# Chapter 1
# Introduction

## 1.1 Motivation

The problem of coordinated control of a network of mobile autonomous robots is of interest in control and robotics because of the broad range of potential applications: planetary exploration, operations in hazardous environments, games such as robot soccer, and so on. Distributed robot networks can potentially exhibit structural flexibility, reliability through redundancy, and simple hardware as compared to a complex individual robot.

The first robot rover exploration of Mars was in 1997—the Mars Pathfinder Mission. The rover, named Sojourner, is shown in Fig. 1.1. You can see the whip antenna. The radio link was used to send commands from Earth to the rover and receive images and other data from the rover. Because the rover radio had a signal range similar to a walkie-talkie, namely, about 10 m, all rover communication was done with the aid of the lander communications interface, as in Fig. 1.2. The rover telecommunications system was a two-way wireless UHF (Ultra High Frequency) radio link between the lander and the rover. The rover's and lander's UHF antennas worked very much like the antennas on walkie talkies or on car radios, using a "monopole" antenna. The signal to be transmitted enters the antenna through a coaxial connector located at the bottom, travels through a short section of balanced coaxial line, and is radiated by the monopole.

It is desirable to have an antenna radiation pattern shaped to match its particular application. Satellite dishes are designed to look at a particular location in space and therefore need to have narrow and directive radiation patterns. The rover antenna did not need to look up into space, but rather needed to look horizontally in 360° given that the lander could be in any direction. An ideal monopole has a 360° radiation pattern that is donut shaped, oriented horizontally. It is not meant to look straight up, and has poor reception in that direction. Certain metallic or rocky structures and ground reflections near the monopole antenna will distort its radiation pattern and cause holes or null zones to form. In these null zones the signal can drop significantly,

 
B.A. Francis and M. Maggiore, *Flocking and Rendezvous in Distributed Robotics*, SpringerBriefs in Control, Automation and Robotics,
DOI 10.1007/978-3-319-24729-8_1


**Fig. 1.1** Sojourner (Jet Propulsion Lab, NASA). (This image is in the public domain and was downloaded from the Web)

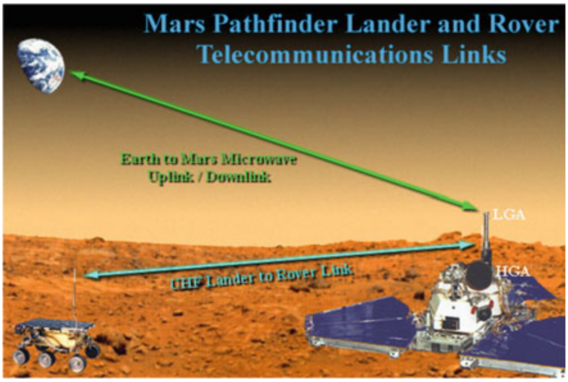


**Fig. 1.2** Communications interface (NASA). (This image is in the public domain and was downloaded from the Web)

causing poor reception. It is important to know where the rover is relative to the lander when these null zones exist, for if two nulls happened to get pointed at each other, there may be no radio reception at all.

As discussed, Sojourner had to be within about 10 m of the lander to send radio signals. This obviously is a limitation for scientific experiments. To get a longer radio link, one could use a higher power signal. But power on a Mars robot is a luxury. Another solution is an antenna array.

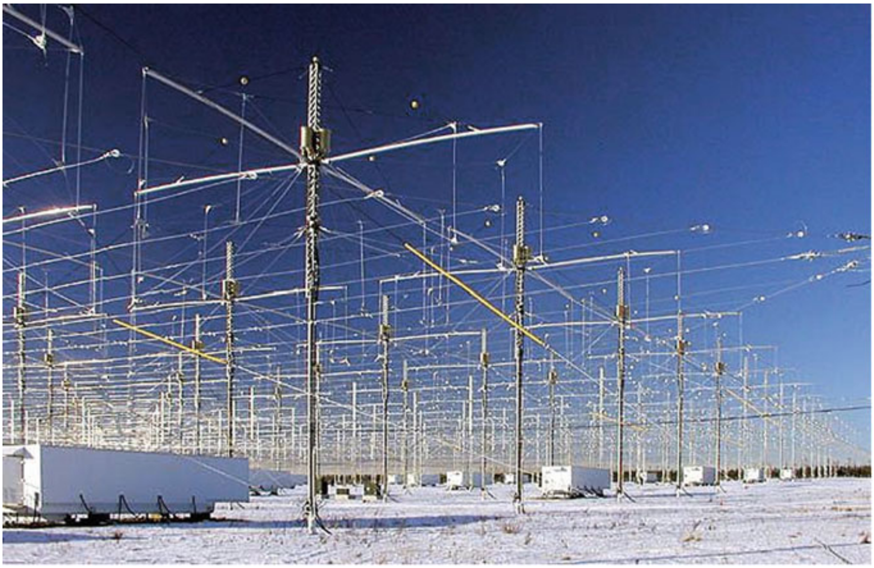

**Fig. 1.3** High Frequency Active Auroral Research Program. (This image is in the public domain and was downloaded from the Web)

The purpose of an antenna array is to achieve directivity, the ability to send the transmitted signal in a preferred direction. If a large number of array elements can be used, it is possible to greatly enhance the strength of the signal transmitted in a given direction. An interesting example is the High Frequency Active Auroral Research Program (HAARP) in Alaska, whose purpose is to study the ionosphere. The site consists of a $15 \times 12$ array of dipole antennas: see Fig. 1.3.

This leads us to the following scenario. A team of rovers doing scientific experiments. Each has, besides scientific instruments, a radio transceiver and an antenna. When it is time to communicate with the lander, the rovers arrange themselves in a suitable formation to become an antenna array in order to optimize the signal strength. In general, the larger the array, the higher the resolution it can achieve.

The preceding example leads to the sub-question: can we get a group of robot rovers, placed initially at random, to form a circle or other shape? We study this question for the simplest possible model of a robot, a point moving in the plane, then for a model of a wheeled rover moving in the plane, and finally for a quadcopter in 3D space.

This monograph is about control theoretic robotics problems. There is frequently a hubbub about the gap between theory and practice. Let us be clear about that: Real problems cannot be solved just by applying formulas. So the methodology of control engineering is to begin with a real problem; to abstract the central issues and formulate an idealized, hypothetical problem; to develop, if necessary, new mathematical methods for its solution; and to work out a rigorous solution. Then one has a framework on which to do the real problem.

For more about robots in space, go to http://www.jpl.nasa.gov.

## 1.2 Models, Sensing, and Control Specifications

In this monograph we present the two most basic distributed robotics problems: flocking and rendezvous. The flocking problem is to get all the robots to move in the same direction at the same speed; the rendezvous problem is to get all the robots to converge to the same meeting point. Such objectives are achieved by, possibly among other factors, interacting with other robots. We call these others **neighbours**. Besides the two objectives of flocking and rendezvous, one may characterize the setup by how neighbours are defined: by proximity or just fixed from the start. For example, one may distinguish $n$ robots by numbering them and displaying the numbers on them. Then the neighbour structure could be sequential, like this: 1's neighbour is 2, 2's neighbour is 3; etc.; $n$'s neighbour is 1. This is called **cyclic pursuit** and is an example of a fixed neighbour structure. Alternatively, the neighbours of robot $i$ could be all other robots within, say, $d$ metres of robot $i$. This is an example of a proximity-based neighbour structure.

We see from the discussion above that there are three dimensions to consider when classifying distributed robotics problems: the model of the robot one wishes to control (e.g., the unicycle model); the sensing constraint (what sensors are available, and who can see whom at any given time); and the control specification (e.g., rendezvous). In this book we focus on three model classes: integrator points, kinematic unicycles, and flying vehicles. We present two types of sensing constraints:

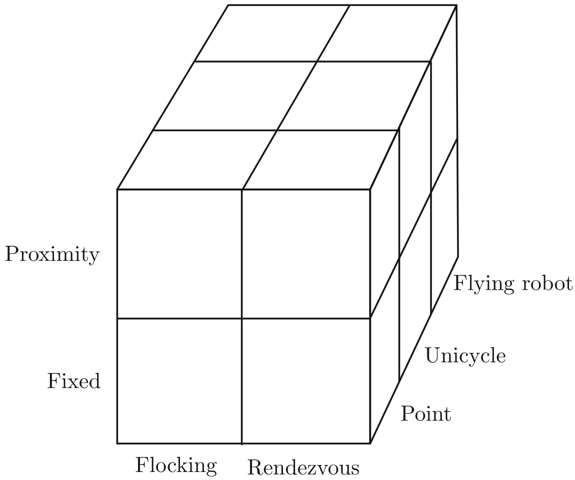


**Fig. 1.4** Twelve problems: the goal can be flocking or rendezvous; the robots can be integrator points, unicycles, or flying vehicles; the neighbour sets can be fixed or proximity dependent

fixed neighbour structure and proximity-based neighbour structure. Finally, we investigate two control specifications: flocking and rendezvous. In this way we have $2 \times 2 \times 3 = 12$ problems (Fig. 1.4).

It will turn out that not all twelve problems make sense, since flocking is a degenerate problem for integrator points. Moreover, many of these problems are as yet open. In the case of a proximity-based neighbour structure, flocking is an open problem for all model classes, and rendezvous has been solved only for integrator points. In the case of a fixed neighbour structure, the flocking problem has been solved for both unicycles and flying vehicles, while the rendezvous problem has been solved only for integrator points and unicycles.

## 1.3 Notation

The notation follows fairly standard conventions in signals and systems. The set of integers—negative, zero, and positive—is denoted $\mathbb{Z}$. Continuous time and discrete time are both denoted by $t$; context will determine whether $t$ is a real number ($t \in \mathbb{R}$) or an integer ($t \in \mathbb{Z}$). Dot, as for example $\dot{x}$, denotes derivative with respect to time $t$.

A vector in $\mathbb{R}^2$ is written as an ordered pair

$$x = (x_1, x_2)$$

or as a column vector

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

whichever is more convenient at the time. Likewise, we might associate

$$(u, v) \text{ and } \begin{bmatrix} u \\ v \end{bmatrix}$$

where $u$ is an $m$-tuple and $v$ an $n$-tuple. This permits us to avoid ugly expressions like $\begin{bmatrix} u^T & v^T \end{bmatrix}^T$ for a column vector.

Mathematically, we can regard the plane as being the Euclidean plane $\mathbb{R}^2$ or the complex plane $\mathbb{C}$. If $x = (x_1, x_2)$, $y = (y_1, y_2)$ are two real vectors in $\mathbb{R}^2$, their dot product is written

$$\langle x, y \rangle = x_1 y_1 + x_2 y_2.$$

Likewise and equivalently, members of $\mathbb{C}$ are written $x_1 + jx_2$. If $x = x_1 + jx_2$ and $y = y_1 + jy_2$, to be consistent with $\mathbb{R}^2$ the dot product of $x$ and $y$ is defined to be (overbar denotes complex conjugate and Re denotes real part)

$$\begin{aligned}\langle x, y\rangle &= \text{Re } x\overline{y}\\ &= \text{Re } (x_1 + jx_2)(y_1 - jy_2)\\ &= x_1y_1 + x_2y_2\\ &= \text{Re } \overline{x}y.\end{aligned}$$

Finally, we let $\{e_1, \ldots, e_n\}$ denote the natural basis of $\mathbb{R}^n$. Thus the vector $e_i$ has a one in the $i$th position and all other elements are zero.