most of these papers only focus on so-called unconstrained consensus applications. When the consensus, that the system is to reach, should fulfil external conditions (such as a common heading of a flock of agents, but in a particular direction), three approaches are usually taken, see [4].

### 3.2.3 Leader-following

The first concept presents a common technique used typically to make formations of autonomous mobile agents follow desired trajectories. The idea is that all agents in the are programmed to follow a designated "leader" node. However, the problem with these architectures is usually that they not only depend heavily on the leader, but it appears that little discussion of the case where the leader adjusts its state based on feedback of the totality of the states of the network has taken place, and most of the systems dealt with in that context are linear.

### 3.2.4 Behaviour based

In the behavioural approach, each agent's behaviour is based on a combination (e.g. weighted sum) of a number of desired behaviours, such as goal seeking, formation keeping, obstacle and collision avoidance, etc. A typical application of these techniques are rendezvous problems with obstacle and collision avoidance, where the agents are to meet in a certain place, but avoid running into obstacles or crashing into each other during the approach.

## 3.3 Obstacle avoidance algorithm

Collision detection concerns the problems of determining if, when, and where two objects come into contact. Gathering information about when and where (in addition to the Boolean collision detection result) is sometimes denoted collision determination. The terms intersection detection and interference detection are sometimes used synonymously with collision detection. Collision detection is fundamental to many varied applications, including computer games, physically based simulations (such as computer animation), robotics, virtual prototyping, and engineering simulations to name a few. Due to this wide application base, numerous techniques have been developed to predict possible collision situations. Velocity space based techniques, like The Dynamic Window approach (DW) and Velocity Obstacles (VO) have been shown to realize collision avoidance taking into account the future behavior of moving objects.

One way to realize Collision avoidance maneuvers is trough the implementation of path planing methods with obstacle compliant geometry. Substantial research on these kind of methods and algorithms for single robots working in environments with static obstacles can be found in the literature. Examples include the geometrical methods like the road map, cell decomposition, or methods based on potential field theory just to name a few. The roadmap and cell decomposition methods rely on rules that are derived using the geometry of the obstacle field. Many problems on motion planning for multiple robots [11] have been solved using the geometrical methods. Different control theories have also been used for path planning for groups of mobile robots.

A centralized path generation for a group of vehicles is realized using a polynomial-based approach, taking into account spatial and temporal constraints. Ensuring inter-vehicle collision avoidance, or even other criteria like simultaneous times of arrival. As mentioned, another approach that has been extensively used for obstacle avoidance for single mobile vehicles, multiple mobile vehicles, and dynamic obstacles is the potential field approach.

In this research, we focus on the mobile robots as particles move through on the workspace, gaining one new position for every iteration, a conditional statement checks to see if the sensor condition of each mobile robot active or not. If one of the sensors is active, the obstacle avoidance section of the algorithm is initiated. The detection range of the sensor is fixed 50mm.

The obstacle avoidance algorithm can be summarized in the following steps. Each mobile robot from the beginning until reaching the target position, always check the condition of the sensor with the scanning method from left to the right. Second if the condition of the sensor is true, the mobile robot will execute the interrupt program for obstacle avoidance. In interrupt condition mobile robot makes moving action according the condition of sensor are listed in the Table 3.1.

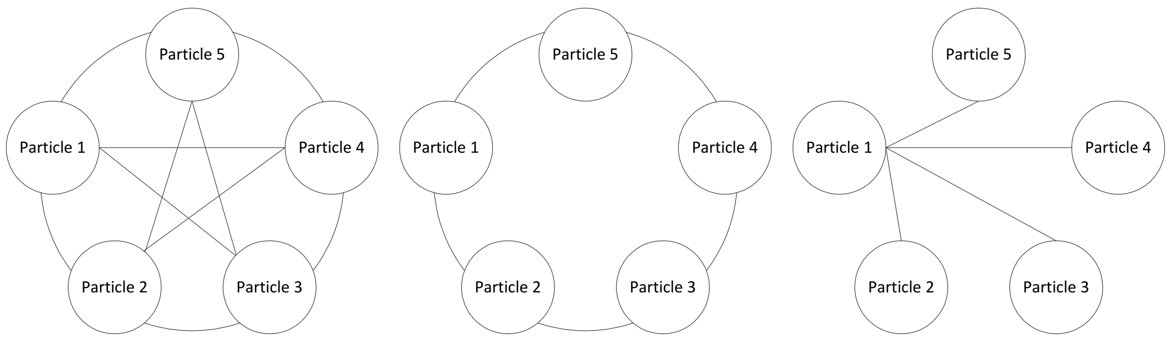
Table 3.1: Obstacle avoidance moving action of mobile robot (P-2)

| Moving Action | Left Sensor | Front Sensor | Right Sensor |
|---|---|---|---|
| Right pivot | True | False | False |
| Slow backward | False | True | False |
| Left pivot | False | False | True |
| Right pivot | True | True | False |
| Slow forward | True | False | True |
| Left pivot | False | True | True |
| Slow backward | True | True | True |

In the experiment each mobile robot after the completion of obstacle avoidance moving action will send the condition of sensor data to the host PC. Then the sensor data will be used to determine the next position that produced by PSO algorithm. The sensor data used to explain that there are obstacles around the mobile robots, if the next position around the obstacle area, PSO algorithm will find another position to avoid the obstacles. In the simulation, the distance sensor simulated with the fix range and drawing by red line in the three position around the mobile robot.

## 3.4 Moving action topologies

In the PSO context, two terms, local versus global are often used. Local refers to an individual neighbourhood while the global refers to the entire swarm as one big neighbourhood. Different neighbourhood structures may effect the performance of the swarm. They determine how information propagate among particles and thus may effect the convergence of partcles, i.e. when and how particles may come together,arrive at some stable state and stop improving the solution.



(a) A star (b) A ring (c) A wheel

Figure 3.1: Simple neighbourhood topologies with 5 particle

Figures 3.1(a)-(c) are the most commonly used neighbourhood structures, i.e. Stars, rings and wheels. In star topology as shown in Fig. 3.1(a), all particles are influenced by one global best location so far in every iteration and move towards the location, so they tend to converge quickly to the global best. In a ring topology as shown in Fig. 3.1(b), the neighbourhood to another and eventually pull all the particles together. By gradually spreading information, the swarm converges slower in a ring than in a star topology. In a wheel topology as shown in Fig. 3.1(c), there existx one and only one central particle, which serves as a buffer. The central particle collects and compares the position of all particles, finds the best one and moves itself towards the best position. Becouse of this buffering effect, a wheel topology may preserve diversity for a bit longer and prevent the swarm from converging too fast on local optima.

## 3.5 Mobile robot model

There are some common mathematical models for mobile robots. The simplest possible model is the kinematic or first order model, which describes a point-like robot moving around in the plane. In this model, the velocity is the control input. The main drawback of the model is that it allows instant velocity change, which is a problem if the vehicle is heavy, relative to its motor power. To fix this problem, we will consider a dynamic model with inertia or second order model.

A more common mobile robot configuration is called the two-wheeled mobile robot, according to the classification made by E. Ferrante. [27], which has two independently actuated fixed wheels. It is clear that their degree of maneuverability is two and their number of steering wheel is zero. The simplest 75 possible model for two-wheeled mobile robot, the kinematic equations of the two-wheeled mobile robot are:

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{\cos(\theta)}{2} & \frac{\cos(\theta)}{2} \\ \frac{\sin(\theta)}{2} & \frac{\sin(\theta)}{2} \\ \frac{1}{2l} & \frac{1}{2l} \end{bmatrix} \begin{bmatrix} V_L \\ V_R \end{bmatrix} \tag{3.1}$$

where $\frac{dx}{dt}$ and $\frac{dy}{dt}$ are velocities of the center of mobile robot, $\theta$ is the angle that represents the orientation of the vehicle, $v_i^L$ and $v_i^R$ are velocities of right and left wheels and $2L$ is the mobile robot base length. Each particle remembers the position that achieves its highest performance also a member of some neighborhood of particles, and remembers which particle achieved the best overall position in that neighborhood.

Figure 3.2 shows the description of a two-wheeled mobile robot model. Where, $P_i$ $(x_i, y_i)$ is the coordinates of the $i$-th mobile robot position, $\vec{V_i}$ is a velocity of the mobile robot.

More details in dynamics of wheeled mobile robot can be found in [10], which include not only the motion of mobile robot but also the motion of the driving wheels.

The evaluation function of distance between each mobile robot and the target, means the distance $d_i^{ref}$ between the current position $P_i$ $(x_i, y_i)$ and the desired position $P_i^{ref}$ $(x_i^{ref}, y_i^{ref})$ of the robot at the next sampling, and is defined by.

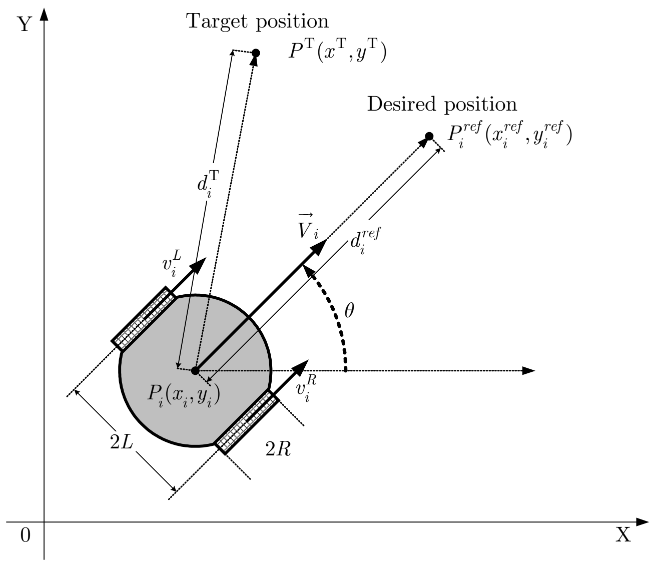


Figure 3.2: Model of two wheeled mobile robot.

$$d_i^{ref} = \sqrt{(x_i^{ref} - x_i)^2 + (y_i^{ref} - y_i)^2} \tag{3.2}$$

The distance $d_i^{\mathrm{T}}$ between the current position $P_i$ $(x_i,\ y_i)$ and the target position $P^{\mathrm{T}}$ $(x^{\mathrm{T}},\ y^{\mathrm{T}})$ of the robot at the next sampling is defined as :

$$d_i^{\mathrm{T}} = \sqrt{(x^{\mathrm{T}} - x_i)^2 + (y^{\mathrm{T}} - y_i)^2} \tag{3.3}$$

And the angle to the desired position $\theta$ is defined as :

$$\theta = \tan^{-1}\left(\frac{y_i^{ref} - y_i}{x_i^{ref} - x_i}\right) \tag{3.4}$$

and in the next sampling program will minimize the distance to the target. From the equation (3.2) we can know where is the nearest target position with the robot position, and the mobile robot will decided and choose the final target. The nearest position of the mobile robot with the target will become *gBest*. The equation (3.1) is used for determining direction of mobile robot toward into the target.

Since dynamics of mobile robots are nonlinear, the technique of feedback linearization can be used to facilitate the control design. Because the nonholonomic constraints in the dynamics of the mobile robot, the mobile robot is not input-state linearizable, most feedback control methods for mobile robots use input-output linearization [14]. Thus, the system can be feedback linearized to a two dimensional double integrator if the orientation is ignored and only focus on the position of an off-wheel axis point on the mobile robot [16].

## 3.6 Moving action of mobile robot

The moving action algorithm of each mobile for tracking the moving targets (T) is shown in Fig.3.3.

In initialization $t = 0$s, first of each mobile robot has some information such as position themselves, others mobile robot position in the surrounding areas and the distance to the target position. This information will be used for PSO, each mobile robot will check the distance between his own position with the moving target position ($d_i^{\mathrm{T}}$). Next, performed tracking the shortest path to reach the target using PSO method.

The moving action algorithm of each mobile for 2 targets (T1 and T2)is shown in Fig. 3.4. In initialization $t = 0$s, first of each mobile robot has some information such as position themselves in unfamiliar surroundings, another mobile robot position in the surrounding areas and the distance to the target position. This information will be used for PSO.

Each mobile robot will check the distance between his own position with target 1 position ($d_i^{\mathrm{T1}}$) and target 2 position ($d_i^{\mathrm{T2}}$).The mobile robots will decides which the target area nearest with the mobile robot. Next, performed tracking the shortest path to reach the target using PSO method.

In an iteration, the mobile robot updates the nearest target position by a Euclidean distance equation. The information about the current position of each mobile robot, $gBest$, $pBest$ and $\vec{V}_i(t + H)$ for the velocity of mobile robot will be calculated by the PSO algorithm to obtain a new position of each mobile robot.

During moving action to reach to the target area, mobile robot also checks availability and distance of the obstacle (i.e. walls or other robots) by using a sensor. If have an obstacle during moving action to reach the target, by using avoidance algorithm, mobile robot will avoid the obstacle and use his last position closer to the obstacle as an input to the PSO algorithm to get the next new position.

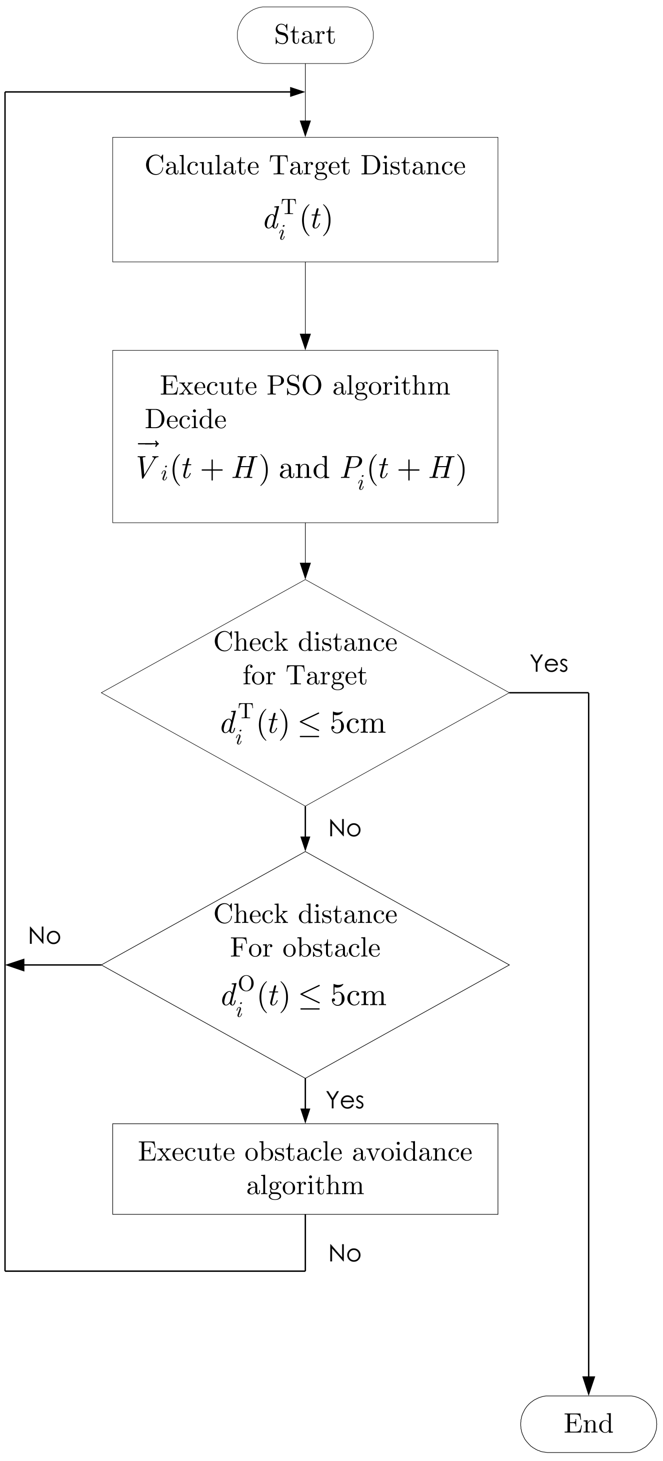

Start
Calculate Target Distance
$d_i^{\mathrm{T}}(t)$
Execute PSO algorithm
Decide
$\overrightarrow{V}_i(t+H)$ and $P_i(t+H)$
Check distance
for Target
$d_i^{\mathrm{T}}(t) \leq 5\mathrm{cm}$
Yes
No
Check distance
For obstacle
$d_i^{\mathrm{O}}(t) \leq 5\mathrm{cm}$
No
Yes
Execute obstacle avoidance
algorithm
No
End

Figure 3.3: Flowchart of moving action for moving target.

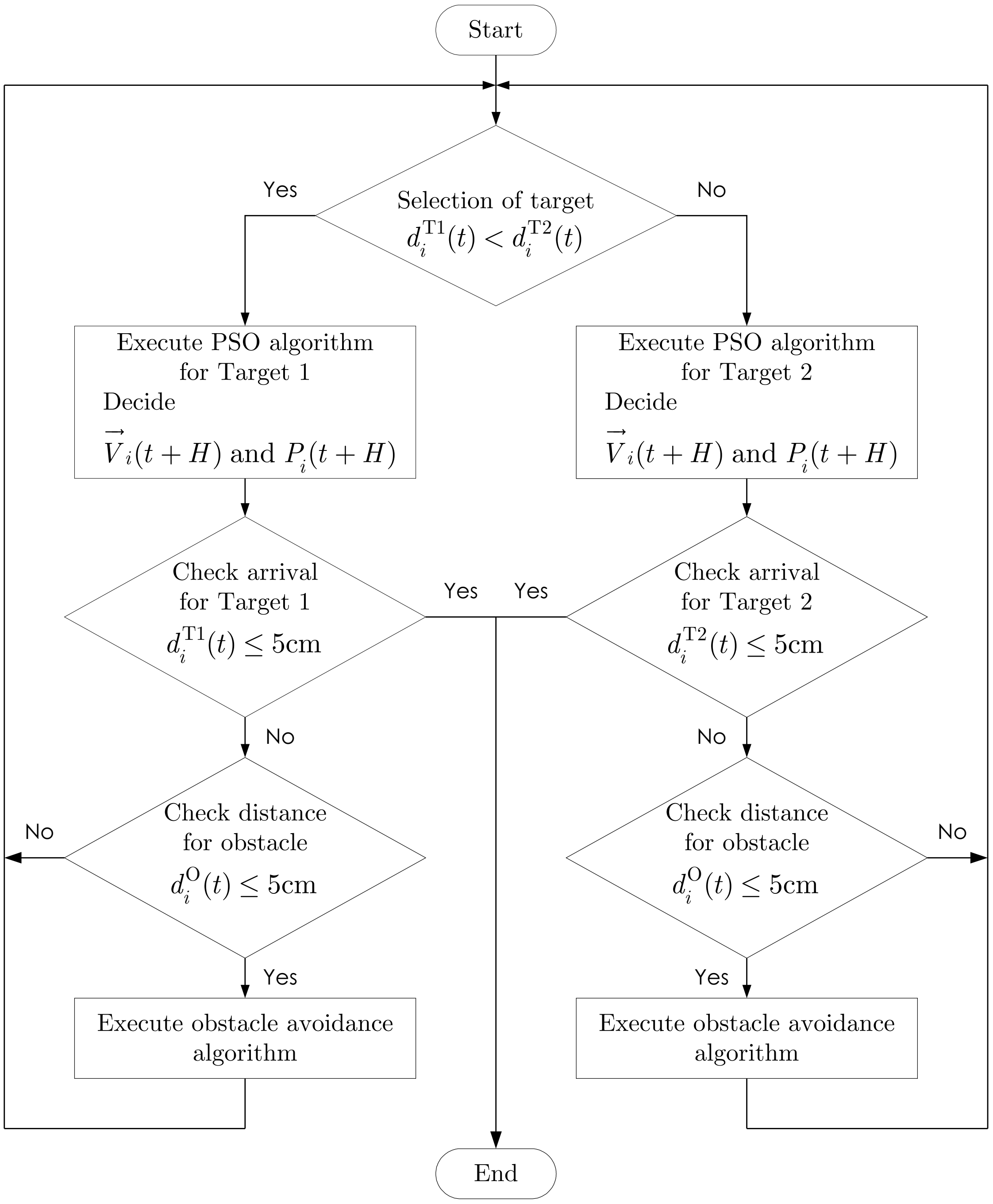

Start
Selection of target
$d_i^{T1}(t) < d_i^{T2}(t)$
Yes
No
Execute PSO algorithm for Target 1
Decide $\vec{V}_i(t+H)$ and $P_i(t+H)$
Execute PSO algorithm for Target 2
Decide $\vec{V}_i(t+H)$ and $P_i(t+H)$
Check arrival for Target 1
$d_i^{T1}(t) \leq 5\text{cm}$
Yes
Yes
Check arrival for Target 2
$d_i^{T2}(t) \leq 5\text{cm}$
No
No
Check distance for obstacle
$d_i^{O}(t) \leq 5\text{cm}$
No
Check distance for obstacle
$d_i^{O}(t) \leq 5\text{cm}$
No
Yes
Yes
Execute obstacle avoidance algorithm
Execute obstacle avoidance algorithm
End

Figure 3.4: Flowchart of moving action for 2 targets.

# CHAPTER 4

# Developed Mobile Robot System

In this study, we discuss the cooperative control system of multiple mobile robots using particle swarm optimization on the 2D coordinate environment. To realize cooperative target tracking experiment using real robots, we developed the multiple mobile robots and environment system.

## 4.1 Developed environment system

In this system, we used the positions of six mobile robots and relative distance between each mobile robot and targets, that is calculated from image information observed from a video camera over workspace in real time process shown in Fig. 4.1.

A host PC has two inputs, one is an image from a video camera mounted on the top of the workspace, from the images received by the PC we get the information about the position of each mobile robot and the distance between each robot to the target. The other input receives distance sensor data in real time also through XBee wireless communication, in this case XBee wireless communication on mobile robot alternately receive command signal and sending distance sensor data. A host PC calculates the control signal for each mobile robot using the cooperative control system with particle swarm optimization algorithm and sends to each mobile robot through XBee wireless communication, in this case XBee wireless communication on host PC alternately receive distance sensor data and sending command signal. Sampling time of the control system is set to 40 ms.

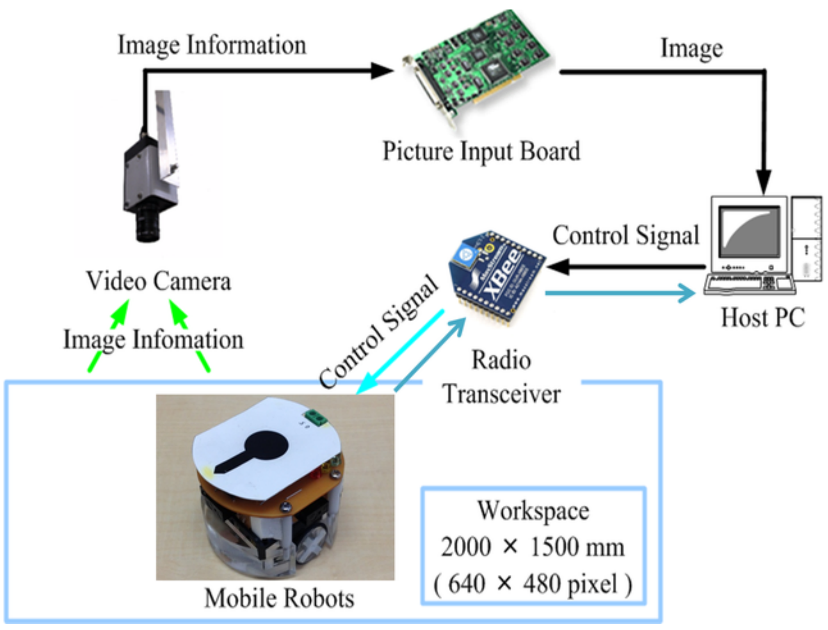
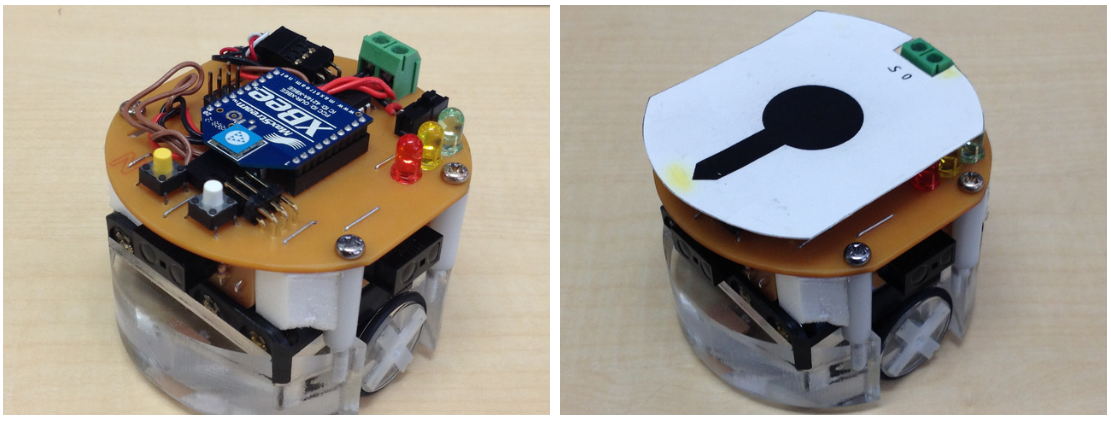


Figure 4.1: Environment system

## 4.2 Developed mobile robots

Figure 4.2 shows the appearance of the developed mobile robot with two wheels named P-2. The overall height is 55 mm, diameter is 70 mm, and the total weight of the P-2 is 550 g.



(a) Mobile robot without sign (b) Mobile robot with sign

Figure 4.2: Developed two wheels mobile robot: P-2

In the each mobile robots there are three fix range distance sensors. The sensors are located on front, right side and left side of each mobile robot. Each distance sensor has a detection fixed distance is 50 mm with a digital output. Their physical specification of a mobile robot is listed in Table 4.1.

The mobile robots with AVR ATmega88p as a controller can receive data commands from the host PC then process it become a moving action and checking distance sensor condition continuously for detecting the obstacle then send the data to the host PC.

Table 4.1: Specifications of P-2

| | |
|---|---|
| Robot Size | |
| Height [mm] | 55 |
| Diameter [mm] | 70 |
| Weight [g] | 550 |
| Microcontroller | |
| Type | AVR ATmega88P |
| Frequency [MHz] | 20 |
| DC geared Motor | |
| Type | GWS PICO/STD/F |
| Max Speed[cm/s] | 9.4 |
| Radio Tranceiver | |
| Type | XBee (3.3V UART) |
| Frequency band [GHz] | 2.4 |
| Baudrate Max. [bps] | 115200 |
| Distance Sensor | |
| Type | Sharp GP2Y0D805Z0F |
| Package size [mm] | 13.6x7x7.95 |
| Power Consumption [mA] | 5 |
| Range [mm] | 50 (Fixed) |

To control the mobile robot, we use AVR ATmega88p micro-controller. The AVR is one of the popular micro-controller families to use on chip flash memory include that ROM, EPROM, and EEPROM. It can output the PWM signal. The parameters and figure for hardware of mobile robot are shown in Appendix A.

## 4.3 Image processing system

For this experiment, we was prepared the environment for mobile robot system with size 2.0x1.5m. We used CCD video camera with resolution 640x480 pixel mounted on the top of the workspace for getting moving image. And Image from CCD video camera will process using opencv based on Microsoft visual C++.

# CHAPTER 5

# Experiment result

In this section, to confirm the validity of cooperative control of multiple mobile robot using PSO algorithm, first we show the simulation result in many different case. Then we show the experimental result in real environment of this research.

## 5.1 Simulation experiment

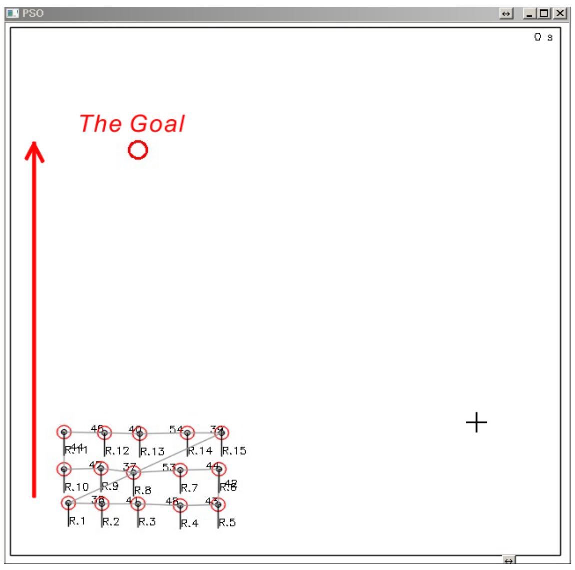
In this section we perform some simulations in different cases to validate the feasibility of the proposed method. The parameters of PSO in the experiment are set as follows : $c_1 = c_2 = 1.5$, $\omega = 0.5$, maximum velocities of mobile robot is 9.4 cm/s and the initial position of each mobile robot is random in unknown environment. For developed simulation program, we use opencv based on Microsoft visual C++ for making moving action animation of mobile robot. In this simulation, there are several types of simulation to show PSO and obstacle avoidance algorithm can solved cooperative control of multiple mobile robot.

### 5.1.1 Simulation result for following moving target

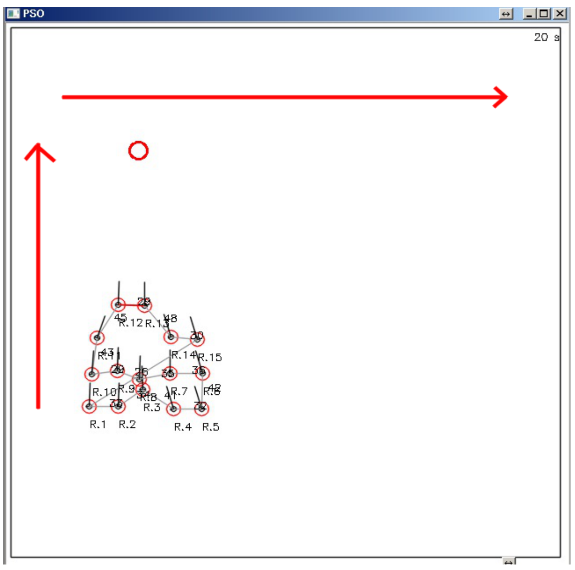
In this research, we have purposed PSO methode which control multiple mobile robot to find and following the moving target position in unknown environment. In this simulation the number of mobile robot is 15. Figure 5.1 shows the response of 15 mobile robot can move toward the goal quickly during avoid wall and others mobile robot in unknown environment. And also each mobile robot can follow the moving target position.

In the beginning of simulation, position of each mobile robot and the targets position in unknown environment is shown as Fig. 5.1(a).

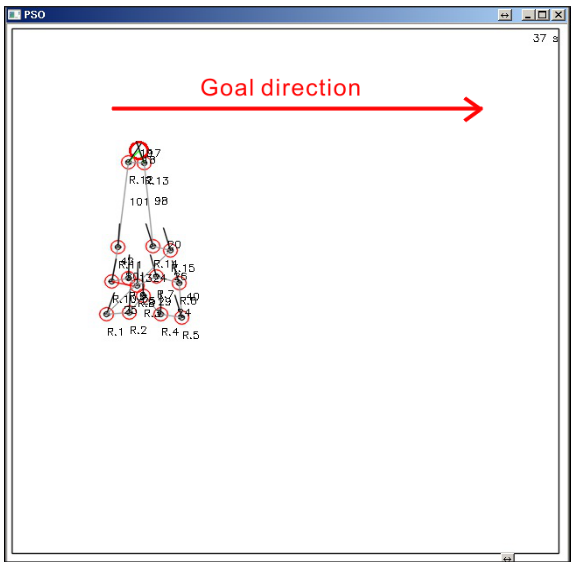
In the $t$=37s shown in Fig. 5.1(d), each mobile robot moves to the nearest target position. After one of the mobile robot touch the target, target will starting move with square trajectory in the unknown environment. Until $t$=125s shown in Fig. 5.1(h), the mobile robots still can follow the moving target.



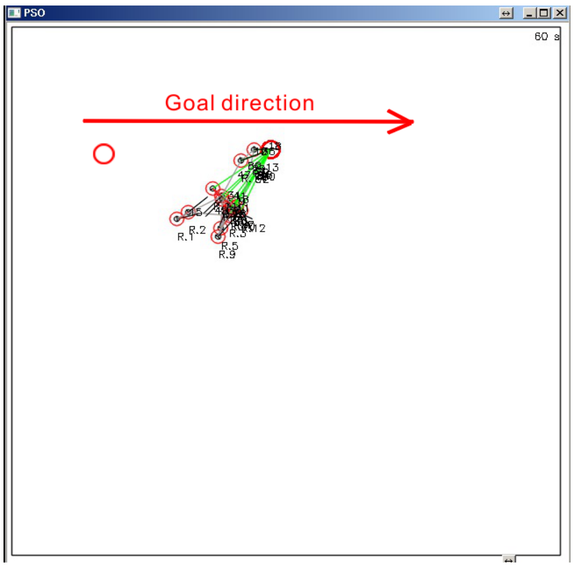
(a) $t$=0s



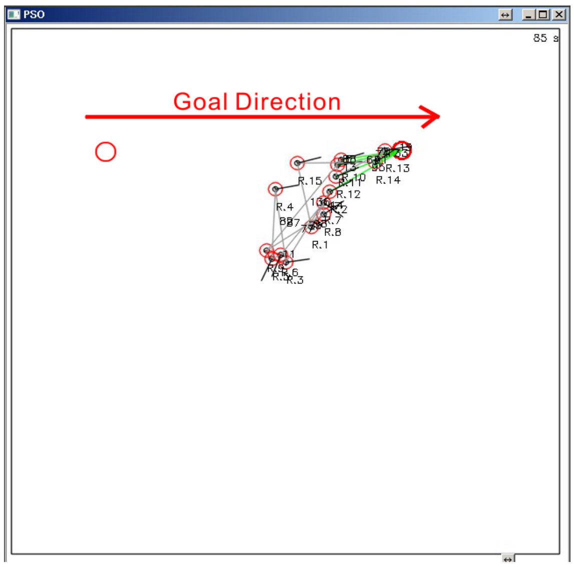
(b) $t$=20s



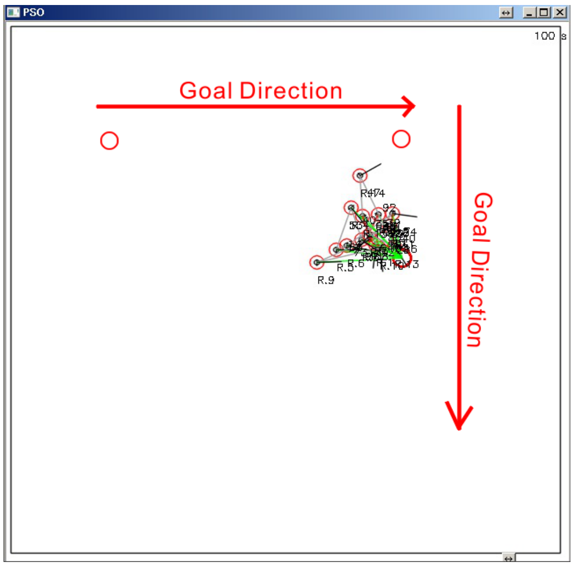
(c) $t$=37s



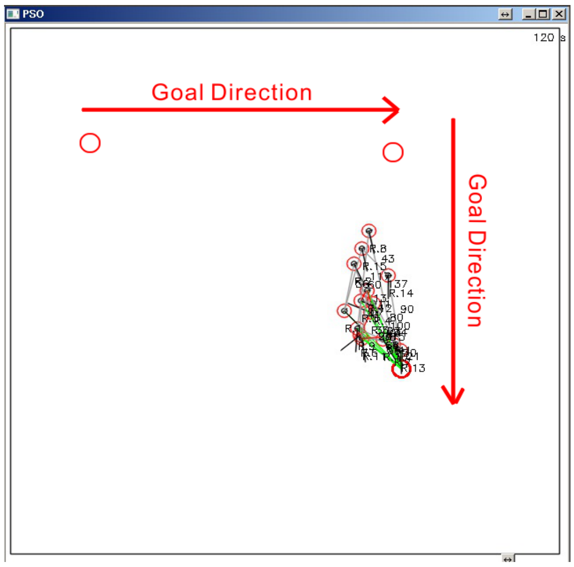
(d) $t$=60s



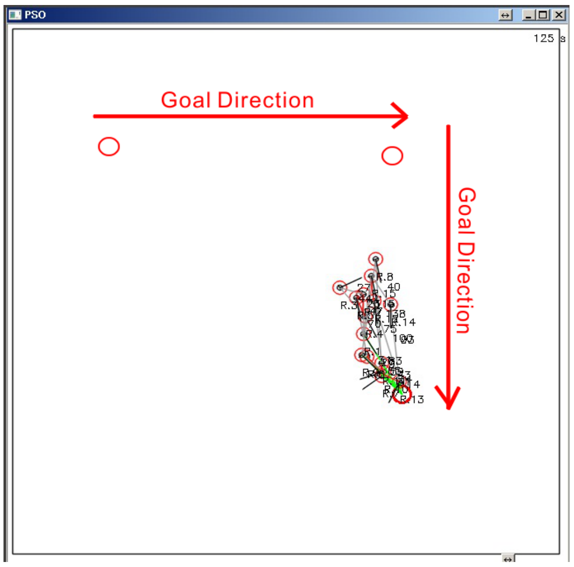
(e) $t$=85s



(f) $t$=100s



(g) $t$=120s



(h) $t$=125s

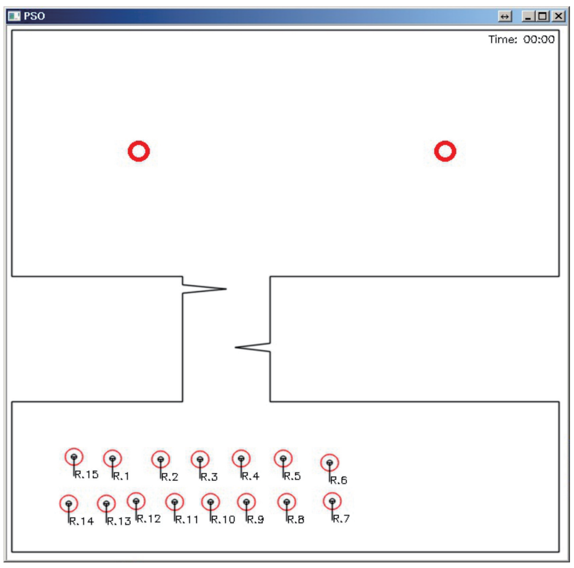
Figure 5.1: Trajectories of 15 two-wheeled mobile robot using PSO-based method

### 5.1.2 Simulation result for tracking two passive target

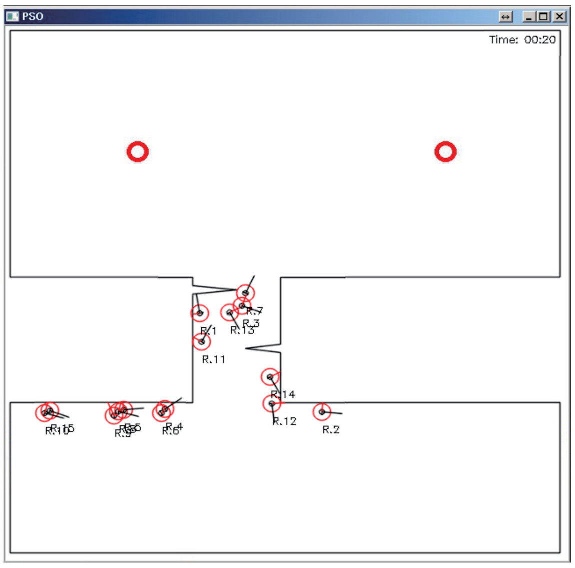
In this research, we have proposed the use of PSO method in order to cooperative control multiple mobile robot to reach two different position of target in unknown environment. Fifteen mobile robots are used in this simulation.

Figures 5.2(a)-5.2(f) show snap shots during the moving of 15 mobile robots to the two different positions of the target in unknown environment. Each mobile robot can move towards the two target quickly and cooperatively during avoid wall and another mobile robot.

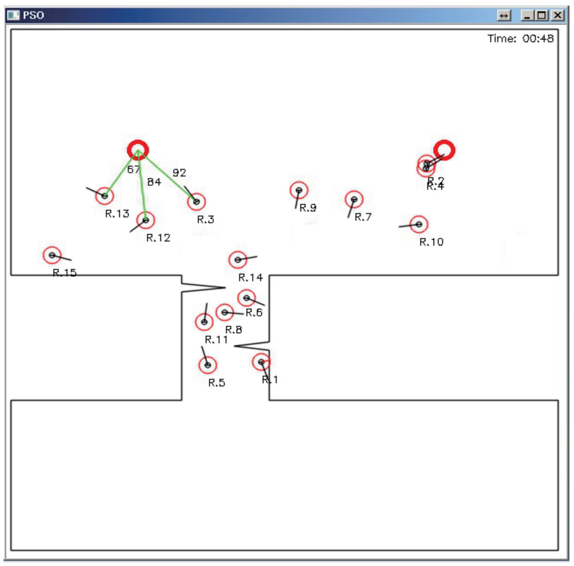
At the beginning of simulation, position of each mobile robot and the targets position in unknown environment is shown as Fig. 5.2(a).



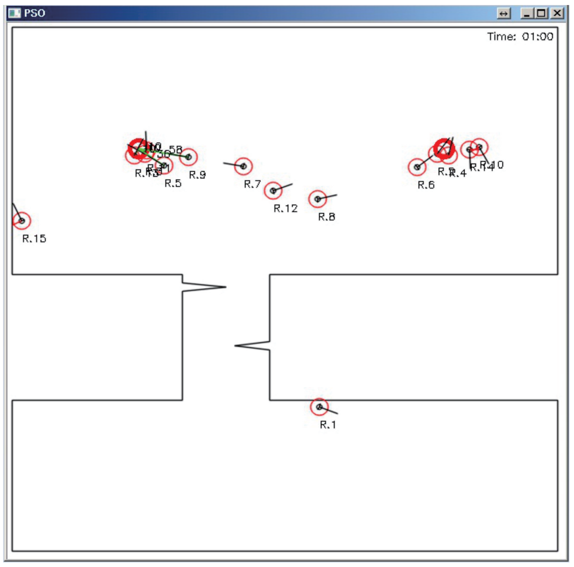
(a) $t$=0s



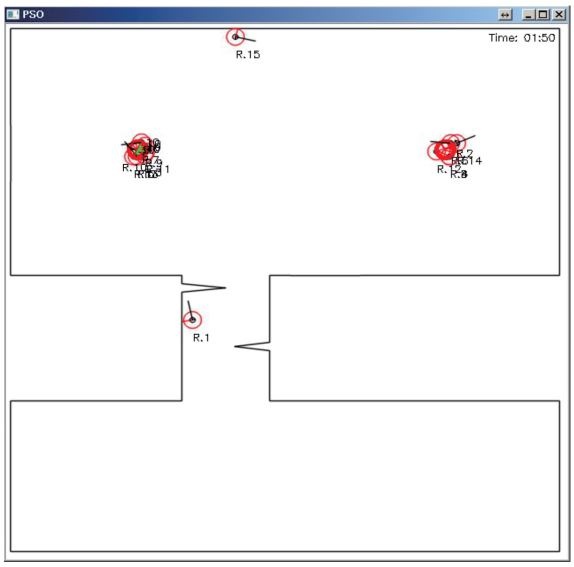
(b) $t$=20s



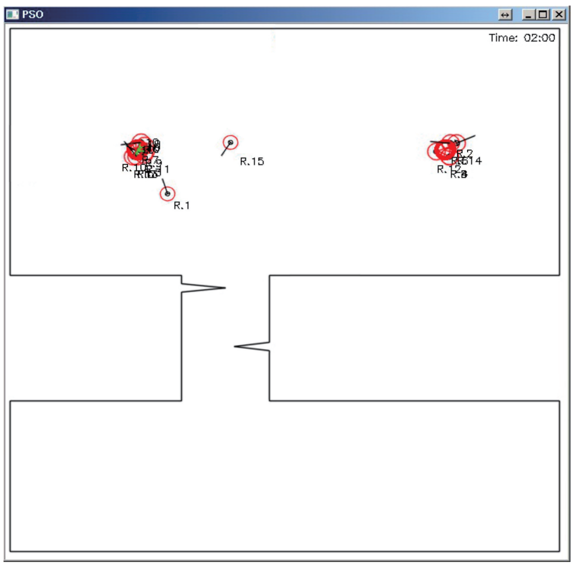
(c) $t$=48s



(d) $t$=60s



(e) $t$=110s



(f) $t$=120s

Figure 5.2: Tracking actions of the 15 mobile robots using PSO method

In Fig. 5.2(a), the mobile robot will moves from the current position to the one of target with its velocity. At thats time each mobile robot have three important information; current position of the mobile robot; another mobile robot position; distance between each mobile robot and distance between each mobile to the target.

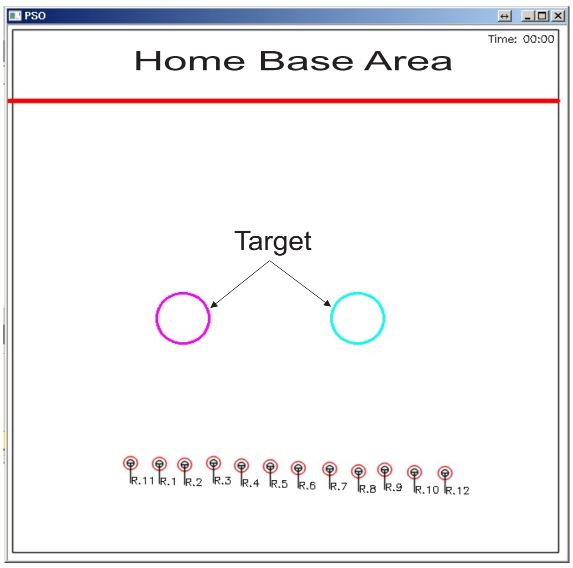
Figure 5.2(b) shows the moving actions of each mobile robot to reach their selected final target by avoidance the obstacles and through the narrow path. At the $t$=48s in Fig. 5.2(c), some of mobile robot can pass through the narrow path with obstacle, while others mobile robot still find the way to reach the target.

In the $t$=60s shown in Fig. 5.2(d), each mobile robot moves to the nearest selected target and some mobile robot already arrived in the final target position shown in Fig. 3(d). Figures 5.2(e) and 5.2(f) show that almost all of mobile robot already reach their selected target at $t$=110s, until $t$=120s two mobile robot still tracking the way go to their selected the final target.

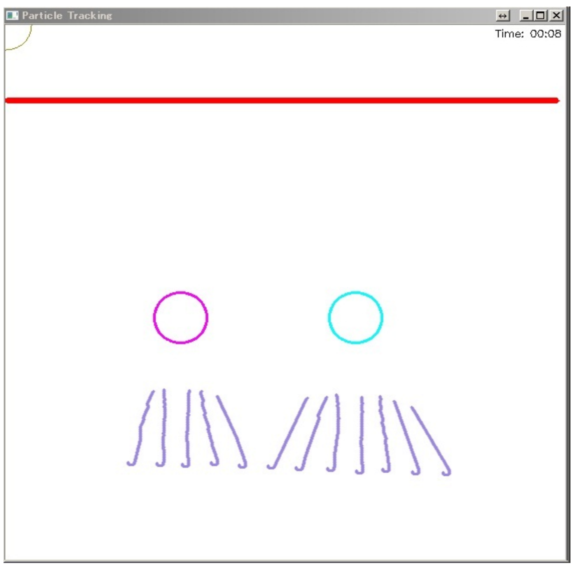
### 5.1.3 Simulation result for moving the target to the goal area

In this section, we perform some simulations in different cases to validate the feasibility of the proposed method. In this simulation the number of mobile robot is 12.

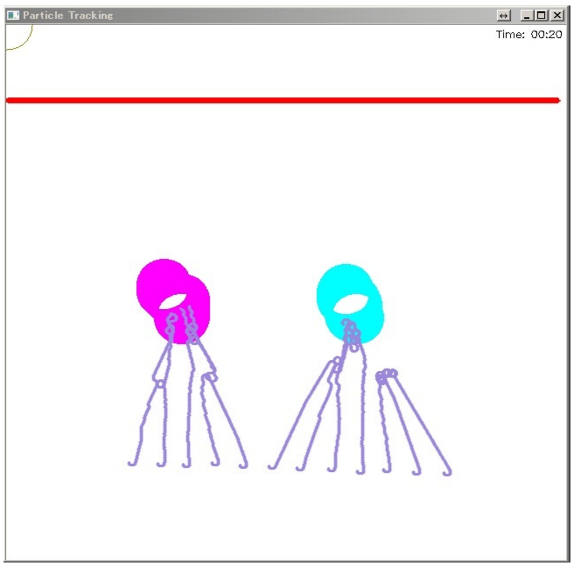
From Figs.5.3(a)-5.3(f), the robots can find the two different targets by using PSO-based method in unknown environment.



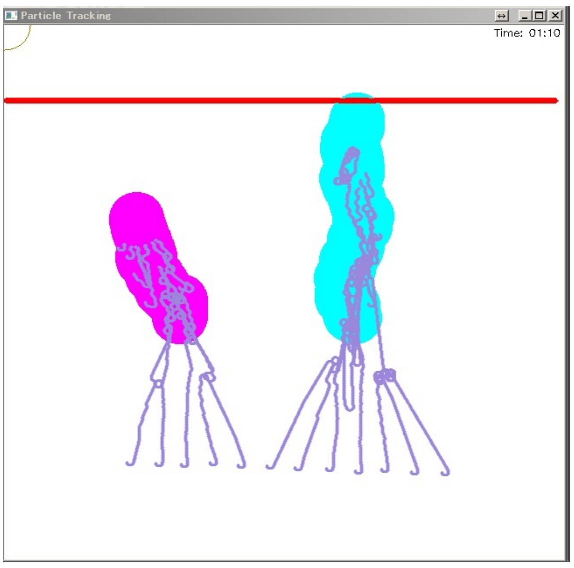
(a) Initial Position($t$=0s)



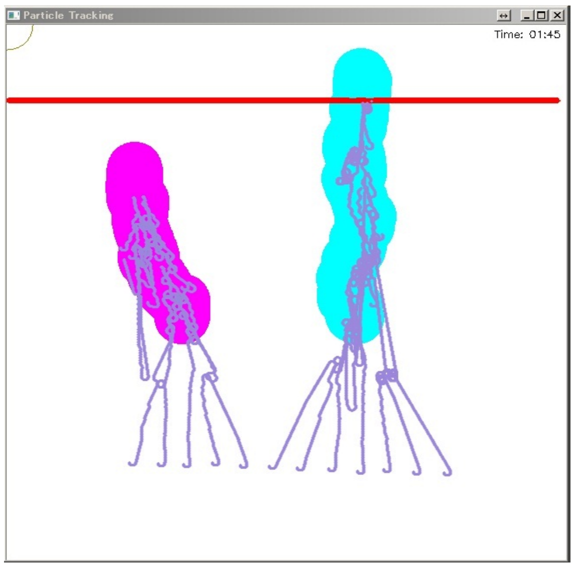
(b) $t$=8s



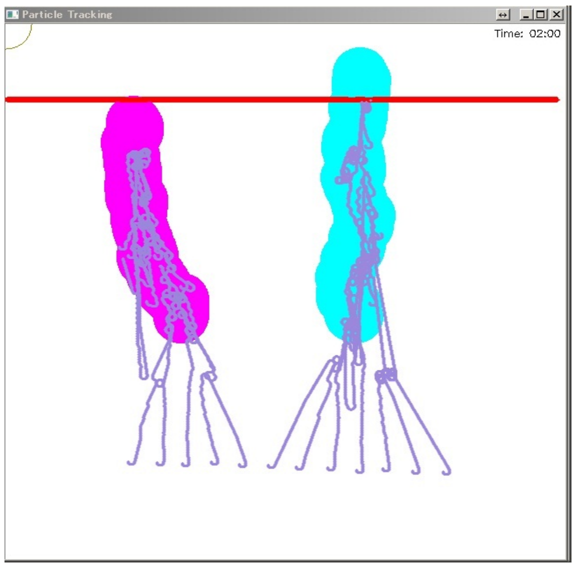
(c) $t$=20s



(d) $t$=70s



(e) $t$=105s



(f) $t$=120s

Figure 5.3: Trajectories of 12 mobile robots and two moving targets

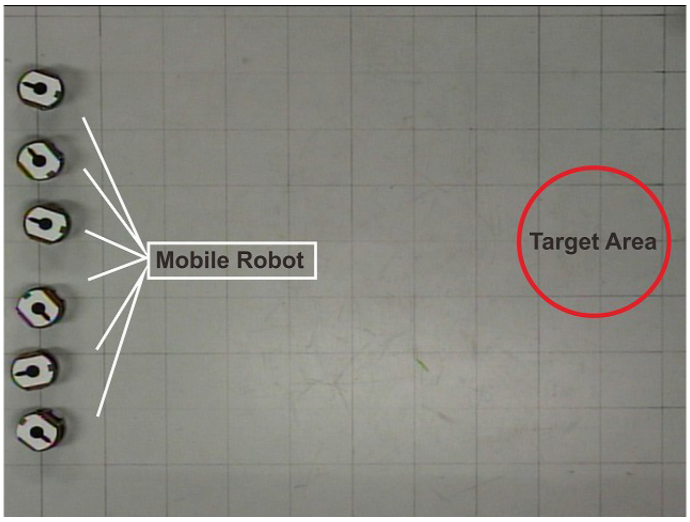
The positions of the two targets can be adjusted by the robot when the robot encourage it. During the simulation, the targets can moved into the home base area with two groups robots using cooperative control method.

Figure 5.3(a) shows the initial position of each mobile robot towards the target and goal area in unknown environment. At the $t$=20s in Fig. 5.3(c), some of mobile robot can touch and push the target go to the goal area, the goal area is the area after the red line. While others mobile robot still find the ways to reach the target.

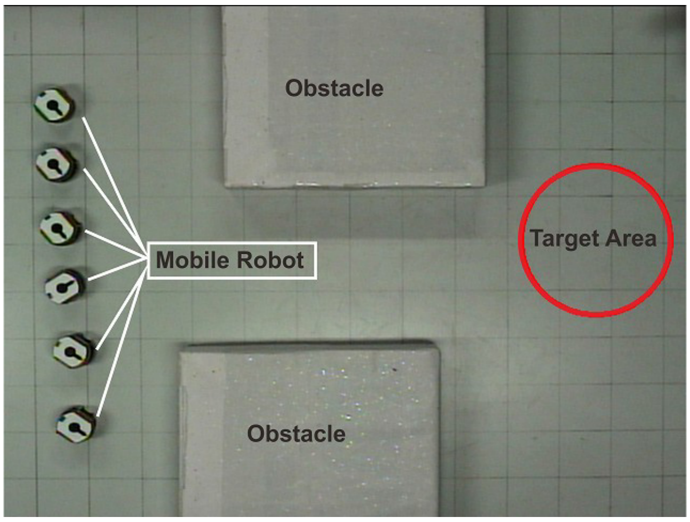
In the $t$=70s shown in Fig. 5.3(d), one of the group robot can reach the goal area with the target. And after $t$=120s two groups of mobile robot can reach to the goal area.

## 5.2 Experiment

In order to verify the validity of the cooperative control system using particle swarm optimization for tracking target, several settings and experiments using the developed mobile robots: P-2 was conducted. Six mobile robots are used in this experiment.



(a) Environment without obstacle



(b) Environment with obstacle

Figure 5.4: Environment setting with 1 target

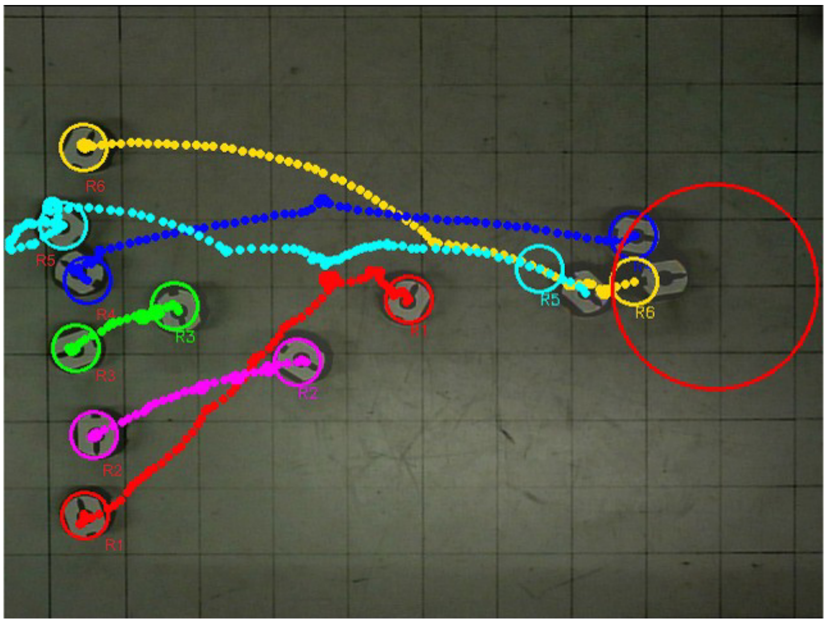
In the experiment, the size of working space is 200x150cm. Figure 5.4(a) shows the appearance of the environment without obstacle and Fig. 5.4(b) shows the appearance of the environment with obstacle. Each mobile robot is programmed in order to find the moving action without crashing the obstacle and other robots using PSO method.

Here, it is assumed that each mobile robots have information about the distance between itself current position with target area, itself position and another robot position close to the mobile robots.

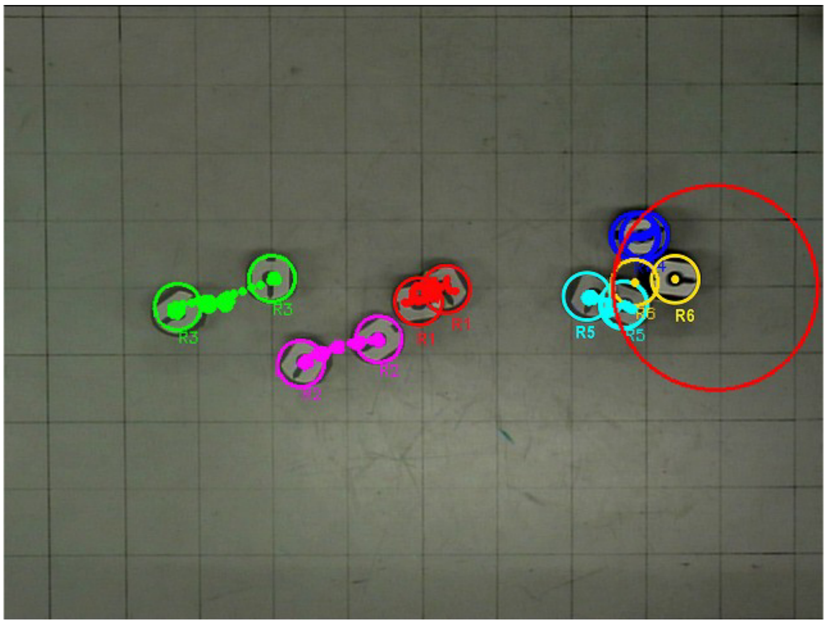
During the experiment, the mobile robot we make full use of the real-time information attained by updating the coordinate position of each mobile robot from a video camera over the workspace and distance sensor condition, in this case the Euclidean distance of the individual robots relative to the target, to analyze the status of their relative current position. The basic PSO algorithm with obstacle avoidance algorithm to accommodate for the obstacle (i.e. Walls or other robots) avoidance with cooperative and collective robotic search applications.

### 5.2.1 Tracking actions of mobile robots in unknown environment without obstacle and with one target area

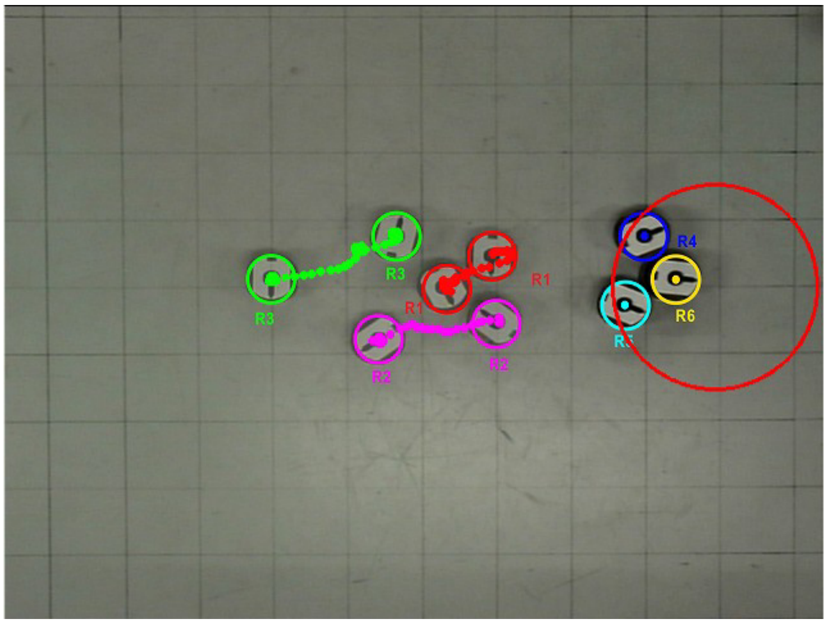
The parameters of PSO in the experiment are set as follows: $c_1$=$c_2$=1.5, $\omega$ = 0.5, $rand(*)$ = [0, 1], and maximum velocities of mobile robot is 9.4 cm/s.



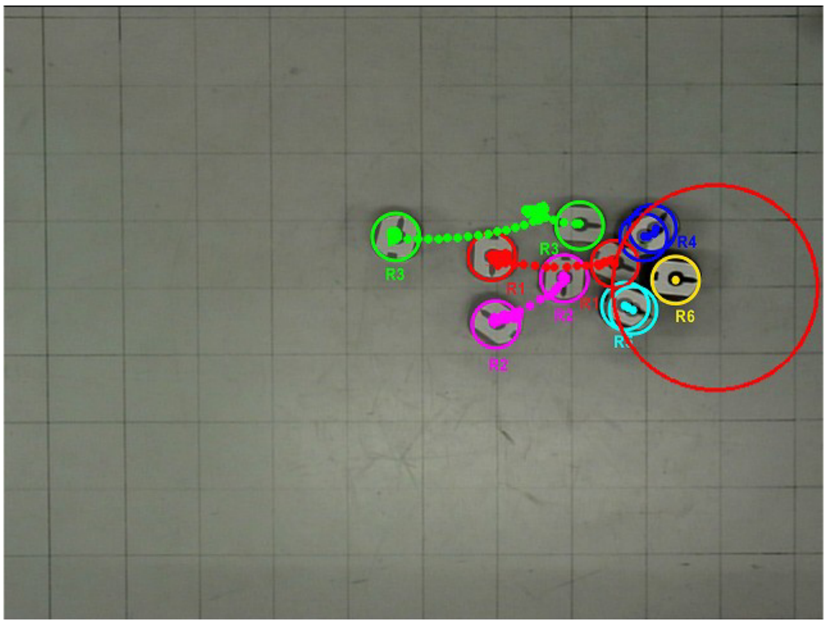
(a) $t$ = 0s - $t$ = 45s



(b) $t$ = 45s - $t$ = 90s



(c) $t = 90$s - $t = 100$s



(d) $t = 100$s - $t = 145$s

Figure 5.5: Tracking actions of mobile robots in unknown environment without obstacle and with one target area

By using particle swarm optimization method, the mobile robots can find and move towards to the target area in unknown environment. The result is shown in Figs.

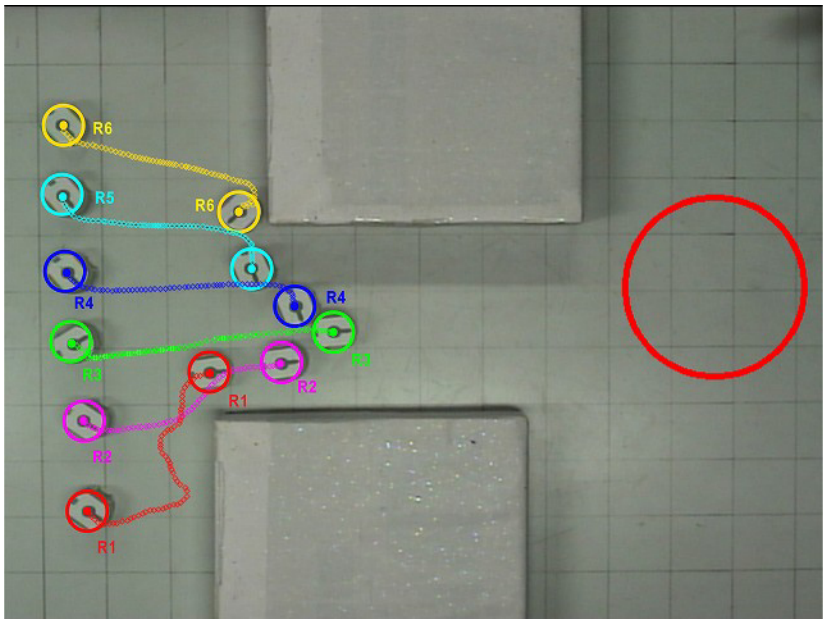
5.5(a)-5.5(d) until Figs. 5.8(a)-5.8(d). At the initial time $t = 0$s, each mobile robot in random positions on the unknown environment.

In the first experiment, we are setting the environment with 1 target area and without obstacle. In initial time $t = 0$s, each mobile robot position is shown in Fig. 5.5(a). After the $t = 45$s, several mobile robots has been reached in the target area is shown in Fig. 5.5(b) by using cooperative control algorithm with PSO method. During $t = 45$s until $t = 100$s, only two mobile robots are still trying to find the target area using information from other robots.

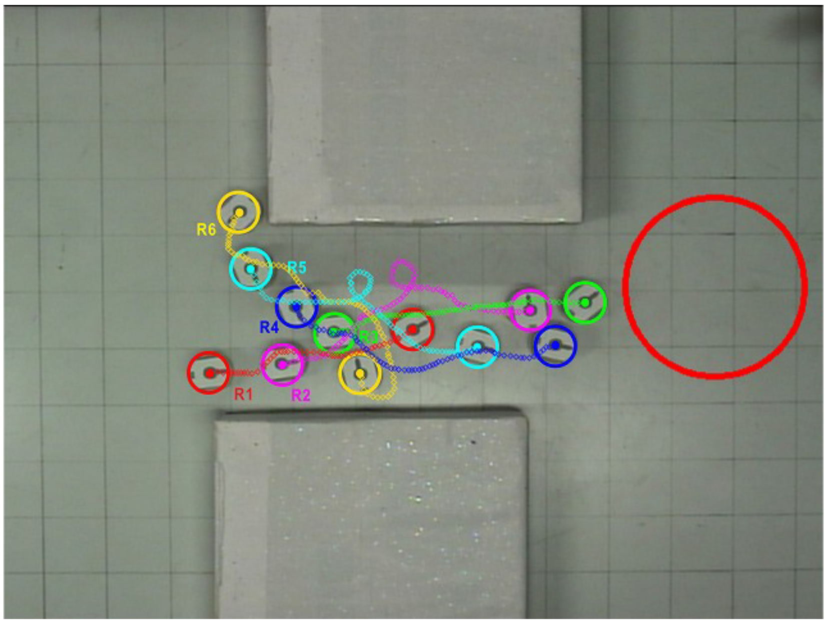
At the end of the experiment $t = 145$s, all of the mobile robots can reach surround the target area is shown in Fig. 5.5(d).

### 5.2.2 Tracking actions of mobile robots in unknown environment with two obstacle and one target area

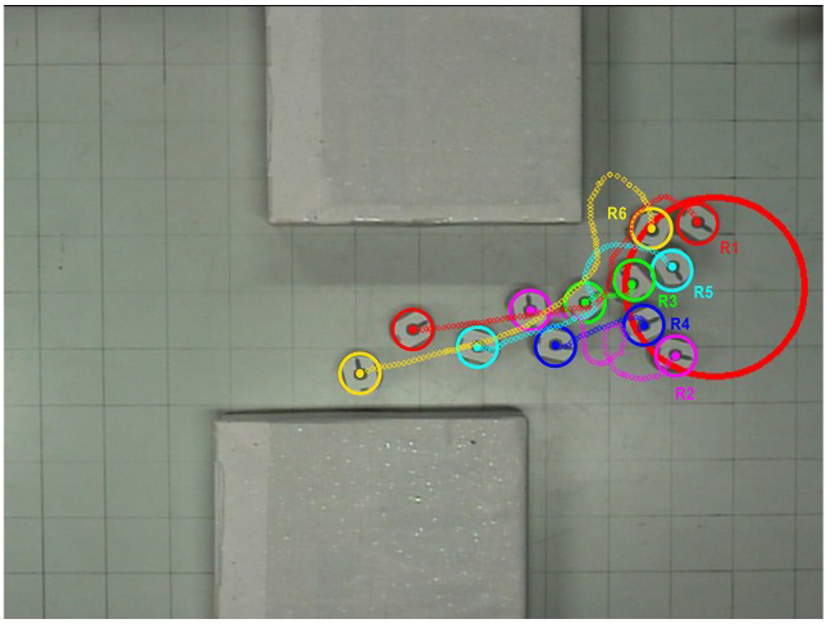
In the second experiment, we use obstacle in the unknown environment with 1 target area.



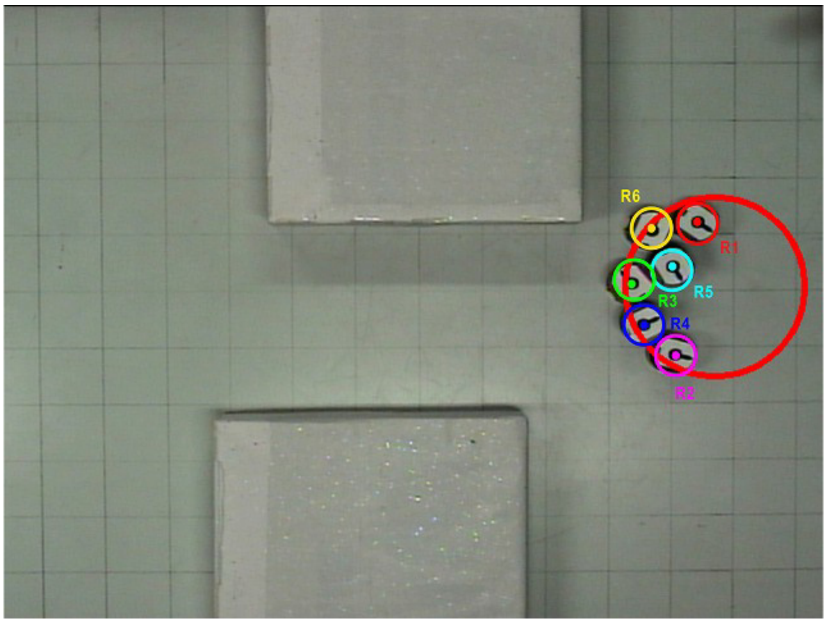
(a) $t = 0\text{s}$ - $t = 45\text{s}$



(b) $t = 45\text{s}$ - $t = 90\text{s}$



(c) $t = 90$s - $t = 110$s



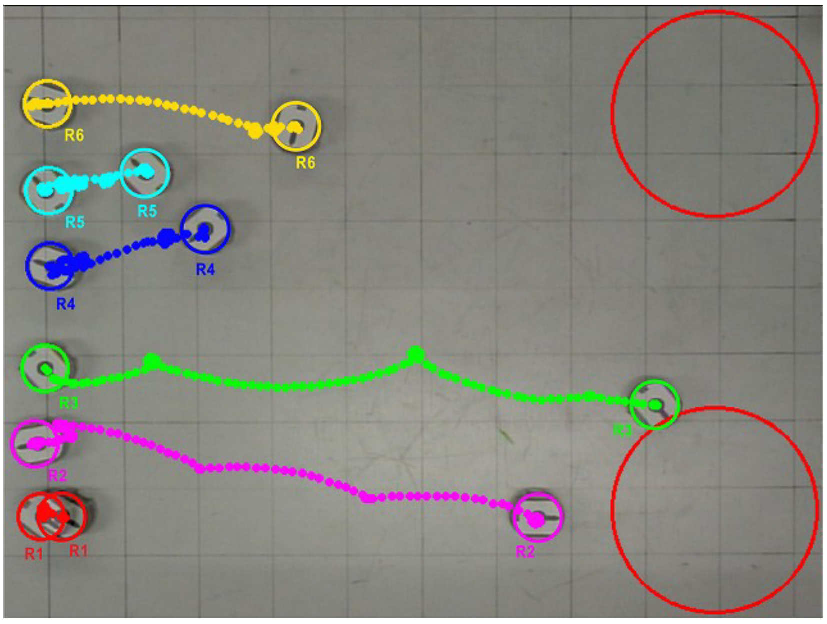
(d) $t = 110$s - $t = 160$s

Figure 5.6: Tracking actions of mobile robots in unknown environment with two obstacle and one target area

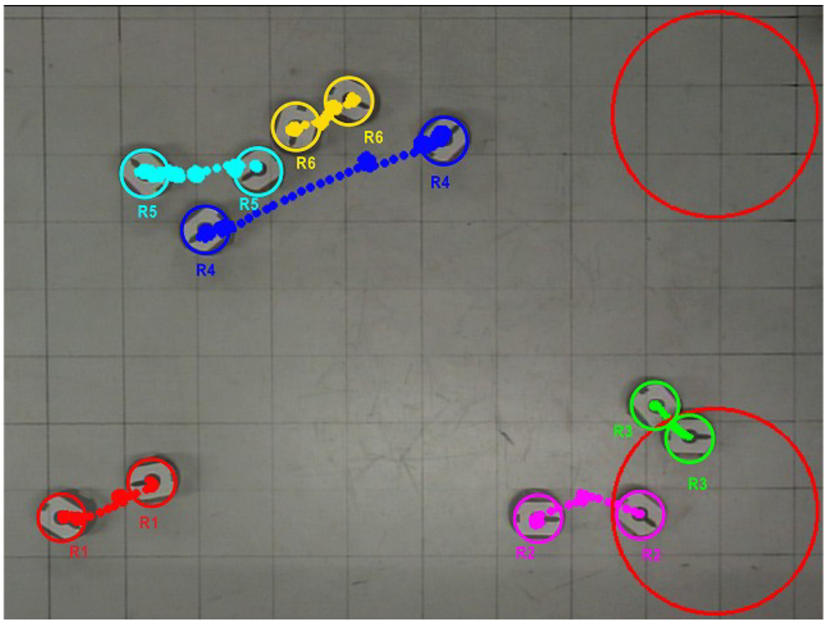
Figure 5.6(a) shows the initial position of each mobile robot in the environment. Until $t = 90$s, the mobile robots can not find the target area is shown as Fig. 5.6(b). After $t = 110$s, Fig. 5.6(c) shows several mobile robots has been reached in the target area. Each mobile robot can move towards the target area and cooperatively during avoid an obstacle and other mobile robot after $t = 160$s is shown in Fig. 5.6(d).

### 5.2.3 Tracking actions of mobile robots in unknown environment without obstacle and with two target areas

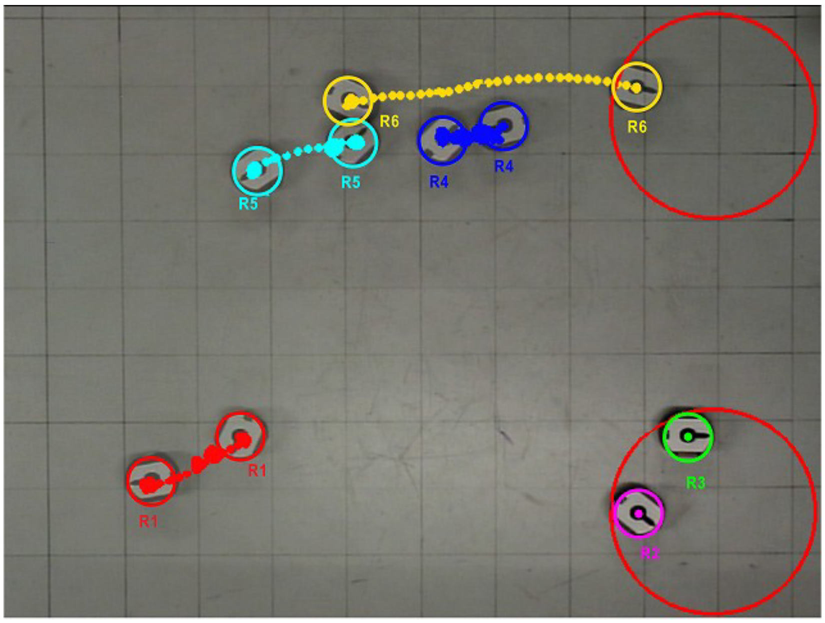
Figures 5.7(a)-5.7(d) show snapshots during the movements of six mobile robots with two target areas in an unknown environment without obstacle.



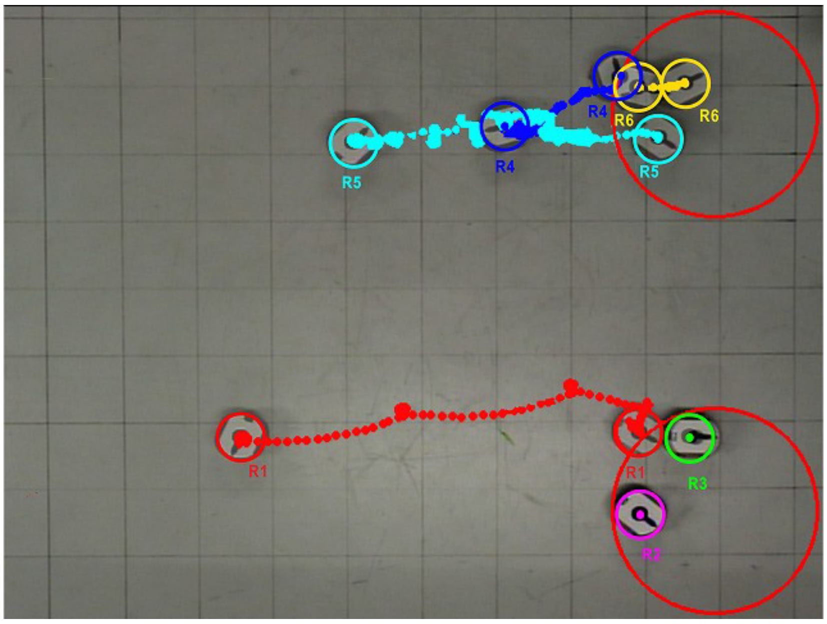
(a) $t$ = 0s - $t$ = 45s



(b) $t$ = 45s - $t$ = 90s



(c) $t = 90$s - $t = 120$s



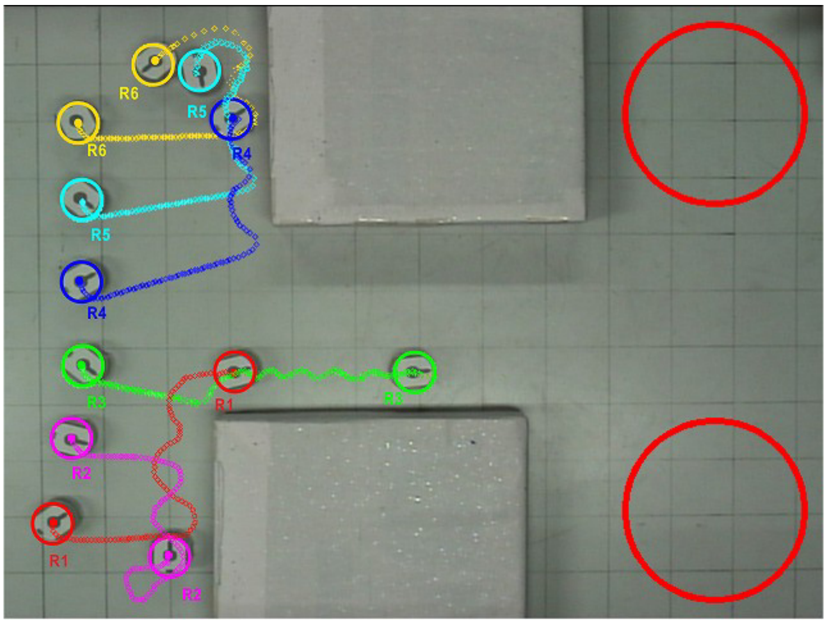
(d) $t = 120$s - $t = 155$s

Figure 5.7: Tracking actions of mobile robots in unknown environment without obstacle and with two target areas

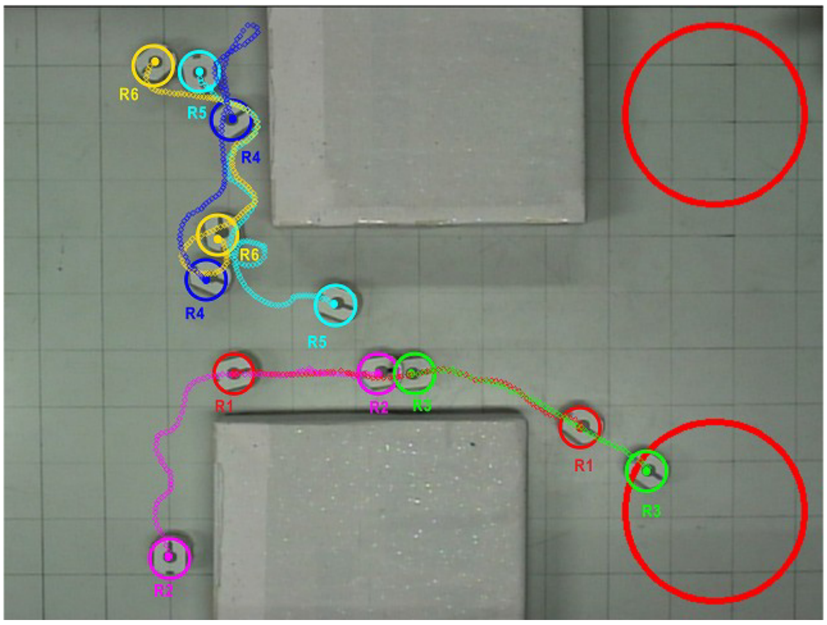
Each mobile robot can move towards the target area quickly and cooperatively during avoid other mobile robots. After the $t = 45$s, only one mobile robot can reach to the target area is shown as Fig. 5.7(a). All of the mobile robots has been reached in the target area after $t = 155$s is shown as Fig. 5.7(d).

### 5.2.4 Tracking actions of mobile robots in unknown environment with obstacles and with two target areas

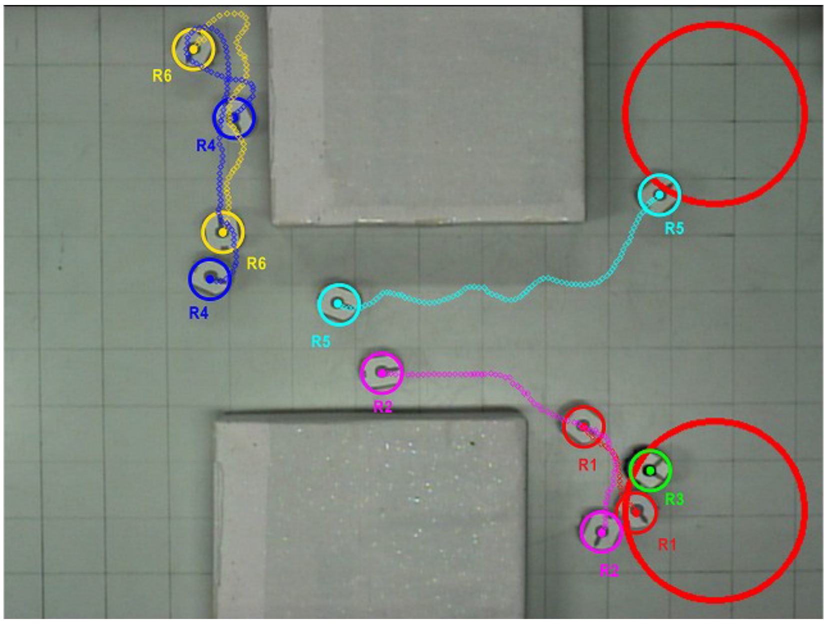
The last experiment, we are setting the environment with two target areas and with obstacles. In initial time $t = 0$s, each mobile robot is at random positions on unknown environment.



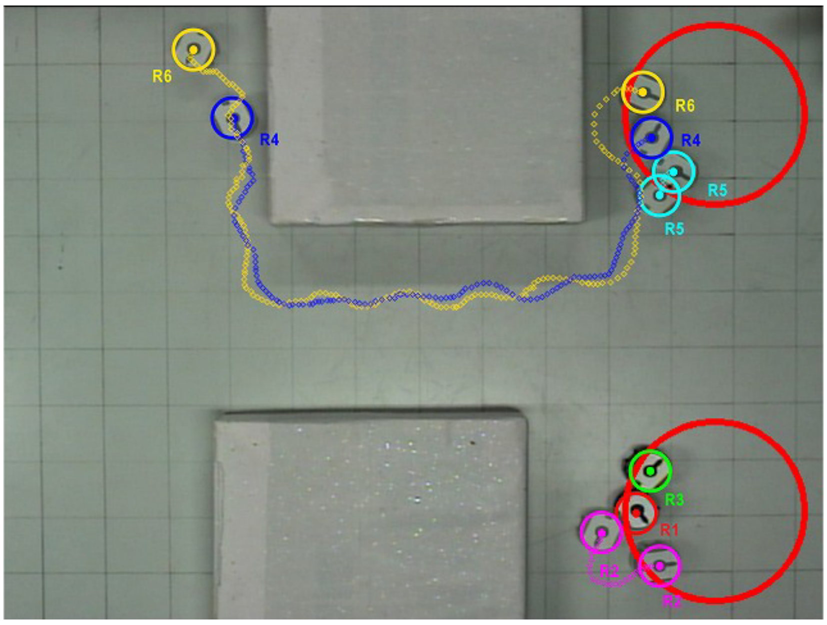
(a) $t = 0$s - $t = 45$s



(b) $t = 45$s - $t = 90$s



(c) $t = 90$s - $t = 120$s



(d) $t = 120$s - $t = 170$s

Figure 5.8: Tracking actions of mobile robots in unknown environment with obstacles and with two target areas

After $t = 90$s, one of mobile robots can reach one of the target areas and one of mobile robots is near one of the target areas as shown in Fig. 5.8 (b).

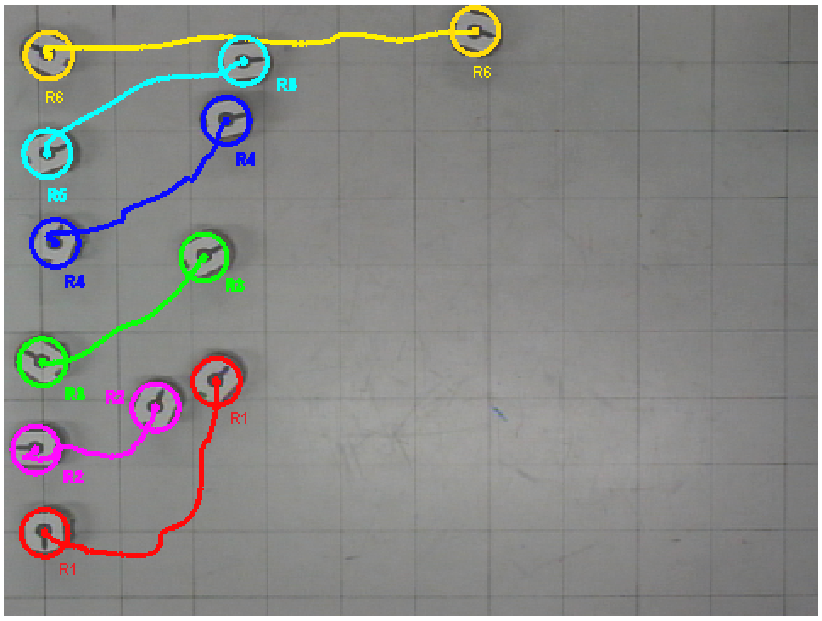
After $t = 120$s, four mobile robot has been reached in target areas is shown as Fig.

5.8 (c). Each mobile robot can move towards to target areas with cooperatively during avoid an obstacle and other mobile robot after $t = 170$s is shown in Fig. 5.8 (d).

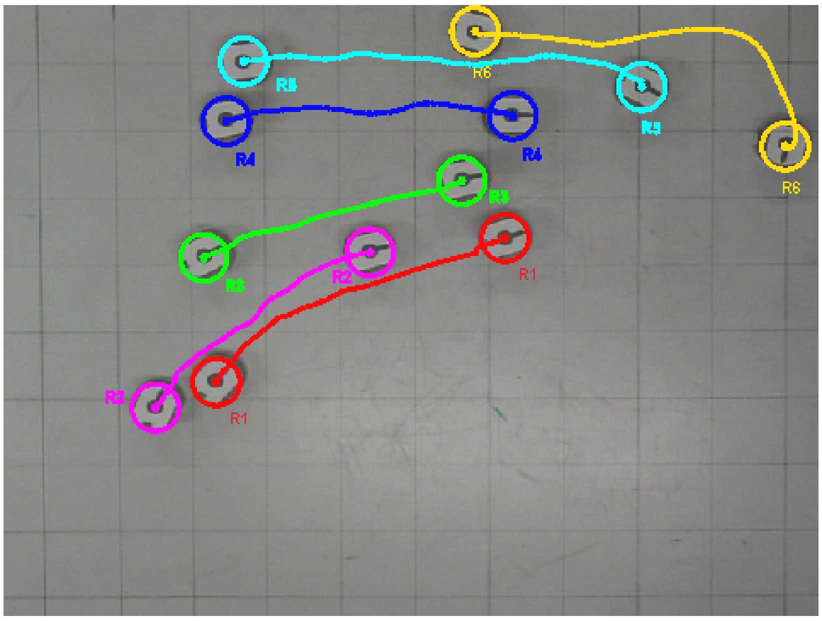
Moving actions of each mobile robot in an unknown environment to target areas shown in Figs. 5.5, 5.6, 5.7 and 5.8. The mobile robot will move from the current position to the one of the target with its velocity. Each mobile robot moves to the nearest selected target using PSO method and cooperative control algorithm.

### 5.2.5 Following the leader in unknown environment

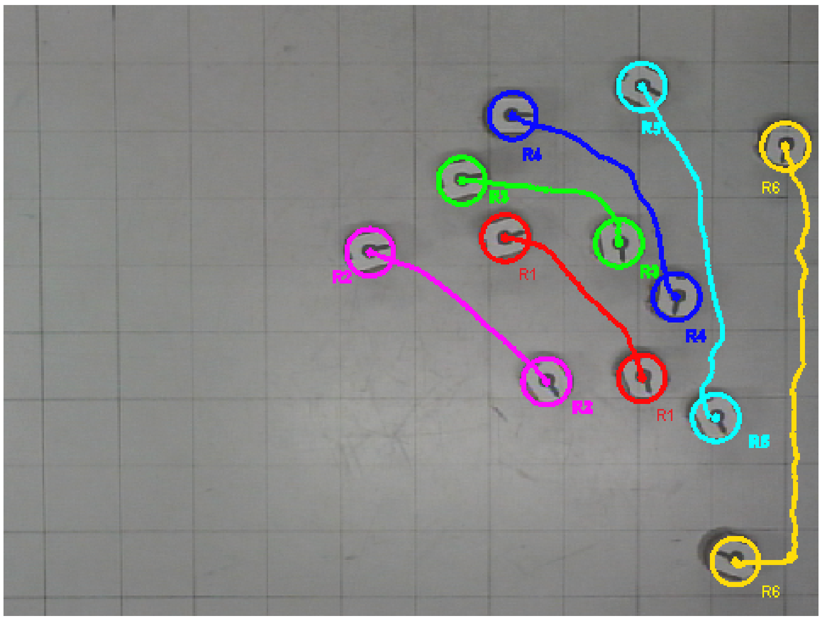
Figures 5.9(a)-5.9(f) show snap shots during the moving of 6 mobile robots to reach the moving target in unknown environment. Each mobile robot can follow and move towards around the leader and cooperatively during avoid wall and another mobile robot.



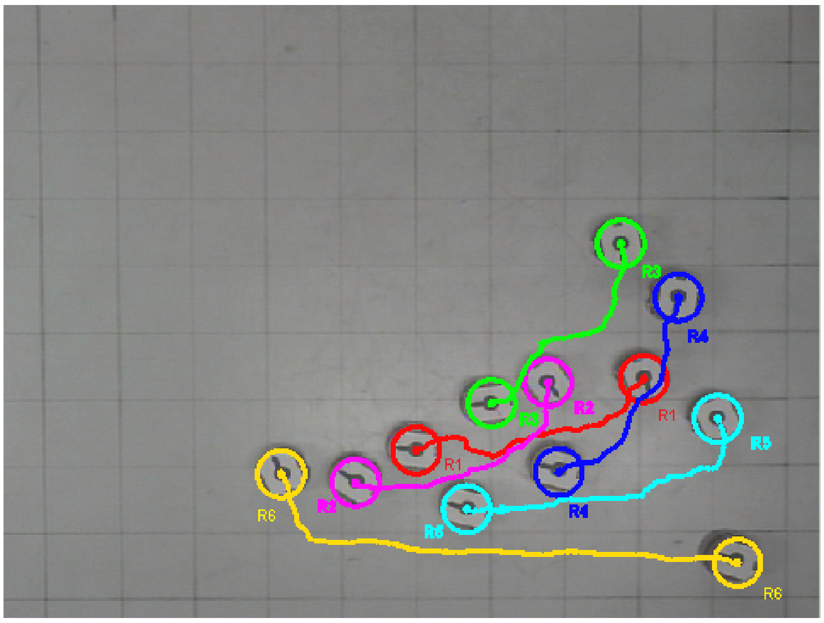
(a) $t$=0s-30s



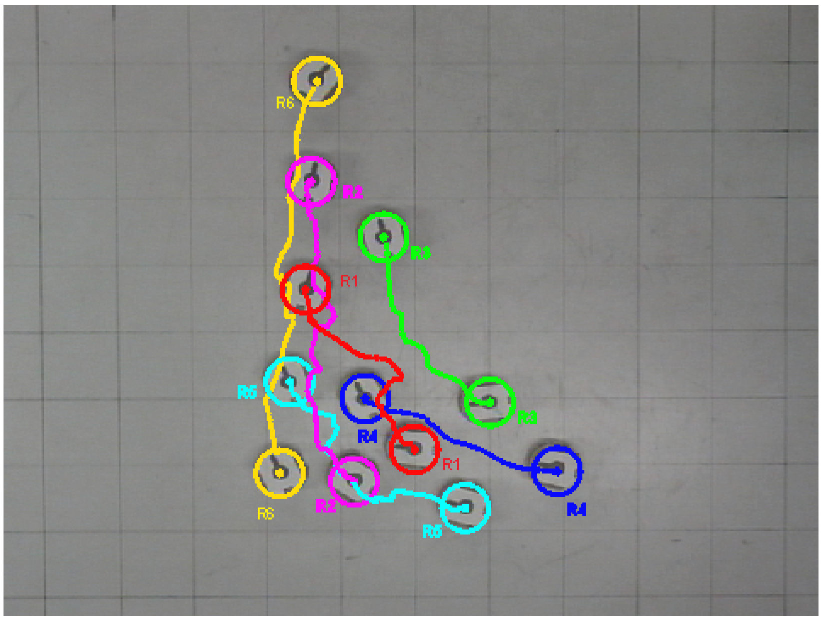
(b) $t$=30s-60s



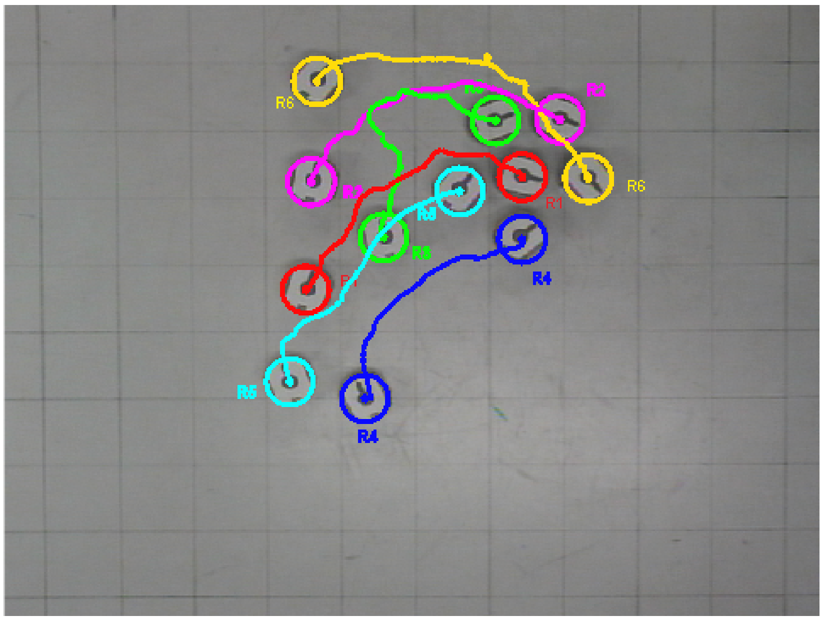
(c) $t$=60s-90s



(d) $t$=90s-120s



(e) $t$=120s-150s



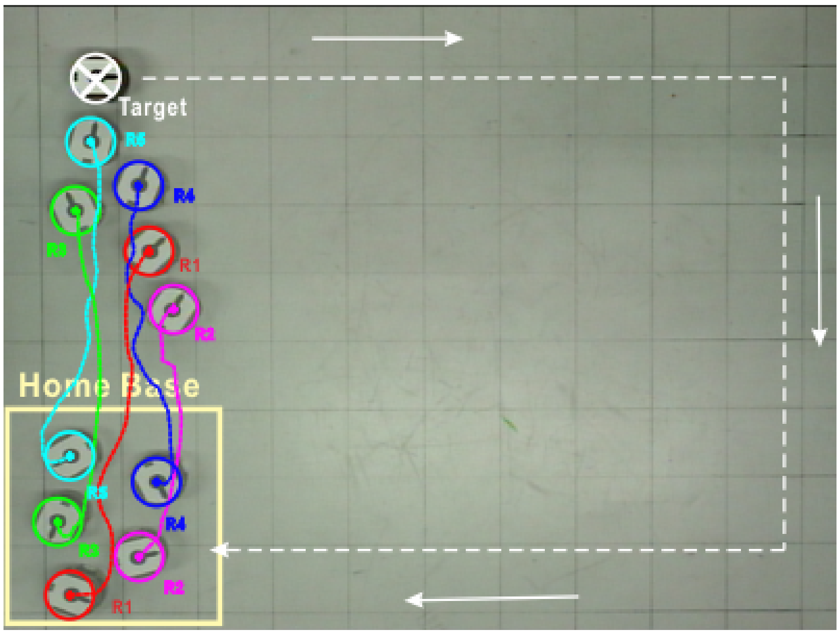
(f) $t$=150s-170s

Figure 5.9: Following the leader in unknown environment

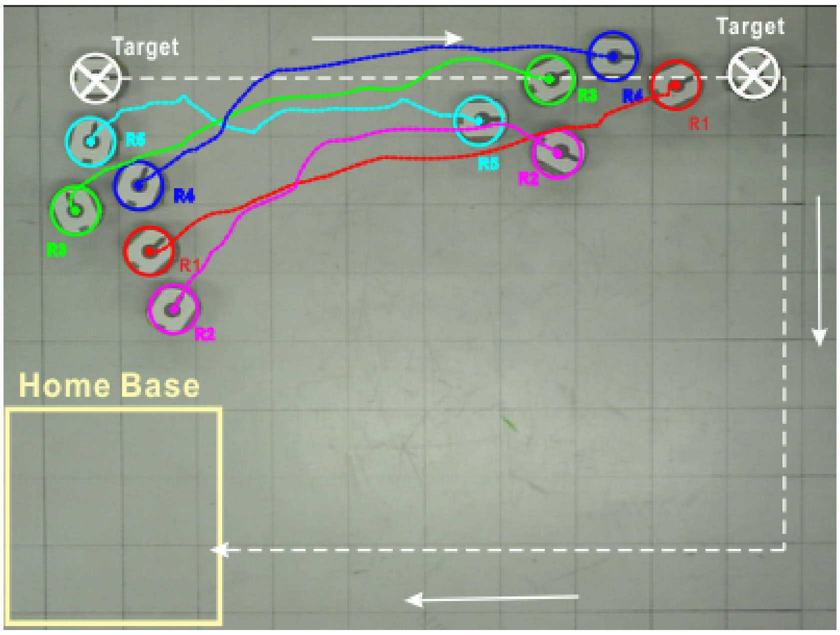
During the experiment, we make full use of the real-time information attained by updating the coordinate position of each mobile robot and distance sensor condition, in this case the Euclidean distance of the individual robots relative to the leader, to analyze the status of their relative current position.

### 5.2.6 Tracking actions in unknown environment with moving target

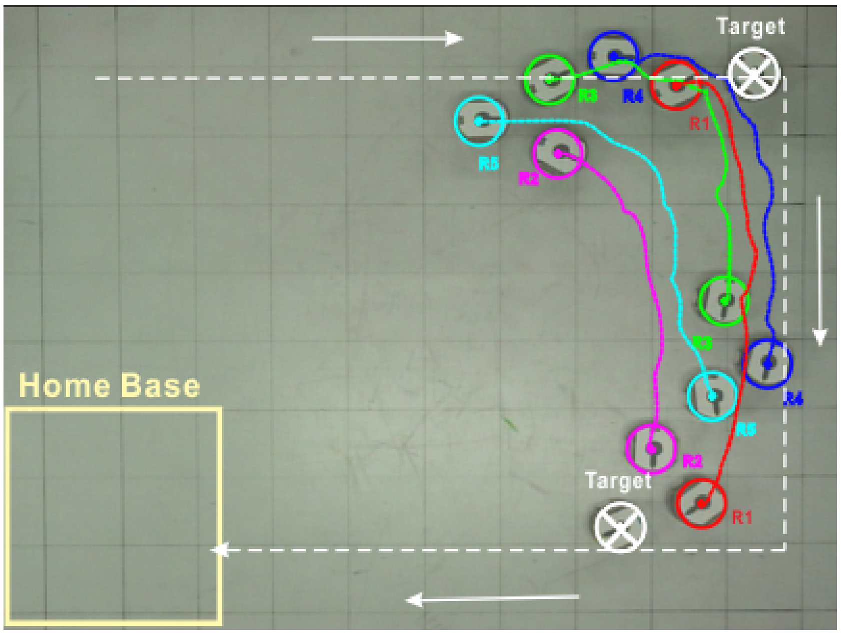
In the last experiment, we set the moving target in the environment. The targets can move with constant velocity and the follow the white dashed lines as shown in Fig. 5.10 (a).



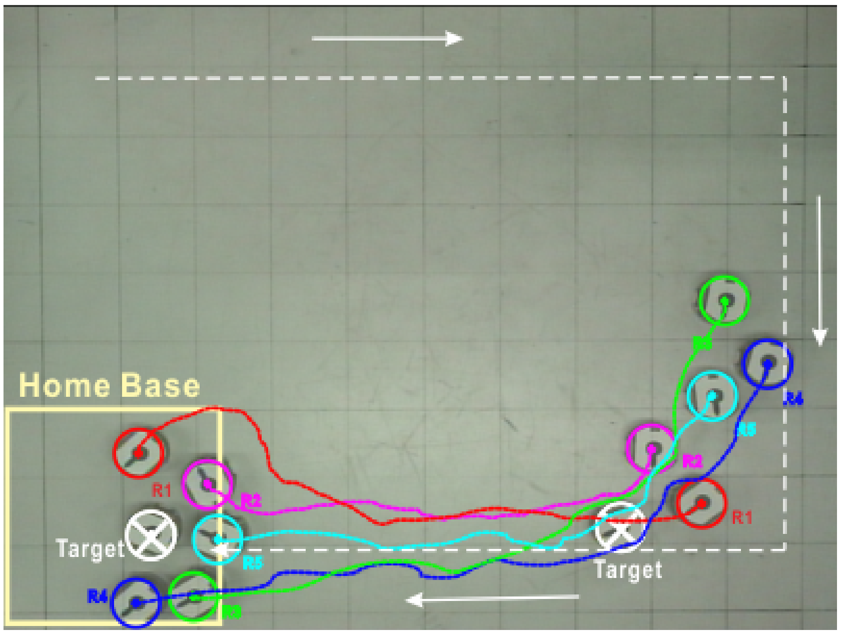
(a) $t$=0s-30s



(b) $t$=30s-60s



(c) $t$=60s-90s



(d) $t$=90s-120s

Figure 5.10: Tracking actions of mobile robots in unknown environment with moving target

In initial time $t$ = 0s, each mobile robot is at random positions in the home base area and the $gBest$ value can get from the mobile robot with light blue line. The target will start moving when the distance between the target and one of mobile robots is less than 15 cm as shown in Fig. 5.10 (a). After $t = 30$s, the target start moving follow the white dashed line and five mobile robots can track the moving target with continuously update the distance between mobile robot and target, the *gBest* value at thats time is from robot with red line is shown in Fig. 5.10 (b). Figure 5.10 (c) show snapshots five mobile robot can still tracking and following the moving target, until it stops at the home base areas followed by five mobile robots as shown in Fig. 5.10 (d).

Moving actions of each mobile robot in an unknown environment to target areas shown in Figs. 5.5, 5.6, 5.7, 5.8, 5.9 and 5.10. The mobile robot will move from the current position to the one of the target with its velocity. Each mobile robot moves to the nearest selected target using PSO method and cooperative control algorithm.

The results are presented in the following to demonstrate the validity of the proposed the cooperative control system using PSO for tracking target.

# CHAPTER 6

# Conclusion

In this research, we proposed cooperative control system of multiple mobile robots using particle swarm optimization (PSO) for tracking target. We regard the problem of tracking target as an optimization problem and solve it with PSO. A conceptual overview of the PSO algorithm and its parameters selection strategies, geometrical illustration and neighborhood topology, advantages and disadvantages of PSO, and mathematical explanation.

The actual implementation of an efficient algorithm like PSO is required when robots need to avoid the randomly placed obstacles in unknown environment and reach the target point. We treats of the cooperative control of multiple mobile robots for tracking target. The control system should have an effective motion to reach one or more different position of target.

The control system have an effective motion to reach one or more different position of target. We have basic information about the environment like the position of each mobile robot and relative distance between mobile robots and target.

The positions of globally best particle in each iterative are selected, and reached by the robot in sequence. Moreover, the positions of obstacles are detected by the robot sensor and applied to update the information about the environment. The optimal path is generated by the robot reaches its target by using PSO algorithm.

We developed a cooperative control system with PSO and obstacle avoidance algorithm in each mobile robot. The mobile robots deployed in an unknown environment reaching and tracking their target by avoiding obstacles encountered on their way. The mobile robots are produced to implement a cooperative control system for tracking targets in unknown environments using PSO and obstacle avoidance method of the size of the group. We use video cameras over the environment to get coordinate position of each mobile robot and target area. And the robots only have the information about the relative distance to the targets area and position of each mobile robot.

The results of the experiment demonstrated that the proposed cooperative control system of multiple mobile robot with limited sensor and information using PSO for tracking target in unknown environment with obstacle.

# ◇ ◇ ◇ REFERENCES ◇ ◇ ◇

[1] S. H. Strogatz. SYNC: The Emerging Science of Spontaneous Order. Hyperion Press,New York, 2003.

[2] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin. Effective leadership and decisionmaking in animal groups on the move. Nature, 433:513516, 2005.

[3] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. Computer Graphics, 21(4):2534, 1987.

[4] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. Proc. IEEE, 95(1):215233, 2007.

[5] W. Ren and R. W. Beard. Distributed Consensus in Multi-Vehicle Cooperative Control. Springer, London, UK, 2008.

[6] R. M. Murray, editor. Control in an Information Rich World: Report of the Panel on Future Directions in Control, Dynamics, and Systems. SIAM, 2002.

[7] R. Srikant. The Mathematics of Internet Congestion Control. Birkhauser, Boston, USA, 2004.

[8] D. Helbing. Traffic and related self-driven many-particle systems. Reviews of Modern Physics, 73(4):10671141, 2001.

[9] M. Pavella and P. G. Murthy, editors. Transient Stability of Power Systems: Theory and Practice. John Wiley and Sons, West Sussex, UK, 1994.

[10] P. Miller. Swarm theory. National Geographic, 7:126147, 2007a.

[11] Kashif Zafar, Shahzad Badar Qazi, A. Rauf Baig, "Mine Detection and Route Planning in Military Warfare using Multi Agent Systems," proceedings of the 30th Annual International Computer Software and Applications Conference, volume 2, pp. 327-332, September 2006.

[12] G. Bekey, "Autonomous Robots," MIT Press, pp. 391-439, 2005.

[13] H. Takeda, Z. D. Wang, Y. Hirara and K. Kosuge, "Load sharing algorithm for transporting an object by two mobile robots in coordination," in Proc. 2004 Intl. Conf. On Intelligent Mechatronics and Automation, pp. 374-378, Aug. 2004.

[14] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," IEEE Trans. Robotics and Automation, vol. 14, no. 6, pp. 926-939, Dec. 1998.

[15] J. Jongusuk and T. Mita, "Tracking control of multiple mobile robots: a case study of inter-robot collision-free problem," in Proc. 2001 IEEE Intl. Conf. On Robotics and Automation, pp. 2885-2890, May 2001.

[16] L. Smith, G. Venayagamoorthy, P. Holloway, "Obstacle Avoidance in Collective Robotic Search Using Particle Swarm Optimization," in Proc. IEEE Swarm Intelligence. Symp., Indianapolis, IN, May 12-14, 2006.

[17] J. Pugh, A. Martinoli, "Inspiring and Modeling Multi-Robot Search with Particle Swarm Optimization," In: Proc. of the 4th IEEE Swarm Int'l. Symposium, Honolulu, HI, USA, 2007.

[18] S. Doctor, G.K. Venayagamoorthy and V. Gudise, "Optimal PSO for Collective Robotic Search Applications," IEEE Congress on Evolutionary Computations, Portland OR, USA, pp. 1390-1395, June 19-23, 2004.

[19] T. Hsiang, E. Arkin, M. Bender, S. Fekete, J. Mitchell, "Online Dispersion Algorithms for Swarms of Robots," ACM SoCG 2003, San Diego, California, USA, June 8-10, 2003.

[20] Min Gyu Park, Jae Hyun Jeon, Min Cheol Lee, "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing," ISIE 2001, vol. 3, pp. 1530-1535, 2001.

[21] Ming Chen, Zhengwei Yao, "Classification Techniques of Neural Networks Using Improved Genetic Algorithms," WGEC 2008, pp.115-119, September 2008.

[22] H. Suzuki, T. Yasuno, S. Urushihara, E. Yasuno and A. Kuwahara : Control Characteristics of Cooperative Conveyance Control for Multiple-Mobile-Robot System Using Complex-Valued Neural Network and Indirect Cooperative Scheme, Proceedings of SICE Annual Conference 2010, pp.3581-3585 , 18-21 Aug. 2010.

[23] Howard Li, Simon X. Yang and Yevgen Biletskiy, "Neural Network Based Path Planning for A Multi-Robot System with Moving Obstacles," 4th IEEE Conference on Automation Science and Engineering Key Bridge Marriott, Washington DC, USA, August 23-26, 2008.

[24] Bin Lei, Wenfeng Li, Fan Zhang, "Flocking algorithm for multirobots formation control with a target steering agent," SMC 2008, pp. 3536-3541, October 2008.

[25] J. Kennedy and R. Eberhart: Particle Swarm Optimization, Proc. IEEE Int. Conf. Neural Network., Vol.4, pp. 1942-1948, 1995.

[26] E. Masehian and D. Sedighizadeh: Multi-objective robot motion planning using a particle swarm optimization model, Journal of Zheijang University, vol. 11, pp. 607-619, Aug 2010.

[27] E. Ferrante, A. E. Turgut, C. Huepe, A. Stranieri, C. Pinciroli, and M. Dorigo: Self-organized flocking with a mobile robot swarm: a novel motion control method, Adaptive Behavior, vol. 20, pp. 460-477, Dec 2012.

[28] Q. Lu and Q.-L. Han, "A probability particle swarm optimizer with information-sharing mechanism for odor source localization," in Proceedings of the 18th World Congress of the International Federation on Automatic Control, Milano, Italy, 2011.

[29] D.A Prasetya and T. Yasuno : Cooperative Control of Multiple Mobile Robot Using Particle Swarm Optimization For Tracking Two Passive Target, Proceedings of SICE Annual Conference(SICE)2012,pp.1751-1754 , 20-23 Aug. 2012.

[30] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, 1995, pp. 3943.

[31] J. Kennedy, The particle swarm: Social adaptation of knowledge, in: Proceedings of the IEEE International Conference on Evolutionary Computation, Indianapolis, USA, 1997, pp. 303308.

[32] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: Proceedings of the Seventh Annual Conference on Evolutionary Programming, New York, USA, 1998, pp.591600.

[33] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE Congress on Evolutionary Computation, Piscataway, USA, 1998, pp. 6973.

[34] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE Press, 1999, pp. 19451950.

[35] A. Carlisle, G. Dozier, An off-the-self pso, in: Proceedings of the Workshop on Particle Swarm Optimization, Indianapolis, USA, 2001.

[36] T. Beielstein, K.E. Parsopoulos, M.N. Vrahatis, Tuning pso parameters through sensitivity analysis, in: Technical Report, Reihe Computational Intelligence CI 124/02, Department of Computer Science, University of Dortmund, 2002.

[37] S. Naka, T. Genji, T. Yura, Y. Fukuyama, A hybrid particle swarm optimization for distribution state estimation, IEEE Transactions on Power Systems 18 (1) (2003) 6068.

[38] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, San Diego, USA, 2000, pp. 8488.

[39] E. Ozcan, C.K. Mohan, Analysis of a simple particle swarm optimization system, in: Intelligent Engineering Systems through Artificial Neural Networks, 1998, pp. 253258.

[40] E. Ozcan, C.K. Mohan, Particle swarm optimization: Surfing the waves, in: Proceedings of the IEEE Congress on Evolutionary Computation, Washington, DC, USA, 1999.

[41] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1)(2002) 5873.

[42] F. van den Bergh, An Analysis of Particle Swarm Optimizers, PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[43] I.C. Trelea, The particle swarm optimization algorithm: Convergence analysis and parameter selection, Information Processing Letters 85 (6) (2003) 317325.

[44] K. Yasuda, A. Ide, N. Iwasaki, Adaptive particle swarm optimization, in: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 2003, pp. 15541559.

[45] Y. Zheng, L. Ma, L. Zhang, J. Qian, On the convergence analysis and parameter selection in particle swarm optimization, in: Proceedings of the International Conference on Machine Learning and Cybernetics, 2003, pp. 18021807.

[46] M. Meissner, M. Schmuker, and G. Schneider, "Optimized particle swarm optimization (OPSO) and its application to artificial neural network training," BMC Bioinformatics, vol. 7, 10 March, 2006.

[47] Y. Zheng, L. Ma, L. Zhang, and J. Qian, "On the convergence analysis and parameter selection in particle swarm optimization," in Proceedings of the Second International Conference on Machine Learning and Cybernetics, 2003, pp. 1802-1807.

[48] S. Naka and Y. Fukuyama, "Practical distribution state estimation using hybrid particle swarm optimization," IEEE Power Engineering Society Winter Meeting, vol. 2, pp. 815-820, Jan. 2001.

[49] A. P. Engelbrecht, Computational Intelligence: An Introduction, 2 ed.: John Wiley and Sons, 2007.

[50] B. Liu, L. Wang, Y. Jin, F. Tang, and D. Huang, "An improved particle swarm optimization combined with chaos," Chaos, Solition and Fractals, vol. 25, pp. 1261-1271, 2005.

[51] Latombe .J. C, 1990, Robot motion planning: Springer Verlag.

[52] Wang. Y, et al.,2000, "Two novel approaches for unmanned underwater vehicle path planning: constrained optimisation and semi-infinite constrained optimisation," Robotica, vol. 18, pp. 123-14.

[53] Shi .P and Y. Zhao,2009, "An efficient path planning algorithm for mobile robot using improved potential field," in IEEE International Conference on Robotics and Biomimetics, pp. 1704-1708.

[54] Ghorbani. A , et al.,2009, "Using Genetic Algorithm for a Mobile Robot Path Planning," in International Conference on Future Computer and Communication, , pp. 164-166.

[55] Sugawara. k, et al.,2004, "Foraging behavior of interacting robots with virtual pheromone," in Proceedings of the International Conference on Intelligent Robots and Systems, pp. 3074-3079 vol. 3.

[56] Qu. H, et al.,2009, "Real-time robot path planning based on a modified pulse-coupled neural network model," Neural Networks, IEEE Transactions on, vol. 20, pp. 1724-1739.

[57] Chen. X and Y. Li, 2006, "Smooth path planning of a mobile robot using stochastic particle swarm optimization," in Proceedings of the IEEE International Conference on Mechatronics and Automation, pp. 1722-1727.

[58] Qin. Y. Q, et al., 2004, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in Proceedings of International Conference on Machine Learning and Cybernetics, pp. 2473-2478 vol. 4.

[59] Hao. Y, et al.,2007 "Real-Time Obstacle Avoidance Method based on Polar Coordination Particle Swarm Optimization in Dynamic Environment," in IEEE Conference on Industrial Electronics and Applications, pp. 1612-1617.

[60] Wang. L, et al.,2006 "Obstacle-avoidance path planning for soccer robots using particle swarm optimization," in IEEE International Conference on Robotics and Biomimetics, pp. 1233-1238.

[61] Karimi.J and pourtakdoust.H, " Optimal maneuver-based motion planning over terrain and threats using a dynamic hybrid PSO algorithm", Aerospace Science and Technology Journal, Elsevier,2012

[62] Wei Ren, Randal W. Beard, and Ella M. Atkins. A survey of consensus problems in multi-agent coordination. In ACC05: Proceedings of the 2005 American Control Conference, volume 3, pages 18591864, Jun. 2005.

[63] Richard M. Murray. Recent research in cooperative control of multivehicle systems. Journal of Dynamic Systems, Measurement, and Control, 129(5):571583, Sept. 2007.

[64] Craig Reynolds. Boids (flocks, herds, and schools: a distributed behavioral model), Sept. 2001.

[65] Edmund Eisenberg and David Gale. Consensus of subjective probabilities: The pari-mutuel method. The Annals of Mathematical Statistics, 30(1):165168, Mar. 1959.

[66] Torsten Norvig. Consensus of subjective probabilities: A convergence theorem. The Annals of Mathematical Statistics, 38(1):221225, Feb. 1967.

[67] Robert L. Winkler. The consensus of subjective probability distributions. Management Science,15(2):B61 烹 75, Oct. 1968.

[68] Morris H. DeGroot. Reaching a consensus. Journal of the American Statistical Association, 69 (345):118121, Mar. 1974.

[69] Ren C. Luo and Michael G. Kay. Multisensor integration and fusion in intelligent systems. IEEE Transactions on Systems, Man, and Cybernetics, 19(5):901931, Sep./Oct. 1989.

[70] Jon A. Benediktsson and Philip H. Swain. Consensus theoretic classification methods. IEEE Transactions on Systems, Man, and Cybernetics, 22(4):688704, Jul./Aug. 1992.

[71] Deborah Estrin, Lewis D. Girod, Gregory J. Pottie, and Mani Srivastava. Instrumenting the world with wireless sensor networks. In ICASSP '01: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, volume 4, pages 20332036, Salt Lake City, UT,USA, May 2001.

[72] Reza Olfati-Saber and Jeff S. Shamma. Consensus filters for sensor networks and distributed sensor fusion. In CDC-ECC 05: Proceedings of the Joint 44th IEEE Conference on Decision and Control and the 2005 European Control Conference, pages 66986703, Dec. 2005.

[73] Susan C. Weller and N. Clay Mann. Assessing rater performance without a 堵 old standard using consensus theory. Medical Decision Making, 17(1):7179, Feb. 1997.

[74] Bernard C. Levy, David A. Casta n, George C. Verghese, and Alan S. Willsky. A scattering framework for decentralized estimation problems. Automatica, 19(4):373384, 1983.

[75] Arthur G. O. Mutambara. Decentralized Estimation and Control for Multisensor Systems. CRC Press, Inc., Boca Raton, FL, USA, 1998.

[76] Vijay Gupta. Distributed Estimation and Control in Networked Systems. Ph.D. thesis, Control and Dynamical Systems, California Institute of Technology, Pasadena, CA, USA, Jun. 2006.

[77] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In SIGGRAPH'87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, pages 2534, Anaheim, CA, USA, Jul. 1987.

[78] Tamas Vicsek, Andras Czirok, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel type of phase transition in a system of self-driven particles. Physical Review Letters, 75(6):1226.1229,Aug. 1995.

[79] Vivek Borkar and Pravin P. Varaiya. Asymptotic agreement in distributed estimation. IEEE Transactions on Automatic Control, 27(3):650655, Jun. 1982.

# APPENDIX A

# Hardware

## A.1 AVR controller

To control the mobile robot, we use AVR AVR ATmega88P micro-controller as shown in figure. A.1. The AVR is one of the popular micro-controller families to use on chip flash memory include that ROM, EPROM, and EEPROM. It can process the speed command from the PC to drive wheel, battery voltage minitoring and we have been communicating with the control PC via wireless communication. The parameters are shown in table. A.1.

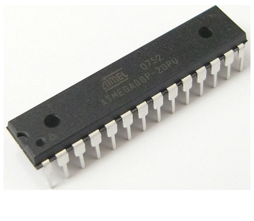


Figure A.1: AVR micro-controller

Table A.1: AVR parameter sets

| | Unit | Value |
|---|---|---|
| Pattern number | | ATmega88 |
| Pin | | 28 |
| Operating Voltage | V | 1.8 5 |
| Max. Operating Frequency | MHz | 20 |
| EEPROM | Bytes | 512 |
| CPU | bit | 16 |
| Flash ROM | KByte | 8 |
| RAM | KByte | 1 |
| Serial Interface | ch | 1 |
| A/D Converter | ch | 6 |

By using duty radio, we can adjust the PWM to drive the DC motor for the mobile robot.

## A.2 Wireless communication unit

The wireless communication with the mobile robot and control for PC, we use the XBee from Digi's company. XBee is a radio capable for transmitting and receiving radio signal by ZigBee with standard UART signal communication and also low power consumption. The mobile robot with low power consumption and low cost are needed. Figure A.2 shows the appearance of the XBee, Specification of Xbee as shown in Table A.2.

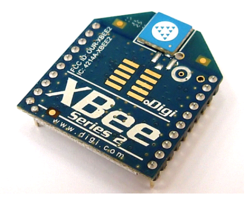


Figure A.2: Xbee wireless communication unit

Table A.2: Xbee parameter sets

| Name | Xbee |
|---|---|
| Wireless standart | IEEE 802.15.4 (ZigBee) |
| Wireless communication speed | 250kbps |
| Transmission power output | 1mW |
| Serial interface | 3.3V CMOS UART |
| Serial communication speed | 1200 - 250kbps |
| Power-supply voltage | 2.8 - 3.4 V |
| Dimensions | 24.38x32.94x7.33mm |

## A.3 Servo DC Motor

The actuator of the mobile robot, we use the RC servo motor PIC+F/BB/F of GWS servo's company. Figure A.3 the appearance of the servo motor. Specification of DC motor as shown in Table A.3.

RC servo motor, DC motor which can be controlled by the angular position command by pulse width Is in, many minutes as actuator small humanoid robot and traditional radio control It is used in the field. RC servo motor potentiometer for the rotation angle measurement inside Pcs in addition data, the position feedback control system, it has a built-in motor driver and reduction gear.

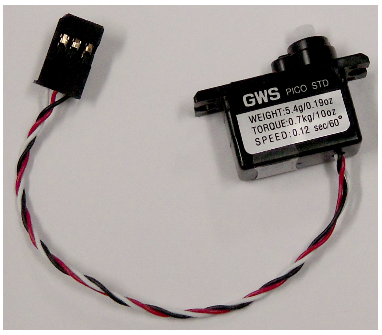


Figure A.3: RC servo motor

Table A.3: RC servo motor parameter

| Name | PIC+F/BB/F |
|---|---|
| Speed | 0.12s / 60deg(4.8V) |
| Torque | 0.79kg . cm (4.8V) |
| Power supply voltage | 4.8V |
| Dimensions | 22.8x9.5x19.8mm |

## A.4 IR distance sensor

Sharp GP2Y0D805Z0F is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector) , IRED (infrared emitting diode) and signal processing circuit.

The output voltage of this sensor stays high in case an object exists in the specified distance range. So this sensor can also be used as proximity sensor. Figure A.4 the appearance of the servo motor. Specification of DC motor as shown in Table A.4.

Figure A.4: Distance sensor

Table A.4: Distance sensor parameter

| Name | Sharp GP2Y0D805Z0F |
|---|---|
| Detecting fix distance | 50mm |
| Power Consumption | 5mA |
| Power supply voltage | 2.7-6.2V |
| Dimensions | 13.6x7x7.95mm |

The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method.

# ◇ ◇ ◇ Publications ◇ ◇ ◇

1. Dwi Arman Prasetya, Takashi Yasuno, Hiroshi Suzuki and Akinobu Kuwahara : "Cooperative Control System of Multiple Mobile Robot Using Particle Swarm Optimization With Obstacle Avoidance For Tracking Target" Journal Signal Processing, Vol., Page, June 25,2013. Accepted
2. Dwi Arman Prasetya and Takashi Yasuno : "Cooperative Control of Multiple Mobile Robot Using Particle Swarm Optimization For Tracking Two Passive Target" SICE Annual Conference 2012, August, 2012, Published.
3. Yong Zhang, Takashi Yasuno, Dwi Arman Prasetya : "Adaptive Walking for Quadruped Robot on Irregular Terrain by Using CPG Network" SICE Annual Conference 2013, Accepted.
4. Dwi Arman Prasetya, Takashi Yasuno, Yong Zhang : "Cooperative Tracking Control of Multiple Mobile Robot for Moving Target Using Particle Swarm Optimization" SICE Annual Conference 2013, Accepted.

# Acknowledgments

In the name of Allah, the Most Gracious and the Most Merciful.

Alhamdulillah, all praise is due to Allah for the strengths and His blessing in completing this thesis. I am thankful to Allah, who supplied me with the courage, the guidance, and the love to complete this research. Also, I cannot forget the ideal man of the world and most respectable personality for whom Allah created the whole universe, Prophet Mohammed (Peace Be Upon Him).

Special appreciation goes to my supervisor, associate Professor Takashi Yasuno. I am very grateful for him since he gave me the chance to work and study on an interesting and practical topic on his laboratory, and for his supervision and constant support. His invaluable help of constructive comments and suggestions throughout the experimental and thesis works have contributed to the success of this research.

Special thanks go to Professor Yoshifumi Nishio and Professor Masaki Hashizume who reviewed my thesis in advance and gave me his comments to improve my defense presentation. Many thanks to my other thesis committee members Professor Takashi Shimamoto and associate Professor Masahide Hojo for their involvement and the time they spared to review this thesis and for their comments to improve this thesis. My acknowledgement also goes to the technician staff of Department of Electrical and Electronic Engineering Tokushima University Akinobu Kuwahara for the co-operations.

I would like to express my appreciation to assistant Professor Hiroshi Suzuki for helping me many things in this research and also thank you for your mobile robot. I am very thankful to Akinori Tsuji sensei for support me about my research and give me excellence idea. I am also very thankful to Takahiro Kitajima and Zhang Yong for taking his time to help and support me about everything, specially for ”Kitajima Shop” and ”make story” for my paper. I will always remember you guys !!!. And Thank you to all Yasuno Lab. member for friendship, support, and memories.

I would like to express my appreciation to Ministry of national education Republic of Indonesia, specially DIKTI for support my study in Japan. I am also very thankful to the KDDI Foundation for the financial support.

Last but not least, my deepest gratitude goes to my beloved parents; Mr. Moch. Chomsun and Mrs. Endang H for their constant support, prayers and best wishes. They uplifted my morale whenever I needed. My sister, Ika and my brother, Nandoz for their endless love, prayers and encouragement. Also not forgetting my nephew, Putri, Lia and Umar for accompany me on skype and for their love and care. I would say thank you to Nina Soelfira for your support and care during my study in Japan. I also say thank you to Ima for support me during my critical time for my study.

I would like to say "Terima Kasih Banyak" to Tim huru hara, Lalala Yeyeye (Rima, Irene, Hotimah), Indonesian student association (PPIJ Tokushima), Ms. Sakabe , Miss. Miyoshi, English Chat Room Community, Mr. BuBu , Indonesia Yuko Kyokai, Mrs. Ayin, Mr. Widianto, Mrs. Sato, My Cute Honda Life, and car team (Zhang Yong and Miss Kim) for all support and nice memories.

For every one of you, I can only pray that God will bestow upon you a blessed life. Terima Kasih....

August 19, 2013
Tokushima
-Arman-