Swarm-Based Cooperative Control for a Group of Mobile Robots

Q. P. HA and N. M. KWOK
ARC Centre for Autonomous Systems, Faculty of Engineering
University of Technology, Sydney
PO Box 123, Broadway NSW 2007
AUSTRALIA
{quangha,ngai.kwok}@eng.uts.edu.au http://www.cas.edu.au

Abstract: **The accomplishment of some robotic tasks requires the coordination of a group of mobile robots, which generally outperform single robots operating independently. This chapter is devoted to the control design for deployment of multiple robots into desired geometric patterns or formations. First, a generic control structure is proposed as an unified approach, making use of its flexibility for various formation configurations while mitigating singularities. Swarm intelligence is then adopted in order to obtain near-optimal settings of the control parameters. Here, the cooperative control of the robots is formulated as an optimization problem, where the particle swarm optimization (PSO) algorithm is applied to satisfy the multi-objective goal of formation establishment and collision avoidance. Numerical results and a remark on experiments are included to illustrate the effectiveness of the proposed framework for a number of formation types.**

Key-Words: **mobile robots, cooperative control, formation, particle swarm optimization.**

1 INTRODUCTION

The control of a group of autonomous vehicles or mobile robots into coordinated operations has a lot of advantages over the deployment of single robots. Potential applications span widely from transportation [1], surveillance, agriculture [2], defense, health-care and to many others. Moreover, the application of robotics in automating the construction process may become an important component in the future development of construction technologies including material handling and delivery, to name a few [3]. Mobile robots are very attractive in these applications and, in particular, multiple robots in cooperation are anticipated to outperform independently-operating robots in terms of flexibility and fault tolerance. Coordination or formation of multiple autonomous robots, therefore, has recently arisen as an active research area in mobile robotics. In this chapter, the focus will be on the control design for robotic formation, which is defined as the coordination of a group of mobile robots to get into and maintain a desired pattern with a certain geometric shape.

One of the critical issues in deploying mobile robots rests on their navigational abilities [4], specifically, with severe spatial constraints and this naturally leads to the need for formation control strategies [5]. In this regard, virtual robot tracking [6], hybrid control [7] and fuzzy logic [8] methods have been applied. The former approaches are theoretically rigorous but generally require precise model descriptions and special considerations may be needed to account for control singularities [9][10][11][12]. On the other hand, the latter techniques require expert design knowledge and controllers are mostly designed in a problem-dependent manner. In general, a configuration-flexible design approach would be a an attractive solution.

The multi-robot coordination problem is concerned with the control or steering of the robots into desired geometrical shapes, e.g. a line, column, wedge, diamond or others. In the context of robotic formation, control approaches can be broadly classified into three categories: behavioral [13], virtual structure [14] and leader-follower [15]. In behavioral control approaches, primitives or reactive behaviors are defined for the vehicles.

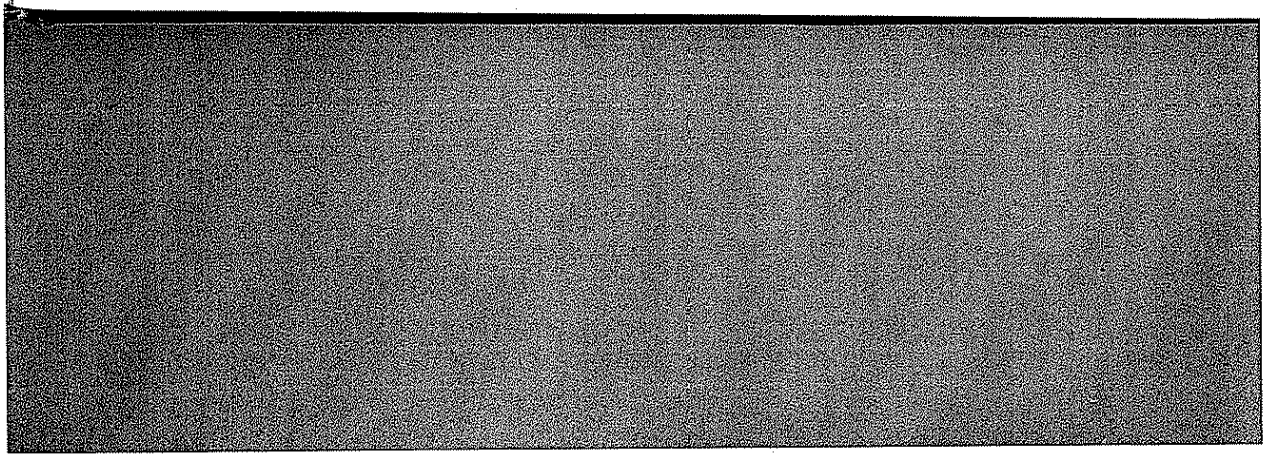
A drive command is generated by aggregating a collection of weighted primitives. However, the determination of behaviors, the selection of their number and appropriate weights for them are issues depending on the designer's experience. The virtual structural control method treats the entire formation as a single or virtual entity. The desired dynamics of the structure are then translated into motion control for individual vehicles. The control of the virtual structure can be realized by a high-level controller where low-level motion controllers, within each robot, can be implemented by using a decentralized strategy [16]. In leader-follower approaches, one or more robots are selected as leaders while the others are assigned as followers and are driven to track the leaders. Desired formations can be considered as established after the followers have tracked their assigned leaders in a hierarchy. Although this method is well-founded, it may need a deliberative motion planning strategy [17], in particular when involving a large number of robots. Moreover, some enhancements need to be incorporated into the formation control design in order to alleviate the difficulties imposed from control constraints. Otherwise, some formation pattern, e.g., a line formation, cannot be established.

In the regard of individual robots, the cooperative motion of a group of mobile robots is attributed from designing appropriate paths to be followed by each robot. Various methodologies for robotic path planning, such as soft-computing and evolutionary computation have drawn the attention of researchers in recent years. For example, a neural network, applying the self-learning principle, was employed in the floor coverage problem scenario [18] in construction, but for just a single robot. In [19], an algorithm was proposed, inspired from natural ant societies for the enumeration of robots in accomplishing a cooperative task. The multi-agent concept, implementing genetic algorithms (GA), was applied in [20] for the roadside following problem in vehicle navigation. In essence, vehicles are treated as living species evolving by adaptation to natural selections imposed by the constraints from the kerb boundaries. However, one of the hurdles in applying the GA is the determination of the control parameters, e.g., selection schemes and crossover/mutation probabilities, during the algorithmic design stage.

From an alternative point of view, the motion of mobile robots can be treated as an aggregation of a group of living species evolving under social interactions, e.g. bird flocks or fish schools. Inspired by this natural phenomenon, the particle swarm optimization (PSO) algorithm [21], in the evolutionary computation family, was developed as an efficient solution searching method for a variety of optimization problems. Applications of PSO can be found in a wide domain in science and engineering practices. For example, a cantilevered beam was designed by using the PSO [22] in a structural application. Setting PID controller parameters was reported in [23] to achieve robustness against adverse operating conditions. The project scheduling problem with resource constraints was tackled in [24] where swarm intelligence led to promising results. Perhaps most remarkable is the application of the PSO for multi-objective optimization problems, see e.g., [25]. The successes reported are largely attributed to the implementation simplicity and flexibility of swarm algorithms. In the field of mobile robotics, the PSO has been applied for path planning [26], navigation control [27] and recently for mobile robot coordination [28].

The configuration structure of the group of mobile robots corresponding to different formation types is however foreshadowed to affect the formation process. This chapter presents a near-optimal design methodology which uses the PSO algorithm to derive the control parameters of a generic controller for configuration-changeable formations within the leader-follower framework. The ultimate objective is a high-performance and collision-free control strategy for various formation patterns. Our rational rests on the possibility of incorporating the control constraints into the fitness function and the effectiveness of PSO in solving a multi-objective optimization problem. In fact, the swarm-based searching for feasible drive commands for mobile robots to establish desired formations has shown its simplicity and flexibility, allowing for incorporation of reactive collision avoidance schemes.

The remainder of this chapter is organized as follows. In Section 2, a generic control structure is presented and its various combination possibilities discussed on the basis of formation performance. The background of particle swarm optimization algorithm and swarm-based control is reviewed in Section 3. Section 4 presents the design of the formation controllers using swarm intelligence for multiple mobile robots. The proposed design framework also includes discussion on attempts to tackle collision avoidance and dead-lock release. Numerical examples and some real-time



test results are given in Section 5 to illustrate the effectiveness of the proposed approaches. Finally, a conclusion is drawn in Section 6.

2 FORMATION CONTROL STRUCTURE

The objective of robotic formation control is to steer a group of robots into and maintain desirable moving patterns of the group. From the leader-follower strategy, let there be a *leader* robot positioned in (x_i, y_i, θ_i) in a world coordinate frame, where the orientation is referred to the x-axis, also let there be a number of *follower* robots at (x_j, y_j, θ_j), where $i, j = 1, \cdots, N$ are the indices for N robots of the group, as shown in Figure 2.1.

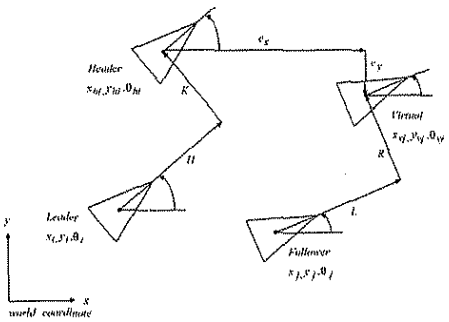


Figure 2.1: Generic formation notation.

2.1 Mobile Robot Model

Assuming that the robots are moving on a flat terrain, their motion models are given by:

$$\begin{aligned} \dot{x}_k &= v_k \cos\theta_k \\ \dot{y}_k &= v_k \sin\theta_k \\ \dot{\theta}_k &= \omega_k, \end{aligned} \tag{2.1}$$

where (x, y) are the Cartesian coordinates, θ is the orientation of the robot, the subscript (k) is an index of the robot.

This model implies that the motion of the robot satisfies the pure-rolling condition:

$$\dot{x}_k \cos\theta_k + \dot{y}_k \sin\theta_k = v_k, \tag{2.2}$$

and the non-slip condition:

$$\dot{x}_k \sin\theta_k - \dot{y}_k \cos\theta_k = 0. \tag{2.3}$$

2.2 Generic Structure

There are a number of formation types available in the literature [10][6][29], however the configuration parameters appear in an incoherent manner. Here, a generic structure for leader-follower formation control is considered by appropriately assigning the configuration parameters to result in various formation types. Let a *header* robot be placed at a position with respect to the leader as:

$$\begin{aligned} x_{hi} &= x_i + H\cos\theta_i - K\sin\theta_i \\ y_{hi} &= y_i + H\sin\theta_i + K\cos\theta_i. \end{aligned} \tag{2.4}$$

Similarly, *virtual* robots are denoted as

$$\begin{aligned} x_{vj} &= x_j + L\cos\theta_j - R\sin\theta_j \\ y_{vj} &= y_j + L\sin\theta_j + R\cos\theta_j. \end{aligned} \tag{2.5}$$

In order to configure a specific formation type, separations H, K, L, R need to be determined accordingly depending on the task requirements. Furthermore, the tracking errors are denoted below as:

$$e_x = x_{vj} - x_{hi},\ e_y = y_{vj} - y_{hi},\ e_\theta = \theta_j - \theta_i. \tag{2.6}$$

The formation control objective is achieved by deriving appropriate commands, velocity v_j and turn-rate ω_j to drive the followers $j = 1, \cdots, N; j \neq i$ such that the tracking errors approach and maintain at zero. To this end, the error dynamics are obtained as:

$$\begin{aligned} e_x = {} & x_j + L\cos\theta_j - R\sin\theta_j \\ & - x_i - H\cos\theta_i + K\sin\theta_i \\ e_y = {} & y_j + L\sin\theta_j + R\cos\theta_j \\ & - y_i - H\sin\theta_i - K\cos\theta_i, \end{aligned} \tag{2.7}$$

which, by differentiation, become

$$\begin{aligned} \dot{e}_x = {} & v_j\cos\theta_j - L\omega_j\sin\theta_j - R\omega_j\cos\theta_j \\ & - v_i\cos\theta_i + H\omega_i\sin\theta_i + K\omega_i\cos\theta_i \\ \dot{e}_y = {} & v_j\sin\theta_j + L\omega_j\cos\theta_j - R\omega_j\sin\theta_j \\ & - v_i\sin\theta_i - H\omega_i\cos\theta_i + K\omega_i\sin\theta_i. \end{aligned} \tag{2.8}$$

The dynamics can be further written in matrix form as:

$$\dot{\mathbf{e}} = \mathbf{A}\mathbf{u}_j + \mathbf{B}\mathbf{u}_i, \tag{2.9}$$

where

$$\dot{\mathbf{e}} = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \end{bmatrix},\ \mathbf{A} = \begin{bmatrix} \cos\theta_j & -L\sin\theta_j - R\cos\theta_j \\ \sin\theta_j & L\cos\theta_j - R\sin\theta_j \end{bmatrix},$$
$$\mathbf{B} = \begin{bmatrix} -\cos\theta_i & H\sin\theta_i + K\cos\theta_i \\ -\sin\theta_i & -H\cos\theta_i + K\sin\theta_i \end{bmatrix},$$
$$\mathbf{u}_j = \begin{bmatrix} v_j \\ \omega_j \end{bmatrix},\ \mathbf{u}_i = \begin{bmatrix} v_i \\ \omega_i \end{bmatrix}.$$

The drive commands are generated by specifying the error attenuation rates at ($\lambda_x > 0, \lambda_y > 0$) such that the errors approach exponentially to zero:

$$\begin{aligned} e_x(t) &= e_x(0)\exp(-\lambda_x t) \\ e_y(t) &= e_y(0)\exp(-\lambda_y t), \end{aligned} \tag{2.10}$$

where $e_x(0)$ and $e_y(0)$ are, respectively, the initial errors at time $t = 0$. Using this approach with the attenuation rate a matrix denoted as $\Lambda = diag(\lambda_x, \lambda_y)$, then

$$\mathbf{u}_j = -\mathbf{A}^{-1}(\mathbf{B}\mathbf{u}_i + \Lambda\mathbf{e}). \tag{2.11}$$

The drive commands are

$$\begin{aligned} v_j = \frac{R}{L}(&\lambda_x e_x \sin\theta_j - \lambda_y e_y \cos\theta_j \\ &- (v_i - \omega_i K)\sin e_\theta + \omega_i H \cos e_\theta) \\ \omega_j = \frac{1}{L}(&\lambda_x e_x \sin\theta_j - \lambda_y e_y \cos\theta_j \\ &- \lambda_x e_x \cos\theta_j - \lambda_y e_y \cos\theta_j \\ &+ (v_i - \omega_i K)\cos e_\theta + \omega_i H \sin e_\theta \\ &- (v_i - \omega_i K)\sin e_\theta + \omega_i H \cos e_\theta), \end{aligned} \tag{2.12}$$

where the linear velocity is related to the angular velocity by

$$v_j = R\omega_j + \bar{v}_j, \tag{2.13}$$

in which

$$\begin{aligned} \bar{v}_j = &-\lambda_x e_x \cos\theta_j - \lambda_y e_y \cos\theta_j \\ &+ (v_i - \omega_i K)\cos e_\theta + \omega_i H \sin e_\theta. \end{aligned}$$

2.2.1 Treatment of Singularities

An inspection of the drive commands (2.12) reveals that a singularity would occur if the configuration is setup such that L is zero. That is, the commands tend to infinity and hence are practically infeasible in real-world systems.

Here, it is observed that $L = 0$ corresponds to a *line* pattern. As virtual clearance H is a free-parameter for a displaced line from the leader, by making use of the redundance of parameter H, forming a line pattern then requires just $H \neq 0$.

Note that parameter H takes effect on the drive commands through the product term $\omega_i H$ and is interpreted as a fictitious orthogonal velocity determined by the magnitude of the leader turn-rate ω_i. Furthermore, this velocity is resolved into two more components with respect to the angular error, namely $\sin e_\theta$ and $\cos e_\theta$.

To this end, the incorporation of parameter H is useful in order for:

1. the drive command singularities to be mitigated, and
2. the leader turn-rate and angular tracking error to be taken into account.

2.2.2 Zero Dynamics

Since the dimension of the control commands ($\mathbf{u}_j \in \mathbb{R}^2$) is one degree-of-freedom less than the system states ($\mathbf{x}_j \in \mathbb{R}^3$), the stability of θ_j is therefore determined by the zeros dynamics.

Re-writing the angular velocity command ω_j and setting it equal to the leader turn-rate ω_i, one has

$$\omega_j = \frac{J}{L}\cos(\theta_j - \varphi) = \omega_i, \tag{2.14}$$

where

$$\begin{aligned} J &= \sqrt{(\lambda_x e_x - (v_i - \omega_i K)\cos\theta_i + \omega_i H \sin\theta_i)^2 + \\ &\quad (-\lambda_y e_y - (v_i - \omega_i K)\sin\theta_i + \omega_i H \cos\theta_i)^2} \\ \varphi &= \arctan\left(\frac{\lambda_x e_x - (v_i - \omega_i K)\cos\theta_i + \omega_i H \sin\theta_i}{-\lambda_y e_y - (v_i - \omega_i K)\sin\theta_i + \omega_i H \cos\theta_i}\right). \end{aligned} \tag{2.15}$$

A solution trajectory for the follower's orientation can be found as

$$\theta_j^* = \arccos\left(\frac{\omega_i L}{J}\right) + \varphi. \tag{2.16}$$

Taking the partial differential of ω_j with respect to θ_j gives

$$\begin{aligned} \frac{\partial\omega_j}{\partial\theta_j}|_{\theta_j=\theta_j^*} &= \frac{-J}{L}\sin(\theta_j^* - \varphi) \\ &= \frac{-1}{L}\sqrt{J^2 - (\omega_i L)^2}. \end{aligned} \tag{2.17}$$

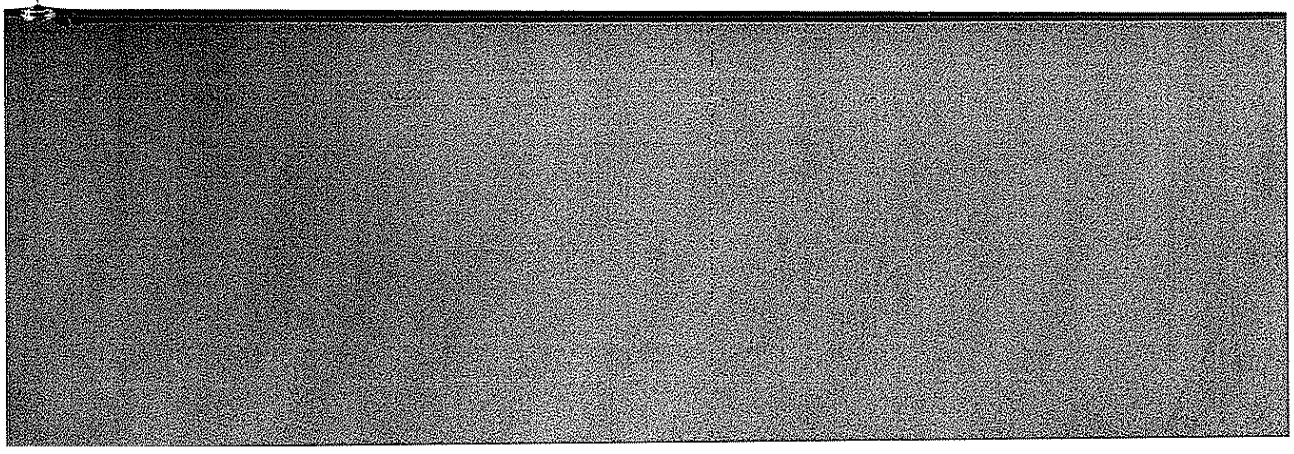
Hence stability is ensured if

$$L > 0 \text{ and } |J| \geq |\omega_i L|, \tag{2.18}$$

which is one of the constraints in the controller design.

2.3 Formation Initialization

Formation initialization is the process of deploying multiple robots from arbitrary initial locations to enter a formation pattern. A generic procedure for that is introduced in [29], but the process is time consuming. It is interesting to note here that some typical formations can be initialized by assigning appropriately configuration parameters (L, R, H, K). In particular, using this parameter setting in a *single-leader* and/or a *multi-leader* scheme with the control laws (2.12) allows



the robots to be initialized into a *line* or *column* formation, or their combination. Let there be a physical leader robot and N follower robots in the platoon then the following schemes are proposed.

2.3.1 Single-Leader: Line Formation

Referring to Figure 2.1, a line formation can be established when $H_i = const, L_j = 0$ or generally, $H_i = L_j = const$. Now, there are two possible settings for parameters K and R.

1. Set $K_i = 0,\ i = 1$ and $R_j = -jR,\ j = 1, \cdots, N$.

   There is a single header in front of the leader and the virtual robots of the followers are separated consecutively according to their indices j.

2. Alternatively, set $K_i = iK,\ i = 1, \cdots, N$ and $R_j = 0,\ j = 1, \cdots, N$.

   The virtual robots are straightly in front of the followers while the headers are displaced from the leader in scale factors according to indices i associated with the follower indices $i \leftarrow j$.

By considering (2.12), a remark can be made on the magnitudes of the commands corresponding to these schemes.

In case 1, $R_j = -jR$ and other configuration parameters are constants, a follower away from the leader (large j) will result in an increased magnitude of the linear velocity command. On the other hand, a low index follower will have a smaller velocity. Notably, nearby followers would suffer from actuator saturation while the faraway followers will suffer from a sluggish response.

For case 2, $R_j = 0$ and $K_i = iK$, the linear velocity becomes $\bar{v}_j$ and depends mainly on parameters K and H. Here, although K changes according to the followers' indices, their effects are modulated by the magnitudes of the leader turn-rate and velocity through the term $(v_i - \omega_i K)$. It can be seen that when the leader is moving on a straight line, i.e. $\omega_i = 0$, the effect of different values of K_i is diminished. Therefore, the configuration scheme in case 2 is practically preferable.

2.3.2 Single-Leader: Column Formation

For column formations, the simplest configuration is to set $K_i = R_j = 0$ and there are also two possibilities in setting parameter H_i.

1. Set $H_i = 0,\ i = 1$ and $L_j = L,\ j = 1, \cdots, N$ where $L = const$.

   Since $H_i = 0$, the headers coincide with the leader. The followers are displaced behind the leader in displacements of multiple Ls in accordance to their indices.

2. Set $H_i = L,\ i = 1$ and $L_j = (j+1)L$.

   Here the header is located in front of the leader for all followers. The followers are displaced from the leader in incremented multiples of L.

As in the line formation, these configurations also bear different effects on the control signals. For case 1, the L_j values are smaller than their counterparts in case 2, therefore, the commands are larger for given errors e_x, e_y while other conditions remained unchanged. Thus, the former case may be prone to saturation. Furthermore, since $H_i = 0$, components $\omega_i H \cos e_\theta$ and $\omega_i H \sin e_\theta$ are absent from the control.

In case 2, the drive commands are scales by L^{-1} and the commands will be much reduced for the robot with a high index, i.e. away from the leader. Also, with $H_i \neq 0$, the effects of $\omega_i H \cos e_\theta$ and $\omega_i H \sin e_\theta$ are complementary to the contributions from $(v_i - \omega_i K) \sin e_\theta$ and $(v_i - \omega_i K) \cos e_\theta$. Hence, the configuration in case 2 is preferable.

2.3.3 Multi-Leader: Line Formation

In line formations with multiple leaders, set $H_i = H = const$, $L_j = 0$, $R_j = 0$, and $K_i = K = const$ for $i, j = 1, \cdots, N$.

In this case, most parameters are kept constant and with $R_j = 0$, the linear velocity command becomes de-coupled from the follower angular velocity.

2.3.4 Multi-Leader: Column Formation

For column formations, set $K_i = H_i = R_j = 0$ and $L_j = L = const$.

Note that L_j is the only non-zero parameter and the drive commands are further de-coupled. Particularly, with $H_i = K_i = 0$, the headers coincide with the leader and the effect of ω_i is totally removed from the drive commands. Cautions have to be paid in this configuration scheme where drives are not sensitive to the leader's angular velocity.

The configuration schemes presented above have, on their own, advantages and disadvantages

based on a variety of criteria. The design of an appropriate and effective controller for formation initialization may remain a challenging task. In the following, the design will be cast into an optimization problem. In essence, a swarm-based evolutionary computing technique, the particle swarm optimization (PSO) algorithm will be adopted in the leader-follower control design.

3 PARTICLE SWARM OPTIMIZATION AND PSO-BASED FORMATION INITIALIZATION

3.1 Background

The particle swarm optimization algorithm can be viewed as a stochastic search method for solving non-deterministic optimization problems and can be described by the following expression,

$$\begin{aligned} v_{k+1}^i &= wv_k^i + c_1(g_{best,k} - x_k^i) + c_2(p_{best,k}^i - x_k^i) \\ x_{k+1}^i &= x_k^i + v_{k+1}, \end{aligned} \tag{3.1}$$

where x is the particle position in the solution space, v is the velocity of the particle movement assuming a unity time step, w is the velocity control coefficient, c_1, c_2 are the gain control coefficients, g_{best} is the global-best position, p_{best} is the position of a particular particle corresponding to the best fitness obtained so far, subscript k is the iteration index and superscript i is the particle index.

Since the pioneering work in PSO [30], the gain control coefficients c_1, c_2 have been conventionally taken as random numbers sampled from a uniform distribution, that is

$$c_1 \sim \mathcal{U}[0, c_{1,max}], \;\; c_2 \sim \mathcal{U}[0, c_{2,max}], \tag{3.2}$$

where $\mathcal{U}[\cdot]$ stands for a uniform distribution and $c_{1,max}$ is the maximum value for c_1 and $c_{2,max}$ for c_2.

3.2 Algorithm

At the start of the algorithm, the particle positions are randomly generated to cover the solution space. Their positions may be deterministically or randomly distributed, given a pre-defined number of particles. In general, a small number reduces the computational load at the expense of extended iterations required to obtain the optimum (but the optimal solution is not known *a priori*). The particle velocity v_0^i can also be set randomly or simply assigned as zeros. A problem dependent fitness function is evaluated and a fitness is assigned to each particle. For the set of fitness, the one with the highest value is taken as the global-best $g_{best,0}$ (for a maximization problem). This set of initial fitness values are denoted as the particle-best $p_{best,0}^i$. The velocity is then calculated using the random gain coefficients. The particle positions are updated and the procedure repeats. Finally, at the satisfaction of some termination criteria, the global-best particle is reported as the near-optimal solution to the problem.

The algorithm contains a recursive iteration loop (generations) and is described by the following pseudo-code:

1. `Initialize particles randomly across the solution space`
2. `Set generation count to zero`
3. `While not terminate`
   (a) `Evaluate the particle fitness`
   (b) `Find the best particle`
   (c) `Find the best instances of particles`
   (d) `Calculate particle velocities`
   (e) `Update particle positions`
   (f) `Increase generation count`
4. `Terminate if generation count expires.`

3.3 Convergence

An analysis on the particle swarm behavior can be conducted using the control theory. Let the position error of a particular particle be denoted as (the particle index (i) is omitted):

$$\varepsilon_k = g_{best,k} - x_k, \tag{3.3}$$

the velocity expression becomes

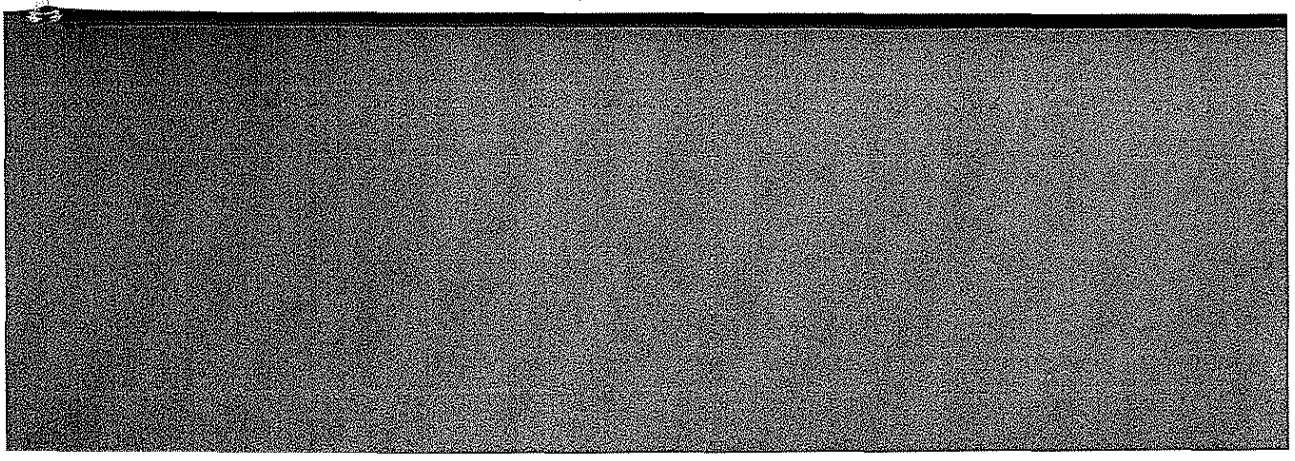
$$\begin{aligned} v_{k+1} &= wv_k + c_1\varepsilon_k + c_2 p_{best,k} - c_2 g_{best,k} + c_2\varepsilon_k \\ &= wv_k + (c_1 + c_2)\varepsilon_k - c_2(g_{best,k} - p_{best,k}). \end{aligned} \tag{3.4}$$

The particle update becomes

$$x_{k+1} = x_k + wv_k + (c_1 + c_2)\varepsilon_k - c_2(g_{best,k} - p_{best,k}). \tag{3.5}$$

By substituting the position error and assuming that the global optimal remains constant, i.e., $g_{best,k+1} = g_{best,k}$, one has:

$$\varepsilon_{k+1} = \varepsilon_k - wv_k - (c_1 + c_2)\varepsilon_k + c_2(g_{best,k} - p_{best,k}). \tag{3.6}$$



The particle position error and velocity can be written in the state-space form as

$$\begin{bmatrix} \varepsilon_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1-c_1-c_2 & -\omega \\ c_1+c_2 & \omega \end{bmatrix} \begin{bmatrix} \varepsilon_k \\ v_k \end{bmatrix} + \begin{bmatrix} c_2 \\ -c_2 \end{bmatrix} (g_{best,k} - p_{best,k}), \tag{3.7}$$

or

$$\mathbf{z}_{k+1} = \mathbf{A}\mathbf{z}_k + \mathbf{B}u_k, \tag{3.8}$$

where $\mathbf{z}_{k+1} = [\varepsilon_{k+1}, v_{k+1}]^T$ and $u_k = g_{best,k} - p_{best,k}$ and matrices $\mathbf{A}$, $\mathbf{B}$ are self-explanatory.

It becomes clear that the requirement for convergence implies $\varepsilon_k \to 0$ and $v_k \to 0$ as iteration $k \to \infty$. When the best solution is found $g_{best,k}$ becomes a constant and $p_{best,k}$ will tend to $g_{best,k}$ if the system is stable. The stability of a discrete control system can be ascertained by constraining the magnitude of the eigenvalues $\lambda_{1,2}$ of the system matrix $\mathbf{A} \in \mathbb{R}^{2\times 2}$ to be less than unity, that is

$$\lambda_{1,2} \in \{\lambda | \lambda^2 - (1+\omega-c_1-c_2)\lambda + \omega = 0\},\ |\lambda_{1,2}| < 1. \tag{3.9}$$

By choosing the maximal random variables c_1 and c_2 to be $c_{1,max} = 2$ and $c_{2,max} = 2$ and take the expectation values from a uniform distribution, then the coefficient averages become $c_1 = 1$ and $c_2 = 1$. This case corresponds to a total feedback of the discrepancy of the particle positions from the desired solution at $g_{best,k}$ [31]. The eigenvalues can be derived as:

$$\lambda_{1,2} = \frac{1}{2}\left(\omega - 1 \pm \sqrt{1 - 6\omega + \omega^2}\right). \tag{3.10}$$

After some manipulations, it can be shown that $\omega < 1$ with $c_{1,max} = c_{2,max} = 2$ will guarantee stability for the system, hence, the particles will converge to $g_{best,k}$.

3.4 PSO-based formulation

To speed up the formation initialization process, it is noted that λ_x and λ_y in (2.10) can be treated as controlled parameters in order to obtain a feasible orientation solution trajectory. Since they are coupled design parameters to be assigned in the controller for satisfactory performance, e.g. maximum error attenuation, the design can be cast as an constrained optimization problem. For this, the particle swarm optimization (PSO) algorithm is proposed owing to its flexibility and efficiency in handling complexity.

It is assumed that a high-level planner is available to assign a task dependent path for the leader-robot, where e_x, e_y, e_θ are tracking errors under control laws (2.12) to be minimized with respect to the parameter defined as:

$$\mathbf{z} = [\lambda_x\ \lambda_y]^T, \tag{3.11}$$

where λ_x and λ_y correspond to the error attenuation rates. The goal is to obtain these rates optimally such that the resultant function J (2.15) satisfies the constraints for stability of zero dynamics while keeping the drive commands within practical bounds. The optimization problem is formulated as:

$$\begin{aligned} \text{minimize:} \quad & |e_x|,\ |e_y|,\ |e_\theta| \\ \text{s.t.:} \quad & |J| > |\omega_i L| \\ & \lambda_x > 0,\ \lambda_y > 0 \\ & -v_{j,min} \le v_j \le v_{j,max} \\ & -\omega_{j,min} \le \omega_j \le \omega_{j,max} \end{aligned} \tag{3.12}$$

where the subscripts (min, max) denote the control bounds on the velocity and turn-rate. The leader-follower controller augmented with the proposed PSO is also able to alleviate singularities with a proper choice of parameter H, as previously stated.

3.5 Initialization Procedure

By combining the controller design and the PSO algorithm, the following optimization procedure is proposed. It is assumed the availability of the robot position information. Furthermore, for implementation simplicity, the zero dynamics stability condition (2.18) is checked at every time step in the following PSO optimization routine.

1. *Iteration* $k = 0$: Initialize a set of P particles, $\mathbf{z}_0^i$, $i = 1, \cdots, P$, as randomly generated potential solutions for (3.11).
2. The particles are then substituted into the controller (2.12) and check for the satisfaction of condition (2.18)
3. For those particles failing the test, reinitialize at Step 1 until all particles satisfy the stability condition.
4. *Iterations* $k > 0$: With the fitness of each particle chosen as the difference $|J| - |\omega_i L|$, the particle with a minimum difference is considered as the best particle in the group $z_{g,k}$ as it is less sensitive to steering angle θ_j.
5. The fitness is compared to the record of individual fitness in previous PSO iterations. The record of every particle that gives $\min\{|J| - |\omega_i L|\}$ is assigned as the best-experience $z_{p,k}^i$.

6. Check if constraints (3.12) are satisfied, if not the corresponding particle is de-activated or removed from further iterations. Those removed particles are replaced by re-initializing (Step 1) additional particles to maintain the total number of particles at P.

7. Repeated from Step 4 until the expiry of a maximum iteration count, e.g., $k = k_{max}$. The drive commands v_j and ω_j are then generated with $\mathbf{z}_{g,k_{max}}$.

The verification of the effectiveness of the formation controller design method will be given in the Section 5. In the sequel, an alternative approach for robotic formation coordination using directly swarm intelligence is proposed in the following to allow for collision avoidance while alleviating the design complexity in association with various formation configurations and control singularities.

4 FORMATION COOPERATIVE CONTROL USING SWARM INTELLIGENCE

In this section, the coordination of multi-vehicles into formations is achieved by combining the PSO algorithm and the behavior-based motion strategy in deriving the velocities and steering angles with inter-robot collisions taken into account.

4.1 Particle Structure

Let there be m robots to be coordinated, hence, there are m sequences of control commands to be determined by the PSO. The approach adopted assumes that a high-level path planner is available to design the required formation and each vehicle knows it current location. This gives a set of formation locations or virtual robots as

$$\mathbf{F} = [\mathbf{F}_1, \cdots, \mathbf{F}_m],\ \mathbf{F}_f = [x^f, y^f]^T,\ f = 1, \cdots, m, \tag{4.1}$$

where each formation location contains its corresponding xy-coordinates and the orientations are aligned with the x-axis. On the other hand, the initial locations of the robots are not specified (i.e., not in a formation) but their locations are known to the PSO algorithm.

The control commands are represented by a set of particles. Note that there are m robots each also contains a set of P particles describing the robots' possible locations. At each time step, the control particles are used to move the robots subject to their dynamics (2.1). The predicted locations are shown in Figure 4.1.

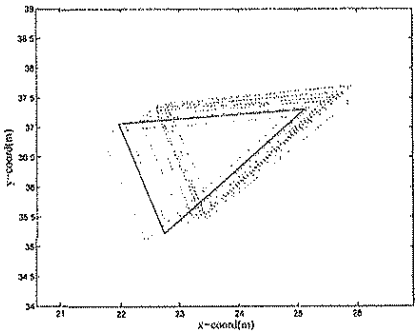


Figure 4.1: Predicted robot location according to the speed and steering commands determined by particles.

4.2 Indexing Robots in a Formation

In order to follow a formation, robots in the group need to be allocated an index corresponding to the virtual ones in the formation. At the start of the formation with the available knowledge of the virtual formation and robot locations, a cross-correspondence distance table is formed by calculating the distances

$$d^{fi} = \sqrt{(x^f - x^i)^2 + (y^f - y^i)^2}, \tag{4.2}$$

where the superscript f denotes the desired formation and i denotes the robot. Then, for each robot, find the minimum distance to its virtual one and assign the correspondence index

$$i \mapsto f, \quad \text{if } \hat{d}^{fi} = \min_f \{d^{fi}\}. \tag{4.3}$$

By adopting this indexing scheme, the initial distances between the real and virtual robots are minimized.

4.3 Formation Strategy

The grouping of robots into a desirable platoon is treated as a tracking problem. Due to the nonholonomic constraints of the vehicles they cannot turn abruptly. First, the distances between the particles (robots) and their virtual ones are calculated, giving respectively $\Delta x_p^{fi}, \Delta y_p^{fi}$. Consider a typical scenario where the formation is required to move along the x-axis from left to right (it is

straightforward to generalize to other formation movements as it is assumed to be performed by a high-level path planner). The strategy adopted here is to assign a pseudo target behind the virtual of a robot such that it chases and tracks the formation. The pseudo target, with superscript v, is given by:

$$\begin{aligned}\Delta x_p^{vi} &= \Delta x_p^{fi} - 0.85 d^{fi} \\ \Delta y_p^{vi} &= \Delta y_p^{fi} - 0.05 d^{fi} sign(\Delta y_p^{fi}).\end{aligned} \tag{4.4}$$

The factors 0.85 and 0.05 are some scale factors empirically determined. Examining the expressions above reveals that the pseudo target will approach the virtual of a robot when it is moving into the vicinity of the formation where $d^{fi} \to 0$ and, thus, providing a smooth tracking.

4.4 Particle Fitness

The PSO algorithm relies on the determination of relative fitness values among the particles where the group-best and personal-best particles are obtained. The fitness function is an aggregation of the robot-to-formation distance and the robot-to-formation angular separation. The distance on a particle basis, d_p^{fi}, is given in (4.2) while the angular separation is obtained by:

$$\theta_p^{fi} = \theta_p^i - \arctan(\Delta y_p^{vi},\ \Delta x_p^{fi} + L_v), \tag{4.5}$$

where L_v is the vehicle (robot) length. Here, each angle is referred to the angular separation between a robot particle and its corresponding virtual. The fitness value for each robot particle is given as:

$$f_p^{fi} = d_p^{fi} + |\theta_p^{fi}|. \tag{4.6}$$

The group-best fitness is obtained from

$$\hat{f}_g^{fi} = \max_p \{f_p^{fi}\}. \tag{4.7}$$

Similarly, the personal-best fitness is

$$\hat{f}_{p,j}^{fi} = \max_j \{f_{p,j}^{fi}\}, \tag{4.8}$$

determined from the history of the fitness of the p^{th} particle up to the j^{th} generation.

Finally, these fitness values are used in calculating the particle velocities and updating their locations in the solution space.

4.5 Control Bounds

Since the PSO is a heuristic search method, the speed and steering commands derived may be infeasible for typical vehicles that exceed their kinematic or dynamic limitations. Therefore, the control signals should be bounded or clamped. Taking this into account, the values of the control particles are assigned as

$$v_{p,k}^i \leftarrow \begin{cases} v_{max}, & v_{p,k}^i > v_{max} \\ 0, & v_{p,k}^i < 0. \end{cases} \tag{4.9}$$

That is, the robot is not allowed to travel backward in normal formation for a smooth motion. Similarly, the steering command is also bounded in magnitude:

$$\gamma_{p,k}^i \leftarrow \begin{cases} \gamma_{max}, & \gamma_{p,k}^i > \gamma_{max} \\ -\gamma_{max}, & \gamma_{p,k}^i < -\gamma_{max}. \end{cases} \tag{4.10}$$

4.6 Inter-robot Collision Avoidance

The path required for a robot to reach a desired location may incur potential collision with nearby robots if collision was not considered in the PSO routine. Therefore, a collision avoidance strategy is developed to address this issue.

For each robot located at $\mathbf{x}_k^i$ at time k, calculate the distances to other vehicles, i.e.,

$$d^{ji} = \sqrt{(x^j - x^i)^2 + (y^j - y^i)^2}, \tag{4.11}$$

which is a 2-dimensional array. Similarly, calculate the angular separation between robots as:

$$\theta_{ji} = \arctan((y^j - j^i), (x^j - x^i)). \tag{4.12}$$

For each robot i, find the distance to its associated virtual robot in the formation,

$$f^{ii} = \sqrt{(x_f^i - x^i)^2 + (y_f^i - y^i)^2}. \tag{4.13}$$

Check for the collision condition defined as

$$(d^{ji} < 2.5 L_v + 0.05 f^{ii}) \wedge (|\theta^{ji} < \pi/3), \tag{4.14}$$

where $2.5L_v$ is taken as a safe distance and $\wedge$ is the logical AND-operator. The scale factors 2.5, 0.05 and $\pi/3$, determining the allowed tolerances for the inter-robot separations in distance and orientation, are obtained empirically.

A potential collision is then declared between robots i and j when they are close to each other or one of them is in front of and blocks the other. This condition has been taken into account for with the dynamic threshold, f^{ii}, depending on the degree of formation establishment, whereby the risk of collision diminishes when the robots enter the formation, i.e., $f^{ii} \to 0$. Reactive movements may also apply to the robots causing potential

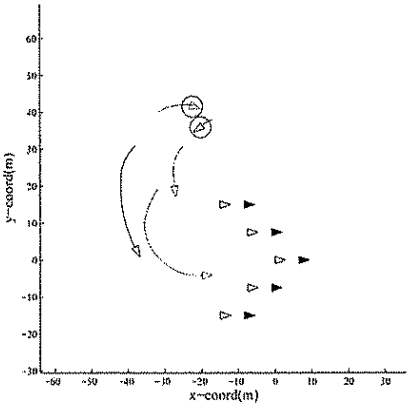


Figure 4.2: Inter-robot collision among vehicles marked by circles, (empty triangles are starting vehicle locations, solid triangles are current positions).

collision by freezing the control commands temporarily for a time step such that their locations become:

$$\begin{aligned} \mathbf{x}^i_{k+1} &= \mathbf{x}^i_k \\ \mathbf{y}^i_{k+1} &= \mathbf{y}^i_k. \end{aligned} \tag{4.15}$$

Figure 4.2 illustrates the situation when a potential collision occurs.

4.7 Dead-lock Release

The inter-robot collision avoidance strategy adopted may, in turn, give rise to a dead-lock condition, especially when the robots are moving towards each other (both in front and block the other). A release of the dead-lock condition is proposed as follows.

During the collision avoidance stage, a list of colliding robot index pairing is maintained [28]. For example, let robots 1 and 3 are temporarily frozen, the list will read as

$$\mathbf{L}_{fz} = \left\{ \begin{array}{cc} 1 & 3 \\ 3 & 1 \end{array} \right\}, \tag{4.16}$$

which implies that robot 1 blocks robot 3 and the reverse also holds, thus, producing a dead-lock.

If the number of paired entries in the list is more than one, the list is searched for duplicative indices. Following the above example and applying the proposed strategy, robot 1 will be reactively-controlled until the corresponding entry in the dead-lock list is removed. The procedure then repeats for other dead-locked pairings such that multiple dead-locks can be subsequently removed.

5 RESULTS

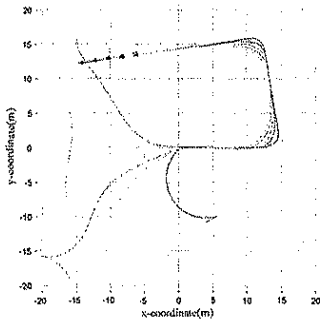
In this section, extensive numerical examples are presented to illustrate the effectiveness of the methodology developed in this chapter. On the basis of equivalent leader trajectories, the impact of configuration schemes, of the generic structure, on the overall formation will be highlighted. Then test cases for PSO-based controller designs are illustrated by investigating the tracking errors obtained from a number of formation patterns. Finally, results from simulations for the PSO-based formation control are given as a proof-of-concept of directly applying PSO in the robotic formation problem. A remark on the implementation is also included.

5.1 Generic Control Structures

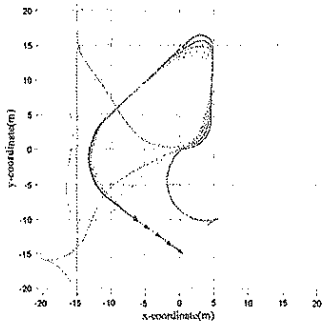
5.1.1 Single-Leader

Results from the single-leader formation scheme are shown in Figure 5.1 for 5 mobile robots deployed in two different trajectories made by their leader. In Fig. 5.1(a), robots (shown as small triangles) are initially driven from the centre of the work-space toward the right-hand-side, then turn upwards and finally follow a straight path to the left-hand side. It can be seen that paths traced by the follower robots are maintained in close vicinities when the formation is moving in a straight line. In periods of turning, the high-index robot (away from the leader) tends to slow down on its previous location until the leader makes a straight line again. This illustrates that the drive commands of the last robot is much scaled-down by a multiple of the displacements L or H.

In Fig. 5.1(b), the turns are sharper than the previous case. In addition, the path contains a third moderate turn (on the left of the work-space). For the two sharper turns, the follower trajectories are similar to the previous case. Notice that for the moderate turn, followers tend to maintain the curvature. It may be concluded that, as the task demands, the single-leader scheme may enable the formation to travel in narrow passages in a column formation.

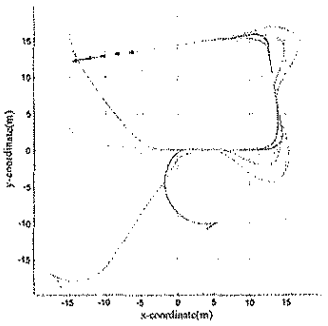


(a)

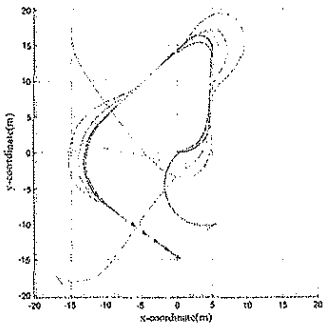


(b)

Figure 5.1: Robot trajectories for single-leader scheme.



(a)



(b)

Figure 5.2: Robot trajectories for multiple-leader scheme.

5.1.2 Multiple-Leader

For the multiple-leader scheme, test results are depicted in Figure 5.2. Similar trajectories are designed for the leader as in the single-leader test case. In Fig. 5.2(a), the followers make diverted paths when they need to make sharp turns. Notably, the followers have to travel extended distances in order to be kept in formation. However, the duration of loss of the *column* formation is thus reduced. In Fig. 5.2(b), a moderate turn is also imposed. The followers also make diverted tours, compromising travel distance to tight formations. Again, as the task demands, tight column formations can be formed at the expense of extended follower travels.

5.2 PSO-based Leader-Follower Initialization

Simulations are conducted with line, column, wedge and diamond patterns for a group of typically 5 mobile robots for the sake illustration (the number of robots in the platoon may be increased). The robots start from arbitrary locations and orientations, and are steered by the drive commands towards forming the desired pattern.

Let the leader initial position be the origin of the world coordinate, i.e., $x_i = y_i = \theta_i = 0$. The leader then moves to the right, turns in the positive y-axis direction. It then makes a further turn towards the left-bottom of the work space. When the leader approaches $x_i = 0$ again, it makes a final turn towards the right-bottom area. See, for example, the trajectory depicted in Fig. 5.3(a).

5.2.1 Line

There are two cases of line formations simulated, namely, lines formed when the followers are on the right and left hand sides of the leader. The configurations are specified by $L = 1$, $R = \pm 1$ and $H = 1$.

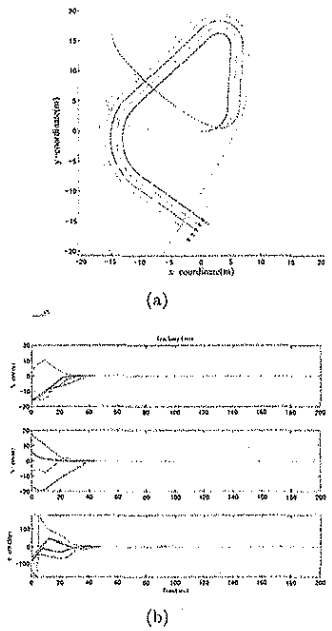


Figure 5.3: Line formation; case 1 - (a) robot trajectories, (b) tracking error.

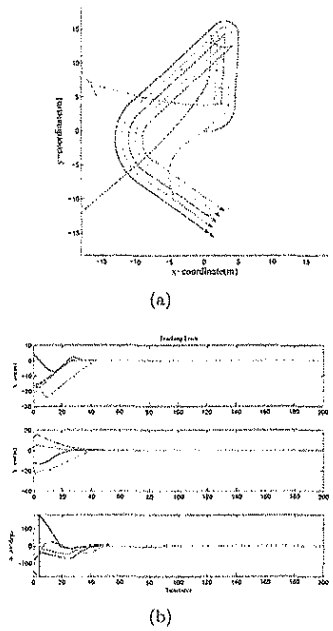


Figure 5.4: Line formation; case 2 - (a) robot trajectories, (b) tracking error.

Right-hand Sided Followers: The path followed by the leader and followers (in small triangles) are shown in Fig. 5.3(a) while the tracking errors are plotted in Fig. 5.3(b). Followers start from arbitrary positions, as shown by the traces, initially move towards the center of the work space where the leader is located. This initialization phase is evident from the extended tracking errors plotted in Fig. 5.3(b). After 40sec and the line formation settles and is then maintained in the rest of the test duration. In this formation, the followers turn in an so-called outer-circle with $R = 1$ and the proposed header-tracking controller is performing satisfactorily.

Left-hand Sided Followers: In this case, the leader trajectory is the same as before but the followers are allocated on the left-hand side of the leader and follow a inner-circle with $R = -1$, Fig. 5.4(a). The formation is also well established and maintained through the test duration. Specifically, when the leader turns an abrupt corner on the center-top area, the inner-most follower has to make a reactive move in order to cope with the sharp turn. Note that, the tracking errors have been kept small as illustrated in Fig. 5.4(b). Here, in this difficult condition, the controller continues to perform satisfactorily.

5.2.2 Column

Results obtained from a column formation are shown in Fig. 5.5(a) and 5.5(b) for the trajectory and tracking errors respectively. It is observed that when the leader turns, the tracking error increases, e.g, during 80 to 100sec. The followers trace circular paths with smaller radius which is equivalent to a higher turn-rate and but performance of the formation establishment is generally acceptable.

5.2.3 Wedge

The wedge formation can be considered as a variation of the line formation where followers are located on both sides of the leader and progressively displaced behind the leader. Results are shown in Fig. 5.6(a) and 5.6(b) for the robot trajectories and tracking errors. A reactive movement is also observed for the inner-follower while making a sharp turn on the center-top area. Nonetheless, the formation has been maintained as expected.

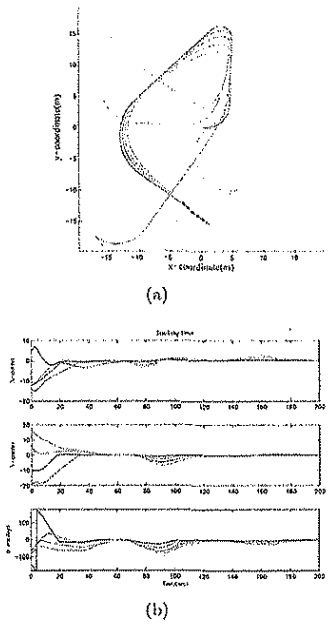


Figure 5.5: Column formation - (a) robot trajectories, (b) tracking error.

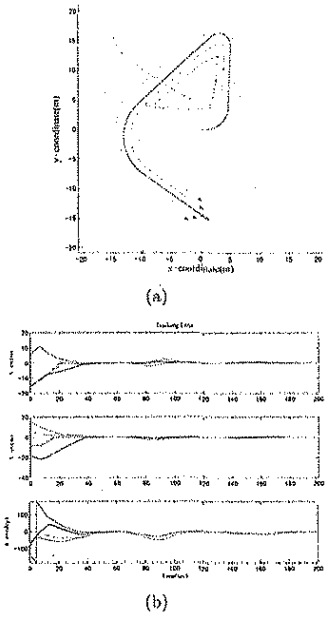


Figure 5.6: Wedge formation - (a) robot trajectories, (b) tracking error.

5.2.4 Diamond

Figures 5.7(a) and 5.7(b) represent the trajectories and tracking errors. Similar trajectories as the previous cases are traced by the robots and the results are also compared favorably to the other cases. The tracking errors are generally smaller than others, except for the line formation. It is because of the close proximity of the robots in the formation, i.e., small L, in order to ensure that the existence of the zeros dynamics has been maintained.

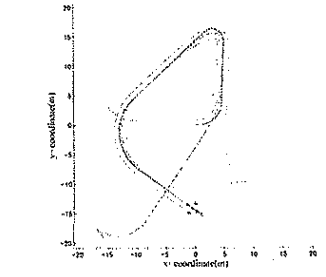
5.3 Swarm-based Cooperative Control

The proposed multi-robot coordination framework for formation initialization and control, using PSO and the incorporation of behavioral control for collision avoidance, is applied in simulations for various formation types with different set of initial locations. The simulation conditions are listed below.

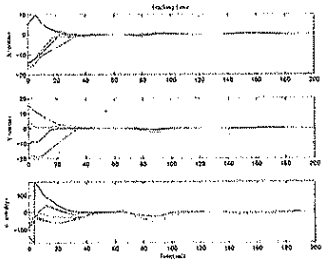
Consider seven vehicles that are homogeneous with each vehicle having measures $3m \times 2m$ in length and width. Let the operation site be bounded, e.g., $\pm 70m$ in the xy-coordinates. The formation starts at the center and the initial vehicle locations are chosen arbitrarily as well as their orientations are also initialized randomly. The formation moves from left to right (increasing x-coordinate) and the formation is allowed to change dynamically. The last virtual vehicle (located at the most negative y-coordinate) moves in a higher speed than the first virtual vehicle.

5.3.1 Line

In the line formation, trajectories of the vehicles are shown in Figure 5.8. Irrespective of the vehicle starting locations and the dynamically adjustment of the formation, the vehicles follow the formation closely. It is also illustrated that the trajectories are smooth and the formation virtual vehicles are well tracked. This observation has verified the satisfactory performances provided by the proposed PSO algorithm. The trajectories although indicate some crossovers, however, they are separated in the time domain where the inter-vehicle collision avoidance procedure has been operating effectively. Notably, as shown in the lower-left region of Fig. 5.8(c), one of the vehicle trajectories contains a rather sharp change. This is caused by the



(a)



(b)

Figure 5.7: Diamond formation - (a) robot trajectories, (b) tracking error.

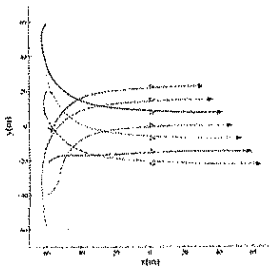
blocked vehicle that has to move backward in order to avoid inter-vehicle collision and dead-lock.

5.3.2 Wedge

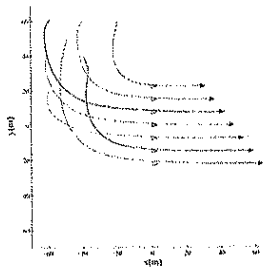
For the wedge formation case, similar levels of performances are also observed from the plots in Figure 5.9. It is noted that the successful establishment of the desired formation, from the initial vehicle locations, is independent on the type of the desired formation (column vs. wedge). Furthermore, the tracking of virtual vehicles is formation-independent and the trajectory complexity is also not related to the type of formation. It is worth noting that the proposed approach can be straightforward extended to cases of a larger number of vehicles and inhomogeneous vehicle sizes.

5.4 A Remark on Implementation

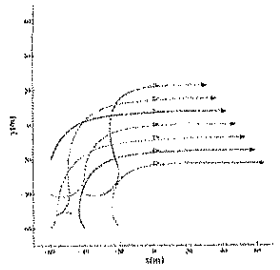
Typical snapshots of experimental formations using two Amigo® and one Pioneer® mobile robots, in *line*, *column* and *wedge* are presented in Figure 5.10. The robots drive commands are derived from a PC and issued to the robots via a wireless



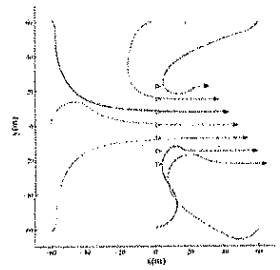
(a) Vehicles started from left



(b) from upper-left region

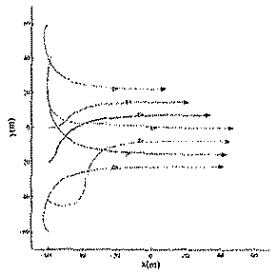


(c) from lower-left region

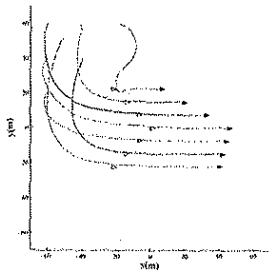


(d) from distributed regions

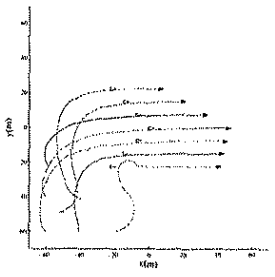
Figure 5.8: Results for Line formation.



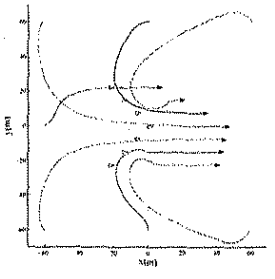
(a) Vehicles started from left



(b) from upper-left region



(c) from lower-left region



(d) from distributed regions

Figure 5.9: Results for Wedge formation.

data link.

The two Amigos steered in line, column and wedge formation patterns are shown in Figures 5.10(a) to 5.10(c) respectively. It is noted that the desired patterns are established successfully only on flat terrain, as required in conditions (2.2) and (2.3). In the wedge formation, the Pioneer (in the front of Figure 5.10(c)), having more sensing capability, acts as the leader while the two Amigos are the followers. The robots moving in a pattern can be controlled to change their formation to avoid obstacles, for example when passing a narrow passage.

6 CONCLUSION

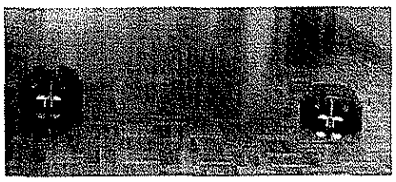
This chapter has presented a swarm-based methodology to deploy and maintain the motion of a group of mobile robots in desired formations. Features of a generic formation configuration schemes are introduced indicating the capability of forming various configurations. As the robotic task required, different schemes are needed for trajectory tracking and formation maintenance while avoiding inter-robot collision and control singularities. A near-optimal leader-follower formation initialization design is developed with advantages adopted from the particle swarm optimization (PSO) algorithm. The application of swarm intelligence has also been demonstrated in the cooperative control design in the robotic formation context. With the use of behavioral collision-avoidance and dead-lock release procedures, the performance of the later method is also satisfactory when dealing with the complicated constraints in establishing various formation configurations and control singularities. Results from extensive numerical studies and some laboratorial work have verified the effectiveness of the proposed swarm-based design approaches for multi-robot formations.

ACKNOWLEDGMENT

This work is supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

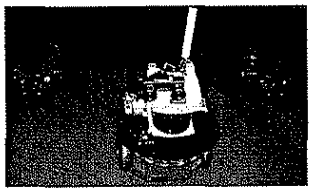
REFERENCES

[1] H. Takeda, Z. D. Wang, Y. Hirara and K. Kosuge, "Load sharing algorithm for transporting an object by two mobile robots



(a) line

(b) column



(c) wedge

Figure 5.10: Experimental formations.

in coordination," in *Proc. 2004 Intl. Conf. on Intelligent Mechatronics and Automation*, Chengdu, China, Aug. 2004, pp. 374-378.

[2] Y. Hao and S. K. Agrawal, "Formation planning and control of UGVs with trailers," in *Autonomous Robots*, vol. 19, pp. 257-270, 2005.

[3] A. Warszawski and R. Navon, "Implementation of robotics in building: current status and future prospects," in *Journal of Construction Engineering and Management*, January/February 1998, pp. 31-41.

[4] S. Lee, T. M. Adams and B. Ryoo, "A fuzzy navigation system for mobile construction robots," in *Automation in Construction*, vol. 6, 1997, pp. 97-107.

[5] J. P. Desai, J. P. Ostrowski and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," in *IEEE Trans. on Robotics & Automation*, vol. 17, no. 6, pp. 905-905-908, Dec. 2001.

[6] 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 & Automation*, Seoul, Korea, May 2001, pp. 2885-2890.

[7] R. Fierro, A. K. Das, V. Kumar and J. P. Ostrowski, "Hybrid control of formations of robots," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, Seoul, Korea, May 2001, pp. 157-162.

[8] M. Sisto and D. Gu, "A fuzzy leader-follower approach to formation control of multiple mobile robots," in *Proc. 2006 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Beijing, China, Oct. 2006, pp. 2515-2520.

[9] Q. P. Ha and G. Dissanayake, "Robust formation using reactive variable structure systems," in *Intl. Trans. on Systems Science and Applications*, vol. 1, no. 2, pp. 183-192, 2006.

[10] J. P. Desai, J. Ostrowski and V. Kumar, "Controlling formations of multiple mobile robots," in *Proc. 1998 IEEE Intl. Conf. on Robotics & Automation*, Leuven, Belgium, May 1998, pp. 2864-2869.

[11] J. Shao, G. Xie, J. Yu and L. Wang, "A tracking controller for motion coordination of multiple mobile robots," in *Proc. 2005 IEEE/RSJ Intl. Conf. on Intelligent Robots & Systems*, Edmonton, Canada, Aug. 2005, pp. 783-788.

[12] A. D. Nguyen, Q. P. Ha and H. T. Nguyen, "Virtual-head robot tracking and three-point *t-l* control for multiple mobile robots," in

Proc. IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications, Prague, Czech Republic, Jun. 2006, pp. 73-78.

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

[14] M. A. Lewis and K. H. Tan, "High precision formation control of mobile robots using virtual structures," in *Autonomous Robots*, vol. 4, pp. 387-403, 1997.

[15] J. Shao, G. Xie, J. Yu and L. Wang, "Leader-following formation control of multiple mobile robots," in *Proc. 2005 IEEE Intl. Symposium on Intelligent Control*, Limassol, Cyprus, Jun. 2005, pp. 808-813.

[16] A. D. Nguyen, Q. P. Ha, S. Huang, and H. Trinh, "Observer-based decentralised approach to robotic formation control," in in *Proc. of the 2004 Australian Conf. on Robotics and Automation*, Canberra, Australia, pp. 1-8, 2004.

[17] V. T. Ngo, A. D. Nguyen, Q. P.Ha, "Toward a generic architecture for robotic formations planning and control," in *Proc. The Sixth International Conference on Intelligent*, Phuket, Thailand, Dec. 2005, pp. 89-96.

[18] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," in *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 34, no. 1, Feb, 2004, pp. 718-725.

[19] Y. Ding, Y. Han and J. Jiang, "Multi-robot cooperation method based on the ant algorithm," in *Proc. 2003 IEEE Swarm Intelligence Symposium*, Indianapolis, Indiana, USA, Apr. 2003, pp. 14-18.

[20] H. Chen and Z. Xu, "Path planning based on a new genetic algorithm," in *Proc. Intl. Conf. on Neural Networks and Brain*, Beijing, China, Oct. 2005, pp. 788-792.

[21] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability and convergence in a multidimensional complex space," in *IEEE Trans. Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, Feb. 2002.

[22] G. Venter and J. S. Sobieski, "Particle swarm optimization," in *Proc. 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Material Conf.*, Denver, Colorado, Apr. 2002, AIAA 2002-1235.

[23] Y. Zheng, L. Ma, L. Zhang and J. Qian, "Robust PID controller design using particle swarm optimizer," in *Proc. 2003 IEEE Intl. Symposium on Intelligent Control*, Houston, Texas, USA, Oct. 2003, pp. 974-979.

[24] H. Zhang, X. Li, H. Li and F. Huang, "Particle swarm optimization-based schemes for resource-constrained project scheduling," in *Automation in Constructions*, vol. 14, 2005, pp. 393-404.

[25] S. L. Ho, S. Yang, G. Ni, E. W. C. Lo and H. C. Wong, "A particle swarm optimization-based method for multiobjective design optimizations," in *IEEE Trans. on Magnetics*, vol. 41, no. 5, May 2005, pp. 1756-1759.

[26] Y. Qin, D. Sun, N. Li and Y. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in *Proc. 3rd Intl. Conf. on Machine Learning & Cybernetics*, Shanghai, China, Aug. 2004, pp. 2473-2478.

[27] S. Doctor and G. K. Venayamoorthy, "Unmanned vehicle navigation using swarm intelligence," in *Proc. IEEE Intl. Conf. on Intelligent Sensing & Information Processing*, Chennai, India, Jan. 2004, pp. 249-253.

[28] N. M. Kwok, Q. P. Ha, V. T. Ngo and S. M. Hong, "Particle swarm optimization-based coordination of a group of construction vehicles," in *Proc. Intl. Symposium on Automation & Robotics in Construction*, Tokyo, Japan, Oct. 2006, pp. 840-845.

[29] A. D. Nguyen, N. M. Kwok, V. T. Ngo and Q. P. Ha, "Collision-free formations with reactively-controled virtual head robot tracking," in *Proc. 2006 IEEE/RSJ Intl. Conf. on Intelligent Robots & Systems*, Beijing, China, Oct. 2006, pp. 2509-2514.

[30] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Intl. Conf. on Neural Networks*, Perth, Australia, Nov. 1995, pp. 1942-1948.

[31] N. M. Kwok, D. K. Liu, K. C. Tan and Q. P. Ha, "An empirical study on the settings of control coefficients in particle swarm optimization," in *Proc. 2006 IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, Jul. 2006, pp. 3165-3172.