1

Introduction

1.1 System Architecture

The objective of this book is to prepare the reader to do research in the exciting and rapidly developing field of autonomous navigation, guidance, and control of unmanned air vehicles. The focus is on the design of the software algorithms required for autonomous and semi-autonomous flight. To work in this area, researchers must be familiar with a wide range of topics, including coordinate transformations, aerodynamics, autopilot design, state estimation, path planning, and computer vision. The aim of this book is to cover these essential topics, focusing in particular on their application to small and miniature air vehicles, which we denote by the acronym MAV.

In the development of the topics, we have in mind the software architecture shown in figure 1.1. The block labeled *unmanned aircraft* in figure 1.1 is the six-degree-of-freedom (DOF) physical aircraft that responds to servo command inputs (elevator, aileron, rudder, and throttle) and wind and other disturbances. The mathematical models required to understand fixed-wing flight are complicated and are covered in chapters 2 to 5 and chapter 9. In particular, in chapter 2 we discuss coordinate frames and transformations between frames. A study of coordinate frames is required since most specifications for MAVs are given in the inertial frame (e.g., orbit a specific coordinate), whereas most of the sensor measurements are with respect to the body frame, and the actuators exert forces and torques in the body frame. In chapter 3 we develop the kinematic and dynamic equations of motion of a rigid body. In chapter 4 we describe the aerodynamic forces and moments that act on fixed-wing aircraft. Chapter 5 begins by combining the results of chapters 3 and 4 to obtain a six-DOF, 12-state, nonlinear dynamic model for a MAV. While incorporating the fidelity desired for simulation purposes, the six-DOF model is fairly complicated and cumbersome. The design and analysis of aircraft control approaches are more easily accomplished using lower-order linear models. Linear models that describe small deviations from trim are derived in chapter 5, including linear transfer function and state-space models.

The block labeled *autopilot* in figure 1.1 refers to the low-level control algorithms that maintain roll and pitch angles, airspeed, altitude, and course heading. Chapter 6 introduces the standard technique of

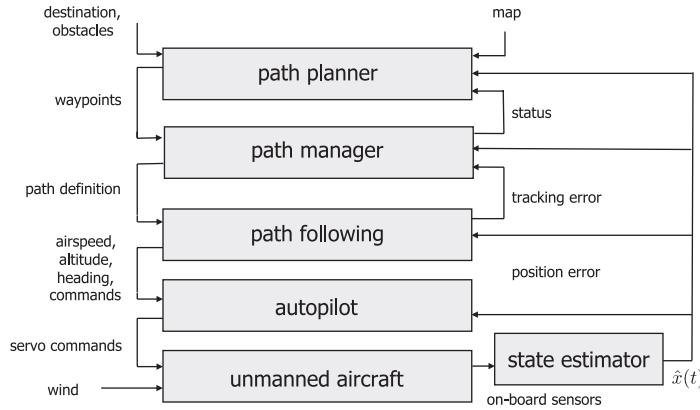

Figure 1.1 The system architecture that will be assumed throughout the book. The path planner produces straight-line or Dubins paths through an obstacle field. The path manager switches between orbit following and straight-line path following to maneuver along the waypoint paths. The path-following block produces commands to the low-level autopilot, which controls the airframe. Each of the blocks relies on estimates of the states produced by filtering the onboard sensors.

successive loop closure to design the autopilot control laws. Nested control loops are closed one at a time, with inner loops maintaining roll and pitch angles and outer loops maintaining airspeed, altitude, and course.

The autopilot and the higher level blocks rely on accurate state estimates obtained by dynamically filtering the onboard sensors, which include accelerometers, rate gyros, pressure sensors, magnetometers, and GPS receivers. A description of these sensors and their mathematical models is given in chapter 7. Because it is not possible to measure all the states of small unmanned aircraft using standard sensors, state estimation plays an important role. Descriptions of several state-estimation techniques that are effective for MAVs are given in chapter 8.

A complete model of the flight dynamics coupled with the autopilot and state estimation techniques represents a high dimensional, highly complex, nonlinear system of equations. The full model of the system is too complicated to facilitate the development of high level guidance algorithms. Therefore, chapter 9 develops low-order nonlinear equations that model the closed-loop behavior of the system. These models are used in subsequent chapters to develop guidance algorithms.

One of the primary challenges with MAVs is flight in windy conditions. Since airspeeds in the range of 20 to 40 mph are typical for MAVs,

and since wind speeds at several hundred feet above ground level (AGL) almost always exceed 10 mph, MAVs must be able to maneuver effectively in wind. Traditional trajectory tracking methods used in robotics do not work well for MAVs. The primary difficulty with these methods is the requirement to be in a particular location at a particular time, which cannot properly take into account the variations in ground speed caused by the unknown and changing effects of the wind. Alternatively, path-following methods that simply maintain the vehicle on a desired path have proven to be effective in flight tests. Chapter 10 describes the algorithms and methods used to provide the capabilities of the *path following* block in figure 1.1. We will focus exclusively on straight-line paths and circular orbits and arcs. Other useful paths can be built up from these straight-line and circular path primitives.

The block labeled *path manager* in figure 1.1 is a finite-state machine that converts a sequence of waypoint configurations (positions and orientations) into sequences of straight-line paths and circular arcs that can be flown by the MAV. This makes it possible to simplify the path planning problem so that the *path planner* produces either a sequence of straight-line paths that maneuver the MAV through an obstacle field, or a Dubin's path that maneuvers through the obstacle field. Chapter 11 describes the *path manager*, while chapter 12 describes the *path planner*. For path planning we consider two classes of problems. The first class of problems is point-to-point algorithms, where the objective is to maneuver from a start position to an end position while avoiding a set of obstacles. The second class of problems is search algorithms, where the objective is to cover a region, potentially having no-go regions, with a sensor footprint.

Almost all applications involving MAVs require the use of an onboard electro-optical/infrared (EO/IR) video camera. The typical objective of the camera is to provide visual information to the end user. Since MAV payload capacities are limited, however, it makes sense to also use the video camera for navigation, guidance, and control. Effective use of camera information is currently an active research topic. In chapter 13 we discuss several potential uses of video cameras on MAVs, including geolocation and vision-based landing. Geolocation uses a sequence of images as well as the onboard sensors to estimate the world coordinates of objects on the ground. Vision-based landing uses video images captured by the MAV to guide it to a target identified in the image plane. We feel that an understanding of these problems will enable further investigations in vision-based guidance of MAVs.

In chapter 13, we use the software architecture shown in figure 1.2, where the *path planner* block has been replaced with the block *vision-based guidance*. However, the vision-based guidance laws interact with

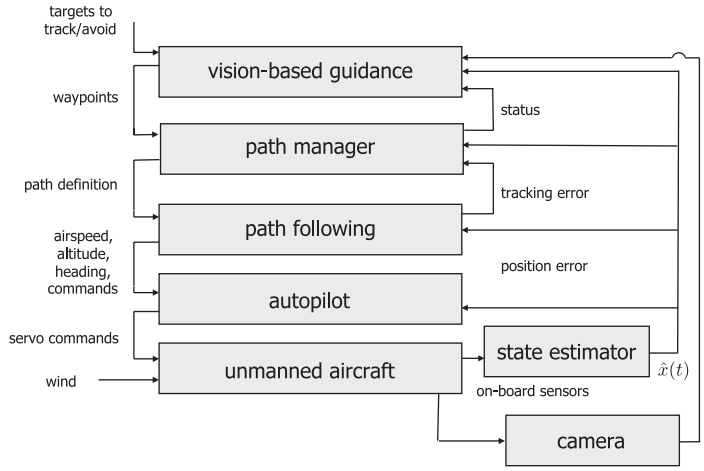

Figure 1.2 System architecture for vision-based navigation, guidance, and control. A video camera is added as an additional sensor and the path planner has been replaced with a vision-based guidance block.

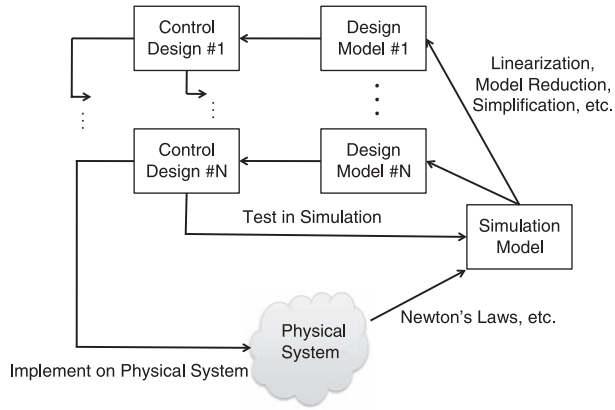

Figure 1.3 The design process. Using principles of physics, the physical system is modeled mathematically, resulting in the simulation model. The simulation model is simplified to create design models that are used for the control design. The control design is then tested and debugged in simulation and finally implemented on the physical system.

the architecture in the same manner as the path planner. The modularity of the architecture is one of its most appealing features.

1.2 Design Models

The design philosophy that we follow throughout the book is illustrated schematically in figure 1.3. The unmanned aircraft operating in its

environment is depicted in figure 1.3 as the "Physical System," and includes the actuators (control flaps and propeller) and the sensors (IMU, GPS, camera, etc.). The first step in the design process is to model the physical system using nonlinear differential equations. While approximations and simplifications will be necessary at this step, the hope is to capture in mathematics all of the important characteristics of the physical system. In this book, the model of the physical system includes rigid body kinematics and dynamics (chapter 3), aerodynamic forces and moments (chapter 4), and the onboard sensors (chapter 7). The resulting model is called the "Simulation Model" in figure 1.3 and will be used for the high fidelity computer simulation of the physical system. However, we should note that the simulation model is only an approximation of the physical system, and simply because a design is effective on the simulation model, we should not assume that it will function properly on the physical system.

The simulation model is typically nonlinear and high order and is too mathematically complex to be useful for control design. Therefore, to facilitate design, the simulation model is simplified and usually linearized to create lower-order design models. For any physical system, there may be multiple design models that capture certain aspects of the design process. For MAVs, we will use a variety of different design models for both low-level control and also for high-level guidance. In chapter 5, we will decompose the aircraft motion into longitudinal (pitching and climbing) motion and lateral (rolling and heading) motion, and we will have different design models for each type of motion. The linear design models developed in chapter 5 will be used in chapter 6 to develop low-level autopilot loops that control the airspeed, altitude, and course angle of the vehicle. In chapter 8, we show how to estimate the states needed for the autopilot loops using sensors typically found on small and micro air vehicles.

The mathematical equations describing the physics of the system, the low-level autopilot, and the state estimation routines, when considered as a whole, are very complex and are not useful for designing the higher level guidance routines. Therefore, in chapter 9 we develop nonlinear design models that model the closed-loop behavior of the system, where the input is commanded airspeed, altitude, and course angle, and the outputs are the inertial position and orientation of the aircraft. The design models developed in chapter 9 are used in chapters 10 through 13 to develop guidance strategies for the MAV.

As shown in figure 1.3, the design models are used to design the guidance and control systems. The designs are then tested against the high fidelity simulation model, which sometimes requires that the design models be modified or enhanced if they have not captured the essential

features of the system. After the designs have been thoroughly tested against the simulation model, they are implemented on the physical system and are again tested and debugged, sometimes requiring that the simulation model be modified to more closely match the physical system.

1.3 Design Project

In this textbook we have decided to replace traditional pencil-and-paper homework problems with a complete and rather extensive design project. The design project is an integral part of the book, and we believe that it will play a significant role in helping the reader to internalize the material that is presented.

The design project involves building a MAV flight simulator from the ground up. The flight simulator will be built using Matlab/Simulink, and we have specifically designed the assignments so that additional add-on packages are not required.[1] The website for the book contains a number of different Matlab and Simulink files that will assist you in developing the flight simulator. Our strategy is to provide you with the basic skeleton files that pass the right information between blocks, but to have you write the internal workings of each block. The project builds upon itself and requires the successful completion of each chapter before it is possible to move to the next chapter. To help you know when the design from each chapter is working, we have included graphs and pictures on the website that show the output of our simulator at each stage.

The project assignment in chapter 2 is to develop an animation of an aircraft and to ensure that you can properly rotate the body of the aircraft on the screen. A tutorial on animating graphics in Matlab is provided in appendix C. The assignment in chapter 3 is to drive the animation using a mathematical model of the rigid body equations of motion. In chapter 4 the force and moments acting on a fixed wing aircraft are added to the simulation. The assignment in chapter 5 is to use the Simulink commands `trim` and `linmod` to find the trim conditions of the aircraft and to derive linear transfer function and state-space models of the system. The assignment in chapter 6 adds an autopilot block that uses the real states to control the aircraft. In chapter 7, a model of the sensors is added to the simulator, and in chapter 8, state estimation

[1] We have also taught the course using the public domain flight simulator Aviones, which is available for download at Sourceforge.net. For those who do not have access to Matlab/Simulink and would prefer to develop the project in C/C++, we encourage the use of Aviones.

schemes are added to estimate the states needed for the autopilot using the available sensors. The result of the project assignment in chapter 8 is a closed-loop system that controls airspeed, altitude, and course angle using only available sensor information. The assignment in chapter 9 is to approximate the closed-loop behavior using simple design models and to tune the parameters of the design model so that it essentially matches the behavior of the closed-loop high-fidelity simulation. The assignment in chapter 10 is to develop simple guidance algorithms for following straight-lines and circular orbits in the presence of wind. In chapter 11, straight-line and orbit following are used to synthesize more complicated paths, with emphasis on following Dubins paths. The assignment in chapter 12 is to implement the RRT path planning scheme to plan Dubins paths through an obstacle field. The project assignment in chapter 13 is to point a camera at a moving ground target and to estimate the inertial position of the target using camera data and onboard sensors (geolocation).