Proceedings of the 2004 IEEE
International Conference on Networking, Sensing & Control
Taipei, Taiwan, March 21-23, 2004

Multi-Agent Based Formation Control Using a Simple Representation *

Harry Chia-Hung Hsu
Department of Electrical Engineering
National Chung Cheng University, Taiwan.
Email: d8742021@ccu.edu.tw

Alan Liu
Department of Electrical Engineering
National Chung Cheng University, Taiwan.
Email: aliu@ee.ccu.edu.tw

Abstract – *Distributed multi-robot systems have been an active research field in recent years, and this paper focuses on the formation control. When vehicles keep a formation, such as platoons, it may increase throughput in automated highway systems. This paper presents a simple but powerful formation representation method, which can present arbitrary or regular infinite formation. This paper also proposes a kinematic control method of mobile robots and it is proved convergence in maintaining formation. A architecture named Virtual Operator Multi-Agent System is used to assist formation control in joining robots into a team or in removing robots from a team. It is also useful in formation split and mergence.*

Keywords: Formation, Robot Control, Graph, Multi-Agent System.

1 Introduction

In recent years, distributed multi-robot systems have been an active research field [1]. One important application domain of the control and cooperation of multi-robot systems is Intelligent Transportation Systems (ITS). Two common goals of ITS are increasing transportation system capacity and safety [2]. When vehicles keep a formation, such as platoons, it may increase throughput of automated highway systems (AHS) [3].

In this paper, we focus our research on formation control. In formation control, [4] uses motor schema to keep formation by three principles: unit-center reference, leader reference, and neighbor reference. The key idea of [5] is to let each robot keep formation with another specific robot, called a friend, with a desired angle. In [6], it uses leader-follower control to maintain robots by a desired separation or a desired bearing, and it can also perform formation switch. [7] models multiple teams of mobile robots by graph theoretic approach. [8] uses virtual structures to maintain formation. [9] uses some simple algorithm to maintain several different formation, such as circle, polygon, line, etc.

In this paper, we proposes a simple formation representation method, which is able to present arbitrary/irregular formations or regular infinite formations. Using the infinite formation method, it is easy to add robots into a team and they still keep the same formation. Conversely, we can randomly remove robots from a team without affecting the formation. In AHS, vehicles generally keep the formation of platoons infinitely, and vehicles can randomly join into or split from the platoons. The graph method in [6, 10] can also present arbitrary formations, but it cannot present a infinite formation. [9] can keep formation with random number of mobile robots in simulation, but the algorithms only support several specific formations and it cannot support arbitrary formation. In addition, it can not move the whole formation to a target. In some situations, such as military attack, a system needs to both keep formation and move to a target. In these situations, the system cannot accomplish it.

Besides the formation representation, we also develop a method of kinematic control to control mobile robots. We present some reference models for mobile robots to maintain formation in the same team, and they are *leader reference*, *predecessor reference*, and *soft-neighbor reference*. For multiple robot teams, an architecture, called VOMAS (Virtual Operator Multi-Agent System), gets the flexibility to control robots in multiple teams.

In Section 2, we describe the method of formation representation. The reference architectures of formation maintenance are shown in Section 3. The kinematic control of mobile robots is presented in Section 4. We present VOMAS architecture in Section 5. The implementation is given in Section 6, and conclusion is given in Section 7.

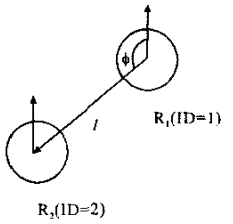

Figure 1: Edge Relation $E(2, 1, l, \phi)$.

2 Formation Representation

We use Directed Acyclic Graph (DAG) to represent an arbitrary formation or a regular infinite formation. From the graph theory view, each robot is viewed as a *node*, and the relation between two robots as *edge*. Each robot is given an identity (ID), and the ID number determines the robot's position. We use leader-follower method to form the formation. The ID number of the leader in a formation

*This research was supported by the National Science Council under grant NSC91-2213-E-194-003

is 1. Other robots depend on their ID and their corresponding edge relation to follow their predecessor with some fixed relation. The form of edge relation is described as follows.

$$E(c, sucID, preID, l, \phi), \quad (1)$$

which means when *condition* c is true, a *successor* robot, whose ID is *sucID*, follows its *predecessor* robot, whose ID is *preID*, with distance l and angle ϕ. Distance l is the distance between two robots' center and angle ϕ is the angle between predecessor robot's progress direction and direction from the predecessor robot to the successor robot, as shown in Figure 1. The robot platform in this paper is a circular holonomic robot, so that the minimal limitation of l is two times of radius of a robot. In Figure 1, it is an example of edge relation $E(1, 2, 1, l, \phi)$, which means robot R_2 (ID=2) follows robot R_1 (ID=1) with distance l and angle ϕ, and the condition is always true (c is 1). Here, robot R_1 is called robot R_2's predecessor and, inversely, robot R_2 is called robot R_1's successor.

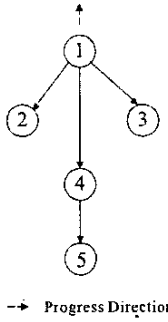

Figure 2: Arbitrary Formation.

Using the basic edge relation, we can express arbitrary formations. In Figure 2, we assume the distance and angle between robot i and robot j are l_{ij} and ϕ_{ij}, so that the formation F can be expressed as

$$\begin{aligned} F = \{ \quad & E(1, 2, 1, l_{12}, \phi_{12}), \\ & E(1, 3, 1, l_{13}, \phi_{13}), \\ & E(1, 4, 1, l_{14}, \phi_{14}), \\ & E(1, 5, 4, l_{45}, \phi_{45})\}. \end{aligned}$$

Using this method, we can build any arbitrary formation.

However, if the number of robots in a formation is large, even infinite, the arbitrary formation has limitation to express it, because an engineer cannot design infinite edge relations. To solve this problem, we can use ID variables as a condition parameter to match the relationship between a predecessor and a successor to extend a formation to infinite. For example, Figure 3 shows a infinite line formation and infinite column formation. For each robot's ID i, robot i uses the following relations to match its position to be a line formation.

$$\begin{aligned} F = \{ \quad & E(1, 2, 1, 50, 90^\circ), \\ & E(i > 2, i, i-2, 50, 90^\circ + 180^\circ(i\%2)) \quad \}, \end{aligned}$$

where % means modulus operator, so that the value of $i\%2$ is the remainder of i over 2. From the relations, robot R_1 is the leader, and it does not need to match its position by edge relations. Robot R_3 ($i = 3$) maintains relation with robot R_1 in distance $50cm$ and in angle 270°. Other robots can use these edge relations to know its predecessor and its corresponding position. Similarly, the column formation in Figure 3 is

$$F = \{E(1, i, i-1, 50, 180^\circ)\}.$$

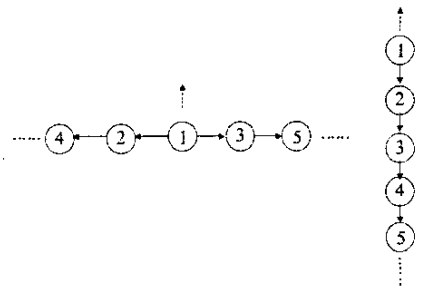

Figure 3: Regular Infinite Line and Column Formations.

Using this formation representation, we can express any regular infinite formation. We give another example of platoons, which is used frequently, such as vehicles in automated highway systems. The edge relations of the platoon formation are shown as follows and its shape is shown in Figure 4.

$$\begin{aligned} F = \{ & E(i \leq 4, i, i-1, 50, 270^\circ), \\ & E(i > 4, i, i-4, 50, 180^\circ)\}. \end{aligned}$$

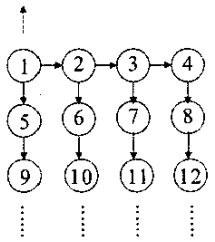

Figure 4: Platoon Formation.

3 Formation Maintenance

The formation representation determines each robot's position in a formation. For formation maintenance, we use leader-follower method to maintain formation. When the leader moves, all other followers move together to keep formation. Several reference models are designed and described as follows.

Leader Reference. The leader reference is used when all followers can get the leader's information. For example, this reference may be used when global sensors are used, such as over-head camera, or when local sensors are used in few robots and each robot can detect

the leader. In the formation representation, it does not describe the edge relation between each robot and the leader. However, each robot can calculate the edge relation between itself and the leader by recursively calculating edge relation between a successor and a predecessor until the leader. To maintain formation in this reference model, each robot follows the leader according to each robot's edge relation with the leader.

Predecessor Reference. In this reference, each successor follows its predecessor so all followers recursively follow the leader. This reference model is used when not all followers can detect the leader's information, such as using local sensors in many or infinite robots. The stability of this reference may be worse than the leader reference, because errors between a predecessor and a successor will propagate and increase to other successor followers. In the leader reference, errors only occur independently between each robot and the leader, and they do not propagate to other followers.

Soft-Neighbor Reference. The soft-neighbor reference does not exist independently, and it is used to improve the stability especially for the predecessor reference. If a robot can sense other neighbors and know their ID, the robot can calculate the edge relation between itself and its neighbors from the formation representation. The robot build one or multiple *soft* relations to follow its neighbors, so that it has multiple predecessors and the stability may be improved.

4 Kinematic Control

In formation control, when the leader moves, other successor robots follow the leader, so that the kinematic control of each successor robot is needed. In Figure 5, it shows the position relationship between a predecessor (R_1) and the current position and velocity direction of a successor (R_2), and its desired position (R_2^d). The relation between R_2 and R_1 is in distance l and angle ϕ (or vector l). Similarly, the relation between R_2^d and R_1 is l^d and ϕ^d (or vector $\mathbf{l^d}$). The velocity of R_1 and R_2 are vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ with corresponding angles θ_1 and θ_2.

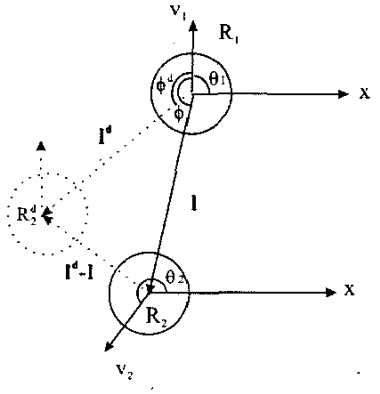

Figure 5: Position Relations.

We want $\mathbf{v_2}$ to lead robot R_2 to approach R_2^d, so that we design the desired velocity of R_2 as $\mathbf{v_2^d}$, shown in Figure 6. In $\mathbf{v_2^d}$, the velocity $\mathbf{v_1}$ will keep robot R_2 follow robot R_1, and velocity $\mathbf{v}_{\Delta \mathbf{l}}$ let robot R_2 approach its desired position. Angle θ_2^d is the corresponding angle of $\mathbf{v_2^d}$, and ρ is the angle between $\mathbf{v_2}$ and $\mathbf{v_2^d}$. We use standard I/O linearization to design vector $\mathbf{v}_{\Delta \mathbf{l}}$ [11]. The equations are

$$\Delta \mathbf{l} = \mathbf{l^d} - \mathbf{l} \quad (2)$$

$$\mathbf{v}_{\Delta \mathbf{l}} = \alpha_1 \Delta \mathbf{l} \quad (3)$$

$$\mathbf{v_2^d} = \mathbf{v_1} + \mathbf{v}_{\Delta \mathbf{l}}, \quad (4)$$

where α_1 is a constant. However, in physical robot control, we cannot control the robot directly by a vector $\mathbf{v_2^d}$. We need to give each robot a physical linear velocity scale v_2 and an angular velocity w_2, and they are designed as:

$$v_2 = \mathbf{v_2^d} \cdot \mathbf{u}_{\mathbf{v_2}} = |\mathbf{v_2^d}| \cos\rho \quad (5)$$

$$w_2 = -\alpha_2 \rho, \quad (6)$$

where α_2 is a constant, w_2 is designed by linearization [11], $\mathbf{u}_{\mathbf{v_2}}$ is the unit vector of $\mathbf{v_2}$, and v_2 and w_2 have maximal limitation of $v_{2,max}$ and $w_{2,max}$ depending on physical hardware limitation.

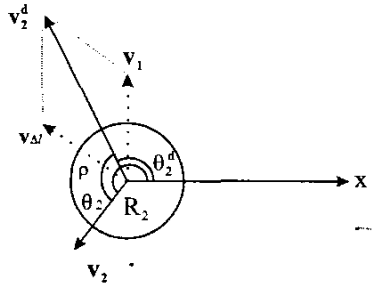

Figure 6: Desired Velocity Vector of Robot R_2.

In the following, we will prove that when robot R_2 is controlled by (5) and (6), it will converge to its desired position and keep the same direction with its predecessor.

Lemma 1 ($\mathbf{v_2}$ converges to $\mathbf{v_2^d}$). Using (5) and (6), the velocity $\mathbf{v_2}$ will converge toward desired $\mathbf{v_2^d}$ when $\mathbf{v_2^d}$ is fixed.

Proof.

$$\theta_2 = \theta_2^d + \rho. \quad (7)$$

$\mathbf{v_2^d}$ is fixed, so θ_2^d is fixed. Hence,

$$w_2 = \theta_2' = \rho'. \quad (8)$$

Using linear differential to solve (6) and (8), ρ can be a time dependent function as

$$\rho(t) = c_1 e^{-\alpha_2 t}, \quad (9)$$

where c_1 is a constant. When $t \to \infty$, $\rho(t) \to 0$. Therefore, the scale of velocity $\mathbf{v_2}$ approaches $|\mathbf{v_2^d}|$ (from (5)), and angle θ_2 approaches θ_2^d (from (7)). It is proved that $\mathbf{v_2}$ approaches $\mathbf{v_2^d}$. □

Lemma 2 (Position Convergence). Using (5) and (6), $\mathbf{v_2}$ will lead robot R_2 to its desired position R_2^d when $\mathbf{v_1}$ is fixed.

Proof. From Lemma 1, $\mathbf{v_2}$ will approach $\mathbf{v_2^d}$, so that we can assume $\mathbf{v_2}$ as (from (4))

$$\mathbf{v_2} = \mathbf{v_1} + \mathbf{v_{\Delta l}}. \tag{10}$$

The velocity difference between robot R_2 and robot R_1 is $\mathbf{v_2} - \mathbf{v_1} = \mathbf{v_{\Delta l}}$. The distance between R_2 and R_2^d is $\Delta\mathbf{l}$. Velocity $\mathbf{v_{\Delta l}}$ affects $\Delta\mathbf{l}$ by

$$\Delta\mathbf{l}' = -\mathbf{v_{\Delta l}}. \tag{11}$$

From (3), (11) and by linear differential, $\Delta\mathbf{l}$ will be

$$\Delta\mathbf{l}(t) = c_2 e^{-\alpha_1 t}, \tag{12}$$

where c_2 is a constant. When $t \to \infty$, $\Delta\mathbf{l}(t) \to 0$. In other words, $\mathbf{l} = \mathbf{l^d}$ (from (2)). It is proved that $\mathbf{v_2}$ will lead robot R_2 to position R_2^d. □

Lemma 3 (Angle and Scale Convergence). Using (5) and (6), $\mathbf{v_2}$ will converge toward $\mathbf{v_1}$, so that a successor's progress direction and velocity scale converge toward its predecessor.

Proof. From Lemma 2 and (3), when $\Delta\mathbf{l} \to 0$, $\mathbf{v_{\Delta l}} \to 0$, so that $\mathbf{v_2} = \mathbf{v_1}$ (from (10)). Therefore, it is proved that a successor's progress direction is the same with its predecessor's direction, and its velocity scale is also the same. □

Theorem 1 (Successor's Convergence). Using robot control of (5) and (6), a successor will maintain formation with its predecessor when the velocity of the predecessor is fixed.

Proof. From Lemma 2, a successor will go to its desired position. From Lemma 3, a successor's angle and velocity scale will converge toward its predecessor. Therefore, a successor will maintain formation with its predecessor. □

Theorem 2 (Formation Convergence). Using robot control of (5) and (6), the whole formation will be maintained when the leader's velocity is fixed.

Proof. From Theorem 1, the leader's successors will maintain formation toward the leader, and these successors' successors will maintain formation toward these successors by recursion. Therefore, the whole formation will be maintained. □

Remark. If the leader's velocity is not fixed, the robot control can not guarantee that the successor's formation is maintained. For example, when a predecessor's angular velocity is fixed and not zero (run a circle), the successor's position will always keep a constant distance to its desired position. In this situation, the successor's formation is not maintained.

5 VOMAS and Multiple Robot Teams

To design and control multiple robot teams, we propose an architecture, called Virtual Operator Multi-Agent System (VOMAS), to control multiple teams. We describe the VOMAS architecture first and then describe the split and merge for multiple robot teams.

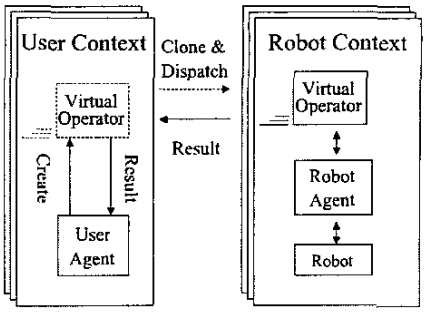

Figure 7: VOMAS Architecture.

5.1 VOMAS

The VOMAS architecture is shown in Figure 7. The main concept of VOMAS is a multi-agent system, especially a mobile agent [12, 13]. VOMAS architecture is designed to control multiple robots with multiple missions for multiple users. User contexts are users' processing environments, maybe computers or PDAs. Robot contexts are generally physical robots. The robot dependent control or low level functions are designed in a robot agent. The robot independent or user dependent functions and high level missions are designed in a virtual operator. These agents are described as follows.

User Agent: A user can control the mobile robot through a user agent with a friendly graphical user interface (GUI). When the user wants to control a robot, the user agent needs to check whether or not the robot is controlled by other virtual operator. If it is available, the user agent `creates` a virtual operator to monitor and control this robot. At last, when the user does not need the robot to execute missions any more, the user agent then `disposes` the virtual operator, and the robot is available for other users.

Virtual Operator: A virtual operator is created by a user agent to receive, execute, and monitor missions for a user. When the virtual operator accepts a mission, it will `clone` itself and send the cloned agent to the robot context to cooperate with the robot agent to accomplish this mission. When the virtual operator finishes its mission, it sends executed results to the user agent and then disposes itself. In formation control, a virtual operator is generally designed to execute high level missions, which need the whole formation team to execute. For example, we can assign a virtual operator to the leader of a team to attack some position.

Robot Agent: Robot dependent functions or low-level functions are designed in a robot agent. The architecture used in the robot agent is the behavior-based control architecture, because behavior-based control can support reactive control. In formation control, robot control of maintaining formation is designed in a robot agent.

The advantages of using VOMAS architecture is to have the capability of dynamically changing executing missions. When a virtual operator executes its mission, the user, or a control center, can dispose current virtual operator and create another virtual operator to execute another mission on the robot. If we use this capability in formation, we can get benefits in joining a robot into or removing a robot from a team. In another aspect, a formation, whose leader is controlled by a virtual operator, can be split into two or more for-

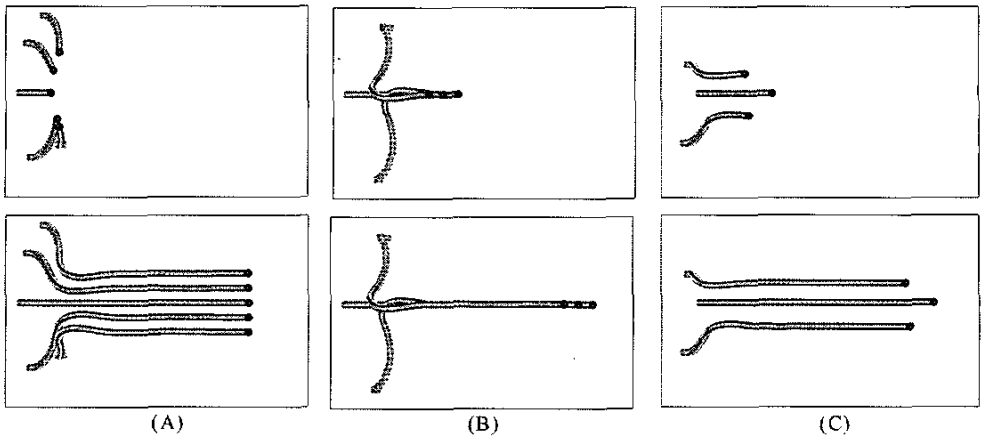

Figure 8: Basic Formations. (A)Line, (B)Column, (C)Wedge.

mation teams by assigning several new leaders and creating several virtual operators to control these leaders. In the following subsections, we describe joining, removing, and split.

5.2 Joining and Removing

When a robot joins into a formation team, the leader virtual operator, which controls the leader robot, assigns another virtual operator, called follower virtual operator, to control the newly robot. The follower virtual operator knows its unique ID, the formation of the whole team (formation representation), high level missions of this formation, and other knowledge about the formation. The follower virtual operator moves to the robot context and cooperates with the robot agent to maintain formation. Conversely, when a robot is removed from a formation, the follower virtual operator is disposed, and its ID is released. Other followers may need to modify their ID and make some changes.

5.3 Split and Mergence

A formation can be split into two or more formation teams. For examples, if the army attacks two positions of the enemy, the formation of the army should split into two formation teams when it approaches the enemy place. In such case, we can assign another leader virtual operator to a suitable robot, so that all its successors will follow it to the newly assigned place to execute another mission. After finishing missions, they can merge together and go back to the original place.

6 Implementation

In this section, we implement several different formations in simulation. The robots in the simulator are circular holonomic robots.

6.1 Basic Formations

From Figure 8, it shows three basic formations, and from left to right are *line*, *column*, and *wedge* formations. The top row shows the simulation results running for a while and it may not very stable. The bottom row shows the stable formation results for a longer time.

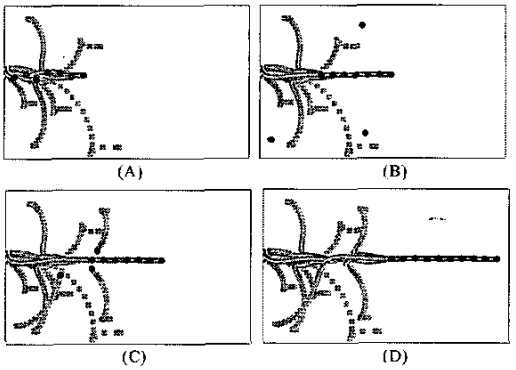

Figure 9: Joining Robots into a Team. (A)Seven robots to form a column formation, (B)Adding three robots, (C)Automatically joining into the formation, (D)To form a column.

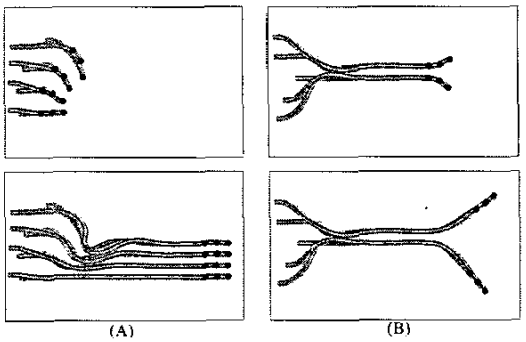

Figure 10: Platoon Formation and Split. (A)Platoon formation, (B) Formation split and approaching two targets.

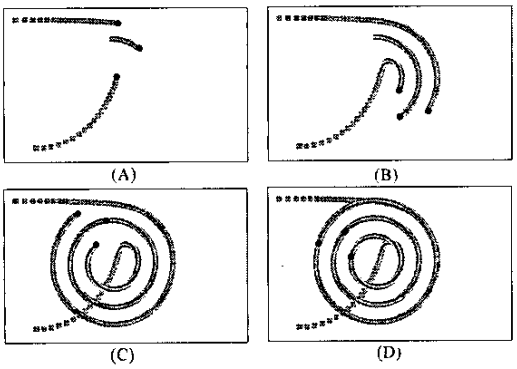

Figure 11: A Wedge Formation Running Circles. (A)(B)(C)From start to running a Circle, (D)Tracks after running several circles.

6.2 Joining Robots

By the formation representation in Section 2, it's easy to add newly robots into the same team and to maintain the same formation, and the simulation results are shown in Figure 9. In Figure 9(A), it shows that seven robots can form a column formation. In Figure 9(B), we randomly add three robots into the team in the simulation. In Figure 9(C), these three robots automatically join into the team. In Figure 9(D), ten robots maintain the column formation stably.

6.3 Platoon and Split

A platoon formation is formed in Figure 10(A). This formation is maintained by predecessor reference, so that the stability is worse than other above experimental results, which use leader reference. In Figure 10(B), it shows that the platoon formation splits into two formations (the top figure) to approach two targets (the bottom figure).

6.4 Running Circles

If we let the leader robot runs at constant angular velocity, the track of the leader will be a circle. In Figure 11, the three robots form a wedge formation. When the leader travels in a circle, other robots also follow the leader to form a circle with different radiuses (shown in Figure 11(A)(B)(C)). Figure 11(D) shows that the tracks after running for several circles.

7 Conclusion

A simple formation representation is proposed to present arbitrary formations or regular infinite formations. To maintain formation, a kinematic control method of mobile robots is described and proved convergence. Leader reference, predecessor reference, and soft-neighbor reference models are used to maintain formation. In implementation, it shows several simple formations, such as line, column, wedge formations. Using VOMAS architecture, it can also do flexible control for multiple teams, such as joining, removing, and split. In the future work, we will realize these concepts into physical robots and investigate a formal method to evaluate formation stability. We will also investigate how the formation can imporve the throughput of ITS.

References

[1] T. Arai, E. Pagello, and L. E. Parker, "Guest editorial: Advances in multirobot systems," *IEEE Transaction on Robotics and Automation*, vol. 18, no. 5, pp. 655–661, Oct. 2002.

[2] B. McMillin and K. L. Sanford, "Automated highway systems," *IEEE Potentials*, vol. 17, no. 4, pp. 7–11, Oct. 1998.

[3] J. Lygeros, D. N. Godbole, and S. Sastry, "Verified hybrid controller for automated vehicles," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 522–539, Apr. 1998.

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

[5] J. Fredslund and M. J. Mataric, "A general algorithm for robot formations using local sensing and minimal communication," *IEEE Transaction on Robotics and Automation*, vol. 18, no. 5, pp. 837–846, Oct. 2002.

[6] A. K. Das, P. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Transaction on Robotics and Automation*, vol. 18, no. 5, pp. 813–825, Oct. 2002.

[7] J. P. Desai, "Modeling multiple teams of mobile robots: A graph theoretic approach," in *Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 381–386.

[8] R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A coordination architecture for spacecraft formation control," *IEEE Transactions on Control Systems Technology*, vol. 9, no. 6, pp. 777–790, Nov. 2001.

[9] K. Sugihara and I. Suzuki, "Distributed algorithms for formation of geometric patterns with many mobile robots," *Journal of Robotic Systems*, vol. 13, no. 3, pp. 127–139, Mar. 1996.

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

[11] J. P. Desai, J. Ostrowski, and V. Kumar, "Controlling formations of multiple mobile robots," in *Proceeding of the 1998 IEEE International Conference on Robotics and Automation (ICRA'98)*, May 1998, pp. 2864–2869.

[12] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Press: New York, 1999.

[13] D. Lange and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley Press, 1998.