1

Introduction

"The whole is more than the sum of its parts."

Aristotle

1.1 Motivation

This book is devoted to *multi-agent systems*. Since this term has different meanings within different research communities, we deem it necessary to precisely define the meaning used here. In this book, a multi-agent system refers to a network of interacting, mobile, *physical* entities that *collectively* perform a complex task beyond their individual capabilities.

Nature is replete with biological systems that fit this definition: a flock of birds, a school of fish, and a colony of insects (see Figure 1.1), to name a few. The behavior of such biological swarms is decentralized since each biological agent does not have access to global knowledge or supervision, but uses its own local sensing, decision, and control mechanisms.

Ants are a model example of a biological multi-agent system. Ant colonies share the common goals of surviving, growing, and reproducing. Their sense of community is so strong that they behave like a single "superorganism" that can solve difficult problems by processing information as a collection [1]. This collective behavior facilitates food gathering, defending nests against enemies, and building intricate structures with tunnels, chambers, and ventilation systems. Ants accomplish such feats without a supervisor telling them what to do. Rather, ant workers perform tasks based on personal aptitudes, communications with colony mates, and cues from the environment. Interactions with other ants and the environment occur via chemicals, which they sense with their antennae [1].

Nature is inspiring humans to *engineer* multi-agent systems that mimic this distributed, coordinated behavior. The agents in such engineering systems are not living beings, but machines such as robots, vehicles, and/or mobile

Formation Control of Multi-Agent Systems: A Graph Rigidity Approach, First Edition.
Marcio de Queiroz, Xiaoyu Cai, and Matthew Feemster.

Companion website: www.wiley.com/go/dequeiroz/formation_control

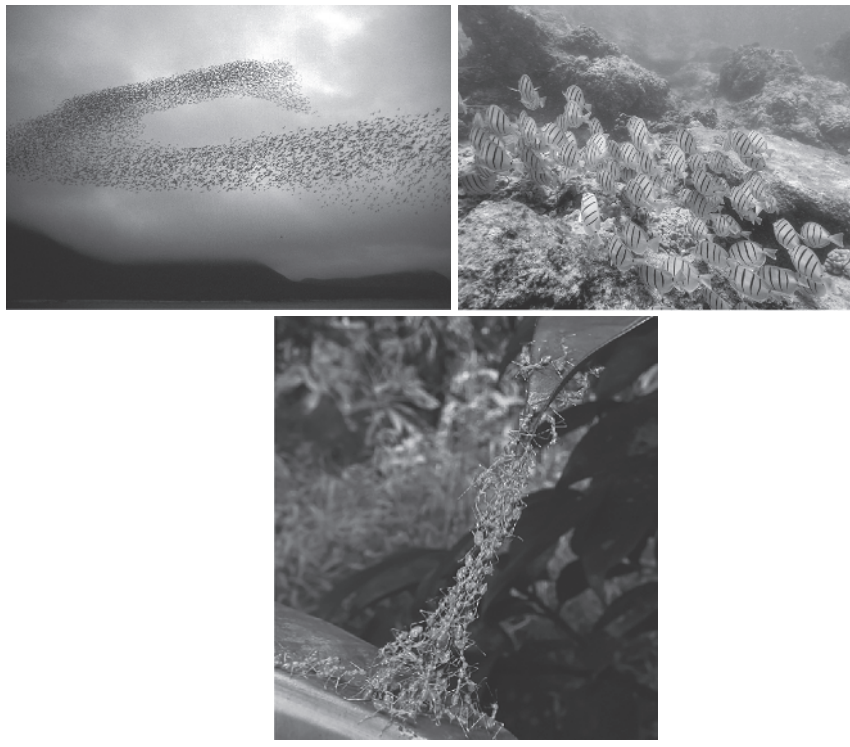

Figure 1.1 Examples of collective behavior in nature: a flock of birds (top left), a school of fish (top right), and a swarm of ants building a bridge (bottom).

sensors (see Figure 1.2). Recent advances in sensor technology, embedded systems, communication systems, and power storage now make it feasible to deploy such swarms of cooperating agents for various civilian and military applications. For instance, a group of autonomous (ground, underwater, water surface, or air) vehicles could be deployed in large disaster areas to perform search, mapping, surveillance, or environmental monitoring and clean up without putting first responders in harm's way. Some recent examples of such situations are Hurricane Katrina in 2005, the BP oil spill in the Gulf of Mexico in 2010, and the Fukushima nuclear disaster in 2011. Another application is a military mission where a group of unmanned air vehicles surround and intercept an intruding or evading aircraft or enemy combatants. Yet another potential application is a team of vehicles cooperatively transporting an object too large and/or heavy for a single vehicle to transport.

One may wonder why a multi-agent system should be used instead of a single "large agent". There are several advantages to doing so: more efficient and complex task execution, robustness when one or more agents fail, scalability,

Figure 1.2 Examples of engineering multi-agent systems.

versatility, adaptability, and lower cost [2]. For example, multiple agents could position themselves relative to each other to create a virtual, large-scale antenna with higher sensitivity to acoustic signals than would be possible with a single antenna. If one of the agents malfunctions, the remaining ones would reconfigure to keep the antenna operational, whereas the stand-alone antenna would be a single point of failure. Malfunctions of an agent are also less likely than in a single system because they are usually much simpler hardware- and software-wise. This simplicity, along with larger quantities, also leads to mass production at a low cost.

On the other hand, multi-agent systems introduce a host of unique challenges, including coordination and cooperation schemes, distribution of information and subtasks, negotiation between team and individual goals, communication protocols, sensing, and collision avoidance. These challenges are exacerbated by the fact that often the task is to be completed with limited computational, communication, and sensing resources. A key design decision is between a centralized coordination scheme and a decentralized/distributed one. In a centralized scheme, each agent has access to measurement and/or control information from a master entity, such as a central processing unit or a global positioning system (GPS). Therefore, centralized schemes have a single point of failure like a single "large agent". They also do not scale well with the number of agents because the processing overhead and number of communication links become prohibitive.

The multi-agent literature has numerous references to decentralized and distributed schemes with the two terms used interchangeably. Unfortunately, there does not seem to be a precise definition for either concept, with different authors using different definitions. In order to avoid misunderstandings, it is important that we define what we mean by decentralized/distributed in this book.

Definition 1.1 A decentralized (or distributed) multi-agent coordination scheme is one where any information—cognitive, sensory, computational, control, etc.—is acquired ***locally*** by each agent via ***onboard*** hardware and software. This includes sensors, communication links, memory, and processors. That is, only the agents themselves are responsible for acquiring information and sharing it as necessary without external aid. Information can be input to an agent by onboard sensors or wireless communication with other agents, or pre-programmed into the agent.

Sometimes it is debatable if a coordination scheme can classified as centralized or decentralized. For example, in a leader–follower scheme, the leader agent can be viewed as the master entity despite being a local component of the multi-agent system. However, the coordination scheme can be designed such that if the leader malfunctions, another agent takes up this role with minimal disruption to the assignment. According to Definition 1.1, a leader–follower scheme where the leader is an integral part of the multi-agent system is deemed decentralized in this book.

A number of coordination and cooperation problems for multi-agent systems are described in the literature: aggregation, consensus, agreement, rendezvous, synchronization, social foraging, flocking, coverage, scheduling, and formation [2–4]. Most of these problems are similar in that the purpose is to drive the multiple agents to some common state (position, velocity, frequency, arrival time, temperature, voltage, etc.), which in general may not be related to their motion. This book is devoted strictly to the class of *formation* problems. Specifically, our focus is on three related problems with increasing levels of complexity: *formation acquisition* (where agents are required to form and maintain a pre-defined geometric configuration in space), *formation maneuvering* (where agents are required to simultaneously acquire a formation and move as a unit following a pre-defined trajectory), and *target interception* (where agents intercept and surround a moving target with a pre-defined formation). Note that formation acquisition is a pre-condition for formation maneuvering and target interception. An example of an application where formation control is necessary is the use of a fleet of unmanned air vehicles (UAVs) to create an aerial image of a large area with high spatial resolution. The UAVs need to be properly positioned relative to each other such that each UAV image can be stitched together, with no gaps, to form the overall area map. Another example is maintaining the optimal placement of a network of mobile sensors during static and dynamic target tracking applications such that the collective sensing performance is improved [5]. The classification of formation problems used in this book may have elements of and partially overlap with some of the coordination/cooperation problems listed above. For instance, formation maneuvering is related to flocking, where agents have to move cohesively without colliding with each other.

Often in formation control the main concern is convergence to the desired spatial configuration irrespective of its exact global position in space. That is, the formation is to be acquired up to rotation and translation of the whole set of agent positions. This means that only the *relative positions* of agents need to be known by the control algorithm. Formation control problems are relatively straightforward to solve when the agents' absolute coordinates (i.e., with respect to an Earth-fixed coordinate frame) are available via a central planner. From this information, the relative positions can be readily calculated. However, a GPS, which is typically used in such cases, can lose accuracy when the line of sight between the GPS receiver and satellite is obstructed (e.g., urban areas, dense vegetation, underwater). Therefore, one would like to use a decentralized formation scheme where information is obtained from a suite of onboard sensors (see Figure 1.3). A number of commercially available sensors can be used for this purpose: inertial-type navigation system, laser range finder, sonar, radar, infrared sensor, camera, and compass.

Accurate control of the agents' relative positions is critical for solving formation problems. This, in turn, is strongly dependent on the model of the agents' motion used to design the formation controller. A common approach is to design a "high-level" control law by assuming that the agent motion is governed by the single-integrator model. In this model, each agent is treated as a *kinematic* point where the state is position and the control input is velocity. Another common but lower level approach is to use the double-integrator model. Here, each agent is treated as a holonomic point mass where the states are position and velocity, and the control input is acceleration. That is, the double-integrator model is a *dynamic* model, albeit simplified, of the agent motion. Neither of these simplified approaches can be directly implemented on an actual multi-agent system since they do not provide actuator-level (i.e., force/torque) inputs. At best, they can be embedded as outer control loops in a nonmodel-based, actuator-level control system that neglects the agent dynamics. For applications where high performance is expected, this approach

Figure 1.3 Centralized (left) and decentralized (right) formation control.

may not yield the necessary control accuracy due to improper compensation of the dynamic effects influencing the agent motion (e.g., inertial, frictional, gravitational, nonholonomic, actuator, etc.). In such cases, the explicit compensation of these effects can only be accomplished by considering the full agent dynamics in the control design.

Graph theory is a natural tool for describing the multi-agent formation shape as well as the inter-agent sensing, communication, and control topology in the decentralized case. This book is based on an important subset of this theory—*rigid graph theory*—since it naturally ensures that the inter-agent distance constraints of the desired formation are enforced through the graph rigidity. This implicitly ensures that collisions between agents are avoided while acquiring the formation. We use the concept of graph rigidity as an abstraction of the rigidity of physical structures. In our case, the "vertices" of the "structure" are the agents and the "bars" connecting the vertices are the inter-agent distance constraints imposed by the desired formation. Within this framework, it is convenient to treat each agent as a point and model their motion with the single-integrator equation. As a result, most graph rigidity-based formation controllers utilize the single-integrator model. In this book we will go beyond this approach and introduce results that are based on the double-integrator model and, subsequently, the full dynamic model.

Independent of the model used to describe the agent motion, the formation control laws will primarily stabilize the inter-agent distance dynamics to desired distances. Despite controlling the inter-agent distances, the control laws will depend on the distance *and* angle between agents (i.e., the relative position vector). This is a case of over-sensing [6, 7], but it is a necessary feature of distance-based formation control laws to the best of our knowledge. We will, however, minimize the number of relative positions that need to be measured by making use of a special property of rigid graphs called *minimal rigidity*.

In addition to concepts from rigid graph theory, our control designs will make use of nonlinear stability theory [8] and the integrator backstepping control technique [9]. The former is needed since the inter-agent distance dynamics are nonlinear. As a result, the stability analyses will require an interplay of rigid graph theory and nonlinear analysis methods. The latter will allow us to seamlessly incorporate the single-integrator control designs into the double-integrator and holonomic dynamics designs.

1.2 Notation

In this section, we compile some mathematical notations used throughout the book. The material is provided primarily for reference purposes and is likely familiar to most readers. Some relevant supporting definitions and results from

matrix theory, linear algebra, signals, and system theory are provided in Appendices A to C.

- $x \in \mathbb{R}^n$ or $x = [x_1, \dots, x_n]$ denotes an $n \times 1$ (column) vector.
- $x = [x_1, \dots, x_n]$ where $x_i \in \mathbb{R}^m$ denotes the stacked $mn \times 1$ vector.
- $\|x\|$ denotes the Euclidean norm or 2-norm of the vector x (we omit the subscript 2 to simplify the notation); $\|x\|_1$ denotes the vector 1-norm.
- For points $\zeta, x \in \mathbb{R}^n$ and set $\mathcal{M}$,

$$\text{dist}(\zeta, \mathcal{M}) := \inf_{x \in \mathcal{M}} \|\zeta - x\|,$$

  i.e., the smallest distance from point ζ to any point in $\mathcal{M}$.
- The convex hull of points x_i, $i = 1, \dots, n$ in Euclidean space is denoted by $\text{conv}\{x_1, \dots, x_n\}$ and represents the smallest convex set that contains all points.
- $\mathbf{1}_n$ is the $n \times 1$ vector of ones.
- I_n is the $n \times n$ identity matrix.
- 0 can mean the scalar zero or a vector of all zeros depending on the situation.
- $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$ denote the minimum and maximum eigenvalues of a matrix, respectively.
- $\text{diag}(a_1, \dots, a_n)$ where $a_i \in \mathbb{R}$ denotes the $n \times n$ diagonal matrix with diagonal elements a_i.
- $\text{diag}(A_1, \dots, A_n)$ where $A_i \in \mathbb{R}^{m \times m}$ denotes the $mn \times mn$ block diagonal matrix.

1.3 Graph Theory

The control algorithms in this book will rely on some basic concepts of rigid graph theory in two and three dimensions. Below we provide an introduction to these concepts. Henceforth, the superscript m in $\mathbb{R}^m$, which denotes the Euclidean space of the graph, will be either 2 or 3.

1.3.1 Graph

An undirected graph G is a pair (V, E) where $V = \{1, 2, \dots, n\}$ is the set of vertices and $E \subset V \times V$ is the set of undirected edges such that if vertex pair $(i, j) \in E$ then so is (j, i) (i.e., the vertex pair is not ordered and is counted only once). Since we will only be dealing with undirected graphs in this book, we will drop the term "undirected" from here on when referring to a graph, with the understanding that the concepts in this chapter are introduced in the context of undirected graphs.

For any graph, the number of edges l belongs to the set

$$l \in \{1, \dots, n(n-1)/2\}. \tag{1.1}$$

We will use the notation $G = (V, E)$ to denote that graph G has vertex set V and edge set E. We only consider graphs where $n > m$ to discount uninteresting, special cases. The set of neighbors of vertex i will be represented by

$$\mathcal{N}_i(E) = \{j \in V \mid (i,j) \in E\}, \tag{1.2}$$

i.e., the set of all vertices that are connected to vertex i with an edge.

A graph G is said to be *complete* if every pair of distinct vertices is connected by an edge, i.e., $l = n(n-1)/2$. A complete graph with n vertices is symbolized by K_n. Figure 1.4 shows an example of the complete graph K_5.

A *path* is a trail that goes from an origin vertex to a destination vertex by traversing edges of the graph. Two vertices i and j are said to be *connected* if there exists a path between these vertices. A graph is connected if there is a path between every pair of vertices of G.

The *adjacency matrix* $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{n\times n}$ of a graph $G = (V, E)$ is defined such that

$$a_{ij} = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}, \quad a_{ij} = a_{ji},\ i \neq j, \quad \text{and} \quad a_{ii} = 0. \tag{1.3}$$

The *Laplacian matrix* $\mathcal{L} = [\ell_{ij}] \in \mathbb{R}^{n\times n}$ is defined as

$$\ell_{ii} = \sum_{j=1}^{n} a_{ij} \quad \text{and} \quad \ell_{ij} = -a_{ij}, i \neq j. \tag{1.4}$$

Note that the Laplacian matrix is symmetric and satisfies

$$\sum_{j=1}^{n} \ell_{ij} = 0, \quad i = 1, \ldots, n. \tag{1.5}$$

The Laplacian matrix has some other interesting and useful properties. If $\lambda_1 \leq \cdots \leq \lambda_n$ denote the n eigenvalues of $\mathcal{L}$, then $\lambda_1 = 0$ and $\lambda_2 \geq 0$. That is,

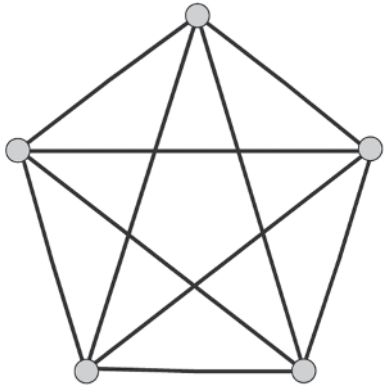

Figure 1.4 The complete graph K_5.

$\mathcal{L}$ is a positive semi-definite matrix. If G is connected, then $\lambda_2 > 0$ (i.e., $\mathcal{L}$ has a single zero eigenvalue). In this case, the eigenvector associated with λ_1 is $\mathbf{1}_n$ such that

$$\mathcal{L}\mathbf{1}_n = 0. \tag{1.6}$$

This implies that

$$\mathcal{L}x = 0$$

if and only if $x \in \mathbb{R}^n$ with $x_i = x_j, \forall i, j$.

Lemma 1.1 ***[10]*** Let $B = \text{diag}(b_1, \dots, b_n)$ be such that $b_i = 1$ or 0, $i = 1, \dots, n$ with at least one nonzero entry, then the matrix

$$M = \mathcal{L} + B$$

is symmetric and positive definite.

1.3.2 Framework

A framework is simply a realization of a graph at given points in Euclidean space. Specifically, if $p_i \in \mathbb{R}^m$ is the coordinate of vertex i with respect to some fixed coordinate frame, then a framework F is a pair (G, p) where $p = [p_1, \dots, p_n] \in \mathbb{R}^{nm}$. We will use the notation $F = (G, p)$ to denote that framework F is composed of graph G with coordinates p. The importance of a framework is that it can model a physical structure. That is, consider the so-called "bar-and-joint" framework, which is a structure made of rigid bars joined at their ends by universal joints.[1] A bar-and-joint framework can be used to describe a wide range of static and dynamic structures, such as bridges, mechanical linkages, and biological molecules (see Figure 1.5).

We are often interested in knowing the length of the edges in a framework. Based on an arbitrary ordering of the edges in E, the edge function $\phi : \mathbb{R}^{nm} \to \mathbb{R}^l$ for a framework $F = (G, p)$ where $G = (V, E)$ is given by

$$\phi(p) = [\dots, \|p_i - p_j\|^2, \dots], \quad (i, j) \in E. \tag{1.7}$$

The kth component of (1.7), $\|p_i - p_j\|^2$, corresponds to the kth edge of E connecting vertices i and j.

Example 1.1 Figure 1.6a shows a graph $G = (V, E)$ in $\mathbb{R}^2$ with $n = 5$ and $l = 6$. By numbering the vertices and edges arbitrarily, we have

$$\begin{aligned} V &= \{1, 2, 3, 4, 5\} \quad \text{and} \\ E &= \{(1, 2), (2, 3), (3, 4), (1, 4), (2, 4), (4, 5)\}. \end{aligned} \tag{1.8}$$

1 A universal joint is one where, if one of the adjacent bars is fixed, the other bar can rotate in every direction.

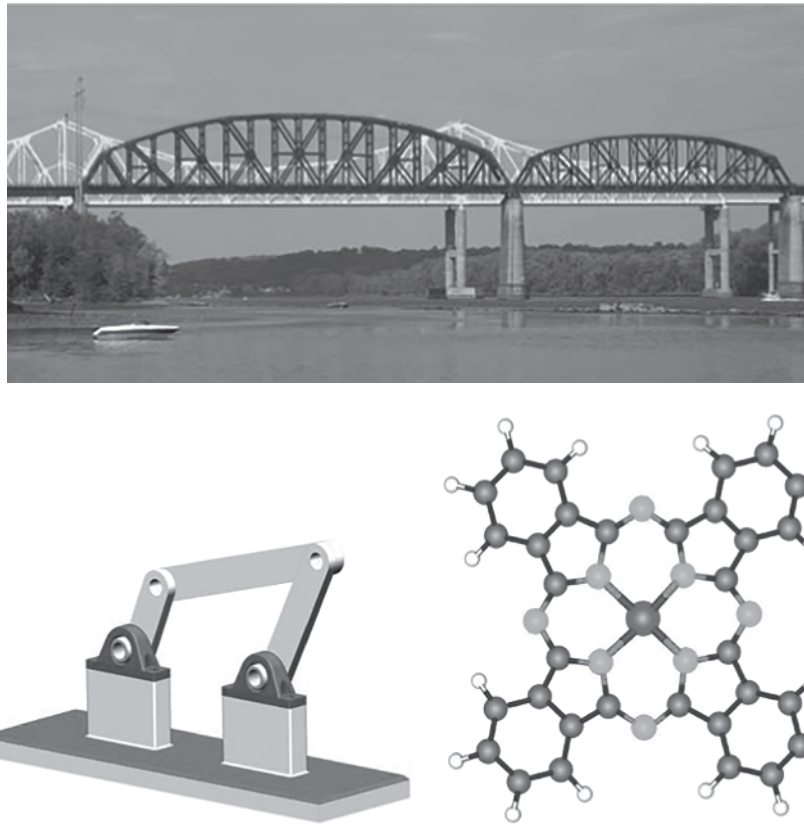

Figure 1.5 Examples of bar-and-joint frameworks, although the joints here are not necessarily universal.

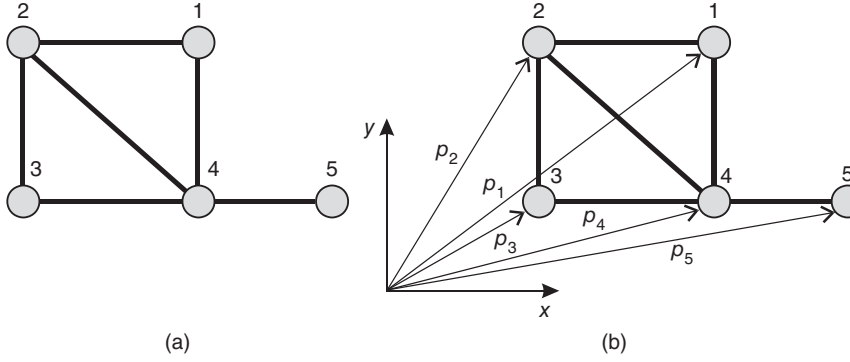

Figure 1.6 Example of (a) a graph and (b) a corresponding framework.

Note that the maximum possible number of edges when $n = 5$ is $l = 10$, which is obtained by adding the following edges to E: $(1,3)$, $(1,5)$, $(2,5)$, and $(3,5)$. The set of neighbors of, for example, vertices 2 and 5 are $\mathcal{N}_2(E) = \{1,3,4\}$ and $\mathcal{N}_5(E) = \{4\}$, respectively. Figure 1.6b shows a framework F associated with G by assigning coordinates to each vertex, i.e., $F = (G,p)$ where $p = [p_1, p_2, p_3, p_4, p_5]$, $p_i = [x_i, y_i]$, and G was defined by (1.8). According to (1.7), the edge function for F is given by

$$\begin{aligned}\phi(p) = [&\|p_1 - p_2\|^2, \|p_2 - p_3\|^2, \|p_3 - p_4\|^2, \|p_1 - p_4\|^2,\\ &\|p_2 - p_4\|^2, \|p_4 - p_5\|^2].\end{aligned} \tag{1.9}$$

1.3.3 Rigid Graphs

Given a bar-and-joint framework, a fundamental question is whether it is *rigid* or not. By rigidity, we mean the non-deformation of the structure. The concept of *rigid* graphs is central to this book, and is an abstraction of the rigidity of civil and mechanical structures. The first reference to graph rigidity dates back to a mathematical problem studied by Leonhard Euler in 1766 [12]. Our goal here is to have a simple means of predicting rigidity. To formalize the concept of rigidity, we need to first introduce the following definitions.

Definition 1.2 ***[11]*** A rigid body is said to be in translation if all points forming the body move along parallel paths (straight or curvilinear). A rigid body is said to be in rotation if all points move in parallel planes along circles centered on a same axis that intersects the body.

If the axis of rotation does not intersect the rigid body, the motion is usually called a revolution or orbit. This type of motion (in fact, any type of rigid body motion) can be decomposed into a translation superimposed on a rotation. That is, the above definitions allow one to *decouple* pure translation from pure rotation. Recall from rigid body kinematics that given body-fixed points p and O (see Figure 1.7), the following relationship holds

$$\dot{\rho} = \dot{R} + \omega \times r \tag{1.10}$$

where $\omega \in \mathbb{R}^3$ denotes the angular velocity of the rigid body about an arbitrary axis $\hat{R}$ passing through point O and $\{x, y, z\}$ is an inertial coordinate frame.

Definition 1.3 ***[13]*** Two frameworks (G,p) and $(G,\hat{p})$ with $G = (V,E)$ are:

- Equivalent if $\|p_i - p_j\| = \|\hat{p}_i - \hat{p}_j\|$ for all $(i,j) \in E$, i.e., the edge function (1.7) is the same;

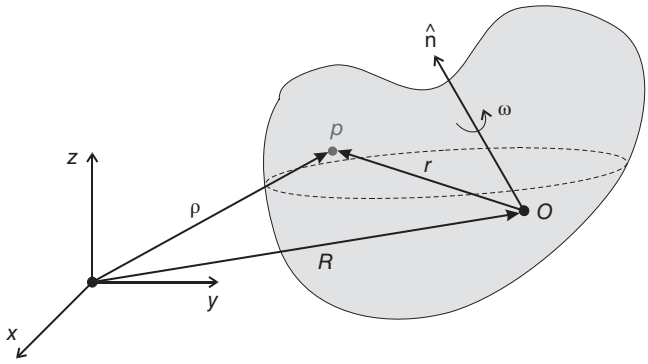

Figure 1.7 Rigid body kinematics.

- Congruent if $\|p_i - p_j\| = \|\hat{p}_i - \hat{p}_j\|$ for all $i, j \in V$ (with $i \neq j$),[2] i.e., all distances between vertices are the same.

Note that congruency implies equivalency, but the reverse is not necessarily true.

Definition 1.4 ***[14]*** An isometry of $\mathbb{R}^m$ is a bijective map $T : \mathbb{R}^m \to \mathbb{R}^m$ such that

$$\|T(x) - T(y)\| = \|x - y\|, \quad \forall x, y \in \mathbb{R}^m. \tag{1.11}$$

Note that T accounts for rotation and/or translation of the vector $x - y$.

Two frameworks (G, p) and $(G, \hat{p})$ are said to be *isomorphic* in $\mathbb{R}^m$ if they are related by an isometry in $\mathbb{R}^m$. It is not difficult to see that isomorphic frameworks are congruent. We denote the set of all frameworks that are isomorphic to F by $\text{Iso}(F)$.

Example 1.2 Consider the framework F in $\mathbb{R}^2$ shown in Figure 1.8, which is an equilateral triangle with sides of length a. Frameworks F_1 and F_2 are isomorphic to F since they can be obtained by translating and/or rotating F.

Definition 1.5 ***[14]*** A motion of a framework $F = (G, p)$ with $G = (V, E)$ is a continuous family of equivalent frameworks $F(t)$ for $t \in [0, 1]$ where $F(0) = F$. That is, each point p_i, $i \in V$ moves along a continuous trajectory $p_i(t)$ while preserving the distances between points connected by an edge.

2 Henceforth, whenever the notation $i.j \in V$ is used, it should be understood that $i \neq j$. Note that the number of such (i, j) pairs is always $n(n-1)/2$ according to (1.1).

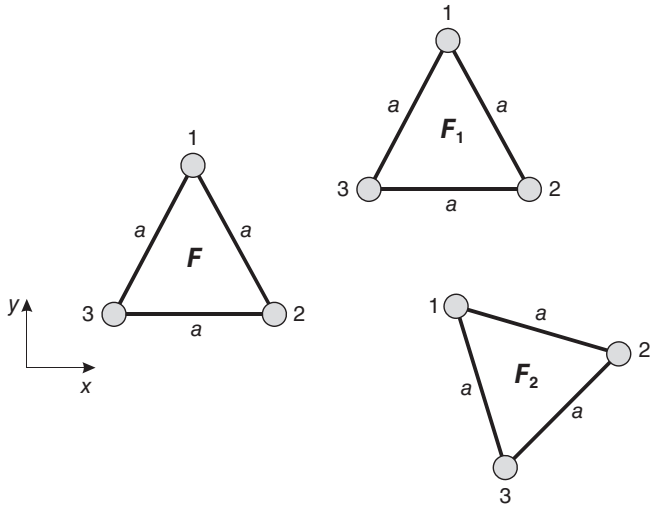

Figure 1.8 Isomorphic frameworks.

This leads us to the following definition of rigidity.

Definition 1.6 ***[14, 15]*** A framework $F = (G, p)$ is rigid in $\mathbb{R}^m$ if all of its motions satisfy $p_i(t) = T(p_i)$, $\forall i \in V$, and $\forall t \in [0, 1]$, i.e., the family of frameworks $F(t)$ is isomorphic. On the other hand, the framework is flexible in $\mathbb{R}^m$ if and only if it is possible to continuously move its vertices to form an equivalent but non-congruent framework.

Example 1.3 The bar-and-joint framework in Figure 1.9a is flexible in $\mathbb{R}^2$ because the motion of vertices 1 and 2 leads to a framework that, although equivalent, is non-congruent to the original one. In particular, the distances between vertices 1 and 3 and vertices 2 and 4 have been altered. This deformation can be prevented by adding an edge (bar) between vertices 1 and 3, for example. Therefore, the framework in Figure 1.9b is now rigid and its motions are restricted to rigid body ones. Notice that the bridge in Figure 1.5 is designed to be a rigid framework (at least one would hope), while the four-bar

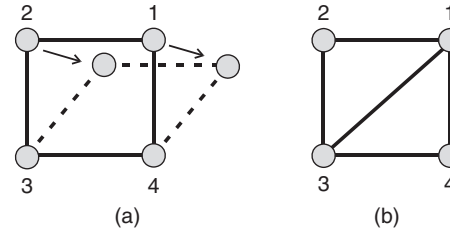

Figure 1.9 Examples of (a) flexible and (b) rigid bar-and-joint frameworks.

mechanism is not. Finally, the framework in Figure 1.6 is flexible since edge (4,5) can be rotated about vertex 4, changing the distance between vertices 1 and 5, for example.

1.3.4 Infinitesimal Rigidity

Although our physical intuition can indicate if certain frameworks are rigid or not, in general it is difficult to determine rigidity from Definition 1.6. Therefore, we will consider a related notion of rigidity called *infinitesimal rigidity*, which can be easily verified via a matrix rank. In general, rigidity does not imply infinitesimal rigidity, but infinitesimal rigidity implies rigidity. Frameworks that are rigid but not infinitesimally rigid usually have collinear or parallel edges. For example, the framework in Figure 1.10 is rigid but not infinitesimally rigid since it has three parallel edges. The reader is referred to [14, 16] for other examples.

For so-called *generic* frameworks, rigidity is equivalent to infinitesimal rigidity [15].[3] When a framework $F = (G, p)$ is generic, then *almost all* of its realizations have the same rigidity property under small perturbations on p [16]. As a result, generic rigidity is a property only of the underlying graph G, i.e., it is independent of almost all p. The term "almost all" is used to exclude certain degenerate configurations, such as frameworks that lie in a hyperplane.[4] A detailed study of generic frameworks can be found in [17].

Example 1.4 The planar framework in Figure 1.11 is nongeneric in $\mathbb{R}^2$ since vertices 1, 2, 3, and 4 are collinear. The planar frameworks in Figure 1.9 are generic in $\mathbb{R}^2$, but nongeneric in $\mathbb{R}^3$ since all vertices are in a plane.

In infinitesimal rigidity, instead of preserving distances during a continuous deformation, we require *first-order* preservation of distances during an *infinitesimal* motion. Therefore, infinitesimal rigidity (also called first-order

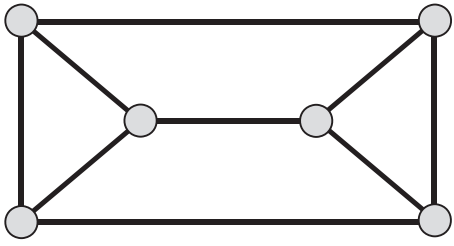

Figure 1.10 Example of a rigid framework that is not infinitesimally rigid.

3 It has been noted in the literature (see [15, 18]) how these nuanced notions of rigidity often cause confusion, with definitions varying from author to author.

4 A hyperplane is a subspace of one dimension less than its Euclidean space. For example, a line is a hyperplane in $\mathbb{R}^2$ and a plane is a hyperplane in $\mathbb{R}^3$.

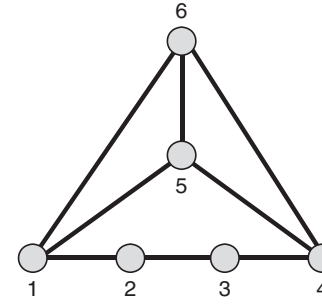

Figure 1.11 A nongeneric framework in $\mathbb{R}^2$.

rigidity) is a linear approximation to rigidity. To be more precise, we first use Definition 1.5 to write

$$\|p_i(t) - p_j(t)\| = \|p_i - p_j\|, \quad (i,j) \in E. \tag{1.12}$$

Note that the right-hand side of (1.12) is constant by definition. Assuming the trajectories $p_i(t)$, $i \in V$ are differentiable on $t \in [0,1]$, we square both sides of (1.12) and differentiate with respect to time to obtain

$$\frac{d}{dt}\|p_i(t) - p_j(t)\|^2 = 2(p_i(t) - p_j(t))^\top(\dot{p}_i(t) - \dot{p}_j(t)) = 0, \quad (i,j) \in E, \tag{1.13}$$

where $\dot{p}_i$ denotes the time derivative of p_i. Rather than require that (1.13) be satisfied for all $t \in [0,1]$, we impose the condition that it only need to hold at $t = 0$:

$$(p_i - p_j)^\top(v_i - v_j) = 0, \quad (i,j) \in E, \tag{1.14}$$

where $v_i := \dot{p}_i(0)$ and, from Definition 1.5, $p_i(0) = p_i$. The above equation leads to a system of l linear equations with the nm unknowns being the velocities v_i. When instantaneous velocities v_i that satisfy (1.14) exist, we say the framework has infinitesimal motion.

When analyzing infinitesimal rigidity, the *rigidity matrix* of a framework comes in handy. The rigidity matrix $R : \mathbb{R}^{nm} \to \mathbb{R}^{l \times nm}$ is defined as

$$R(p) = \frac{1}{2}\frac{\partial \phi(p)}{\partial p} \tag{1.15}$$

where $\phi(p)$ was given in (1.7). Note that the rigidity matrix has a row for each edge and m columns for each vertex. That is, for the kth edge of E connecting vertices i and j, the kth row of R is

$$[0 \ \ldots \ 0 \ (p_i - p_j)^\top \ 0 \ \ldots \ 0 \ (p_j - p_i)^\top \ 0 \ \ \ldots \ \ 0] \tag{1.16}$$

where $(p_i - p_j)^\top$ is in the columns for vertex i, $(p_j - p_i)^\top$ is in the columns for vertex j, and all other elements are zero.

Example 1.5 The rigidity matrix of the framework with edge function (1.9) is given by

$$R(p) = \begin{bmatrix} (p_1 - p_2)^\top & (p_2 - p_1)^\top & 0 & 0 & 0 \\ 0 & (p_2 - p_3)^\top & (p_3 - p_2)^\top & 0 & 0 \\ 0 & 0 & (p_3 - p_4)^\top & (p_4 - p_3)^\top & 0 \\ (p_1 - p_4)^\top & 0 & 0 & (p_4 - p_1)^\top & 0 \\ 0 & (p_2 - p_4)^\top & 0 & (p_4 - p_2)^\top & 0 \\ 0 & 0 & 0 & (p_4 - p_5)^\top & (p_5 - p_4)^\top \end{bmatrix}. \tag{1.17}$$

Using (1.15), we can conveniently rewrite (1.14) as

$$R(p)v = 0 \tag{1.18}$$

where $v = [v_1, \ldots, v_n] \in \mathbb{R}^{nm}$. Therefore, infinitesimal motion exists when (1.18) has a nontrivial (nonzero) solution v. We would like to know when this is the case.

Note that if a framework undergoes any rigid body motion, then according to (1.10) vertex i has velocity

$$v_i = v^* + \omega \times p_i \tag{1.19}$$

where $v^* \in \mathbb{R}^3$ is the translational velocity and $\omega \in \mathbb{R}^3$ is the angular velocity.[5] It is not difficult to check that (1.18) holds in this case since, from (1.16), we have that

$$\begin{aligned} (p_i - p_j)^\top v_i + (p_j - p_i)^\top v_j &= (p_i - p_j)^\top (v^* + \omega \times p_i) \\ &\quad + (p_j - p_i)^\top (v^* + \omega \times p_j) \\ &= (p_i - p_j) \cdot (\omega \times p_i) + (p_j - p_i) \cdot (\omega \times p_j) \\ &= p_i \cdot \omega \times p_i - p_j \cdot \omega \times p_i + p_j \cdot \omega \times p_j \\ &\quad - p_i \cdot \omega \times p_j \\ &= -p_j \times \omega \cdot p_i - p_i \cdot \omega \times p_j = 0 \end{aligned} \tag{1.20}$$

upon use of the properties of vector products. In other words, rigid body motions produce infinitesimal motions. This leads us to the following definition.

Definition 1.7 ***[16]*** A framework is infinitesimally rigid if the only solutions to (1.18) arise from rigid body motions. Otherwise, it is infinitesimally flexible.

---

5 Notice that the vectors are defined here with dimension 3 irrespective of the Euclidean space of the framework. If the framework is planar, then $v^* = [v_x^*, v_y^*, 0]$, $\omega = [0, 0, \omega_z]$, and $p_i = [p_{ix}, p_{iy}, 0]$.

A useful structural property of the rigidity matrix, which follows from (1.16) and will be exploited in some of the control designs of this book, is the following.

Property 1.1 Given any vector $x \in \mathbb{R}^m$,

$$R(p)(\mathbf{1}_n \otimes x) = 0.$$

We can use Definition 1.7 to determine if a framework is infinitesimally rigid or flexible by attempting to assign a velocity vector to each vertex such that (1.18) holds. For example, consider the framework in Figure 1.11 and assign velocity $v_2 \in \mathbb{R}^2$ to vertex 2 such that it is any vector *perpendicular* to $p_1 - p_2$ and $p_2 - p_3$ while keeping all other velocities at zero, i.e., $v = [0, v_2, 0, 0, 0, 0]$. It is easy to see that $R(p)v = 0$ in this case although v is not due to a rigid body motion. Therefore, the framework is infinitesimally flexible.[6] This trial-and-error method becomes cumbersome as the complexity of the framework increases or when the framework is generic. More important, it cannot be easily implemented computationally.

Fortunately, there is an easy way of checking whether a framework is infinitesimally rigid via the rank of the rigidity matrix. Before presenting this useful result, we need the following facts. First, the rank of an $m \times n$ matrix is equal to n minus the dimension of its null space. Second, the number of independent rigid body motions for a generic framework in $\mathbb{R}^m$ is $\frac{m(m+1)}{2}$ [18]. This means that there exist *three* independent rigid body motions in $\mathbb{R}^2$ (translation along x, translation along y, and rotation about z) and *six* in $\mathbb{R}^3$ (translations along x, y, z and rotations about x, y, z).

Given the above, we deduce that $\dim(\text{Null}(R(p))) \geq \frac{m(m+1)}{2}$ and therefore $\text{rank}(R(p)) \leq nm - \frac{m(m+1)}{2}$ [15]. From Definition 1.7, we know that for infinitesimally rigid frameworks $\dim(\text{Null}(R(p))) = \frac{m(m+1)}{2}$, which gives us the following result.

Result 1.1 ***[14]*** A (generic) framework in $\mathbb{R}^m$ is infinitesimally rigid if and only if

$$\text{rank}(R(p)) = nm - \frac{m(m+1)}{2}. \tag{1.21}$$

Therefore, $\text{rank}(R(p)) = 2n - 3$ in $\mathbb{R}^2$ and $\text{rank}(R(p)) = 3n - 6$ in $\mathbb{R}^3$.

Example 1.6 Consider the framework in Figure 1.6b whose rigidity matrix is given in (1.17), and let $p_1 = [1, 1]$, $p_2 = [0, 1]$, $p_3 = [0, 0]$, $p_4 = [1, 0]$, and

6 This framework is, however, rigid since it is nongeneric.

$p_5 = [2, 0]$. Then,

$$R(p) = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \end{bmatrix} \tag{1.22}$$

and $\text{rank}(R(p)) = 6$, which is less than $2n - 3 = 7$. Thus, the framework is infinitesimally flexible. If we now remove vertex 5 and edge 6 (connecting vertices 4 and 5) from Figure 1.6b, we obtain a framework similar to the one in Figure 1.9b. Its rigidity matrix will be the 5×8 submatrix of (1.22) that results from deleting the last row and the last two columns. In this case, $\text{rank}(R(p)) = 5$ $(= 2n - 3 = 2 \times 4 - 3)$ and the framework is infinitesimally rigid in $\mathbb{R}^2$.

Example 1.7 Consider the three-dimensional framework in Figure 1.12 where $E = \{(1, 2), (2, 3), (1, 3), (1, 4), (2, 4), (3, 4)\}$ and $p_1 = [0, 0, 0]$, $p_2 = [0, 1, 0]$, $p_3 = [1, 0, 0]$, and $p_4 = [0.5, 0.5, 1]$. The rigidity matrix is given by

$$R(p) = \begin{bmatrix} 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -0.5 & -0.5 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 1 \\ 0 & 0 & 0 & -0.5 & 0.5 & -1 & 0 & 0 & 0 & 0.5 & -0.5 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & -0.5 & -1 & -0.5 & 0.5 & 1 \end{bmatrix}$$

whose rank is 6 $(= 3n - 6)$ so the framework is infinitesimally rigid in $\mathbb{R}^3$.

As discussed earlier, the rigidity property of a generic framework $F = (G, p)$ is invariant under small perturbations on p. With the intent of formalizing this idea, we introduce the following result.

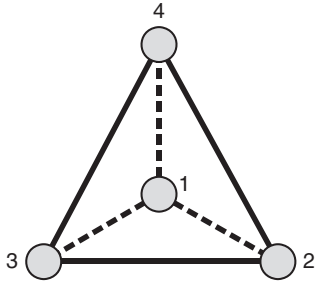

Figure 1.12 A tetrahedron framework.

Result 1.2 Consider two frameworks $F = (G, p)$ and $\overline{F} = (G, \overline{p})$ sharing the same graph. If F is infinitesimally rigid and $\text{dist}(\overline{p}, \text{Iso}(F)) \leq \varepsilon$ where ε is a sufficiently small positive constant, then $\overline{F}$ is also infinitesimally rigid.

Proof: Let $\hat{F} = (G, \hat{p}) \in \text{Iso}(F)$ be such that

$$\text{dist}(\overline{p}, \text{Iso}(F)) = \inf_{x \in \text{Iso}(F)} \|\overline{p} - x\| = \|\overline{p} - \hat{p}\|. \tag{1.23}$$

Obviously, $\hat{F}$ is infinitesimally rigid since it is isomorphic to F. Therefore, $\text{rank}(R(\hat{p})) = 2n - 3$ according to Result 1.1, and there exists a $(2n-3) \times (2n-3)$ submatrix of $R(\hat{p})$, $R_s(\hat{p})$, such that $\det[R_s(\hat{p})] \neq 0$. The submatrix $R_s(\hat{p})$ has nonzero elements of the form $(\hat{p}_i - \hat{p}_j)^T$, $(i, j) \in E$. Since $\text{dist}(\overline{p}, \text{Iso}(F)) = \|\overline{p} - \hat{p}\| \leq \varepsilon$, it is not difficult to show that $[\overline{p}_i]_k = [\hat{p}_i]_k + \gamma_{ik}$ where γ_{ik} is a sufficiently small positive constant. Thus, the nonzero elements of $R_s(\overline{p})$ have the form $[\overline{p}_i]_k - [\overline{p}_j]_k = [\hat{p}_i]_k - [\hat{p}_j]_k + \gamma_{ik} - \gamma_{jk}$, which are continuously dependent on $\hat{p}$. Since the eigenvalues of a matrix depend continuously on its elements [19], and the determinant of a matrix is the product of its eigenvalues, it follows that the determinant continuously depends on the elements of the matrix. Thus, for sufficiently small γ_{ik}, we have that $\det[R_s(\overline{p})] \neq 0$ and $\text{rank}(R_s(\overline{p})) = \text{rank}(R_s(\hat{p})) = 2n - 3$. Now, since $R_s(\overline{p})$ is a full rank submatrix of $R(\overline{p})$, we know $\text{rank}(R(\overline{p})) = 2n - 3$, so $\overline{F}$ is infinitesimally rigid.

1.3.5 Minimal Rigidity

It is obvious that adding edges to a graph does not destroy rigidity. A natural question is then: What is the *minimum* number of edges that ensures rigidity ? This is important in practice because it ensures that a formation of multiple agents is rigid with the minimum number of sensing and communication links.[7]

Definition 1.8 ***[6]*** A graph is minimally rigid if it is rigid and the removal of a single edge causes it to lose rigidity.

Example 1.8 The graph in Figure 1.13a is minimally rigid because the removal of any single edge will make it flexible. On the other hand, any single edge can be removed from the graph in Figure 1.13b and it remains rigid.

Just like with infinitesimal rigidity, we want a checkable condition for minimal rigidity. Fortunately, the following condition exists.

---

7 The disadvantage of minimal rigidity is that it lacks robustness to the loss of a sensor or communication link since redundant edges are removed from the graph [6].

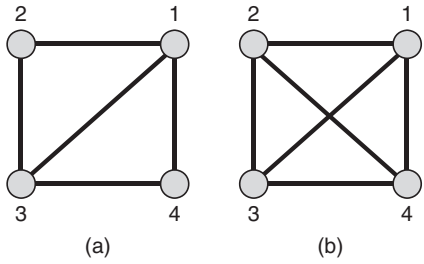

Figure 1.13 (a) Minimally and (b) nonminimally rigid graphs.

Result 1.3 *[6]* A rigid graph is minimally rigid if and only if $l = mn - \frac{m(m+1)}{2}$.

It is interesting to note the recurrence of the term $mn - \frac{m(m+1)}{2}$ in Results 1.1 and 1.3. This leads to the following corollary, which will be very useful for the control designs in this book.

Corollary 1.1 If a framework is infinitesimally and minimally rigid, then its rigidity matrix has *full row rank* and $R(p)R^{\top}(p)$ is positive definite.

Proof: Given that the rigidity matrix has l rows and $\text{rank}(R(p)) = nm - \frac{m(m+1)}{2}$ and $l = mn - \frac{m(m+1)}{2}$ for infinitesimally and minimally rigid frameworks, we know $R(p)$ is full row rank. Now, let $y = R^{\top}(p)x$ and

$$V = y^{\top}y \geq 0.$$

Note that V is positive definite with respect to y and positive semi-definite with respect to x. Given that $\text{rank}(R(p)) = \text{rank}(R(p)R^{\top}(p))$ and $R(p)$ is full row rank, we know $R(p)R^{\top}(p)$ is invertible and has no zero eigenvalues. Therefore, V is positive definite with respect to x and $R(p)R^{\top}(p)$ is a positive definite matrix.

1.3.6 Framework Ambiguities

Consider that a graph $G = (V, E)$ and the length of each edge (i.e., $\|p_i - p_j\|$, $(i, j) \in E$) are given. We want to know all frameworks $F = (G, p)$ that are consistent with this data *excluding* isometries. Can a framework have multiple (nonisomorphic) realizations? There are several sources of nonunique realizations. The trivial one is flexibility, such as in Figure 1.6. Since edge (4,5) can continuously rotate about vertex 4, an infinite number of equivalent frameworks exist that satisfy the edge constraints. Infinitesimally rigid frameworks can also suffer from nonuniqueness. Two types of nonuniqueness can occur in this case,

but unlike flexibility they lead to a finite number of equivalent frameworks [20, 21]:

- Flip ambiguity: This occurs when a graph in $\mathbb{R}^m$ has a set of vertices lying in a $(m-1)$-dimensional subspace about which a portion of the graph can be reflected. This is illustrated in Figure 1.14a, where edges $(1,2)$ and $(1,4)$ are reflected over the "mirror" edge $(2,4)$. Notice that there cannot be any edges between the portions of the graph separated by the mirror edge. Multiple flips can happen in a given framework. For example, if edges $(2,3)$ and $(3,4)$ also reflect over edge $(2,4)$, one ends up with the completely reversed framework.
- Flex ambiguity: This occurs in minimally rigid graphs since the removal of an edge allows a portion of the graph to flex. If the removed edge is reinserted after a flexing, one may obtain an equivalent framework with a different configuration. An example is shown in Figure 1.14b, where the temporary remove of edge $(2,3)$ allows edges $(2,5)$, $(1,2)$, and $(1,4)$ to flex.

This leads us to the following definition.

Definition 1.9 If two infinitesimally rigid frameworks are equivalent but not congruent, then they are said to be *ambiguous*.

We denote the set of all ambiguities of an infinitesimally rigid framework F by $\mathrm{Amb}(F)$. We assume that all frameworks in $\mathrm{Amb}(F)$ are also infinitesimally rigid.[8] The existence of framework ambiguities is problematic for formation control since the control law cannot distinguish the actual framework from an ambiguous one if only the edge lengths (i.e., inter-agent distances) are being controlled. In such a case, one solution is to initialize the agents sufficiently close to the desired framework to avoid their convergence to an ambiguous

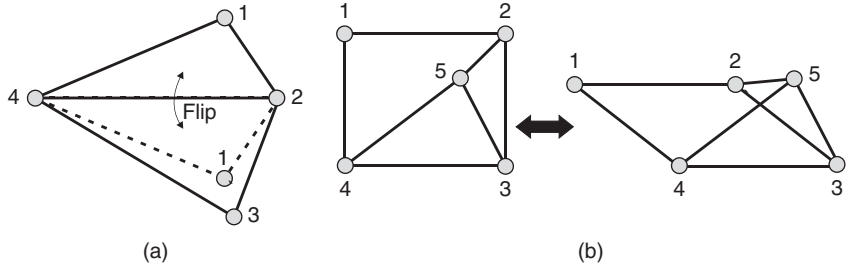

Figure 1.14 Examples of nonunique realizations for infinitesimally rigid frameworks: (a) flip ambiguity and (b) flex ambiguity.

8 This assumption is reasonable and, in fact, holds almost everywhere; see [6] and Theorem 3 of [22] for details.

framework. From a control theory standpoint, this means that stability results will be *local* in nature rather than global. The following corollary to Result 1.2 will be helpful in establishing the stability sets in this book.

Corollary 1.2 Let $F = (G, p)$ and $\overline{F} = (G, \overline{p})$ be two frameworks sharing the same graph, and consider the function

$$\Psi(\overline{F}, F) = \sum_{(i,j)\in E} (\|\overline{p}_i - \overline{p}_j\| - \|p_i - p_j\|)^2. \tag{1.24}$$

If F is infinitesimally rigid and $\Psi(\overline{F}, F) \leq \delta$ where δ is a sufficiently small positive constant, then $\overline{F}$ is also infinitesimally rigid.

Proof: First, note that $\Psi(\overline{F}, F) = 0$ implies that $\overline{F} \in \text{Iso}(F)$ or $\overline{F} \in \text{Amb}(F)$. Therefore, $\Psi(\overline{F}, F) \leq \delta$ implies that there is a sufficiently small positive constant ε such that $\text{dist}(\overline{p}, \text{Iso}(F)) \leq \varepsilon$ or $\text{dist}(\overline{p}, \text{Amb}(F)) \leq \varepsilon$. From Result 1.2, we know that $\overline{F}$ is infinitesimally rigid if $\text{dist}(\overline{p}, \text{Iso}(F)) \leq \varepsilon$. Since the elements of $\text{Amb}(F)$ are infinitesimally rigid, the proof of Result 1.2 can be followed with $\text{Iso}(F)$ replaced by $\text{Amb}(F)$ to show that $\text{dist}(\overline{p}, \text{Amb}(F)) \leq \varepsilon$ implies $\overline{F}$ is infinitesimally rigid.

1.3.7 Global Rigidity

A final variation of graph rigidity is the concept of *global rigidity*.

Definition 1.10 ***[13, 23]*** A framework (G, p) is globally rigid if every framework that is equivalent to (G, p) is congruent to (G, p). In more technical terms, the framework is globally rigid if $\phi^{-1}(\phi(p)) = \phi_K^{-1}(\phi_K(p))$ where ϕ_K denotes the edge function of the framework (K, p) and K is the complete graph with the same number of vertices as G.

Global rigidity is a stricter concept than (plain) rigidity since a framework can be rigid but not globally rigid. In such cases, the framework can be converted to a globally rigid one by the addition of edges (see Figure 1.15 for an example).

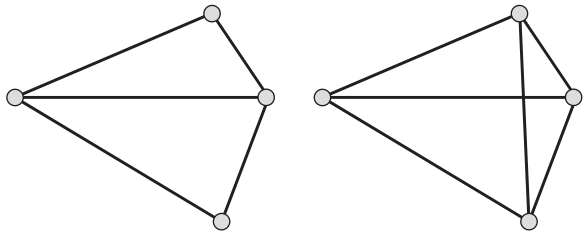

Figure 1.15 (a) Rigidity versus (b) global rigidity.

Global rigidity avoids the occurrence of framework ambiguities since it ensures that the shape of the framework is unique.

1.4 Formation Control Problems

This book will address a number of different formation control problems for multi-agent systems. In this section, we formally introduce these problems.

Consider a system of n mobile agents where $q_i \in \mathbb{R}^m$ is the position of the ith agent relative to an Earth-fixed coordinate frame, and $u_i \in \mathbb{R}^m$ is the corresponding control input. In subsequent chapters, u_i will be a velocity-, acceleration-, or actuator-level input depending on the mathematical model used to describe the agent motion.

Let the *desired* formation for the agents be represented by an *infinitesimally and minimally rigid* framework $F^* = (G^*, q^*)$ where $G^* = (V^*, E^*)$ is the formation graph, $\dim(V^*) = n$, $\dim(E^*) = l$, and $q^* = [q_1^*, \dots, q_n^*]$. The constant desired distance between agents i and j is given by[9]

$$d_{ij} = \|q_i^* - q_j^*\| > 0, \quad i, j \in V^*. \tag{1.25}$$

In practice, the geometric shape/structure of the desired formation is dictated by the mission to be accomplished by the agents. When translating the desired shape into a framework, one needs to include enough edges to ensure that F^* is indeed infinitesimally and minimally rigid.

The *actual* formation of the agents is represented by the framework $F(t) = (G_s, q(t))$ where G_s represents the sensor graph and $q = [q_1, \dots, q_n]$. It is important to clarify the difference between the formation graph G^* and the sensor graph G_s, which in general need not be the same. G^* indicates the minimum number of inter-agent distances that need to be controlled for the desired formation to be successfully reached. On the other hand, G_s indicates the agent pairs that can sense and/or communicate with each other.

We make the following assumptions regarding the desired and actual formations:

Assumption 1 The set where the agents achieve the desired formation is nonempty, i.e., there exist q^* such that $\phi(q^*) = d$ where $d = [..., d_{ij}^2, ...] \in \mathbb{R}^l$ and ϕ was defined in (1.7).

Assumption 2 The formation and sensor graphs are the same, i.e., $G_s = G^*$. Furthermore, inter-agent connectivity is always maintained in the sense that agent i is always within the sensing/communication range of agent j, $\forall j \in \mathcal{N}_i(E^*)$. In other words, G^* is fixed.[10]

---

9 In Sections 2.5 and 3.5 we will consider the case where the desired distances are time varying.

10 Connectivity maintenance prevents the occurrence of flex ambiguities since temporary loss of edges cannot happen.

Assumption 3 At $t = 0$, the agents do not satisfy the desired inter-agent distance constraints, i.e., $\|q_i(0) - q_j(0)\| \neq d_{ij}$, $i, j \in V^*$.

Assumption 4 The only position information being measured is the *relative* position of agent pairs in E^*, $q_i - q_j$, $(i, j) \in E^*$.[11] That is, the global position of the agents, q_i, $i = 1, \ldots, n$, are not available to the control.

We will deal with three types of control problems: formation acquisition, formation maneuvering, and target interception.

Problem 1 ***(Formation Acquisition)*** The goal is for the agents to acquire and maintain a pre-defined geometric shape in space. The control objective for formation acquisition, which serves as the *common*, primary objective for the other two problems, can be mathematically described as to design u_i such that

$$F(t) \to \text{Iso}(F^*) \text{ as } t \to \infty. \tag{1.26}$$

Note that (1.26) is equivalent to

$$\|q_i(t) - q_j(t)\| \to d_{ij} \text{ as } t \to \infty, \quad i, j \in V^*. \tag{1.27}$$

Since only the inter-agent distances are to be directly controlled, the actual formation can converge to any isometry of F^*. That is, the meaning of (1.26) is that the formation will converge to one framework in the set $\text{Iso}(F^*)$ with the specific one being determined by the initial position of the agents, $q_i(0)$, $i = 1, ..., n$. △

Problem 2 ***(Formation Maneuvering)*** The agents are required to simultaneously acquire a formation (i.e., satisfy (1.26)) and maneuver cohesively according to some pre-defined trajectory. Thus, the secondary objective is

$$\dot{q}_i(t) - v_{di}(t) \to 0 \text{ as } t \to \infty, \quad i = 1, \ldots n \tag{1.28}$$

where $v_{di} \in \mathbb{R}^3$ represents the desired rigid body velocity for the swarm of agents. That is, the fixed-shape, desired formation evolves in space as a virtual rigid body undergoing translation and/or rotation.

In practice, the selection of v_{di} is mission dependent. For example, it could be related to a path planning algorithm that provides an optimal solution to the coverage problem where agents cooperatively maximize the coverage area of a given mission under certain time and/or fuel consumption constraints.

When v_{di} only includes a translation velocity, the formation maneuvering problem is also called *flocking*. For the case where v_{di} has a rotational component, we assign the nth agent (without lost of generality) to be the "leader" while

11 As we will see in the following chapters, the control could also be a function of other, nonposition-related variables depending on the agent model and formation problem being solved.

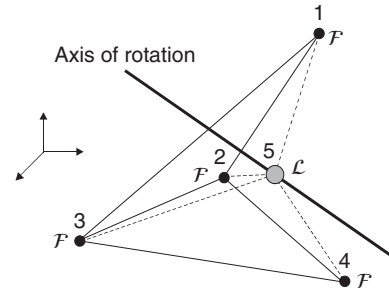

Figure 1.16 Example of the construction of F^*: a tetrahedron formation where $\mathcal{L}$ stands for leader and $\mathcal{F}$ for follower.

the remaining agents are "followers". This assignment is for the sole purpose of one agent serving as a reference point for the axis of rotation of the virtual rigid body (i.e., point O in Figure 1.7). Therefore, F^* should be constructed with the following additional conditions:

- $q_n^* \in \text{conv}\{q_1^*, ..., q_{n-1}^*\}$;
- $(i, n) \in E^*$, $i = 1, \dots, n-1$, i.e., there is an edge between each follower and the leader.

An example of F^* is illustrated by the 3D formation in Figure 1.16 where the leader is located in the interior of the tetrahedron. The axis of rotation passes through the leader, which is inside the tetrahedron. Since $n = 5$, we need $3n - 6 = 9$ for the framework to be minimally rigid. The solid lines indicate edges that form the faces of the tetrahedron while the dashed lines are edges in its interior. Notice that edge $(1, 4)$ is not necessary.

The association of a leader agent (instead of a *virtual* leader) with the axis of rotation is done for convenience (not necessity) since the leader's relative position to the followers can be measured and it will not have to undergo any rotation. Note that if one uses a virtual leader, its location would have to be known in order to calculate its position relative to the agents (see (1.10)). This in turn would require extra measurements and/or calculations. △

Problem 3 ***(Target Interception)*** The agents should intercept and surround a (possibly evading) moving target with a pre-defined formation. Here, we will also use the leader–follower approach by taking the nth agent to be the leader while the remaining agents are followers. The control protocol will consist of: (i) selecting F^* such that $q_n^* \in \text{conv}\{q_1^*, ..., q_{n-1}^*\}$[12], (ii) the leader chasing the target, and (iii) the followers tracking the leader while maintaining the desired formation. Thus, if $q_T \in \mathbb{R}^m$ denotes the target position, the secondary

12 Unlike formation maneuvering with rotation, we do not need $(i, n) \in E^*$, $i = 1, \dots, n-1$ for target interception.

objective for this problem is that $q_T(t)$ approach $\text{conv}\{q_1(t), q_2(t), ..., q_{n-1}(t)\}$ as time evolves, which (with abuse of notation) we express as

$$q_T(t) \in \text{conv}\{q_1(t), q_2(t), ..., q_{n-1}(t)\} \text{ as } t \to \infty. \tag{1.29}$$

△

1.5 Book Overview and Organization

The subsequent chapters of this book will introduce a class of graph rigidity-based formation controllers for multi-agent systems. The control laws primarily stabilize the inter-agent distance dynamics to desired distances (formation acquisition) using system models of increasing complexity and accuracy: single integrator, double integrator, and robotic vehicle kinematics and dynamics. For problems of formation maneuvering and target interception, the control will contain additional terms to account for these secondary objectives. The inter-agent distance dynamics and, consequently, the control laws will be formulated in terms of the rigidity matrix. The stability of these dynamics in closed loop with the control will depend on the rigidity matrix having full row rank, thus the need for requiring that the desired formation F^* is infinitesimally and minimally rigid (see Corollary 1.1). The book includes theoretical, simulation, and experimental results, and is organized as follows.

We naturally begin the presentation with the simple single-integrator model in Chapter 2. Here, we first design a formation acquisition controller that ensures that the desired formation is exponentially stable. The control is distributed in the sense that the control input of each agent is only a function of the relative position of neighboring agents in the graph. The formation acquisition controller is the *basic* control term since it will appear in all other control algorithms developed in the book. Building upon this result, we show how the formation acquisition control can be augmented with a term to enable the agents to perform formation maneuvering or target interception simultaneously with formation acquisition. In the formation maneuvering3 problem, the swarm (group) velocity is first assumed to be known to all agents. We then consider a variant of this problem where the agents are just flocking with a constant velocity; however, this velocity is known only to a subset of agents. An observer is introduced to estimate the flocking velocity. Next, the idea behind the control design for formation maneuvering is extended to the target interception problem. We assume the target's relative position to the leader agent is known and broadcast to the followers; however, the target's velocity is *unknown*to all agents. To deal with this uncertainty, the target interception component of the control law will contain a continuous dynamic robust mechanism, inspired by the work in [24], to estimate the target velocity. Our stability analysis for both problems provides exponential formation

acquisition and asymptotic formation maneuvering or target interception. We close this chapter with the case where the desired formation is *dynamic* rather than static. This is motivated by situations, such as obstacle avoidance and limited communication range/bandwidth, where the formation size and/or geometric shape need to vary in time. We also briefly discuss how the control law can be modified to perform formation maneuvering on top of tracking the time-varying formation.

In Chapter 3, the results from Chapter 2 are extended to the double-integrator model. The backstepping control technique is a natural tool for solving this problem since it allows us to treat the velocity-level inputs designed in Chapter 2 as *fictitious* control inputs, which are to be tracked by the new, acceleration-level inputs. For formation acquisition, the control input of each agent is dependent on the relative position and relative velocity of neighboring agents in the graph and the agents' own velocity. The formation maneuvering control is dependent on the desired swarm acceleration in addition to the velocity. In the target interception problem, we assume the target's relative position to the leader and velocity are known and can be broadcast to the followers; however, the target's *acceleration* is unknown to all agents. To deal with this uncertainty, the target interception component of the control law will contain a variable structure-type term to compensate for the unknown target acceleration. The stability results of Chapter 2 are preserved in these results.

In Chapter 4, we account for the nonlinear kinematics and dynamics of the agents during the control design process. The control here is limited to the formation acquisition problem. We consider a class of robotic vehicles moving on the plane, such as unicycle robots, marine (surface) vessels, underwater vehicles with constant depth, and aircraft with constant altitude. We first deal with the nonholonomic kinematic equation only, and show how the formation acquisition control from Chapter 2 can be indirectly used. We then account for the vehicle dynamics by transforming the equations of motion into an Euler–Lagrange-like system so we can exploit its structural properties in the control design and stability analysis. The backstepping technique is again applied to incorporate the velocity-level inputs from Chapter 2 into the torque-level control law in a rigorous manner. We consider the cases where the dynamics are fully known as well as subject to parametric uncertainty. In the latter case, we show how the control law can be redesigned with parameter adaptation to compensate for the uncertainties.

The book culminates with the demonstration of the experimental implementation of the controllers from Chapters 2, 3, and 4 on an actual car-like robotic platform. Three customized, unmanned ground vehicles (UGVs) were used for this purpose. We show how the high-level, formation algorithms can be embedded in the motor-level commands to the UGVs in order to acquire and maneuver a given triangular formation.

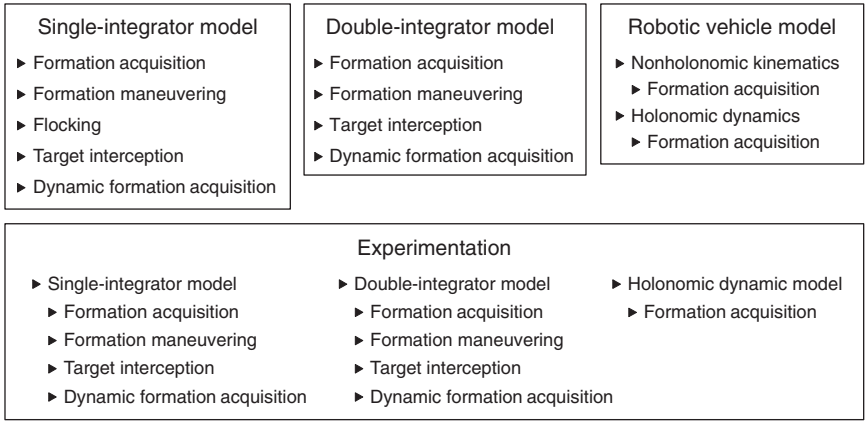

Figure 1.17 Overview of the book organization.

An overview of the book organization is shown in Figure 1.17.

1.6 Notes and References

In-depth coverage of rigid graph theory can be found in, for example, [13–18, 25–29]. Rigid graphs were first applied to formation control in [30]. Some early work on the application of graph theory to multi-agent formation appeared in [31–35]. An overview of rigid graph theory and its application to sensing, communication, and control architectures for formations of autonomous vehicles was presented in [6]. A number of books have been published in the past few years that deal with the general topic of cooperative control of multi-agent systems [2–4, 37–39]. A recent survey of multi-agent formation control was provided in [40].