CHAPTER 9

Multi-Robotic Systems

The topic of multi-robotic systems is quite popular now. It is believed that such systems can have the following benefits:

- Improved performance ("winning by numbers")
- Distributed sensing
- Distributed actuation
- Fault tolerance

On the other hand, such systems also introduce new problems such as interference, communication cost and uncertainty.

In this chapter we focus on cooperative systems.

9.1. Consensus problem

We begin by considering a flock of N birds. Each bird flies with the same speed but with possibly different directions. Namely

$$v_i = (v\cos\theta_i, v\sin\theta_i)^T,$$

where θ_i is the heading of bird i, and for the sake of simplicity, we assume the birds fly on a plane.

Now suppose for each bird, it changes its heading by the following model:

$$\dot{\theta}_i = u_i,$$

or

$$\theta_i(t+1) = \theta_i(t) + u_i(t) := \omega_i(t).$$

An interesting question is how each bird should update its heading so eventually we have

$$\theta_1(t) = \cdots = \theta_N(t).$$

It turns out

$$\omega_i(t) = \frac{1}{N}\sum_{j=1}^{N}\theta_j(t)$$

will do the trick. One thing we should be careful here is to calculate the average heading properly. We can rewrite the control as

$$\omega_i(t) = \theta_i(t) + \frac{1}{N}\sum_{j\neq i}(\theta_j(t) - \theta_i(t)),$$

or,

$$u_i(t) = \frac{1}{N}\sum_{j\neq i}(\theta_j(t) - \theta_i(t)). \tag{9.1}$$

If we use a local coordinate $\theta_i \in (-\pi, \pi]$, then $\omega_i(t)$ can be calculated properly.

A problem with the controller (9.1) is that each bird needs to know the relative headings of all the other birds. In the following we show that this is not necessary.

Now we consider a system of N agents:

$$\dot{x}_i = u_i,\ i = 1, \cdots, N \tag{9.2}$$

where x_i can be viewed as heading, position or other quantities.

We define the consensus problem as follows:

Consensus problem:

Find $u_i(t)$ such that as $t \to \infty$ we have

$$x_1(t) = x_2(t) = \cdots = x_N(t),$$

here we assume that agent i can only detect relative errors $x_j - x_i$ of its neighbors, namely $j \in N_i$.

Similar to the flocking problem, we consider a controller of the following type:

$$u_i(t) = \sum_{j\in N_i} a_{ij}(x_j - x_i), \tag{9.3}$$

where a_{ij} are positive weights. If we let $x = (x_1, \cdots, x_N)^T$, then

$$\dot{x} = -Lx, \tag{9.4}$$

where

$$L = D - A = diag(\sum_{j\neq 1} a_{1j}, \cdots, \sum_{j\neq N} a_{Nj}) - [a_{ij}].$$

Now define

$$V(x) = x^T Lx = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i} a_{ij}(x_j - x_i)^2.$$

Proposition 9.1. *The consensus problem is solved, namely as $t \to \infty$, $x_1(t) = \cdots = x_N(t)$ in (9.4), if and only if*

$$V(x) = 0 \iff x_1 = x_2 = \cdots = x_N. \tag{9.5}$$

In fact, in this case

$$\lim_{t\to\infty} x_i(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(0).$$

9.1.1. Connection to graph theory. An undirected graph $\mathcal{G}$ of order N consists of a vertex set $\mathcal{V} = \{1, 2, \cdots, N\}$ and an edge set $\mathcal{E} = \{(i,j) : i, j \in \mathcal{V}\} \subset \mathcal{V} \times \mathcal{V}$. A weighted adjacency matrix $A = [a_{ij}] \in R^{N\times N}$, where $a_{ii} = 0$ and $a_{ij} = a_{ji} \geq 0$. $a_{ij} > 0$ if and only if there is an edge between agent i and agent j (i.e., $a_{ij} = a_{ji} > 0 \Leftrightarrow (i,j) \in \mathcal{E}$) and the two agents are called adjacent (or they are mutual neighbors). The set of neighbors of vertex i is denoted by $\mathcal{N}_i = \{j \in \mathcal{V} : (i,j) \in \mathcal{E},\ j \neq i\}$. The Laplacian

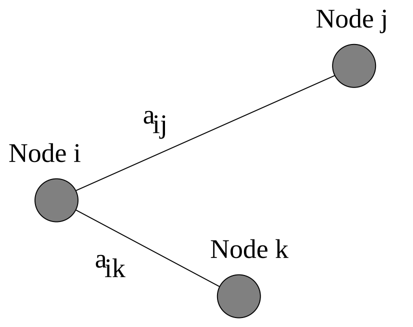

FIGURE 1

$L = [l_{ij}] \in R^{N\times N}$ of a weighted graph $\mathcal{G}$ is defined as

$$l_{ij} = \begin{cases} \sum\limits_{k\neq i} a_{ik}, & j = i, \\ -a_{ij}, & j \neq i. \end{cases}$$

By the definition, every row sum of L is zero.

If there is a path between any two vertices of a graph $\mathcal{G}$, then $\mathcal{G}$ is *connected*, otherwise disconnected.

The following lemma shows some basic properties of the Laplacian L.

LEMMA 9.2. *Let L be the Laplacian of an undirected graph $\mathcal{G}$ with N vertices, $l_1 \leq \cdots \leq l_N$ be the eigenvalues of L. Then*

(1) *0 is an eigenvalue of L and $\mathbf{1}_N$ is the associated eigenvector, where $\mathbf{1}_N = (1, 1, \cdots, 1)^T$. Namely, $L\mathbf{1}_N = 0$;*
(2) *If $\mathcal{G}$ is connected, then 0 is the algebraically simple eigenvalue of L and $l_2 = \min\limits_{\xi\neq 0, \xi\perp \mathbf{1}_N} \frac{\xi^T L \xi}{\xi^T \xi} > 0$, which is called the algebraic connectivity of $\mathcal{G}$;*
(3) *If 0 is the simple eigenvalue of L, then it is an n multiplicity eigenvalue of $L \otimes I_n$ and the corresponding eigenvectors are $\mathbf{1}_N \otimes e_i$, $i = 1, \cdots, n$.*

Now let us go back to the consensus problem.

PROPOSITION 9.3. *The consensus problem is solved if and only if the associated graph is connected.*

Now let us consider a set general linear system:

$$\dot{x}_i = Ax_i + Bu_i, \quad i = 1, \cdots, N \tag{9.6}$$

where $x_i \in R^n,\ u_i \in R^m$. Define the relative state error for agent i as

$$z^i = \sum_{j \in \mathcal{N}_i} (x_i - x_j).$$

The consensus is said to be achieved using local information if there is a local state error feedback control

$$u_i = Kz^i$$

such that

$$\lim_{t \to \infty} \|x_i - x_j\| = 0, \quad \forall i,\ j.$$

After plug in the control, it is now well known that the consensus problem is equivalent to the simultaneous stabilization problem of the following systems:

$$\dot{x}_i = Ax_i + \lambda_i BK, \quad i = 2, \cdots, N, \tag{9.7}$$

where $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N$ are eigenvalues of the graph Laplacian L. Note that the graph is assumed to be undirected and connected.

PROPOSITION 9.4. *Consider a finite set of linear systems*

$$\dot{x}_i = Ax_i + \alpha_i Bu_i, \quad i = 1, \cdots, N, \tag{9.8}$$

where (A, B) is controllable, and $\alpha_i > 0$. Then there exists

$$u_i = Kx_i$$

such that $A + \alpha_i BK$ is Hurwitz for $i = 1, \cdots, k$.

This proposition can be easily proven as follows. Assume α_1 is the smallest. Since (A, B) is controllable, the algebraic Riccati equation

$$A^T P + PA - \alpha_1 PBB^T P + Q = 0$$

where $Q > 0$, has a solution $P > 0$. Thus for $K = -B^T P$ we have

$$(A - \alpha_i BB^T P)^T P + P(A - \alpha_i BB^T P) = -Q - (2\alpha_i - \alpha_1) PBB^T P$$

$i = 1, \cdots, N$.

Based on the proposition, we can show easily that as long as (A, B) is controllable and the associated graph is connected, the consensus can be reached. In this case, the consensus state is in general not a constant, it is governed by the following equation:

$$\dot{\bar{x}} = A\bar{x}.$$

9.1.2. Consensus and distributed optimization. [1] One application of multi-agent consensus is to do optimization in a decentralized way. For large-scale networked complex systems, since the lack of centralized authority, it is difficult to do optimization in a normal way. For standard iterative methods, global information may needed for each node/agent in the network at each iteration. However, those global information is extremely difficult to obtain in many practical large-scale networks, e.g., Internet, sensor networks, and economic networks. The idea of distributed optimization is to use local communication and cooperation to achieve optimal/suboptimal solutions to the global objective.

Consider a network with n cooperative agents towards a single objective:

$$\min \sum_{i=1}^{n} f_i(x) \tag{9.9}$$

Can each agent reach the desired global minimizer by using only local information, namely by communicating with only neighboring agents in the network?

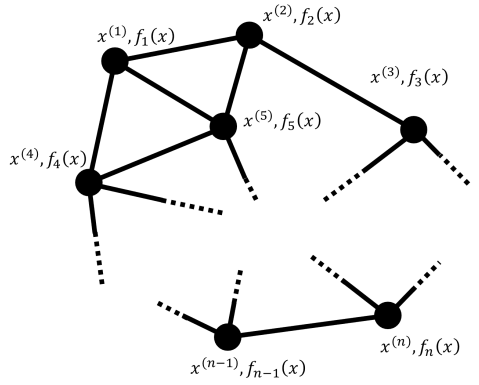

FIGURE 2. A communication network where x is the global state variable, $x^{(i)}$ is the local estimation of the optimal solution for agent i, and $f_i(x)$ is the local cost for agent i.

The standard steepest descent method tells that the update of the variable x should be

$$x^{(i)}(k+1) = x^{(i)}(k) - \alpha_k^i \sum_{i=1}^{n} \nabla f_i(x^{(i)}(k))^T, \tag{9.10}$$

where α_k^i is the step-length chosen by some line-search algorithm. However, the update requires all the gradient information $\nabla f_i(x^{(i)}(k))$ for each agent, which can be a problem in practice. One approach to avoid using all gradients is called the two-step approach:

[1]The contribution by Yuecheng Yang and Wendy Song is acknowledged.

- step 1: Each agent take step in gradient descent direction with the same step-length:

$$\hat{x}^{(i)}(k+1) = x^{(i)}(k) - \alpha_k \nabla f_i(x^{(i)}(k))^T. \tag{9.11}$$

- step 2: Taking the network-wide average:

$$x^{(i)}(k+1) = \frac{1}{n}\sum_{j=1}^{n} \hat{x}^{(j)}(k). \tag{9.12}$$

- The combination of the two steps recovers that standard gradient method with the same step length $\frac{\alpha_k}{n}$:

$$\begin{aligned} x^{(i)}(k+1) &= \frac{1}{n}\sum_{j=1}^{n}\Big(x^{(j)}(k) - \alpha_k \nabla f_j(x^{(j)}(k))^T\Big) \\ &= \frac{1}{n}\sum_{j=1}^{n} x^{(j)}(k) - \frac{\alpha_k}{n}\sum_{j=1}^{n}\nabla f_j(x^{(j)}(k))^T \\ &= x^{(i)}(k) - \frac{\alpha_k}{n}\sum_{j=1}^{n}\nabla f_j(x^{(j)}(k))^T. \end{aligned}$$

Note that network-averaging is possible with peer-to-peer communication within a connected graph (consensus). Therefore, this approach is a distributed steepest descent algorithm requires only local gradient information.

The two step approach is a distributed algorithm and it converges due to the convergence of both the steepest descent algorithm and the consensus algorithm. However, within each optimization step, the agents need to exchange enough information to reach network average in order to move on to the next iteration. The structure of the network will limit the convergence rate of the algorithm. Unfortunately, the agents have to wait until they reach consensus because the errors will accumulate during each iteration otherwise. One extreme algorithm that shorten the averaging time is to do gradient based optimization and neighborhood averaging simultaneously. This results to the following interleaved version of the approach:

$$x^{(i)}(k+1) = \frac{1}{\|\mathcal{N}_i\|}\sum_{j\in\mathcal{N}_i} x^{(j)}(k) - \alpha\nabla f_i(x^{(i)}(k))^T, \tag{9.13}$$

for a fixed step-length α, where $\mathcal{N}_i$ denotes the set of neighboring agents of i. The convergence of this algorithm can be proven.However, the limit point may not be the global minimizer. The bounds for the error can be found in the literature [**14**].

Example

The objective function is $\sum_{i=1}^{3} f_i(x) = 3x^2 - 7x + 6$ and the global optimal solution is

$$x^* = 7/6 \approx 1.1667.$$

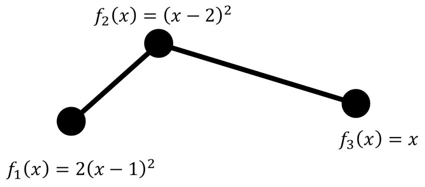

FIGURE 3

The two-step algorithm (9.11) & (9.12), gives the following iterates with the fixed step-length $\alpha = 0.1$:

Iteration	0	1	2	3	4	5	10	20	30	50
$x^{(1)}$	1	1.0333	1.0600	1.0813	1.0984	1.1121	1.1488	1.1647	1.1665	1.1667
$x^{(2)}$	1	1.0333	1.0600	1.0813	1.0984	1.1121	1.1488	1.1647	1.1665	1.1667
$x^{(3)}$	1	1.0333	1.0600	1.0813	1.0984	1.1121	1.1488	1.1647	1.1665	1.1667

The interleaved algorithm (9.13), gives the following iterates with the fixed step-length $\alpha = 0.1$:

Iteration	0	1	2	3	4	5	10	20	50	100
$x^{(1)}$	1	1	1.1000	1.1067	1.1319	1.1424	1.1833	1.2074	1.2127	1.2128
$x^{(2)}$	1	1.2000	1.1933	1.2424	1.2584	1.2809	1.3391	1.3750	1.3829	1.3830
$x^{(3)}$	1	0.9500	0.9500	1.9717	1.0071	1.0327	1.1191	1.1714	1.1829	1.1830

One can calculate the equilibrium of the iteration by solving the following equations:

$$\begin{aligned}\frac{x^{(1)}+x^{(2)}}{2} - 4\alpha(x^{(1)}-1) &= x^{(1)}\\ \frac{x^{(1)}+x^{(2)}+x^{(3)}}{3} - 2\alpha(x^{(2)}-2) &= x^{(2)}\\ \frac{x^{(2)}+x^{(3)}}{2} - \alpha &= x^{(3)}\end{aligned}$$

For positive α, the solution to this system of equations is

$$\begin{aligned}x^{(1)} &= \frac{48\alpha^2+18\alpha}{48\alpha^2+14\alpha}\\ x^{(2)} &= \frac{80\alpha^2+18\alpha}{48\alpha^2+14\alpha}\\ x^{(3)} &= \frac{-96\alpha^3+52\alpha^2+18\alpha}{48\alpha^2+14\alpha}\end{aligned}$$

Hence, the agents will not reach consensus when $\alpha > 0$. When $\alpha \to 0$, the algorithm converges to a solution $x^{(1)} = x^{(2)} = x^{(3)} = 9/7$, which is not the theoretical global optimal.

9.2. Formation control

In the maturing field of mobile robot control, a natural extension to the traditional trajectory tracking problem is that of *coordinated tracking*. In its most general formulation, the problem is to find a coordinated control scheme for multiple robots that make them maintain some given, possibly time-varying, formation at the same time as the robots, viewed as a group, executes a given task. The possible tasks could range from exploration of unknown environments, navigation in hostile environments where multiple robots make the system redundant and thus robust, to coordinated path following.

The multi-agent system that we consider is given by m mobile robots, each of which is governed by its own set of system equations

$$\begin{aligned} \dot{\mathbf{z}}_i &= f_i(\mathbf{z}_i) + g_i(\mathbf{z}_i)\mathbf{u}_i \\ \mathbf{x}_i &= h(\mathbf{z}_i), \end{aligned} \tag{9.14}$$

where $\mathbf{z}_i \in \Re^{p_i}$ is the state, $\mathbf{u}_i \in \Re^{k_i}$ is the control, and $\mathbf{x}_i \in \Re^n$ are geometric variables used for defining the formation in $\Re^n$. The m robots should keep a certain relative position and orientation, while moving along a given path. If the formation keeping is perfect, then we can treat the polygon formed by having the robots as vertices as a rigid body and the problem can be restated as that we want to keep the formation while letting a chosen point of the polygon x_0 (ex. the geometric center) move along a given path.

Let us first define what we mean by a formation:

Definition 9.1. **(Formation Constraint Function)** *Given a differentiable, positive definite ($F = 0$ only at one point) map $F : \Re^n \times \ldots \times \Re^n \to \Re^+$. If $F(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ is strictly convex, then we say that $F(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ is a formation constraint function. The shape and orientation of the robot formation is uniquely determined by $(\mathbf{x}_1, \ldots, \mathbf{x}_m) = F^{-1}(0)$.*

It is obvious that for a given formation, the corresponding formation constraint function is not unique. For example, for a given polygon in $\Re^2$, one can choose either

$$F = \sum_{i=2}^{m} [(\|\mathbf{x}_{i-1} - \mathbf{x}_i\|^2 - d_i)^2 + (\|\mathbf{x}_i\|^2 - r_i)^2] + \|\mathbf{x}_1 - \mathbf{a}_1\|^2$$

or

$$F = \sum_{i=1}^{m} \|\mathbf{x}_i - \mathbf{a}_i\|^2.$$

From the implementation point of view, the former is preferable since the relative distance is coordinate-free and easier to measure than the absolute position.

We, of course, want to allow for the possibility of having a moving formation, and we thus need to specify a motion for the *virtual leader*, $\mathbf{x}_0$. We choose to parameterize the trajectory for $\mathbf{x}_0$ as

$$\mathbf{x}_0 = p_0(s_0), \tag{9.15}$$

where we assume that the trajectory is smooth, i.e. $\|\frac{\partial p_0(s_0)}{\partial s_0}\| \neq 0$ for all s_0.

The reason for calling $\mathbf{x}_0$ together with its dynamics a virtual leader is because of the role it plays. Using this terminology, our additional task is to design m new virtual robots for the individual robots to follow. We are thus free to design the evolution of these additional virtual vehicles, and we ignore the question concerning how to actually track these new virtual vehicles for the time being.

In light of the previous paragraph, it is more convenient to consider a moving frame with coordinates centered at $\mathbf{x}_0$. In the new coordinates we have $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_0$. Let the desired trajectories (subscript d), or virtual vehicles, be defined in the moving frame by

$$\begin{aligned} \tilde{\mathbf{x}}_{id} &= p_i(s_i), i = 1, \dots, m \\ \dot{\tilde{\mathbf{x}}}_{id} &= \frac{\partial p_i(s_i)}{\partial s_i}\dot{s}_i, \end{aligned} \tag{9.16}$$

where $\frac{\partial p_i(s_i)}{\partial s_i}$ and $\dot{s}_i \in \Re$ should be chosen in a systematic fashion so that the formation constraint is respected.

The solution we propose is to let the desired trajectories be given by the steepest descent direction to the desired formation, i.e., we set

$$\frac{\partial p(s)}{\partial \mathbf{s}} = -\nabla F(\tilde{\mathbf{x}}_d), \tag{9.17}$$

where we have grouped the contributions from the different robots together as

$$\begin{aligned} \nabla F(\tilde{\mathbf{x}}_d)^T &= \left(\frac{\partial F(\tilde{\mathbf{x}}_d)}{\partial \tilde{\mathbf{x}}_{1d}}^T, \dots, \frac{\partial F(\tilde{\mathbf{x}}_d)}{\partial \tilde{\mathbf{x}}_{md}}^T\right) \\ p(\mathbf{s})^T &= (p_1^T(s_1), \dots, p_m^T(s_m)) \\ \mathbf{s}^T &= (s_1, \dots, s_m) \\ \tilde{\mathbf{x}}_d^T &= (\mathbf{x}_{1d}^T - \mathbf{x}_0^T, \dots, \mathbf{x}_{md}^T - \mathbf{x}_0^T). \end{aligned} \tag{9.18}$$

REMARK 9.1. *Equation* (9.17) *defines a group of ordinary differential equations with respect to* $\mathbf{s}$. *Since* $F(\tilde{\mathbf{x}}_d)$ *is well defined (in many cases just a polynomial), calculating* (9.17) *online is not a problem.*

The idea now is to let the evolution of the different virtual vehicles be governed by differential equations containing error feedback in order to make the control scheme robust. This idea can be viewed as a combination of the conventional trajectory tracking, where the reference trajectory is parameterized in time, and a dynamic path following approach, where the criterion is to stay close to the geometric path, but not necessarily close to an *a priori* specified point at a given time.

We should point out that even using the same methodology, an alternative design would be that one only designs the dynamics for the virtual leader, and then uses the formation constraint (can be viewed as a rigid body constraint) to specify the motion of the other virtual vehicles. The reason for designing virtual vehicles individually here is that we, by actively controlling the evolution of the reference points, gain additional control power, and from the implementation point of view, this is more robust with respect to measurement errors and uncertainties in localization. Although with this approach, the formation constraint is not necessarily respected initially by the virtual vehicles, we will show that they converge to the exact formation exponentially fast, provided the actual tracking errors are bounded.

In order to accomplish this, we define the evolution of the reference points as

$$\dot{s}_i = ce^{-\alpha_i \rho_i}, \ i = 1, \dots, m, \tag{9.19}$$

where $c, \alpha_i > 0$ and $\rho_i = \|\mathbf{x}_i - \mathbf{x}_{id}\| = \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{id}\|$. As already mentioned, we want the motion of s_0 to capture how well the formation is being respected. Define

$$\rho_a = \sum_{i=1}^{m} \rho_i$$

and set

$$\dot{s}_0 = \frac{c_0}{\|\frac{\partial p_0(s_0)}{\partial s_0}\|} e^{-\alpha_0 \rho_a}. \tag{9.20}$$

With these designs we have the following stability theorem:

Theorem 9.5. **(Coordinated Tracking and Formation Control)** *Under the assumption that the real robots track their respective reference trajectory perfectly, it holds that*

$$\lim_{t \to \infty} F(\tilde{\mathbf{x}}_d) = 0. \tag{9.21}$$

Remark 9.2. *This theorem shows that we have quite some freedom in initializing the virtual vehicles and the algorithm is robust to measurement noises.*

Proof:

$$\frac{d}{dt} F(\tilde{\mathbf{x}}_d) = \nabla F(\tilde{\mathbf{x}}_d)^T \dot{\tilde{\mathbf{x}}}_d = -\sum_{i=1}^{m} \|\frac{\partial F}{\partial \tilde{\mathbf{x}}_{id}}\|^2 ce^{-\alpha_i \rho_i}. \tag{9.22}$$

Now assume that we have perfect tracking, i.e. $\rho_i = 0, \ i = 1, \dots, m$. This assumption, combined with the assumption that F is positive definite and convex, implies that $\frac{d}{dt} F(\tilde{\mathbf{x}}_d)$ is negative definite since otherwise F would have a local minima. This concludes the proof. ∎

Proposition 9.6. *If all the tracking errors are bounded, i.e. it holds that $\rho_i \le \rho < \infty, \ i = 1, \dots, m$, then*

$$\lim_{t \to \infty} F(\tilde{\mathbf{x}}_d) = 0. \tag{9.23}$$

The proof of this corollary is just a straight forward extension of the proof of the previous theorem. This corollary is furthermore very useful since one typically does not want $\rho = 0$ due to the potential chattering that such a control strategy might give rise to. Instead it is desirable to let $\rho > 0$ be the look-ahead distance at which the robots should track their respective reference trajectories.

9.3. Formation control with limited sensor information

Many of the already available control algorithms for a robot tracking a moving target are designed to work under ideal conditions and with perfect sensor data. Much effort is still needed in order to design robust algorithms that can operate in real time and tolerate large measurement errors without needing to use computationally heavy filtering algorithms before data can be used.

Let us assume that there are n mobile robots, $R_1, ..., R_n$, where R_1 is the leader of the formation or the target. We assume that the motion of any given robot, $R_i, i > 1$, can be modeled as

$$\begin{aligned} \dot{x}_i &= v_i \cos\phi_i \\ \dot{y}_i &= v_i \sin\phi_i \\ \dot{\phi}_i &= \omega_i, \end{aligned} \tag{9.24}$$

where x_i and y_i are the position coordinates and ϕ_i denotes the orientation angle of the robot with respect to a global coordinate system.

If the control aim is to keep a fixed distance and bearing angle to the neighbor closest ahead, we end up with a line formation as in fig. 4.

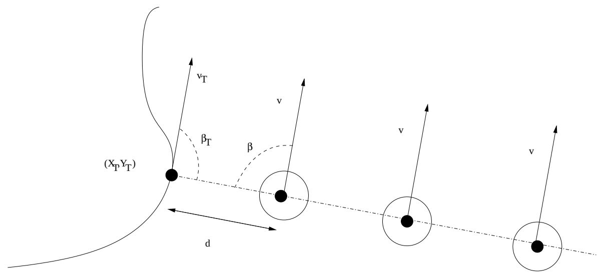

FIGURE 4. Horizontal tracking and formation keeping.

Now let d_0 be the desired distance between two robots while d_i is the actual distance between R_i and R_{i-1} as measured by the sensors on R_i. In the same way, let $0 \leq \beta_0 \leq \frac{\pi}{2}$ be the desired relative angle between two robots (see fig. 4) while β_i is the corresponding measured value for R_i, i.e., the measured angle to R_{i-1} with respect to R_i's angle of orientation. We also introduce the notation $\gamma_i = \phi_i - \phi_{i-1}$ to denote the difference in orientation between two adjacent robots. For $i = 2, ..., n$, the system (9.24) can be rewritten in terms of the measured units d_i and β_i as

$$\begin{aligned} \dot{d}_i &= -v_i \cos\beta_i + v_{i-1}\cos(\gamma_i + \beta_i) \\ \dot{\gamma}_i &= \omega_i - \omega_{i-1} \\ \dot{\beta}_i &= -\omega_i + \frac{v_i}{d_i}\sin\beta_i - \frac{v_{i-1}}{d_i}\sin(\gamma_i + \beta_i). \end{aligned} \tag{9.25}$$

This is apparently a cascaded system due to the appearance of v_{i-1} and ω_{i-1}. Our control objective can now be defined as :
Given $v_1(t)$ and $\omega_1(t)$, find control $v_i(t)$ and $\omega_i(t)$ such that for $i = 2, ..., n$

$$d_i \to d_0, \gamma_i \to 0, \beta_i \to \beta_0$$

as $t \to \infty$ and $0 \le \beta_0 \le \frac{\pi}{2}$.

Let $\Delta d_i = d_i - d_0$ and $\Delta\beta_0 - \beta_i$. We can then rewrite the system (9.25) as

$$\begin{aligned} \Delta\dot{d}_i &= -v_i\cos(\Delta\beta_i - \beta_0) + v_{i-1}\cos(\Delta\beta_i - \gamma_i - \beta_0) \\ \dot{\gamma}_i &= \omega_i - \omega_{i-1} \\ \Delta\dot{\beta}_i &= \omega_i + \frac{v_i\sin(\Delta\beta_i - \beta_0) - v_{i-1}\sin(\Delta\beta_i - \gamma_i - \beta_0)}{d_0 + \Delta d_i}, \end{aligned} \tag{9.26}$$

with the control objective: *Given $v_1(t)$ and $\omega_1(t)$, find control $v_i(t)$ and $\omega_i(t)$ such that for $i = 2, ..., n$*

$$\Delta d_i \to 0, \gamma_i \to 0, \Delta\beta_i \to 0$$

as $t \to \infty$ and $0 \le \beta_0 \le \frac{\pi}{2}$.

Note that when $\beta_0 = 0$ for all vehicles, the formation is a vertical line and when $\beta_0 = \frac{\pi}{2}$ the formation is a horizontal line. Apparently no simple controller can cover the whole range of β_0 from 0 to $\frac{\pi}{2}$. In the rest of this section we present two control algorithms that together will cover the whole range.

9.3.1. Serial formation control. Controlling the formation when β_0 is close or equal to zero is relatively easy, especially if we do not require $\gamma_i \to 0$. For $\beta_0 = 0$ it could, for example, be achieved by defining a reference point (x_0, y_0) on the robots axis of orientation at a distance d_0 from the center of the robot, so that

$$\begin{aligned} x_0 &= x_i + d_0\cos\phi_i \\ y_0 &= y_i + d_0\sin\phi_i. \end{aligned} \tag{9.27}$$

The reference point can then be driven towards (x_{i-1}, y_{i-1}) with a proportional control that is globally stable. Similarly, we can define the reference point to lie at an angle β_0 from the axis of orientation. The robot will then follow the target with a tracking angle β_0. By choosing the reference point as in this way we obtain not only a point that can be driven arbitrarily close to the target/leader without causing the angle to target to be undefined. We also obtain control of the robots orientation, since controlling

(x_0, y_0) towards the center of the target, (x_{i-1}, y_{i-1}), means simultaneously controlling the angle of orientation so that the relative angle to target, β_i, approaches β_0.

The expressions for the velocity v_i and angular velocity ω_i of Robot R_i, corresponding to the control described above are

$$\begin{aligned} v_i &= \frac{kd_i \cos(\beta_i - \beta_0) - kd_0 + v_{i-1}\cos(\gamma_i + \beta_0)}{\cos(\beta_0)} \\ \omega_i &= \frac{kd_i \sin(\beta_i) - kd_0 \sin(\beta_0) - v_{i-1}\sin(\gamma_i)}{d_0 \cos(\beta_0)}. \end{aligned} \tag{9.28}$$

Proposition 9.7. *With the control (9.28), $(x_0(t), y_0(t))$ converges to $(x_{i-1}(t), y_{i-1}(t))$ as $t \to \infty$ and $0 \leq \beta_0 < \frac{\pi}{2}$.*

The control is globally stable and therefore also very robust, but it will obviously become singular as β_0 approaches $\frac{\pi}{2}$.

9.3.2. Parallel formation control. When the desired bearing β_0 is equal to $\frac{\pi}{2}$, the serial formation control apparently does not work. In fact, parallel tracking is quite different from serial tracking in the sense that it is not really designed in order to *follow* a target, but rather to *predict* the motion of the target and to move accordingly.

The prediction of the targets motion can be done in more or less sophisticated ways, depending on which information that is available. With full information on the speed and angular velocity of the target, the robot can do a very good prediction of the target motion, under the assumption that the motion of the target is smooth. If the speed and angular velocity of the target is not directly accessible, estimations can be made based on measurements of the targets position from at least one step back in time. Estimations made from noisy sensor data tend to increase measurement errors in a very unfortunate way, and especially if the amount of data saved from previous time steps is small, the control algorithm tend to become quite sensitive to errors. A more robust but less sophisticated approach to the problem is to design a control with the main objective to correct any positioning errors, rather than trying to calculate the motion of the target. One example of this is the control algorithm we shall use in this paper.

Using the same notations for angles and distances as before, we propose our controls for the velocity v_i and angular velocity ω_i of Robot R_i as

$$\begin{aligned} \omega_i &= c_1 \Delta\beta_i - c_2(d_0 - d\sin(\beta_i)) - c_3\gamma_i \\ v_i &= c_4 v_{i-1} + c_5 \Delta d_i \cos(\beta_i) - c_6 \Delta\beta_i, \end{aligned} \tag{9.29}$$

where when $i = 2$ $v_1 = v_T$ is an estimate of the velocity of the target and $c_1, ..., c_6$ are positive constants of which c_4 should ideally be equal to one.

Proposition 9.8. *Suppose the target speed $v_T(t)$ and its time derivative are bounded, and $|v_T(t)| \geq v_0 > 0$, Then the cascaded system* (9.25) *with control* (9.29) *is locally exponentially stabilized around the equilibrium* ($d_i = d_0,\ \gamma_i = 0,\ \beta_i = \frac{\pi}{2}$.