Chapter 1

Introduction

"A Robot Ping-Pong Player: Experiment in Real-Time Intelligent Control" analyzes the construction of the first robot to play ping-pong* against humans and machines (Figure 1). What are the limitations of present robots? What makes ping-pong hard? Why investigate ping-pong? What are the applications of the technology we developed? The following introductory chapter will address these issues, then provide an overview of the book's organization.

1.1 Introduction to Robots and Their Limitations

The general public thinks of R2D2 and C3PO when someone mentions robots, but today's robots are far from that stage, mechanically or intellectually. When confronted with a real robot, the mechanism makes the biggest impression. However, in our (admittedly biased) view, it is not the mechanism that is of greatest interest, but the computer system and its programming that controls the mechanism. Figure 2 divides the robot control system into three levels. Each level maintains a specific duration of foresight, with a corresponding constraint on the time available for planning.

The robot mechanism occupies the hierarchy's lowest level: motors, encoders, and their projections into the electronic domain. Specialized joint servo processors interact with the interfaces of each joint, implementing control algorithms that cause the robot to attain specified positions, velocities, or torques. The actuator bandwidths restrict the available processing time to a millisecond per iteration. The joint servos

* Ping-Pong is a trademark of Parker Brothers.

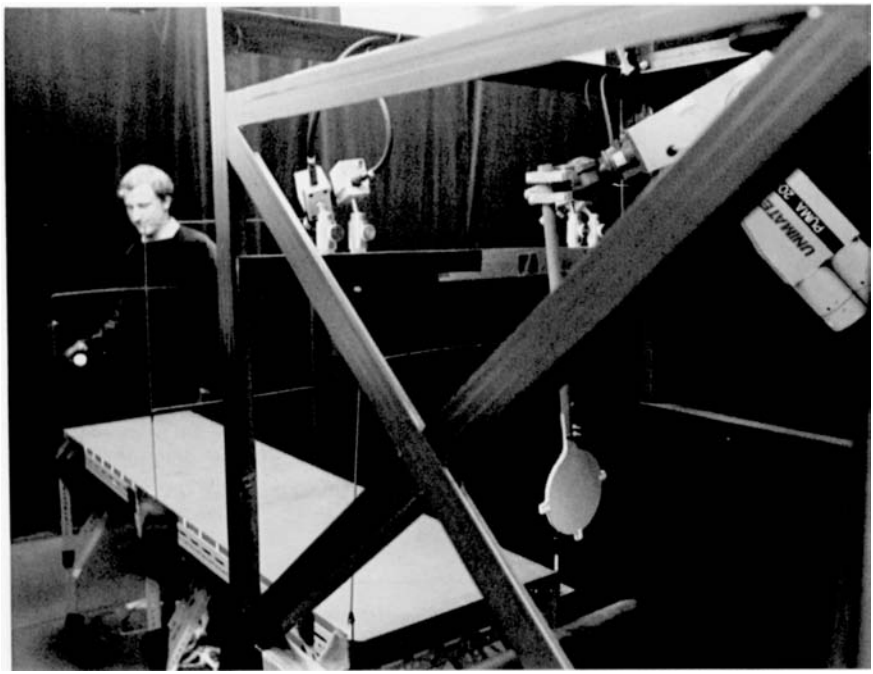

Figure 1. Robot and Environment. The robot ping-pong player takes on a human opponent, the author, at robot ping-pong. Two of four video cameras and the robot arm lurk in the supporting framework.

present a clean electronic interface to the middle control level, which can specify desired robot trajectories.

At the hierarchy's summit, a high-level planner generates objectives for the robot and monitors its performance. The planner manipulates an abstract view of the robot system, even though it may supply details such as the sizes of parts from a CAD/CAM database. The planner is slower than and desynchronized from robot operation, but must be able to monitor and modify the robot system to improve its performance. Either a program or a human may implement the high-level planner.

The intermediating controller translates the planner's demands to the servo controller's realities. The controller must plan a sequence of trajectories for the servos which satisfy both the abstract requirements of

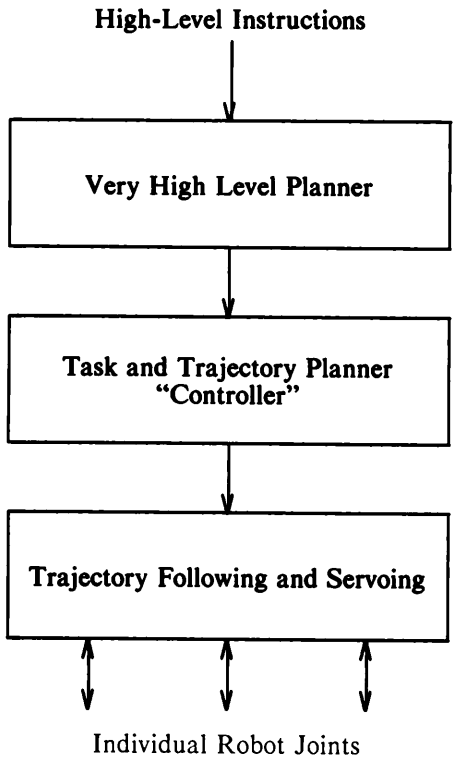

Figure 2. High-Level Robot Diagram. The planner is an artificial or natural intelligence system. We will be concerned with the controller, which must integrate the task and robot constraints.

the task and the complex and concrete limitations of the physical apparatus. In the face of ignorance, the controller has no choice but to adopt very conservative estimates of the manipulator's abilities. Note that the generic term "robot controller" refers to both the task and trajectory planner, and the lower-level servoing functions.

Consider a conventional robot feeding a punch. The high-level planner might produce this plan:

```
move_to input_bin
wait_for part_present
withdraw part
move_to punch
activate punch
wait_for punch_done
withdraw part
wait_for output_empty
move_to output_bin
release part
```

The robot operates in a world that changes in very discrete steps, in response to clearly defined causes. Robot actions interlock with binary status lines and control signals to the outside world, such as "`part_present`" or "`activate punch`." Sensors such as vision can be brought into this framework by restricting their output to discrete events (though their function is also limited to start with), for example, take a picture now, the part is missing, the part is bad, or a good part is at (10 cm, 15 cm, 45°).

The robot moves from one state to another in indivisible steps: the time variable is effectively suppressed. In this "discrete-time" scenario, the physics of the environment and robot motion are considered to take place in series, because the robot assumes the world is static while it moves. The high-level planner need not consider how long any operation might take. A simple robot controller might take as long as it wants to execute the task, in the absence of temporal constraints. It need not know exactly how fast the arm can move, but can move deliberately, though it sacrifices cost-effectiveness to do so. Although we can solve many useful tasks this way, the approach begins to break down as the robot, task, and environment become more sophisticated.

Consider retrieving an object from a conveyor belt. To save time and fixturing, we wish to do so without stopping the conveyor. The manipulator must be at the right place at the right time at the right velocity to make a smooth pickup. In contrast to the discrete-time punch feeder, a continuous-time approach is required. The continuous-time robot must model the physics of the environment as it evolves during robot operation. Continuous-time systems require high sensor bandwidth — many sensor data points per second, and low latency — the time from the acquisition of data until it is applied to the control output. The lack of adequate sensor systems has hampered previous investigation of this area.

The conventional approach to the conveyor problem is to find the object's position from a single snapshot, then update the position using an encoder mechanically mounted on the conveyor belt [49]. Servo equations

cause the robot to track the object. Pragmatically, this is fine for low-accuracy, low-speed applications.

To increase the robot's object-retrieval rate, the system must pick a distance by which to lead the object, such that the robot can arrive at that position at the same time as the object. To lead by as little as possible, we must know how fast the arm can move. If we have a good sensor system, we can continue to watch the object while the arm moves, fine-tuning the arm's trajectory.

In this book we will investigate the construction of a sophisticated continuous-time system, the robot ping-pong player. We will see that new problems are encountered, and new techniques must be applied, to create a robust, functioning system.

We claim that robot controllers already limit robots' performance. We can increase the robot system's speed and functionality by increasing the controller's knowledge of the manipulator's physical capabilities, and by providing effective ways to use this information.

To accomplish the robot ping-pong task, we have created an "expert controller" able to operate in the symbolic domain of the high-level planner, and the numeric domain of the low-level system. The expert controller integrates a specialized expert system and robot controller. The robot controller still exists to perform low-level functions, but the expert controller replaces and extends the robot controller's upper levels. The expert controller must exploit its knowledge of the task and robot characteristics to generate a working plan that satisfies the constraints of both task and robot. The system design is drastically affected by the need for rapid response and the need to compensate for, and be robust to, new sensor data.

Of course, we can't investigate a particular robot subsystem in isolation — we need to have working sensor, low-level control, and actuator components as well. The expert controller places its own requirements on their design. Each component poses its own problems which must be solved to create a working system.

1.2 Introduction to Robot Ping-Pong

Let us briefly overview the ping-pong system and outline the function of its components. We use the robot ping-pong rules proposed by Billingsley [8] as an international challenge and competition. The most important differences from human ping-pong are that the table is narrower than the human table, and that wire frames have been added at each end and the

middle through which the ball must pass (Figure 3). The geometry places a premium on accurate placement, rather than raw speed, because the net's height limits the ball's maximum speed. The game can be played by a small stationary robot, but is challenging for both man and machine.

The computer implementation can be divided into four stages, each residing on one or more computers: a 3-D vision system, which locates the ball each sixtieth of a second; a trajectory analyzer, which determines and extrapolates the ball's trajectory; an expert controller, by far the most complex component, which implements the ping-pong strategy to produce the robot's desired trajectory; and a servo system, which causes the robot to follow the trajectory.

To understand what a robot ping-pong system must do, let us consider the decisions which must be made in the course of the return of a single ball, starting from the point at which it is hit by the opponent.

We visually track the ball, finding the (x,y,z,t) position of the ball at the end of each camera frame. After several frames, we fit some trajectory to the data, including the effect of air drag and spin. The observed trajectory is used to predict the future trajectory of the ball after it bounces on the table. We continually monitor the ball trajectory to detect infringements of the rules, and award points as necessary.

The robot must pick the position along the post-bounce trajectory at which the ball will be hit, establishing the hit position and time. A trajectory to return the ball to the opponent must be computed, considering the incoming trajectory, knowledge of the rules, and some strategy. The paddle orientation and velocity that transfers the ball from one trajectory to the other must be computed from knowledge of the paddle's characteristics.

Although ping-pong requires only five degrees of freedom (three of position, two of orientation), the robot has six. The extra dimension corresponds to rotating the paddle's handle in the plane normal to the paddle's orientation vector. Having chosen a value for this degree of freedom, the robot's joint angles may be computed.

Robot motions are then planned so that the motions complete (moving joints stabilize at constant velocity) slightly before the hit instant, taking into account the acceleration capabilities of the manipulator. Heavily stressed joints may have no setup time at all. Once the motion has been planned, the robot may be started towards the ball.

In the meantime, the ball continues moving towards the robot. We must continue to watch the ball until it is hit, refining our motion to make it as likely as possible that the ball will be returned as planned.

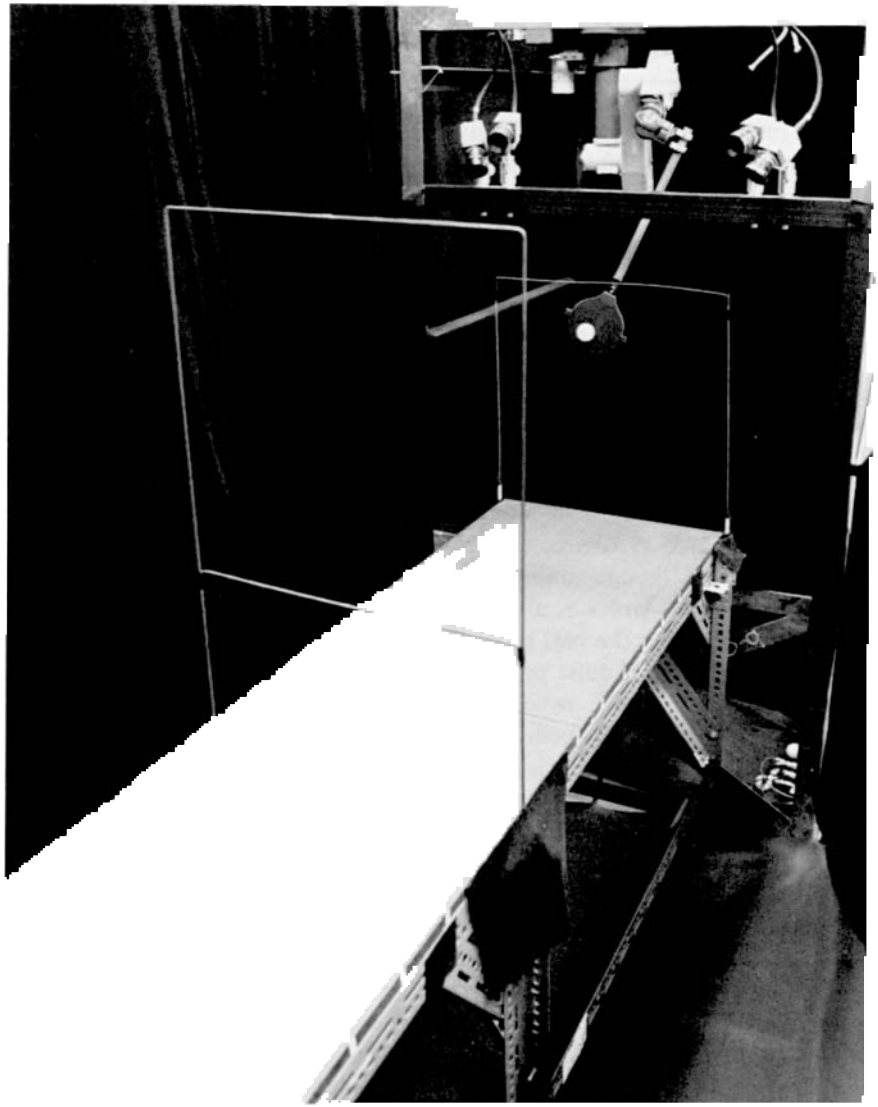

Figure 3. Table and Robot. The frames at the center and end of the table may be seen, with the robot and video cameras at back.

1.3 Why Ping-Pong is a Good Problem

As implied by the title of this book, we use robot ping-pong as an experimental task for investigating more intelligent robot control methods. Ping-pong has many advantages as an experimental task; our reasons for choosing it are both technical and nontechnical.

Starting first with the less technical reasons, we can tell whether or not we are succeeding — it is possible to visually observe improvements in the machine's performance. Observers can readily understand the problem. Ping-pong is also a stable problem: unlike many industrial problems, technological advances won't make the entire problem obsolete overnight.

Ping-pong's breadth means that we can use ping-pong as a strong common focus in describing work across many subfields of robotics, from sensing to processing to actuation to system design. The reaction time allowed to the entire robot system: sensors, processors, and actuators, is quite short. Each system component must be efficacious, fast, and have a low latency. Although many might regard the requirement for high performance in all facets as a liability, we regard it as a strength because it forces us to consider the real problems.

The sensor system must provide estimates of the ball trajectory as early as possible so that the robot may begin moving to approximately the correct place as early as possible. High accuracy must be obtained when the ball is far away, not just close up, so that the best possible spin estimates can be computed when they are most helpful — before the spin has had much of a chance to perturb the trajectory, and maximum reaction time is available. The trajectory analyzer must use a complex flight dynamics model.

The low-level robot controller must be able to operate the actuator as close as possible to its physical and electrical capabilities, unlike conventional controllers, which sacrifice performance for simplicity. The robot must be able to be made to arrive not only at a specific place, but accurately at a specific time and velocity as well.

The robot system must act long before accurate data is available. The initial data is guaranteed to be incorrect because of the long integration time required to compute spin data, and because physically the bounce does not occur until very late, yet it can have major effects on the trajectory. If the robot waits or is indecisive, there will be no time for motion. The robot must be prepared to start moving in the right direction and to continue to update the planned return as more data becomes available.

Robot ping-pong has no best solution: best can only be defined in terms of some arbitrary evaluator. We know only one true evaluator: "Who won the point?" The expert controller must pick a suitable return from those feasible for a given incoming trajectory. Despite the extra degrees of freedom, the planned return must satisfy the constraints of the robot and the rules of ping-pong. As additional data arrives, we must correct beleaguered constraints at the expense of unstressed ones.

It is this complexity that makes robot ping-pong a good experimental task. By focusing on producing a working ping-pong system, we ensure that we find the "real" problems, which aren't always obvious, and that our solutions are feasible, rather than being paper tigers.

1.4 A Preview of the Work

A primary contribution of this book is the expert controller used to do task planning and control. Conventional AI (artificial intelligence) programs manipulate symbols, whereas conventional robot controllers perform numeric calculations: to play ping-pong, the expert controller must fuse the two approaches. For example, symbolic data may represent what data is available, what alternative action has been selected, or might categorize the state of the environment.

To support numeric processing, the entire system is written in "C," but the program is structured to support symbol manipulation as well. To facilitate the combined symbolic and numeric approach, a specialized data structure, the "model," was developed which can simultaneously generate both types of information.

A crucial element of the ping-pong task is that new sensor data arrives all the time, forcing the plan to be continually modified. We will describe a task planning architecture able to cope with, and indeed exploit, the necessity of continuously updating its plan as the input data changes. The plan must satisfy the constraints of both the ping-pong task and the robot manipulator. Initially, the system intelligently "guesses" at a plan, without precise knowledge of these complex, interrelated constraints. Models encode much of the guessing strategy.

As new sensor data arrives, the expert controller modifies the plan not only to account for the changing sensor data, but to avoid problems, such as an impossible kinematic configuration, an impending degeneracy, or an overloaded joint motor. We encapsulate the expertise to modify the plan in modules called "tuners," which evaluate and incrementally improve the quality of the plan.

Of course, when a problem does occur, the system must correct it and continue operating. An exception handling system rapidly selects a low-level agent (a tuner) capable of solving a problem. The mechanism uses a hierarchical delegation of authority so that conditional responses may be made when exceptions occur in the course of correcting a previous exception.

To track the ball, we have created a 60 Hz, low-latency, four-dimensional vision system, the four dimensions being x, y, z, and additionally, t. Moments provide accurate ball centroids in the camera image plane; we describe how we compute them in real time. Our stereo vision algorithms concentrate not on the usual correspondence problem (there is only one object to correspond), but instead on high numeric accuracy: we compensate for the camera placements, lens distortion, and the ball's light intensity distribution. Unlike prior motion interpretation work, a complex model is fitted to the ball's trajectory, and used to predict its future motion.

A robot controller has been developed which provides features necessary for operation in a very dynamic environment. Several types of trajectory generators have been compared; we will justify and describe our trajectory generator. The controller generates smooth motions using quintic polynomials, which have specified initiation and completion times, positions, velocities, and accelerations. The polynomials may be respecified while the arm moves. To facilitate planning, the controller estimates the arm's capabilities based on motor torques, inertia as a function of configuration, and gravity loading. Predictive algorithms for back electro-motive-force (EMF) limiting on quintic trajectories will also be presented. Extensive feed-forward techniques ensure that the arm performs according to the estimates.

A robot consisting of sensors, processing, and actuators distributed across multiple processors must achieve accurate performance in the temporal domain, as well as the physical domain. A clock synchronization system and appropriately designed peripherals make this possible.

A complex system such as the robot ping-pong player can not be turned on suddenly one day to meet its first opponent. Instead, the system was gradually created, debugged, and analyzed to suggest further improvements. Debugging a real-time system is harder than usual, because the debugger must not alter the system's temporal behavior. A fast data logging system and a domain-independent debugger reveal the system's performance to the designer. The same logging system can be used in real time by the robot to analyze its own behavior, without adverse impact on performance, to report faults, and to provide support for learning algorithms.

1.5 Implications and Application Areas

Let us consider the application areas of the techniques we will discuss in this book.

A robot's speed is always critically important, as it determines the robot's effective cost. By making a robot system operate in continuous time, we can almost certainly increase its speed, as the robot's actions can overlap activity in the environment. The precise knowledge of time, and temporal control over motions and sensing, allow the system to be constructed so that everything happens "on schedule," rather than having to leave significant dead times. Similarly, we can plan robot motions to use all available arm capability. By understanding motion in the environment, we may not have to stop objects to operate on them, for example, we can operate on objects as they move on a conveyor belt, saving both the time and machinery necessary to stop the object.

We have developed techniques to control systems with redundant degrees of freedom. One simple example of such a system is an arm with seven or more degrees of freedom, such as the human arm. The extra degrees of freedom can be used to avoid degeneracies [63], or to maneuver past obstacles in the environment [46]. A hand is a particularly extreme example of redundancy, as hands have 12 or more degrees of freedom. Our work might be used to decide how to position objects within the hand, and how to change grasps when manipulating an object. At an intermediate level of complexity, which is perhaps of greatest interest, we can consider the control of an arm mounted on a movable base. Humans reposition their arms by moving both their legs and torso. A robot arm might be mounted on a wheeled mobile robot or a sliding track. In any case, our techniques may be used to allocate desired motions between the sluggish but long-range mobile element, and the faster but limited-reach arm. The same argument may be made for mounting limited-view sensors on a mobile platform.

The two areas described above, namely continuous-time systems and redundant systems, are outside of most current robot application areas. Continuous-time and redundant systems are more complex than current applications, but offer important advantages such as faster operation and increased reach.

There is an additional synergistic advantage to be had by building systems that are both continuous-time and redundant: robustness. When a system runs continuously, and has redundant degrees of freedom to adjust, it can be much more robust than otherwise, because the system may avoid

problems by adjusting the redundant degrees of freedom. The increased robustness alone may well justify the increased complexity of making systems continuous-time and redundant.

1.6 Organization of This Book

This book contains several different classes of details which may appeal to different readers, ranging from ping-pong physics to computer system implementations. We have noted each such especially detailed section, so that readers may readily establish if they wish to skip a section.

Chapter 2 details the characteristics of the robot ping-pong task, especially areas of difficulty, and includes a comparison to the human game. The physics of the sport — aerodynamics and impact dynamics — are discussed in detail.

Chapter 3 outlines the entire system and its decomposition into subtasks. System-wide issues, such as the computing network, task partitioning, clock synchronization, data logging, and debugging are also examined.

Chapter 4 highlights the real-time vision system, which measures and then predicts the ball's position as a function of time. A variety of compensations provide maximal accuracy for objects following complex trajectories.

Chapter 5 illuminates the lower levels of robot control system, focusing primarily on generic task-independent software. The robot controller estimates the manipulator's capabilities, and ensures that the robot acts accordingly.

The expert controller requires two chapters for an adequate description. Chapter 6 sets the stage for the expert controller by presenting two principal components: the program flow architecture, and a data structure for symbolic and numeric data. In Chapter 7, we build upon these components to form modules for initial planning, temporal updating, and exception handling.

Chapter 8 ties together all the previous chapters, detailing the expert controller's utilization in the robot ping-pong system, including case studies of program components and an analysis of program performance.

Finally, the book is summarized in Chapter 9, and some long-term possibilities are explored. Will machines triumph over man?